Probabilistic deep learning: from efficient sampling to principled generation

mgr inż. Marcin Sendera

Jagiellonian University
Doctoral School of Exact and Life Sciences
Faculty of Mathematics and Computer Science

Doctoral dissertation

Under the supervision of prof. dr hab. Jacek Tabor

Kraków, 2025

To my wife, Marysia, for the constant joy you bring into my life — a gift that continues to surprise me. Your unwavering belief in me has been my greatest source of strength, and I am eternally grateful for you.

Acknowledgements

My journey in a complex and challenging field would not have been possible without the intellectual and personal support of the many remarkable individuals and groups. To all of them, I extend my deepest gratitude. I owe you more than I can ever express.

First and foremost, I am profoundly grateful to my advisor, **Professor Jacek Tabor**. Your exceptional guidance, profound insights, and unwavering support have been the cornerstones of this doctoral journey. You taught me not only how to conduct rigorous research but also how to pursue problems with a deep sense of intellectual curiosity. I am truly thankful for the trust you placed in me and for the great freedom you gave me to explore new ideas—even those that sometimes proved to be too ambitious or simply incorrect—which ultimately made this work possible.

This dissertation, which is a series of collaborative publications, is a testament to the remarkable people with whom I had the privilege of working. I am immensely grateful for the opportunity to have worked with **Professor Yoshua Bengio**, and thank him for his trust and support during my research internship at the **Mila – Quebec AI Institute**. I am also deeply thankful to **Dr. Nikolay Malkin**, who provided exceptional guidance and mentorship from both a research and personal perspective during that time. I am truly grateful for everything I learned from you, Kolya. My deepest thanks also go to my collaborators in Poland, especially **Professor Maciej Zięba** and **Dr. Tomasz Kuśmierczyk**, for their invaluable contributions and for the stimulating discussions that pushed this research forward. I am also thankful for the intellectual environment fostered by the **GMUM research group** and the entire team at **Yoshua's lab**, where I spent a transformative year. **My fellow doctoral students and lab mates** deserve a special thank you for their camaraderie, support, and friendship through the highs and lows of research.

I am also grateful to the **many colleagues and co-authors** from all around the world with whom I shared countless hours of brainstorming, problem-solving, and celebrating small victories. The generous grants from the **National Science Centre in Poland** and **Jagiellonian University** were essential in providing the resources necessary for this research.

Finally, and most importantly, I thank my **friends and family** for their endless encouragement, love, and patience. Foremost, I thank my wonderful wife, **Marysia**, who believed in me even when I was losing spirit and hope. Your love and unwavering belief in me were my constant source of strength, providing the foundation of support that allowed me to dedicate myself to this work. To my parents, my brother, and the rest of my family and friends, thank you for influencing my growth and making this possible. This achievement is as much a testament to your support as it is to my own efforts.

Abstract

In recent years, artificial intelligence has undergone rapid and transformative progress, largely driven by the rise of deep learning over the past two decades. Initially focused on simple tasks like image and language processing, AI methods now play a transformative role in a wide range of domains, including protein folding, drug design, and climate change forecasting. These developments are reshaping both modern science and society.

At the core of many of these advances lies probabilistic deep learning — a paradigm that combines the representational power of deep neural networks with principled approaches to uncertainty, inference, and sampling. These models are notable for their ability to capture complex data distributions and often generate new samples that approximate the true distribution. Despite these strengths, probabilistic models face significant challenges, such as scaling to high-dimensional problems. These challenges underscore the need for robust uncertainty estimation and effective sampling methods. Moreover, as large language models begin to act in real-world environments, there is a growing demand for understanding their Bayesian posteriors and the mechanisms underlying their decision-making processes.

This thesis, comprising nine publications, explores the three core pillars of probabilistic deep learning: density estimation, efficient sampling, and Bayesian inference. Accordingly, the thesis is structured into three parts.

The first part (Publications [I] to [III]) focuses on normalising flows as models for density estimation. These models learn diffeomorphic transformations between a simple base distribution (*e.g.*, Gaussian) and complex data distributions, allowing exact likelihood computation via the change-of-variables formula. Publications [I] and [II] address out-of-distribution detection, a critical task for preventing undesirable behavior in deep learning models. Specifically, Publication [I] demonstrates how normalising flows, combined with the Bernstein estimator, can tightly enclose data within a minimal volume bound. Publication [II] employs a constant-volume normalising flow to enhance the classical SVDD method, preventing it from collapsing to a single point. Finally, Publication [III] reveals a novel phenomenon where a Glow normalising flow, trained only on corrupted images (with missing parts), can infer the true data distribution and generate high-quality samples.

The second part investigates the challenge of efficient sampling from the unnormalised target densities, addressed using diffusion models formulated as stochastic differential equations (SDEs) — the current state-of-the-art in generative modelling. These models progressively add Gaussian noise to data during training and reverse this process during inference to generate high-quality samples. However, when the dataset is unavailable, and only the target density (*e.g.*, from energy-based models) is accessible, challenges arise — common in Bayesian deep learning and critical applications like protein folding, molecular design, and astrophysics. Publications [IV] and [V] address these issues by optimizing continuous-time samplers with off-policy variational objectives. Publication [IV] proposes scalable techniques such as replay buffer updated via Langevin MCMC and enhanced with physics-informed Langevin parametrisation, which are important in high dimensions and prevent catastrophic forgetting. Publication [V] establishes formal equivalences between entropic RL methods (GFlowNets) and continuous-time objects (path space measures), and further improves sample efficiency through coarse time discretisation.

The final part adopts a Bayesian perspective, introducing probabilistic reasoning into deep learning via explicit Bayesian inference. This allows learning over distributions of functions, which is especially valuable in meta-learning, where models must adapt rapidly to new tasks from limited data. Publications [VI] to [VIII] address few-shot regression and classification, where models must generalise from few examples. Bayesian methods excel in these scenarios, as they generate distributions of solutions with minimal examples. Publication [VI] enhances the Bayesian deep kernel framework for Gaussian processes by incorporating ODE-based transformations, enabling the modelling of locally non-Gaussian posteriors. Publications [VII] and [VIII] adopt the hypernetwork paradigm, where one network generates weights (or distributions over weights) for another. We demonstrate that even generating a subset of weights (Publication [VII]) or full weight distributions (Publication [VIII]) is sufficient for adapting deep networks to novel tasks. Finally, Publication [IX] addresses the theoretical challenge of transferring informative priors from Gaussian processes into Bayesian neural networks, showing that optimal transport divergences and learned activation functions allow for efficient and principled prior transfer.

The included publications have been presented at leading machine learning conferences and journals, including two A* conferences, three A conferences, one workshop at an A* conference, and a top-tier journal. I am the first or co-first author on all but one of these works. Moreover, I have authored several other publications on deep learning challenges, presented at major venues. These achievements stem from extensive national and international collaborations. Additionally, I served as Principal Investigator for a Preludium grant (National Science Centre, Poland) and a Jagiellonian University Excellence Initiative grant.

During my doctoral studies, I completed an almost year-long research internship at Mila – Quebec AI Institute, working with Prof. Yoshua Bengio and establishing collaborations with leading researchers in the field. Finally, I have contributed to the academic community as a reviewer for top-tier conferences and as a volunteer at deep learning summer schools.

Keywords: probabilistic deep learning, density estimation, efficient sampling, Bayesian inference, generative modelling, Bayesian deep learning, normalising flows, diffusion models

Streszczenie

Na przestrzeni ostatnich lat mogliśmy zaobserwować nagłe przyspieszenie w badaniach nad metodami sztucznej inteligencji (*artificial intelligence*; AI), napędzany, przede wszystkim, poprzez rozwój uczenia głębokiego (*deep learning*), w ostatnich dwóch dekadach. Metody, które jeszcze niedawno były używane do rozwiązywania prostych zadań jak przetwarzanie obrazów i języka, znajdują teraz zastosowanie w najbardziej skomplikowanych wyzwaniach - jak zwijanie białek, projektowanie leków i prognozowanie zmian klimatycznych. Właśnie te osiągnięcia na nowo kształtują zarówno współczesną naukę, jak i społeczeństwo.

U podstaw wielu ze wspomnianych osiągnięć leży probabilistyczne podejście do uczenia głębokiego, czyli paradygmat łączący zdolności głębokich sieci neuronowych do reprezentowania danych ze ścisłym podejściem do przetwarzania niepewności modelu, wnioskowania i próbkowania z nieznanych rozkładów. Modele probabilistyczne wyróżniają się zdolnością do reprezentowania złożonych rozkładów danych i często są w stanie generować nowe próbki w oparciu o dobrą aproksymację prawdziwego rozkładu. Pomimo swoich niewątpliwych możliwości, modele probabilistyczne często napotykają znaczące wyzwania, takie jak skalowanie ich działania do problemów wysokowymiarowych. Pojawiające się wyzwania jedynie jeszcze mocniej podkreślają potrzebę tworzenia efektywnych metod estymacji niepewności i próbkowania z nieznanych rozkładów. Dodatkowo, w miarę jak duże modele językowe (*large language models*; LLMs) zaczynają samodzielnie działać w rzeczywistym świecie, rośnie zapotrzebowanie na zrozumienie ich bayesowskich rozkładów a posteriori oraz mechanizmów odpowiadających za podejmowane decyzje.

Prezentowana rozprawa doktorska jest cyklem dziewięciu publikacji, które rozważają trzy podstawowe problemy probabilistycznego podejścia do uczenia głębokiego, czyli estymację gęstości prawdopodobieństwa rozkładu, efektywne próbkowanie z nieznanych rozkładów oraz wnioskowanie bayesowskie. Z tego powodu, niniejsza praca została także podzielona na trzy części odpowiadające odpowiednim zagadnieniom.

W pierwszej części rozprawy (publikacje od [I] do [III]) rozważane są przepływy normalizujące (*normalising flows*) jako podstawowe modele estymacji gęstości prawdopodobieństwa. Modele te uczą się przekształceń dyfeomorficznych pomiędzy prostym rozkładem bazowym

(np. rozkładem Gaussa) a złożonymi rozkładami danych. Umożliwiają one obliczenie funkcji wiarygodności (*likelihood*) w sposób analityczny poprzez wykorzystanie formuły zamiany zmiennych losowych. W pierwszych dwóch publikacjach (publikacje [I] i [II]) rozważany jest problem wykrycia danych pochodzących spoza rozkładu, będącym krytycznym zadaniem dla zapobiegania niepożądanym zachowaniom w modelach uczenia głębokiego. W szczególności, publikacja [I] demonstruje, jak przepływy normalizujące, w połączeniu z estymatorem Bernsteina, mogą ściśle ograniczyć dane otoczce o minimalnej objętości. Natomiast, publikacja [II] wykorzystuje przepływ normalizujący o stałej objętości do ulepszenia klasycznej metody SVDD, zapobiegając jej zbiegnięciu do pojedynczego punktu. Wreszcie, w publikacji [III] zaprezentowane zostało nowo zaobserwowane zjawisko, gdzie przepływ normalizujący (tutaj model Glow), trenowany jedynie na uszkodzonych obrazach (z brakującymi częściami), może opisać prawdziwy rozkład danych i generować wysokiej jakości próbki.

Druga część rozprawy rozważa problem efektywnego próbkowania z nieznanych i nienormalizowanych funkcji gęstości z wykorzystaniem modeli dyfuzyjnych zadanych przez stochastyczne równania różniczkowe (*stochastic differential equations*; SDEs) — obecnie najlepsze podejście w modelowaniu generatywnym. Takie modele są trenowane, by nauczyć się stopniowego dodawania szumu pochodzącego z rozkładu Gaussa do prawdziwych danych, a w trakcie generowania, odwracają proces treningowy, żeby uzyskać wysokiej jakości próbki. Jednak gdy nie ma żadnego zbioru danych, a dostępna jest jedynie funkcja gęstości (np. zadana przez modele oparte o funkcję energii; *energy-based models*) pojawiają się nowe wyzwania. Niestety, jest to powszechne w bayesowskim uczeniu głębokim i ważnych zastosowaniach takich jak zwijanie białek, projektowanie cząsteczek chemicznych czy problemach astrofizycznych. Publikacje [IV] i [V] rozwiązują te problemy poprzez optymalizację wariacyjnych funkcji celu dla ciągłych samplerów dyfuzyjnych uczonych przy pomocy trajektorii generowanych w inny sposób niż z obecnej polityki (*off-policy*). W publikacji [IV] zaproponowane zostały skalowalne techniki, takie jak, inspirowaną fizyką, parametryzację Langevina oraz bufor danych do powtórzenia (*replay buffer*) aktualizowany poprzez Langevin MCMC. Pokazano, że są one istotne w przypadku problemów wysoko wymiarowych, a także zapobiegają zapominaniu (*catastrophic forgetting*). Z kolei publikacja [V] wprowadza w sposób formalny równoważności między metodami uczenia ze wzmocnieniem (*reinforcement learning*; RL) sterowanego entropią (GFlowNets), a obiektami zdefiniowanymi w czasie ciągłym (tutaj *path space measures*) i dodatkowo poprawia efektywność próbkowania poprzez rzadszą dyskretyzację.

Ostatnia część przyjmuje perspektywę bayesowską, wprowadzającego probabilistyczne rozumowanie do uczenia głębokiego poprzez jawne wnioskowanie bayesowskie. Pozwala

to na operowanie na rozkładach prawdopodobieństwa nad funkcjami, co jest szczególnie wartościowe w meta-uczeniu (*meta-learning*), gdzie konieczna jest szybka adaptacja do nowych zadań na podstawie ograniczonych danych. Publikacje [VI] – [VIII] rozważają regresję oraz klasyfikację *few-shot*, gdzie modele muszą generalizować na podstawie kilku przykładów. Metody bayesowskie doskonale sprawdzają się w tych scenariuszach, ponieważ generują rozkłady rozwiązań przy minimalnej liczbie przykładów. Publikacja [VI] ulepsza procesy gaussowskie wykorzystujące jądra (*kernel functions*) dodając przekształcenia oparte na zwykłych równaniach różniczkowych (*ordinary differential equations*; ODEs). Umożliwia to modelowanie lokalnie niegaussowskich rozkładów a posteriori. Publikacje [VII] i [VIII] operują w paradygmacie hipersieci (*hypernetworks*), gdzie jedna sieć generuje wagi (lub rozkłady nad wagami) dla innej. Pokazują one, że nawet generowanie podzbioru wag (publikacja [VII]) lub pełnych rozkładów nad tymi wagami (publikacja [VIII]) jest wystarczające do adaptacji głębokich sieci do nowych zadań. Wreszcie, Publikacja [IX] rozważa w jaki sposób można wymusić zastosowanie informatywnych rozkładów a priori z procesów gaussowskich do bayesowskich sieci neuronowych. W szczególności, ta publikacja pokazuje, że uczenie funkcji aktywacji i optymalizowanie funkcji kosztu danej przez problem optymalnego transportu, pozwalają na efektywny transfer rozkładów a priori.

Prace zawarte w prezentowanym cyklu publikacji zostały zaprezentowane na wiodących konferencjach i w czasopismach z zakresu uczenia maszynowego, w tym na dwóch konferencjach rangi A*, trzech konferencjach rangi A, jednym warsztacie konferencji rangi A* oraz w czołowym czasopiśmie w tej dziedzinie. We wszystkich, za wyjątkiem jednej z nich, jestem pierwszym lub współ-pierwszym autorem. Jestem także autorem kilku innych publikacji dotyczących zagadnień uczenia głębokiego, prezentowanych na najważniejszych konferencjach. Te osiągnięcia wynikają z rozległej współpracy krajowej i międzynarodowej. Ponadto, pełniłem rolę kierownika projektu w grancie Preludium (Narodowe Centrum Nauki) oraz w grancie w ramach Inicjatywy Doskonałości Uniwersytetu Jagiellońskiego. W czasie studiów doktoranckich odbyłem prawie roczny staż badawczy w Mila — Quebec AI Institute, pracując z prof. Yoshuą Bengio i nawiązując współpracę z czołowymi badaczami w dziedzinie. Wreszcie, pracowałem na rzecz społeczności akademickiej jako recenzent na czołowych konferencjach oraz jako wolontariusz kilku szkół letnich dotyczących uczenia głębokiego.

Słowa kluczowe: probabilistyczne podejście do uczenia głębokiego, estymacja gęstości prawdopodobieństwa, efektywne próbkowanie z nieznanych rozkładów, wnioskowanie bayesowskie, modelowanie generatywne, bayesowskie uczenie głębokie, modele przepływowe, modele dyfuzyjne

List of publications:

[I] Maziarka, Łukasz, Marek Śmieja, **Marcin Sendera**, Łukasz Struski, Jacek Tabor, and Przemysław Spurek. "OneFlow: One-class flow for anomaly detection based on a minimal volume region." *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, no. 11 (2021): 8508-8519., CORE A*, 200 MSHE (*Ministry of Science and Higher Education*) points.

[II] **Sendera, Marcin**, Marek Śmieja, Łukasz Maziarka, Łukasz Struski, Przemysław Spurek, and Jacek Tabor. "Flow-based SVDD for anomaly detection." In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+ 2020)*, 2020.

[III] **Sendera, Marcin**, Łukasz Struski, and Przemysław Spurek. "Missing Glow Phenomenon: Learning Disentangled Representation of Missing Data." In *International Conference on Neural Information Processing*, pp. 196-204. Cham: Springer International Publishing, 2021., CORE A, 140 MSHE points (when submitted and published).

[IV] **Sendera, Marcin**, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. "Improved off-policy training of diffusion samplers." *Advances in Neural Information Processing Systems 37* (2025): 81016-81045., CORE A*, 200 MSHE points.

[V] Berner, Julius*, Lorenz Richter*, **Marcin Sendera***, Jarrid Rector-Brooks, and Nikolay Malkin. "From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training." *arXiv preprint arXiv:2501.06148* (2025). *Under review.*

[VI] **Sendera, Marcin**, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzcinski, Przemysław Spurek, and Maciej Zieba. "Non-Gaussian Gaussian Processes for Few-Shot Regression." *Advances in Neural Information Processing Systems 34* (2021): 10285-10298., CORE A*, 200 MSHE points.

[VII] **Sendera, Marcin***, Marcin Przewięźlikowski*, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. "Hypershot: Few-shot learning by kernel hypernetworks." In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2469-2478. 2023., CORE A, 140 MSHE points.

* *denotes equal contribution*

[VIII] **Sendera, Marcin***, Marcin Przewięźlikowski*, Jan Miksa, Mateusz Rajski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. "The general framework for few-shot learning by kernel HyperNetworks." *Machine Vision and Applications 34*, no. 4 (2023): 53., CORE B, 100 MSHE points.

[IX] **Sendera, Marcin***, Amin Sorkhei, and Tomasz Kuśmierczyk*. "Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations." In *The 41st Conference on Uncertainty in Artificial Intelligence*. 2025., CORE A, 140 MSHE points.

Other works published during the PhD studies, but not included in this dissertation, are briefly presented in Chapter §5.

Table of contents

List of figures

List of figures

List of tables

1 Introduction

The probabilistic imperative in modern AI

As artificial intelligence systems become increasingly integrated into critical aspects of human life, a troubling pattern has emerged that threatens their continued deployment and acceptance. When a medical AI confidently misdiagnoses a rare condition (Abdar et al., 2021; Rajpurkar et al., 2022), or an autonomous vehicle fails to recognise an unprecedented scenario (McAllister et al., 2017; Michelmore et al., 2020), the consequences extend far beyond algorithmic failure — they reveal a fundamental limitation in how we build intelligent systems. Despite unprecedented advances in deep learning, our most powerful AI systems lack a crucial capability that even humans possess naturally: the ability to know what they don't know (Bengio et al., 2025; Gal et al., 2016; Kendall and Gal, 2017; Ovadia et al., 2019).

This limitation has become increasingly critical as AI systems are deployed at unprecedented scale across high-stakes domains. Healthcare diagnostics, autonomous navigation, financial decision-making, and large language models now affect millions of lives daily (Bommasani, 2021; Kaddour et al., 2023). Yet these systems operate with a dangerous blindness to their own uncertainty, making confident predictions even when encountering situations far outside their training experience.

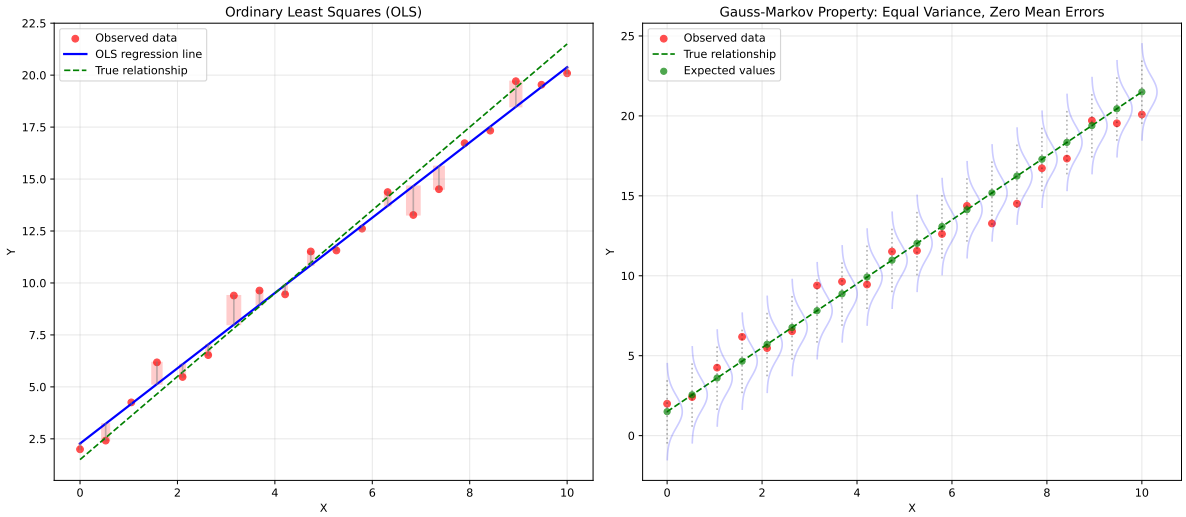


Fig. 1.1 Two complementary perspectives on linear regression — geometric represented with ordinary least squares method (**left plot**) and probabilistic given by the Gauss-Markov theorem (**right plot**). It turns out that the least squares solution is the maximum likelihood estimate under a Gaussian noise model.

Standing on the shoulders of giants: the dual nature of data

While machine learning and artificial intelligence may appear to be recently established fields, their intellectual foundations reach back centuries —to the first principled methods for describing real-world data. The remarkable development we witness today is possible only because we stand on the shoulders of giants who recognised that data and problems could be understood from two fundamentally different yet complementary perspectives: geometric and probabilistic (Hald, 1998; Herr, 1980; Stigler, 1986).

To understand how these perspectives relate and illuminate different aspects of learning, consider one of the first and simplest machine learning models: linear regression. Despite its simplicity, linear regression has enabled profound scientific discoveries and continues to serve as a foundation for modern methods (Bishop, 2006; Hastie et al., 2009). The classical approach solves this problem through the method of least squares, which minimises the distance between a fitted curve and observed data points. This represents a purely geometric perspective — finding the hyperplane that minimises the average distance from observed data. The inventors of least squares, including Newton (who provided partial results in Newton (1711, 1736)) and later Legendre (Legendre, 1806) and Gauss (Gauss, 1809) (whose priority remains disputed, *see* Plackett (1949, 1972); Seal (1967); Stigler (1981)), employed this geometric thinking to describe phenomena ranging from the movement of celestial bodies to terrestrial measurements.

However, the same problem can be approached from an entirely different angle: the probabilistic perspective. Here, we explicitly acknowledge uncertainty in our observations and possibly in our model itself. We seek the unbiased linear estimate of minimum variance, and remarkably, this estimate turns out to be identical to the least squares solution. In modern probabilistic terminology, the least squares solution is the maximum likelihood estimate under a Gaussian noise model — precisely the insight that Gauss developed in his later works through what we now call the Gauss-Markov theorem (Aitken, 1936; Gauss, 1809; Markov, 1900; Rao, 1973) (*see* Figure 1.1).

This raises a natural question: *what additional insights does the probabilistic perspective provide?* The answer, as demonstrated by another ancestor of modern machine learning, Thomas Bayes, is substantial (Bayes, 1763; Laplace, 1812; McGrayne, 2011). Bayes showed how to systematically update our beliefs about specific phenomena in light of new evidence. Moreover, the probabilistic framework allows us to incorporate prior knowledge about problems and update this knowledge with novel data. While Bayes' theorem appears deceptively simple, it is remarkably powerful — and notoriously difficult to apply in practice due to the intractability of the probability density functions involved (Gelman et al., 1995; Robert et al., 1999).

These computational challenges did not discourage the pioneers of computer science and artificial intelligence. Visionaries like Turing, Solomonoff, and Kolmogorov (Kolmogorov, 1965; Solomonoff, 1964a,b; Turing, 1950) quickly recognised that probabilistic reasoning, rather than purely symbolic manipulation, would be essential for creating thinking machines (MacKay, 2003; Murphy, 2012; Pearl, 1988). Their insights laid the groundwork for the probabilistic revolution that would eventually transform artificial intelligence.

The deep learning revolution and its limits

Building upon these foundational insights, and following the AI winter of the 1980s (Goodfellow et al., 2016; Schmidhuber, 2015), the past decade has witnessed an extraordinary renaissance through deep learning. Neural networks have achieved remarkable breakthroughs across computer vision (Krizhevsky et al., 2012; LeCun et al., 2015), natural language processing (Vaswani et al., 2017), and scientific discovery — advances so profound they have garnered Nobel Prize recognition (Baker et al., 2024; Hopfield and Hinton, 2024; Jumper et al., 2021). This success, driven primarily by deterministic optimization of massive overparameterised models, has pushed the boundaries of artificial intelligence and led some to question whether probabilistic approaches remain relevant in an era of brute-force optimization and using larger and larger neural networks (Brown et al., 2020; Hoffmann et al., 2022; Kaplan et al., 2020; Sutton, 2019).

Yet this very success has exposed critical vulnerabilities in purely deterministic paradigms. Modern deep learning systems exhibit four fundamental failure modes that threaten their real-world deployment: **Overconfident predictions** on out-of-distribution data, where systems fail to recognise when they encounter situations beyond their training scope (Hein et al., 2019; Hendrycks and Gimpel, 2016; Liang et al., 2017). **Silent failures** where systems don't know what they don't know, leading to catastrophic errors without warning (Amodei et al., 2016; Nguyen et al., 2015). **Poor calibration** between stated confidence and actual accuracy, rendering their uncertainty estimates unreliable (Guo et al., 2017; Minderer et al., 2021; Nixon et al., 2019). **Inability to incorporate prior knowledge** or domain expertise, forcing systems to learn everything from scratch rather than building on established understanding (Battaglia et al., 2018; Lake et al., 2017; Marcus, 2018).

These limitations are not merely technical inconveniences — they represent existential challenges to AI safety and reliability. As foundation models become increasingly integrated into critical decision-making processes, the absence of principled uncertainty quantification poses unprecedented risks to accuracy, fairness, and accountability (Bhatt et al., 2021; Doshi-Velez and Kim, 2017; Rudin, 2019).

The probabilistic imperative

The growing consensus within the AI community is clear: the current paradigm of point-estimate-based deep learning is fundamentally insufficient for trustworthy AI systems (Fort and Jastrzebski, 2019; Maddox et al., 2019; Wilson and Izmailov, 2020). We require a paradigm shift toward principled uncertainty quantification — one that preserves the remarkable capabilities of deep learning while addressing its fundamental blind spots.

This shift demands advances across three interconnected foundations of probabilistic modelling, each addressing critical gaps between current capabilities and practical needs:
Density estimation forms the mathematical foundation for understanding what constitutes normal versus anomalous behaviour in complex, high-dimensional data. While traditional approaches exist, they struggle with the scale and complexity of modern deep learning applications, necessitating novel solutions capable of handling the distributions encountered in real-world deployments (Kobyzev et al., 2020; Papamakarios et al., 2021; Rezende and Mohamed, 2015).
Efficient sampling enables exploration of complex probability landscapes, proving particularly crucial when we can evaluate objectives or energy functions but lack comprehensive training data—scenarios common in scientific discovery, molecular design, and inverse problems. Current sampling methods often fail to scale to the high-dimensional, multimodal distributions characteristic of modern applications (Ho et al., 2020; Sohl-Dickstein

et al., 2015; Song et al., 2021b).

Bayesian reasoning provides the principled framework for uncertainty quantification and belief updating under the complex conditions of modern AI deployment. Despite decades of research, existing Bayesian approaches face critical scalability limitations with contemporary architectures, requiring fundamental innovations to bridge theory and practice (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017; Wenzel et al., 2020).

These foundations are deeply interconnected: density estimation reveals the structure of uncertainty in data, efficient sampling enables exploration of the complex probability landscapes that density estimation uncovers, and Bayesian reasoning provides the principled framework for making decisions under the uncertainty that both foundations help quantify.

Dissertation contributions and structure

This dissertation is a series of publications that present a comprehensive research program addressing these interconnected challenges. It consists of nine (mostly published) works that collectively advance the state-of-the-art in probabilistic deep learning. Each chapter begins with detailed motivation and technical background, followed by comprehensive presentation of individual contributions. **Chapter §5** presents the complete academic profile, while **Chapter §6** synthesises the broader impact on probabilistic deep learning. Full publications appear in the Appendix.

The contributions span three coherent research directions:

Density estimation with normalising flows (Chapter §2) introduces novel applications that leverage the exact likelihood computation and invertibility properties of normalising flows. This includes pioneering approaches to anomaly detection that can reliably identify out-of-distribution samples (Kirichenko et al., 2020; Nalisnick et al., 2018) (Publications [I] and [II]) and robust methods for modelling missing data scenarios while preserving distributional properties (Dinh et al., 2016; Kingma and Dhariwal, 2018) (Publication [III]).

We tackle the fundamental problem of anomaly detection by finding the minimal volume hypersphere that encloses nominal data (*i.e.*, data from the underlying distribution). However, we approach this challenge from two complementary perspectives, resulting in two novel one-class models that achieve state-of-the-art performance. Unlike traditional approaches that find minimal volume hyperspheres, **Section §2.3** (Publication [I]) proposes OneFlow, a method that *does not* rely on the structure of outlier data. This innovation combines the density estimation power of normalising flows with the pioneering application of the smooth *Bernstein quantile estimator*. This combination ensures that only data close to the decision boundary is affected, creating a tight bounding region around nominal samples.

In **Section §2.4** (Publication [II]), we address how to formulate the classical Support Vector Data Description (SVDD) objective for anomaly detection within the deep learning framework, without collapsing to trivial solutions or imposing restrictive constraints on neural network parameterization. We formulate the SVDD objective in the latent space of a constant-determinant normalising flow model, which prevents hypersphere collapse while maintaining the geometric properties essential for anomaly detection. This work presents FlowSVDD, the first flow-based adaptation of the SVDD objective.

Finally, **Section §2.5** (Publication [III]) explores normalising flows in missing data scenarios, where we discover and investigate a surprising behaviour of general flow models trained on datasets with missing regions — the *Missing Glow Phenomenon*. These models remain capable of generating complete images during inference despite being trained on incomplete data. Beyond providing theoretical explanation for this phenomenon, we propose a novel method for imputing missing regions when using Glow models, demonstrating how normalizing flows can naturally handle incomplete data through their invertible architecture.

Efficient sampling with diffusion models (Chapter §3) develops breakthrough methods for generating samples from complex, often unnormalised probability distributions. This includes practical improvements to off-policy training methods for enhanced stability (Bengio et al., 2021; Malkin et al., 2022) (Publication [IV]) and theoretical contributions that bridge discrete-time Markov decision processes with continuous-time neural SDE formulations (Publication [V]), enabling efficient exploration of complex energy landscapes in data-scarce environments (Anderson, 1982; Song and Ermon, 2019).

Section §3.3 (Publication [IV]) addresses one of the most fundamental challenges in diffusion-style samplers: improving credit assignment for *off-policy diffusion samplers*, specifically continuous generative flow networks (GFlowNets), to enable effective exploration in complex, high-dimensional distributions. We systematically evaluate existing inductive biases and propose a novel, efficient method that incorporates local search mechanisms with a replay buffer strategy. Our approach delivers superior performance while simultaneously reducing computational demands, enabling more robust and effective training of continuous GFlowNets for challenging sampling tasks.

Section §3.4 (Publication [V]) focuses on establishing the theoretical connections between training objectives of discrete-time policies (represented by continuous generative flow networks) and continuous-time processes (neural SDEs). While intuitive connections existed within the sampling community, no rigorous theoretical framework had established their asymptotic relationships. We present the first theoretical findings on the asymptotic

behaviour of discrete-time policy objectives as they approach continuous-time processes, providing crucial theoretical foundations for the field. Additionally, we conduct comprehensive empirical studies of different discretisation schemes for training discrete-time policies, demonstrating the superior efficiency of *nonuniform* discretisation schemes compared to typical *uniform* approaches.

Bayesian deep learning for modern applications (Chapter §4) directly tackles the scalability challenges that have limited Bayesian inference in contemporary deep learning. Novel contributions include hybrid architectures combining Gaussian processes with neural networks for improved uncertainty quantification (Liu et al., 2020; Rasmussen and Williams, 2006) (Publication [VI]), meta-learning approaches for few-shot learning scenarios (Finn et al., 2017; Snell et al., 2017) (Publications [VII] and [VIII]), and principled methods for incorporating function-space priors into Bayesian neural networks (Blundell et al., 2015; Louizos and Welling, 2017; Neal, 1996) (Publication [IX]).

This dissertation addresses the application of Bayesian inference in contemporary problems. One of the most suitable applications for this approach is meta-learning, particularly few-shot learning scenarios where uncertainty quantification is crucial for reliable performance. **Section §4.3** (Publication [VI]) proposes a novel probabilistic framework for regression tasks in few-shot regimes. While we employ Gaussian processes (GPs) as our foundational Bayesian framework, we demonstrate how to enable them to effectively model non-Gaussian behaviours. The key innovation lies in integrating ODE-based invertible transformations into the GP framework, enabling the modeling of complex distributions through *locally non-Gaussian posteriors*. This approach, termed *non-Gaussian Gaussian processes* (NGGPs), achieves superior performance in both predictive accuracy and probabilistic calibration due to its enhanced flexibility.

The next **Section §4.4** (Publication [VII] and Publication [VIII]) extends Bayesian approaches to few-shot classification problems. Publication [VII] introduces HyperShot, a novel approach based on the *hypernetwork paradigm*. The method considers relationships between embeddings from support samples through kernel transformations, with the target classifier conditioned on this information and sampled via hypernetworks. Publication [VIII] presents a subsequent extension that incorporates Bayesian reasoning into this framework. By modeling the target classifier as a *Bayesian neural network* (BNN) and training it through *variational inference*, BayesHyperShot introduces principled uncertainty quantification — particularly valuable in settings involving distributional shifts. Both approaches effectively generalise across tasks, achieving performance comparable or superior to non-probabilistic baselines while providing crucial uncertainty estimates.

Finally, **Section §4.5** (Publication [IX]) addresses the fundamental challenge of *transferring GP function-space priors into BNNs*, framed as an optimal transport problem. Unlike existing approaches that require computationally expensive two-level differentiation schemes, we propose using the closed-form solution of the *2-Wasserstein divergence* to ensure efficiency. We discover that joint adaptation of BNN activation functions and priors (over weights and biases) significantly improves the optimization process and prevents mode collapse. Our extensive experiments demonstrate the effectiveness of this approach, and we believe this method will enable the integration of Bayesian principles into standard deep learning architectures — for example, by incorporating BNN layers that mimic expected GP-like behaviour into conventional deep neural networks.

Towards trustworthy AI

Through this integrated approach, this work demonstrates how collective advances in density estimation, efficient sampling, and Bayesian inference can fundamentally address the uncertainty quantification challenge in modern AI systems. The implications extend far beyond theoretical contributions, promising to transform critical applications: healthcare AI that knows when to defer to human experts (Ahn et al., 2022; Raghu et al., 2019), autonomous systems that navigate safely through uncertainty (Filos et al., 2020; Kahn et al., 2017), and scientific discovery platforms that explore complex hypothesis spaces with principled confidence (Jumper et al., 2021; Rives et al., 2021).

This dissertation ultimately paves the way for the next generation of AI systems — ones that combine the remarkable capabilities of deep learning with the principled uncertainty reasoning essential for trustworthy deployment in our uncertain world (Bengio et al., 2025, 2024; Dalrymple et al., 2024). As AI systems become increasingly powerful and ubiquitous, the question is not whether we need probabilistic approaches, but how quickly we can develop and deploy them at scale.

2

The geometry of data: on density estimation and its applications

2.1 Research scope

On density estimation in probabilistic modelling

Let us begin our exploration of probabilistic deep learning by diving into one of the most fundamental challenges in the field: estimating the *probability density function* (PDF) over objects, denoted as $p(\cdot)$. We argue that this problem is not only central to probabilistic modelling but also crucial for the development of modern AI systems (Tomczak, 2024). To make reliable decisions, AI models must first understand their environment and be able to quantify their beliefs and uncertainties in the language of probability (Bishop, 2013; Ghahramani, 2015).

Beyond its theoretical importance, estimating $p(\cdot)$ has numerous practical applications. It enables us to determine whether a given object has been observed before, assess the confidence of an AI system's predictions, and finally generate new samples from the learned distribution (Tomczak, 2024). This last aspect gives rise to generative modelling, which is often understood within deep learning from the data generation perspective. However, we argue that probability density estimation — learning $p(\cdot)$ itself — has much broader

implications beyond mere data generation, making it a fundamental problem in deep learning. Consequently, in this dissertation, even generative models will be seen primarily from a density estimation perspective.

Among various approaches to estimating probability densities, normalising flows stand out due to their unique properties: exact probability density computation, efficient inference, and invertibility (Papamakarios et al., 2021; Rezende and Mohamed, 2015). The probability density transformation follows directly from the *change-of-variables formula* (Bishop, 2006; Murphy, 2012), which allows normalising flows to compute exact likelihoods, a key advantage over models like variational autoencoders (VAEs; Kingma and Welling, 2014), where we only have access to an approximate lower bound on the likelihood (ELBO; Tomczak, 2024).

On normalising flows

Normalising flows can be defined as a series of K invertible functions f_i, parameterised as neural networks, that transform a complex and unknown data distribution into a simple one — typically a standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ — where probability densities can be computed in closed form. When stacked together, these functions form an overall invertible transformation $f = f_K \circ \cdots \circ f_1$, known as a **normalising flow** or **flow-based model** (Papamakarios et al., 2021; Rezende and Mohamed, 2015). The probability density function transformation follows directly from the *change-of-variables formula*, and the model is trained by maximising the log-likelihood of the data under this transformation.

Normalising flows primarily serve as *density estimators*, capturing the stochastic dependencies between continuous random variables. While extensive research on normalising flows for discrete data exists, the publications that form this dissertation focus on their application in the continuous random variables regime.

From a deep learning perspective, designing an effective normalising flow requires constructing an invertible neural network where the logarithm of the determinant of its Jacobian can be computed efficiently (Behrmann et al., 2019; Gresele et al., 2020; Hasenclever et al., 2017; Papamakarios et al., 2021). This design enables exact and efficient inference, making normalising flows particularly attractive for probabilistic modelling.

The formulation discussed so far describes **discrete-time normalising flows**. However, normalising flows can also be defined in a continuous-time regime, where they are parameterised as *ordinary differential equations* (ODEs; Chen et al., 2018; Grathwohl et al., 2019). In this view, a discrete-time normalising flow can be seen as an ODE whose integral is computed on a fixed grid of $K+1$ equidistant points (assuming K transformations creating a flow-based model).

Role of normalising flows in the publications that form this dissertation

Having established that normalising flows are powerful tools for probability density estimation, it is essential to highlight their wide-ranging applications. Beyond density estimation, they have been used in Bayesian deep learning (Louizos and Welling, 2017), variational inference (Rezende and Mohamed, 2015), and generative deep learning (Kingma and Dhariwal, 2018).

This chapter presents a series of publications that form part of this dissertation. These works leverage normalising flows for probabilistic modelling problems, focusing on their efficiency in density estimation.

In particular, two of these works — Publication [I] and Publication [II] (discussed in Sec. §2.3 and Sec. §2.4, respectively) — address anomaly detection, leveraging the bijectivity, volume preservation, and density estimation properties of normalising flows. Meanwhile, Publication [III] (covered in Sec. §2.5) explores a newly observed phenomenon in normalising flow models trained on missing data. We demonstrate that under specific conditions, such models can effectively sample from the density of the original dataset, even without explicit access to the missing regions.

With this high-level foundation in place, we now introduce the mathematical preliminaries of normalising flows before proceeding to a detailed discussion of the publications in Sec. §2.2, starting with their theoretical underpinnings and moving toward their applications in probabilistic modelling. This section serves as a self-contained reference and may be skipped by readers already familiar with the area.

2.2 Background

Having outlined the rationale behind normalising flows from a high-level perspective, we now proceed to a more technical treatment. In this section, we present the detailed construction of normalising flows in both the discrete-time and continuous-time regimes. As the publications comprising this dissertation operate exclusively on continuous random variables, our discussion will focus on normalising flows defined for continuous domains. For a comprehensive overview of flows in the discrete setting, we refer the reader to Kobyzev et al. (2020) and Papamakarios et al. (2021).

From a general mathematical point of view, a normalising flow can be understood as a *diffeomorphism* — an invertible mapping between differentiable submanifolds that is continuous and differentiable in both directions. In practice, however, it is sufficient to consider flows as smooth, bijective functions operating in Euclidean space (typically $\mathbb{R}^N$). This implies *continuity* (ensuring the density can be evaluated at any point), *differentiability* (permitting gradient-based optimisation), and *invertibility* (enabling bidirectional mapping between spaces).

In what follows, we build upon the standard formulation of discrete-time normalising flows, which was briefly introduced in Section §2.1.

Normalising flows

The core idea behind normalising flows is to use the *change-of-variables* formula to model a complex, high-dimensional probability distribution p of a random variable $\mathbf{x} \in \mathbb{R}^N$, starting from a simple *base distribution* π, which has a known *probability density function* $\pi(\cdot)$ (typically, $\mathcal{N}(\cdot \mid \mathbf{0}, \mathbf{I})$). To this end, one expresses $\mathbf{x}$ as a transformation f of a latent variable $\mathbf{z}_0$ sampled from $\pi(\cdot)$:

$$\mathbf{x} = f(\mathbf{z}_0), \qquad \text{where } \mathbf{z}_0 \sim \pi(\cdot). \tag{2.1}$$

The defining property of a *normalising flow* is that the transformation f must be *invertible*, and both f and its inverse f^{-1} must be *differentiable*. Such transformations are known as *diffeomorphisms*, and require that $\mathbf{z}_0$ and $\mathbf{x}$ lie in spaces of the same dimensionality, *e.g.*, $\mathbf{z}_0 \in \mathbb{R}^N$ (Milnor and Weaver, 1997).

Under these conditions, the density $p(\mathbf{x})$ is well-defined and can be computed via the *change-of-variables formula* (Bogachev and Ruas, 2007; Rudin, 1987):

$$p(\mathbf{x}) = \pi(\mathbf{z}_0) \left| \det \mathbf{J}_f(\mathbf{z}_0) \right|^{-1}, \tag{2.2}$$

where $\mathbf{z}_0 = f^{-1}(\mathbf{x})$ and $\mathbf{J}_f$ is the *Jacobian* of f.

In practice, the term *normalising flow* typically refers to a class of bijective transformations for which both the inverse mapping and the determinant of the Jacobian are computationally tractable (Kobyzev et al., 2020).
A common design strategy involves composing a sequence (*chain*) of K invertible transformations, $f_k : \mathbb{R}^N \to \mathbb{R}^N$. Composing them yields the overall mapping:

$$F = f_K \circ \cdots \circ f_1, \tag{2.3}$$

where each f_k maps $\mathbf{z}_{k-1}$ to $\mathbf{z}_k$, with $\mathbf{z}_0 = \mathbf{z}$ and $\mathbf{z}_K = \mathbf{x}$ (*see* Fig. 2.1). Since each f_k is invertible, the entire mapping F is invertible as well.

This explains the origin of the term *normalising flows*. The word *flow* refers to the trajectory that a sample $\mathbf{z}_0 \sim \pi(\cdot)$ follows as it is gradually transformed through the chain of mappings into a sample from the target distribution p. Conversely, the term *normalising* highlights the role of the inverse transformation, which maps data samples back to the base distribution, thereby *normalising* complex data into a simpler one — typically, the multivariate normal distribution.

Overall, the full transformation defined by a sequence of K invertible mappings can be written as:

$$p(\mathbf{x}) = \pi(\mathbf{z}_0 = F^{-1}(\mathbf{x})) \prod_{k=1}^{K} \left|\det \mathbf{J}_{f_k}(\mathbf{z}_{k-1})\right|^{-1}, \qquad \text{where } \mathbf{J}_{f_k} \text{ is the } \textit{Jacobian} \text{ of } f_k. \tag{2.4}$$

This follows from the composability of the transformations f_k and the chain rule for Jacobians applied to composed mappings.

From a deep learning perspective, a key advantage of normalising flows is that the individual transformations f_k can be parameterised by neural networks. These models can be trained efficiently by minimising the negative log-likelihood (NLL; $-\ln p(\mathbf{x})$), which is tractable under the *change-of-variables formula*. Assuming that the base density $\pi(\mathbf{z}_0)$ has a known closed-form expression, we obtain the following training objective:

$$\mathcal{L}_{\text{NLL}} = -\ln p(\mathbf{x}) = -\left(\ln \pi(\mathbf{z}_0 = F^{-1}(\mathbf{x})) - \sum_{k=1}^{K} \ln \left|\det \mathbf{J}_{f_k}(\mathbf{z}_{k-1})\right|\right). \tag{2.5}$$

As a concrete example, if we choose the base distribution to be a standard multivariate Gaussian, *i.e.*, $\pi(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0 \mid \mathbf{0}, \mathbf{I})$, then the objective simplifies to:

$$-\ln p(\mathbf{x}) = -\left(\ln \mathcal{N}(\mathbf{z}_0 = F^{-1}(\mathbf{x}) \mid 0, I) - \sum_{k=1}^{K} \ln \left|\det \mathbf{J}_{f_k}(\mathbf{z}_{k-1})\right|\right).$$

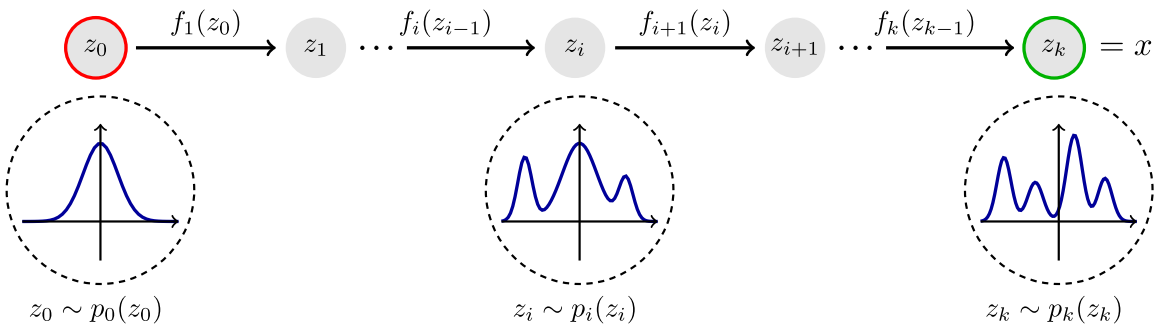


Fig. 2.1 **Normalising flows:** general scheme showing the transformation of a simple base distribution (here $\pi_0 = p_0(\cdot)$) into a complex target distribution p (here $p = p_k$) via a sequence of invertible mappings f_i. *Image source:* https://github.com/janosh/awesome-normalizing-flows

Discrete-time normalising flows

Designing discrete-time normalising flows in the deep learning setting presents several challenges. First, all transformations must be invertible, which requires the use of invertible neural networks. However, an even more pressing difficulty is the computation of the Jacobian log-determinant term, $\sum_{k=1}^{K} \ln \left| \det \mathbf{J}_{f_k}(\mathbf{z}_{k-1}) \right|$, which is generally computationally intractable for arbitrary transformations. Therefore, practical implementations focus on architectures where both invertibility and efficient Jacobian determinant computation are tractable.

Over the years, a variety of such architectures have been proposed. These include, for example, Planar Normalising Flows (Rezende and Mohamed, 2015), Sylvester Normalising Flows (van den Berg et al., 2018), and Residual Flows (Behrmann et al., 2019; Chen et al., 2019a). Each of these aims to balance expressivity with computational efficiency.

Among the most influential architectures is the Real-valued Non-Volume Preserving flow (RealNVP; Dinh et al., 2016). RealNVP significantly extends the earlier NICE (Dinh et al., 2014) architecture and is itself a precursor to Glow (Kingma and Dhariwal, 2018). In fact, many discrete-time normalising flows follow the general architectural principles introduced by NICE and refined in RealNVP.

RealNVP is primarily composed of two types of layers: *coupling layers* and *permutation layers*. The key idea behind a coupling layer is to partition the input vector $\mathbf{x}$ into two parts, $\mathbf{x} = [\mathbf{x}_a, \mathbf{x}_b]$, and process them asymmetrically: one part ($\mathbf{x}_a$) remains unchanged, while the other ($\mathbf{x}_b$) is transformed through an invertible function conditioned on $\mathbf{x}_a$. This structure ensures that the overall transformation is both invertible and has a tractable Jacobian determinant. The partitioning may be as simple as a channel-wise split (*e.g.*, halving) or follow a more complex scheme, such as a *checkerboard* or *channel-wise masking* pattern. A

typical coupling transformation is defined as:

$$\begin{aligned} \mathbf{y}_a &= \mathbf{x}_a, \\ \mathbf{y}_b &= \exp\left(s(\mathbf{x}_a)\right) \odot \mathbf{x}_b + t(\mathbf{x}_a), \end{aligned} \tag{2.6}$$

where s and t are arbitrary neural networks responsible for *scaling* and *translation*, respectively. This transformation is easily invertible, and its Jacobian determinant is given in closed form as:

$$|\det \mathbf{J}| = \prod_{i=1}^{D-d} \exp(s_i(\mathbf{x}_a)) = \exp\left(\sum_{i=1}^{D-d} s_i(\mathbf{x}_a)\right),$$

assuming the division of $\mathbf{x}$ into $[\mathbf{x}_{1:d}, \mathbf{x}_{d+1:D}]$. Thus, the log-determinant is straightforward to compute.

Permutation layers simply reorder the components of the input vector. As these operations are volume-preserving (*i.e.*, have a Jacobian determinant of 1), they can be freely applied between coupling layers to facilitate information mixing. A common choice is to reverse the order of variables, although more expressive alternatives, such as invertible 1×1 convolutions, have also been proposed (*e.g.,* Kingma and Dhariwal (2018)).

Research into improving discrete-time normalising flows has extended beyond the development of new invertible layers. Due to the wide adoption of normalising flows, many techniques have been introduced to increase their flexibility and training stability, such as *e.g.*, factoring out part of variables (Dinh et al., 2016), *rezero trick* (Bachlechner et al., 2021), masking (Dinh et al., 2016), and squeezing (Dinh et al., 2016). However, the most significant advances include the use of learnable base distributions and replacing a fixed permutation by invertible 1×1 convolution (Kingma and Dhariwal, 2018).

Normalising flows remain an active area of research, particularly in the context of density estimation and generative modelling. They have been successfully applied to a wide range of tasks, including, variational inference (Hoogeboom et al., 2021; Rezende and Mohamed, 2015; Tomczak and Welling, 2016, 2017), compression (Ho et al., 2019), density estimation on manifolds (Brehmer and Cranmer, 2020), and modelling discrete probability distributions (Hoogeboom et al., 2019; Tomczak, 2021; van den Berg et al., 2021).

Continuous-time normalising flows

In the previous part, we introduced discrete-time normalising flows, which are constructed as a series of parametrised invertible transformations. This structure implies an exact and fixed discretisation scheme of the joint transformation.

An alternative strategy is to construct normalising flows in the *continuous-time* regime. This is achieved by parameterising the infinitesimal dynamics of the flow and then re-

covering the transformation through integration. In other words, the normalising flow's transformation is defined via an Ordinary Differential Equation (ODE), which describes the flow's evolution over time.

Let us denote the flow's state at time t by $\mathbf{z}_t$. We assume the evolution runs continuously from t_0 to t_1 (*i.e.*, over an interval $t \in [t_0, t_1]$), where $\mathbf{z}_{t_0} = \mathbf{u}$ and $\mathbf{z}_{t_1} = \mathbf{x}$. We define the normalising flow as a parametrised function g_ϕ representing the time derivative of $\mathbf{z}_t$. This yields the following ODE:

$$\frac{d\mathbf{z}_t}{dt} = g_\phi(t, \mathbf{z}_t). \tag{2.7}$$

This means that the function g_ϕ takes the current time t and state $\mathbf{z}_t$ as input and returns the time derivative of $\mathbf{z}_t$ in timepstep t (*i.e.*, $\frac{d\mathbf{z}_t}{dt}$). The only requirements on g_ϕ are that it must be uniformly Lipschitz continuous in $\mathbf{z}_t$ and continuous in t (Chen et al., 2018). From *Picard's existence theorem*, we know that under these conditions, the considered ODE has a unique solution (Coddington and Levinson, 1955). Fortunately, many neural network architectures satisfy these conditions, which means we do not need to explicitly enforce invertibility or tractability of their Jacobian determinant.

This formulation enables flexible mappings between a simple base distribution and an unknown data distribution by integrating the dynamics either forward or backward in time, as presented in the Fig. 2.2. In particular, to compute the forward transformation $\mathbf{x} = T(\mathbf{u})$, we integrate the dynamics from t_0 to t_1:

$$\mathbf{x} = \mathbf{z}_{t_1} = \mathbf{u} + \int_{t=t_0}^{t_1} g_\phi(t, \mathbf{z}_t)\, \mathrm{d}t. \tag{2.8}$$

The inverse transformation T^{-1} is obtained by integrating backward in time:

$$\mathbf{u} = \mathbf{z}_{t_0} = \mathbf{x} + \int_{t=t_1}^{t_0} g_\phi(t, \mathbf{z}_t)\, \mathrm{d}t = \mathbf{x} - \int_{t=t_0}^{t_1} g_\phi(t, \mathbf{z}_t)\, \mathrm{d}t, \tag{2.9}$$

where, for convenience, we used the final expression, which shows that continuous-time normalising flows have equal computational complexity for transformations in both directions.

Then, to compute the change in log-density, we use the following differential equation:

$$\frac{d \log p(\mathbf{z}_t)}{dt} = -\mathrm{Tr}\left\{\mathbf{J}_{g_\phi(t,\cdot)}(\mathbf{z}_t)\right\}, \tag{2.10}$$

where Tr denotes the *trace operator* and $\mathbf{J}_{g_\phi(t,\cdot)}(\mathbf{z}_t)$ is the *Jacobian* of g_ϕ evaluated at $\mathbf{z}_t$. This result follows from the *Fokker–Planck equation* in the special case of *zero diffusion* (Risken,

1996). Thus, integrating both sides gives the log-density of $\mathbf{x}$:

$$\log p_x(\mathbf{x}) = \log p_u(\mathbf{u}) - \int_{t=t_0}^{t_1} \operatorname{Tr}\left\{\mathbf{J}_{g_\phi(t,\cdot)}(\mathbf{z}_t)\right\} \mathrm{d}t. \tag{2.11}$$

While continuous-time normalising flows offer conceptual simplicity and allow great flexibility of neural network parametrisation, they come with computational challenges that required careful treatment.

First, computing the trace of the Jacobian is computationally extremely expensive. To mitigate this, Grathwohl et al. (2019) proposed using the *Hutchinson trace estimator* as an approximation. Moreover, Chen and Duvenaud (2019) proposed a solution, where for a restricted and carefully designed neural network architecture, the Jacobian trace can be computed with a single backpropagation pass.

Second, solving the ODE generally requires numerical integration, as closed-form solutions for complex integrals are rare. A broad class of methods, such as the *Runge–Kutta* family, including *Euler's method*, can be used, although these rely on fixed-step discretisation. A more elegant and often more efficient approach is the *adjoint sensitivity method* (Pontryagin, 1962), which was popularised in this context by Chen et al. (2018).

Having introduced the key concepts behind both discrete- and continuous-time normalising flows, we are now ready to describe the original publications creating this Ph.D. dissertation. In this chapter, we will focus on the works employing discrete-time normalising flows, while in one of the later chapters (*see* Sec. §4.3), we will present works exploring the use of continuous-time formulation.

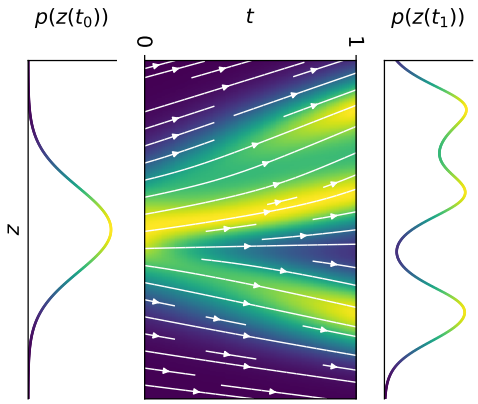


Fig. 2.2 Transformation dynamics in continuous-time normalising flows. This ODE transforms $\mathbf{z}$ sampled from the simple Gaussian distribution (**left**) into complex data distribution (**right**).*Image source:* Grathwohl et al. (2019).

2.3 Oneflow: one-class flow for anomaly detection based on a minimal volume region [I]

In the following section, we focus on Publication [I], in which we propose a novel anomaly detection method based on the density estimation properties of normalising flows and the *quantile Bernstein estimator* (Bernstein, 1912; Cheng, 1995; Lorentz, 1953). The proposed model, called **OneFlow**, constructs a minimal-volume bounding region that encloses nominal data without relying heavily on the structure of outlier data.

By leveraging this combination of techniques, OneFlow effectively creates a tight bounding region for nominal data and, as we show empirically, consistently achieves state-of-the-art or competitive performance across a wide range of anomaly detection benchmarks.

2.3.1 Motivation

Anomaly detection involves identifying abnormal data points within a large set of normal instances and recognising unusual system behaviours (Miljković, 2010). It is a fundamental problem in deep learning applications, with real-world relevance in areas such as fraud detection (Phua et al., 2010) and the identification of adversarial examples (Roth et al., 2019).

Unlike standard binary classification, anomaly detection is significantly more challenging because anomalies do not necessarily follow any specific probability distribution. Instead, they are simply deviations from expected behaviour. Moreover, entirely novel types of anomalies may emerge at test time, further increasing the complexity of the problem.

Due to these challenges, anomaly detection is usually approached from an unsupervised learning perspective. In particular, the typical one-class classifiers focus on characterising only the distribution of nominal data (Chen et al., 2013; Schölkopf et al., 2001; Wang et al., 2019c). Consequently, any instance that significantly deviates from this learned behaviour is classified as an anomaly.

2.3.2 Content

In this work, we propose **OneFlow**, a novel anomaly detection method that functions as a deep one-class classifier based on normalising flow models. While normalising flows are typically used for density estimation in the context of anomaly detection, OneFlow is designed to identify a minimal-volume bounding region that encloses a specified proportion of the data. This approach reduces dependency on the exact structure of outliers, making the method more robust (Maziarka et al., 2021).

Unlike previous approaches that use minimal-volume sets for anomaly detection (Scott and Nowak, 2006; Zhao and Saligrama, 2009), OneFlow offers significantly better scalability

to larger datasets. Additionally, it enables direct optimisation of the minimal-volume set problem, whereas kernel-based methods (*e.g.*, Support Vector Data Description; SVDD; Tax and Duin, 2004) require a problem reformulation to obtain a convex objective.

OneFlow constructs a minimal-volume region by integrating two key components: flow-based models (Dinh et al., 2014, 2016; Rezende and Mohamed, 2015) and the *quantile Bernstein estimator* (Cheng, 1995). This combination allows for gradient propagation primarily through data points near the decision boundary, enhancing learning efficiency and precision.

Problem statement. The majority of standard anomaly detection methods are based on either one-class learning (*e.g.*, One-Class SVM; OCSVM; Schölkopf et al., 2001) or minimal-volume region formulations (*e.g.*, Support Vector Data Description; SVDD; Tax and Duin, 2004). With the growing adoption of deep learning in this domain, many approaches extend these formulations by adapting the SVDD loss to deep learning models (Kim et al., 2015; Ruff et al., 2018). However, this adaptation can lead to **hypersphere collapse**, necessitating the development of various solutions. In this work, we also operate within the framework of enclosing data in a minimal-volume region.

To establish a foundation for our approach, we first formally define the anomaly detection problem as finding a bounding region with minimal volume that contains a specified proportion of the data.
Consider a density function g in $\mathbb{R}^D$ and a subset of data sample $X \subset \mathbb{R}^D$ generated by g. Additionally, let $\alpha \in (0,1)$ be a p-value hyperparameter that determines the proportion of possible outliers, false positives, or abnormal data during training. We define a set $U \subset \mathbb{R}^D$ as a $(1-\alpha)$-**bounding region** of g if a data point drawn from g belongs to U with probability $1-\alpha$, *i.e.,*

$$\int_U g(\mathbf{x})\, d\mathbf{x} = 1-\alpha.$$

Consequently, the $(1-\alpha)$-bounding region covers approximately $(1-\alpha)$ of the data. Formally the problem can be stated as follows:

Problem 1 (Finding the minimal-volume region)**.** *Find a* $(1-\alpha)$*-bounding region* $U \subset \mathbb{R}^D$ *with minimal volume for a density* g *generating data, i.e.,*

$$\arg\min_U \{\mathrm{vol}(U) \mid U : \int_U g(\mathbf{x})\, d\mathbf{x} = 1-\alpha\}.$$

One-class flow based on minimal volume region — OneFlow. While minimal volume sets have been studied in Scott and Nowak (2006); Zhao and Saligrama (2009), these works focus

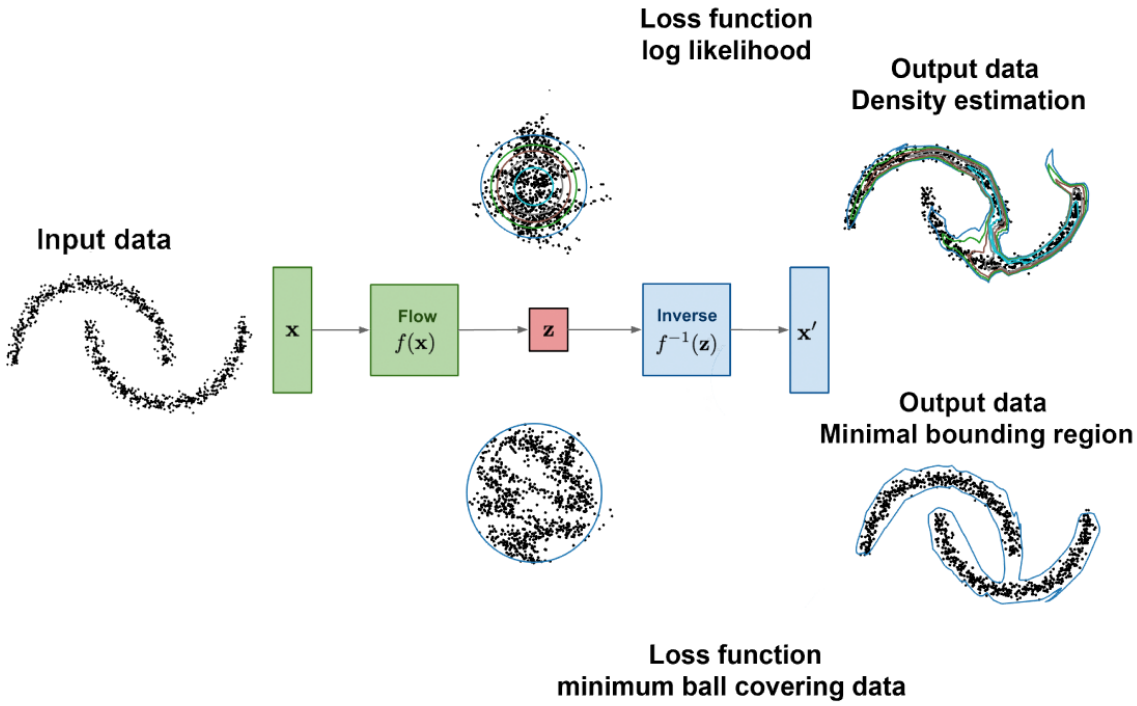


Fig. 2.3 OneFlow identifies a minimal-volume bounding region that encloses a given proportion of data, using a hypersphere in the latent space of the flow model.

on theoretical aspects and propose algorithms that do not scale well. Kernel methods like SVDD (Tax and Duin, 2004) reformulate the problem to enforce convexity but, in doing so, solve a related problem: instead of finding the minimal-volume hypersphere that encloses a given proportion of the data, SVDD penalises points outside the hypersphere.

We introduce OneFlow, a method that directly optimises the criterion in Problem 1, which propagates the objective function gradient only through points near the decision boundary. OneFlow achieves this by integrating two key components: a *normalising flow-based parametrisation* (Dinh et al., 2014, 2016) and the *quantile Bernstein estimator* (Cheng, 1995), which enables the estimation of the optimal size of the minimal-volume region during the optimisation process. Figure 2.3 illustrates the high-level intuition behind the proposed method.

First, we employ a normalising flow parametrisation for efficient density estimation (*i.e.*, the log-likelihood function) and, consequently, volume estimation, while also enabling straightforward sample generation. In our approach, we operate on one of the simplest definitions of flow-based models:

2.3 Oneflow: one-class flow for anomaly detection based on a minimal volume region [I]

Definition 2.3.1 (Flow-based model)**.** *Consider a neural network f such that $f : \mathbb{R}^D \to \mathbb{R}^D$. We define f as **flow-based model** if the inverse mapping f^{-1} is explicitly available and the determinant of its Jacobian (*$\det df(\mathbf{x})$*) can be efficiently computed.*

For the purpose of this work, we focus on discrete-time normalising flow models, such as NICE (Dinh et al., 2014) and RealNVP (Dinh et al., 2016), which can be categorised into two classes: *constant-determinant flows* and *general flows*. The distinction is based on whether $\det(df(\mathbf{x}))$ remains constant for all $\mathbf{x}$ or varies across different points.
We are particularly interested in finding the volume of U given by:

$$\text{vol}(U) = \text{vol}(f^{-1}(B(0,r))).$$

For the matter of simplicity in the final expressions, we define the function:

$$w(\mathbf{x}) = \det d(f^{-1})(\mathbf{x}) = \frac{1}{\det df(f^{-1}(\mathbf{x}))},$$

which remains constant ($w(\mathbf{x}) = w$) if the determinant of Jacobian of the flow transformation is constant. In this paper, we formally demonstrate that for **constant-determinant flows**, the volume of U has a closed-form expression (*see* Observation 1 in Publication [I]):

$$\text{vol}(U) = \text{vol}(B(0,1)) \cdot w \cdot r^D. \tag{2.12}$$

Conversely, for **general flows**, the volume of U can only be approximated (*see* Eq. (2) in Publication [I]):

$$\begin{aligned}\text{vol}(U) = \int_{B(0,r)} w(x)\,dx &\approx \text{vol}(B(0,r)) \cdot \frac{1}{m} \sum_{k=1}^{m} w(r \cdot e_k) \\ &= \frac{\text{vol}(B(0,1))}{m} \cdot r^D \cdot \sum_{k=1}^{m} w(r \cdot e_k),\end{aligned} \tag{2.13}$$

where e_k are points sampled uniformly from $B(0,1)$.

Optimisation. Another key component of OneFlow is the optimisation algorithm for computing the $(1-\alpha)$-bounding region, leveraging the previously derived Equations 2.12 and 2.13.
We begin by considering a mini-batch $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subset X$, a flow model f_θ with parameters θ, and a given p-value $\alpha \in (0,1)$. To apply Equations 2.12 and 2.13, we must estimate the radius of the ball that encloses a $(1-\alpha)$ propotion of the data. This is done by first computing $r_i(\theta) = \| f_\theta(\mathbf{x}_i) \|$ for each sample in the mini-batch, followed by applying an estimator of the upper $(1-\alpha)$-quantile.

From an optimisation perspective, selecting an appropriate quantile estimator is crucial. We specifically seek an estimator that ensures smoothness and performs well with a small number of samples. These requirements are met by the *Bernstein estimator*[1], introduced in Definition 2.3.2.

Definition 2.3.2 (Bernstein estimator)**.** *Consider a set of samples $s_1, \ldots, s_n$ drawn from the same distribution. First, reorder these samples in ascending order: $s_{(1)} \leq \ldots \leq s_{(n)}$. The Bernstein estimate (Cheng, 1995) of the $(1-\alpha)$-quantile, denoted as s_α for $\alpha \in (0,1)$, is then given by:*

$$s_\alpha = \sum_{k=1}^{n} \binom{n-1}{k-1} \alpha^{k-1} (1-\alpha)^{n-k} \cdot s_{(k)}.$$

To estimate the $(1-\alpha)$-quantile of $r(\theta)$, we first sort $r_i(\theta)$, obtaining the sequence $(r_{(i)}(\theta))$. Then, applying the Bernstein estimator from Definition 2.3.2, we compute:

$$R_\alpha(\theta) = \sum_{k=1}^{n} \binom{n-1}{k-1} \alpha^{k-1} (1-\alpha)^{n-k} \cdot r_{(k)}(\theta). \tag{2.14}$$

By combining Eq. 2.14 with Eq. 2.12, we derive the volume of the bounding region for constant-determinant flows:

$$\mathrm{vol}(U) = w \cdot \mathrm{vol}(B(0,1)) \cdot R_\alpha(\theta)^D.$$

Similarly, using Eq. 2.13, we approximate the volume for general flows as:

$$\mathrm{vol}(U) \approx \tfrac{\mathrm{vol}(B(0,1))}{m} \cdot R_\alpha^D(\theta) \cdot \sum_{k=1}^{m} w(R_\alpha(\theta) e_k),$$

where (e_k) is a sequence of m points randomly chosen from the uniform distribution on the unit ball $B(0,1)$.

To enhance numerical stability, we reformulate these volume estimates as objective functions by taking their logarithm. For constant-determinant flows, the objective function is:

$$\mathcal{L}_\alpha(\theta) = \log(\mathrm{vol}(B(0,1))) + \log(w) + D \log R_\alpha(\theta). \tag{2.15}$$

For general flows, the objective function is:

$$\mathcal{L}_\alpha(\theta) = \log\big(\tfrac{\mathrm{vol}(B(0,1))}{m}\big) + D \log R_\alpha(\theta) + \log\Big(\sum_{k=1}^{m} w(R_\alpha(\theta) e_k)\Big). \tag{2.16}$$

[1] See *Remark 2* in Publication [I] for a detailed discussion on the choice of estimator.

Results. To thoroughly evaluate OneFlow, we conduct an extensive set of experiments across various anomaly detection tasks, comparing its performance against state-of-the-art methods, including One-Class SVM (OCSVM; Schölkopf et al., 2001), Deep Structured Energy-Based Models (DSEBM; Zhai et al., 2016), Deep Autoencoding Gaussian Mixture Model (DAGMM; Zong et al., 2018), Deep Support Vector Data Description (DSVDD; Ruff et al., 2018), and multiple variants of the Multivariate Quantile Map (MQT; Wang et al., 2019a).

In particular, we provide a quantitative assessment on real-world anomaly detection datasets such as *Thyroid*, *KDDCUP* (Asuncion and Newman, 2007), and the *PIDForest benchmark* (Gopalan et al., 2019). Additionally, we evaluate OneFlow in out-of-distribution (OOD) detection tasks, conducting extensive experiments on MNIST (Deng, 2012) and Fashion-MNIST (Xiao et al., 2017) image datasets. Moreover, we also present a way how OneFlow can be used to find closed meshes of 3D objects represented by point clouds. Across all experiments, OneFlow achieves comparable or superior performance relative to existing state-of-the-art models.
Whereas the complete results for the mentioned benchmarks are available in Sec. 4 in Publication [I], in this section, we will focus on the performance on MNIST and Fashion-MNIST datasets, as these are the closest to the rest of our publications forming this Ph.D. dissertation.

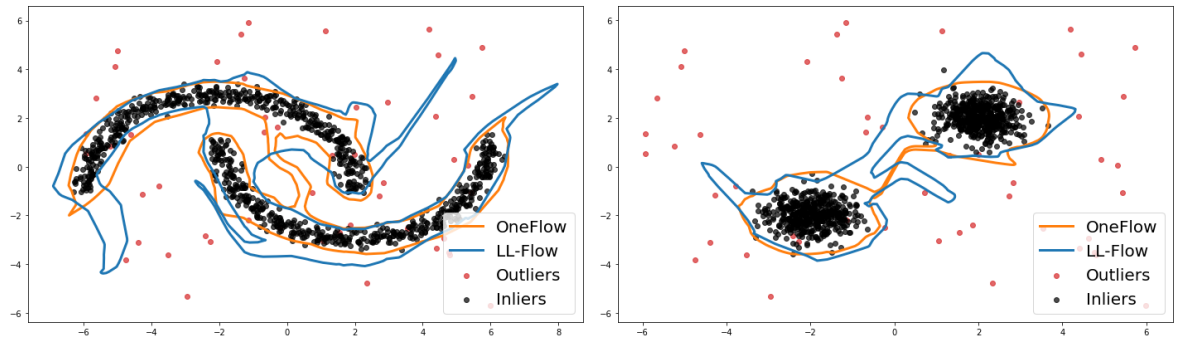


Fig. 2.4 Comparison of bounding regions constructed by OneFlow and a flow-based density model (LL-Flow). Nominal data points are shown as black dots, while anomalies are represented by red dots.

Before presenting the quantitative results, let us first develop an intuition for how OneFlow constructs a minimal-volume bounding region and compare it to a typical normalising flow model optimising log-likelihood. In Fig. 2.4, we illustrate the bounding regions produced on two simple 2D datasets. OneFlow generates significantly tighter bounds compared to a standard normalising flow, highlighting its ability to better encapsulate the data distribution.

Next, we evaluate OneFlow on vision datasets, adapting them to the anomaly detection setting. Specifically, each of the ten classes in MNIST and FashionMNIST is treated as the nominal class, while the remaining nine are considered anomalies. This setup results in ten scenarios for each dataset. In this scenario, we also evaluate the following methods: Variational Autoencoder

We report the empirical results in Fig. 2.5, using the AUC score (area under the ROC curve) as the evaluation metric. Our observations show that the overall ranking of OneFlow and OneFlow-Gen is comparable to the best-performing methods across both datasets. However, due to the high variation in results, it is challenging to definitively determine the top-performing method.

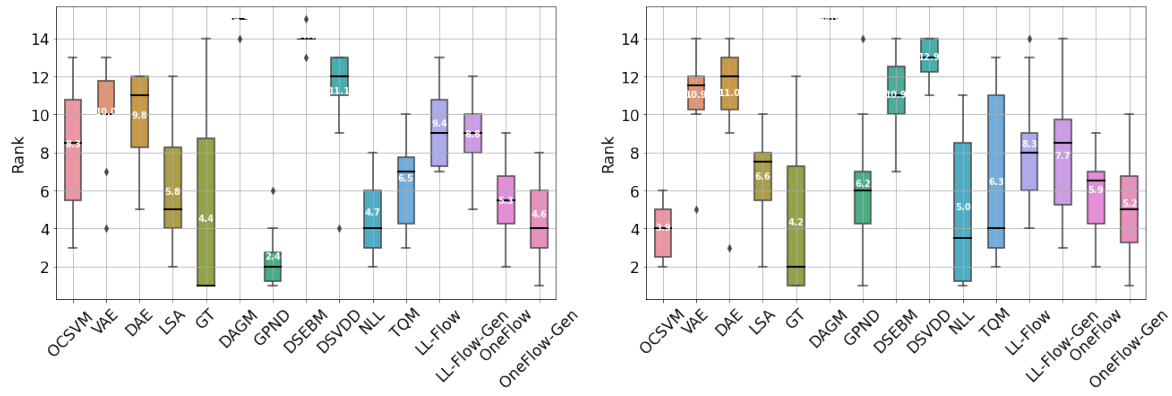


Fig. 2.5 Box plots of rankings based on AUC scores for vision datasets: MNIST (**left**) and FashionMNIST (**right**). The median ranking is indicated by a line, while the average ranking is displayed as a number.

2.3.3 Contribution and impact

I am one of the co-authors of this publication (Maziarka et al., 2021), which marked my first contribution during my Ph.D. studies. My primary responsibilities included implementing and conducting experiments for the proposed method, particularly on low-dimensional settings such as standard 2D benchmarks. I also proposed and developed a novel application of the OneFlow framework for generating meshes from 3D point clouds — an extension that broadened the scope of the method and demonstrated its versatility.

Beyond the experimental work, I actively contributed to the conceptual development of the approach, its software implementation, and the preparation of the manuscript. The paper has been published in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, one of the leading journals in computer vision and machine learning. As of August 2025, it has been cited 10 times, primarily by works in the one-class classification and anomaly detection communities (Hojjati and Armanfard, 2023; Huang et al., 2024; Kim et al., 2023; Ruff, 2021; Su et al., 2024).

2.4 Flow-based SVDD for anomaly detection [II]

In the following section, we focus on Publication [II], where we continue improving other methods operating on minimal-volume bounding regions for anomaly detection. Specifically, we propose a novel deep learning formulation of SVDD that effectively removes the limitations present in existing deep learning adaptations. Unlike previous approaches, our method, **FlowSVDD**, effectively prevents hypersphere collapse without imposing restrictive constraints that limit the flexibility of neural networks.

This is achieved by incorporating a constant-determinant normalising flow model and appropriately redefining the classical SVDD objective. Experimental results in Publication [II] demonstrate that FlowSVDD significantly outperforms other deep learning adaptations of SVDD.

2.4.1 Motivation

Similarly to Publication [I], we approach anomaly detection by seeking a minimal-volume hypersphere that encloses a desired proportion of data. However, in this work, our primary goal is different — we aim to develop a proper deep learning formulation of a classical one-class method.

As discussed previously (*see* Sec. §2.3.1 and Sec. §2.3.2), Support Vector Data Description (**SVDD**; Tax and Duin, 2004) also tackles the problem of minimal-volume bounding regions. However, SVDD cannot directly solve this problem. Instead, to enforce convexity, it reformulates the objective, introducing a soft margin and penalising points outside the bounding region.

SVDD remains one of the strongest classical anomaly detection methods, leading to deep learning adaptations such as DeepSVDD (DSVDD; Ruff et al., 2018). However, directly minimising the SVDD loss in deep learning can result in a trivial solution where the hypersphere collapses to a single point (Chong et al., 2020). Our work addresses this limitation by proposing a novel formulation that ensures non-collapsing behaviour in the deep learning regime.

2.4.2 Content

Following Publication [II], we first present the SVDD objective function and then introduce a deep learning formulation that prevents collapse into a trivial solution.

Support vector data description (SVDD). To derive the SVDD objective, let us assume that f is a function mapping input data to an output kernel space. The objective function is

defined as:

$$\mathcal{L}_{\text{SVDD}}(R, \mathbf{c}; f) = R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max(0, \|f(\mathbf{x}_i) - \mathbf{c}\|^2 - R^2), \tag{2.17}$$

where $\mathbf{c} \in \mathbb{R}^D$ and $R \in \mathbb{R}$ are the centre and radius of the bounding hypersphere, respectively. Here, n denotes the batch size, and ν controls the trade-off between the hypersphere's volume and the penalty for boundary violations, effectively setting the fraction of allowed outliers.

Existing deep learning adaptations of SVDD parameterise f as a neural network. However, directly minimising Eq. 2.17 can lead to **hypersphere collapse**, where the solution degenerates to a single point $\mathbf{c}$ (Chong et al., 2020). To mitigate this, prior approaches have imposed constraints such as ensuring c is non-trivial, removing biases and bounded activation functions, or introducing additional regularisers. Unfortunately, these solutions significantly restrict the flexibility of the neural network.

Flow-based SVDD — FlowSVDD. Instead of using unconstrained neural networks, we propose parameterising the mapping function f as a **constant-determinant flow**, such as the NICE model (Dinh et al., 2014), so $f : \mathbb{R}^D \to \mathbb{R}^D$. This leads to a reformulation of the SVDD objective (Eq. 2.17), resulting in our method, **FlowSVDD**.

To derive the objective, let $w(\mathbf{x}) = \det(df(\mathbf{x}))$ denote the determinant of the Jacobian. For the class of flow models we use, this determinant is constant, meaning $w(\mathbf{x}) = w$. Notably, when $w = 1$, the volume of input data remains unchanged, preventing hypersphere collapse. In Publication [II], we extend this idea to the general case ($w \neq 1$) and redefine the SVDD objective as:

$$\mathcal{L}_{\text{FlowSVDD}}(R, \mathbf{c}; f) = R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max(0, \| \frac{f(\mathbf{x}_i)}{w^{\frac{1}{D}}} - \mathbf{c}\|^2 - R^2). \tag{2.18}$$

This formulation inherently prevents collapse since the function $\frac{f}{w^{\frac{1}{D}}}$ maintains a Jacobian determinant of 1. Moreover, we can show that all inliers (normal data) lie inside the ball $B(\mathbf{c}w^{\frac{1}{D}}, Rw^{\frac{1}{D}})$.

Results. To comprehensively evaluate FlowSVDD, we conduct experiments similar to those in Publication [I]. Specifically, we consider real-world anomaly detection benchmarks (*Thyroid* and *KDDCUP*) (Asuncion and Newman, 2007), vision datasets (MNIST and FashionMNIST) (Deng, 2012; Xiao et al., 2017), and 2D toy problems to visualise FlowSVDD's bounding regions. A complete description of the experimental setup is provided in Publication [II] and Sec. §2.3.2.

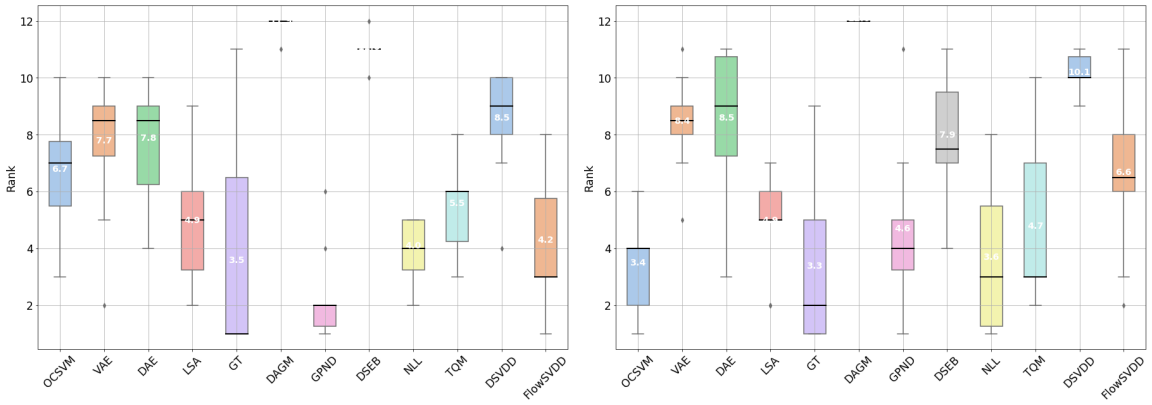


Fig. 2.6 Box plots of rankings based on AUC scores for vision datasets: MNIST (**left**) and FashionMNIST (**right**). The median ranking is indicated by a line, while the average ranking is displayed as a number. FlowSVDD performs comparably to state-of-the-art methods on MNIST and slightly worse on FashionMNIST, but significantly outperforms its main competitor, DSVDD.

Fig. 2.6 presents the results for image datasets. FlowSVDD achieves performance similar to state-of-the-art methods on MNIST and slightly lower on FashionMNIST. However, its primary competitor, DSVDD, performs significantly worse. We hypothesise that this is due to hypersphere collapse in DSVDD.

To further analyse FlowSVDD's behaviour, Fig. 2.7 illustrates how the method encloses inliers in a Gaussian distribution within its latent space. Notably, we do not observe hypersphere collapse, confirming the effectiveness of our approach.

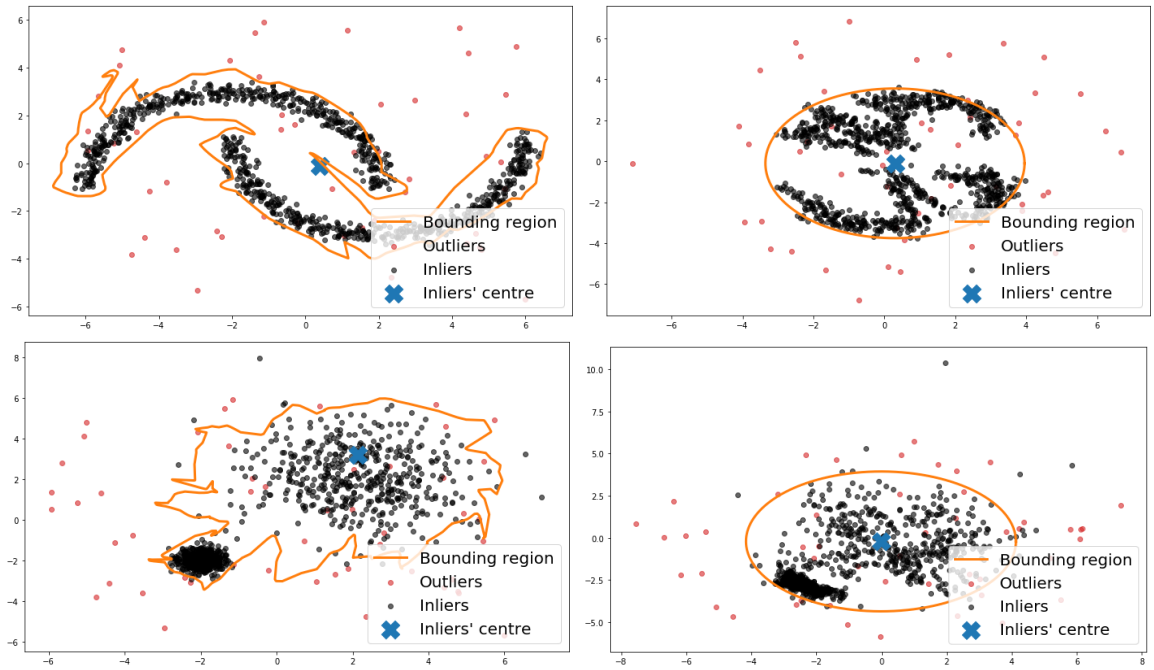


Fig. 2.7 Enclosing hyperspheres in the latent space of FlowSVDD (**right**) and the corresponding bounding regions in the input space (**left**).

2.4.3 Contribution and impact

I am the first author and primary contributor to this paper (Sendera et al., 2021a), which builds upon the insights and experience gained from my earlier work in Publication [I]. My main contribution was the conceptual development of a novel deep learning formulation of the Support Vector Data Description objective, enabling its integration with normalising flows.

In addition to leading the conceptual work, I implemented the proposed method, designed and conducted the experiments, and performed a thorough comparative analysis against baseline models. I was also responsible for writing the majority of the manuscript.

The paper was accepted to the 2nd edition of the ICML workshop on normalising flows (INNF+), titled *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*. I presented this work on the workshop during the virtual ICML 2020 conference — CORE A* event.

2.5 Missing glow phenomenon: learning disentangled representation of missing data [III]

In the previously introduced publications, we primarily focused on developing novel anomaly detection methods and enhancing existing ones by leveraging key properties of normalising flows, such as efficient density estimation and bijective mappings. In the following Publication [III], we shift our focus to a broader property of discrete-time flow models, specifically Glow (Kingma and Dhariwal, 2018). We investigate an intriguing behaviour that emerges when these models are trained on datasets with missing data — samples where certain regions are absent (Goodfellow et al., 2016; Little and Rubin, 2019).

We introduce this newly observed behaviour, which we term the *Missing Glow Phenomenon*, and provide a theoretical explanation for its underlying cause, including a method to estimate the expected number of missing parts in sampled images. We empirically validate our hypothesis, showing a close but imperfect alignment between theory and observations, and demonstrate how this mechanism can be utilised for imputing missing regions.

2.5.1 Motivation

Normalising flows (Dinh et al., 2014; Rezende and Mohamed, 2015), particularly discrete-time models like Glow (Kingma and Dhariwal, 2018), have gained significant attention in generative modelling due to their bijective mappings and exact likelihood estimation. While these models are typically trained on complete datasets, real-world data is often corrupted, containing missing or unobserved regions.

In this work, we investigate the potential of normalising flows for handling missing data, hypothesising that their exact likelihood computation and well-structured latent space make them well-suited for imputing missing parts. Surprisingly, we observe that Glow can generate coherent, full samples despite being trained on incomplete observations (*see* Fig. 2.8). We introduce this mechanism, and hypothesise that it emerges due to the model's inductive biases and the way it learns structured representations. Understanding this phenomenon is crucial both for theoretical insights into normalising flows and for practical applications, such as image inpainting (Xie et al., 2012; Yeh et al., 2017).

2.5.2 Content

To investigate the problem of imputing missing regions, we first trained a Glow model $f : \mathbb{R}^N \to \mathbb{R}^N$ on a specially designed dataset with missing regions (**missing dataset**). Specifically, we used the CelebA dataset (Liu et al., 2015), rescaled to 64×64 pixels, where we randomly placed grey occlusions of fixed sizes, ranging from 2×2 to 20×20 pixels. In each experiment,

Fig. 2.8 Glow was trained on a dataset with missing regions and then sampled using a temperature of $t = 0.7$(the standard setting for Glow), resulting in approximately 60% of generated images being fully reconstructed.

the occlusion size was consistent across all samples, and each image had its occlusion fixed in the same position throughout training.

Missing glow phenomenon. After training the Glow model, we sampled from its latent representation using various temperature values t. Surprisingly, we observed that sampling with a typical temperature ($t = 0.7$) produced a significant number of fully reconstructed images, with approximately 60% of samples containing no missing regions. Moreover, lowering the sampling temperature ($t \leq 0.7$) further increased the proportion of complete images, while increasing it ($t \geq 0.7$) led to more samples retaining missing regions (*see* Fig. 2.9). We refer to this behaviour as the *Missing Glow Phenomenon*.

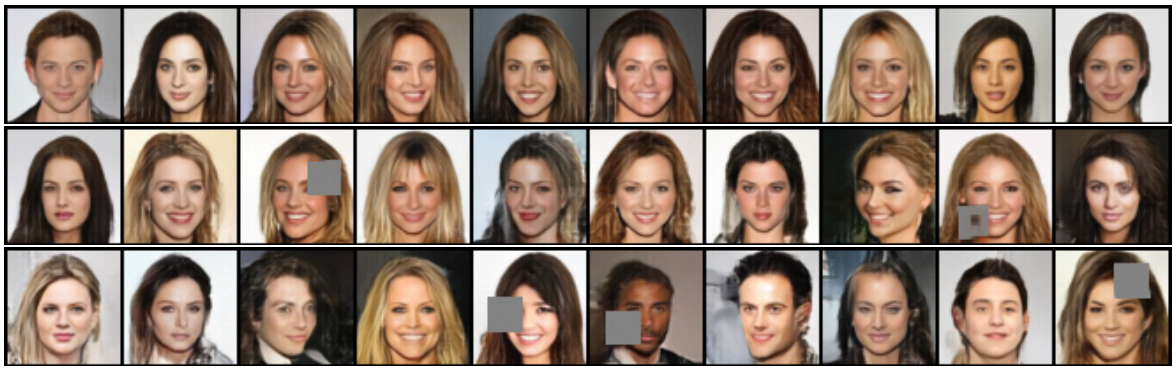


Fig. 2.9 Samples drawn from Glow at different sampling temperatures ($t = \{0.5, 0.6, 0.7\}$, **top to bottom**). Lower temperatures produce more complete images, while higher temperatures introduce more missing regions. In each experiment, missing regions were fixed in size across all training samples.

Moreover, we hypothesise that this behaviour arises due to the structure of the Glow latent space. Specifically, we assume that images from the missing dataset can be decomposed into two independent random variables—one representing the original image and the other representing the grey occlusion. We analyse how these components interact in the latent space and how their combination depends on the sampling temperature t. Based on this reasoning, we formulate our hypothesis as an expected value for the number of missing

squares s when sampling from the Glow model f with temperature t:

$$\mathbb{E}[s \mid \mathbf{x} \sim f^{-1}(\cdot; t)] = t^2. \tag{2.19}$$

In Publication [III], we empirically validate this hypothesis, demonstrating that for typical sampling temperatures ($t \in (0.5, 0.9)$), our estimation aligns well with observed results, though not perfectly. The discrepancy between theoretical and empirical values likely stems from the inherent imperfections of the trained model f.

Imputing missing parts. Beyond describing this novel phenomenon in flow models trained on datasets with missing regions, we also propose a method to leverage this knowledge for imputing the missing regions. This results in reconstructions as shown in Fig. 2.10.

Fig. 2.10 Images reconstructed using the proposed procedure. The parameter α scales the latent representation of the original image, where $\alpha = 1.0$ corresponds to the original image.

Specifically, we introduce the following procedure:

1. Let $\mathbf{x}$ be an image from the missing dataset.
2. Compute its latent representation $\mathbf{z}$ using the Glow model: $\mathbf{z} = f(\mathbf{x})$.
3. Adjust the latent representation by scaling it with a parameter $\alpha \in (0, 1]$ forming a new vector $\mathbf{z}' = \alpha\mathbf{z}$.
4. Reconstruct the imputed image $\mathbf{x}'$ via: $\mathbf{x}' = f^{-1}(\mathbf{z}') = f^{-1}(\alpha\mathbf{z}) = f^{-1}(\alpha f(\mathbf{x}))$.

2.5.3 Contribution and impact

I am the first author and primary contributor of this paper (Sendera et al., 2021b). My work involved identifying and describing the unexpected behaviour observed in Glow-based models, designing and implementing the experimental setup, and conducting an in-depth analysis of the phenomenon.

Additionally, I contributed to formulating the mathematical hypothesis that explains the observed behaviour and was responsible for writing the majority of the manuscript.
The paper was accepted to the ICONIP 2021 conference, which was classified as a CORE A-ranked event at the time of submission. I presented this work during the virtual conference.

3

From noise to structure: on efficient sampling with diffusion processes

3.1 Research scope

On the importance of sampling in probabilistic deep learning

In the previous chapter, we considered *density estimation* — one of the fundamental problems in probabilistic deep learning — and demonstrated how normalising flows can be used as exact density estimators.

In this chapter, we turn our attention to another, perhaps even more challenging, problem: *approximating and sampling from unknown densities or energy functions*, especially when we have limited or no access to data (Duane et al., 1987; Hoffman et al., 2014; Roberts and Tweedie, 1996). This problem is central not only to statistical modelling and Bayesian inference (Gabrié et al., 2021) but also to the development of reliable, general-purpose AI systems (Bengio et al., 2025, 2024; Dalrymple et al., 2024).

From the perspective of core machine learning research, sampling plays a crucial role in mitigating dataset biases, enhancing generalisation, and enabling principled uncertainty quantification (Murphy, 2012; Wilson and Izmailov, 2020). This becomes especially important when moving beyond point estimates to reasoning over entire distributions — whether in the context of posterior inference or planning under uncertainty (Barber, 2012). In fact,

efficient density approximation and sampling are essential components of any form of Bayesian inference and reasoning in AI systems (Griffiths et al., 2010). Such considerations are particularly relevant when models act as interactive agents, such as reinforcement learning (RL) policies (Kaelbling et al., 1996; Levine, 2018; Sutton et al., 1998) operating in real-world environments, where in-the-wild decision-making is critical. Moreover, even when models are pretrained (*e.g.* language models or generative priors), we often wish to steer them toward new distributions, like those induced by a specific goal or downstream objective, which again requires sampling from a desired posterior (Venkatraman et al., 2024b).

Yet, the need for efficient sampling extends beyond deep learning applications (Harrison et al., 2024; Hernández-Lobato and Adams, 2015; Izmailov et al., 2021) and touches on some of the most fundamental scientific challenges (Jing et al., 2022; Noé et al., 2019). In many domains — from molecular discovery (Holdijk et al., 2023) to inverse problems in physics (Adam et al., 2022) — we often have only a limited number of observations or none at all, yet we can evaluate complex energy functions or reward-like objectives. For instance, in drug design, we may be able to compute the energy or predicted efficacy of a candidate molecule, but the space of valid molecular structures is astronomically large and unknown. Similarly, in astrophysics, quantum mechanics, or nuclear physics, we frequently face inverse problems where the true generative process is only partially observed and only accessible via indirect measurements or simulations (Scimeca et al., 2025).

While these real-world examples, like working with unknown energy landscapes, molecular configurations, or decision-making under uncertainty, may differ in surface form, they share a common structure. In each case, access to ground-truth data or distributions is limited or even non-existent. The target densities are complex, high-dimensional and multivariate, and are usually *unnormalised* (have an unkown partition function). Making progress in such settings requires methods that can efficiently explore, generalise, and finally sample from these intractable densities (Wilson and Izmailov, 2020).

So, how can we sample from such unknown, unnormalised distributions? The gold standard is to use classical approaches like Markov Chain Monte Carlo methods (MCMC; Duane et al., 1987; Grenander and Miller, 1994; Roberts and Tweedie, 1996), which offer asymptotically unbiased estimates. However, MCMC methods often suffer from poor scalability for higher-dimensionalities and high computational overhead, making them infeasible for considered challenges (Chopin, 2002; Del Moral et al., 2006; Halton, 1962). Variational inference (VI) offers a more scalable alternative but typically involves designing a parametric variational functions family for each variable and solving an optimisation problem for each target distribution.

All of those methods, generally lack amortisation (*e.g.*, cannot re-use learned inference across new tasks). In contrast, amortised variational inference techniques — often realised through flexible, learnable samplers like diffusion-based ones (Zhang and Chen, 2022) — offer a promising direction. These approaches are the central focus of the publications presented in this chapter.

Diffusion samplers

In general, diffusion models can be seen as deep, hierarchical generative models that learn to transform noise into data through a continuous-time process, formalised as a stochastic differential equation (SDE; Song et al., 2021b). Their recent popularity in generative modelling, mostly comes from their ability to act as powerful density estimators that generate high-quality samples (Ho et al., 2020). Moreover, formalisation as SDEs rather than ODEs (like *normalising flows*) removes the bijectivity requirement, which grants them greater modelling flexibility, allowing them to approximate arbitrary distribution given access to data samples.

This raises questions: *can we train diffusion processes to sample from unknown densities even without access to data? Can they explore energy landscapes efficiently?*
To answer these questions, we should acknowledge the unique nature of diffusion models. They can be simultaneously seen as: deep hierarchical generative models (like hierarchical VAEs), SDEs simulated in discrete-time, and policies in a Markov decision process (MDP; Lahlou et al., 2023). While the first two views were already mentioned, the third — viewing diffusion models as policies (Bengio et al., 2023) — is particularly important for our purposes. This interpretation naturally connects diffusion models to RL and the stochastic optimal control theory. As a result, diffusion-based samplers can be trained to explore a target density, and in fact they can be trained with only access to an energy function.

Diffusion-based samplers can be formulated from two complementary perspectives. The first comes from the continuous-time SDE view, where training involves tools from stochastic control theory and neural SDEs (Richter and Berner, 2024). The second builds on a discrete-time perspective, where the sampler is viewed as a policy in an MDP. This perspective is realised in the framework of continuous generative flow networks (GFlowNets), where training is based on forward–reverse consistency or GFlowNet-like objectives (Bengio et al., 2023). This also enables the use of RL techniques — particularly those designed for improved exploration — which are crucial in domains like scientific discovery and AI safety (*e.g.*, for avoiding failure modes, out-of-distribution generalisation, or steering) (Bengio et al., 2025). GFlowNet-based samplers further implement amortised variational inference and enable efficient scaling to complex, high-dimensional distributions (Malkin et al., 2023).

Despite the success of diffusion models and — more recently, diffusion samplers — many open problems remain. Diffusion samplers can still be computationally expensive to train, even if they outperform classical MCMC methods in efficiency. When formulated as neural SDEs, their performance often depends on the trajectory length, discretisation schemes, and the need to compute gradients through the entire generative trajectory (Zhang and Chen, 2022). In contrast, GFlowNet-based samplers offer a significant advantage: they can be trained *off-policy*, *i.e.*, without backpropagating through the full simulation process, improving their scalability (Malkin et al., 2022; Sendera et al., 2024a).

Another key challenge lies in the design of inductive biases, architecture choices, and proposal mechanisms that facilitate more efficient exploration. Finally, a central research question remains: can we design more efficient and theoretically grounded training strategies for diffusion samplers?

Role of diffusion samplers in the publications that form this dissertation

In this chapter, we present the works that form this Ph.D. series of publications and contribute to addressing the open problems outlined above.

First, in Publication [IV], we focus on improving off-policy diffusion-style samplers by incorporating better inductive biases and proposals. We benchmark existing approaches and introduce a novel off-policy training method that leverages a replay buffer combined with a parallel MALA local search. We demonstrate its effectiveness on an extensive set of sampling tasks, including unnormalised densities and Bayesian posteriors.

While these developments improve the training efficiency of diffusion-style samplers, a deeper question arises: *how are diffusion samplers based on neural SDEs and those based on GFlowNets connected? More precisely, what happens when policies in a discrete Markov chain approach a continuous-time limit?*

In Publication [V], we build a theoretical bridge between these two perspectives. We show that the training objectives of discrete-time diffusion samplers asymptotically converge to their continuous-time counterparts. As a side, we also explore the role of discretisation schemes and demonstrate that using non-uniform, coarser discretisations during training results in faster convergence.

This chapter is organised as follows. In Section §3.2, we introduce the mathematical background on diffusion-style samplers, covering both the continuous and discrete perspectives. This section serves as a self-contained reference and may be skipped by readers already familiar with the area. Next, Section §3.3 presents our contributions to improving off-policy training methods, and Section §3.4 establishes the theoretical connections between neural SDEs and GFlowNets.

3.2 Background

To better present the novelty and contribution of Publication [IV] and Publication [V] to the field of diffusion-based samplers, and for the completeness of this dissertation, we begin with a high-level overview of the most relevant concepts.
We start by formulating the general problem of sampling from a complex, multivariate distribution when only an unnormalised probability density or energy function is available, as stated in Problem 2.

Problem 2 (Sampling from an unnormalised density or energy function)**.** *Let $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ be a differentiable* ***energy function*** *and we define the* ***reward*** *or* ***unnormalised target density*** *as $R(\mathbf{x}) = \exp(-\mathcal{E}(\mathbf{x}))$.*
Assuming the integral $Z := \int_{\mathbb{R}^d} R(\mathbf{x})\, d\mathbf{x}$ exists, the energy function $\mathcal{E}$ defines a ***Boltzmann density*** *$p_{\text{target}}(\mathbf{x}) = \frac{R(\mathbf{x})}{Z}$ on $\mathbb{R}^d$.*
The core problem is to sample from p_{target} and to estimate the ***partition function*** *Z, given access only to $\mathcal{E}$ and possibly to its gradient $\nabla\mathcal{E}$.*

This problem is exceptionally challenging in high-dimensional spaces or when the distribution exhibits multiple disjoint modes (Bandeira et al., 2022). Such scenarios frequently arise in probabilistic deep learning (Harrison et al., 2024; Hernández-Lobato and Adams, 2015; Izmailov et al., 2021; Mittal et al., 2023; Radev et al., 2020) and various scientific applications (Adam et al., 2022; Albergo et al., 2019; Holdijk et al., 2023; Jing et al., 2022; Noé et al., 2019). An example is presented in Fig. 3.1, which shows the complex energy function landscape of *alanine dipeptide* (Phillips and Cipcigan, 2024).

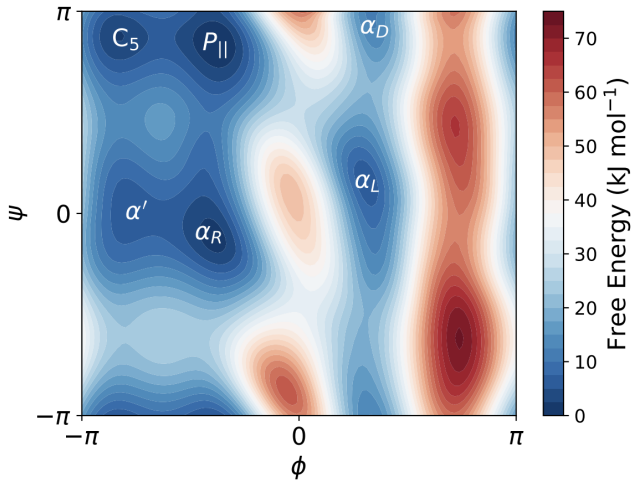


Fig. 3.1 Free energy surface of alanine dipeptide in explicit solvent. The energy function landscape exhibits multiple disjoint modes. *Image source:* Phillips and Cipcigan (2024).

Currently, the gold standard for such sampling tasks comprises Markov chain Monte Carlo (MCMC) methods, known for providing asymptotically unbiased estimators. Among the most popular are Metropolis-adjusted Langevin algorithms (MALA; Grenander and Miller, 1994; Roberts and Rosenthal, 1998; Roberts and Tweedie, 1996) and Hamiltonian MC (HMC; Duane et al., 1987; Hoffman et al., 2014). However, these methods suffer from slow mixing between modes, and entail high computational costs, especially for complex distributions.

Another set of approaches are a variants of the previous, *e.g.*, sequential Monte Carlo (SMC; Chopin, 2002; Del Moral et al., 2006; Halton, 1962) or nested sampling (Buchner, 2021; Lemos et al., 2023; Skilling, 2006), which offer better mode coverage but face even greater challenges in scaling to higher-dimensional problems.

These limitations have motivated the arising and development of amortised variational inference techniques, where we fit the parametric models to approximate sample from the target densities in a computationally efficient, reusable way.

On diffusion processes for sampling

Diffusion models are a family of continuous-time stochastic processes that transform samples from a simple distribution (typically Gaussian) into samples from a more complex target distribution by gradually removing noise. This process, illustrated in Fig. 3.2, facilitates effective mode mixing (De Bortoli, 2022). Traditionally, diffusion models have been used as density estimators, and as such have found broad application in generative modelling, while being learned from true data (Ho et al., 2020; Nichol and Dhariwal, 2021; Rombach et al., 2021; Sohl-Dickstein et al., 2015; Song et al., 2021b).

However, the problem of using diffusion models for sampling from an *unnormalised* density or energy function — *see* Problem 2 — only very recently gained significant attention.

This problem can be viewed from two distinct but closely related perspectives: (1) through the lens of neural stochastic differential equations (**SDEs**) and stochastic control, and (2) through the framework of continuous-time generative flow networks (**GFlowNets**).

In the first perspective, the key insight is the connection between diffusion (learning the denoising process) and stochastic control (learning the Föllmer drift (Föllmer, 1985)). This connection has led to the recent development of methods such as the Path Integral Sampler (PIS; Zhang and Chen, 2022), the Denoising Diffusion Sampler (DDS; Vargas et al., 2023), and the Time-Reversed Diffusion Sampler (DIS; Berner et al., 2022). These approaches were recently unified by Richter et al. (2023) and Vargas et al. (2024).

Whereas, the second perspective builds on continuous-time GFlowNets, which are deep reinforcement learning algorithms adapted to variational inference. GFlowNets offer stable

off-policy training and thus flexible exploration even in complex energy landscapes (Malkin et al., 2023). This perspective resulted in the few approaches (Lahlou et al., 2023; Zhang et al., 2024).

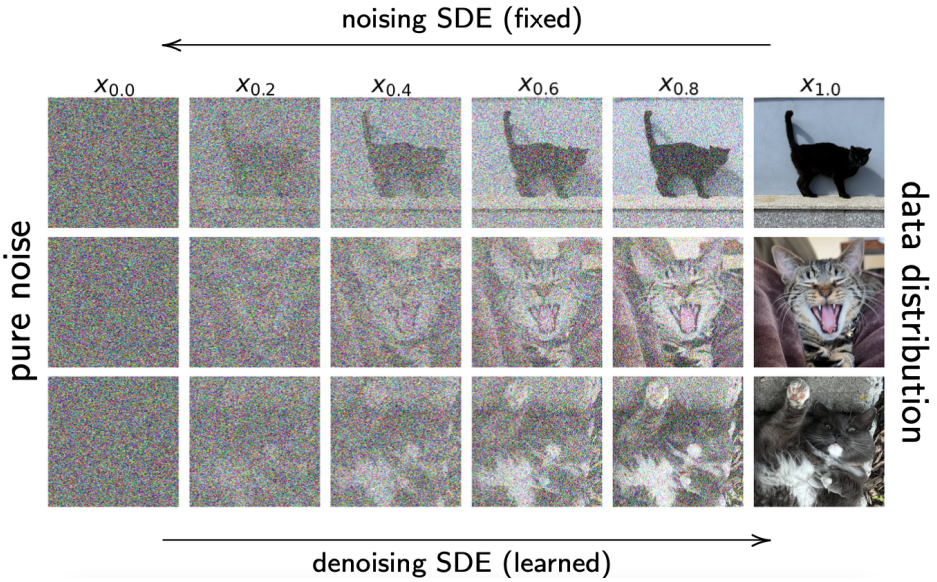


Fig. 3.2 Illustration of a diffusion process that transforms a sample from a simple distribution (*e.g.*, pure Gaussian noise) into one from a complex data distribution. We are usually learning a denoising SDE, while keeping its reverse, noising SDE, fixed.

Stochastic control perspective (Euler-Maruyama hierarchical samplers)

Generative modelling with SDEs. First, diffusion models assume having continuous-time generative processes, modelled using neural stochastic differential equations (SDEs; Øksendal, 2003; Särkkä and Solin, 2019; Tzen and Raginsky, 2019). The general form of such an SDE is:

$$d\mathbf{x}_t = u(\mathbf{x}_t, t; \theta)\, dt + g(\mathbf{x}_t, t; \theta)\, d\mathbf{w}_t, \tag{3.1}$$

where $\mathbf{x}_0$ comes from a fixed probability density distribution μ_0, typically a point mass or Gaussian. For the fixed distribution μ_0 and the dynamics given by Eq. 3.1, this SDE induces a marginal distribution p_t over $\mathbb{R}^d$ at each time $t > 0$. In neural SDEs, both the drift function u and diffusion function g are parameterised by neural networks with learnable parameters θ, which are optimised so that the terminal distribution p_1 approximates the target distribution p_{target}. Sampling from p_1 is then straightforward: draw $\mathbf{x}_0 \sim \mu_0$ and simulate the SDE (Eq. 3.1) from time $t = 0$ to $t = 1$.

To enforce the **uniqueness of the SDE** pushing μ_0 to p_{target}, we fix a reverse-time SDE — also called the *noising process* — which evolves from the target density p_{target} at $t = 1$ to the initial distribution μ_0 at $t = 0$. Under mild conditions, this reverse process determines a unique forward-time SDE (*i.e.*, *denoising process*). Typically, we design such a noising process, so that we can learn the drift u of the denoising process *given samples from* p_{target} using stochastic regression objectives from samples of p_{target}, while the diffusion rate g is available in closed form (Ho et al., 2020; Song et al., 2021b).

Time discretisation. Since we cannot directly operate on continuous-time objects like SDEs, we approximate them via discretisation schemes. The most common choice is the Euler–Maruyama integration scheme, which discretises the continuous-time process (SDE) into a discrete-time Markov chain, $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \mathbf{x}_{2\Delta t} \to \cdots \to \mathbf{x}_1$ of T steps:

$$\mathbf{x}_0 \sim \mu_0, \quad \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + u(\mathbf{x}_t, t; \theta)\,\Delta t + g(\mathbf{x}_t, t; \theta)\sqrt{\Delta t}\,\mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \tag{3.2}$$

where $\Delta t = 1/T$ is the fixed time increment.
The corresponding transition density of the discretised forward process p_F can be written explicitly:

$$p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+\Delta t}; \mathbf{x}_t + u(\mathbf{x}_t, t; \theta)\Delta t, g(\mathbf{x}_t, t; \theta)^2 \Delta t \mathbf{I}_d). \tag{3.3}$$

Moreover, the joint distribution over a trajectory starting from $\mathbf{x}_0$ under this forward process is also given explicitly:

$$p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0) = \prod_{i=0}^{T-1} p_F(\mathbf{x}_{(i+1)\Delta t} \mid \mathbf{x}_{i\Delta t}). \tag{3.4}$$

Analogously, if consider a reverse-time discrete Markov chain: $\mathbf{x}_1 \to \mathbf{x}_{1-\Delta t} \to \mathbf{x}_{1-2\Delta t} \to \cdots \to \mathbf{x}_0$, with transition densities $p_B(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$. The joint trajectory distribution under this backward process is:

$$p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1) = \prod_{t=1}^{T} p_B(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}). \tag{3.5}$$

We need the forward (starting from μ_0) and backward (starting from p_{target}) processes to be true time-reversals of each other in order to define the same distribution over trajectories. For this, we require the following consistency condition:

$$\mu_0(\mathbf{x}_0)\, p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0) = p_{\text{target}}(\mathbf{x}_1)\, p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1). \tag{3.6}$$

When this condition holds, the marginal densities of $\mathbf{x}_1$ are equal to p_{target} — under forward and backward processes. Finally, we can then use the forward process to sample from the target density.

SDE learning as hierarchical variational inference. The problem of learning the parameters θ of the forward SDE process (Eq. 3.1) can be seen as hierarchical variational inference. This interpretation arises from the following observations:

- The backward process transforms the terminal state $\mathbf{x}_1$ into the initial state $\mathbf{x}_0$ via a sequence of latent variables $\mathbf{x}_{1-\Delta t}, \ldots, \mathbf{x}_0$.
- The forward process is trained to match the posterior distribution over these variables and approximate the stochastic dynamics (Eq. 3.1).

When we train diffusion models with access to true data samples from p_{target}, the forward process is typically optimised by simply minimising the KL divergence between the forward and backward trajectory distributions:

$$D_{\text{KL}}(p_{\text{target}} \cdot p_B \| \mu_0 \cdot p_F),$$

which is equivalent to maximising a variational lower bound on the data log-likelihood (Song et al., 2021a).

However, in this setting, we sample from an unnormalised density p_{target} — we lack unbiased estimators for this (forward) KL divergence. In such a case, we instead minimise the reverse KL:

$$\begin{aligned} &D_{\text{KL}}(\mu_0 \cdot p_F \| p_{\text{target}} \cdot p_B) \\ &= \int \log \frac{\mu_0(\mathbf{x}_0) p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0)}{p_{\text{target}}(\mathbf{x}_1) p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1)} d\mu_0(\mathbf{x}_0) p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0) \; d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_1. \end{aligned} \tag{3.7}$$

Various estimators of this objective have been proposed, including approaches based on the reparameterisation trick (Kingma and Welling, 2014). These reformulate Eq. 3.7 as an expectation over noise variables $\mathbf{z}_t$ that participate in the hierarchical sampling of the intermediate states $\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1$, as in PIS (Zhang and Chen, 2022) and DDS (Vargas et al., 2023). Although some of these estimators are unbiased, they typically require backpropagation through the forward process simulation, which can be computationally expensive.

Continuous GFlowNets (Euler-Maruyama samplers as GFlowNets)

An alternative view of enforcing the presented consistency condition (Eq. 3.6) is to formulate it as a reinforcement learning (RL) problem, as proposed by continuous generative flow networks (GFlowNets) (Lahlou et al., 2023). Below, we present their connection to neural SDE and the associated learning objectives.

State and action space. To express the sampling process in an RL paradigm, we must first define the state and action spaces. We consider sampling using a T-step Euler-Maruyama integration scheme, with μ_0 taken as a point mass at $\mathbf{0}$. The state space is then composed of elements $(\mathbf{x}, t)$, representing the sampling agent being at position $\mathbf{x}$ at time t:

$$\mathcal{S} = \{(\mathbf{0}, 0) \cup \left\{(\mathbf{x}, t) : \mathbf{x} \in \mathbb{R}^d, t \in \{\Delta t, 2\Delta t, \dots, 1\}\right\}.$$

The sampling agent starts from the *initial state* $\mathbf{x}_0 := (\mathbf{0}, 0)$ and proceeds through the sequence of states $(\mathbf{x}_{\Delta t}, \Delta t)$, $(\mathbf{x}_{2\Delta t}, 2\Delta t)$, …, $(\mathbf{x}_1, 1)$. Sampling terminates at a *terminal state* $(\mathbf{x}, t = 1)$, and the collection of all such terminal states is denoted as $\mathcal{X}$. The entire sequence $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$[1] forms a *complete trajectory*.
From each non-terminal state $(\mathbf{x}_t, t)$, we can perform an *action*, which is to transition to a new state $(\mathbf{x}_{t+\Delta t}, t + \Delta t)$ achievable from the current state via one step of Euler-Maruyama integration scheme.
Having established the state and action spaces, we now proceed to define the forward and backward policies.

Forward policy and learning problem. A *forward policy* is a set of continuous distributions over the states reachable via a single action of every state $(\mathbf{x}, t)$, excluding the terminal one. In the context of the sampling problem, the forward policy is a collection of conditional probability densities $p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t; \theta)$, which represent the transition kernel from $\mathbf{x}_t$ to $\mathbf{x}_{t+\Delta t}$. During training of a GFlowNet, we optimise the parameters θ — typically the weights of a neural network that outputs the density over $\mathbf{x}_{t+\Delta t}$ given the current state $\mathbf{x}_t$ as input.

The forward policy p_F induces a distribution over complete trajectories $\tau = (\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1)$ as follows:

$$p_F(\tau; \theta) = \prod_{i=0}^{T-1} p_F(\mathbf{x}_{(i+1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \theta).$$

[1] For simiplicity, we will sometimes denote $\mathbf{x}_t$ in place of the state $(\mathbf{x}_t, t)$.

In particular, this induces a marginal density over terminal states:

$$p_F^{\top}(\mathbf{x}_1;\theta)=\int p_F(\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1;\theta)\, d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_{1-\Delta t}. \tag{3.8}$$

Our goal is to find prameters θ of a policy p_F such that the terminal distribution of the forward policy matches the target distribution (*i.e.*, $p_F^{\top} = p_{\text{target}}$):

$$p_F^{\top}(\mathbf{x}_1;\theta) = \frac{R(\mathbf{x}_1)}{Z} \quad \forall \mathbf{x}_1 \in \mathbb{R}^d. \tag{3.9}$$

This objective is intractable due to both the marginalisation in Eq. 3.8 and the unknown Z. Therefore, in practice, auxiliary components are introduced in the optimisation objective to enforce Eq. 3.9.

Importantly, if the forward policy is a Gaussian distribution with mean and variance parametrise parametrise by neural networks that take $\mathbf{x}_t$ and t as input, then learning the policy corresponds to learning the drift $u(\mathbf{x}_t, t;\theta)$ and diffusion $g(\mathbf{x}_t, t;\theta)$ functions of an SDE (Eq. 3.1). This setup is equivalent to fitting a neural SDE. An illustration of sampling using the forward policy is provided in Fig. 3.3.

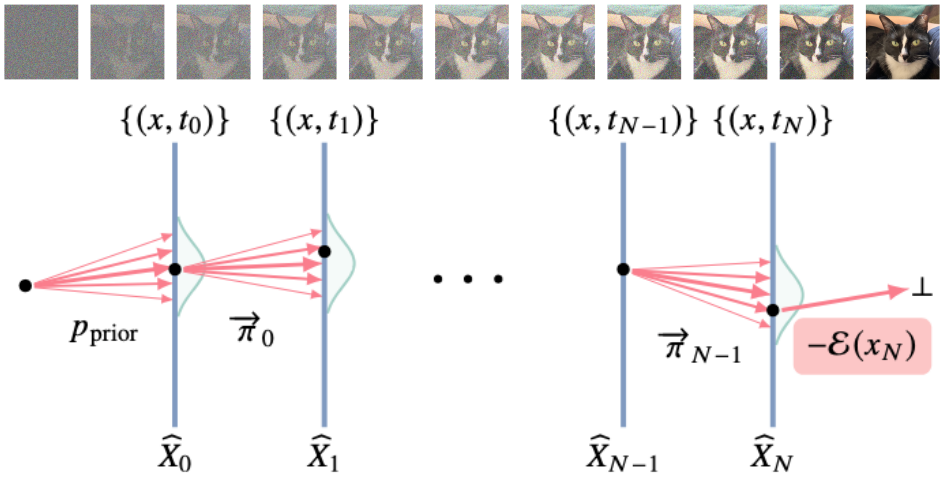


Fig. 3.3 Sampling using the forward policy with continuous-time GFlowNets, we start from the prior distribution at time $t_0 = 0$, and finish in the terminal state.

Backward policy and trajectory balance. Analogously to the forward case, a *backward policy* is defined as a collection of conditional probability densities $p_B(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t;\psi)$ that describe transitions from a state $\mathbf{x}_t$ to its predecessor $\mathbf{x}_{t-\Delta t}$. This backward policy induces a

distribution over complete trajectories τ, conditioned on their terminal state $\mathbf{x}_1$:

$$p_B(\tau \mid \mathbf{x}_1; \psi) = \prod_{i=1}^{T} p_B(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \psi),$$

where we set $p_B(\mathbf{x}_0 \mid \mathbf{x}_{\Delta t}) = 1$ since μ_0 is a point mass.

A key observation is that the forward policy p_F samples from the target distribution *if and only if* there exists a backward policy p_B and a scalar normalisation constant Z_θ such that the *trajectory balance* (TB) condition holds for every complete trajectory $\tau = (\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1)$ (Lahlou et al., 2023):

$$Z_\theta \, p_F(\tau; \theta) = R(\mathbf{x}_1) \, p_B(\tau \mid \mathbf{x}_1; \psi). \tag{3.10}$$

When Eq. 3.10 is satisfied, the parametrised (and learned) normalisation constant Z_θ equals to the true partition function $Z = \int_{\mathbf{x}} R(\mathbf{x}) \, d\mathbf{x}$.
Based on this constraint, we define the *trajectory balance objective* as the squared log-ratio of both sides of Eq. 3.10:

$$\mathcal{L}_{\text{TB}}(\tau; \theta, \psi) = \left(\log \frac{Z_\theta \, p_F(\tau; \theta)}{R(\mathbf{x}_1) \, p_B(\tau \mid \mathbf{x}_1; \psi)} \right)^2. \tag{3.11}$$

Minimising this objective $\mathcal{L}_{\text{TB}}(\tau; \theta, \psi)$ to 0, with respect to the parameters θ and ψ, over trajectories τ sampled from a *training policy* $\pi(\tau)$ leads to satisfaction of the learning condition in Eq. 3.9.
Although Eq. 3.11 can, in principle, be optimised jointly with respect to both the forward and backward policy parameters, it is common practice to *fix* the backward policy — *e.g.*, to a discretised Brownian bridge — and only optimise the forward policy p_F and the partition function estimate Z_θ.

It is worth noting that trajectory balance is not the only objective that enforces the learning constraint in Eq. 3.9. lternative formulations include *e.g.*, SubTB (Madan et al., 2022) and VarGrad (Nüsken and Richter, 2021; Richter et al., 2020). For additional details, we refer the interested reader to the full text of Publication [IV].

Off-policy optimisation. Unlike the reverse KL divergence objective, which requires taking an expectation over the distribution of the current forward process, the trajectory balance objective (Eq. 3.11) supports fully *off-policy* optimisation — *i.e.*, using trajectories sampled from any distribution π with full support. This flexibility allows GFlowNets to explore diverse regions of the state space and discover multiple modes of the target distribution. To promote

exploration of diverse regions and discovery of multiple modes, various reinforcement learning techniques have been proposed, including noisy exploration or tempering (Bengio et al., 2021), replay buffers (Deleu et al., 2022), or Thompson sampling (Rector-Brooks et al., 2023). In the context of continuous GFlowNets, Lahlou et al. (2023); Malkin et al. (2023) suggest adding a small constant to the policy variance when sampling training trajectories to maintain sufficient exploration.
This ability to optimise off-policy is a key advantage of GFlowNets over variational methods such as Path Integral Samplers (PIS), which require on-policy optimisation (Malkin et al., 2023).

Moreover, the trajectory balance objective $\mathcal{L}_{\text{TB}}$ can also be optimised *on-policy* — using trajectories sampled from the forward policy p_F itself. In this case, one obtains an unbiased estimator (up to a constant) of the gradient of the reverse KL divergence (Eq. 3.7) with respect to the parameters of p_F (Malkin et al., 2023; Richter et al., 2020; Zimmermann et al., 2023):

$$\mathbb{E}_{\tau\sim p_F(\tau)}\left[\nabla_{\theta'}\mathcal{L}_{\text{TB}}(\tau;\theta,\psi)\right] = 2\nabla_{\theta'}D_{\text{KL}}(p_F(\tau;\theta)\|p_{\text{target}}(\mathbf{x}_1)p_B(\tau\mid\mathbf{x}_1;\psi)),$$

where $\nabla_{\theta'}$ denotes the gradient with respect only to the prameters of forward policy p_F, *but not* Z_θ.

With the theoretical foundations of continuous-time GFlowNets and neural SDEs for sampling from unnormalised densities in place, we are now prepared to present the original works creating this Ph.D. series. Each of the following publications builds upon and extends the ideas discussed above.

3.3 Improved off-policy training of diffusion samplers [IV]

In this section, we focus on Publication [IV], in which we address the problem of sampling from an unnormalised density or energy function. In particular, we tackle the problem of improved credit assignment for *off-policy diffusion samplers* (continuous generative flow networks). This work systematically evaluates existing inductive biases for off-policy diffusion samplers (continuous generative flow networks), revealing nuanced differences in their practical utility.

Additionally, we introduce a novel and efficient exploration strategy that incorporates *local search* and a *replay buffer*, offering competitive or superior performance while reducing computational demands. This contribution enables more robust training of GFlowNets in continuous spaces and highlights the value of exploration and off-policy learning in diffusion-based samplers.

3.3.1 Motivation

As previously discussed, sampling from complex unnormalised distributions or energy functions — particularly in high-dimensional spaces — remains a fundamental challenge in probabilistic deep learning. In this publication, we aim to enhance the scalability and effectiveness of diffusion-style samplers, including both simulation-based variational approaches and off-policy methods (continuous generative flow networks; GFlowNets).

Our investigation is driven by two key goals. First, we aim to benchmark existing algorithms and credit assignment approaches to determine which methods offer relative improvements for diffusion-style samplers. This involves an empirical evaluation to verify existing claims under a unified experimental setup.
Second, in Publication [IV] we seek to improve exploration and credit assignment in off-policy methods. To this end, we propose a novel exploration mechanism based on local search and a replay buffer.

3.3.2 Content

In the Publication [IV] being one of the papers creating this dissertation series, we focus on the problem of sampling from an unnormalised density without access to true samples Our contribution in this work is twofold — we benchmark several prior methods across a range of target densities, evaluating the efficacy of existing credit assignment strategies, and we introduce a novel exploration strategy for off-policy approaches.

Credit assignment methods. Before introducing our proposed exploration method and presenting the main findings, we begin by reviewing existing credit assignment strategies used in the context of diffusion samplers.

Partial energies and subtrajectory-based learning. One of the existing credit assignment approaches incorporates an inductive bias on the state flow that resembles the geometric interpolation in Máté and Fleuret (2023). This method was proposed in Zhang et al. (2024), which builds upon the work of Lahlou et al. (2023) and employs the SubTB objective rather than TB — allowing operating on subtrajectories rather than complete trajectories.

The specific inductive bias, referred to as *forward-looking* (FL) by Pan et al. (2023), imposes the following form when applied to the state flow function:

$$\log f(\mathbf{x}_t;\theta) = (1-t)\log p_t^{\text{ref}}(\mathbf{x}_t) + t\log R(\mathbf{x}_t) + \text{NN}(\mathbf{x}_t, t;\theta), \tag{3.12}$$

where NN is a neural network and $p_t^{\text{ref}}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; 0, \sigma^2 t\mathbf{I}_d)$ is the marginal density of a Brownian motion with rate σ at $\mathbf{x}_t$.

A common understanding was that the use of the target density $\log R(\mathbf{x}_t) = -\mathcal{E}(\mathbf{x}_t)$ in the state flow function was providing an effective signal, driving the sampler to high-density states at early steps in the trajectory.

Langevin dynamics inductive bias. An alternative strategy is using an inductive bias operating directly on the drift function of the neural SDE, $u(\mathbf{x}_t, t;\theta)$, mimicking the Langevin dynamics on the target distribution, as proposed in Zhang and Chen (2022). In the language of GFlowNets, this corresponds to modifying the mean of the Gaussian transition kernel $p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t;\theta)$.
We refer to this approach as the *Langevin parametrisation* (LP), expressed in the following way:

$$u(\mathbf{x}_t, t;\theta) = \text{NN}_1(\mathbf{x}_t, t;\theta) + \text{NN}_2(t;\theta)\nabla\mathcal{E}(\mathbf{x}_t), \tag{3.13}$$

where NN_1 and NN_2 are neural networks producing a vector and a scalar, respectively. The second term in Eq. 3.13 introduces a scaled gradient of the target energy — effectively mimicking the drift of a Langevin SDE — while the first term provides a learnable correction.

The *Langevin parametrisation* placed on the policies represents a different way of incorporating the reward signal at the intermediate steps of the trajectory, steering the sampler towards low-energy regions. Unlike the *forward-looking*, which influences trajectory distribution indirectly via the state flow, *Langevin parametrisation* injects gradient information

from the target energy directly into the policy. This enables more fine-grained control over the sampler's behaviour, especially in high-dimensional or rugged energy landscapes.

Local search with parallel MALA — improving off-policy training. While the already presented credit assignment methods — *forward-looking* (FL) and *Langevin parametrisation* (LP) — have proven effective, they often incur substantial computational overhead during training. To address this limitation, we propose an alternative approach that requires no additional computational cost per training trajectory.

In particular, our method integrates two components — a local search mechanism inspired by prior work (Hu et al., 2023; Kim et al., 2024b; Zhang et al., 2022) and a replay buffer, previously shown to be effective for GFlowNets in discrete spaces (Deleu et al., 2022; Vemgal et al., 2023). Unlike the previous methods, we do not need to define MCMC kernels via GFlowNet policies. Instead, our method directly applies a parallel *Metropolis-Adjusted Langevin Algorithm* (MALA) (Grenander and Miller, 1994; Roberts and Tweedie, 1996) in the target space. This approach is inherently parallelisable and only needs to be used occasionally, minimising computational cost.

For a detailed algorithmic description of local search, we refer the reader to Algorithm 2 presented in Publication [IV]. Below, we provide an intuitive summary of our local search procedure.
Initially, we draw M candidate samples from the sampler (forward policy):

$$\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\} \sim p_F^{\top}(\cdot).$$

Then, we run parallel MALA across M independent chains over K transitions, starting from these candidate samples. After discarding the first $K_{\text{burn-in}}$ transitions as burn-in, the accepted samples are added to a local search buffer, denoted $\mathcal{D}_{\text{LS}}$. We occasionally update the buffer using MALA step and replay samples from the buffer to minimise the computational demands of iterative local search.

After introducing the local search and replay buffer procedure, the next question is: **how they can be used to train diffusion samplers?**

When using the replay buffer, we begin by drawing a sample $\mathbf{x}$ from $\mathcal{D}_{\text{LS}}$. Then, we sample a trajectory τ leading to $\mathbf{x}$ using the backward process p_B, and finally perform a gradient update on the trajectory balance objective associated with τ.

When training with local search guidance, we alternate between two types of training steps — *i.e.*, updates on *forward trajectories*, which use either on-policy or exploratory forward

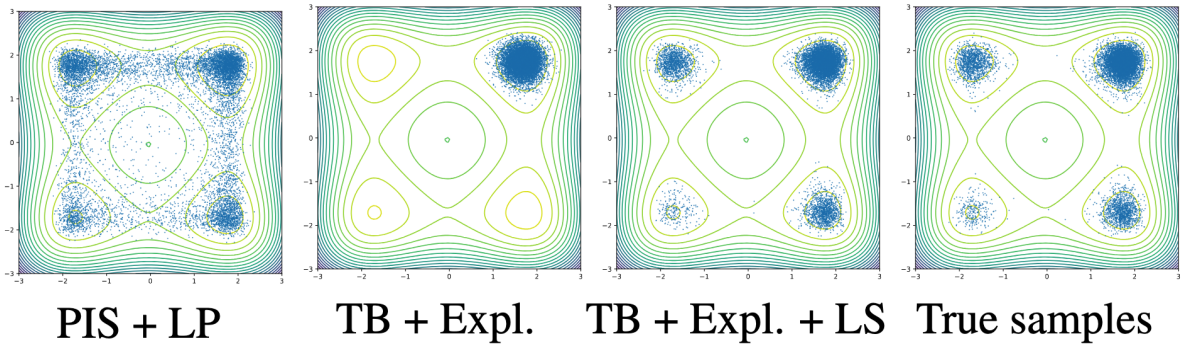


Fig. 3.4 Two-dimensional projections of **Manywell** samples from models trained using different algorithms. Our proposed replay buffer with local search effectively prevents mode collapse, leading to broader coverage of the distribution.

sampling, and *backward trajectories*, which leverage samples from the local search buffer $\mathcal{D}_{\text{LS}}$ to guide training. This combined procedure encourages the sampler to explore both diverse regions of the state space and low-energy (*i.e.*, high-probability) configurations, thus preventing mode collapse.

The effectiveness of our local search method is supported by the empirical results on standard benchmark energy functions.For visual intuition, we provide an examples of the samples on the **Manywell** density (Fig. 3.4). For a detailed description of training procedure using both local search and replay buffer, we refer the reader to Algorithm 1 presented in Publication [IV].

Results and findings. To empirically evaluate both prior credit assignment methods and our proposed local search with replay buffer, we conducted an extensive series of experiments. Specifically, we evaluated performance on two tasks: sampling from unnormalised densities and sampling from the Bayesian posterior of a VAE.

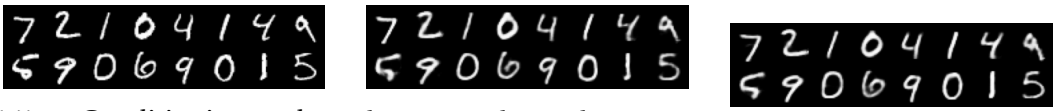
As shown in Table 3.1, the usage of *local search* (LS), and, optionally, *Langevin parametrisation* (LP), consistently yields the best or near-best results in terms of the ELBO gap. We observed similar results under the *2-Wasserstein* metric, and for those results, we refer the reader to the full text of Publication [IV].

When consider sampling from the Bayesian posterior of a VAE, our models effectively and almost fully recover the true posterior distribution. This is visually demonstrated in Fig. 3.5, where samples drawn from the learned posterior are decoded by a fixed VAE. These decoded samples are nearly indistinguishable from direct reconstructions of the ground-truth data, illustrating the high fidelity of the posterior approximation. Quantitative results supporting this claim can be found in Publication [IV].

Table 3.1 Log-partition function estimation errors for unconditional modelling tasks (mean and standard deviation over 5 runs). The four groups of models are: MCMC-based samplers, simulation-driven variational methods, baseline GFlowNet methods using different learning objectives, and methods augmented with Langevin parametrisation and local search.

Energy →	25GMM ($d=2$)		Funnel ($d=10$)		Manywell ($d=32$)		LGCP ($d=1600$)	
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\log \hat{Z}$	$\log \hat{Z}^{\text{RW}}$
SMC	$0.569_{\pm 0.010}$		$0.561_{\pm 0.801}$		$14.99_{\pm 1.078}$		N/A	
GGNS (Lemos et al., 2023)	$0.016_{\pm 0.042}$		$0.033_{\pm 0.173}$		$0.292_{\pm 0.454}$		N/A	
DIS (Berner et al., 2022)	$1.125_{\pm 0.056}$	$0.986_{\pm 0.011}$	$0.839_{\pm 0.169}$	$0.093_{\pm 0.038}$	$10.52_{\pm 1.02}$	$3.05_{\pm 0.46}$	$299.83_{\pm 0.67}$	$361.15_{\pm 6.48}$
DDS (Vargas et al., 2023)	$1.760_{\pm 0.08}$	$0.746_{\pm 0.389}$	$0.424_{\pm 0.049}$	$0.206_{\pm 0.033}$	$7.36_{\pm 2.43}$	$0.23_{\pm 0.05}$	$471.64_{\pm 1.20}$	$489.30_{\pm 0.62}$
PIS (Zhang and Chen, 2022)	$1.769_{\pm 0.104}$	$1.274_{\pm 0.218}$	$0.534_{\pm 0.008}$	$0.262_{\pm 0.008}$	$3.85_{\pm 0.03}$	$2.69_{\pm 0.04}$	$381.14_{\pm 1.42}$	$414.42_{\pm 2.06}$
+ LP (Zhang and Chen, 2022)	$1.799_{\pm 0.051}$	$0.225_{\pm 0.583}$	$0.587_{\pm 0.012}$	$0.285_{\pm 0.044}$	$13.19_{\pm 0.82}$	$0.07_{\pm 0.85}$	$471.45_{\pm 0.18}$	$487.82_{\pm 2.26}$
TB (Lahlou et al., 2023)	$1.176_{\pm 0.109}$	$1.071_{\pm 0.112}$	$0.690_{\pm 0.018}$	$0.239_{\pm 0.192}$	$4.01_{\pm 0.04}$	$2.67_{\pm 0.02}$	$336.70_{\pm 56.22}$	$379.50_{\pm 49.99}$
TB + Expl. (Lahlou et al., 2023)	$0.560_{\pm 0.302}$	$0.422_{\pm 0.320}$	$0.749_{\pm 0.015}$	$0.226_{\pm 0.138}$	$4.01_{\pm 0.05}$	$2.68_{\pm 0.06}$	$346.10_{\pm 55.54}$	$389.21_{\pm 44.13}$
VarGrad + Expl.	$0.615_{\pm 0.241}$	$0.487_{\pm 0.250}$	$0.642_{\pm 0.010}$	$0.250_{\pm 0.112}$	$4.01_{\pm 0.05}$	$2.69_{\pm 0.06}$	$370.37_{\pm 0.26}$	$410.37_{\pm 6.70}$
FL-SubTB	$1.127_{\pm 0.010}$	$1.020_{\pm 0.010}$	$0.527_{\pm 0.011}$	$0.182_{\pm 0.142}$	$3.98_{\pm 0.07}$	$2.72_{\pm 0.05}$	$365.20_{\pm 6.08}$	$402.65_{\pm 8.36}$
+ LP (Zhang et al., 2024)	$0.209_{\pm 0.025}$	$0.011_{\pm 0.024}$	$0.563_{\pm 0.021}$	$0.155_{\pm 0.317}$	$4.23_{\pm 0.12}$	$2.66_{\pm 0.22}$	$465.44_{\pm 1.26}$	$483.90_{\pm 1.95}$
TB + Expl. + LS (*ours*)	$0.171_{\pm 0.013}$	$0.004_{\pm 0.011}$	$0.653_{\pm 0.025}$	$0.285_{\pm 0.099}$	$4.57_{\pm 2.13}$	$0.19_{\pm 0.29}$	$384.90_{\pm 0.83}$	$419.55_{\pm 2.14}$
TB + Expl. + LP (*ours*)	$0.206_{\pm 0.018}$	$0.011_{\pm 0.010}$	$0.666_{\pm 0.615}$	$0.051_{\pm 0.616}$	$7.46_{\pm 1.74}$	$1.06_{\pm 1.11}$	$452.82_{\pm 1.50}$	$477.62_{\pm 1.79}$
TB + Expl. + LP + LS (*ours*)	$0.190_{\pm 0.013}$	$0.007_{\pm 0.011}$	$0.768_{\pm 0.052}$	$0.264_{\pm 0.063}$	$4.68_{\pm 0.49}$	$0.07_{\pm 0.17}$	$471.14_{\pm 0.25}$	$489.03_{\pm 1.38}$
VarGrad + Expl. + LP + LS (*ours*)	$0.207_{\pm 0.016}$	$0.015_{\pm 0.015}$	$0.920_{\pm 0.118}$	$0.256_{\pm 0.037}$	$4.11_{\pm 0.45}$	$0.02_{\pm 0.21}$	$468.65_{\pm 0.63}$	$487.34_{\pm 1.34}$

Highlight : mean indistinguishable from best in column with $p < 0.05$ under one-sided Welch unpaired t-test.



((a)) Conditioning data (MNIST test set)

((b)) VarGrad + Expl. + LP (samples decoded)

((c)) VAE reconstruction

Fig. 3.5 Our sampler (VarGrad + Expl. + LP) is conditioned by a subset of never-seen data coming from the ground truth distribution (**left**). The conditional samples were then decoded by the the fixed VAE (**middle**). For the comparison, we show the reconstruction of the real data by VAE (**right**). We observed that the decoded samples are visually very similar to the reconstructions making these two pictures almost indistinguishable. Both, decoded samples and reconstruction, are more blurry than the ground truth data, which is caused by a limited capacity of the VAE's latent space.

We also empirically validated several hypotheses from prior work. In particular, we found that SubTB and FL-SubTB underperform compared to the standard TB objective combined with LP. In addition, FL-SubTB introduces significant computational overhead. We observe that LP consistently improves performance across all evaluated settings. Finally, the usage of decaying noise schedule substantially benefits the quality of exploration and sample diversity, as illustrated in Fig 3.6.

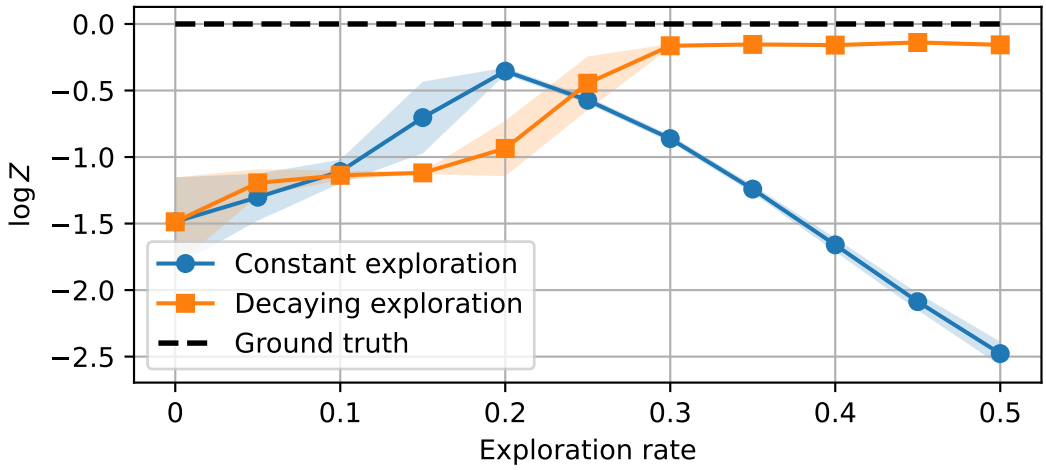


Fig. 3.6 Effect of exploration variance on models trained with TB on the **25GMM** energy. Exploration promotes mode discovery, but should be decayed over time to optimally allocate the modelling power to high-likelihood trajectories.

3.3.3 Contribution and impact

This paper (Sendera et al., 2024a) was developed during my research internship at Mila – Quebec AI Institute, where I collaborated primarily with Prof. Yoshua Bengio and Dr. Nikolay Malkin on improving the scalability of continuous GFlowNets. I am the first author and the primary contributor to this work.

My contributions include the development of a novel off-policy training method for diffusion-style samplers, the implementation of both the proposed approach and relevant baselines, and the design and execution of the experimental setup. I also helped in conceptualising the method and was involved in preparing the manuscript.

The paper was accepted to NeurIPS 2024, a top-tier (CORE A*) machine learning conference, and was well-received by the community. As of August 2025, it has been cited 54 times and is gaining traction within the diffusion sampling community (Behjoo and Chertkov, 2025; Kim et al., 2024a; Lee et al., 2025). In particular, several subsequent works have built upon this paper in contexts such as sampling from Bayesian posteriors (Bengio et al., 2025; Mittal et al., 2025), steering diffusion priors (Bartoldson et al., 2025; Silva et al., 2024; Venkatraman et al., 2024b), and evaluating reasoning capabilities (Yu et al., 2024).

3.4 From discrete-time policies to continuous-time diffusion samplers: asymptotic equivalences and faster training [V]

In Publication [V], we study the problem of training *neural stochastic differential equations* (neural SDEs) or diffusion models to sample from unnormalised target densities, without access to any data samples. While the previously presented work (*see* Sec. §3.3) focused on improving the performance of diffusion-style samplers, this study shifts attention to analysing the theoretical equivalences between two families of approaches in the continuous-time limit: differentiable simulation and off-policy *reinforcement learning* (RL).

Our contributions in this paper are twofold: (1) theoretical findings on asymptotic behaviour of the objectives for discrete-time policies as approaching the continuous-time processes, and (2) empirical validation of various discretisation schemes during training of these discrete-time policies. In particular, we show the effectiveness and efficiency of *nonuniform* discretisation schemes when comparing to the typical *uniform* ones.

3.4.1 Motivation

While previously presented work (*i.e.*, Publication [IV]) focused on developing better off-policy diffusion-style samplers, in this publication we consider two distinct but related approaches using: (1) differentiable simulations, or (2) off-policy reinforcement learning. These methods are often treated separately in the literature, but we show they are asymptotically equivalent in the continuous-time limit.

In particular, we prove that these two paradigms become equivalent under infinitesimal discretisation, thereby connecting entropic RL methods (such as GFlowNets) to continuous-time objects — *partial differential equations* (PDEs) and *path-space measures*. This theoretical insight motivates the design of more effective and coarser time discretisation schemes that allow for maintaining strong empirical performance of diffusion samplers while significantly reducing their training computational costs.

3.4.2 Content

Recent research has proposed using diffusion models — originally known from data-driven generative modelling — for sampling from unnormalised densities without access to data. The general idea is to model generation as the reverse of a diffusion (*noising*) process, formulated in either discrete or continuous-time (*see* Fig. 3.7). Specifically, we begin with a simple prior distribution p_{prior} (*e.g.*, Gaussian) and gradually transport samples by a sequence of stochastic transitions toward a complex target distribution p_{target}.

Note that, in the classical data-driven setting, one can learn such models via score matching (Song et al., 2021a) using access to samples from p_{target}. However, this is not possible in our setup.

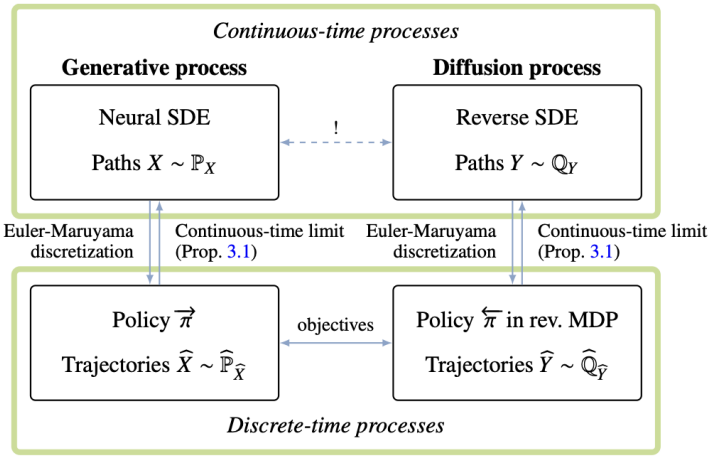


Fig. 3.7 The problem of enforcing continuous-time forward and reverse processes to determine the same path-space measure is approximated by matching distributions over discrete-time trajectories.

We begin by analysing the differences between continuous-time and discrete-time generative processes.

Continuous-time and discrete-time generative processes. **In the continuous-time setting**, the generative process is formulated relatively simple, as a stochastic differential equation (SDE), which defines a continuous-time object. The formulation assumes an initial distribution p_{prior} and a diffusion coefficient $\sigma(t)$. Overall, the generative dynamics is described by:

$$\mathrm{d}X_t = \overrightarrow{\mu}(X_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \quad X_0 \sim p_{\text{prior}}, \tag{3.14}$$

where $\overrightarrow{\mu}$ denotes the drift function and W_t is a standard Wiener process (*Brownian motion*). When the drift $\overrightarrow{\mu}$ is parametrised with a neural network, this process (Eq. 3.14) is referred to as a *neural SDE* (Kidger et al., 2021a; Song et al., 2021b; Tzen and Raginsky, 2019).
The objective is to optimise this neural SDE such that the marginal distribution of X_1 closely approximates the target distribution p_{target}. Thus, the neural SDE effectively maps p_{prior} to p_{target} via a continuous-time stochastic process.

In the discrete-time setting, the generative process is described instead as a Markov chain, where the dynamics is governed by transition kernels $\overrightarrow{\pi}_n(\widehat{X}_{n+1} \mid \widehat{X}_n)$ for $n = 0, \ldots, N-1$. The considered Markov chain has the initial distribution, $\widehat{X}_0 \sim p_{\text{prior}}$, and the learned

transition probabilities $\overrightarrow{\pi}_n$ so that the final distribution $\widehat{X}_N$ approximates p_{target}. This formulation underlies recent models such as stochastic normalising flows (Hagemann et al., 2023), and can be seen as a special case of (continuous) generative flow networks (GFlowNets; Bengio et al., 2021; Lahlou et al., 2023).

Despite the structural differences between continuous and discrete-time processes, both approaches are typically trained in a similar way. Specifically, by minimising a bound on a divergence between distributions over trajectories — those induced by the generative process and those defined by the target distribution combined with a noising process. However, the specific training objectives can differ. Some methods rely on *differentiable simulation* of the generative process (Kidger et al., 2021b; Li et al., 2020; Zhang and Chen, 2022), while others use *off-policy reinforcement learning* and optimise objectives defined over trajectories obtained via exploration (Malkin et al., 2023; Nüsken and Richter, 2021). These objectives can be further categorised into *global* (involving the complete trajectory) and *local* (focused on single transitions).
We present typical objective functions and illustrate their relationships in Fig. 3.8.

Theoretical findings. We know that any SDE induces the discrete-time policy when discretised using the specific numerical integration scheme, *e.g.*, a widely used Euler–Maruyama method. Conversely, any discrete-time policy converges to a continuous-time process in the limit of infinitesimal timesteps (Kloeden and Platen, 1992). In the described publication, our goal was to characterise and formalise the relationships between training objectives for discrete-time and continuous-time generative processes in the limit of infinitesimal timesteps.
In particular, we established formal links between RL-based methods and concepts from stochastic control and dynamic measure transport. We show and rigorously prove the following results:

- **Global objectives** in discrete time converge to objectives that minimise divergences between *path space measures* induced by the forward and reverse processes in continuous time.

- **Local constraints**, as enforced by GFlowNet training objectives, asymptotically approach *partial differential equations* that govern the time evolution of the marginal densities of the SDE under the generative and noising processes.

We refer the reader to the full text of Publication [V] for the complete theoretical formalism and detailed proofs of these results.

An overview of the discrete-time policies and continuous-time SDEs — as well as the relationships between their training objectives — is presented in Fig. 3.8. The vertical arrows in the figure represent our formal theoretical contributions, establishing asymptotic equivalences between corresponding objectives.

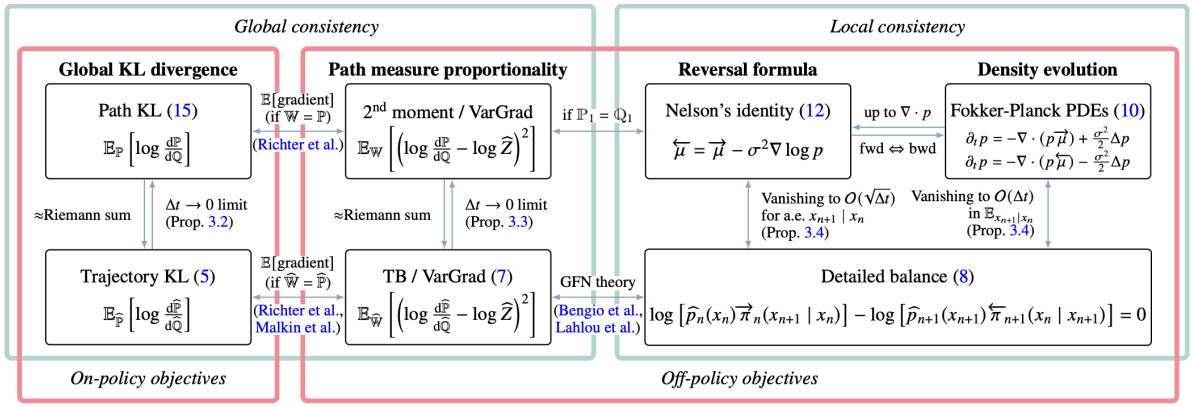


Fig. 3.8 Training objectives for neural SDEs (**top row**) and their approximations via objectives for discrete-time policies (**bottom row**). On-policy objectives minimise divergences by differentiating through the SDE integration, while off-policy objectives enforce local or global consistency constraints. Our results explain the asymptotic behaviour of discrete-time objectives as the time discretisation becomes finer.

Coarser discretisation. When derived the theoretical contributions on the relationships between discrete-time policies and continuous-time processes, we particularly focus on discretisation schemes. We hypothesised that carefully choosing integration steps during training could improve both sampling efficiency and quality under a fixed computational budget.

This hypothesis is grounded in the observation that training with shorter trajectories — obtained by coarser time discretisations — allows the use of time-local objectives without the computationally expensive bootstrapping techniques that are necessary when training with long trajectories.

To empirically evaluate this hypothesis, we conducted an extensive set of experiments oacross standard sampling benchmarks, including a diverse set of unnormalised densities and Bayesian posteriors. In particular, we examined a variety of training objectives and inductive biases, like Langevin parametrisation. Moreover, we evaluated different discretisation schemes, including *uniform*, *random*, and *equidistant* schemes, as well as varying the number of timesteps.

Our experiments empirically confirmed that discrete-time policies converge to the true partition function, $\log Z$, more accurately as the number of integration steps increases.

Moreover, the *nonuniform* discretisation scheme consistently led to faster convergence across all evaluated settings. For instance, Fig. 3.9 illustrates that the ELBO gap decreases significantly faster under *nonuniform* schemes compared to uniform ones.
Additionally, we compared training efficiency for different discretisation strategies, as shown in Fig. 3.10. These results demonstrate that random (*nonuniform*) discretisation achieves a better trade-off between computational cost and ELBO quality.

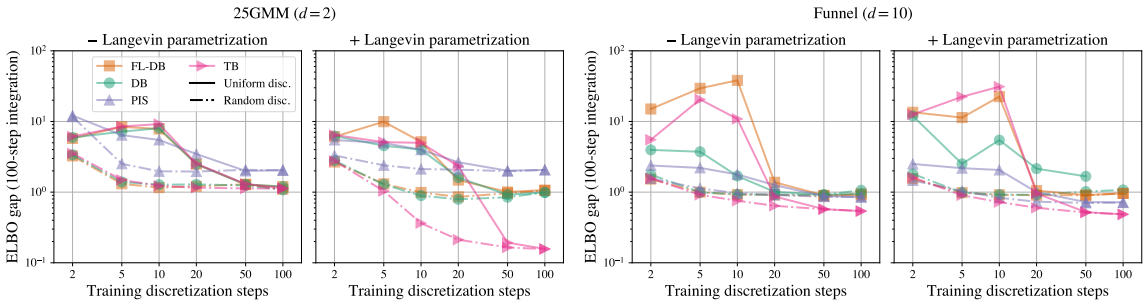


Fig. 3.9 Difference between the true log-partition function $\log Z$ and the ELBO as a function of N_{train}, evaluated with a fixed 100-step uniform integration.

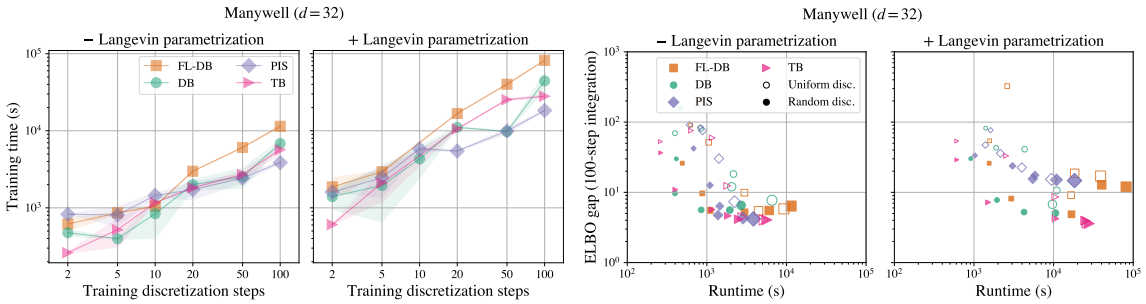


Fig. 3.10 **Left:** Training time for 25k iterations on the **Manywell** benchmark as a function of N_{train}, with mean and standard deviation over 3 runs (note the log-log scale). **Right:** Runtime vs ELBO gap, showing that **random** discretisation provides a superior balance between performance and speed.

To further assess the impact of coarse training discretisation, we evaluated models trained with $N_{\text{train}} = 10$ using varying numbers of integration steps at test time (N_{eval}). As shown in Fig. 3.11, samplers trained with nonuniform schemes generalise better to finer evaluation discretisations, resulting in lower ELBO gaps.

Finally, we found that training with *nonuniform* time discretisations — much coarser than those used at inference — achieves comparable performance to state-of-the-art methods, while requiring only a fraction of the typical computational cost.

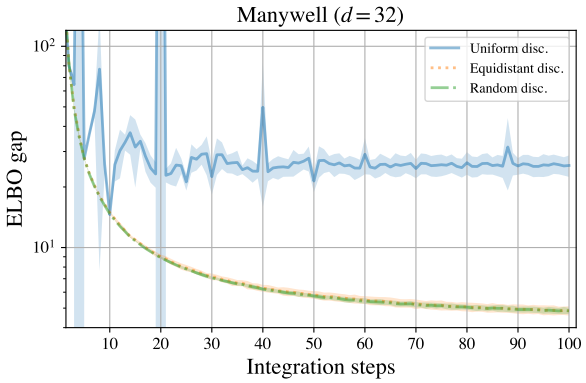


Fig. 3.11 ELBO gaps for models trained with various discretisation schemes and $N_{\text{train}} = 10$, then evaluated with different numbers of integration steps N_{eval}.

3.4.3 Contribution and impact

This paper (Berner et al., 2025) was developed during my research internship at Mila – Quebec AI Institute, in extensive collaboration with Dr. Nikolay Malkin. The project aimed to establish formal connections between discrete-time policies in MDPs and continuous-time generative objects, neural SDEs. I am a co-first author of this paper and contributed equally with Dr. Julius Berner and Dr. Lorenz Richter.

My primary responsibilities included implementing the proposed methods, designing and executing experiments, and analysing the results. I was also involved in the conceptual development of the framework and contributed to writing the manuscript.

The paper is currently available as an arXiv preprint and is under review at a leading deep learning journal. As of August 2025, it has been cited 10 times and has received positive attention within the generative modelling (Venkatraman et al., 2025a) and sampling communities (Gritsaev et al., 2025; Kim et al., 2025; Woo et al., 2025).

4

Principled adaptation: on generalisation via Bayesian reasoning

4.1 Research scope

On the importance of Bayesian reasoning in modern AI

Finally, in the last group of the presented publications, we consider what is arguably the most fundamental topic in probabilistic deep learning — namely, the *Bayesian perspective* (Wang and Yeung, 2020; Wilson and Izmailov, 2020). While some challenges of Bayesian inference have already appeared in earlier chapters, this section focuses on them directly.

Let us begin with an intuition for Bayes' theorem, followed by its application to deep learning, and the key arguments for the fundamental role that Bayesian reasoning plays in the development of modern artificial intelligence (Bengio et al., 2024; Papamarkou et al., 2024).

The probabilistic approach to statistical reasoning is built upon a deceptively simple, yet profoundly influential, result by Rev. Thomas Bayes, formulated posthumously in the 1760s. *Bayes' theorem* (Bayes, 1763) lies at the heart of philosophical debates about the nature of probability — whether it reflects objective frequencies of events or subjective beliefs (Carnap, 1945, 1962; Fisher, 1958; Reichenbach, 1971). In its classical, frequentist view, the theorem

appears as a simple formula for conditional probability. However, its meaning is much deeper — it describes how we should update our beliefs in light of new evidence: it expresses the posterior distribution in terms of three others probability density functions.
More importantly, Bayesian inference offers a principled framework for updating beliefs as new data becomes available and accommodating uncertainty in model's parameters. This simple idea, revolutionised the scientific world, from Bayesian statistics in the beginning of the 20th-century to the *Bayesian revolution* in its second half (Jaynes, 2003). Today, Bayesian thinking continuous to influence many disciplines, including artificial intelligence (Barber, 2012; Ghahramani, 2015).

In this context, *Bayesian deep learning* (BDL) — which combines Bayesian principles with deep learning — has emerged as an important, though still not mainstream, research area (Papamarkou et al., 2024). Bayes' theorem allows for probabilistic understanding and interpretation of the parameters, even in overparametrised models as deep neural networks, which incorporate more uncertainty rather than less. Contrary to a common misconception, it is not that deep learning makes uncertainty obsolete; in fact, the opposite is true. In an era dominated by massive, overparameterised models, such as *large language models* (LLMs), we argue that deep learning needs Bayesian principles more than ever (Bengio et al., 2024; Papamarkou et al., 2024).

So why does Bayesian deep learning matter?
Arguably, its most important contribution lies in uncertainty quantification (Hüllermeier and Waegeman, 2021; Kendall and Gal, 2017). In the age of agentic LLMs, understanding model behaviour (Kadavath et al., 2022), reasoning about the confidence of predictions (Rudner et al., 2022a; Tran et al., 2022b), and aligning AI decisions with human values are essential. Many researchers in the AI safety community emphasise that BDL is one of the few viable frameworks for building trustworthy and safe AI systems — capable of recognising when they don't know, and preventing potentially harmful decisions (Bengio et al., 2025, 2024).
Moreover, BDL naturally supports adaptability to new and evolving domains, which is crucial for RL agents and AI systems playing with non-stationary data distributions. It also enables data-efficient methods, which are beneficial for settings such as few-shot (Patacchiola et al., 2020; Yoon et al., 2018), active (Gal et al., 2017; Margatina et al., 2023), and continual learning (Nguyen et al., 2018; Rudner et al., 2022b). Bayes' theorem enables also incorporating prior domain knowledge and structured inductive biases into systems, which enhances interpretability and awareness of model misspecification. Finally, Bayesian frameworks also promote scientific-style reasoning, offering a coherent probabilistic lens for evaluating competing hypotheses and refining models in light of new data.
While many of these advantages have been known for years, the urgency of adopting

Bayesian reasoning has become far more acute. As AI systems become more powerful, autonomous, and widely deployed, particularly in safety-critical contexts, the ability to quantify and reason under uncertainty is no longer optional. Addressing these challenges through scalable and principled Bayesian methods will not only advance deep learning but may prove essential to ensuring its safe and responsible integration into society (Bengio et al., 2025, 2024).

On challenges and open problems

Despite its potential impact and years of research, Bayesian deep learning (BDL) still faces a number of fundamental challenges preventing its practical application to modern large-scale deep learning. The primary one among these is the issue of scalability, especially when dealing with the modern architectures comprising billions of parameters. Achieving time and memory efficiency is the holy grail of Bayesian inference (Papamarkou et al., 2024).

Scalability issue can be seen from different perspectives — one of them is the incurred computational cost (Izmailov et al., 2021). This results in the usage of Gaussian Processes (GPs) as the preferred models for demanding scientific applications (Griffiths et al., 2023) — even if GPs already incorporate significant cost ($\mathcal{O}(n^3)$). Nevertheless, GPs and GP-based hybrid models retain their value, especially in few-shot learning, where we operate on limited data (Patacchiola et al., 2020; Sendera et al., 2021c).

More generally, the two dominant bottlenecks in scalable Bayesian deep learning are *posterior computation*, and *prior specification*. The overall goal is to develop Bayesian methods that are both efficient and competitive with standard deterministic approaches, especially in an era of foundation models. These large models have shifted the priorities of the Bayesian deep learning community towards the question on how to merge Bayesian approaches with foundation models. In this context, BDL becomes essential not just for uncertainty quantification, but also for steering and Bayesian fine-tuning of these powerful models. All of the considered perspectives on scalability result in a number of research directions with their own challenges.

Considering posterior approximation, we might notice that classic methods like the Laplace approximation are still widely used due to their simplicity (Daxberger et al., 2021a). However, they typically find only single modes within posteriors, and thus provide a poor approximation for the multi-modal and complex posteriors that arise in overparameterised models like neural networks.

A more flexible and currently very active line of research is *variational inference* (VI). VI methods approximate the posterior by optimising a lower bound on the marginal likelihood. Among these methods, recent work has explored a promising, amortised version of VI, *e.g.*,

realised via diffusion-style samplers, which show promise for large models due to their scalability and ability to model complex posteriors.
Another way of approximating Bayesian posteriors is through ensembles. These provide a practical way to capture uncertainty via model diversity, but require training multiple models in parallel, making them infeasible at foundation model scale.
Recently, much of the community's attention has moved toward another perspective on the scalability issue — *i.e.*, improving posterior sampling algorithms, with the goal of achieving asymptotic convergence to the true posterior while reducing computational overhead. Again, more and more studies consider (amortised) variational inference approaches, *e.g.*, the ones based on diffusion-style samplers (Malkin et al., 2023; Sendera et al., 2024a). These methods represent a shift away from traditional *Markov Chain Monte Carlo* approaches, which, while theoretically grounded, are often prohibitively expensive for deep learning-scale problems. Moreover, they are considered as the only way to incorporate Bayesian perspective into modern neural networks.

The second major challenge concerns the specification of priors — as model dimensionality grows, designing meaningful priors over parameter space becomes increasingly difficult. Yet, we know from recent theoretical insights that the prior over functions, rather than parameters, governs generalisation (Wilson and Izmailov, 2020). Moreover, the prior over parameters induces the prior over functions. This makes the design of function-space priors an essential but underdeveloped component of modern BDL.
One promising approach involves building Bayesian neural networks whose induced function-space priors mimic those of Gaussian Processes, which naturally encode distributions over functions. While conceptually elegant, making this approach efficient and scalable remains a major challenge.

Given these challenges, the BDL community has identified several promising research directions that aim to bridge the gap between Bayesian theory and modern deep learning practice. These include developing novel posterior sampling algorithms, deep kernel processes, or hybrid Bayesian approaches, as well as using Bayesian methods in transfer and continual learning.
However, the most important research direction is the integration of large-scale foundation models with Bayesian inference — for example, treating LLMs as prior distributions and developing mechanisms to update or sample from their posteriors. Such a perspective could open entirely new paradigms for interpretable, controllable, and trustworthy AI.

The role of Bayesian perspective in the works comprising this dissertation

Having established the growing importance of the Bayesian perspective in modern AI and outlined the key research challenges in this area, we now highlight how the works comprising this dissertation contribute to the field of Bayesian deep learning (BDL) from multiple angles.

Among the various challenges discussed earlier, perhaps the most pressing is the scalability of posterior inference. This topic has already been addressed in detail in Chapter §3. To briefly summarise, Publications [IV] and [V] propose solutions based on amortised variational inference, realised by diffusion-style samplers. Specifically, we introduce a novel off-policy training procedure, benchmark the effect of various inductive biases, and demonstrate how to train diffusion samplers efficiently using coarser, non-uniform discretisation schemes. Moreover, we establish a theoretical connection between discrete-time Markov processes and continuous-time neural SDEs.

In the current chapter, we begin by considering the meta-learning setting, especially focusing on few-shot learning. Publication [VI] introduces a novel deep kernel method that combines a GP with learnable projection and kernel functions, integrated with a continuous-time normalising flow mapping. This hybrid architecture allows GPs to model locally non-Gaussian posteriors, thus justifying the name Non-Gaussian Gaussian Processes, discussed in detail in Section §4.3.

Continuing in Section §4.4, we further explore the advantages of Bayesian modelling in few-shot learning. Publications [VII] and [VIII] present new ways to incorporate the Hypernetwork paradigm into few-shot problems. While [VII] focuses on the predictive capabilities of Hypernetworks (Ha et al., 2017), [VIII] extends this to a Bayesian formulation, enabling principled uncertainty quantification while moving to the novel task.

Finally, in Section §4.5, we tackle the second major challenge outlined earlier: prior specification. In Publication [IX], we propose a method for enforcing GP-like behaviour in a single hidden layer Bayesian neural network, thereby creating a function-space prior. Our framework is both theoretically justified and computationally more efficient than existing alternatives, making it a promising direction for future research in BDL.

Beyond the scope of this series of publications, we have also conducted work on sampling from Bayesian posteriors with foundation models as priors, including vision and language diffusion models. Moreover, we used to work on solving *Bayesian inverse problems* in scientific domains such as astrophysics, which is briefly discussed in Chapter §5.

We now proceed to Section §4.2, where we outline the Bayesian deep learning preliminaries used in the publications presented throughout this chapter. This section may be omitted by readers already familiar with the relevant background. Following that, we present and discuss the individual works that comprise this dissertation.

4.2 Background

By now, we have developed some high-level intuition about the fundamental importance of Bayes' theorem — not only for scientific reasoning in general, but also for the core of deep learning, particularly its probabilistic variant. However, an attentive reader may have noticed that although Bayesian ideas have appeared in previous chapters, we have not yet formally introduced them or explicitly worked within a Bayesian framework. In contrast to earlier publications presented in this dissertation, the works discussed in this chapter require some background in commonly used Bayesian methods.

It is important to emphasise that the publications comprising this dissertation address a range of essential topics in probabilistic deep learning. While they incorporate the Bayesian perspective, they are not limited to it. Consequently, we will present only the key concepts and mechanisms necessary to understand the contributions of the included publications, without attempting to provide a comprehensive introduction to Bayesian statistics. For readers seeking a deeper and more detailed exposition, we recommend excellent resources such as Gelman et al. (1995), and other deep learning-focused publications like: Barber (2012); Bishop (2006); Murphy (2012).

On Bayesian and frequentist perspectives on probability

Let us begin with a simple, yet non-trivial observation: there are two main ways of interpreting probability, based on different philosophical understandings of its nature. These perspectives lead to distinct practical approaches. For instance, it is relatively straightforward to say that the probability of a fair coin landing heads is 50%, but this statement can be interpreted in two, fundamentally different ways.

The first view, known as the **frequentist** perspective, predominates in traditional probability courses. It defines probability as the long-run frequency of events that can occur multiple times. In the coin example, if we flip the coin a very large number of times, we expect it to land heads roughly 50% of the time, and tails the other 50%. While this interpretation is intuitive and widely used, it has notable limitations — such as its reliance on repeatable events. What if we are dealing with an event that cannot be repeated or observed frequently?

This is where the **Bayesian** interpretation offers a compelling alternative. In Bayesian perspective, probability quantifies our uncertainty or ignorance about an outcome. This interpretation is closely linked to ideas from information theory. Returning to the coin example, the Bayesian approach interprets the 50% probability as a reflection of our belief that heads and tails are equally likely on the next toss, based on current information. Crucially, this view allows us to reason about one-off events that cannot be repeated. For example,

consider estimating the probability that humans will land on Mars by the year 2050. This event will either happen once or not at all — it is not repeatable. However, the Bayesian approach allows us to quantify our uncertainty about this event. Moreover, based on our assessment of this event's probability, we can determine optimal actions.

The strong emphasis that the Bayesian framework places on uncertainty also leads to a useful distinction between two types of uncertainty. The first, called **epistemic uncertainty**, represents our lack of knowledge about the hidden causes or mechanisms generating our data; it can, in principle, be reduced with more information or data. The second, known as **aleatoric uncertainty**, captures the inherent noise or randomness in the data itself, and be eliminated, even with additional data.

On Bayes' theorem

In a standard probability course grounded in the frequentist view, Bayes' theorem may appear as a relatively simple formula for computing conditional probabilities. However, the theorem, published posthumously in the 1760s (Bayes, 1763), laid the groundwork for a fundamentally different approach to statistical reasoning. In this sense, it is far more than a basic identity—it serves as the foundation for probabilistic reasoning in the Bayesian paradigm. At its core, this theorem articulates the relationship between evidence and beliefs — specifically, how we should update our beliefs given new evidence.
Let us now formally state Bayes' theorem in a manner most relevant to machine learning:

Theorem 1 (Bayes' theorem, 1763)**.** *Let $\theta \in \mathbb{R}^n$ denote a parameter, $\mathcal{D}$ be the evidence (e.g., training dataset), and $p(\theta|\mathcal{D})$ the posterior probability density function of θ given evidence $\mathcal{D}$. Then, the posterior distribution $p(\theta|\mathcal{D})$ can be expressed in terms of three other probability density functions as follows:*

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\,p(\theta)}{p(\mathcal{D})}. \tag{4.1}$$

The individual terms (probability density functions) in this expression are defined as:

- *the **prior** $p(\theta)$: the distribution over θ before observing any evidence $\mathcal{D}$;*
- *the **likelihood** $p(\mathcal{D}|\theta)$: the probability of observing data $\mathcal{D}$ given a specific parameter θ;*
- *the **marginal likelihood** or **evidence** $p(\mathcal{D})$: the probability of observing the data under all possible parameter values.*

At first glance, Theorem 1 appears to have a straightforward mathematical form, suggesting an equally straightforward application to real-world problems. However, a closer inspection reveals that each term presents its own challenges. Let us examine them in turn.

We begin with the marginal likelihood $p(\mathcal{D})$, which serves as a normalising constant in Bayes' theorem. Computing it involves evaluating the following integral:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)\, p(\theta)\, d\theta. \tag{4.2}$$

This integral is often analytically intractable or computationally expensive, particularly in high-dimensional parameter spaces. The evidence term is crucial as it serves as a normalization constant in Bayes' theorem, meaning accurate probability estimation is impossible without properly evaluating the posterior. Efficient sampling methods are therefore necessary to approximate this term. Although techniques such as Markov Chain Monte Carlo (MCMC) are considered the gold standard, they suffer from slow mixing times and are practically unscalable to high-dimensional problems.

The challenges of Bayesian modelling extend beyond computational difficulties. Another significant challenge in Bayesian modelling lies in the choice of the prior $p(\theta)$. A poorly chosen prior can lead to model misspecification. A common approach is to use simple priors — such as zero-mean Gaussian distributions — which simplify computations and, under certain additional assumptions, allow closed-form solutions. However, these are typically *uninformative priors*, reflecting a lack of prior knowledge about the phenomenon. There are ways to incorporate *informative priors*, for example by designing specific kernels in Gaussian Processes, where kernel hyperparameters encode domain knowledge. But how can we transfer such informative priors into more complex Bayesian models, like Bayesian Neural Networks (BNNs)? One of the publications comprising this dissertation and presented in this chapter (Publication [IX]) addresses this very issue — transferring informative priors from Gaussian Processes into BNNs.

The likelihood term $p(\mathcal{D}|\theta)$ is generally the least problematic, but still requires a modelling choice: what distribution should the likelihood follow? This choice depends on the nature of the problem itself, the Bayesian model being used, and computational considerations—ideally, the likelihood should admit a tractable or closed-form expression.

Now, let us turn to the process of **Bayesian inference**, which is equally challenging. Full Bayesian inference involves computing the predictive distribution for a new input x given as the integral over the posterior:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\theta, \mathbf{x})\, p(\theta|\mathcal{D})\, d\theta, \tag{4.3}$$

which is rarely tractable without simplifying assumptions or approximations.

These challenges explain why practical applications of Bayes' theorem often rely on various approximations—especially in the context of deep learning models, which involve

extremely high-dimensional parameter spaces. Nevertheless, the insight offered by Bayes' theorem makes such efforts worthwhile.

There exists a wide variety of approaches and frameworks that enable models to be fully Bayesian, or at least incorporate Bayesian elements to some extent. Covering all of them is beyond the scope of this introductory section. Instead, we will briefly introduce those methods directly employed in the publications comprising this dissertation and presented in this chapter — namely, Gaussian Processes, Bayesian Neural Networks, and Variational Inference.

On Gaussian processes

Gaussian processes (GPs) represent one of the most widely adopted Bayesian frameworks in contemporary deep learning solutions, particularly valued for their ability to yield closed-form posteriors under certain design constraints. They are especially prominent in meta-learning and continual learning applications, and serve as powerful function-space priors. Among the works comprising this dissertation, two extensively leverage GPs, *i.e.,* the Publication [VI], which applies GPs to few-shot learning scenarios, and Publication [IX], which explores transferring GP behaviour to Bayesian neural networks (BNNs). To understand these contributions, we now formally introduce Gaussian processes.

Definition 4.2.1 (Gaussian process (GP))**.** *A Gaussian process (GP) is a collection of random variables such that the joint distribution of every finite subset of random variables from this collection follows a multivariate Gaussian distribution (Quinonero-Candela and Rasmussen, 2005).*
We denote a GP as $f(\cdot) \sim \mathbf{GP}(\mu(\cdot), k(\cdot,\cdot))$*, where* $\mu(\mathbf{x})$ *and* $k(\mathbf{x},\mathbf{x}')$ *represent the mean and covariance functions, respectively.*

In the absence of prior knowledge, common practice is to select μ as the zero constant function, and k as a kernel function. Popular kernel choices include the Linear kernel, Radial Basis Function (RBF) kernel, Spectral Mixture kernel (Wilson and Adams, 2013), or Matérn kernel.
It is worth noting that an alternative approach to constructing a kernel function involves taking an inner product in a feature space induced by a mapping $\psi : X \to V$:

$$k(\mathbf{x},\mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle_V. \tag{4.4}$$

This formulation enables direct modelling of ψ using a neural network, creating a bridge between traditional kernel methods and deep learning architectures.

This construction enables GPs to model probability distributions over functions and provides the posterior distributions in a closed-form. Consider a standard regression problem:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i, \text{ for } i = 1, \ldots, m, \tag{4.5}$$

where ϵ_i represents *i.i.d.* noise variables with independent $\mathcal{N}(0, \sigma^2)$ distributions.

Let $\mathbf{X}$ denote the matrix of all samples $\mathbf{x}_i$ and $\mathbf{y}$ the corresponding vector of all targets $\mathbf{y}_i$. Assuming that $f(\cdot) \sim \mathbf{GP}(0, k(\cdot,\cdot))$, we obtain:

$$\mathbf{y}|\mathbf{X} \sim \mathcal{N}(0, \mathbf{K} + \sigma^2\mathbf{I}), \tag{4.6}$$

where $k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Inference over unknown samples is elegantly accomplished through conditioning the normal distribution. Given the training data $(\mathbf{X}, \mathbf{y})$ and test data $(\mathbf{X}_*, \mathbf{y}_*)$, the distribution of $\mathbf{y}_*$ is also Gaussian (Rasmussen and Williams, 2006):

$$\mathbf{y}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_* \sim \mathcal{N}(\mu_*, \mathbf{K}_*), \tag{4.7}$$

where:

$$\boldsymbol{\mu}_* = \mathbf{K}(\mathbf{X}_*, \mathbf{X}) \left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y},$$

$$\mathbf{K}_* = \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) + \sigma^2\mathbf{I} - \mathbf{K}(\mathbf{X}_*, \mathbf{X}) \left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\right)^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}_*).$$

On Bayesian neural networks

Bayesian neural networks (BNNs) represent another realisation of the Bayesian paradigm, emerging as the Bayesian counterpart to deep neural networks (DNNs) that incorporate epistemic uncertainty in a principled manner. While definitions vary slightly across the literature, a common understanding is that **BNNs are stochastic artificial neural networks trained using Bayesian inference techniques**.

A natural way to introduce BNNs is by first reviewing a conventional DNN and then highlighting the key differences. The primary goal of any deep neural network is to represent an arbitrary function $y = \Phi(\mathbf{x})$, where $\mathbf{x}$ represents D-dimensional input data, y denotes the target values, and $\Phi : \mathbb{R}^D \to \mathbb{R}$. Consider the simplest case — a feedforward network composed of an input layer l_0, a sequence of $N-1$ hidden layers $l_1, l_2, \ldots, l_{N-1}$, and an output layer l_N. This architecture encompasses $N+1$ layers in total, where each layer l_i is a linear transformation potentially followed by a nonlinear operation σ_i, commonly referred

to as an *activation function*. Formally, a standard DNN can be expressed as:

$$\begin{aligned} l_0 &= \mathbf{x}, \\ l_i &= \sigma_i\left(\mathbf{W}_i l_{i-1} + \mathbf{b}_i\right) \quad i = 1, \ldots, N; \\ y &= l_N. \end{aligned} \tag{4.8}$$

The set of learnable parameters ($\boldsymbol{\theta}$) comprises all network connection weights $\mathbf{W}$ and biases $\mathbf{b}$, such that $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$.

In standard deep learning, we optimise these learnable parameters using the backpropagation algorithm to minimise a specified objective function. This objective function typically takes some form of the *log likelihood*. This corresponds to either **maximum likelihood estimation (MLE)** or, when incorporating additional regularisation, **maximum a posteriori (MAP)** estimation (Jospin et al., 2022).

Bayesian neural networks differ from conventional DNNs in two fundamental ways. First, rather than operating with deterministic parameter values, BNNs employ stochastic weights to simulate all possible models $\boldsymbol{\theta}$ with their associated prior distribution $p(\cdot)$ (*see* Fig. 4.1). This stochastic formulation improves uncertainty estimation, as a BNN can be conceptualised as an ensemble of numerous DNNs sharing the same architecture. Due to this stochastic nature, the prediction process of a BNN can be described as:

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\cdot), \\ y &= \Phi_{\boldsymbol{\theta}}(\mathbf{x}) + \epsilon, \end{aligned} \tag{4.9}$$

where ϵ represents observation noise, included because $\Phi_{\boldsymbol{\theta}}$ serves only as an approximation. The resulting prediction is a distribution over outputs, reflecting uncertainty over both the model and the data.

The second distinctive feature of BNNs lies in their utilisation of Bayesian inference (Equation 4.3) for training — to compute the posterior distribution over the parameters, $p(\boldsymbol{\theta} \mid \mathcal{D})$. However, exact inference is generally intractable due to the complexity of neural networks. Approximate inference techniques such as *Markov Chain Monte Carlo* (MCMC) or variational inference are therefore used in practice. Unfortunately, these methods tend to be computationally intensive, which remains a key challenge in scaling BNNs. Addressing this issue is one of the motivations behind *diffusion-based off-policy samplers*, a topic explored in Chapter §3.

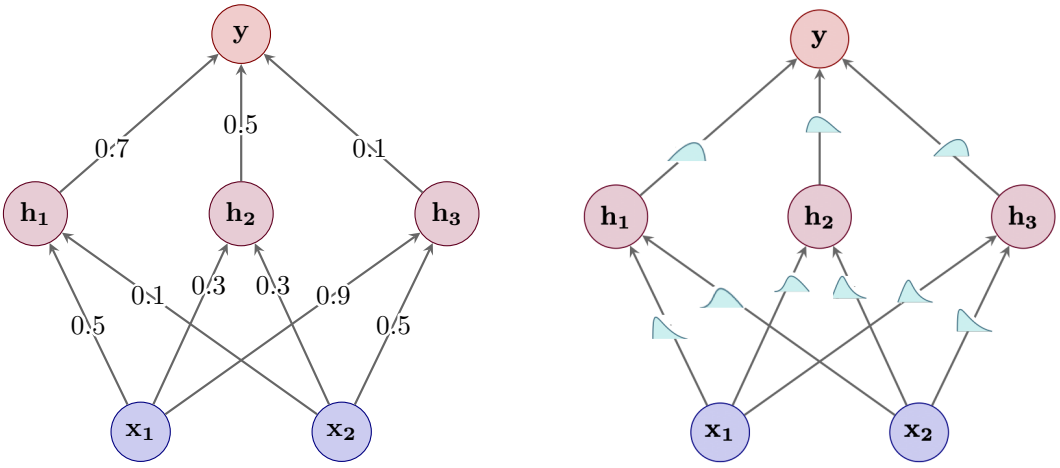


Fig. 4.1 Comparison of the deterministic parameters of DNNs (**left**) and stochastic realisation of BNNs (**right**). In the DNN, the input random variables x_1 and x_2 are processes via deterministic weights, resulting in the same output value y. Whereas, the BNN models the whole distributions over weights, resulting in the whole family of deep models sharing the same architecture. *Image adapted from:* Daxberger et al. (2021b)

On variational inference

As discussed previously, classical methods for training Bayesian models typically require complex sampling approaches that scale poorly to higher-dimensional problems. This raises a fundamental question: how can we determine a posterior distribution over a Bayesian model (*e.g.*, BNN) with thousands or millions of parameters?

The most prevalent solution to this challenge is the **variational Bayes** approach. The core idea of **variational inference** (VI) is to cast inference as an optimisation problem, rather than relying on sampling from the intractable posterior.

Concretely, suppose we are given an intractable probability distribution p. Instead of sampling from it directly, variational inference introduces a family of tractable distributions $\mathcal{Q}$ and attempts to find a member $q^* \in \mathcal{Q}$ that is as close as possible to the true posterior via optimisation procedure. Once identified, this optimal distribution q^* serves as an approximate solution. Then q^* can be used in place of the exact posterior for downstream inference.

To determine q^*, we must define both the parametric family $\mathcal{Q}$ and select an appropriate objective function for optimisation. Following an information-theoretic approach, we can employ the *Kullback-Leibler divergence* (KL divergence; Kullback and Leibler, 1951), which quantifies the dissimilarity between the approximate posterior $q(\mathbf{z})$ and the true posterior

$p(\mathbf{z} \mid \mathbf{x})$. We present KL divergence here in its variational inference formulation:

$$D_{\mathrm{KL}}(q||p) = \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right]. \tag{4.10}$$

However, we cannot compute this expression (Equation 4.10) directly, since it depends on the intractable marginal likelihood $\log p(\mathbf{x})$. Please, remember that dependance on marginal likelihood represents the very challenge that motivated our consideration of variational inference. Let us rewrite KL divergence as:

$$D_{\mathrm{KL}}(q||p) = \mathbb{E}\left[\log q(\mathbf{z})\right] - \mathbb{E}\left[\log p(\mathbf{z},\mathbf{x})\right] + \log p(\mathbf{x}). \tag{4.11}$$

Instead, we can optimise an alternative objective. By applying Jensen's inequality (Jensen, 1906), we derive a lower bound on the marginal log-likelihood:

$$\log p(\mathbf{x}) \geq \mathbb{E}\left[\log p(\mathbf{z},\mathbf{x})\right] - \mathbb{E}\left[\log q(\mathbf{z})\right]. \tag{4.12}$$

This inequality (RHS expression) defines the **evidence lower bound** (ELBO), the central objective function in variational inference:

$$\mathrm{ELBO}(q) = \mathbb{E}\left[\log p(\mathbf{z},\mathbf{x})\right] - \mathbb{E}\left[\log q(\mathbf{z})\right]. \tag{4.13}$$

Maximising the ELBO is equivalent to minimising the KL divergence between $q(\mathbf{z})$ and the true posterior (up to the constant $\log p(\mathbf{x})$). Hence, the optimisation problem becomes:

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathbb{E}q(\mathbf{z})\left[\log q(\mathbf{z})\right] - \mathbb{E}q(\mathbf{z})\left[\log p(\mathbf{z},\mathbf{x})\right]. \tag{4.14}$$

Variational inference (VI) encompasses a broad family of methods (*e.g.*, mean-field VI, factorised VI, black-box VI), which differ in the choice of variational family $\mathcal{Q}$ and optimisation technique. Standard VI techniques typically learn a separate set of parameters for each latent variable or data point, which can be inefficient in large-scale problems. Further scalability can be achieved through **amortised variational inference** methods. While conventional VI learns separate parametric distributions for each latent variable, amortized VI instead learns a common inference function mapping each observation to its corresponding latent variable's variational posterior ($q(\mathbf{z} \mid \mathbf{x})$), thereby enhancing scalability. One example of amortised VI is continuous generative flow networks, presented in Chapter §3.

An important detail in variational inference is the asymmetry of the KL divergence, since $D_{\mathrm{KL}}(q||p) \neq D_{\mathrm{KL}}(p||q)$. These two directions lead to different approximations — the **reverse KL**, which is mode-seeking and the **forward KL**, which is mass-covering. For instance, the previously mentioned diffusion samplers utilise the TB objective, $\mathcal{L}_{\mathrm{TB}}$, whose gradient equates to that of the reverse KL. However, amortised VI introduces a constraint: we must ensure that the optimal solution q^* lies within the chosen amortised variational family $\mathcal{Q}_{\mathrm{amortised}}$ (Margossian and Blei, 2024). If the true posterior cannot be well approximated by this family, performance may degrade.

Despite these limitations, (amortised) variational inference is currently considered as the primary approach for scaling the Bayesian paradigm to address contemporary challenges in deep learning. These challenges actively seeking Bayesian solutions include AI safety, uncertainty quantification in Large Language Models, and AI applications in scientific domains (*e.g.*, drug design).

On current challenges and future directions

Bayesian deep learning has proven to be particularly well-suited for scenarios such as **continual learning** and **meta-learning**, where access to data is extremely limited or where the underlying data distribution shifts over time. In such settings, rapid adaptation and principled uncertainty estimation are critical. Empirical support for this adaptability is provided by the results in Publications [VI], [VII] and [VIII], which form a core part of this dissertation and are discussed in Sections §4.3 and §4.4.

Despite the growing success of Bayesian methods in deep learning, several open challenges remain and continue to be the focus of active research. These include:

- **Scalability of inference procedures** and the development of improved **posterior approximation techniques**, which are addressed in Publications [IV] and [V], discussed in Chapter §3.
- **Model misspecification** and the challenge of selecting well-informed and flexible **prior distributions**, which are tackled in Publication [IX], presented in Section §4.5.

These research directions are critical for making Bayesian deep learning practical and reliable in real-world applications, especially where data is scarce, distributional shifts are common, or interpretability and uncertainty are essential. As the field continues to evolve, improving scalability, robustness, and flexibility in Bayesian inference will remain key objectives.

4.3 Non-Gaussian Gaussian processes for few-shot regression [VI]

This section discusses Publication [VI], which addresses one of the central problems in meta-learning: *few-shot regression*. In particular, we propose a reformulation of the classical Bayesian framework of Gaussian Process (GP) to enable faster and more flexible adaptation to novel tasks based on only a handful of examples. The key innovation lies in our **Non-Gaussian Gaussian Processes** (NGGPs) framework, which integrates ODE-based invertible transformations into the GP framework. By leveraging continuous-time normalising flows to map data into a Gaussian distribution, this method enables the modelling of complex distributions through *locally non-Gaussian posteriors*.

The resulting NGGP not only improves over traditional GP-based methods like DKT but also achieves state-of-the-art performance across a range of few-shot regression benchmarks, both in terms of predictive accuracy and probabilistic calibration.

4.3.1 Motivation

The presented work is motivated by the goal of improving Bayesian methods — specifically *Gaussian Processes* (GPs) — in the context of *meta-learning*. Meta-learning (Bengio et al., 1991; Schmidhuber, 1987) has been defined in various ways; here, we follow the terminology established by Hospedales et al. (2021), in which meta-learning involves training an **inner model** to solve a distribution of tasks in a way that improves a **meta-objective**, such as learning speed or adaptability, typically in an end-to-end fashion.

Among the different meta-learning settings, we consider the few-shot learning scenario, where a model is trained to solve an upcoming supervised task with only a few labelled examples available. This setting is particularly well suited for GP-based approaches, such as Deep Kernel Transfer (DKT) (Patacchiola et al., 2020), which benefit from the probabilistic structure imposed by GPs and their ability to compute posterior distributions in closed form from limited data.

However, this strength also comes with significant limitations. The generalisation power of GPs comes at the cost of reduced flexibility when the target distributions are complex — *e.g.*, exhibiting high skewness, heavy tails, or multiple modes. Moreover, standard GPs typically assume high similarity between tasks, a condition that is rarely met in real-world scenarios where task distributions may vary over time, as in heteroscedastic regression. These challenges extend to broader settings such as multi-modal learning or, more generally, multi-label regression.

4.3.2 Content

To better understand the consider problem setting and the method proposed in Publication [VI], we first describe the *few-shot learning* scenario in more detail. This will provide the necessary context for grasping the proposed solution. Note that we do not elaborate on the general meta-learning paradigm, as it is out of scope of the main focus of the presented series of publications. For a comprehensive introduction to meta-learning, we refer the interested reader to Hospedales et al. (2021) and Vanschoren (2018). The preliminaries on continuous-time normalising flows were already introduced in Section §2.2.

Few-shot learning. Few-shot learning is one of the most predominent problems within meta-learning. We begin with a high-level description of few-shot learning as typically presented in the meta-learning literature. For clarity, we first introduce the classification case and then briefly explain its adaptation to the regression tasks.

Assume we are given a *meta-dataset* of tasks $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$, where each task $\mathcal{T}_n$ consists of a *support set* $\mathcal{S}_n$ and a *query set* $\mathcal{Q}_n$. The construction of these sets depends on the considered task. In general, a support set contains L labelled input-output pairs with the equal class distribution:

$$\mathcal{S} = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^{L}. \tag{4.15}$$

If each class is represented by a single example and $L = K$, where K is the number of classes, we refer to it as a *one-shot* scenario. In *few-shot* settings, each class has a few (typically, 2, 5, or 10) representatives in the support set $\mathcal{S}$. Similarly, the query set (also referred to as the *target set*) is defined as:

$$\mathcal{Q} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M} \tag{4.16}$$

where M is usually one order of magnitude larger than K.

For simplicity, we group the support and query sets into a single task: $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\}$. During training, both support and query contain labelled pairs $(\mathbf{x}, y)$, and the objective is to predict the targets y in $\mathcal{Q}$ given only the examples from $\mathcal{S}$. At test time, we are given a previously unseen task $\mathcal{T}_* = (\mathcal{S}_*, \mathcal{Q}_*)$, and the model must infer the targets for the unlabeled query examples (in $\mathcal{Q}_*$). The general overview of few-shot learning is presented in Fig. 4.2.

While the above description assumes a classification setting, it extends naturally to **few-shot regression**. The only difference is that the outputs y are continuous scalar values, while the inputs $\mathbf{X}$ are now vectors. The overall framework remains the same.

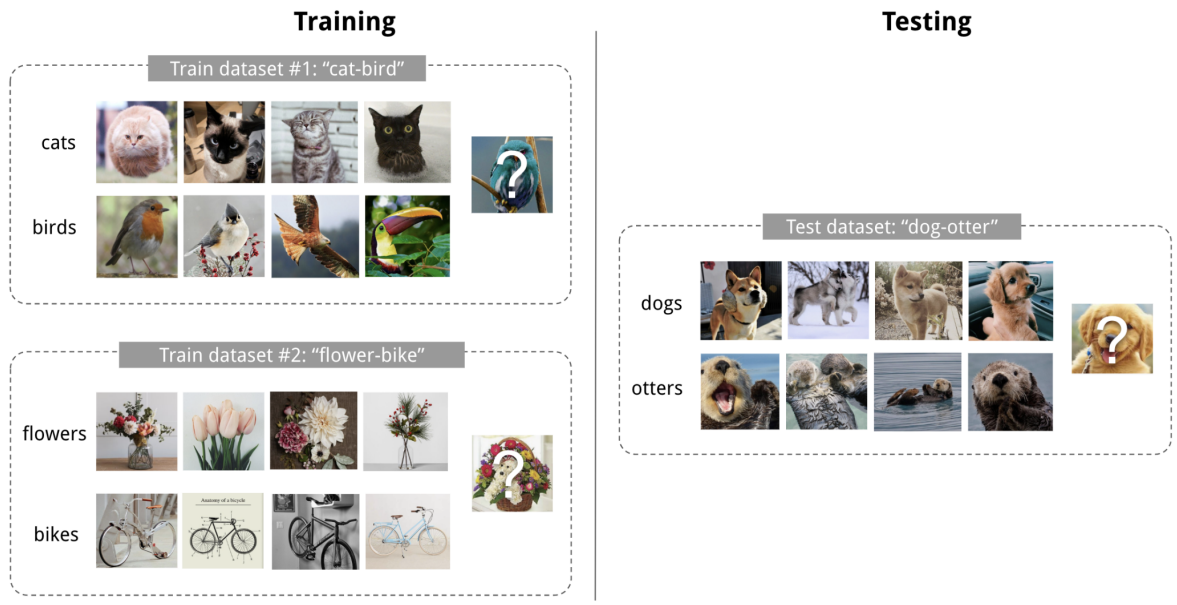


Fig. 4.2 In few-shot learning scenario, we consider training tasks (**left**) and testing tasks (**right**). All of them consist of support and query sets, and our aim is to classify the query samples. Note that during training we have access to the true labels of the query samples. *Image source:* https://lilianweng.github.io.

Deep kernel transfer. Bayesian deep learning methods are particularly effective in meta-learning and few-shot settings due to the underlying probabilistic (Bayesian) structure they impose into the considered problems. For example, Gaussian processes assume a Gaussian distribution over functions, which allows them to generalise from a very small number of support points to find the whole family of functions that fullfil the desired behaviour.

This makes GPs a natural fit for few-shot learning. A prominent example is the **deep kernel transfer** (DKT) method (Patacchiola et al., 2020), which jointly adapts GP parameters and an additional neural network using gradient-based optimisation. The neural network, referred to as the *backbone*, performs representation learning by projecting the inputs into a feature space where GPs can operate more effectively.

Despite these advantages, existing GP-based methods suffer from several limitations. In particular, they lack flexibility when dealing with complex (non-Gaussian) data distributions or when the assumption of task similarity (homoscedasticity) is violated. All of these assumptions are usually not met in the real-world scenarios, such as multi-modal distributions or multi-label regression tasks.

Non-Gaussian Gaussian processes (NGGPs). In Publication [VI], we address the limitations of standard GP-based methods by introducing a novel probabilistic framework that

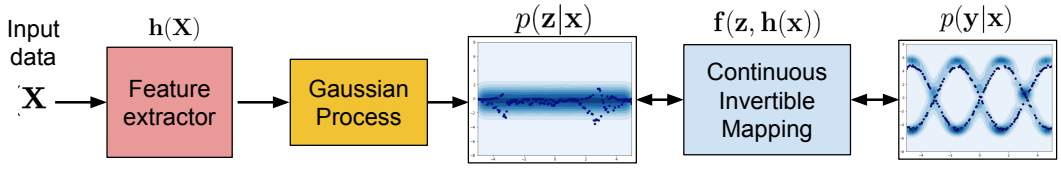


Fig. 4.3 The general idea of NGGP. The input data are embedded by the feature extractor $\mathbf{h}(\cdot)$ and then used to create a kernel for the GP. Next, the output of the GP is adjusted using an invertible mapping $\mathbf{f}(\cdot)$ which is conditioned on the output of the feature extractor, which allows to model complex distributions of the target values $\mathbf{y}$.

enhances the expressiveness of Gaussian processes through locally non-Gaussian posteriors. Our approach, called **non-Gaussian Gaussian processes** (NGGPs), couples the GP model with invertible ODE-based mappings, which process the marginals of multivariate random variables. Thereby, NGGPs is able to capture complex, non-Gaussian behaviours in the predictive distribution.

Rather than simply stacking a continuous-time normalising flow on top of a GP to model the multidimensional distribution over $\mathbf{y}$, NGGPs take a more precise approach — we learn an invertible ODE-based mapping using both the input $\mathbf{x}_i$ and the task-specific features, and create the specific mapping for each datapoint. This allows us to model the marginals flexibly, while the GP captures dependencies across dimensions.

The overall idea of NGGPs is illustrated in Fig. 4.3. Given input observations $\mathbf{x}_i$ stored in the data matrix $\mathbf{X}$, we first apply a feature extractor $\mathbf{h}(\cdot)$ (often referred to as the **backbone**) to obtain the latent representations of $\mathbf{X}$. These latent features are used to create a distinct invertible CNF-based mapping for the specific task. These features are then used both as the GP's input and to condition the CNF-based invertible mapping $\mathbf{f}(\cdot)$. Finally, the GP is used to model the distribution of the latent variable $\mathbf{z}$.

The question is how to enforce such behaviour? Let $\mathbf{h}_{\boldsymbol{\phi}}(\cdot)$ denote the feature extractor and $k_{\boldsymbol{\theta}}(\cdot,\cdot)$ the kernel function, parametrised by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ respectively. For input $\mathbf{X}$ of dimensionality D and corresponding latent variable $\mathbf{z}$, the marginal log-probability of the GP is given by:

$$\log p(\mathbf{z}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{z}^{\mathrm{T}}\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{z} - \frac{1}{2}\log\left|\mathbf{K}+\sigma^2\mathbf{I}\right| - \frac{D}{2}\log(2\pi), \tag{4.17}$$

where $k_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_i), \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_j))$ and $\mathbf{K}$ is the resulting kernel matrix.

Then, we apply a continuous normalising flow to map the GP latent output $\mathbf{z}$ to the observed data $\mathbf{y}$. Since we use ODE-based flows, the transformation's log-density correction

follows Equation 2.11, allowing to compute the log marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\beta}) = \log p(\mathbf{z}|\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}}{\partial \mathbf{z}(t)}\right) dt, \tag{4.18}$$

where $\mathbf{g}_{\boldsymbol{\beta}}$ is the velocity field of the ODE and $\boldsymbol{\beta}$ denotes its parameters.

In NGGP, this transformation is applied *independently* to each output (y) dimension $d = 1,\ldots,D$, while a GP captures the dependency across the variables. Though, the final marginal log-likelihood becomes:

$$\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\beta}) = \log p(\mathbf{z}^{\mathbf{h}}|\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta}) - \sum_{d=1}^{D} \int_{t_0}^{t_1} \frac{\partial g_{\boldsymbol{\beta}}}{\partial z_d(t)} dt, \tag{4.19}$$

where $\mathbf{z}^{\mathbf{h}} = \mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}))$ represents the element-wise inverse transformation of the observed output.

The ODE-based transformation is parametrised using the FFJORD model (Grathwohl et al., 2019), and its forward and reverse transformations are computed via Equations 2.8 and 2.9, respectively.

At inference time, we are interested in evaluating the posterior predictive distribution: $p(\mathbf{y}_*|\mathbf{X}_*,\mathbf{y},\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\beta})$, *i.e.*, the probability of D_**-dimensional* outputs $\mathbf{y}_*$ at test inputs $\mathbf{X}_*$ given access to the training data $(\mathbf{X},\mathbf{y})$.

Due to the structure of the model — the invertible transformation operating independently on the outputs — this posterior can also be computed in closed form:

$$\log p(\mathbf{y}_*|\mathbf{X}_*,\mathbf{y},\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\beta}) = \log p(\mathbf{z}_*^{\mathbf{h}}|\mathbf{X}_*,\mathbf{z}^{\mathbf{h}},\mathbf{X},\boldsymbol{\phi},\boldsymbol{\theta}) - \sum_{d=1}^{D_*} \int_{t_0}^{t_1} \frac{\partial g_{\boldsymbol{\beta}}}{\partial z_d(t)} dt, \tag{4.20}$$

where $\mathbf{z}_*^{\mathbf{h}} = f_{\boldsymbol{\beta}}^{-1}(\mathbf{y}_*, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}_*))$, $\mathbf{z}^{\mathbf{h}} = f_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}))$ are the inverse transformations of test and train data, and $p(\mathbf{z}_*^{\mathbf{h}}|\cdot)$ is the standard GP predictive posterior.

Training of NGGPs follows the general paradigm introduced in the DKT framework (Patacchiola et al., 2020), enabling end-to-end optimisation of all model parameters ($\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and $\boldsymbol{\beta}$) using standard gradient-based methods. For the detailed description, including algorithmic implementation, we refer the reader to the full text of Publication [VI].

Results. In order to appropriately evaluate the proposed NGGP method, we conduct an extensive set of experiments across both few-shot and classical regression tasks. We compare NGGP against a diverse selection of meta-learning approaches, including — but not limited to — Deep Kernel Transfer (DKT Patacchiola et al., 2020), Model-Agnostic Meta-

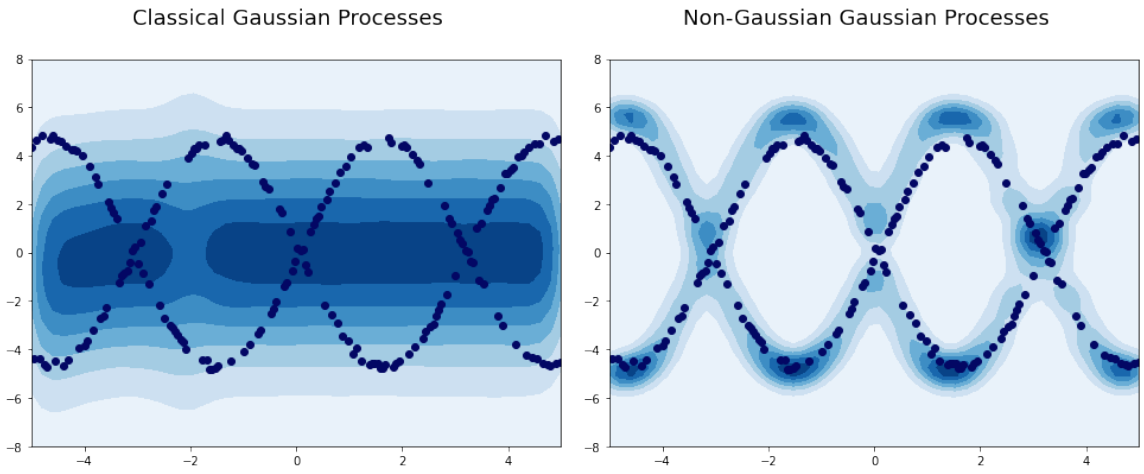


Fig. 4.4 Comparison of NGGPs and classical GPs on 1-d multifunction problem. Classical GPs (**left**) assume underlying Gaussian posterior, and cannot model such complex distributions, whereas NGGPs (**right**) utilise a flexible ODE-based mapping to enforce locally non-Gaussian posteriors.

Learning (MAML; Finn et al., 2017), and Fine-Tuning (FT). To provide a broader context for evaluating NGGP's advantages, we also include standard regression baselines such as classical Gaussian Processes (GPs; Rasmussen, 2003) and Conditional Neural Processes (CNPs; Garnelo et al., 2018).

In all experimental settings, we employ two complementary evaluation metrics: **negative log-likelihood** (NLL) as a probabilistic measure, and **mean squared error** (MSE) as a sample-based measure. This combination offers a more comprehensive view of the models' predictive performances. Finally, we observe that NGGP consistently achieves state-of-the-art performance in most conditions.

Before presenting the quantitative results, we first aim to build some intuition for the advantages of NGGP over DKT, and to highlight when NGGP becomes particularly useful. Fig. 4.4 shows a multi-function task based on *sine* functions. In this setting, NGGP accurately captures the underlying structure of the data, whereas DKT fails. This result is not surprising, since DKT is based on classical GPs, which assume Gaussian marginals and thus tend to collapse to the mean when faced with non-Gaussian outputs. In contrast, NGGP employs flexible ODE-based invertible mappings that allow it to model rich, non-Gaussian distributions effectively — even in low-data regimes.

In Publication [VI], we provide a complete set of quantitative results on various few-shot regression benchmarks including sines, object pose prediction, power, NASDAQ, and EEG datasets. In this description of the presented series of publications, we focus on one

representative task: head-pose trajectory. This problem involves predicting the trajectory of head motion given a sequence of head-pose images. Table 4.1 presents results for both in-range and out-of-range test settings. Interestingly, while NGGP shows comparable MSE to other methods, it clearly outperforms them in terms of NLL, demonstrating its superior ability to model uncertainty and capture complex output distributions.

Table 4.1 Quantitative results for Queen Mary University of London for *in-range* and *out-of-range* settings, taking into account *NLL* and *MSE* measures.

Method	**in-range**		**out-of-range**	
	MSE	*NLL*	*MSE*	*NLL*
Feature Transfer/1	0.25±0.04	-	0.20±0.01	-
Feature Transfer/100	0.22±0.03	-	0.18±0.01	-
MAML (1 step)	0.21±0.01	-	0.18±0.02	-
DKT + RBF	0.12±0.04	0.13±0.14	0.14±0.03	0.71±0.48
DKT + Spectral	0.10±0.01	0.03±0.13	0.07±0.05	0.00±0.09
DKT + NN Linear	0.04±0.03	-0.12±0.12	0.12±0.05	0.30±0.51
NGGP + NN Linear	0.02±0.02	-0.47±0.32	0.06±0.05	0.24±0.91
NGGP + Spectral	**0.03±0.03**	**-0.68±0.23**	**0.03±0.03**	**-0.62±0.24**

4.3.3 Contribution and impact

In this paper (Sendera et al., 2021c), I am the first author and the primary contributor. I proposed the integration of Gaussian Processes (GPs) with continuous-time Normalising Flows to enhance model flexibility, leading to the development of the NGGPs framework. I was responsible for implementing the proposed method and designing the experimental setup. I conducted a substantial portion of the experiments and analysed the results. Additionally, I contributed significantly to the preparation of the manuscript.

This work was accepted and presented at NeurIPS 2021, a top-tier machine learning conference (CORE A*), during the virtual event. It was well-received by the research community and has been cited 26 times as of August 2025, particularly in the context of meta-learning and few-shot learning papers published at leading conferences (Dai et al., 2023; Lorek et al., 2022; Patacchiola et al., 2022).

4.4 HyperShot and BayesHyperShot ([VII] and [VIII])

In this section, we present another two publications of the Ph.D. series, both of which address few-shot classification problems and achieve results competitive with state-of-the-art approaches. Publication [VII] introduces a novel paradigm in few-shot learning, called **HyperShot**, which leverages *hypernetworks* in combination with *kernel functions*. Publication [VIII] then extends this method into a probabilistic setting, resulting in **BayesHyperShot**, a Bayesian version of the original framework designed to better handle *uncertainty*.

The proposed HyperShot method operates on the relations between support set embeddings, using a kernel function to extract relational structure and condition the target classifier via a hypernetwork. This approach allows the model to effectively generalise across tasks, achieving performance that is comparable or superior to strong non-probabilistic baselines.

The subsequent extension, BayesHyperShot, incorporates Bayesian reasoning into the framework. By modelling the target classifier as a BNN and training it via variational inference, this version introduces the crucial ability to quantify uncertainty. This makes the model especially well-suited for settings involving distributional shifts or *out-of-distribution* (OOD) samples, where uncertainty estimation plays a critical role.

4.4.1 Motivation

The central aim of these works is to advance the *learning how to learn* paradigm by developing meta-learning algorithms that do not rely directly on the gradient optimisation solutions. Instead, we propose an approach in which information is first aggregated through a kernel function, and subsequently passed to a hypernetwork, which generates a set of task-specific weights for a *target network*. This decoupling from gradient-based updates allows the model to make larger, more direct adaptations when facing new tasks—improving both efficiency and generalisation.

In contrast to the regression setting explored in Section §4.3, the focus here is on the few-shot classification problem — as the most canonical and widely studied form of meta-learning.

The motivation for this line of research comes from the intution that effective and efficient meta-learners should not be constrained by the incremental nature of standard gradient descent. Instead, they should be capable of taking larger steps or even *jumping* across the solution space when adapting to the novel data. Thus, hypernetworks, which can directly generate task-specific parameters conditioned on context, provide a promising mechanism for achieving this.

However, most existing hypernetwork-based approaches have no constraints for uncertainty quantification, which severely limits their applicability in real-world scenarios where calibrated predictions are essential. To address this, we propose also a Bayesian extension that enables the model to express uncertainty in its predictions and better handle out-of-distribution or ambiguous data.

4.4.2 Content

Hypernetwork paradigm — HyperShot. To address the limitations of gradient-based adaptation in few-shot learning, we propose **HyperShot**, a method based on the **hypernetwork paradigm** (Ha et al., 2017). A hypernetwork is a neural network, which generates the weights of another neural network — in our case, a classification model. We may see the hypernetwork as a special conditioning. This makes it particularly well-suited for few-shot scenarios, as it enables fast adaptation to completely novel tasks by directly predicting task-specific classifier parameters conditioned on the support set.

A key challenge in this setting is extracting informative features from the support set in a permutation-invariant way. In few-shot learning, we typically operate on embeddings of examples obtained from a shared, trainable neural network (refered to as *backbone*). In order to extract the task-specific information, we propose to use a **parametrised kernel function**, which captures **relations between support samples** rather than relying on pure values. These kernel values are then passed to the hypernetwork, which predicts the classification model's parameters for the current task. Importantly, this formulation improves adaptation to new tasks, even if their embedding space is not align with those seen previously. Note that during inference, classification, is also performed using the kernel values computed with respect to the support set.

We describe HyperShot from a high-level perspective, following the flow in the architecture diagram in Fig. 4.5. Suppose we want to predict the class distribution $p(\mathbf{y}|\mathcal{S},\mathbf{x})$, given a query image $\mathbf{x}$ and a support set $\mathcal{S}$ consisting of K labelled examples. We first group the support samples $\mathbf{x}_l$ by class compute their embeddings via the encoder $E(\cdot)$, resulting in $\mathbf{z}_l = E(\mathbf{x}_l)$. These are then stacked into a matrix $\mathbf{Z}$, still ordered by label.

Next, we compute the **kernel matrix** $\mathbf{K}_{S,S}$, where each entry $k_{i,j} = k(\mathbf{z}_i,\mathbf{z}_j)$ denotes a kernel similarity between support embeddings. In fact, this matrix encodes the relational information between support samples for a given task. We then flatten matrix $\mathbf{K}_{S,S}$ and feed it into a hypernetwork $H(\cdot)$, resulting in the predicted task-specific parameters $\boldsymbol{\theta}_T$ for the classifier $T(\cdot)$.

For a given query sample $\mathbf{x}$, we first compute its embedding $z_\mathbf{x} = E(\mathbf{x})$ and then evaluate its similarity towards support set via a specially constructed kernel vector: $\mathbf{k}_{\mathbf{x},S} =$

$[k(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\pi(1)}), \ldots, k(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\pi(K)})]^{\mathrm{T}}$. This kernel vector is used as input to the target model $T(\cdot)$, which returns the final prediction.

During the training part, we optimise the parameters of the encoder, hypernetwork, and kernel function jointly. Because we focus on the classification problems, we use a *cross-entropy* loss over query predictions:

$$\mathcal{L}_{CE} = -\sum_{\mathcal{T}_i \in \mathcal{D}} \sum_{m=1}^{M} \sum_{k=1}^{K} y_{i,m}^{k} \log p(y_{i,m}^{k} | \mathcal{S}_i, \mathbf{x}_{i,m}), \tag{4.21}$$

where $\mathcal{Q}_i = \{(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})\}_{m=1}^{M}$ is the query set for task $\mathcal{T}_i$, consisting of query samples $(\mathbf{x}_{i,n}, \mathbf{y}_{i,n})$, and the prediction $p(\mathbf{y}|\mathcal{S}, \mathbf{x})$ is obtained from the classifier generated by the hypernetwork.

During inference, predictions for a new task can be made directly using the learned model — by taking the probability values $p(\mathbf{y}|\mathcal{S}_*, \mathbf{x})$ assuming the given support set $\mathcal{S}_*$ and single query observation $\mathbf{x}$ from $\mathcal{X}_*$. Optionally, the model can also be firstly *fine-tuned* on the support set to further improve performance.

In both Publication [VII] and Publication [VIII], we provide detailed algorithmic descriptions, and discussions on the choice of a kernel function for better adaptation. However, we reveal here that in our experiments, cosine similarity kernel consistently yielded the best performance. We refer interested readers to the original papers for further discussion.

Bayesian extension — BayesHyperShot. In Publication [VIII], we propose a Bayesian extension of HyperShot, named **BayesHyperShot**, which allows for effective uncertainty quantification — particularly valuable when samples from a novel task significantly differ from those seen previously. To this end, we model the target network $\mathcal{T}$ as a **Bayesian neural network** (BNN) by assuming a prior distribution over its parameters. Specifically, we adopt a typical BNNs' prior, a standard Gaussian distribution:

$$p(\boldsymbol{\theta}_{\mathcal{T}}) = \mathcal{N}(\boldsymbol{\theta}_{\mathcal{T}} \mid 0, I).$$

This necessitates adapting the hypernetwork to output both means and variances for the parameters of the target network. Thus, the target network $\mathcal{T}$ can serve as a probabilistic model during inference.

The most challenging is performing Bayesian inference effectively within a framework where all parameters are optimised via gradient-based methods. To maintain this paradigm, we have to employ **variational inference**.

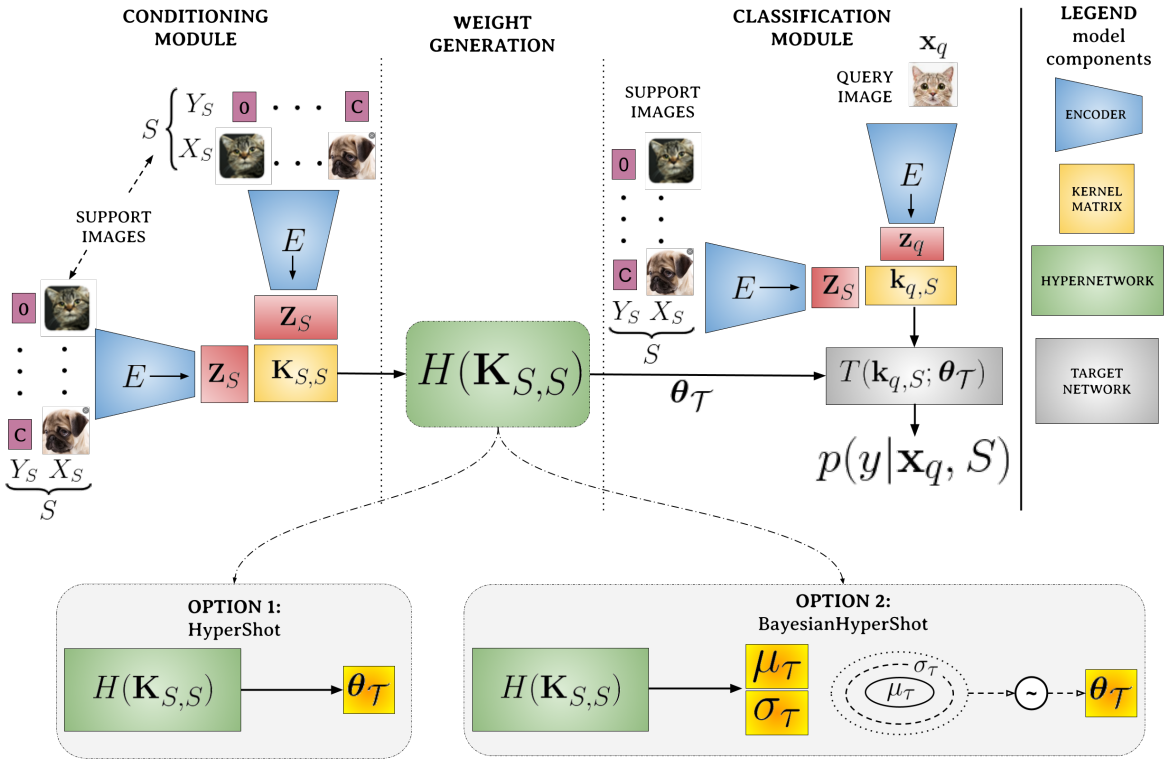


Fig. 4.5 Architectures of HyperShot and BayesHyperShot.

In particular, we define the following amortised variational posterior:

$$q(\boldsymbol{\theta}_{\mathcal{T}} \mid \mathcal{S}_i, \boldsymbol{\theta}_H) = \mathcal{N}(\boldsymbol{\theta}_{\mathcal{T}} \mid \mu_{\boldsymbol{\theta}_H}(\mathcal{S}_i), \sigma_{\boldsymbol{\theta}_H}(\mathcal{S}_i)), \tag{4.22}$$

where the Gaussian parameters — mean $\mu(\mathcal{S}_i)$ and standard deviation $\sigma(\mathcal{S}_i)$, are generated by the hypernetwork $H_{\boldsymbol{\theta}_H}$, conditioned on a given support set $\mathcal{S}_i$:

$$(\mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)) = H_{\boldsymbol{\theta}_H}(\mathcal{S}_i, \boldsymbol{\theta}_H).$$

The proposed formulation leads to updating the objective to the following variational inference loss:

$$\mathcal{L}_{VI} = \sum_{\mathcal{T}_i \in \mathcal{D}} \left[\frac{1}{P} \sum_{p=1}^{P} \left[\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}^p_{\mathcal{T}_i}) - \gamma D_{\mathrm{KL}}(q(\boldsymbol{\theta}^p_{\mathcal{T}_i} \mid \mathcal{S}_i, \boldsymbol{\theta}_H) \mid \mathcal{N}(0, I)) \right] \right], \tag{4.23}$$

where $\boldsymbol{\theta}^1_{\mathcal{T}_i}, \ldots, \boldsymbol{\theta}^P_{\mathcal{T}_i} \sim q(\boldsymbol{\theta}_{\mathcal{T}} \mid \mathcal{S}_i, \boldsymbol{\theta}_H)$ are the target parameters sampled by variational posterior, modeled as the hypernetwork. The term $\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}^p_{\mathcal{T}_i})$ denotes the cross-entropy loss function evaluated using the target model parametrised with $\boldsymbol{\theta}^p_{\mathcal{T}_i}$. Finally, $D_{\mathrm{KL}}(\cdot,\cdot)$ represents the

Kullback–Leibler divergence (Kullback and Leibler, 1951) between the posterior and the standard Gaussian distribution.
To improve the training stability, we introduce a reversely annealed coefficient γ that controls the trade-off between cross-entropy loss and the strength of the regularisation term.
During inference, the final prediction is obtained by averaging the outputs among sampled target networks, which captures predictive uncertainty.

Results. To comprehensively evaluate the two proposed methods, we perform an extensive series of few-shot classification experiments, including *in-domain* and also *cross-domain* adaptation ones. In particular, we focus on standard 1-*shot* and 5-*shot* scenarios across several benchmark datasets: CUB, Omniglot, EMNIST, and mini-Imagenet. We compare our approaches against the most of baselines and state-of-the-art approaches[1] including, *e.g.*, ProtoNet (Snell et al., 2017), RelationNet (Kang et al., 2021), Feature Transfer (Zhuang et al., 2020), Baseline++ (Chen et al., 2019b), MAML (Finn et al., 2017), LLAMA (Grant et al., 2018), VERSA (Gordon et al., 2018), and DKT (Patacchiola et al., 2020).
Across multiple benchmarks, HyperShot consistently ranks among the top-performing non-Bayesian methods, often achieving the best or second-best results in terms of accuracy. Meanwhile, BayesHyperShot, when evaluated against other Bayesian approaches, typically ranks third or fourth, slightly behind the best-performing Bayesian competitors.

Uncertainty estimation. However, BayesHyperShot achieves slightly worse results than HyperShot, but it allows for the model's uncertainty estimation.

Although BayesHyperShot achieves slightly lower accuracy than its deterministic counterpart, HyperShot, it offers an essential advantage: the ability to quantify predictive uncertainty. In particular, BayesHyperShot can indicate whether a given sample comes from the known distribution — same distribution as the support set of the current few-shot task or is an out-of-distribution example. To assess this capability, we conduct a set of experiments where a trained BayesHyperShot model is exposed to both in-distribution and OOD samples. We observe that the model yields confident (low-variance) predictions on support and query examples while exhibiting high variance on OOD inputs, as presented in Fig. 4.6. These results confirm that the model can effectively quantify uncertainty, *i.e.*, here recognising when a sample is dissimilar to the current few-shot task.

[1] Available at the time of experimentation.

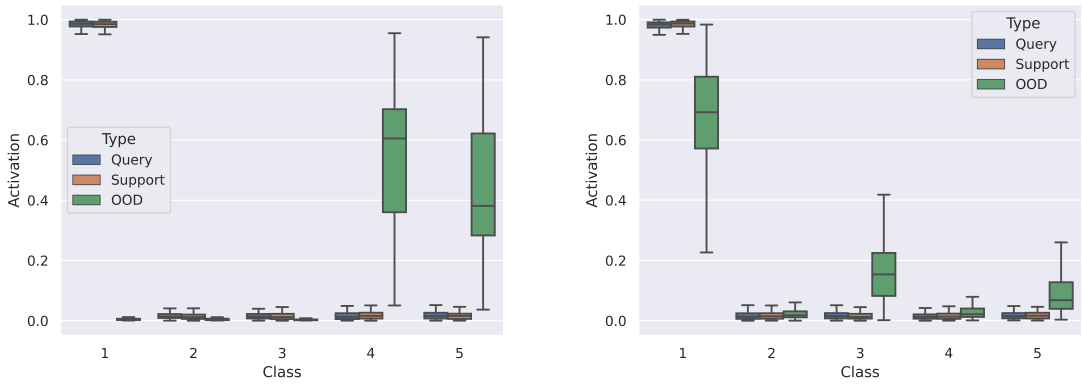


Fig. 4.6 We visualise box plots of distributions of activations produced by the sampled models for the two different sets of support / query / out-of-distribution images. We observe that BayesHyperShot always yields similar predictions for elements from both the support and query sets, while returning high variance for OOD examples.

4.4.3 Contribution and impact

In both of these papers (Sendera et al., 2022, 2023), I am a co–first author and share equal contribution with Marcin Przewięźlikowski. My contributions across both works were analogous: I co-developed the theoretical frameworks and was responsible for their implementation. In particular, I proposed and formalised the conditioning mechanism based on kernel functions, which became a core element of the method. I also ran a subset of the experiments, analysed the results, and was significantly involved in drafting and refining the manuscripts.

Publication [VII] was presented at WACV 2023, a reputable computer vision conference (CORE A). It was very well received and selected as one of approximately ten finalists in the Best Paper Award competition. As a result, we were invited to submit an extended version (Publication [VIII]) to a special issue of the *Machine Vision and Applications* journal. As of August 2025, Publication [VII] has been cited 41 times and Publication [VIII] 8 times. The HyperShot framework has gained traction within the few-shot learning community, both in terms of theoretical exploration (Chauhan et al., 2024; Sen et al., 2023; Yang and Wang, 2024) and practical applications (Gordon et al., 2024; Mehta et al., 2024).

4.5 Revisiting the equivalence of Bayesian neural networks and Gaussian processes: on the importance of learning activations [IX]

In the previous sections, we presented publications that employed probabilistic (Bayesian) frameworks for specific applications, mainly in meta-learning. In this section, we turn our attention to Publication [IX], which focuses on transferring the behaviour of *Gaussian processes* (GPs) to *Bayesian neural networks* (BNNs). We revisit the well-known equivalence theorem between GPs and wide single-hidden-layer BNNs, and demonstrate the crucial role of learning activation functions in bridging the gap between theory and practice. Our proposed framework not only enjoys strong theoretical support but also achieves state-of-the-art performance.

Our proposed methodology is centred around optimising an approximated *Wasserstein divergence* and introduces a joint learning of the BNN's activation function and priors over its weights and biases. This approach greatly improves scalability and adaptability, and we propose solutions to several practical challenges encountered when transferring GP-like behaviour into a BNN. Our extensive experiments support the effectiveness of the method, which we believe will allow for the seamless integration of Bayesian principles into standard deep learning models.

4.5.1 Motivation

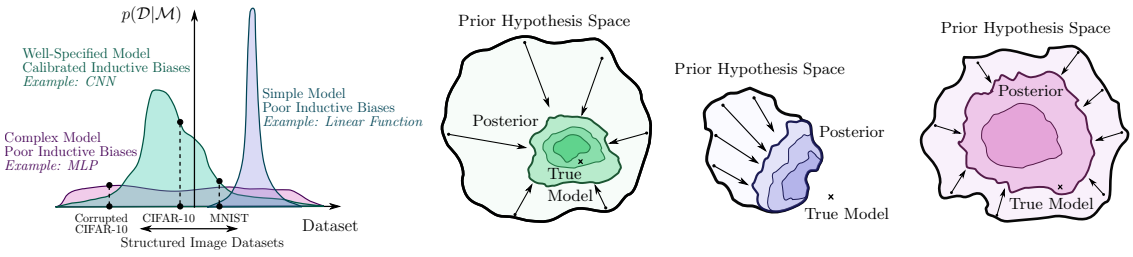
A central challenge in Bayesian deep learning is the selection of an appropriate prior. In practice, the prior heavily influences the posterior, and if it is poorly chosen, the true posterior may not be recoverable — even with infinite data, as presented in Fig. 4.7. To enable effective uncertainty estimation and generalisation, we require more powerful, expressive priors, *e.g.*, defined directly in function space.

Gaussian processes are well-known from providing a convenient way to specify the function-space priors, making them a natural choice for uncertainty quantification — an essential aspect of Bayesian deep learning. However, their applicability is limited by poor scalability, primarily due to the $\mathcal{O}(n^3)$ computational cost. In contrast, Bayesian neural networks offer significantly better scalability but are more difficult to train and lack some of the key advantages of GPs. Bridging these two worlds by transferring a GP behaviour to a single hidden-layer BNN could yield scalable models with well-defined function-space priors, which are able to imitate specific GP in any neural network.

Transferring GP-like behaviour to BNNs is therefore a natural line of research. While it is a known fact that wide single-hidden-layer BNNs with certain activation functions and

priors converge to GPs with specific kernels, the practical realisation of this correspondence remains challenging. Closed-form solutions exist only for very specific cases, typically involving simple kernel–activation pairs. Moreover, existing GP-to-BNN transfer methods often lack theoretical justification and rely heavily on heuristics.

Given these limitations, we aim to develop a principled, theoretically grounded approach for transferring GP-like behaviour into BNNs. Such a method would allow BNNs to imitate a given GP, ensuring convergence to the wanted behaviour.



((a)) ((b)) ((c)) ((d))

Fig. 4.7 Prior selection is a key challenge in Bayesian deep learning, as we observe the chose of a prior and a model influence directly in the possibility of modelling the true posterior. *Image source:* Wilson and Izmailov (2020).

4.5.2 Content

In this part, we begin by revisiting the classical result on the equivalence between GPs and a specific class of BNNs. We then introduce the framework proposed in Publication [IX], which focuses on transferring function-space priors from GPs onto BNNs, highlighting the key challenges and proposed improvements.

GP - BNN equivalence. Let us start by recalling the classical equivalence between GPs and infinitely wide Bayesian neural networks, first introduced in Neal (1996) and Williams (1996), and later generalised for deep architectures (Lee et al., 2017; Matthews et al., 2018).
Consider a neural network defined as follows:

$$
\begin{aligned}
f_i^0(\mathbf{x}) &= \sum_j^I \mathbf{w}_{ij}^0 \mathbf{x}_j + \mathbf{b}_i^0, \quad i = 1, \ldots, H_0; \\
f^l(\mathbf{x}) &= \sum_j^{H_0} \mathbf{w}_{ij}^l \phi(f_j^0(\mathbf{x})) + \mathbf{b}^l,
\end{aligned}
\tag{4.24}
$$

where $\mathbf{x} \in \mathbb{R}^I$ is the input vector, H_0 is the width of the l-*th* hidden layer, and $f^l(\mathbf{x})$ is its output. A neural network has a fixed nonlinear activation function, $\phi(\cdot)$.

When his network is viewed as a probabilistic (Bayesian) model, the terms $\mathbf{z}_{ij}(\mathbf{x}) = \mathbf{w}^l_{ij}\phi(f^0_j(\mathbf{x}))$ are treated as random variables. Specifically, we typically assume common prior distribution over all weights w_{ij} and another common prior over biases b_i. This implies that the outputs of the hidden layer, and therefore the input to the next layer, are random variables.
Importantly, the output of the l-th layer, $f^l(\mathbf{x})$, becomes a sum over *i.i.d.* random variables. By the *Central Limit Theorem*, this sum converges in distribution to a Gaussian as $H_0 \to \infty$. Consequently, in the infinite-width limit, the BNN converges to a Gaussian Process.

We can explicitly derive the covariance of the output of l-th layer of a BNN, $f^l(\mathbf{x})$. Consider two inputs $\mathbf{x}$ and $\mathbf{x}'$, then this covariance has the following form:

$$\mathrm{Cov}(f^l(\mathbf{x}), f^l(\mathbf{x}')) = {\sigma^l_b}^2 + {\sigma^l_w}^2 \mathbb{E}_{f^0_j}[\phi(f^0_j(\mathbf{x}))\phi(f^0_j(\mathbf{x}'))], \tag{4.25}$$

where ${\sigma^l_w}^2$ and ${\sigma^l_b}^2$ denote the variances of the weights and biases in the l-th layer, respectively. Notice that the expectation is taken over the distribution induced by the random weights and biases in the previous layer — w^0 and b^0.

Moreover, we can derive the GP kernel corresponding to this BNN functional prior as:

$$\kappa^l_f(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f^l(\mathbf{x}), f^l(\mathbf{x}')). \tag{4.26}$$

For simplicity, we assume zero-centered GPs, which corresponds to setting $\mathbb{E}[w] = 0$ and $\mathbb{E}[b] = 0$ in the BNN. Additionally, to ensure variance stability as the layer's width grows, we apply appropriate scaling:

$$\mathrm{Var}[w^l_{ij}] = \frac{{\sigma^l_w}^2}{H_0}.$$

This result provides a theoretical bridge between BNNs and GPs, motivating the idea that it is possible to design BNNs that behave similarly to specific GPs.

Transfer of priors from a GP into a BNN. Having established the theoretical equivalence between GPs and BNNs, let us now consider how to transfer the behaviour of one model into the other. Transferring the behaviour from a pretrained BNN into a GP is relatively straightforward — we just need to estimate the covariance function of the BNN via Monte Carlo methods (*see* Eq. 4.25).
The inverse problem — transferring a specific GP-like behaviour into a BNN — is significantly more challenging. We formalise this task in the following way:

Problem 3. *Identify appropriate priors on w^l, b^l, w^0, b^0, and the activation ϕ introduced in the Eq. 4.25, s.t., the covariance induced by the BNN aligns with a target GP kernel κ.*

More precisely, we aim to achieve the behaviour $f^l \sim GP(0,\kappa)|_{\mathcal{X}}$*, meaning that the distribution over BNN outputs aligns with that of a GP on the input domain* $\mathcal{X}$*. Ideally, this implies:*

$$p_{\text{NN}}(f^l) = p_{\text{GP}}(f^l),$$

however in practice, we can only require approximate solution over finite index sets:

$$p_{\text{NN}}(f^l(X)) \approx p_{\text{GP}}(f^l(X)), \quad \textit{for} \quad X \subset \mathcal{X}.$$

To solve Problem 3, we consider the following optimisation task:

$$p^*_{\text{NN}} = \arg\min_{p_{\text{NN}}} \frac{1}{S} \sum_{X \sim p_X} D(p_{\text{NN}}(f^l(X)), p_{\text{GP}}(f^l(X))), \tag{4.27}$$

where D is an *arbitrary*, differentiable divergence measure (*e.g.*, KL divergence, Wasserstein distance), and S denotes number of samples.

Differentiable priors and activations. A key question in the GP-to-BNN transfer problem is how to appropriately parameterise the BNN model. Notice that the GP-BNN equivalence holds under the assumption of weights and biases distributions with finite variance and light tails.

This observation informs us how to design prior distributions — we adopt zero-centered factorised Gaussian priors with learnable variances for all weights and biases, *i.e.*, $p(w|\sigma^2_w)$ and $p(b|\sigma^2_b)$. Moreover, leveraging the reparameterisation trick (Kingma and Welling, 2014), enables gradient-based optimisation of these parameters.

However, we find that optimising only the variational parameters of weights and biases is not enough, because we limit the flexibility of the BNN to approximate a target GP. This limitation arises because such optimisation explores only a *single* set of BNN priors' parameters that fulfills the optimisation problem, **while operating on a fixed activation function**. Recall that the desired GP-like behaviour might be realised by multiple prior-activation pairs.

To address this, in Publication [IX], we propose to employ the parameterisation with **parametric and differentiable activations**, denotes $\phi(\cdot|\eta)$. This results in the complete set of parameters: $\lambda = \{\sigma^0_b, \sigma^0_w, \sigma^l_b, \sigma^l_w, \eta\}$ and allows us to reformulate the optimisation problem from Eq. 4.27 as:

$$\lambda^* = \arg\min_{\lambda} \frac{1}{S} \sum_{X \sim p_X} D\Big(p_{\text{NN}}\Big(f^l(X|\lambda)\Big), p_{\text{GP}}\Big(f^l(X)\Big)\Big). \tag{4.28}$$

An additional contribution introduced in Publication [IX] is the choice of the divergence measure D. We observe that the most intuitive divergences, such as the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951), often fail in this setting — either due to numerical problems or because they collapse to trivial solutions during optimisation.

Instead, inspired by prior work (*e.g.*, Tran et al. (2022a)), we use the *p-Wasserstein distance*, which compares distributions in function space:

$$W_p = \left(\inf_{\varsigma \in \Gamma(p_{\text{NN}}, p_{\text{GP}})} \int_{F \times F} d(f, f')^p \varsigma(f, f') df df' \right)^{1/p}, \tag{4.29}$$

where $\Gamma(p_{\text{NN}}, p_{\text{GP}})$ denotes the set of all joint distributions with the respective marginals, and $d(\cdot, \cdot)$ is a metric over the function space F. Moreover, thanks to the *Kantorovich-Rubinstein duality* (Kantorovich and Rubinstein, 1958), we know that *1-Wasserstein distance* can be present in another way:

$$W_1 = \sup_{|\Psi|_L \leq 1} \mathbb{E}_{p_{\text{NN}}}[\Psi(f)] - \mathbb{E}_{p_{\text{GP}}}[\Psi(f)], \tag{4.30}$$

where we consider the *supremum* over all *Lipschitz functions* with the *Lipschitz norm* ≤ 1.

However, unlike Tran et al. (2022a), we propose to use the **2-Wasserstein metric**, which in the case of multivariate Gaussians has a closed-form solution:

$$W_2^2 = ||\boldsymbol{\mu_1} - \boldsymbol{\mu_2}||_2^2 + \text{Tr}\left(\boldsymbol{\Sigma_1} + \boldsymbol{\Sigma_2} - 2\sqrt{\sqrt{\boldsymbol{\Sigma_1}}\boldsymbol{\Sigma_2}\sqrt{\boldsymbol{\Sigma_1}}} \right), \tag{4.31}$$

where $\boldsymbol{\mu_{1/2}}$, $\boldsymbol{\Sigma_{1/2}}$ are respectively expectations and covariance matrices estimated for p_{NN} and p_{GP} from samples $\{f^l(X)\}$.

This approximation, allows a computationally efficient and numerically stable objective, since it avoids the costly internal optimisation loop required for solving the *Kantorovich-Rubinstein dual problem* (Kantorovich and Rubinstein, 1958) as in Tran et al. (2022a).

Additional improvements. Behind proposing a general framework for transferring GP functional priors to BNNs, we also addressed common challenges in this domain.

First, in Gaussian Processes we employ *stationary kernels*, which implies globally stationary behaviour of a GP, *i.e.*, consistent function properties even outside the training input domain. In contrast, BNNs typically exhibit only *local* stationarity due to the nature of their priors and activation functions. As a result, BNNs often become overconfident or underconfident when extrapolating beyond the observed input range, which results in failing to mimic the desired GP-like behaviour.

We address this issue by introducing *trainable activations designed to exhibit periodic behaviour*, inspired by the findings of Meronen et al. (2021), which show that stationarity in BNNs might be induced by periodic activations. Specifically, we propose the following parametric form of activations:

$$\phi(\mathbf{x}|\psi, A) = \sum_{i=1}^{K} A_i \cos(2\pi\psi_i \mathbf{x}) + \sum_{j=1}^{K} A_j \sin(2\pi\psi_j \mathbf{x}), \tag{4.32}$$

which can be learned jointly with the BNN parameters to induce the desired stationary functional prior. These activations rely on K components equipped with the variational parameters $\eta = \{\psi_i, A_i, \psi_j, A_j\}$ describing frequencies and amplitudes.

Another limitation of existing approaches to transfer functional priors to BNNs is the need to fix GP hyperparameters — such as the lengthscale — *a priori*, or alternatively to define complex hierarchical kernels that sample these hyperparameters from additional distributions. Both approaches limit model flexibility and increase computational cost.
To mitigate these issues, we propose to condition the priors over weights, biases, and activations directly on the GP kernel hyperparameters. This allows for inducing GP hyperparameters directly into BNNs. In practice, we propose to use the hypernetworks (Ha et al., 2017), which take GP kernel hyperparameters as inputs and generate BNN priors parameters. It results in significantly enhancing model flexibility.

For additional details and design choices, training procedures, and empirical results, we refer the reader to the full text of Publication [IX].

Results. In order to adequately evaluate the proposed novel methodology for transferring GP function-space priors into BNNs, we perform an extensive set of experiments. These included comparisons with existing methods, closed-form solutions (where available), and standard Gaussian Processes. The primary baselines considered were Meronen et al. (2020) and Tran et al. (2022a). For evaluation, we report common performance metrics such as negative log-likelihood (NLL), root mean square error (RMSE).

Here, we present a selected subset of results that clearly show the superiority of the proposed methodology across several benchmarks including multi-dimensional regression tasks (Tab. 4.2) and 2d classification (Fig. 4.8).

Table 4.2 Performance on UCI regression datasets, measured using RMSE and NLL. We observe similar behaviour of our approach and Tran et al. (2022a).

Dataset →	Boston ($d = 12$)		Concrete ($d = 8$)		Energy ($d = 8$)		Protein ($d = 9$)		Wine ($d = 11$)	
Method ↓ Metric →	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*
baseline	$2.8402_{\pm 0.8986}$	$2.4778_{\pm 0.1481}$	$4.4628_{\pm 0.7511}$	$2.9728_{\pm 0.0876}$	$0.3431_{\pm 0.0613}$	$0.3482_{\pm 0.1607}$	$0.5038_{\pm 0.0093}$	$0.7447_{\pm 0.0142}$	$0.4736_{\pm 0.0420}$	$0.8725_{\pm 0.0285}$
ours	$2.8643_{\pm 0.8386}$	$2.4937_{\pm 0.1798}$	$4.9143_{\pm 0.7528}$	$3.0201_{\pm 0.1011}$	$0.3692_{\pm 0.0566}$	$0.4064_{\pm 0.1347}$	$0.4957_{\pm 0.0058}$	$0.7251_{\pm 0.0094}$	$0.4833_{\pm 0.0398}$	$0.8673_{\pm 0.0255}$

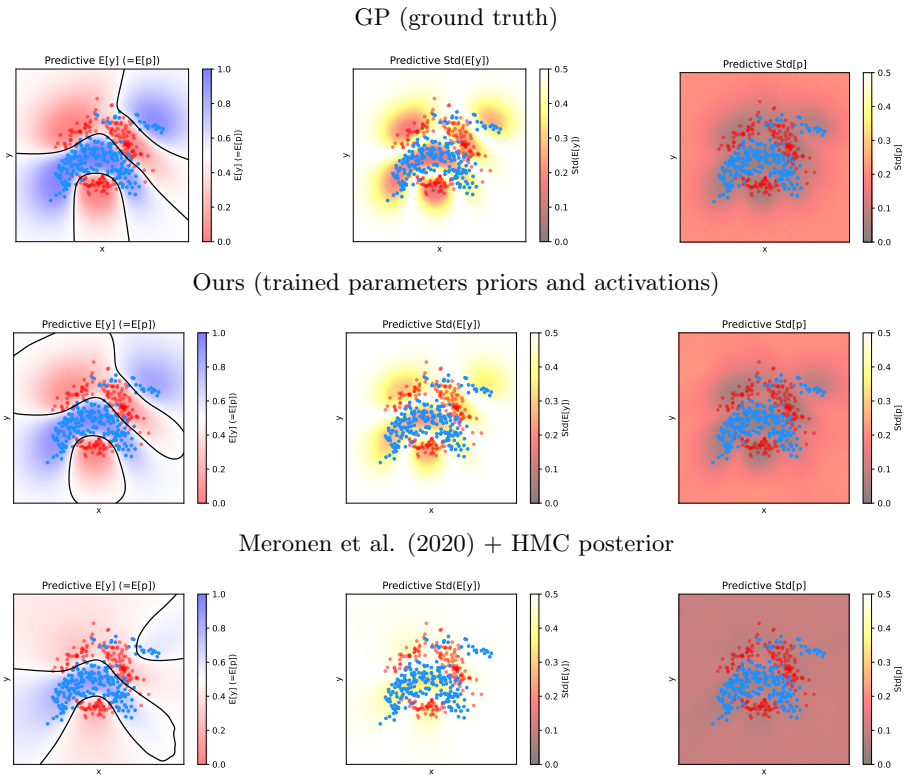


Fig. 4.8 Comparison of the ground truth GP with Matern kernel (**top row**) and induced BNNs behaviours on a 2D classification problem. We compare our approach (**middle row**) and the closed-form solution from Meronen et al. (2020), observing better uncertainty estimation of our approach — closer to the ground truth.

Moreover, we empirically verified our research hypotheses and empirically validated the following key findings:

- Learning activations significantly improves learning of function-space priors.
- Expressive function-space priors **do not require** networks to be deep.
- Learned activations can match the performance of existing closed-form solutions.
- The proposed periodic activations successfully induce stationarity in BNNs.
- The model selection is possible with transfer priors.
- The approach scales well even to the largest problems from the GP literature (Wang et al., 2019b).

4.5.3 Contribution and impact

In this work (Sendera et al., 2025a), I am the first author and share equal contribution with Dr Tomasz Kuśmierczyk, the last author. I explored addressing the problem through optimisation of the Wasserstein metric and introduced the use of its closed-form solution as a key improvement. I was actively involved in conceptualising the method, addressing both theoretical and practical challenges. Additionally, I implemented the method, as well as part of the baselines and experimental setup. I conducted approximately half of the experiments and analysed their results. I also contributed substantially to writing the manuscript.
The paper was recently presented (July 2025) at UAI 2025 — a top-tier (CORE A) conference. The work is currently available as a preprint on arXiv and already cited once.

5

Broader impact and academic contributions

In this chapter, we present additional research papers co-authored by the Ph.D. candidate during the doctoral studies, followed by a summary of research internships, awarded grants, academic service, and other professional achievements. While the following publications are not included in the main series of Ph.D. publications, they nonetheless represent significant contributions to the field of deep learning.

5.1 Other publications

This section outlines all other publications published during the Ph.D. studies that are not part of the core dissertation series.

These works can be broadly categorised into three research areas: (1) diffusion samplers for efficient posterior inference, (2) applications of deep learning to scientific and medical challenges, and (3) AI safety, particularly in the context of machine unlearning.

In the first group, we explore the use of diffusion models for posterior inference — a central problem in probabilistic deep learning. In Venkatraman et al. (2025a), we consider sampling posteriors of generative models such as VAEs and GANs. Specifically, we propose to amortise the cost of sampling by using diffusion models that sample a distribution in the latent space.

In Venkatraman et al. (2024b), we introduce a new objective, *Relative Trajectory Balance (RTB)*, which allows one to sample from a target posterior distribution when the prior is defined by a diffusion model.
The second group focuses on the application of deep learning to scientific discovery. In Sendera et al. (2020), we introduce a novel data assimilation technique for complex system simulation, with potential applications in weather prediction. In Newsham et al. (2024), we develop a deep learning-based classifier for multi-class prediction of cancer types from tissue methylation profiles, achieving high accuracy and offering potential for early cancer detection. In Akhound-Sadegh et al. (2024), we return to the challenge of sampling from unnormalised densities, proposing a novel method called *Iterated Denoising Energy Matching (iDEM)*, especially effective in molecular energy landscapes as Lennard-Jones system.
The third group addresses critical aspects of AI safety, specifically machine unlearning in generative models. In Sendera et al. (2025c), we propose *SEMU*, a method based on singular value decomposition (SVD) that enables selective forgetting of specific data points. This is achieved through a compact, low-dimensional projection, allowing for targeted unlearning while preserving the model's overall performance.

In addition to these conference and journal publications, we have presented several workshop papers at top-tier venues such as *NeurIPS*, *ICLR*, and *ICML*. These workshop contributions typically present preliminary results, applications, or condensed versions of the main conference papers.
Specifically, Venkatraman et al. (2024a) and Scimeca et al. (2025) explore Bayesian inverse problems in computer vision and astrophysics, extending the RTB objective proposed in Venkatraman et al. (2024b). The workshop paper Venkatraman et al. (2025b) serves as a shorter version of Venkatraman et al. (2025a). Similarly, Sendera et al. (2024b) presents preliminary findings that were later developed into the full paper Sendera et al. (2025a), and Sendera et al. (2025b) offers a condensed version of Sendera et al. (2025c), focusing specifically on machine unlearning in generative AI models.

In the following sections, we present these additional publications in reverse chronological order. For conference and journal papers, we include abstracts and the number of citations as of August 2025. For workshop papers, we provide a comprehensive list.

5.1.1 Conference and journal papers

SEMU: singular value decomposition for efficient machine unlearning

Sendera, M., Struski, Ł., Książek, K., Musiol, K., Tabor, J., Rymarczyk, D., 2025, July. SEMU: Singular Value Decomposition for Efficient Machine Unlearning. In *The International Conference on Machine Learning*. PMLR. (Sendera et al., 2025c)
CORE A*, 200 MSHE points, cited: 3 times.

Abstract: While the capabilities of generative foundational models have advanced rapidly in recent years, methods to prevent harmful and unsafe behaviors remain underdeveloped. Among the pressing challenges in AI safety, machine unlearning (MU) has become increasingly critical to meet upcoming safety regulations. Most existing MU approaches focus on altering the most significant parameters of the model. However, these methods often require fine-tuning substantial portions of the model, resulting in high computational costs and training instabilities, which are typically mitigated by access to the original training dataset.In this work, we address these limitations by leveraging Singular Value Decomposition (SVD) to create a compact, low-dimensional projection that enables the selective forgetting of specific data points. We propose Singular Value Decomposition for Efficient Machine Unlearning (SEMU), a novel approach designed to optimize MU in two key aspects. First, SEMU minimizes the number of model parameters that need to be modified, effectively removing unwanted knowledge while making only minimal changes to the model's weights. Second, SEMU eliminates the dependency on the original training dataset, preserving the model's previously acquired knowledge without additional data requirements.Extensive experiments demonstrate that SEMU achieves competitive performance while significantly improving efficiency in terms of both data usage and the number of modified parameters.

Outsourced diffusion sampling: efficient posterior inference in latent spaces of generative models

Venkatraman, S.*, Hasan, M.*, Kim, M., Scimeca, L., **Sendera, M.**, Bengio, Y., Berseth, G. Malkin, N., 2025, July. Outsourced Diffusion Sampling: Efficient Posterior Inference in Latent Spaces of Generative Models. In *The International Conference on Machine Learning*. PMLR. (Venkatraman et al., 2025a)
CORE A*, 200 MSHE points, cited: 3 times.

Abstract: Any well-behaved generative model over a variable **x** can be expressed as a deterministic transformation of an exogenous (*'outsourced'*) Gaussian noise variable

$\mathbf{z}$: $\mathbf{x} = f_\theta(\mathbf{z})$. In such a model (*e.g.*, a VAE, GAN, or continuous-time flow-based model), sampling of the target variable $\mathbf{x} \sim p_\theta(\mathbf{x})$ is straightforward, but sampling from a posterior distribution of the form $p(\mathbf{x} \mid \mathbf{y}) \propto p_\theta(\mathbf{x}) r(\mathbf{x}, \mathbf{y})$, where r is a constraint function depending on an auxiliary variable $\mathbf{y}$, is generally intractable. We propose to amortize the cost of sampling from such posterior distributions with diffusion models that sample a distribution in the noise space ($\mathbf{z}$). These diffusion samplers are trained by reinforcement learning algorithms to enforce that the transformed samples $f_\theta(\mathbf{z})$ are distributed according to the posterior in the data space ($\mathbf{x}$). For many models and constraints of interest, the posterior in the noise space is smoother than the posterior in the data space, making it more amenable to such amortized inference. Our method enables conditional sampling under unconditional GAN, (H)VAE, and flow-based priors, comparing favorably both with current amortized and non-amortized inference methods. We demonstrate the proposed *outsourced diffusion sampling* in several experiments with large pretrained prior models: conditional image generation, reinforcement learning with human feedback, and protein structure generation.

Amortizing intractable inference in diffusion models for vision, language, and control
Venkatraman, S.*, Jain, M.*, Scimeca, L.*, Kim, M.*, **Sendera, M.***, Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., Adam, A., Rector-Brooks, J., Bengio, Y., Berseth, G. and Malkin, N., 2024, December. Amortizing intractable inference in diffusion models for vision, language, and control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. (Venkatraman et al., 2024b)
CORE A*, 200 MSHE points, cited: 44 times.

Abstract: Diffusion models have emerged as effective distribution estimators in vision, language, and reinforcement learning, but their use as priors in downstream tasks poses an intractable posterior inference problem. This paper studies *amortized* sampling of the posterior over data, $\mathbf{x} \sim p^{\text{post}}(\mathbf{x}) \propto p(\mathbf{x}) r(\mathbf{x})$, in a model that consists of a diffusion generative model prior $p(\mathbf{x})$ and a black-box constraint or likelihood function $r(\mathbf{x})$. We state and prove the asymptotic correctness of a data-free learning objective, *relative trajectory balance*, for training a diffusion model that samples from this posterior, a problem that existing methods solve only approximately or in restricted cases. Relative trajectory balance arises from the generative flow network perspective on diffusion models, which allows the use of deep reinforcement learning techniques to improve mode coverage. We illustrate the broad potential of unbiased inference of arbitrary posteriors under diffusion priors across a collection of experiments: in vision (classifier guidance), language (infilling under a discrete diffusion LLM), and multimodal data (text-to-image generation). Beyond generative modeling, we apply relative trajectory balance to the problem of continuous control with a score-based

behavior prior, achieving state-of-the-art results on benchmarks in offline reinforcement learning. Code is available at this link.

Iterated denoising energy matching for sampling from Boltzmann densities
Akhound-Sadegh, T.*, Rector-Brooks, J.*, Bose, J.*, Mittal, S., Lemos, P., Liu, C.H., **Sendera, M.**, Ravanbakhsh, S., Gidel, G., Bengio, Y., Malkin, N., and Tong. A., 2024, July. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. In *International Conference on Machine Learning* (pp. 760-786). PMLR. (Akhound-Sadegh et al., 2024)
CORE A*, 200 MSHE points, cited: 72 times.

Abstract: Efficiently generating statistically independent samples from an unnormalized probability distribution, such as equilibrium samples of many-body systems, is a foundational problem in science. In this paper, we propose Iterated Denoising Energy Matching (iDEM), an iterative algorithm that uses a novel stochastic score matching objective leveraging solely the energy function and its gradient – and no data samples – to train a diffusion-based sampler. Specifically, iDEM alternates between (I) sampling regions of high model density from a diffusion-based sampler and (II) using these samples in our stochastic matching objective to further improve the sampler. iDEM is scalable to high dimensions as the inner matching objective, is simulation-free, and requires no MCMC samples. Moreover, by leveraging the fast mode mixing behavior of diffusion, iDEM smooths out the energy landscape enabling efficient exploration and learning of an amortized sampler. We evaluate iDEM on a suite of tasks ranging from standard synthetic energy functions to invariant n-body particle systems. We show that the proposed approach achieves state-of-the-art performance on all metrics and trains $2-5\times$ faster, which allows it to be the first method to train using energy on the challenging 55-particle Lennard-Jones system.

Early detection and diagnosis of cancer with interpretable machine learning to uncover cancer-specific DNA methylation patterns
Newsham, I., **Sendera, M.**, Jammula, S.G. and Samarajiwa, S.A., 2024. Early detection and diagnosis of cancer with interpretable machine learning to uncover cancer-specific DNA methylation patterns. *Biology Methods and Protocols*, 9(1), p.bpae028. (Newsham et al., 2024)
Impact Factor: 2.5, 20 MSHE points, cited: 7 times.

Abstract: Cancer, a collection of more than two hundred different diseases, remains a leading cause of morbidity and mortality worldwide. Usually detected at the advanced

*denotes equal contribution

stages of disease, metastatic cancer accounts for 90% of cancer-associated deaths. Therefore, the early detection of cancer, combined with current therapies, would have a significant impact on survival and treatment of various cancer types. Epigenetic changes such as DNA methylation are some of the early events underlying carcinogenesis. Here, we report on an interpretable machine learning model that can classify 13 cancer types as well as non-cancer tissue samples using only DNA methylome data, with 98.2% accuracy. We utilize the features identified by this model to develop EMethylNET, a robust model consisting of an XGBoost model that provides information to a deep neural network that can generalize to independent data sets. We also demonstrate that the methylation-associated genomic loci detected by the classifier are associated with genes, pathways and networks involved in cancer, providing insights into the epigenomic regulation of carcinogenesis.

Supermodeling: the next level of abstraction in the use of data assimilation

Sendera, M., Duane, G. S., and Dzwinel, W. (2020). Supermodeling: the next level of abstraction in the use of data assimilation. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings*, Part VI 20 (pp. 133-147). Springer International Publishing. (Sendera et al., 2020)
CORE A[†], 140 MSHE points, cited: 9 times.

Abstract: Data assimilation (DA) is a key procedure that synchronizes a computer model with real observations. However, in the case of overparametrized complex systems modeling, the task of parameter-estimation through data assimilation can expand exponentially. It leads to unacceptable computational overhead, substantial inaccuracies in parameter matching, and wrong predictions. Here we define a *Supermodel* as a kind of ensembling scheme, which consists of a few *sub-models* representing various instances of the baseline model. The *sub-models* differ in parameter sets and are synchronized through couplings between the most sensitive dynamical variables. We demonstrate that after a short pre-training of the fully parametrized small *sub-model* ensemble, and then training a few latent parameters of the low-parameterized *Supermodel*, we can outperform in efficiency and accuracy the baseline model matched to data by a classical DA procedure.

[†]At the time of publication.

5.1.2 Workshop papers

- **Sendera, M.**, Struski, Ł., Książek, K., Musiol, K., Tabor, J., Rymarczyk, D., 2025, July. Embarrassingly Efficient Unlearning with SVD. In *ICML 2025 Workshop on Machine Unlearning for Generative AI*. (Sendera et al., 2025b)

- Venkatraman, S.*, Hasan, M.*, Kim, M., Scimeca, L., **Sendera, M.**, Bengio, Y., Berseth, G. Malkin, N., 2025, April. Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. In *The Frontiers in Probabilistic Inference: Sampling meets Learning (FPI) workshop at ICLR 2025*. (Venkatraman et al., 2025b)

- Scimeca, L.*, Venkatraman, S.*, Jain, M.*, Kim, M.*, **Sendera, M.***, Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., Adam, A., Rector-Brooks, J., Hezaveh, Y., Perreault-Levasseur, L., Bengio, Y., Berseth, G., Malkin, N., 2025, April. Solving Bayesian inverse problems with diffusion priors and off-policy RL. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. (Scimeca et al., 2025)

- Venkatraman, S.*, Jain, M.*, Scimeca, L.*, Kim, M.*, **Sendera, M.***, Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., Adam, A., Rector-Brooks, J., Bengio, Y., Berseth, G. and Malkin, N., 2024, December. Amortizing intractable inference in diffusion models for Bayesian inverse problems. In *Proc. Workshop on Machine Learning and the Physical Sciences. Accessed*, pp. 02-06. (Venkatraman et al., 2024a)

- **Sendera, M.***, Sorkhei, A. and Kuśmierczyk, T.*, 2024, December. Hi-fi functional priors by learning activations. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*. (Sendera et al., 2024b)

5.2 Current research topics

The current research interests of the Ph.D. candidate remain centred on core challenges in probabilistic deep learning, although they now extend beyond those addressed in the main series of publications.

In particular, I continue to work on scaling amortised variational methods — especially diffusion-based samplers — to high-dimensional settings. A key direction involves applying these methods to train energy-based models and to model Bayesian posteriors in scientific contexts (*e.g.*, AI Scientist and AI Mathematician) as well as in AI safety applications. These include enabling reasoning in large language models (LLMs), such as by training models to learn and solve mathematical problems expressed in formal languages like Lean. The goal is to develop models capable of stating hypotheses or conjectures and attempting to prove them. I am also exploring how diffusion samplers can be used to model Bayesian posteriors over LLMs and to enforce probabilistic safety constraints on them.

Additionally, I am actively engaged in research on AI safety, particularly machine unlearning. In this line of work, the objective is to enable generative models — such as diffusion models — to selectively forget specific datapoints (in line with the *"right to be forgotten"*) or to remove harmful concepts such as nudity from their generative capabilities.

5.3 Internships

During and prior to my Ph.D. studies, I had the opportunity to collaborate with world-class researchers during research internships. These experiences include working with Prof. Yoshua Bengio on diffusion-based samplers and with Dr Shamith Samarajiwa on applying deep learning methods to early cancer detection. These internships have led to several scientific publications and initiated ongoing collaborative projects. Brief descriptions are provided below:

- **Mila – Québec Artificial Intelligence Institute, Université de Montréal – Canada**
  Reaserch Internship with Prof. Yoshua Bengio
  2023–2024 (approx. 11 months)
  I focused on learning Bayesian posteriors and improving the adaptation of diffusion samplers, formulated as continuous GFlowNets. This internship resulted in at least three publications at CORE A* conferences and has led to ongoing collaborations on multiple research projects.

- **Hutchison – MRC Research Centre, University of Cambridge – United Kingdom**
  Summer Studentship with Dr Shamith Samarajiwa
  2018 (approx. 3 months)
  I worked on applying deep learning techniques for the early detection and multi-class classification of various cancer types. This work resulted in a peer-reviewed journal publication.

5.4 Established collaborations

During the Ph.D. studies, I had the pleasure of collaborating with many outstanding researchers from around the world, establishing valuable scientific connections. Below, I outline the most significant national and international collaborations, which have resulted in published papers or preprints that are currently under review.

International collaborations

- Prof. Yoshua Bengio at Mila – Québec AI Institute and Université de Montréal
  Co-written four published papers on GFlowNets and diffusion samplers, published at top-tier conferences (ICML 2024, NeurIPS 2024, ICML 2025).

- Dr Nikolay Malkin at University of Edinburgh
  Co-written four published papers on GFlowNets and diffusion samplers, published at top-tier conferences (ICML 2024, NeurIPS 2024, ICML 2025). Currently working on training energy-based models, diffusion samplers and AI models for mathematics.

- Dr Julius Berner at Caltech and NVIDIA
  Co-written a preprint on diffusion samplers, being currently under review.

- Dr Lorenz Richter at Zuse Institute Berlin and dida
  Co-written a preprint on diffusion samplers, being currently under review.

- Dr Massimiliano Patacchiola at University of Cambridge
  Co-written a paper on Non-Gaussian Gaussian Processes in meta-learning, presented at NeurIPS 2021.

- Other researchers at Mila – Québec AI Institute, Université de Montréal and McGill University, being co-authors of at least one published paper:

  - *Interns*: Minsu Kim (from KAIST)
  - *Ph.D. students*: Moksh Jain, Sarthak Mittal, Mohsin Hasan, Siddarth Venkatraman, Jarrid Rector-Brooks, Avishek Joey Bose, Tara Akhound-Sadegh, Cheng-Hao Liu, Luke Rowe, Alexandre Adam
  - *Postdocs*: Luca Scimeca, Pablo Lemos, Alexander Tong
  - *Researchers from Recursion*: Emmanuel Bengio
  - *Professors*: Yashar Hezaveh, Laurence Perreault-Levasseur, Siamak Ravanbakhsh, Gauthier Gidel, Glen Berseth

National collaborations

- Prof. Maciej Zięba and Konrad Karanowski at Wrocław University of Science and Technology
  Co-written papers on Non-Gaussian Gaussian Processes and hypernetworks in meta-learning, presented at, e.g., NeurIPS 2021, WACV 2023.

- Tomasz Trzciński at Warsaw University of Technology
  Co-written a paper on Non-Gaussian Gaussian Processes in meta-learning, presented at NeurIPS 2021.

- Dr Michał Stypułkowski at University of Wrocław
  Co-written a preprint on fine-tuning diffusion models with LoRA layers, currently under review.

5.5 Research grants

Throughout my Ph.D. studies, I was awarded two research grants as a principal investigator — one from the Jagiellonian University and another from the National Science Centre (NCN) in Poland. In addition, I contributed to several other research projects as a co-investigator. All of these grants and projects are outlined below.

Received fundings

- **Awarded the PRELUDIUM grant by the National Science Centre in Poland (NCN)**
  How to learn faster: towards better adaptation in Meta-Learning.
  Funds: 139 471 PLN
  2022–*now*

- **Awarded the Mini-grant for young scientists at Jagiellonian University**
  Deep Gaussian Processes for motion tracking with the use of Normalizing Flows.
  Funds: 20 000 PLN
  2021–2022

Grants to which I contributed (as a co-investigator)

- **Member of the project group in the CIFAR AI Catalyst Grant**
  AI Mathematician.
  PI: Dr Nikolay Malkin (University of Edinburgh)
  2024–*now*

- **Stipendist in OPUS grant by the National Science Centre in Poland (NCN)**
  Meta-learning in deep neural networks.
  PI: Prof. Jacek Tabor (Jagiellonian University)
  2024–*now*

- **Stipendist in OPUS grant by the National Science Centre in Poland (NCN)**
  Generative flow-based models in application to uncertainty modeling for machine learning tasks.
  PI: Prof. Maciej Zięba (Wrocław University of Science and Technology)
  2022–2023

- **Stipendist in OPUS grant by the National Science Centre in Poland (NCN)**
  Deep generative models for 3D representations.
  PI: Prof. Maciej Zięba (Wrocław University of Science and Technology)
  2021–2022

5.6 Summer schools on machine learning

During my Ph.D. studies, I had the opportunity to attend several virtual summer schools focused on various aspects of machine learning. These are outlined below:

- *Machine Learning Meets Neuroscience Summer School* (MLSS^{N}) 2022
- *Oxford Machine Learning Summer School* (OxML): 2022
- *Eastern European Machine Learning Summer School* (EEML): 2021, 2022

5.7 Service for the research community

In addition to the previously described activities, I had the pleasure of serving the research community in several capacities throughout my Ph.D. studies.
I volunteered for and co-organised machine learning summer schools in Kraków, where I was responsible for tasks such as programme coordination and inviting speakers. During the events, I also served as a session chair for talks given by prominent researchers including Prof. Yarin Gal, Prof. José Miguel Hernández-Lobato, and Prof. Marco Cuturi.
Furthermore, I acted as a reviewer for several top-tier conferences and workshops in machine learning.
I also represented Ph.D. students on two academic bodies at Jagiellonian University: the Council of the Faculty of Mathematics and Computer Science (2022–2024) and the Council of the Institute of Computer Science (2022–2025).

The specific conferences I reviewed for, as well as the summer schools I co-organised and supported, are listed below.

Summer schools I co-organised and supported

- *Machine Learning Summer School on Drug and Materials Discovery* (MLSS^{D}) 2025
- *Machine Learning Summer School on Applications in Science* (MLSS^{S}) 2023
- *Machine Learning Meets Neuroscience Summer School* (MLSS^{N}) 2022

Conferences I reviewed for

- *International Conference on Machine Learning* (ICML): 2022, 2024, 2025; CORE A*, 200 MSHE points.
- *Conference on Neural Information Processing Systems* (NeurIPS): 2023, 2025; CORE A*, 200 MSHE points.
- *International Conference on Learning Representations* (ICLR): 2024, 2025; CORE A*, 200 MSHE points.
- *IEEE/CVF Winter Conference on Applications of Computer Vision* (WACV): 2023; CORE A, 140 MSHE points.
- *Conference on Lifelong Learning Agents* (CoLLAs): 2023, 2024; unrated by CORE, unrated by MSHE.

6

Synthesis and future directions

In this work, we summarise nine papers that collectively form the Ph.D. research series of publications, addressing key challenges in probabilistic deep learning. In addition, we outline the author's broader scientific contributions, including established research collaborations, reviewing duties, awarded research grants, volunteering, and involvement in organising scientific events.

The nine publications can be grouped into three interconnected categories, each centred around a distinct research theme within probabilistic deep learning.
Publications [I], [II], [III] focus on density estimation using normalising flow models, with applications to anomaly detection and missing data settings. Specifically, Publications [I] and [II] demonstrate how normalising flows, when combined with the Bernstein estimator or a deep learning formulation of the Support Vector Data Description (SVDD) objective, enable effective anomaly detection. Publication [III] investigates and reports a surprising behaviour: a normalising flow model trained on data with missing regions is able to complete images at inference time. These works address practical challenges associated with real-world data and contribute to building more robust and reliable deep learning systems.
Publications [IV] and [V] tackle another fundamental issue in probabilistic deep learning: sampling from unnormalised densities. These papers propose novel methods for diffusion-based sampling using the entropic reinforcement learning framework, specifically through

discrete-time policies in Markov Decision Processes (*i.e.*, continuous GFlowNets). Publication [IV] introduces a new off-policy training approach, based on a replay buffer and a parallel MALA algorithm, and benchmarks the existing alternatives. Publication [V] establishes formal links between entropic RL methods (GFlowNets) and continuous-time objects (path-space measures). Moreover, the empirical findings show that coarse and non-uniform time discretisations can significantly improve training efficiency. These contributions advance the scalability of diffusion samplers and pave the way for sampling from complex densities, such as posteriors of large language models.
Publications [VI], [VII], [VIII], [IX] focus on arguably the most important aspect of probabilistic deep learning — Bayesian inference — and its two principal challenges: posterior sampling and prior specification. Publication [VI] introduces Non-Gaussian Gaussian Processes (NGGPs), a generalisation of classical GPs that allows modelling of locally non-Gaussian posteriors, enabling effective few-shot regression. Publications [VII] and [VIII] propose a family of methods based on hypernetworks with kernel-based conditioning and Bayesian updates, achieving more efficient adaptation in few-shot classification. Finally, Publication [IX] presents a theoretically and empirically grounded method for imposing informed function-space priors in Bayesian Neural Networks (BNNs), effectively transferring desirable Gaussian Process behaviours into BNNs. Collectively, these contributions provide more flexible, yet tractable, Bayesian inference mechanisms and improve prior modelling.

The overarching motivation for this research is rooted in the urgent challenges facing the deep learning community: we must better understand the behaviour of models — especially those trained with reinforcement learning objectives — ensure their safety through probabilistic measures, and scale them reliably to solve complex scientific problems. As argued throughout this thesis, a probabilistic perspective is essential for developing safe and trustworthy AI systems.
The presented publications contribute to three foundational areas of probabilistic deep learning: density estimation, efficient sampling, and Bayesian inference. Together, they move us closer to the next generation of AI models that are not only more reliable, interpretable, and safe, but also capable of reasoning and assisting scientific discovery. These are currently the central research topics of the Ph.D. candidate.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297.

Adam, A., Coogan, A., Malkin, N., Legin, R., Perreault-Levasseur, L., Hezaveh, Y., and Bengio, Y. (2022). Posterior samples of source galaxies in strong gravitational lenses with score-based priors. *arXiv preprint arXiv:2211.03812*.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Aitken, A. C. (1936). Iv.—on least squares and linear combination of observations. *Proceedings of the Royal Society of Edinburgh*, 55:42–48.

Akhound-Sadegh, T., Rector-Brooks, J., Bose, J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., et al. (2024). Iterated denoising energy matching for sampling from Boltzmann densities. In *International Conference on Machine Learning*, pages 760–786. PMLR.

Albergo, M. S., Kanwar, G., and Shanahan, P. E. (2019). Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, 100(3):034515.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., and McAuley, J. (2021). Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pages 1352–1361. PMLR.

Baker, D., Hassabis, D., and Jumper, J. (2024). The Nobel prize in chemistry 2024. *Stockholm, Sweden: The Royal Swedish Academy of Sciences*.

Bandeira, A. S., Maillard, A., Nickl, R., and Wang, S. (2022). On free energy barriers in Gaussian priors and failure of cold start MCMC for high-dimensional unimodal distributions. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 381.

Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

Bartoldson, B. R., Venkatraman, S., Diffenderfer, J., Jain, M., Ben-Nun, T., Lee, S., Kim, M., Obando-Ceron, J., Bengio, Y., and Kailkhura, B. (2025). Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable llm post-training. *arXiv preprint arXiv:2503.18929*.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, (53):370–418.

Behjoo, H. and Chertkov, M. (2025). Harmonic path integral diffusion. *IEEE Access*.

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. In *International conference on machine learning*, pages 573–582. PMLR.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*.

Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume 2, page 969. IEEE.

Bengio, Y., Cohen, M., Fornasiere, D., Ghosn, J., Greiner, P., MacDermott, M., Mindermann, S., Oberman, A., Richardson, J., Richardson, O., et al. (2025). Superintelligent agents pose catastrophic risks: Can scientist ai offer a safer path? *arXiv preprint arXiv:2502.15657*.

Bengio, Y., Cohen, M. K., Malkin, N., MacDermott, M., Fornasiere, D., Greiner, P., and Kaddar, Y. (2024). Can a Bayesian oracle prevent harm from an agent? *arXiv preprint arXiv:2408.05284*.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023). GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

Berner, J., Richter, L., Sendera, M., Rector-Brooks, J., and Malkin, N. (2025). From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training. *arXiv preprint arXiv:2501.06148*.

Berner, J., Richter, L., and Ullrich, K. (2022). An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*.

Bernstein, S. (1912). Demonstration du th'eoreme de Weierstrass fondee sur le calcul des probabilites. *Comm. Kharkov Math. Soc.*, 13(1):1–2.

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., et al. (2021). Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 401–413.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, Berlin, Heidelberg.

Bishop, C. M. (2013). Model-based machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20120222.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR.

Bogachev, V. I. and Ruas, M. A. S. (2007). *Measure theory*, volume 1. Springer.

Bommasani, R. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Brehmer, J. and Cranmer, K. (2020). Flows for simultaneous manifold learning and density estimation. *Advances in neural information processing systems*, 33:442–453.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Buchner, J. (2021). Nested sampling methods. *arXiv preprint arXiv:2101.09675*.

Carnap, R. (1945). The two concepts of probability: The problem of probability. *Philosophy and phenomenological research*, 5(4):513–532.

Carnap, R. (1962). *Logical foundations of probability*, volume 2. Citeseer.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., and Clifton, D. A. (2024). A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):1–29.

Chen, R. T., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. (2019a). Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32.

Chen, R. T. and Duvenaud, D. K. (2019). Neural networks with cheap differential operators. *Advances in Neural Information Processing Systems*, 32.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. (2019b). A closer look at few-shot classification. In *International Conference on Learning Representations*.

Chen, Y., Qian, J., and Saligrama, V. (2013). A new one-class svm for anomaly detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3567–3571. IEEE.

Cheng, C. (1995). The Bernstein polynomial estimator of a smooth quantile function. *Statistics & probability letters*, 24(4):321–330.

Chong, P., Ruff, L., Kloft, M., and Binder, A. (2020). Simple and effective prevention of mode collapse in deep one-class classification. *arXiv preprint arXiv:2001.08873*.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, 89(3):539–552.

Coddington, E. A. and Levinson, N. (1955). *Theory of ordinary differential equations*. McGraw-Hill New York.

Dai, Y., Yan, W., and Yin, F. (2023). Graphical multioutput gaussian process with attention. In *The Twelfth International Conference on Learning Representations*.

Dalrymple, D., Skalse, J., Bengio, Y., Russell, S., Tegmark, M., Seshia, S., Omohundro, S., Szegedy, C., Goldhaber, B., Ammann, N., et al. (2024). Towards guaranteed safe ai: A framework for ensuring robust and reliable ai systems. *arXiv preprint arXiv:2405.06624*.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021a). Laplace redux-effortless Bayesian deep learning. *Advances in neural information processing systems*, 34:20089–20103.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. (2021b). Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pages 2510–2521. PMLR.

De Bortoli, V. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research (TMLR)*.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436.

Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. (2022). Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Filos, A., Tigkas, P., McAllister, R., Rhinehart, N., Levine, S., and Gal, Y. (2020). Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Fisher, R. (1958). The nature of probability. *The Centennial Review of Arts & Science*, 2:261–274.

Föllmer, H. (1985). An entropy approach to the time reversal of diffusion processes. pages 156–163.

Fort, S. and Jastrzebski, S. (2019). Large scale structure of neural network loss landscapes. *Advances in Neural Information Processing Systems*, 32.

Gabrié, M., Rotskoff, G. M., and Vanden-Eijnden, E. (2021). Efficient Bayesian sampling using normalizing flows to assist Markov chain Monte Carlo methods. *arXiv preprint arXiv:2107.08001*.

Gal, Y. et al. (2016). Uncertainty in deep learning.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. M. A. (2018). Conditional neural processes. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR.

Gauss, C. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Carl Friedrich Gauss Werke. Sumtibus F. Perthes et I.H. Besser.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis*. Chapman and Hall/CRC.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press.

Gopalan, P., Sharan, V., and Wieder, U. (2019). Pidforest: Anomaly detection via partial identification. In *Advances in Neural Information Processing Systems*, pages 15809–15819.

Gordon, C., MacDonald, L. E., Saratchandran, H., and Lucey, S. (2024). D'OH: Decoder-only random hypernetworks for implicit neural representations. In *Proceedings of the Asian Conference on Computer Vision*, pages 2507–2526.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. (2018). Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2019). FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*.

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581.

Gresele, L., Fissore, G., Javaloy, A., Schölkopf, B., and Hyvarinen, A. (2020). Relative gradient optimization of the Jacobian term in unsupervised deep learning. *Advances in neural information processing systems*, 33:16567–16578.

Griffiths, R.-R., Klarner, L., Moss, H., Ravuri, A., Truong, S., Du, Y., Stanton, S., Tom, G., Rankovic, B., Jamasb, A., et al. (2023). Gauche: a library for Gaussian processes in chemistry. *Advances in Neural Information Processing Systems*, 36:76923–76946.

Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in cognitive sciences*, 14(8):357–364.

Gritsaev, T., Morozov, N., Tamogashev, K., Tiapkin, D., Samsonov, S., Naumov, A., Vetrov, D., and Malkin, N. (2025). Adaptive destruction processes for diffusion samplers. *arXiv preprint arXiv:2506.01541*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Ha, D., Dai, A. M., and Le, Q. V. (2017). Hypernetworks. In *International Conference on Learning Representations*.

Hagemann, P. L., Hertrich, J., and Steidl, G. (2023). *Generalized normalizing flows via Markov chains*. Cambridge University Press.

Hald, A. (1998). *A history of mathematical statistics from 1750 to 1930*. John Wiley Sons, New York.

Halton, J. H. (1962). Sequential Monte Carlo. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 57–78. Cambridge University Press.

Harrison, J., Willes, J., and Snoek, J. (2024). Variational Bayesian last layers. *International Conference on Learning Representations (ICLR)*.

Hasenclever, L., Tomczak, J. M., van den Berg, R., Welling, M., et al. (2017). Variational inference with orthogonal normalizing flows. In *Workshop Bayesian Deep Learn.(NeurIPS)*.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, 2nd ed. edition.

Hein, M., Andriushchenko, M., and Bitterwolf, J. (2019). Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 41–50.

Hendrycks, D. and Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. *International Conference on Machine Learning (ICML)*.

Herr, D. G. (1980). On the history of the use of geometry in the general linear model. *The American Statistician*, 34(1):43–47.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*.

Ho, J., Lohn, E., and Abbeel, P. (2019). Compression with flows via local bits-back coding. *Advances in Neural Information Processing Systems*, 32.

Hoffman, M. D., Gelman, A., et al. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1):1593–1623.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hojjati, H. and Armanfard, N. (2023). Dasvdd: Deep autoencoding support vector data descriptor for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3739–3750.

Holdijk, L., Du, Y., Hooft, F., Jaini, P., Ensing, B., and Welling, M. (2023). Stochastic optimal control for collective variable free sampling of molecular transition paths. *Neural Information Processing Systems (NeurIPS)*.

Hoogeboom, E., Cohen, T., and Tomczak, J. M. (2021). Learning discrete distributions by dequantization. In *Third Symposium on Advances in Approximate Bayesian Inference*.

Hoogeboom, E., Peters, J., Van Den Berg, R., and Welling, M. (2019). Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32.

Hopfield, J. J. and Hinton, G. E. (2024). The Nobel prize in physics 2024. *Stockholm, Sweden: The Royal Swedish Academy of Sciences*.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169.

Hu, E. J., Malkin, N., Jain, M., Everett, K., Graikos, A., and Bengio, Y. (2023). GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*.

Huang, C., Guan, H., Jiang, A., Zhang, Y., Spratling, M., Wang, X., and Wang, Y. (2024). Few-shot anomaly detection via category-agnostic registration learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. (2021). What are Bayesian neural network posteriors really like? *International Conference on Machine Learning (ICML)*.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Jensen, J. L. W. V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. (2022). Torsional diffusion for molecular conformer generation. *Neural Information Processing Systems (NeurIPS)*.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. (2022). Hands-on Bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. (2022). Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., and McHardy, R. (2023). Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. (2017). Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*.

Kang, D., Kwon, H., Min, J., and Cho, M. (2021). Relational embedding for few-shot classification.

Kantorovich, L. V. and Rubinstein, S. (1958). On a space of totally additive functions. *Vestnik of the St. Petersburg University: Mathematics*, 13(7):52–59.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.

Kidger, P., Foster, J., Li, X., and Lyons, T. J. (2021a). Neural SDEs as infinite-dimensional GANs. *International Conference on Machine Learning (ICML)*.

Kidger, P., Foster, J., Li, X. C., and Lyons, T. (2021b). Efficient and accurate gradients for neural SDEs. *Neural Information Processing Systems (NeurIPS)*.

Kim, M., Choi, S., Yun, T., Bengio, E., Feng, L., Rector-Brooks, J., Ahn, S., Park, J., Malkin, N., and Bengio, Y. (2024a). Adaptive teachers for amortized samplers. *arXiv preprint arXiv:2410.01432*.

Kim, M., Seong, K., Woo, D., Ahn, S., and Kim, M. (2025). On scalable and efficient training of diffusion samplers. *arXiv preprint arXiv:2505.19552*.

Kim, M., Yu, J., Kim, J., Oh, T.-H., and Choi, J. K. (2023). An iterative method for unsupervised robust anomaly detection under data contamination. *IEEE Transactions on Neural Networks and Learning Systems*.

Kim, M., Yun, T., Bengio, E., Zhang, D., Bengio, Y., Ahn, S., and Park, J. (2024b). Local search GFlowNets. *International Conference on Learning Representations (ICLR)*.

Kim, S., Choi, Y., and Lee, M. (2015). Deep learning with support vector data description. *Neurocomputing*, 165:111–117.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*.

Kirichenko, P., Izmailov, P., and Wilson, A. G. (2020). Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589.

Kloeden, P. E. and Platen, E. (1992). *Stochastic differential equations*. Springer.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979.

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. (2023). A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30.

Laplace, P. (1812). *Théorie analytique des probabilités*. Courcier, Paris.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*.

Lee, J., Plainer, M., Du, Y., Holdijk, L., Brekelmans, R., Gomes, C. P., Beaini, D., and Neklyudov, K. (2025). Scaling deep learning solutions for transition path sampling. In *ICLR 2025 Workshop on Generative and Experimental Perspectives for Biomolecular Design*.

Legendre, A. M. (1806). *Nouvelles méthodes pour la détermination des orbites des comètes*. Courcier.

Lemos, P., Malkin, N., Handley, W., Bengio, Y., Hezaveh, Y., and Perreault-Levasseur, L. (2023). Improving gradient-guided nested sampling for posterior inference. *International Conference on Machine Learning (ICML)*.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. (2020). Scalable gradients for stochastic differential equations. *Artificial Intelligence and Statistics (AISTATS)*.

Liang, S., Li, Y., and Srikant, R. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax Weiss, T., and Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, 33:7498–7512.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738.

Lorek, P., Nowak, R., Trzcinski, T., and Zieba, M. (2022). Flowhmm: Flow-based continuous hidden markov models. *Advances in Neural Information Processing Systems*, 35:8773–8784.

Lorentz, G. G. (1953). *Bernstein polynomials*. American Mathematical Soc.

Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A., Bosc, T., Bengio, Y., and Malkin, N. (2022). Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for Bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in gflownets. *Neural Information Processing Systems (NeurIPS)*.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. (2023). GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Margatina, K., Schick, T., Aletras, N., and Dwivedi-Yu, J. (2023). Active learning principles for in-context learning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5011–5034.

Margossian, C. C. and Blei, D. M. (2024). Amortized variational inference: when and why? In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, pages 2434–2449.

Markov, A. A. (1900). *Calculus of probabilities*. An. A. Markov, St. Petersburg. Published in Russian.

Máté, B. and Fleuret, F. (2023). Learning interpolations between Boltzmann densities. *Transactions on Machine Learning Research (TMLR)*.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *ICLR*.

Maziarka, Ł., Śmieja, M., Sendera, M., Struski, Ł., Tabor, J., and Spurek, P. (2021). Oneflow: One-class flow for anomaly detection based on a minimal volume region. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8508–8519.

McAllister, R. T., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

McGrayne, S. B. (2011). *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, & Emerged Triumphant from Two Centuries of Controversy*. Yale University Press.

Mehta, N., Liang, K. J., Huang, J., Chu, F.-J., Yin, L., and Hassner, T. (2024). Hypermix: Out-of-distribution detection and classification in few-shot settings. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2410–2420.

Meronen, L., Irwanto, C., and Solin, A. (2020). Stationary activations for uncertainty calibration in deep learning. *Advances in Neural Information Processing Systems*, 33:2338–2350.

Meronen, L., Trapp, M., and Solin, A. (2021). Periodic activation functions induce stationarity. *Advances in Neural Information Processing Systems*, 34:1673–1685.

Michelmore, R., Wicker, M., Laurenti, L., Cardelli, L., Gal, Y., and Kwiatkowska, M. (2020). Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 7344–7350. IEEE.

Miljković, D. (2010). Review of novelty detection methods. In *The 33rd International Convention MIPRO*, pages 593–598. IEEE.

Milnor, J. W. and Weaver, D. W. (1997). *Topology from the differentiable viewpoint*, volume 21. Princeton university press.

Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., and Lucic, M. (2021). Revisiting the calibration of modern neural networks. *Advances in neural information processing systems*, 34:15682–15694.

Mittal, S., Bengio, Y., Malkin, N., and Lajoie, G. (2025). In-context parametric inference: Point or distribution estimators? *arXiv preprint arXiv:2502.11617*.

Mittal, S., Bracher, N. L., Lajoie, G., Jaini, P., and Brubaker, M. A. (2023). Exploring exchangeable dataset amortization for Bayesian posterior inference. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. (2018). Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks, Vol. 118 of Lecture Notes in Statistics*.

Newsham, I., Sendera, M., Jammula, S. G., and Samarajiwa, S. A. (2024). Early detection and diagnosis of cancer with interpretable machine learning to uncover cancer-specific dna methylation patterns. *Biology Methods and Protocols*, 9(1):bpae028.

Newton, I. (1711). *De analysi per aequationes numero terminorum infinitas*. by William Jones. Written in 1669, published posthumously in 1711.

Newton, I. (1736). *The Method of Fluxions and Infinite Series: With Its Application to the Geometry of Curve-lines*. Henry Woodfall, London. Translated from the author's Latin original, not previously published.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations*.

Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *International Conference on Machine Learning (ICML)*.

Nixon, J., Dusenberry, M. W., Zhang, L., Jerfel, G., and Tran, D. (2019). Measuring calibration in deep learning. In *CVPR workshops*, volume 2.

Noé, F., Olsson, S., Köhler, J., and Wu, H. (2019). Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147.

Nüsken, N. and Richter, L. (2021). Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2(4):48.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Springer.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32.

Pan, L., Malkin, N., Zhang, D., and Bengio, Y. (2023). Better training of GFlowNets with local credit and incomplete trajectories. *International Conference on Machine Learning (ICML)*.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64.

Papamarkou, T., Skoularidou, M., Palla, K., Aitchison, L., Arbel, J., Dunson, D., Filippone, M., Fortuin, V., Hennig, P., Hernández-Lobato, J. M., et al. (2024). Position: Bayesian deep learning is needed in the age of large-scale ai. In *International Conference on Machine Learning*, pages 39556–39586. PMLR.

Patacchiola, M., Bronskill, J., Shysheya, A., Hofmann, K., Nowozin, S., and Turner, R. (2022). Contextual squeeze-and-excitation for efficient few-shot image classification. *Advances in Neural Information Processing Systems*, 35:36680–36692.

Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A. J. (2020). Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Phillips, D. and Cipcigan, F. (2024). MetaGFN: Exploring distant modes with adapted metadynamics for continuous GFlowNets. *arXiv preprint arXiv:2408.15905*.

Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*.

Plackett, R. L. (1949). A historical note on the method of least squares. *Biometrika*, 36(3/4):458–460.

Plackett, R. L. (1972). Studies in the history of probability and statistics. xxix: The discovery of the method of least squares. *Biometrika*, 59(2):239–251.

Pontryagin, L. S. (1962). *Mathematical theory of optimal processes*. Routledge.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. (2020). Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE transactions on neural networks and learning systems*, 33(4):1452–1466.

Raghu, M., Blumer, K., Sayres, R., Obermeyer, Z., Kleinberg, B., Mullainathan, S., and Kleinberg, J. (2019). Direct uncertainty prediction for medical second opinions. In *International conference on machine learning*, pages 5281–5290. PMLR.

Rajpurkar, P., Chen, E., Banerjee, O., and Topol, E. J. (2022). Ai in health and medicine. *Nature medicine*, 28(1):31–38.

Rao, C. R. (1973). *Linear statistical inference and its applications*, volume 2. Wiley New York.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

Rasmussen, C. E. and Williams, C. (2006). Gaussian processes for machine learning the mit press. *Cambridge, MA*.

Rector-Brooks, J., Madan, K., Jain, M., Korablyov, M., Liu, C.-H., Chandar, S., Malkin, N., and Bengio, Y. (2023). Thompson sampling for improved exploration in GFlowNets. *arXiv preprint arXiv:2306.17693*.

Reichenbach, H. (1971). *The theory of probability*. Univ of California Press.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

Richter, L. and Berner, J. (2024). Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*.

Richter, L., Berner, J., and Liu, G.-H. (2023). Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*.

Richter, L., Boustati, A., Nüsken, N., Ruiz, F. J. R., and Ömer Deniz Akyildiz (2020). VarGrad: A low-variance gradient estimator for variational inference. *Neural Information Processing Systems (NeurIPS)*.

Risken, H. (1996). *Fokker-planck equation*. Springer.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118.

Robert, C. P., Casella, G., and Casella, G. (1999). *Monte Carlo statistical methods*, volume 2. Springer.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Roth, K., Kilcher, Y., and Hofmann, T. (2019). The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

Rudin, W. (1987). *Real and complex analysis*. McGraw-Hill, Inc.

Rudner, T. G., Chen, Z., Teh, Y. W., and Gal, Y. (2022a). Tractable function-space variational inference in Bayesian neural networks. *Advances in Neural Information Processing Systems*, 35:22686–22698.

Rudner, T. G., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. (2022b). Continual learning via sequential function-space variational inference. pages 18871–18887.

Ruff, L. (2021). *Deep one-class learning: a deep learning approach to anomaly detection*. Technische Universitaet Berlin (Germany).

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In *International conference on machine learning*, pages 4393–4402.

Särkkä, S. and Solin, A. (2019). *Applied stochastic differential equations*. Cambridge University Press.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1(2):48.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Scimeca, L., Venkatraman, S., Jain, M., Kim, M., Sendera, M., Hasan, M., Adam, A., Hezaveh, Y., Perreault-Levasseur, L., Bengio, Y., et al. (2025). Solving Bayesian inverse problems with diffusion priors and off-policy rl. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*.

Scott, C. D. and Nowak, R. D. (2006). Learning minimum volume sets. *Journal of Machine Learning Research*, 7(Apr):665–704.

Seal, H. L. (1967). Studies in the history of probability and statistics. xv the historical development of the gauss linear model. *Biometrika*, 54(1-2):1–24.

Sen, B., Singh, G., Agarwal, A., Agaram, R., Krishna, M., and Sridhar, S. (2023). Hyp-nerf: Learning improved nerf priors using a hypernetwork. *Advances in Neural Information Processing Systems*, 36:51050–51064.

Sendera, M., Duane, G. S., and Dzwinel, W. (2020). Supermodeling: the next level of abstraction in the use of data assimilation. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI 20*, pages 133–147. Springer.

Sendera, M., Kim, M., Mittal, S., Lemos, P., Scimeca, L., Rector-Brooks, J., Adam, A., Bengio, Y., and Malkin, N. (2024a). Improved off-policy training of diffusion samplers. *Neural Information Processing Systems (NeurIPS)*.

Sendera, M., Przewięźlikowski, M., Karanowski, K., Zięba, M., Tabor, J., and Spurek, P. (2022). Hypershot: Few-shot learning by kernel hypernetworks. *arXiv preprint arXiv:2203.11378*.

Sendera, M., Przewiźlikowski, M., Miksa, J., Rajski, M., Karanowski, K., Ziba, M., Tabor, J., and Spurek, P. (2023). The general framework for few-shot learning by kernel hypernetworks. *Machine Vision and Applications*, 34(4):53.

Sendera, M., Śmieja, M., Maziarka, Ł., Struski, Ł., Spurek, P., and Tabor, J. (2021a). Flow-based svdd for anomaly detection. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Sendera, M., Sorkhei, A., and Kuśmierczyk, T. (2024b). Hi-fi functional priors by learning activations. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*.

Sendera, M., Sorkhei, A., and Kuśmierczyk, T. (2025a). Revisiting the equivalence of Bayesian neural networks and Gaussian processes: On the importance of learning activations. In *The 41st Conference on Uncertainty in Artificial Intelligence*.

Sendera, M., Struski, Ł., Książek, K., Musiol, K., Tabor, J., and Rymarczyk, D. (2025b). Embarrassingly efficient unlearning with svd. In *ICML 2025 Workshop on Machine Unlearning for Generative AI*.

Sendera, M., Struski, Ł., Książek, K., Musiol, K., Tabor, J., and Rymarczyk, D. (2025c). Semu: Singular value decomposition for efficient machine unlearning. In *International Conference on Machine Learning*. PMLR.

Sendera, M., Struski, Ł., and Spurek, P. (2021b). Missing glow phenomenon: Learning disentangled representation of missing data. In *International Conference on Neural Information Processing*, pages 196–204. Springer.

Sendera, M., Tabor, J., Nowak, A., Bedychaj, A., Patacchiola, M., Trzcinski, T., Spurek, P., and Zieba, M. (2021c). Non-Gaussian Gaussian processes for few-shot regression. *Advances in Neural Information Processing Systems*, 34:10285–10298.

Silva, T., de Souza, D. A., and Mesquita, D. (2024). Streaming Bayes gflownets. *Advances in Neural Information Processing Systems*, 37:27153–27177.

Skilling, J. (2006). Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833 – 859.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning (ICML)*.

Solomonoff, R. J. (1964a). A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22.

Solomonoff, R. J. (1964b). A formal theory of inductive inference. part ii. *Information and control*, 7(2):224–254.

Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021a). Maximum likelihood training of score-based diffusion models. *Neural Information Processing Systems (NeurIPS)*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*.

Stigler, S. M. (1981). Gauss and the invention of least squares. *the Annals of Statistics*, pages 465–474.

Stigler, S. M. (1986). *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press.

Su, Q., Tian, B., Wan, H., and Yin, J. (2024). Anomaly detection under contaminated data with contamination-immune bidirectional gans. *IEEE Transactions on Knowledge and Data Engineering*.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38.

Sutton, R. S., Barto, A. G., et al. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine learning*, 54(1):45–66.

Tomczak, J. M. (2021). General invertible transformations for flow-based generative modeling. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Tomczak, J. M. (2024). *Deep Generative Modeling*. Springer: Cham, Switzerland.

Tomczak, J. M. and Welling, M. (2016). Improving variational auto-encoders using Householder flow. *arXiv preprint arXiv:1611.09630*.

Tomczak, J. M. and Welling, M. (2017). Improving variational auto-encoders using convex combination linear inverse autoregressive flow. *arXiv preprint arXiv:1706.02326*.

Tran, B.-H., Rossi, S., Milios, D., and Filippone, M. (2022a). All you need is a good functional prior for Bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56.

Tran, D., Liu, J. Z., Dusenberry, M. W., Phan, D., Collier, M., Ren, J., Han, K., Wang, Z., Mariet, Z. E., Hu, H., et al. (2022b). Plex: Towards reliability using pretrained large model extensions. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

Tzen, B. and Raginsky, M. (2019). Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*.

van den Berg, R., Gritsenko, A. A., Dehghani, M., Sønderby, C. K., and Salimans, T. (2021). Idf++: Analyzing and improving integer discrete flows for lossless compression. In *International Conference on Learning Representations*.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. (2018). Sylvester normalizing flows for variational inference. In *UAI*, pages 393–402.

Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.

Vargas, F., Grathwohl, W., and Doucet, A. (2023). Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*.

Vargas, F., Padhy, S., Blessing, D., and Nüsken, N. (2024). Transport meets variational inference: Controlled Monte Carlo diffusions. *International Conference on Learning Representations (ICLR)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vemgal, N. M., Lau, E., and Precup, D. (2023). An empirical study of the effectiveness of using a replay buffer on mode discovery in gflownets. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*.

Venkatraman, S., Hasan, M., Kim, M., Scimeca, L., Sendera, M., Bengio, Y., Berseth, G., and Malkin, N. (2025a). Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. In *International Conference on Machine Learning*. PMLR.

Venkatraman, S., Hasan, M., Kim, M., Scimeca, L., Sendera, M., Bengio, Y., Berseth, G., and Malkin, N. (2025b). Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. In *The Frontiers in Probabilistic Inference: Sampling meets Learning (FPI) workshop at ICLR 2025*.

Venkatraman, S., Jain, M., Scimeca, L., Kim, M., Sendera, M., Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., et al. (2024a). Amortizing intractable inference in diffusion models for Bayesian inverse problems. In *Proc. Workshop on Machine Learning and the Physical Sciences. Accessed*, pages 02–06.

Venkatraman, S., Jain, M., Scimeca, L., Kim, M., Sendera, M., Hasan, M., Rowe, L., Mittal, S., Lemos, P., Bengio, E., et al. (2024b). Amortizing intractable inference in diffusion models for vision, language, and control. *Neural Information Processing Systems (NeurIPS)*.

Wang, H. and Yeung, D.-Y. (2020). A survey on Bayesian deep learning. *ACM computing surveys (csur)*, 53(5):1–37.

Wang, J., Sun, S., and Yu, Y. (2019a). Multivariate triangular quantile maps for novelty detection. In *Advances in Neural Information Processing Systems*, pages 5061–5072.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019b). Exact Gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32.

Wang, S., Zeng, Y., Liu, X., Zhu, E., Yin, J., Xu, C., and Kloft, M. (2019c). Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. In *Advances in Neural Information Processing Systems*, pages 5960–5973.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. (2020). How good is the Bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, pages 10248–10259.

Williams, C. (1996). Computing with infinite networks. In Mozer, M., Jordan, M., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press.

Wilson, A. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075. PMLR.

Wilson, A. G. and Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708.

Woo, D., Kim, M., Kim, M., Seong, K., and Ahn, S. (2025). Energy-based generator matching: A neural sampler for general state space. *arXiv preprint arXiv:2505.19646*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, J., Xu, L., and Chen, E. (2012). Image denoising and inpainting with deep neural networks. *Advances in neural information processing systems*, 25:341–349.

Yang, X. and Wang, X. (2024). Neural metamorphosis. In *European Conference on Computer Vision*, pages 1–19. Springer.

Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N. (2017). Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353.

Yu, F., Jiang, L., Kang, H., Hao, S., and Qin, L. (2024). Flow of reasoning: Efficient training of llm policy with divergent thinking. *arXiv preprint arXiv:2406.05673*.

Zhai, S., Cheng, Y., Lu, W., and Zhang, Z. (2016). Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*.

Zhang, D., Chen, R. T. Q., Liu, C.-H., Courville, A., and Bengio, Y. (2024). Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. (2022). Generative flow networks for discrete probabilistic modeling. *International Conference on Machine Learning (ICML)*.

Zhang, Q. and Chen, Y. (2022). Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*.

Zhao, M. and Saligrama, V. (2009). Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in neural information processing systems*, pages 2250–2258.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, PP:1–34.

Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. (2023). A variational perspective on generative flow networks. *Transactions on Machine Learning Research (TMLR)*.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection.

A

Full text of the publications

This chapter contains the publications described in the main text, in the following order:

[I] Maziarka, Łukasz, Marek Śmieja, **Marcin Sendera**, Łukasz Struski, Jacek Tabor, and Przemysław Spurek. "OneFlow: One-class flow for anomaly detection based on a minimal volume region." *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, no. 11 (2021): 8508-8519., CORE A*, 200 MSHE points.

[II] **Sendera, Marcin**, Marek Śmieja, Łukasz Maziarka, Łukasz Struski, Przemysław Spurek, and Jacek Tabor. "Flow-based SVDD for anomaly detection." In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+ 2020)*, 2020.

[III] **Sendera, Marcin**, Łukasz Struski, and Przemysław Spurek. "Missing Glow Phenomenon: Learning Disentangled Representation of Missing Data." In *International Conference on Neural Information Processing*, pp. 196-204. Cham: Springer International Publishing, 2021., CORE A, 140 MSHE points (when submitted and published).

[IV] **Sendera, Marcin**, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. "Improved off-policy training of diffusion samplers." *Advances in Neural Information Processing Systems 37* (2025): 81016-81045., CORE A*, 200 MSHE points.

[V] Berner, Julius*, Lorenz Richter*, **Marcin Sendera***, Jarrid Rector-Brooks, and Nikolay Malkin. "From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training." *arXiv preprint arXiv:2501.06148* (2025). *Under review.*

[VI] **Sendera, Marcin**, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzcinski, Przemysław Spurek, and Maciej Zieba. "Non-Gaussian Gaussian Processes for Few-Shot Regression." *Advances in Neural Information Processing Systems 34* (2021): 10285-10298., CORE A*, 200 MSHE points.

[VII] **Sendera, Marcin***, Marcin Przewięźlikowski*, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. "Hypershot: Few-shot learning by kernel hypernetworks." In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2469-2478. 2023., CORE A, 140 MSHE points.

[VIII] **Sendera, Marcin***, Marcin Przewięźlikowski*, Jan Miksa, Mateusz Rajski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. "The general framework for few-shot learning by kernel HyperNetworks." *Machine Vision and Applications 34*, no. 4 (2023): 53., CORE B, 100 MSHE points.

[IX] **Sendera, Marcin***, Amin Sorkhei, and Tomasz Kuśmierczyk*. "Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations." In *The 41st Conference on Uncertainty in Artificial Intelligence*. 2025., CORE A, 140 MSHE points.

* *denotes equal contribution*

OneFlow: One-class flow for anomaly detection based on a minimal volume region

Łukasz Maziarka, Marek Śmieja, Marcin Sendera, Łukasz Struski, Jacek Tabor, Przemysław Spurek

Abstract—We propose OneFlow – a flow-based one-class classifier for anomaly (outlier) detection that finds a minimal volume bounding region. Contrary to density-based methods, OneFlow is constructed in such a way that its result typically does not depend on the structure of outliers. This is caused by the fact that during training the gradient of the cost function is propagated only over the points located near to the decision boundary (behavior similar to the support vectors in SVM). The combination of flow models and a Bernstein quantile estimator allows OneFlow to find a parametric form of bounding region, which can be useful in various applications including describing shapes from 3D point clouds. Experiments show that the proposed model outperforms related methods on real-world anomaly detection problems.

Index Terms—Anomaly detection, outlier detection, normalizing flows.

1 Introduction

Anomaly (novelty/outlier) detection refers to the identification of abnormal or novel patterns embedded in a large amount of (nominal) data [1]. The goal of anomaly detection is to identify unusual system behaviors, which are not consistent with its typical state. Anomaly detection algorithms find application in fraud detection [2], discovering failures in industrial domain [3], detection of adversarial examples [4], etc. [5], [6], [7].

In contrast to typical binary classification problems, where every class follows some probability distribution, an anomaly is a pattern that does not conform to the expected behavior. In other words, a completely novel type of outliers, which is not similar to any known anomalies, can occur at a test time. Moreover, in most cases, we do not have access to any anomalies at training time. In consequence, novelty detection is usually solved using unsupervised approaches, such as one-class classifiers, which focus on describing the behavior of nominal data (inliers) [8], [9], [10], [11]. Any observation, which deviates from this behavior, is labeled as an outlier.

Following the above motivation, we propose OneFlow – a deep one-class classifier based on flow models. In contrast to typical (generative) flow-based density models, which focus on density estimation, OneFlow does not depend strongly on the structure of outliers since it finds a bounding region with a minimal volume for a fixed $(1-\alpha)$ portion of data, e.g. $\alpha = 5\%$, see Figure 1 for a comparison. This is realized by finding a hypersphere with a minimal radius in the output space of a neural network, which contains $(1-\alpha)$ percentage of data, see Figure 2. While minimum volume sets were considered previously in [12], [13] in the context of anomaly detection, these works were mainly devoted to theoretical aspects and the algorithms proposed there do not scale well to large datasets. On the other hand, in the case of kernel methods, such as Support Vector Data Description (SVDD) [14], the minimum volume problem has been reformulated to obtain a convex objective, which solves a slightly different problem. In particular, instead of enclosing $(1-\alpha)$ fraction of data within a hypersphere, SVDD adds a penalty for data points outside the hypersphere. Making use of neural networks, we do not need to stick to convex optimization and, therefore, OneFlow solves the original problem directly. In OneFlow, the gradient of the cost function is propagated only through the points located near the decision boundary (a behaviour which is similar to that of support vectors in SVM), see Remark 1. To the authors' best knowledge, this is the first model, which applies this paradigm in deep learning without any relaxations described above.

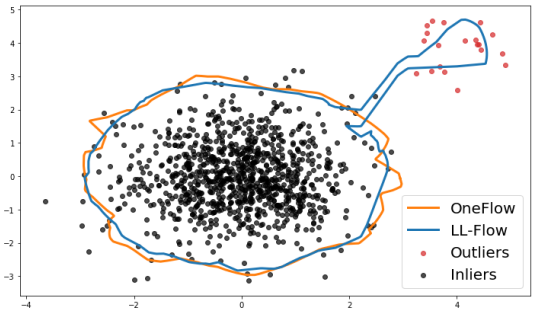


Fig. 1. Bounding regions constructed by OneFlow and typical log-likelihood flow-based density model (LL-Flow). LL-Flow puts a similar weight to both blobs and marks a few examples from the smaller one as nominal data. OneFlow finds a bounding region with a minimal volume for a fixed percentage of data. To minimize the volume it focuses on a bigger blob and considers the smaller one as anomalies.

OneFlow uses two important ingredients. The first one is the application of flow-based models [15], [16], which give an explicit formula for the inverse mapping and allow us to

• *Ł. Maziarka, M. Śmieja, M. Sendera, Ł. Struski, J.Tabor, P. Spurek are with the Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6,30-348 Kraków, Poland.*
E-mail: lukasz.maziarka@ii.uj.edu.pl, marek.smieja@uj.edu.pl

Manuscript received April 19, 2005; revised August 26, 2015.

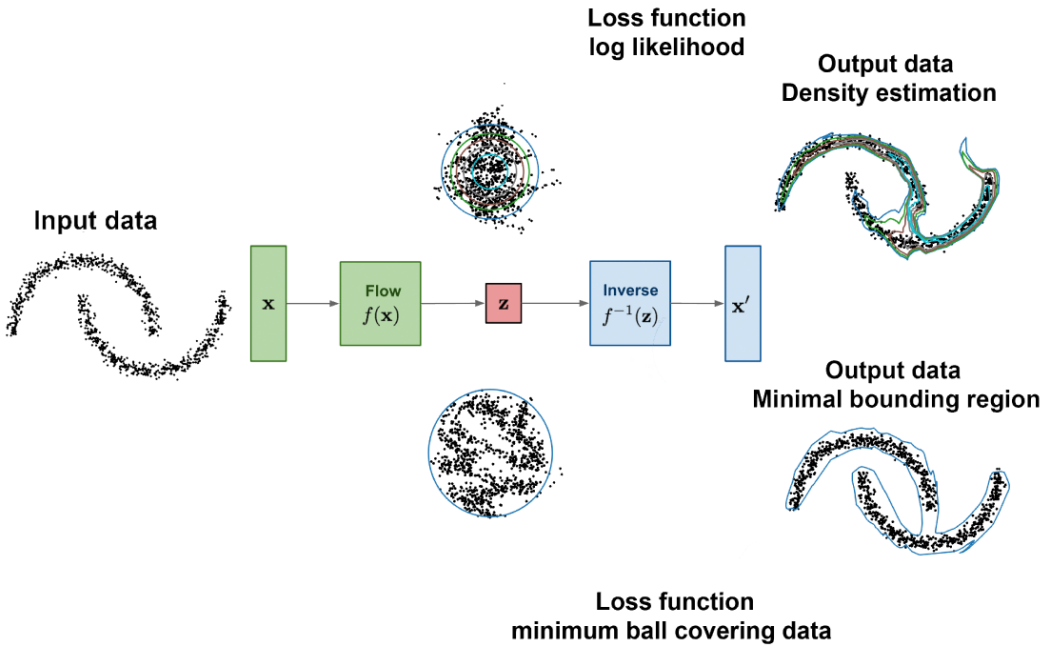


Fig. 2. OneFlow finds a bounding region with a minimal volume for a fixed percentage of data using a hypersphere in the latent space of flow model.

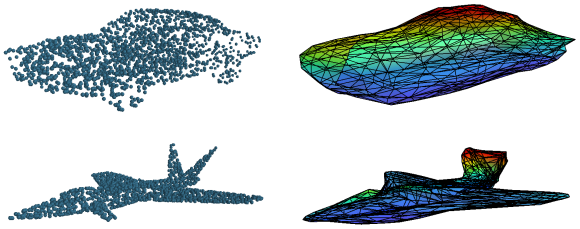


Fig. 3. Mesh representations generated by OneFlow (right) for the shapes represented as 3D point clouds (left). Our method automatically removes outliers, which may be generated from other shapes in the background, and gives an explicit parametric form of the boundary of objects using the inverse mapping of the flow model.

calculate a Jacobian of a neural network at every point. In consequence, minimizing the volume of the hypersphere in the feature space leads to the minimization of the volume of the corresponding bounding region in the input space. Moreover, making use of the inverse mapping, we automatically get a parametric form for the corresponding bounding region in the input space, which is useful, for example, in describing shapes from 3D point clouds [17], [18], see Figure 3 for details. The second ingredient is the Bernstein polynomial estimator [19] of the upper $(1-\alpha)$-quantile, which is used in OneFlow loss to estimate a hypersphere for $(1-\alpha)$ fraction of data.

We performed extensive experiments on image datasets (MNIST and Fashion-MNIST), examples retrieved from UCI [20], openML [21] repositories as well as KDD CUP 1999. Experimental results show that OneFlow gives comparative or even better performance than state-of-the-art models for anomaly detection. In particular, OneFlow outperforms typical flow-based density models, which use log-likelihood objective as well as deep SVDD method.

Our contribution is summarized as follows:

1) We introduce a one-class model for anomaly detection based on invertible flows, which focuses on finding a bounding region for nominal data.
2) We show that the application of Bernstein quantile estimator in our neural network framework allows us to estimate the volume of the bounding region for a fixed $(1-\alpha)$ percentage of data in a closed-form.
3) We experimentally analyze the behavior of the proposed approach and compare it with state-of-the-art methods.

The code is available at github.com/gmum/OneFlow.

2 Related Works

One of the most successful approaches to anomaly detection is based on one-class learning. One-class SVM (OCSVM) [8] and SVDD [14] are two well known kernel methods for one-class classification. While OCSVM directly uses SVM to separate the data from the origin (considered as the only negative sample), SVDD aims to enclose most of the data points inside a hypersphere with minimal volume. To provide a unique local minimum, SVDD relaxes this problem to the convex objective by penalizing data points outside the hypersphere. In a similar spirit, [22] apply Ranking SVM based on rankings created from pairwise comparison of nominal data. In contrast to SVDD, which reformulates minimum volume problem, [12], [13] derived algorithms, which, under some assumptions on data density, are provably optimal. However, despite obtaining important theoretical results, these methods do not scale well to large datasets. In contrast to these works, we do not use any simplifications in the cost function, but solve the minimum volume set problem efficiently.

Recent research on anomaly detection is dominated by methods based on deep learning. Attempts for adapting SVDD to the case of neural networks are presented in [23], [24], [25], [26]. However, the direct minimization of SVDD loss may lead to hypersphere collapse to a single point. To avoid this negative behavior, DSVDD (deep SVDD) recommends that the center must be something other than the all-zero-weights solution, and the network should use only unbounded activations and omit bias terms [23]. While the first two conditions can be accepted, omitting bias terms in a network may lead to a sub-optimal feature representation due to the role of bias in shifting activation values. To eliminate these restrictions a recent work [26] proposes two regularizers, which prevent from hypersphere collapse, and use an adaptive weighting scheme to control the amount of penalization between the SVDD loss and the respective regularizer. Making use of flow models, we are able to directly calculate the volume of the bounding region in the original space (not only the volume of the transformation in the latent space). Due to the direct correspondence between original density and prior density given by a flow model, we avoid degenerate solutions, which may appear in DSVDD models.

The vast majority of deep learning methods use neural network representation learning capability to generate a latent representation to preserve the details of the given class based on auto-encoder reconstruction error [9], [10], [27]. This line of research includes strictly unsupervised techniques [11] as well as supervised and semi-supervised methods [28]. Nevertheless, there is no theoretical justification that reconstruction error captures enough information to separate nominal data from outliers. Many research proposes GAN-based approaches as the criterion considered introduced anomaly score [29], [30], sample's representation in the discriminator's latent space [31], [32] or even separability of the enhanced inliers and distorted outliers [33]. Another direction of related work is the discovery of out-of-distribution instances (which are basically anomalies). Various forms of thresholding are used on the classification output to detect anomalies [34], [35], [36]. [37] defined a general approach for anomaly detection, which is based on thresholding multivariate quantile function. Analogically to our approach, they use flow-based models, but in the context of density estimation. Another use of density-based flow models is presented in [38]. Moreover, there are methods, which uses flow-based models with their ability to density estimation as key part of anomaly detection models - [39], [40]. A comprehensive review on deep anomaly detection can be found in [41], [42].

3 The Proposed Model

In this section, we first give a precise formulation of outlier detection task using a bounding region with minimum volume. Next, we present our solution based on flow models. Finally, we discuss optimization issues, which are solved using the Bernstein quantile estimator. The notation used in the following section is summarized in Appendix A.

3.1 Problem formulation.

Our goal is to find a bounding region with a minimal volume, which contains a fixed amount of data, e.g. 95%. This refers to one of the typical ideas used in one-class classification [12], [13], [14], where we describe the behavior of nominal data. Ignoring a small number of data allows us to deal with anomalies in training as well as to focus only on the most important features. In contrast to density-based approaches, which estimate a density of the whole data, we solve the easier task by separating nominal data from abnormal examples.

Let g be a density in $\mathbb{R}^D$, and let $X \subset \mathbb{R}^D$ be a sample generated by g. We assume that we are given a p-value $\alpha \in (0,1)$, which determines the percentage of possible outliers, which can also be seen as a false alarm rate[1]. We say that $U \subset \mathbb{R}^D$ is a $(1-\alpha)$-bounding region of g if a data point generated from a density g belongs to U with a probability $1-\alpha$, i.e. $\int_U g(x)\,dx = 1-\alpha$. Intuitively, the $(1-\alpha)$-bounding region covers approximately $(1-\alpha)$ percentage of data, which allows us to deal with outliers or noise in training data.

Our problem is formally formulated below:

Problem 1. Find a $(1-\alpha)$-bounding region $U \subset \mathbb{R}^D$ with a minimal volume for a density g generating data, i.e.

$$\arg\min_U \big\{\mathrm{vol}(U) \,|\, U : \int_U g(x)\,dx = 1-\alpha\big\}.$$

To allow for sufficient flexibility in defining the form of U, we use deep neural networks. Given a neural network $f : \mathbb{R}^D \to \mathbb{R}^D$, we aim at finding $r > 0$ such that $U := f^{-1}(B(0,r))$, where $B(0,r)$ denotes a ball centered at 0 with radius r in the latent (feature) space of the neural network. In other words, the $(1-\alpha)$-bounding region U is the inverse of a ball with radius r in the feature space. While the computation of f^{-1} can be difficult for arbitrary neural networks, we restrict our attention to flow-based models, which give an explicit form of inverse mapping f^{-1}.

First, we demonstrate that the volume $\mathrm{vol}(f^{-1}(B(0,r)))$ can be calculated efficiently for flow-based models. Next, we show that the application of Bernstein estimator allows us to find a hypersphere for $(1-\alpha)$ percentage of data, which is the solution of our optimization problem.

3.2 Volume calculation using flow-based models.

Let us recall that a neural network $f : \mathbb{R}^D \to \mathbb{R}^D$ is a flow-based model if the inverse mapping f^{-1} is given explicitly and the Jacobian determinant $\det df(x)$ can be easily calculated. Flow-based models have been usually used in the case of generative models because a direct form of f^{-1} allows one to generate samples from the prior distribution, while the condition for Jacobian makes the optimization of log-likelihood function possible. Their direct application in the context of anomaly detection can be compared to the use of GMMs. Given a distribution of data, we discard α percentage of data or a region of data space with a probability α. Since we want to realize a different objective, we need to redefine the loss for flow-based models.

1. If not stated otherwise, we use $\alpha = 5\%$, which is motivated by a typical approach used in hypothesis testing.

As mentioned, flow-based models are designed to calculate the Jacobian of f effectively, which allows us to optimize the log-likelihood function in the case of neural networks directly. From this perspective, flow-based models can be divided into two natural classes. The first class, referred to as *const-det flows*, contains models where $\det df(x)$ is constant (does not depend on x), e.g. NICE [15]. The models from the second class, called here *general flows*, can change the derivative at different points, e.g. Real NVP [43].

We show that for const-det flows we can obtain the exact formula for $\mathrm{vol}(f^{-1}(B(0,r)))$, while for general flows its approximation can be derived. For this purpose, we introduce a notation:

$$w(x) = \det d(f^{-1})(x) = \frac{1}{\det df(f^{-1}(x))}.$$

Note that for const-det flows, $w(x) = w$ is a constant function.

Observation 1. *Let f be a const-det flow model, i.e. $w = w(x)$ is constant. The volume of $U = f^{-1}(B(0,r))$ is given by:*

$$\mathrm{vol}(U) = \mathrm{vol}(B(0,1)) \cdot w \cdot r^D. \tag{1}$$

Proof. Clearly, by a change of variables,

$$\mathrm{vol}(U) = \mathrm{vol}(f^{-1}(B(0,r))) = \int_{B(0,r)} \det d(f^{-1})(x)\, dx.$$

Since w is a constant function, we get $\mathrm{vol}(U) = \int_{B(0,r)} w(x)\, dx = \mathrm{vol}(B(0,1)) \cdot w \cdot r^D$. □

If $w(x)$ depends on x, then the situation is more complex, but we can still obtain an approximation of the volume for general flows as

$$\begin{aligned}\mathrm{vol}(U) = \int_{B(0,r)} w(x)dx &\approx \mathrm{vol}(B(0,r)) \cdot \frac{1}{m}\sum_{k=1}^{m} w(r \cdot e_i) \\ &= \frac{\mathrm{vol}(B(0,1))}{m} \cdot r^D \cdot \sum_{k=1}^{m} w(r \cdot e_i),\end{aligned} \tag{2}$$

where e_i are points randomly chosen with respect to uniform distribution on $B(0,1)$. This is a type of the Monte Carlo sampling and it is generally difficult to control the accuracy of this estimation

3.3 Optimization algorithm.

We presented how to find formulas for computing the volume of the bounding region using flow-based models. Now, we apply this fact to construct the optimization procedure for computing $(1-\alpha)$-bounding region.

Let $(x_1, \ldots, x_n) \subset X$, be a mini-batch, θ be the weights of the flow model f_θ and $\alpha \in (0,1)$ be a given p-value. To apply the formula (1) for const-det flow, we need to find the radius of the ball, which contains $(1-\alpha)$ percentage of data. We estimate this radius by first computing

$$r_i(\theta) = \|f_\theta(x_i)\|,$$

and next applying the estimator of upper $(1-\alpha)$-quantile.

As a quantile estimator, we use Bernstein polynomial estimator [19], [44], [45] – see Remark 2 for the justification of this selection. Let us recall that given a sample $s_1, \ldots, s_n$ drawn from the same distribution, the Bernstein estimation of $(1-\alpha)$-quantile s_α, where $\alpha \in (0,1)$, is constructed in the following way. First, we reorder s_i so that $s_{(1)} \leq \ldots \leq s_{(n)}$. Then the Bernstein estimator of $(1-\alpha)$-quantile is defined by:

$$s_\alpha = \sum_{k=1}^{n} \binom{n-1}{k-1} \alpha^{k-1}(1-\alpha)^{n-k} \cdot s_{(k)}$$

Bernstein polynomials are known to yield very smooth estimates, even from the small sample size, that typically have acceptable behavior at the boundaries.

Applying the above construction to our case, we do as follows:

- the sequence $(r_{(i)}(\theta))$ is obtained by sorting $r_i(\theta)$ in a descending order,
- the Bernstein polynomial estimator of upper $(1-\alpha)$-quantile is given by

$$R_\alpha(\theta) = \sum_{k=1}^{n} \binom{n-1}{k-1} \alpha^{k-1}(1-\alpha)^{n-k} \cdot r_{(k)}(\theta). \tag{3}$$

In consequence, the volume of the bounding region for const-det flows is given by

$$\mathrm{vol} = w \cdot \mathrm{vol}(B(0,1)) \cdot R_\alpha(\theta)^D.$$

To avoid potential numerical problems in the cost function, one can minimize logarithm of the volume (instead of the volume itself):

$$\mathrm{cost}(\theta) = \log(\mathrm{vol}(B(0,1))) + \log(w) + D \log R_\alpha(\theta).$$

For general flows, we use the formula (2) in the above calculations. Thus the estimation for the volume of the $(1-\alpha)$-bounding region is given by

$$\mathrm{vol} \approx \tfrac{\mathrm{vol}(B(0,1))}{m} \cdot R_\alpha^D(\theta) \cdot \sum_{k=1}^{m} w(R_\alpha(\theta)e_k),$$

where (e_k) is a sequence of m randomly chosen points from the uniform distribution on the unit ball $B(0,1)$. The final cost function in the logarithmic form equals:

$$\begin{aligned}&\mathrm{cost}(\theta) = \\ &\log(\tfrac{\mathrm{vol}(B(0,1))}{m}) + D \log R_\alpha(\theta) + \log\big(\sum_{k=1}^{m} w(R_\alpha(\theta)e_k)\big).\end{aligned}$$

Remark 1. We explain now that, in contrast to flow-based density models, the gradient of our loss function is propagated only over a small number of points, which are located close to the decision boundary. For $n\alpha > 9$ (e.g. for $\alpha = 0.05$ and $n = 256$), the Bernoulli distribution can be approximated by the Gaussian distribution $N(m, \sigma^2)$ with $m = n\alpha$ and $\sigma^2 = n\alpha(1-\alpha)$. Thus by the 3σ law, we obtain that numerically essential weights are only for k examples, where $k \in [n\alpha - 3\sqrt{n\alpha(1-\alpha)}, n\alpha + 3\sqrt{n\alpha(1-\alpha)}]$. Consequently, we obtain that only the following percentage of samples from the batch obtain nonzero gradient:

$$\frac{6\sqrt{n\alpha(1-\alpha)}}{n} = \frac{3\sqrt{19}}{160} \approx 0.08,$$

where $6\sqrt{n\alpha(1-\alpha)}$ is the length of the above interval.

Remark 2. The simplest estimator of the $(1-\alpha)$-quantile for the sorted sample $s_{(1)} \leq \ldots \leq s_{(n)}$ can defined by:

$$s_\alpha^{sim} = \begin{cases} s_{(\alpha n)}, & \text{if } \alpha n \text{ is an integer,} \\ s_{([\alpha n]+1)}, & \text{otherwise,} \end{cases}$$

where $[a]$ is the greatest integer which is not greater than $a \in \mathbb{R}$. If such an estimator was used in our loss function, then the gradient would be based only on a single point that establishes the gradient. This may result in a very slow and inefficient training. In the case of Bernstein polynomial estimator, the gradient of the loss function in OneFlow is propagated over a number of points, which are located close to the decision boundary (see Remark 1) – the behavior that is similar to SVM models. Another thing is that the Bernstein estimator is smooth [45, Section 2.3], while the simple estimator s_α^{sim} is not even continuous.

It is also instructive to see the difference between OneFlow and typical flow-based method applying log-likelihood loss. Since the log-likelihood method focuses on a density estimation, the gradient is based on all points, which could lead to an overfitting and not optimal setting of the border. Density estimation can be seen as an additional task in anomaly detection, while the essential problem is to estimate the quantile, which is directly solved by OneFlow.

4 Experiments

In this section, we experimentally examine OneFlow and compare it with several state-of-the-art approaches. OneFlow is implemented using the architecture of NICE flow model and $\alpha = 5\%$ (see Appendix B for the experimental setting). An analysis of parameter α is presented at the end of this section. If not stated otherwise, we consider a variant of const-det flow (Jacobian determinant is constant).

4.1 Benchmark data for anomaly detection.

First, we provide a quantitative assessment and take into account the Thyroid[2] and KDDCUP[3] datasets, which are real-world benchmark datasets for anomaly detection. We use the standard training and test splits and follow exactly the same evaluation protocol as in [37]. The performance was measured using F1 score, because this metric was reported for all methods considered.

We use two variants of OneFlow. The first one (OneFlow) uses constant Jacobian while the second (OneFlow-Gen) allows for changing the Jacobian at every point. Our models are compared with the following algorithms (see Appendix C for more detailed description): One-class SVM (OC-SVM) [8], Deep structured energy-based models (DSEBM) [46], Deep autoencoding Gaussian mixture model (DAGMM) [47], variants of MQT – multivariate quantile map (NLL, TQM_1, TQM_2, TQM_∞) [37], Deep Support Vector Data Description (DSVDD) [23], two variants of log-likelihood flow model (LL-Flow and LL-Flow-Gen). The bounding region of LL-Flow is constructed by taking the smallest hypersphere in the latent space, which covers $(1-\alpha)$ percentage of data (the hypersphere is determined by the prior Gaussian distribution). Due to the change of variable rule used in flow models, this strategy corresponds to thresholding the density in the original space so that to cover $(1-\alpha)$ percentage of data.

The results presented in Table 1 show that both variants of our model perform almost equally on KDDCUP and they are better than all competitive methods. In the case of Thyroid, OneFlow presents the best performance, while OneFlow-Gen gives third best score. The most similar method, DSVDD, was not able to obtain similar performance on these datasets, which shows that a direct minimization of the bounding region implemented by our method is more beneficial. While the results of Triangular Quantile Maps (TQM and NLL) depends heavily on the assumed norm and cost function, there is no objective criteria for selecting these parameters. Other methods produce worse results.

4.2 Image datasets

To provide further experimental verification, we use two image datasets: MNIST and Fashion-MNIST. In contrast to the previous comparison, these two datasets are usually used for multiclass classification and thus need to be adapted to the problem of anomaly detection. For this purpose, each of the ten classes is deemed as the nominal class while the rest of the nine classes are deemed as the anomaly class, which results in 10 scenarios for each dataset. To be consistent with [37], we report AUC (area under ROC curve).

We additionally compare with the following models: Geometric transformation (GT) [48], Variational autoencoder (VAE) [49], Denoising autoencoder (DAE) [50], Generative probabilistic novelty detection (GPND) [51], Latent space autoregression (LSA) [9]. In contrast to previous experiment, we only use TQM_2 and NLL as the only implementations of MTQ, because they output the highest value of AUC [37].

To present the results, we compute the ranking on each of 10 scenarios and summarize it using box plot, see Figure 4 (detailed results are included in Appendix D in Tables 6 and 7). It is evident that OneFlow and OneFlow-Gen outperform related LL-Flow and LL-Flow-Gen, which confirms that the proposed loss function suits better for one-class classification problems than typical log-likelihood function. Our methods give also better scores than DSVDD, which implements a similar loss function. The overall ranking of OneFlow and OneFlow-Gen is comparative to the best performing methods on both datasets. It is difficult to clearly determine which method performs best, because of the high variation in the results. While GPND seems to outperform other methods on MNIST, its result on Fashion-MNIST is similar to OneFlow. We emphasize, however, that both MNIST and Fashion-MNIST do not represent typical anomaly detection datasets.

Next, we analyze which samples from the nominal class are localized close to or furthest from the center of bounding hypersphere. It is evident from Figure 5 that OneFlow maps images with regular structure, which are easy to recognize, in the hypersphere center. On the other hand, examples localized far from the center (outside the bounding region) do not look visually plausible and one cannot be sure about their class. It means that OneFlow gives results consistent with our intuition.

2. http://odds.cs.stonybrook.edu/thyroid-disease-dataset/

3. http://kdd.ics.uci.edu/databases/kddcup99/kddcup.testdata.unlabeled_10_percent.gz

TABLE 1
Performance on two anomaly detection datasets (measured by Precision, Recall and F1 score). The results marked with * are taken from [37].

	Thyroid			KDDCUP		
	Precision	Recall	F1	Precision	Recall	F1
OC-SVM *	.3639	.4239	.3887	.7457	.8523	.7954
DSEBM *	.0404	.0403	.0403	.7369	.7477	.7423
DAGMM *	.4766	.4834	.4782	.9297	.9442	.9369
DSVDD	.6989	.6989	.6989	.6898	.7055	.6975
NLL *	.7312	.7312	.7312	.9622	.9622	.9622
TQM_1 *	.5269	.5269	.5269	.9621	.9621	.9621
TQM_2 *	.5806	.5806	.5806	.9622	.9622	.9622
TQM_∞ *	.7527	.7527	.7527	.9622	.9622	.9622
LL-Flow	.6989	.6989	.6989	.6782	.6524	.6650
LL-Flow-Gen	.6808	.6955	.6881	.9268	.9268	.9268
OneFlow	**.7634**	**.7634**	**.7634**	.9702	.9702	.9702
OneFlow-Gen	.7097	.7097	.7097	**.9712**	**.9712**	**.9712**

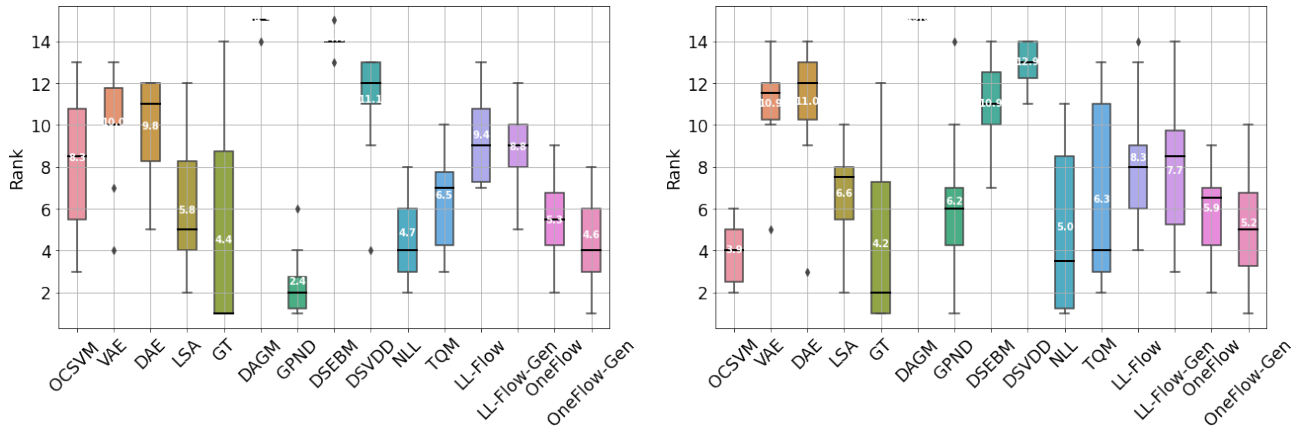


Fig. 4. Box plots for rankings calculated on MNIST (left) and Fashion-MNIST (right) using AUC score. The median ranking is marked by a line, while the average ranking is marked with a number. The results of cometitive methods (except DSVDD, LL-Flow and LL-Flow-Gen) are taken from [37].

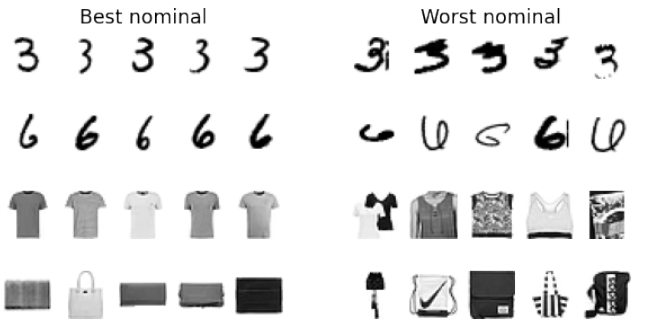


Fig. 5. Best nominal (left) and worst nominal (right) examples determined by OneFlow for MNIST (top) and Fashion-MNIST (bottom).

To give a better insight, we calculate AUC for every pair of classes, see Figures 6 and 7. More precisely, every entry of the heatmap shows AUC obtained for a given nominal class listed in a column and a given anomaly class listed in a row. For example, it occurs that OneFlow and OneFlow-Gen have the biggest problems with detecting anomalies represented by class "1" when trained on class "8". Generally, it is evident that the class "1" is the hardest to describe by our model. It may be explained by the fact that handwritten digit "1" can be written in various styles, which makes it similar to other classes.

4.3 PIDForest benchmark

We make additional benchmarks following the experimental setting of [52]. More specifically, we test OneFlow and OneFlow-Gen on the following eight datasets from the UCI [20], openML repository [21] and KDD Cup 1999: Thyroid, Mammography, Seismic, Satimage-2, Vowels, Musk, http, smtp.

For a comparison, we use the following methods: PIDForest [52], IsolationForest (iForest) [53], Robust Random Cut Forest (RRCF) [54], Local Outlier Factor (LOF) [55], k-Nearest Neighbour (kNN), Principal Component Analysis (PCA) [56]. Every method was trained on the whole dataset (outliers included) and evaluation measure was reported on the same set. To reproduce the results from [52], we use AUC as the evaluation measure. Figure 8 (left) presents box plot of ranks, which summarizes the experiment (Table 4 from Appendix D contains detailed scores). While the performance of OneFlow and OneFlow-Gen in overall is slightly worse than the best algorithms (PIDForest and iForest), it outperforms these methods on some of the datasets (Satimage, Vowels, http).

Let us recall that AUC does not take into account a single decision boundary, but effectively considers all possible test

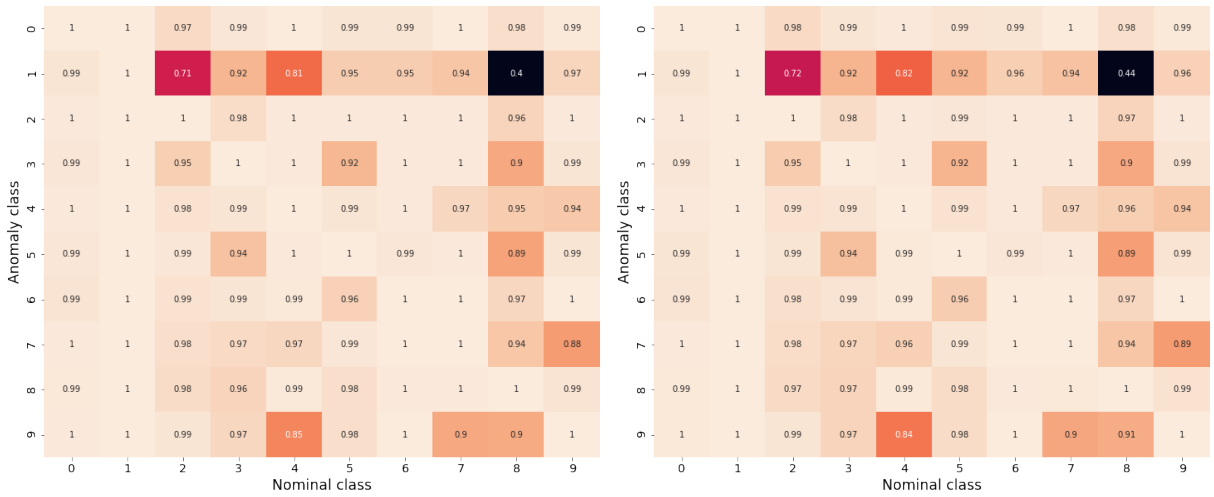


Fig. 6. AUC obtained for one nominal class (in columns) and one anomaly class (in rows) by OneFlow (left) and OneFlow-Gen (right) on MNIST dataset.

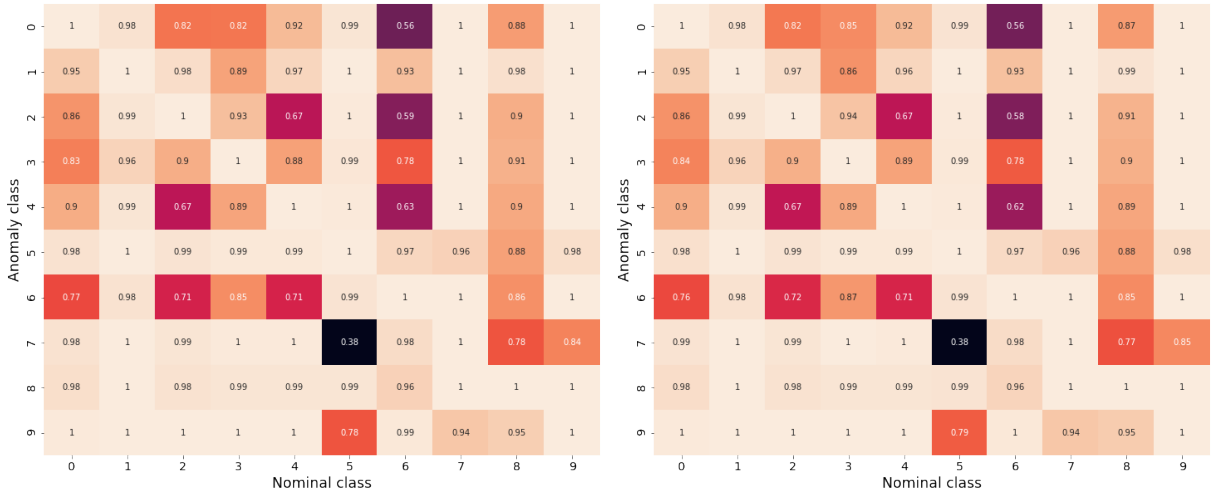


Fig. 7. AUC obtained for one nominal class (in columns) and one anomaly class (in rows) by OneFlow (left) and OneFlow-Gen (right) on Fashion-MNIST dataset.

set thresholds. In the production environment, we need to have a classification rule and verify which algorithm detects outliers and nominal data correctly. For this reason, we also test all algorithms in a discriminative setting, in which 5% of farthest examples (according to a given loss function) are deemed as anomalies. The results are evaluated using the F1 score. We present rank plot in Figure 8 (right) (detailed scores are reported in Appendix D in Table 5). One can see, that the performance of OneFlow is comparable to the best algorithms (again PIDForest and iForest). OneFlow or OneFlow-Gen is better than PIDForest and iForest in 4 out of 8 datasets, performs the same in 2 out of 8 datasets and is worse in 2 out of 8 datasets. This experiment confirms that the proposed method is better at finding outliers, which is not the same as ranking elements according to the loss function, but is crucial in practice.

4.4 Test on Out-Of-Distribution dataset

We now focus on comparing OneFlow with LL-Flow, which represents its natural baseline, and follow the experiment recently suggested in [57]. In this setting, each model is trained on the Fashion-MNIST train set with $\alpha = 5\%$. Next, we test these models on the data coming from test sets of both MNIST and Fashion-MNIST. We expect that the models will be able to classify MNIST examples as anomalies.

Figure 9 illustrates the distance of latent representations from the center of bounding hypersphere. As expected, in both cases, Fashion-MNIST (nominal) data are localized closer to the center than MNIST (outliers) data, which is correct behavior. However, OneFlow maps out-of-distribution data (MNIST) much further from the center than LL-Flow (see the range on x-axis). We verified that the percentage of correctly classified anomalies equals 92.47% for OneFlow and 89.59% for LL-Flow, which means that the discriminative power of OneFlow is higher than capabilities of LL-Flow. We also performed the second experiment when

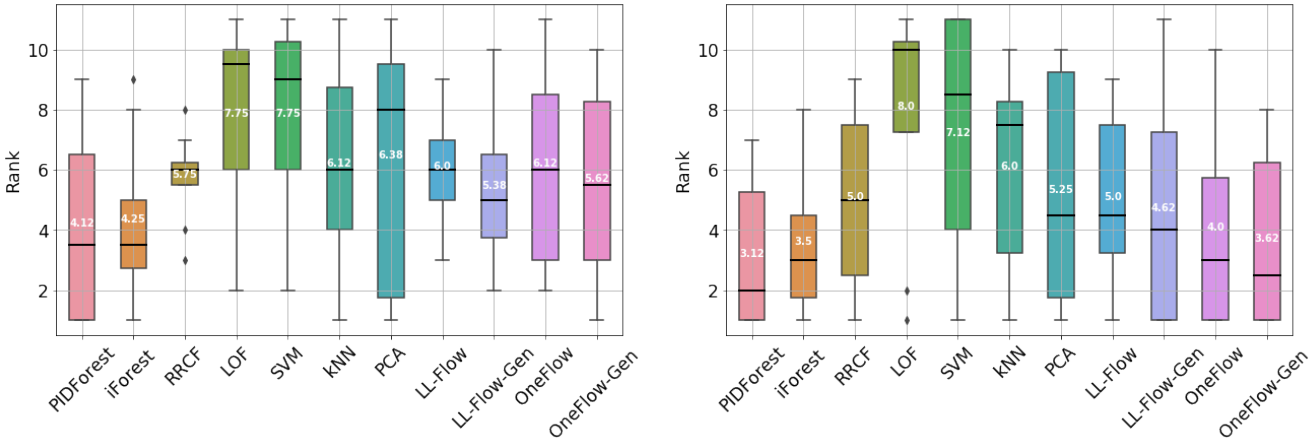


Fig. 8. Box plots of ranks calculated using AUC (left) and F1 (right) on datasets from PIDForest benchmark. The median ranking is marked by a line, while the average ranking is marked with a number. The results of comparative methods (except LL-Flow and LL-Flow-Gen) are taken from [52].

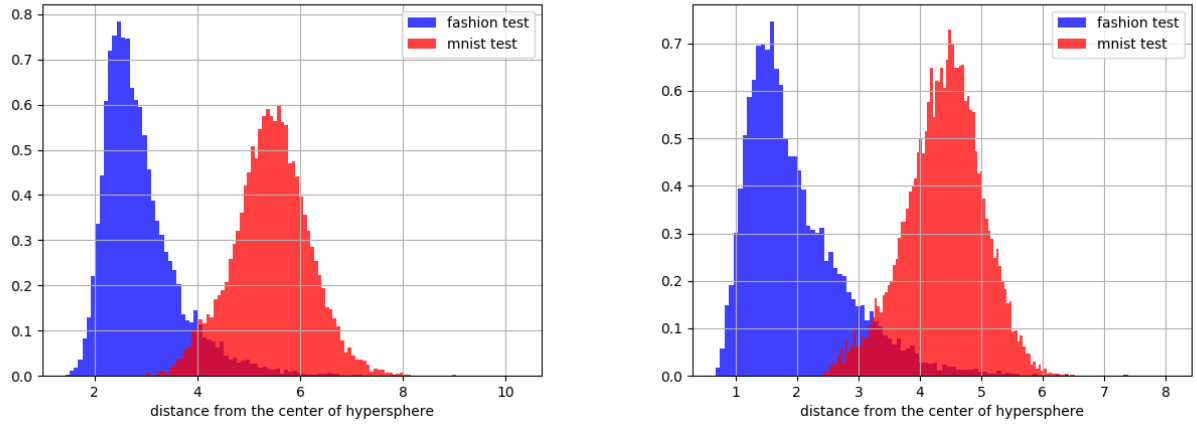


Fig. 9. Distance of Fashion-MNIST nominal data (blue) and MNIST (representing outliers) data (red) from the center of bounding hypersphere in the latent space of OneFlow (left) and LL-Flow (right). The percentage of detected outliers equals 92.47% for OneFlow and 89.59% for LL-Flow.

MNIST was considered as nominal data and Fashion-MNIST represented outliers and both models obtained almost perfect performance in this situation.

4.5 Illustrative examples

To find key differences between OneFlow and LL-Flow, we consider 2-dimensional examples, which are easy to visualize and represent a typical benchmark for comparing anomaly detection algorithms (additional illustrative examples on 3D point clouds are presented in Figure 3).

At first glance, both flow models give similar results in most cases, see Figure 10. However, a closer inspection reveals that the decision boundaries created by OneFlow are smoother and shorter than the ones resulted from LL-Flow. While minimizing the area of the bounding region should lead to a more accurate description of the nominal class, minimizing the length of the decision boundary reduces the model complexity. Making an analogy with typical supervised models, smooth, short, and simple decision boundaries usually increases the generalization performance of the model to unseen examples. To confirm this observation, we calculate the volume of the bounding region and the length of the corresponding decision boundary, see Table 2. It is evident that these quantities are smaller in the case of OneFlow.

A notable difference between both models can be seen in a doughnut shape distribution. LL-Flow is trying to remove anomalies from the centre of the doughnut, however is too rigid, which causes the long and sharp decision boundary. On the other hand, OneFlow optimizes the length of the decision boundary and treats points on the doughnut's outer edge as outliers. This behavior is typical for OneFlow's logic, because our model is not interested in density estimation.

Another example, in which the behavior of OneFlow and LL-Flow is different, is shown in Figure 1, where a dataset consists of two diverse blobs – the one with 98% of data and the second with remaining 2% of data. While LL-Flow focuses on the whole data and considers a few examples from the smaller blob as nominal data, OneFlow directly solves a one-class problem and deems the whole smaller blob as anomalies. It shows that OneFlow is not very

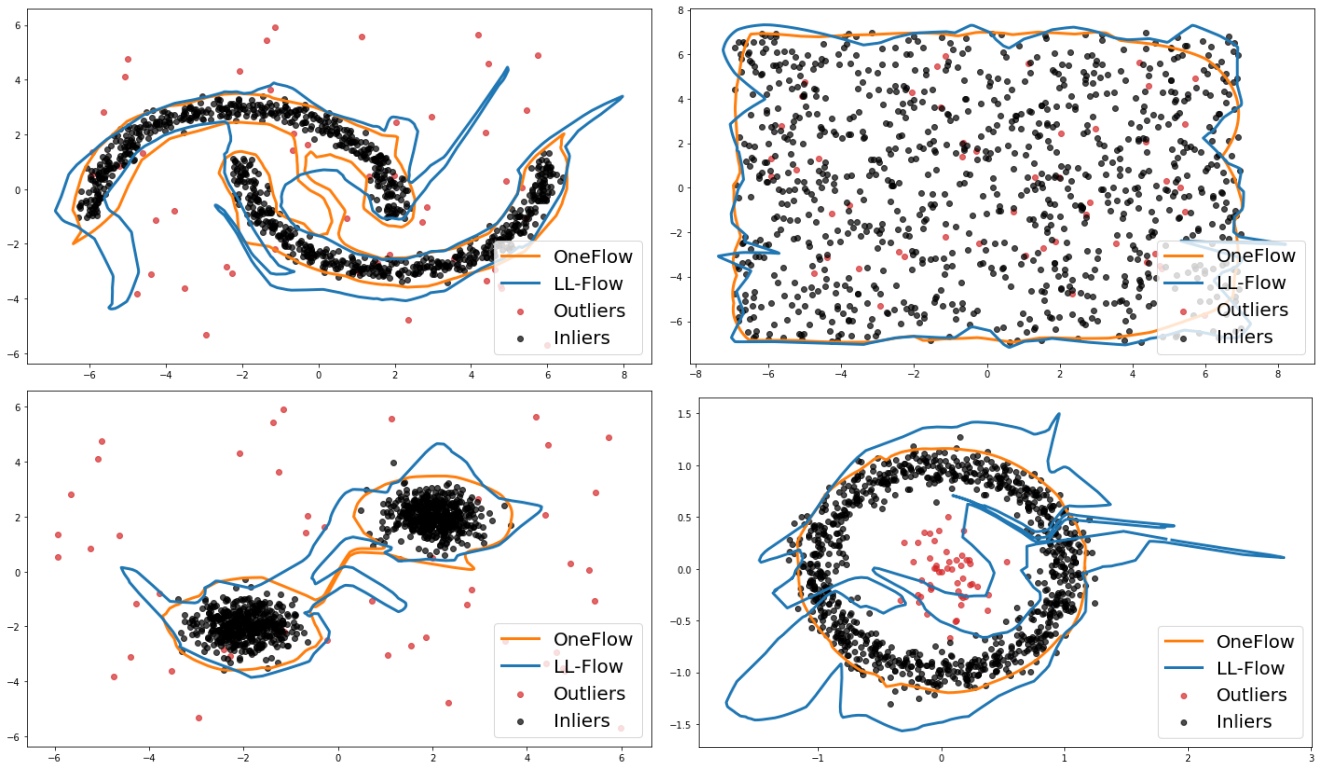


Fig. 10. Comparing bounding regions constructed by the proposed OneFlow with a flow-based density model (LL-Flow).

TABLE 2
Comparing the length of decision boundaries and the volume of bounding regions constructed by OneFlow and LL-Flow for 2D examples (lower is better).

Dataset	Length		Volume	
	OneFlow	LL-Flow	OneFlow	LL-Flow
Two Moons	65.401	84.015	31.410	41.533
Big Uniform	52.015	68.076	182.702	184.431
Two Blobs	26.739	36.863	16.749	21.058
Doughnut	7.482	27.205	4.316	5.435
Diverse Blobs	19.754	26.685	25.661	25.823

sensitive to the structure and the distribution of anomalies, because they are automatically ignored in a training phase. On the other hand, LL-Flow fits a prior density to the whole data, and, in consequence, a distribution of anomalies has an influence on the final results. Analogical behavior of both models was observed when we changed the proportions of clusters. Indeed, OneFlow always deemed smaller cluster as anomalies if it contains at most 5% of the whole data (larger clusters cannot be considered as anomalies in practice). In contrast, LL-Flow could not separate the small clusters from nominal data in this case.

4.6 Analysis of parameter α

Previous experiments were performed for OneFlow with $\alpha = 5\%$. A natural question is: what is the influence of α on the behavior of OneFlow?

To partially answer this question, we corrupt a training nominal data with anomalies. We consider 5 noise levels with: 0%, 0.1%, 1%, 5% and 10% of anomalies in a training set. In each case, we run OneFlow with $\alpha = 1\%, 5\%, 10\%$. Evaluation on test set remains exactly the same as before. We take into account MNIST and Fashion-MNIST datasets. For a comparison, we also included LL-Flow.

It is clear from Figure 11 that the performance of OneFlow slightly deteriorates as the number of anomalies in training set increases regardless of the value of α. Moreover, the model with a high value of α is able to deal with a large number of anomalies in training better than the model with small α. Indeed, OneFlow with $\alpha = 10\%$ leaves approximately 10% of data outside the bounding region and thus can still provide a good description of nominal data as long as the number of anomalies does not exceed 10%. Moreover on almost all settings OneFlow outperforms LL-Flow, being worse only in case of Fashion-MNIST with 10% anomalies in the training dataset.

Another observation is that OneFlow with small α works better for MNIST than for Fashion-MNIST when the number of anomalies is low (less than 1%). It may be explained by the fact that MNIST is a relatively simple dataset and almost all examples from each class are similar. In consequence, OneFlow with $\alpha = 1\%$ performs better than with $\alpha = 10\%$ for negligible amount of anomalies in training. On the other hand, the variation in each class of Fashion-MNIST is greater (in the noiseless case, AUC for Fashion-MNIST is 4 percentage points lower than for MNIST) so the bounding region created for $\alpha = 1\%$ is too loose. In the test phase, such a bounding region may contain too many anomalies.

The above analysis suggests that α should be large if the underlying anomaly detection task is hard or we have many

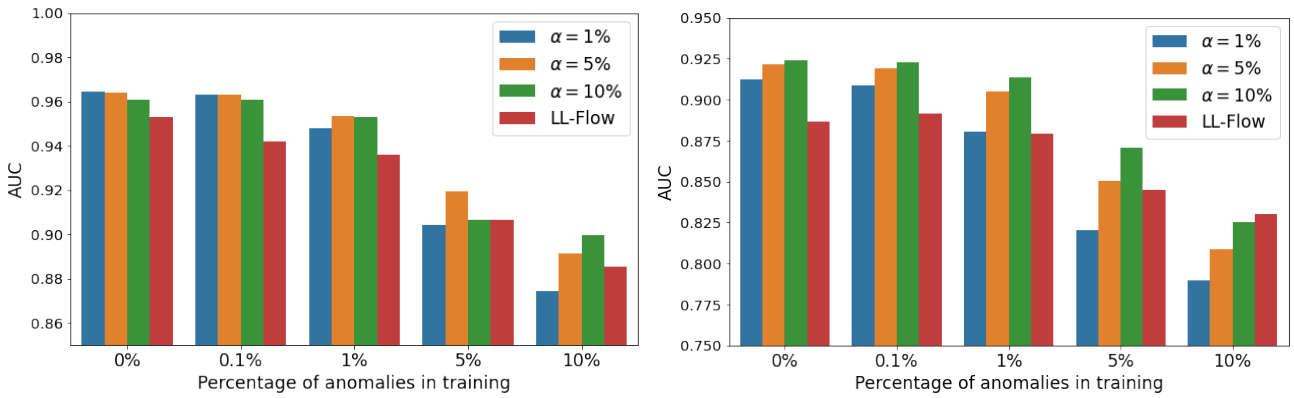


Fig. 11. Evaluation of OneFlow with three levels of $\alpha = 1\%, 5\%, 10\%$ on MNIST (left) and Fashion-MNIST (right), in which nominal class is corrupted with anomalies at training time.

anomalies in training. Otherwise, we should keep α low.

4.7 Analysis of training epochs number

In the following subsection we show scores obtained by OneFlow using different number of training epochs. It is evident from Figure 12 that 1000 epochs that we used in our benchmarks provides good trade off between training time and obtained performance. Indeed for both MNIST as well as Fashion-MNIST datasets obtained ROC AUC grows until it reaches 1000 epochs and then flats out.

4.8 Analysis of backbone model

In our experiments we used a Nice-based [15] architecture as our backbone flow model. In order to check whether the change of backbone would improve the OneFlow results, we replaced Nice with Real-NVP [43] and Glow [16] architectures, and tested the obtained models on MNIST and Fashion-MNIST datasets.

The results presented in Table 3 show that changing the Nice architecture to the more sophisticated ones significantly worsens the performance of OneFlow. This could be caused by the fact that these models are too complex for the task of outlier detection on MNIST and Fashion-MNIST. Indeed, OneFlow trained with Nice backbone obtained results close to the state-of-the-art on this task. Considering this and the fact that Real-NVP and Glow are designed to operate on image data, we decided to use Nice as a default backbone model in OneFlow. More detailed results could be found in Appendix D in Tables 6 and 7 (models are signed as OF-RNVP and OF-Glow).

TABLE 3
Comparison of the flow backbone model for OneFlow. Real-NVP and Glow are more sophisticated flow architectures designed to work on images.

	MNIST	Fashion-MNIST
OneFlow	.964	.922
Real-NVP	.683	.769
Glow	.895	.775

5 Conclusion

The paper introduced OneFlow, which realizes a well-known one-class paradigm using deep learning tools. Making use of a flow-based model and a Bernstein quantile estimator, we find a minimal volume bounding region for a given percentage of data. On the one hand, the constructed bounding region does not depend on the structure of outliers as in the density-based models, while, on the other hand, the bounding region is given by an explicit parametric form. Experimental results demonstrate that OneFlow presents state-of-the-art performance.

Acknowledgments

The work of L. Maziarka was supported by the National Science Centre (Poland) grant no. 2018/31/B/ST6/00993. The work of M. Śmieja, Ł. Struski were supported by the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund in the POIR.04.04.00-00-14DE/18-00 project is carried out within the Team-Net programme. The work of J. Tabor was supported by the National Science Centre (Poland) Grant No. 2017/25/B/ST6/01271. The work of P. Spurek was supported by the National Science Centre (Poland) Grant No. 2019/33/B/ST6/00894.

References

[1] D. Miljković, "Review of novelty detection methods," in *The 33rd International Convention MIPRO*. IEEE, 2010, pp. 593–598.
[2] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *arXiv preprint arXiv:1009.6119*, 2010.
[3] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 38–44.
[4] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," *arXiv preprint arXiv:1902.04818*, 2019.
[5] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.
[6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

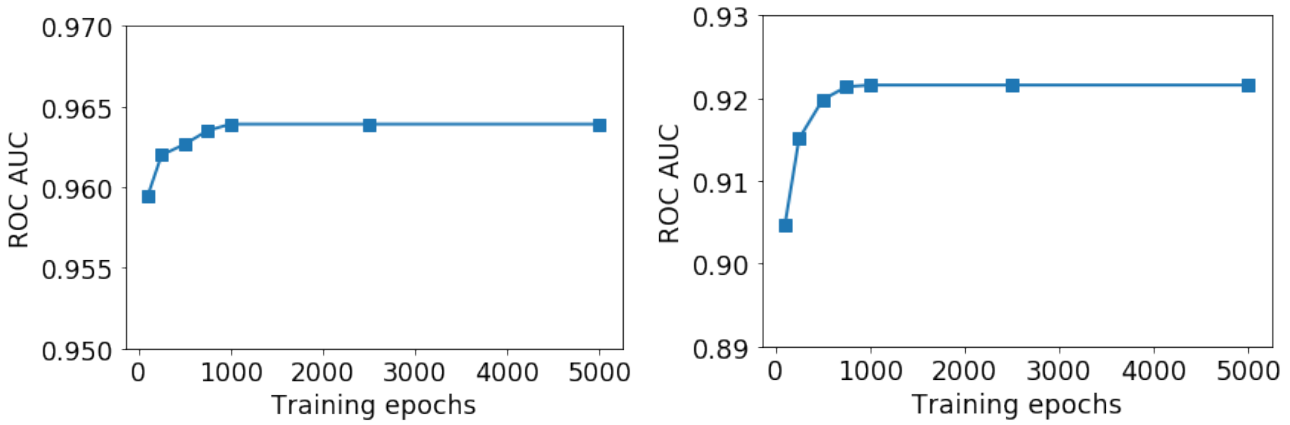


Fig. 12. Evaluation of OneFlow on MNIST (left) and Fashion-MNIST (right) with seven different numbers of training epochs: 100, 250, 500, 750, 1000, 2500, 5000.

[7] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 2017, pp. 140–145.

[8] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[9] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "Latent space autoregression for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 481–490.

[10] D. Li, D. Chen, J. Goh, and S.-K. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *arXiv preprint arXiv:1809.04758v3*, 2019.

[11] S. Wang, Y. Zeng, X. Liu, E. Zhu, J. Yin, C. Xu, and M. Kloft, "Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network," in *Advances in Neural Information Processing Systems*, 2019, pp. 5960–5973.

[12] C. D. Scott and R. D. Nowak, "Learning minimum volume sets," *Journal of Machine Learning Research*, vol. 7, no. Apr, pp. 665–704, 2006.

[13] M. Zhao and V. Saligrama, "Anomaly detection with score functions based on nearest neighbor graphs," in *Advances in neural information processing systems*, 2009, pp. 2250–2258.

[14] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[15] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 215–10 224.

[17] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4541–4550.

[18] P. Spurek, S. Winczowski, J. Tabor, M. Zamorski, M. Zieba, and T. Trzcinski, "Hypernetwork approach to generating point clouds," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9099–9108.

[19] C. Cheng, "The bernstein polynomial estimator of a smooth quantile function," *Statistics & probability letters*, vol. 24, no. 4, pp. 321–330, 1995.

[20] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[21] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, 2014.

[22] Y. Chen, J. Qian, and V. Saligrama, "A new one-class svm for anomaly detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3567–3571.

[23] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*, 2018, pp. 4393–4402.

[24] S. Kim, Y. Choi, and M. Lee, "Deep learning with support vector data description," *Neurocomputing*, vol. 165, pp. 111–117, 2015.

[25] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," *arXiv preprint arXiv:1906.02694*, 2019.

[26] P. Chong, L. Ruff, M. Kloft, and A. Binder, "Simple and effective prevention of mode collapse in deep one-class classification," *arXiv preprint arXiv:2001.08873*, 2020.

[27] S. Dasgupta, T. C. Sheehan, C. F. Stevens, and S. Navlakha, "A neural data structure for novelty detection," *Proceedings of the National Academy of Sciences*, vol. 115, no. 51, pp. 13 093–13 098, 2018.

[28] L. Shu, H. Xu, and B. Liu, "Unseen class discovery in open-world classification," *arXiv preprint arXiv:1801.05609*, 2018.

[29] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International conference on information processing in medical imaging*. Springer, 2017, pp. 146–157.

[30] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-anogan: Fast unsupervised anomaly detection with generative adversarial networks," *Medical image analysis*, vol. 54, pp. 30–44, 2019.

[31] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft, "Image anomaly detection with generative adversarial networks," in *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 2018, pp. 3–17.

[32] P. Perera, R. Nallapati, and B. Xiang, "Ocgan: One-class novelty detection using gans with constrained latent representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2898–2906.

[33] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[34] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[35] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[36] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[37] J. Wang, S. Sun, and Y. Yu, "Multivariate triangular quantile maps for novelty detection," in *Advances in Neural Information Processing Systems*, 2019, pp. 5061–5072.

[38] M. Schmidt and M. Simic, "Normalizing flows for novelty detection in industrial time series data," *arXiv preprint arXiv:1906.06904*, 2019.

[39] B. Nachman and D. Shih, "Anomaly detection with density estimation," *Physical Review D*, vol. 101, no. 7, p. 075042, 2020.

[40] L. Wellhausen, R. Ranftl, and M. Hutter, "Safe robot navigation

via multi-modal anomaly detection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1326–1333, 2020.
[41] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, 2021.
[42] G. Pang, C. Shen, L. Cao, and A. v. d. Hengel, "Deep learning for anomaly detection: A review," *arXiv preprint arXiv:2007.02500*, 2020.
[43] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.
[44] A. Leblanc, "On estimating distribution functions using bernstein polynomials," *Annals of the Institute of Statistical Mathematics*, vol. 64, no. 5, pp. 919–943, 2012.
[45] R. Zielinski, *Optimal Quantile Estimators Small Sample Approach*. Polish Academy of Sciences. Institute of Mathematics, 2004.
[46] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," *arXiv preprint arXiv:1605.07717*, 2016.
[47] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.
[48] I. Golan and R. El-Yaniv, "Deep anomaly detection using geometric transformations," in *Advances in Neural Information Processing Systems*, 2018, pp. 9758–9769.
[49] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[50] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.
[51] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," in *Advances in neural information processing systems*, 2018, pp. 6822–6833.
[52] P. Gopalan, V. Sharan, and U. Wieder, "Pidforest: Anomaly detection via partial identification," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 809–15 819.
[53] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.
[54] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *International conference on machine learning*, 2016, pp. 2712–2721.
[55] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.
[56] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
[57] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint arXiv:1810.09136*, 2018.

APPENDIX A
NOTATIONS

- g – a density in $\mathbb{R}^D$ producing data,
- $X \subset \mathbb{R}^D$ – a sample generated by density g,
- $(x_1, \ldots, x_n) \subset X$ – a mini-batch,
- $\alpha \in (0, 1)$ – a p-value, which determines the percentage of possible outliers in a dataset,
- $U \subset \mathbb{R}^D$ – a $(1-\alpha)$-bounding region (i.e. $\int_U g(x)\,dx = 1-\alpha$),
- s_α – the Bernstein estimation of $(1-\alpha)$-quantile based on a sample $s_1, \ldots, s_n$.
- $f : \mathbb{R}^D \to \mathbb{R}^D$ – a transformation given by a neural network.
- $f_\theta : \mathbb{R}^D \to \mathbb{R}^D$ – a transformation given by a neural network with weights θ.
- $\det df(x)$ – the Jacobian determinant of neural network f.
- $B(0, r)$ – a ball centered at 0 with radius r.

APPENDIX B
EXPERIMENTAL SETTING

In all our experiments, OneFlow and LL-Flow are implemented using the architecture of NICE flow model [15], with the following hyperparameters:

B.1 2D datasets

- Number of flow layers: 4
- Number of coupling layers: 4
- Hidden dimension: 16
- Number of epochs: 1000
- Batch size: 1000
- Learning rate: 0.001

B.2 PIDForest benchmark

- Number of flow layers: 2
- Number of coupling layers: 6
- Hidden dimension: 64
- Number of epochs: 2000
- Batch size: 1000
- Learning rate: 0.001

B.3 Other anomaly detection and image datasets

- Number of flow layers: 4
- Number of coupling layers: 4
- Hidden dimension: 256
- Number of epochs: 1000
- Batch size: 1000
- Learning rate: 0.001

APPENDIX C
DESCRIPTION OF COMPARATIVE ALGORITHMS

Below, we give a brief description of algorithms used in the experimental section:

- **One-class SVM (OC-SVM)** [8]. It is a traditional kernel-based one-class classifier (we use the RBF kernel).
- **Deep structured energy-based models (DSEBM)** [46]. This model employs a deterministic deep neural network to output the energy function, such as negative log-likelihood, which is used to form the density of nominal data.
- **Deep autoencoding Gaussian mixture model (DAGMM)** [47]. It combines deep autoencoder with a Gaussian mixture estimation network to output the joint density of the latent representations and some reconstruction features from the autoencoder.
- **Four variants of MQT – multivariate quantile map (NLL, TQM$_1$, TQM$_2$, TQM$_\infty$)** [37]. MQT is a general model, which thresholds a given score function to describe nominal data. As a score function, we use negative log-likelihood (NLL) as well as 1-norm, 2-norm, and infinity norm of quantile (TQM).
- **Deep Support Vector Data Description (DSVDD)** [23]. It is an implementation of SVDD using deep neural networks, which penalizes data points that lie outside the hypersphere.

- **Two variants of log-likelihood flow model (LL-Flow and LL-Flow-Gen)**. These are generative flow models based on log-likelihood function that mimic OneFlow and OneFlow-Gen, respectively.
- **Geometric transformation (GT)** [48]. It uses a multi-class model to discriminate between dozens of geometric transformations applied to examples from the nominal class. The scoring function is the conditional probability of the softmax responses of the classifier given the geometric transformations.
- **Variational autoencoder (VAE)** [49]. The evidence lower bound is used as the scoring function.
- **Denoising autoencoder (DAE)** [50]. The reconstruction error is used as the scoring function.
- **Generative probabilistic novelty detection (GPND)** [51]. GPND, based on adversarial autoencoders, uses data density as the scoring function. Density is approximated by linearizing the manifold that nominal data resides on.
- **Latent space autoregression (LSA)** [9]. A parametric autoregressive model is used to estimate the density of the latent representation generated by a deep autoencoder. The sum of the normalized reconstruction error and log-likelihood is used as the scoring function.
- **PIDForest** [52]. A random forest based algorithm that finds outliers based on the value of PIDScore, which is a geometric anomaly measure for a point. The scoring function measures the minimum density of data points over all subcubes containing the point.
- **IsolationForest (iForest)** [53]. A random forest based algorithm. iForest isolates observations by randomly selecting a feature and a split value. The number of splittings required to isolate a sample is the scoring function.
- **Robust Random Cut Forest (RRCF)** [54]. An outlier detection algorithm that is based on a binary search tree. The scoring function is measured by its collusive displacement (CoDisp): if including a new point significantly changes the model complexity (i.e. bit depth), then that point is more likely to be an outlier.
- **Local Outlier Factor (LOF)** [55]. The scoring function is based on measuring the local deviation of a given data point with respect to its k-nearest neighbours.
- **k-Nearest Neighbour (kNN)**. The distance from k-nearest neighbours is considered as the scoring function.
- **Principal Component Analysis (PCA)** [56]. The scoring function is calculated as the distance from the axes in feature space.

APPENDIX D
DETAILED RESULTS FOR EXPERIMENTS SECTION

In Tables 4 and 5, we present detailed results obtained for PIDForest benchmark.

In Tables 6 and 7, we present detailed results obtained for MNIST and Fashion-MNIST datasets.

To illustrate previous results, in Figures 13, 14, 15, 16 we show:

- examples from nominal class, which lie closest to the center of bounding hypersphere (1st column),
- examples from nominal class, which are farthest from the center of bounding hypersphere (2nd column),
- anomalies, which lie closest to the center of bounding hypersphere (3rd column),
- anomalies, which are farthest from the center of bounding hypersphere (4th column),

Interestingly, the examples from the class "1" are frequently localized close to the center of bounding hypersphere. It partially explains the behavior observed in previous heatmaps. Another observation is that the examples closest to the center are very regular (first column), while examples farthest from the center look worse, and humans can make mistakes in classifying these images.

TABLE 4
AUC obtained on PIDForest benchmarks. The results of comparative methods (except LL-Flow and LL-Flow-Gen) are taken from [52]. Experiments were repeated 3 times, we present the mean as well as standard deviation of obtained scores.

Data set	PIDForest	iForest	RRCF	LOF	SVM	kNN	PCA	LL-Flow	LL-Flow-Gen	OneFlow	OneFlow-Gen
Thyroid	.876 ± .013	.819 ± .013	.739 ± .004	.737	.547	.751	.673	.848 ± .012	.856 ± .012	.605 ± .147	.691 ± .078
Mammo.	.840 ± .010	.862 ± .008	.830 ± .002	.720	.872	.839	.886	.856 ± .001	.871 ± .007	.710 ± .039	.715 ± .057
Seismic	.733 ± .006	.698 ± .004	.701 ± .004	.553	.601	.740	.682	.705 ± .007	.708 ± .012	.709 ± .004	.719 ± .011
Satimage	.987 ± .001	.994 ± .001	.991 ± .002	.540	.421	.936	.977	.911 ± .007	.982 ± .010	.997 ± .000	.998 ± .001
Vowels	.741 ± .008	.736 ± .026	.813 ± .007	.943	.778	.975	.606	.837 ± .019	.670 ± .012	.846 ± .036	.839 ± .056
Musk	1.00 ± .000	.998 ± .003	.998 ± .000	.416	.573	.373	1.00	.968 ± .005	.989 ± .006	.688 ± .011	.946 ± .092
http	.986 ± .004	1.00 ± .000	.993 ± .000	.353	.994	.231	.996	.992 ± .001	.991 ± .005	.994 ± .000	.994 ± .000
smtp	.923 ± .003	.908 ± .003	.886 ± .017	.905	.841	.895	.823	.878 ± .016	.900 ± .005	.857 ± .016	.841 ± .011

TABLE 5
F1 obtained on PIDForest benchmarks. Experiments were repeated 3 times, we present the mean as well as standard deviation of obtained scores.

Data set	PIDForest	iForest	RRCF	LOF	SVM	kNN	PCA	LL-Flow	LL-Flow-Gen	OneFlow	OneFlow-Gen
Thyroid	.265 ± .054	.311 ± .018	.263 ± .008	.242	.103	.255	.226	.299 ± .010	.328 ± .013	.269 ± .024	.312 ± .043
Mammo.	.245 ± .022	.231 ± .025	.226 ± .010	.176	.229	.212	.254	.190 ± .017	.243 ± .005	.188 ± .010	.211 ± .020
Seismic	.159 ± .014	.139 ± .013	.100 ± .004	.053	.000	.140	.153	.140 ± .014	.126 ± .018	.164 ± .003	.129 ± .024
Satimage	.371 ± .002	.377 ± .004	.377 ± .006	.099	.011	.210	.354	.145 ± .002	.344 ± .052	.381 ± .000	.387 ± .004
Vowels	.205 ± .025	.198 ± .039	.189 ± .017	.407	.228	.585	.146	.284 ± .022	.135 ± .020	.298 ± .062	.309 ± .082
Musk	.774 ± .001	.773 ± .000	.759 ± .003	.024	.000	.040	.773	.531 ± .030	.661 ± .081	.278 ± .045	.632 ± .122
http	.145 ± .000	.145 ± .000	.145 ± .000	.000	.145	.006	.014	.145 ± .000	.145 ± .000	.145 ± .000	.145 ± .000
smtp	.009 ± .000	.009 ± .000	.009 ± .000	.009	.009	.009	.009	.009 ± .000	.009 ± .000	.009 ± .000	.009 ± .012

TABLE 6
AUC obtained on MNIST dataset. The results of comparative methods (except DSVDD, LL-Flow and LL-Flow-Gen) are taken from [37].

Class	OCSVM	VAE	DAE	LSA	GT	DAGM	GPND	DSEBM	DSVDD	NLL	TQM	LL-Flow	LL-Flow-Gen	OneFlow	OneFlow-Gen	OF-RNVP	OF-Glow
0	.995	.985	.982	.998	.982	.500	.999	.320	.980	.995	.993	.990	.990	.994	.995	.600	.934
1	.999	.997	.998	.999	.893	.766	.999	.987	.997	.998	.997	.998	.998	.999	.999	.997	.997
2	.926	.943	.936	.923	.993	.326	.980	.482	.917	.953	.948	.930	.927	.945	.944	.523	.913
3	.936	.916	.929	.974	.987	.319	.968	.753	.919	.963	.957	.952	.936	.967	.972	.683	.896
4	.967	.945	.940	.955	.993	.368	.980	.696	.949	.966	.963	.937	.941	.954	.949	.712	.909
5	.955	.929	.928	.966	.994	.490	.987	.727	.885	.962	.960	.961	.949	.973	.974	.652	.828
6	.987	.977	.982	.992	.999	.515	.998	.954	.983	.992	.990	.984	.985	.990	.991	.591	.911
7	.966	.975	.971	.969	.966	.500	.988	.911	.946	.969	.966	.970	.971	.976	.975	.824	.948
8	.903	.864	.857	.935	.974	.467	.929	.536	.939	.955	.951	.841	.851	.870	.874	.545	.729
9	.962	.967	.974	.969	.993	.813	.993	.905	.965	.977	.976	.965	.970	.972	.972	.701	.883
avg	.960	.950	.950	.968	.977	.508	.982	.727	.948	.973	.970	.953	.952	.964	.964	.683	.895

TABLE 7
AUC obtained on Fashion-MNIST dataset. The results of comparative methods (except DSVDD, LL-Flow and LL-Flow-Gen) are taken from [37].

Class	OCSVM	VAE	DAE	LSA	GT	DAGM	GPND	DSEBM	DSVDD	NLL	TQM	LL-Flow	LL-Flow-Gen	OneFlow	OneFlow-Gen	OF-RNVP	OF-Glow
0	.919	.874	.867	.916	.903	.303	.917	.891	.791	.922	.917	.914	.901	.917	.918	.766	.741
1	.990	.977	.978	.983	.993	.311	.983	.560	.940	.958	.950	.989	.989	.989	.989	.917	.962
2	.894	.816	.808	.878	.927	.475	.878	.861	.830	.899	.899	.614	.876	.893	.894	.793	.721
3	.942	.912	.914	.923	.906	.481	.945	.903	.829	.930	.925	.930	.934	.930	.932	.792	.748
4	.907	.872	.865	.897	.907	.499	.906	.884	.870	.922	.921	.841	.582	.903	.903	.724	.686
5	.918	.916	.921	.907	.954	.413	.924	.859	.803	.894	.884	.904	.909	.902	.901	.664	.766
6	.834	.738	.738	.841	.832	.420	.785	.782	.749	.844	.838	.828	.804	.820	.820	.683	.704
7	.988	.976	.977	.977	.981	.374	.984	.981	.942	.980	.972	.989	.989	.989	.989	.950	.952
8	.903	.795	.782	.910	.976	.518	.916	.865	.791	.945	.943	.888	.873	.893	.890	.521	.653
9	.982	.965	.963	.984	.994	.378	.876	.967	.932	.983	.983	.968	.975	.979	.980	.881	.817
avg	.928	.884	.881	.922	.937	.472	.911	.855	.847	.928	.923	.887	.883	.922	.922	.769	.775

Fig. 13. Samples for MNIST dataset for OneFlow model.

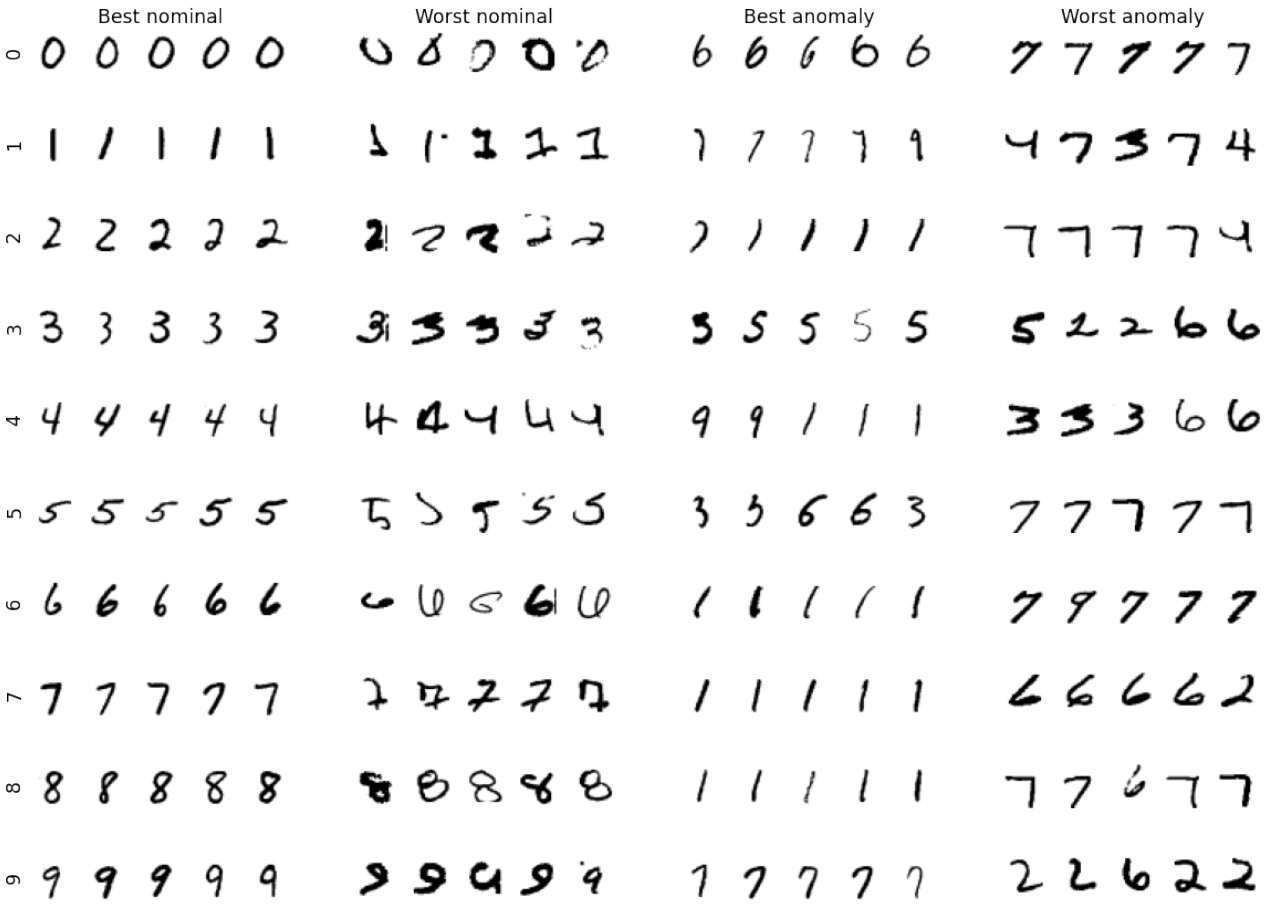


Fig. 14. Samples for MNIST dataset for OneFlow-Gen model.

Fig. 15. Samples for Fashion-MNIST dataset for OneFlow model.

Fig. 16. Samples for Fashion-MNIST dataset for OneFlow-Gen model.

Flow-based SVDD for anomaly detection

Marcin Sendera [1] **Marek Śmieja** [1] **Łukasz Maziarka** [1] **Łukasz Struski** [1] **Przemysław Spurek** [1] **Jacek Tabor** [1]

Abstract

We propose FlowSVDD – a flow-based one-class classifier for anomaly/outliers detection that realizes a well-known SVDD principle using deep learning tools. Contrary to other approaches to deep SVDD, the proposed model is instantiated using flow-based models, which naturally prevents from collapsing of bounding hypersphere into a single point. Experiments show that FlowSVDD achieves comparable results to the current state-of-the-art methods and significantly outperforms related deep SVDD methods on benchmark datasets.

1. Introduction

Anomaly (novelty/outlier) detection refers to the identification of novel or abnormal patterns embedded in a large amount of typical (normal) data (Miljković, 2010). Anomaly detection algorithms find application in fraud detection systems, discovering failures in the industrial domain, detection of adversarial examples, etc..

In contrast to typical binary classification problems, where every class follows some probability distribution, an anomaly is a pattern that does not conform to the expected behavior. In consequence, a completely novel type of anomalies can occur at test time, which is not similar to any known anomalies. Moreover, in most cases, we do not have access to any anomalies at training time. In consequence, novelty detection is usually solved using unsupervised approaches, such as one-class classifiers, which focus on describing the behavior of available data (inliers). Any observation, which deviates from this behavior, is labeled as an outlier.

Our research is motivated by the idea of Support Vector Data Description (SVDD) (Tax & Duin, 2004), which obtains a spherically shaped boundary around a dataset by usage of soft margin and penalization of data points from outside the bounding region. We propose FlowSVDD – a one-class classifier based on flow-based models (Dinh et al., 2014), which finds a hypersphere with a minimal volume that encloses data. Since flow-based models are commonly used in the context of generative models, we redefine their cost function to minimize the volume of the bounding hypersphere instead of maximizing the log-likelihood function. On one hand, flow-based models allow us to calculate a Jacobian of a neural network at every point. In consequence, minimizing the volume of the hypersphere in the feature space leads to the minimization of the volume of the corresponding bounding region in the input space. On the other hand, since flow-based models give an explicit formula for the inverse mapping, we automatically get a parametric form for the corresponding bounding region in the input space. In contrast to deep SVDD models, our approach eliminates the problem of hypersphere collapse, which makes it easy to use.

Extensive experiments performed on typical benchmark datasets show that our method significantly outperforms the deep SVDD model while being comparative to state-of-the-art models for anomaly detection.

Our contribution is summarized as follows:

1. We propose an adaptation of the SVDD method to deep neural networks with the use of flow models.
2. We show that the realization of the SVDD loss function on flow-based models prevents from hypersphere collapse.
3. We experimentally compare FlowSVDD with Deep SVDD and current state-of-the-art methods.

2. Proposed model

Preliminaries: SVDD. Our approach is motivated by a classical Support Vector Data Description (SVDD) (Tax & Duin, 2004), which tries to find a minimal hypersphere to enclose the data. To allow the possibility of outliers in the training set, SVDD uses a soft margin and penalizes data points that lie outside the bounding hypersphere. If f maps input data to the output kernel space, then SVDD loss

*Equal contribution [1]Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland. Correspondence to: Marcin Sendera <marcin.sendera@gmail.com>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

equals:

$$F(R, c; f) = R^2 + \frac{1}{\nu n} \sum_i \max(0, \|f(x_i) - c\|^2 - R^2) \quad (1)$$

where $c \in \mathbb{R}^D, R \in \mathbb{R}$ is the center and the radius of the hypersphere, respectively, and ν is the trade-off between the volume and boundary violations of the hypersphere, i.e. fraction of outliers.

The realization of SVDD using deep neural networks was presented in (Ruff et al., 2018) (it was termed DSVDD). However, direct minimization of the SVDD loss may lead to a trivial solution, i.e. the hypersphere collapses to a single point c. To avoid this negative behavior, it has been recommended that the center c must be something other than the all-zero-weights solution, and the network should use only unbounded activations and omit bias terms. While the two first conditions can be accepted, omitting bias terms in a network may lead to a sub-optimal feature representation due to the role of bias in shifting activation values.

To eliminate the above restrictions a recent work (Chong et al., 2020) proposes two regularizers, which prevent hypersphere collapse, and uses an adaptive weighting scheme to control the amount of penalization between the SVDD loss and the respective regularizer.

Flow-based SVDD. As an alternative to DSVDD, we realize the SVDD objective using flow models. Let us recall that a neural network $f : \mathbb{R}^D \to \mathbb{R}^D$ is a flow model if the inverse mapping f^{-1} is given explicitly and the Jacobian determinant $w(x) = \det df(x)$ can be easily calculated. In our approach, we use a special class of flow models, in which Jacobian determinant is constant at every point, i.e. $w(x) = w$, such as NICE (Dinh et al., 2014). In this case, we get a natural correspondence between the volume of the bounding hypersphere in the output space and the volume of a bounding region in the input space, see below.

Let us first consider the simplest situation when $w = 1$. In such a scenario, the volume of any shape in the input space equals the volume of its image in the output feature space. In consequence, a direct minimization of the SVDD objective does not lead to the hypersphere collapse.

In a more general scenario, when $w \neq 1$, we need to include w in the SVDD objective. Observe that the Jacobian determinant of the mapping $f/w^{1/D}$ equals 1. Thus to get the equality of the volume in the input and output space, we redefine the SVDD loss (1) as follows:

$$F(R, c; f) = R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max(0, \|f(x_i)/w^{1/D} - c\|^2 - R^2). \quad (2)$$

In a test phase, a given example x is deemed as an outlier if:

$$\|f(x)/w^{1/D} - c\| > R,$$

which is equivalent to

$$\|f(x) - w^{1/D} c\| > R w^{1/D}.$$

In other words, inliers lie inside the ball $B(w^{1/D} c; R w^{1/D})$.

3. Experiments

In this section, we experimentally examine FlowSVDD and compare it with several state-of-the-art approaches. FlowSVDD is implemented using the architecture of the NICE flow model (4 coupling layers – each consisted of 4 layers and 256 hidden dimensions) with constant Jacobian determinant and $\nu = 0.05$.

Illustrative example. To get the intuition behind FlowSVDD, we first consider 2-dimensional examples, which are easy to visualize. The results presented in Figure 1 show the resulting hyperspheres in the latent space and the corresponding bounding regions in the input space. At first glance, we can observe that the bounding region in the original space is close to the structure of inliers. In the latent space, FlowSVDD finds the center point c and radius R to enclose $(1 - \nu)$ percentage of data inside the ball $B(c; R)$. Observe that, unlike the density-based flow models, FlowSVDD does not transform data into Gaussian distribution in a latent space.

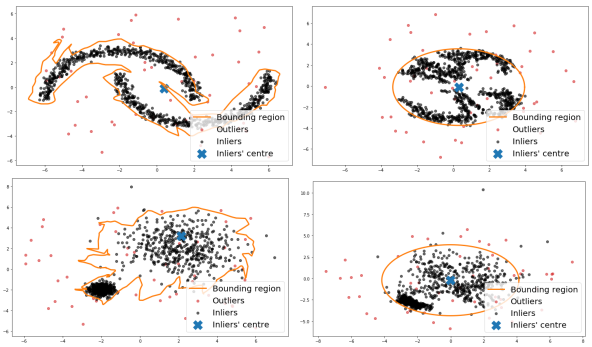


Figure 1. Enclosing hyperspheres in the latent space of FlowSVDD (right) and the corresponding bounding regions in the input space (left).

Benchmark data for anomaly detection. To provide quantitative assessment, we take into account Thyroid[1] and KDDCUP[2] datasets, which are typically used for anomaly

[1] http://odds.cs.stonybrook.edu/thyroid-disease-dataset/

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup.testdata.unlabeled_10_percent.gz

detection. We use the standard training and test splits and follow exactly the same evaluation protocol as in (Wang et al., 2019). In particular, we use the F1 score and the Area Under Receiver Operating Characteristic curve (AUC).

Our model is compared with the following algorithms: (1) One-class SVM (OC-SVM) (Schölkopf et al., 2001), (2) Deep structured energy-based models (DSEBM) (Zhai et al., 2016), (3) Deep autoencoding Gaussian mixture model (DAGMM) (Zong et al., 2018), (4) variants of MQT – multivariate quantile map (NLL, TQM_1, TQM_2, TQM_∞) (Wang et al., 2019) and (5) Deep Support Vector Data Description (DSVDD) (Ruff et al., 2018) - another implementation of SVDD cost function in deep neural networks.

The results presented in Table 1 show that FlowSVDD model performs better than most methods on the Thyroid dataset and is significantly better than DSVDD in terms of the AUC metric. In the case of KDDCUP, FlowSVDD achieves a score in between the classical methods and current state-of-the-art.

Image datasets. To provide further experimental verification, we use two image datasets: MNIST and Fashion-MNIST. In contrast to the previous comparison, these two datasets are usually used for multiclass classification and thus need to be adapted to the problem of anomaly detection. For this purpose, each of the ten classes is deemed as the nominal class while the rest of the nine classes are deemed as the anomaly class, which results in 10 scenarios for each dataset.

We additionally compare FlowSVDD with the following models: (1) Geometric transformation (GT) (Golan & El-Yaniv, 2018), Variational autoencoder (VAE) (Kingma & Welling, 2013), Denoising autoencoder (DAE) (Vincent et al., 2008), Generative probabilistic novelty detection (GPND) (Pidhorskyi et al., 2018), Latent space autoregression (LSA) (Abati et al., 2019). In contrast to previous experiment, we only use TQM_2 and NLL as the only implementations of MTQ, because they output the highest value of AUC (Wang et al., 2019).

To present the results, we compute the ranking on each of 10 scenarios and summarize it using a box plot, see Figure 2. The results show that FlowSVDD significantly outperforms DSVDD in both datasets. In the case of the MNIST dataset, we observe that FlowSVDD is almost as good as the current state-of-the-art methods, like GT and NLL.

Finally, we analyze, which samples are localized close to or furthest from the center of bounding hypersphere. Results in Figure 3 shows that FlowSVDD maps regular images in the hypersphere center. Contrary, examples localized far from the center, could be hard to identify. It means that FlowSVDD gives results consistent with our intuition.

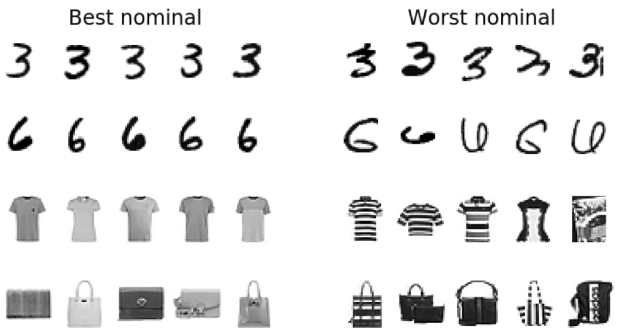


Figure 3. Best nominal (left) and worst nominal (right) examples determined by FlowSVDD for MNIST (top) and Fashion-MNIST (bottom).

4. Conclusion

The paper introduced FlowSVDD, which realizes the SVDD paradigm in the case of neural networks. Making use of flow-based models and an appropriate SVDD-like cost function, we find a minimal bounding region for a majority of data. Unlike other deep SVDD realizations, FlowSVDD does not change the determinant of a Jacobian matrix, which means that the resulting hypersphere cannot collapse in a latent space. The experimental results demonstrate that FlowSVDD presents a very good performance in the case of both artificial and real-world one-class settings.

Acknowledgements

The work of M. Śmieja was supported by the National Centre of Science (Poland) Grant No. 2018/31/B/ST6/00993. The work of P. Spurek was supported by the National Centre of Science (Poland) Grant No. 2019/33/B/ST6/00894. The work of J. Tabor was supported by the National Centre of Science (Poland) Grant No. 2017/25/B/ST6/01271.

References

Abati, D., Porrello, A., Calderara, S., and Cucchiara, R. Latent space autoregression for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 481–490, 2019.

Chong, P., Ruff, L., Kloft, M., and Binder, A. Simple and effective prevention of mode collapse in deep one-class classification. *arXiv preprint arXiv:2001.08873*, 2020.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Golan, I. and El-Yaniv, R. Deep anomaly detection us-

Table 1. Performance on two anomaly detection datasets.

Thyroid									
	OC-SVM	DSEBM	DAGMM	NLL	TQM_1	TQM_2	TQM_∞	DSVDD	FlowSVDD
F1	.3887	.0403	.4782	.7312	.5269	.5806	.7527	-	.7097
AUC	-	-	-	-	-	-	-	0.749	.9797
KDDCUP									
	OC-SVM	DSEBM	DAGMM	NLL	TQM_1	TQM_2	TQM_∞	DSVDD	FlowSVDD
F1	.7954	.7423	.9369	.9622	.9621	.9622	.9622	-	.9030
AUC	-	-	-	-	-	-	-	-	.9384

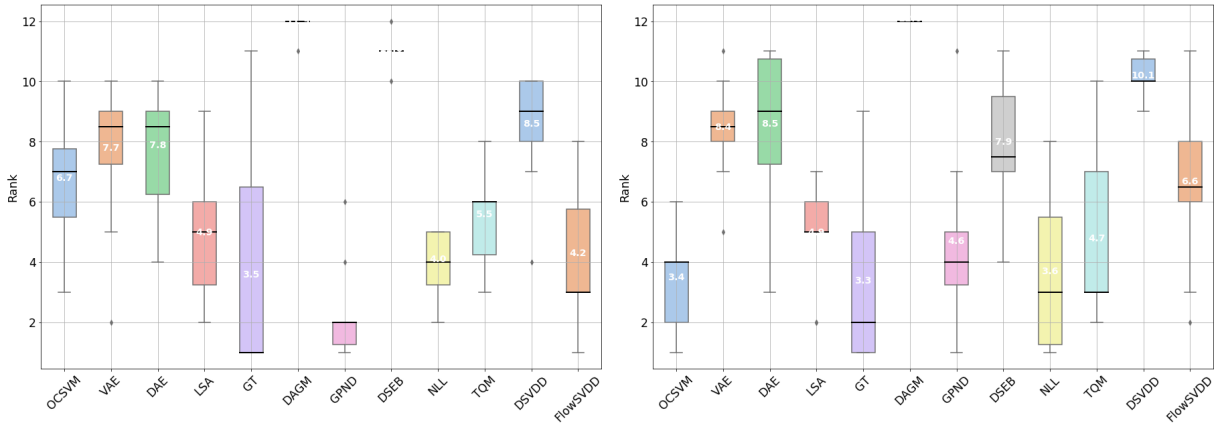


Figure 2. Box plots for rankings calculated on MNIST (left) and Fashion-MNIST (right). The median ranking is marked by a line, while the average ranking is marked with a number.

ing geometric transformations. In *Advances in Neural Information Processing Systems*, pp. 9758–9769, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Miljković, D. Review of novelty detection methods. In *The 33rd International Convention MIPRO*, pp. 593–598. IEEE, 2010.

Pidhorskyi, S., Almohsen, R., and Doretto, G. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in neural information processing systems*, pp. 6822–6833, 2018.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *International conference on machine learning*, pp. 4393–4402, 2018.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7): 1443–1471, 2001.

Tax, D. M. and Duin, R. P. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Wang, J., Sun, S., and Yu, Y. Multivariate triangular quantile maps for novelty detection. In *Advances in Neural Information Processing Systems*, pp. 5061–5072, 2019.

Zhai, S., Cheng, Y., Lu, W., and Zhang, Z. Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*, 2016.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. 2018.

Missing Glow Phenomenon: learning disentangled representation of missing data

Marcin Sendera[1][0000−0002−8741−6919], Łukasz Struski[1][0000−0003−4006−356X], and Przemysław Spurek[1][0000−0003−0097−5521]

Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
marcin.sendera@ii.uj.edu.pl
lukasz.struski@uj.edu.pl
Przemyslaw.Spurek@ii.uj.edu.pl

Abstract. Learning from incomplete data has been recognized as one of the fundamental challenges in deep learning. There are many more or less complicated methods for processing missing data by neural networks in the literature. In this paper, we show that flow-based generative models can work directly on images with missing data to produce full images without missing parts. We name this behavior Missing Glow Phenomenon. We present experiments that document such behaviors and propose theoretical justification of such phenomena.

Keywords: Generative models · Normalizing Flows · Glow · Disentanglement · Missing data.

1 Introduction

Missing data is a widespread problem in real-life machine learning challenges [5]. A typical strategy for working with incomplete inputs relies on filling absent attributes based on observable ones [14], which is the mean imputation. Another possible solution is to replace the typical neuron's response in the first hidden layer by its expected value [20]. All of the existing methods use a more or less complicated mechanism to solve the above problems. In generative models [22,23], we have the same problem. To produce full images without missing parts, we have to use an analogical solution like the classification task.

This paper shows that flow-based generative models like Glow [8] can work directly on images with missing data to produce full images without missing parts.
We name this behavior Missing Glow Phenomenon.

The above phenomenon is obtained by producing a disentangled representation in latent space. During the training procedure, the model simultaneously creates representations of objects with missing parts and full images. The above factors are independent, and therefore we can sample from two separated areas of latent space to produce full images and items with missing parts. The

behavior is presented in Fig. 1. We use the original Glow model with sampling temperature $t = 0.7$, for which we should expect to appear about 60% of full images as shown in Tab. 1.

Fig. 1: Missing Glow Phenomenon. Glow was trained on a missing dataset and then sampled with temperature $t = 0.7$ (standard temperature for Glow), which gives about 60% of the full images.

The proposed approach's main advantage is the ability to train a neural network on datasets containing incomplete samples to produce a generative model on full images (the model does not see any full images during training). This approach distinguishes it from recent models like denoising autoencoder [22] or modified generative adversarial network [23], which require complete data as an input of the network in training.

In the paper, we show experiments that describe Missing Glow Phenomenon (see Section 3) and present the theoretical explanation of such behaviors (see Section 4).

2 Related works

There is a rich literature on handling missing values [11]. Theoretical guarantees of estimation strategies or imputation methods rely on assumptions regarding the missing-data mechanism, i.e., the cause of the lack of data. There are three mechanisms for the formation of missing data. Missing Completely At Random (MCAR) if the probability of being missing is the same for all observations. Missing At Random (MAR) if the probability of being missing only depends on observed values, and the last Missing Not At Random (MNAR) if the unavailability of the data depends on both observed and unobserved data such as its value itself.

A classical approach to estimating parameters with missing values consists of maximizing the observed likelihood, using, for instance, an Expectation - Maximization algorithm [3]. One of its main drawbacks is to rely on strong parametric assumptions for the distributions of the covariates. Another popular strategy to fix the missing values issue is predicting the missing values based on observable ones, e.g., mean or k-NN imputation. One can also train separate models, e.g., neural networks [18], extreme learning machines (ELM) [21], k-nearest neighbors [1].

In [17], the appropriateness of ignoring the missing process when approach likelihood-based or Bayesian inference was introduced and formalized. Under certain assumptions on the missing mechanism, we can build a probabilistic model of incomplete data, which is subsequently fed into a particular learning model [19].

A body of research on deep frameworks that can learn from partially observed data has emerged in recent years. Initial work focused on extensions of

generative models such as Variational Auto Encoders (VAEs) [9] and Generative Adversarial Networks (GANs) [6]. In [24] used generative adversarial net (GAN) to fill in absent attributes with realistic values. In [7] was proposed the supervised imputation, which learns a replacement value for each missing attribute jointly with the remaining network parameters. This group also includes models such as the Missing Data Importance-Weighted Autoencoder (MIWAE) [13] and the Generative Adversarial Imputation Network (GAIN) [25]. More recently, the state-of-the-art benchmark on learning from incomplete data has been pushed by bidirectional generative frameworks, which leverage the ability to map back and forth between the data space and the latent space. Two such examples include the Monte-Carlo Flow model (MCFlow) [16] and the Partial Bidirectional GAN (PBiGAN) [10].

Fig. 2: Reconstructed images for 20×20 missing dataset. Parameter α is a scaling factor of original images' representations in a latent space. For $\alpha = 1.0$, we have an original missing image.

3 Experiments

3.1 Missing dataset

For the purpose of the experiments, we construct a special dataset with missings, based on a CelebA - a large-scale dataset with more than 200K celebrities' images annotated with about 40 attributes [12]. We resize each of the images into 64×64 pixels size and impute a grey square box, which is a missing part of an image, as shown in Fig. 1 (left image). The missing squares are placed at the beginning of the training, and their locations come from the uniform distribution. Thus, the model always sees each of the original images with the square in the same position. We used different sizes of missing squares in the following experiments - from 2×2 up to 20×20 pixels. Note that the small missing square (e.g. 2×2 or 4×4) could be seen as an attribute by a model. Whereas images with the missing squares larger than 20×20 are, in fact, too corrupted to recreate.

3.2 Glow model architecture

Normalizing Flows Across many areas of research in the field of deep learning, generative models are perhaps one of the most popular. However, besides the well-known Generative Adversarial Networks (GAN) [6] and Variational Autoencoders (VAE) [9], there is another group of generative models becoming popular - Normalizing Flows (NF) [15]. The main idea standing for Normalizing Flows

is to transform a simple prior distribution $P(Z)$ (e.g., Gaussian one) defined on the latent space Z into a complex one in the original space, represented by data distribution X. It could be done by stacking together the series of n invertible mappings: $f_1, \ldots, f_n : Z \to X$. The probability distribution of the final target variable is obtained by flowing through the following chain of transformations:

$$x = F(z) = f_n \circ f_{n-1} \circ \cdots \circ f_1(z).$$

Moreover, for Normalizing Flows models, the log-probability density of the output variable is given by the change of variables formula:

$$\log P(x) = \log P(z) - \sum_{k=1}^{n} \log \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|,$$

where the usage of inverse flow with its form:

$$z = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_n^{-1}(x),$$

allows for computing z from x.

In the general framework of Normalizing Flows, the main challenge is to design intermediate layers f_i in such a way to assure that both inverse map and the Jacobian are easily computable. In the case of *continuous* data x, the Continuous Normalizing Flows [2] are the extensions of the described *discrete* approach. In this setting, we allow the mappings to be defined by a solution to a differential equation $\frac{\partial z(t)}{\partial t} = f(z(t), t)$, and f could be a neural network of unrestricted architecture. The most fundamental advantage of the discrete and continuous Normalizing Flows is that we are not restricted to any specific class of functions, e.g., Gaussian densities. Moreover, unlike GANs and VAEs, the Normalizing Flows models learn the original data distribution explicitly and are able to describe densities of high dimensional images [8].

Glow model In order to process the missing dataset properly, we use a powerful flow-based model - Glow [8], where invertible 1×1 convolutions assure invertibility. Glow is known for its competitive results among other Normalizing Flow models, e.g., RealNVP [4], on describing the densities of high dimensional image data.

The Glow model consists of a series of flow steps. However, each of its flow steps could be divided into the following three substeps:

1. **Activation normalization (actnorm)**, for performing an affine transformation of the activations using a scale and bias parameter per channel;
2. **Invertible** 1×1 **convolution**, which is a generalization of a permutation operation;
3. **Affine coupling layer**, designed in the same way as in RealNVP [4].

For all of our experiments, we use the same standard Glow architecture with the following hyperparameters: 4 blocks, 32 flows in each block, 5 bits, and 3 input channels.

3.3 Imputing missings

We learned a Glow model on a specific missing dataset for each of the missing squares' sizes. In our setting, we use batch size 16 and 200k iterations. After training, we process the data follow the procedure: (1) take image x from missing dataset; (2) create its latent representation z with Glow by the embedding: $z = \Phi(x)$, where $\Phi : R^N \to R^N$; (3) move along the vector z in a latent space by multiplying it by the parameter α. For $\alpha \in (0.0, 1.0]$ create a vector $z' = \alpha z$; (4) reconstruct imputed image x' by the operation $x' = \Phi^{-1}(z')$, which is the same as $x' = \Phi^{-1}(\alpha z)$ for $\alpha \in (0.0, 1.0]$.

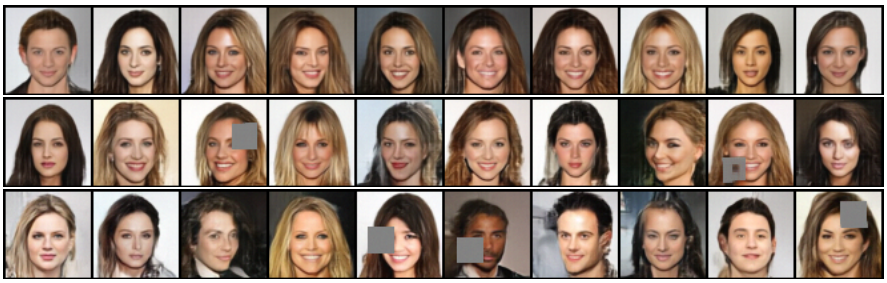


Fig. 3: Images sampled from Glow with the temperatures $t = 0.5$, $t = 0.6$, and $t = 0.7$ (from top to bottom) - in these regions of the latent space, Glow put good faces with a small number of missing squares.

We present the partial results for the missing dataset with 20×20 missing boxes in Fig. 2. Glow model was trained for 200k iterations, and we reconstruct the images by multiplying latent representations by parameter $\alpha \in \{0.1, 0.2, \ldots, 1.0\}$. However, for parameter α in range $(0.5, 0.7)$, the reconstruction is actually an original image with an imputed missing part.

3.4 Sampling from a latent representation

All the previously introduced experiments were initial parts for sampling images from Glow's latent space with a specific temperature t.

We observe various behaviors in different parts of latent space. As shown in Fig. 3, the images sampled with a temperature $t \in (0.5, 0.7)$ are characterized by faces similar to the original ones with a relatively small number of missing squares. On the other hand, Glow samples fuzzy images, full of missing regions, when taking the temperatures $t \geq 1.0$ (Fig. 4).

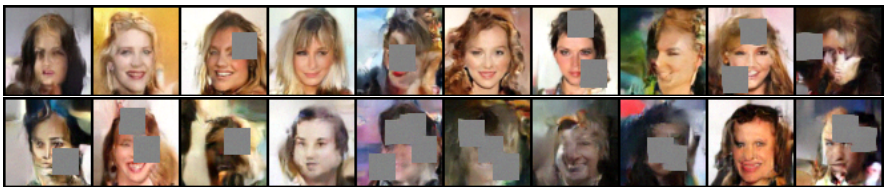


Fig. 4: Exemplary images sampled with the temperatures $t = 1.0$ (top image) and $t = 1.2$ (bottom image). We observe that images in this region are fuzzy and with a large number of missing boxes.

4 Theoretical studies

In this section, we are going to explain the Missing Glow Phenomenon showed in the experiments. Firstly, we introduce the theoretical justification, and then, we provide its ablation study.

4.1 Theoretical explanation

Let us recall that to sample from temperature t, we sample z in the latent and return $\Phi^{-1}(tz)$.

For intuition, consider the case when the data comes from the normal distribution $N(0, \Sigma)$, and then the flow can be chosen to be a linear function. Then by sampling from temperature t, we sample, in fact, from the normal density $N(0, t^2\Sigma)$. Consequently, for t going to zero, we sample from the data with covariance converging to zero, while increasing t leads to the covariance increase. Moreover, if $X \sim N(0, \Sigma)$, then to sample from the temperature $t = \sqrt{l}$, we can sample $X_1, \ldots, X_l$ independently and return $X_1 + \ldots + X_l$. The last follows from the fact that the covariance of the sum of independent random variables is the sum of their covariances.

Temperature t	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
	frequency (%)														
0 squares	100	100	100	98	92	70	64	56	50	42	30	30	24	22	22
1 square	0	0	0	2	8	26	30	34	38	44	54	44	46	46	44
2 squares	0	0	0	0	0	4	6	10	12	10	10	18	22	24	24
3 squares	0	0	0	0	0	0	0	0	0	4	4	4	4	4	6
4 squares	0	0	0	0	0	0	0	0	0	0	2	4	4	4	4
	number of missing squares in the reconstruction image														
Empirical E	0.00	0.00	0.00	0.02	0.08	0.34	0.42	0.54	0.62	0.76	0.94	1.08	1.18	1.22	1.26
Theoretical E	0.01	0.04	0.09	0.16	0.25	0.36	0.49	0.64	0.81	1.00	1.21	1.44	1.69	1.96	2.25

Table 1: Distribution of the number of missing squares when sampling with a specific temperature. We provide both empirical and theoretical expected values E.

So let us now continue our discussion in the case when the data is an independent sum of two Gaussian variables. Assume that the random vector X generating the data can be decomposed into the sum of independent Gaussian random vectors: $X = V + W$. Then sampling from X with temperature t is equivalent to sampling from V and W with temperature t.

Let us now try to interpret the above in the case of our experiment. Then X denotes the random vector, which selects an image and then changes color to gray at the randomly chosen square of size $K \times K$. Thus we can interpret our model as the independent composition of two operations:

- V: sampling a random image from our original dataset,
- W: changing the color of the randomly chosen square of size $K \times K$ to gray.

Then sampling from V with temperature $t = \sqrt{2}$ is easy, as we simply sample two images, add their latent representations, and process back to input space.

However, a crucial role is played by the operation W. By applying the above reasoning informally, we obtain that for covariance $t\text{Cov}W$, where $t = \sqrt{l}$, we obtain t^2 times randomly covering by squares of $K \times K$. Extrapolating this formula for all t, we obtain the theoretical value for the expected number of squares

$$E(\text{number of missing squares} \mid \text{random sample from temperature } t) = t^2.$$

For verification, we compare this with the values obtained in the experiments.

4.2 Ablation study

Moreover, we provide an ablation study for the expected number of missing squares when sampling with a set temperature t and compare the results to the theoretical results. We sampled a set of 50 images for each of the temperatures and manually calculated the number of missing squares. Then, we calculated empirical expected values

$$E(\text{number of missing squares} \mid \text{random sample from temperature } t)$$

and compared them to the theoretical ones.

As shown in Tab. 1, the empirical expected number of missing squares is much different from theoretical numbers. The reason for these dissimilarities is the fact that Glow models are weak and imperfect learners. Moreover, we suppose that Glow's restriction on the rigid representation could cause such differences. However, we noticed similar expected numbers of missing squares for the sampling temperature $t = 0.7$, which is the original Glow model [8], which suggests that for such t, Glow fulfills its theoretical properties.

5 Conclusion

In this paper, we show that flow-based generative models like Glow [8] can be trained on images with missing data and produce full images without missing parts. The above phenomenon is obtained by producing a disentangled representation in latent space. During the training procedure, the model simultaneously creates representations of objects with missing parts and full images. The above factors are independent, and therefore we can sample from two separated areas of latent to produce full images and items with missing parts.

References

1. Batista, G.E., Monard, M.C., et al.: A study of k-nearest neighbour as an imputation method. His **87**(251-260), 48 (2002)
2. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. arXiv preprint arXiv:1806.07366 (2018)
3. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39**(1), 1–22 (1977)
4. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv:1605.08803 (2016)
5. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT Press (2016)

6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint arXiv:1406.2661 (2014)
7. Gupta, M., Cotter, A., Pfeifer, J., Voevodski, K., Canini, K., Mangylov, A., Moczydlowski, W., Van Esbroeck, A.: Monotonic calibrated interpolated look-up tables. The Journal of Machine Learning Research **17**(1), 3790–3836 (2016)
8. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. arXiv preprint arXiv:1807.03039 (2018)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
10. Li, S.C.X., Marlin, B.: Learning from irregularly-sampled time series: A missing data perspective. In: International Conference on Machine Learning. pp. 5937–5946. PMLR (2020)
11. Little, R.J., Rubin, D.B.: Statistical analysis with missing data, vol. 793. John Wiley & Sons (2019)
12. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of the IEEE international conference on computer vision. pp. 3730–3738 (2015)
13. Mattei, P.A., Frellsen, J.: Miwae: Deep generative modelling and imputation of incomplete data sets. In: International Conference on Machine Learning. pp. 4413–4423. PMLR (2019)
14. McKnight, P.E., McKnight, K.M., Sidani, S., Figueredo, A.J.: Missing data: A gentle introduction. Guilford Press (2007)
15. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International Conference on Machine Learning. pp. 1530–1538. PMLR (2015)
16. Richardson, T.W., Wu, W., Lin, L., Xu, B., Bernal, E.A.: Mcflow: Monte carlo flow models for data imputation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14205–14214 (2020)
17. Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976)
18. Sharpe, P.K., Solly, R.: Dealing with missing values in neural network-based diagnostic systems. Neural Computing & Applications **3**(2), 73–77 (1995)
19. Śmieja, M., Struski, Ł., Tabor, J., Marzec, M.: Generalized rbf kernel for incomplete data. Knowledge-Based Systems **173**, 150–162 (2019)
20. Smieja, M., Struski, Ł., Tabor, J., Zieliński, B., Spurek, P.: Processing of missing data by neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 2724–2734 (2018)
21. Sovilj, D., Eirola, E., Miche, Y., Björk, K.M., Nian, R., Akusok, A., Lendasse, A.: Extreme learning machine for missing data using multiple imputations. Neurocomputing **174**, 220–231 (2016)
22. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. Advances in neural information processing systems **25**, 341–349 (2012)
23. Yeh, R.A., Chen, C., Yian Lim, T., Schwing, A.G., Hasegawa-Johnson, M., Do, M.N.: Semantic image inpainting with deep generative models. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5485–5493 (2017)
24. Yoon, J., Jordon, J., Schaar, M.: Gain: Missing data imputation using generative adversarial nets. In: International Conference on Machine Learning. pp. 5689–5698. PMLR (2018)
25. Zhang, Y., Zheng, Z., Hu, R.: Super resolution using segmentation-prior self-attention generative adversarial network. arXiv preprint arXiv:2003.03489 (2020)

Improved off-policy training of diffusion samplers

Marcin Sendera
Mila, Université de Montréal
Jagiellonian University

Minsu Kim
Mila, Université de Montréal
KAIST

Sarthak Mittal
Mila, Université de Montréal

Pablo Lemos
Mila, Université de Montréal
Ciela Institute
Dreamfold

Luca Scimeca
Mila, Université de Montréal

Jarrid Rector-Brooks
Mila, Université de Montréal
Dreamfold

Alexandre Adam
Mila, Université de Montréal
Ciela Institute

Yoshua Bengio
Mila, Université de Montréal
CIFAR

Nikolay Malkin
Mila, Université de Montréal
University of Edinburgh

{marcin.sendera,...,nikolay.malkin}@mila.quebec

Abstract

We study the problem of training diffusion models to sample from a distribution with a given unnormalized density or energy function. We benchmark several diffusion-structured inference methods, including simulation-based variational approaches and off-policy methods (continuous generative flow networks). Our results shed light on the relative advantages of existing algorithms while bringing into question some claims from past work. We also propose a novel exploration strategy for off-policy methods, based on local search in the target space with the use of a replay buffer, and show that it improves the quality of samples on a variety of target distributions. Our code for the sampling methods and benchmarks studied is made public at (link) as a base for future work on diffusion models for amortized inference.

1 Introduction

Approximating and sampling from complex multivariate distributions is a fundamental problem in probabilistic deep learning [*e.g.*, 26, 34, 25, 47, 56] and in scientific applications [3, 51, 37, 1, 31]. The problem of drawing samples from a distribution given only an unnormalized probability density or energy is particularly challenging in high-dimensional spaces and when the distribution of interest has many separated modes [5]. Sampling methods based on Markov chain Monte Carlo (MCMC) – such as Metropolis-adjusted Langevin [MALA; 23, 64, 63] and Hamiltonian MC [HMC; 19, 30] – may be slow to mix between modes and have a high cost per sample. While variants such as sequential MC [SMC; 24, 12, 15] and nested sampling [68, 9, 42] have better mode coverage, their cost may grow prohibitively with the dimensionality of the problem. This motivates the use of amortized variational inference, *i.e.*, fitting parametric models that sample the target distribution.

Diffusion models, continuous-time stochastic processes that gradually evolve a simple distribution to a complex target, are powerful density estimators with proven mode-mixing properties [14]; as such, they have been widely used in the setting of generative models learned from data [69, 71, 27, 49, 65]. However, the problem of training diffusion models to sample from a distribution with a given black-box density or energy function has attracted less attention. Recent work has drawn connections between diffusion (learning the denoising process) and stochastic control (learning the Föllmer

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

drift [20]), leading to approaches such as the path integral sampler [PIS; 87], denoising diffusion sampler [DDS; 77], and time-reversed diffusion sampler [DIS; 8]; such approaches were recently unified by [62] and [78]. Another line of work [41, 85] is based on continuous generative flow networks (GFlowNets), which are deep reinforcement learning algorithms adapted to variational inference that offer stable off-policy training and thus flexible exploration [45].

Despite the advances in sampling methods and attempts to unify them theoretically [62, 78], the field suffers from some failures in benchmarking and reproducibility, with the works differing in the choice of model architectures, using unstated hyperparameters, and even disagreeing in their definitions of the same target densities (see §B.1). The **first main contribution** of this paper is a unified library for diffusion-structured samplers. The library has a focus on off-policy methods (continuous GFlowNets) but also includes simulation-based variational objectives such as PIS. Using this codebase, we are able to benchmark methods from past work under comparable conditions and confirm claims about exploration strategies and desirable inductive biases, while calling into question other claims on robustness and sample efficiency. Our library also includes several new modeling and training techniques, and we provide preliminary evidence of their utility in possible future work (§5.3).

Our **second contribution** is a study of methods for improving exploration and credit assignment – the propagation of learning signals from the target density to the parameters of earlier sampling steps – in diffusion-structured samplers (§4). First, our results (§5.2) suggest that the technique of utilizing partial trajectory information [43, 54], as done in the diffusion setting by [85], offers little benefit, and a higher training cost, over on-policy [87] or off-policy [41] trajectory-based optimization. Second, we examine the utility of a gradient-based variant which parametrizes the denoising distribution as a correction to a Langevin process [87]. We show that this inductive bias is also beneficial in the off-policy (GFlowNet) setting despite higher computational cost. Finally, motivated by recent approaches in discrete sampling, we propose an efficient exploration technique based on local search in the target space with the use of a replay buffer, which improves sample quality across various target distributions.

2 Prior work

Amortized variational inference approaches use a parametric model q_θ to approximate a given target density p_{target}, typically through stochastic optimization [29, 57, 2]. Notably, explicit density models like autoregressive models and normalizing flows have been extensively utilized in density estimation [59, 18, 80, 21, 50]. However, these models impose structural constraints, thereby limiting their expressive power [13, 22, 86]. The adoption of **diffusion processes** in generative models has stimulated a renewed interest in hierarchical models as density estimators [79, 27, 75]. Approaches like PIS [87] leverage stochastic optimal control for sampling from unnormalized densities, albeit still struggling with scalability in high-dimensional spaces.

Generative flow networks, originally defined in the discrete case by [6, 7], view hierarchical sampling (*i.e.*, stepwise generation) as a sequential decision-making process and represent a synthesis of reinforcement learning and variational inference approaches [45, 89, 72, 17], expanding from specific scientific domains [*e.g.*, 35, 4, 88] to amortized inference over a broader array of latent structures [*e.g.*, 76, 33]. Their ability to efficiently navigate trajectory spaces via off-policy exploration has been crucial, yet they encounter challenges in training dynamics, such as credit assignment and exploration efficiency [44, 43, 54, 58, 67, 38, 36]. These challenges have repercussions in the scalability of these methods in more complex scenarios, which this paper addresses in the continuous case.

3 Setting: Diffusion-structured sampling

Let $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ be a differentiable *energy function* and define $R(\mathbf{x}) = \exp(-\mathcal{E}(\mathbf{x}))$, the *reward* or *unnormalized target density*. Assuming the integral $Z := \int_{\mathbb{R}^d} R(\mathbf{x})\, d\mathbf{x}$ exists, $\mathcal{E}$ defines a *Boltzmann density* $p_{\text{target}}(\mathbf{x}) = R(\mathbf{x})/Z$ on $\mathbb{R}^d$. We are interested in the problems of sampling from p_{target} and approximating the *partition function* Z given access only to $\mathcal{E}$ and possibly to its gradient $\nabla\mathcal{E}$.

We describe two closely related perspectives on this problem: via neural SDEs and stochastic control (§3.1) and via continuous generative flow networks (§3.2).

3.1 Euler-Maruyama hierarchical samplers

Generative modeling with SDEs. Diffusion models assume a continuous-time generative process given by a neural stochastic differential equation [SDE; 74, 53, 66]:

$$d\mathbf{x}_t = u(\mathbf{x}_t, t; \theta)\, dt + g(\mathbf{x}_t, t; \theta)\, d\mathbf{w}_t, \tag{1}$$

where $\mathbf{x}_0$ follows a fixed tractable distribution μ_0 (such as a Gaussian or a point mass). The initial distribution μ_0 and the stochastic dynamics specified by (1) induce marginal densities p_t on $\mathbb{R}^d$ for each $t > 0$. The functions u and g have learnable parameters that we wish to optimize, using some objective, so as to make the terminal density p_1 close to p_{target}. Samples can be drawn from p_1 by sampling $\mathbf{x}_0 \sim \mu_0$ and simulating the SDE (1) to time $t = 1$.

The SDE driving μ_0 to p_{target} is not unique. However, if one fixes a reverse-time SDE, or *noising process*, that pushes p_{target} at $t = 1$ to μ_0 at $t = 0$, then its reverse, the forward SDE (1), is uniquely determined under mild conditions and is called the *denoising process*. For usual choices of the noising process, there are stochastic regression objectives for learning the drift u of the denoising process *given samples from* p_{target}, and the diffusion rate g is available in closed form [27, 71].

Time discretization. In practice, the integration of the SDE (1) is approximated by a discrete-time scheme, the simplest of which is Euler-Maruyama integration. The process (1) is replaced by a discrete-time Markov chain $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \mathbf{x}_{2\Delta t} \to \cdots \to \mathbf{x}_1$, where $\Delta t = \frac{1}{T}$ is the time increment and and T is the number of steps:

$$\mathbf{x}_0 \sim \mu_0, \quad \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + u(\mathbf{x}_t, t; \theta)\Delta t + g(\mathbf{x}_t, t; \theta)\sqrt{\Delta t}\, \mathbf{z}_t \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{2}$$

The density of the transition kernel from $\mathbf{x}_t$ to $\mathbf{x}_{t+\Delta t}$ can explicitly be written as

$$p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+\Delta t}; \mathbf{x}_t + u(\mathbf{x}_t, t; \theta)\Delta t, g(\mathbf{x}_t, t; \theta)^2 \Delta t \mathbf{I}_d), \tag{3}$$

where p_F denotes the transition density of the discretized forward SDE. This density defines a joint distribution over trajectories starting at $\mathbf{x}_0$:

$$p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0) = \prod_{i=0}^{T-1} p_F(\mathbf{x}_{(i+1)\Delta t} \mid \mathbf{x}_{i\Delta t}). \tag{4}$$

Similarly, a discrete-time reverse process $\mathbf{x}_1 \to \mathbf{x}_{1-\Delta t} \to \mathbf{x}_{1-2\Delta t} \to \cdots \to \mathbf{x}_0$ with transition densities $p_B(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t)$ defines a joint distribution[1] via

$$p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1) = \prod_{t=1}^{T} p_B(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}). \tag{5}$$

If the forward and backward processes (starting from μ_0 and p_{target}, respectively) are reverses of each other, then they define the same distribution over trajectories, *i.e.*, for all $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$,

$$\mu_0(\mathbf{x}_0) p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0) = p_{\text{target}}(\mathbf{x}_1) p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1). \tag{6}$$

In particular, the marginal densities of $\mathbf{x}_1$ under the forward and backward processes are then equal to p_{target}, and the forward process can be used to sample the target distribution.

Because the reverse of a process with Gaussian increments is, in general, not itself Gaussian, (6) can be enforced only approximately, but the discrepancy vanishes as $\Delta t \to 0$ (*i.e.*, increments are infinitesimally Gaussian), an application of the central limit theorem that is key to stochastic calculus [53].

SDE learning as hierarchical variational inference. The problem of learning the parameters θ of the forward process so as to enforce (6) is one of hierarchical variational inference. The backward process transforms $\mathbf{x}_1$ into $\mathbf{x}_0$ via a sequence of latent variables $\mathbf{x}_{1-\Delta t}, \ldots, \mathbf{x}_0$, and the forward process aims to match the posterior distribution over these variables and thus to approximately enforce (6).

In the setting of diffusion models learned from data, where one has samples from p_{target}, one can optimize the forward process by minimizing the KL divergence $D_{\text{KL}}(p_{\text{target}} \cdot p_B \| \mu_0 \cdot p_F)$ between the distribution over trajectories given by the reverse process and that given by the forward process.

[1] In the case that μ_0 is a point mass, we assume the distribution $\mathbf{x}_0 \mid \mathbf{x}_{\Delta t}$ to also be a point mass, which has density $p_B(\mathbf{x}_0 \mid \mathbf{x}_{\Delta t}) = 1$ with respect to the measure μ_0.

This is equivalent to the typical training of diffusion models, which optimizes a variational bound on the data log-likelihood (see [70]). However, in the setting of an intractable density p_{target}, unbiased estimators of this divergence are not available. Instead, one can optimize the reverse KL:[2]

$$\begin{aligned} &D_{\text{KL}}(\mu_0 \cdot p_F \| p_{\text{target}} \cdot p_B) \\ &= \int \log \frac{\mu_0(\mathbf{x}_0) p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0)}{p_{\text{target}}(\mathbf{x}_1) p_B(\mathbf{x}_0, \ldots, \mathbf{x}_{1-\Delta t} \mid \mathbf{x}_1)} d\mu_0(\mathbf{x}_0) p_F(\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1 \mid \mathbf{x}_0)\ d\mathbf{x}_{\Delta t} \ \ldots \ d\mathbf{x}_1. \end{aligned} \tag{7}$$

Various estimators of this objective are available. For instance, the path integral sampler objective [PIS; 87] uses the reparametrization trick to express (7) as an expectation over noise variables $\mathbf{z}_t$ that participate in the hierarchical sampling of $\mathbf{x}_{\Delta t}, \ldots, \mathbf{x}_1$, yielding an unbiased gradient estimator, but one that requires backpropagation into the simulation of the forward process. The related denoising diffusion sampler [DDS; 77] applies the same principle in a different integration scheme.

3.2 Euler-Maruyama samplers as GFlowNets

Continuous generative flow networks (GFlowNets) [41] express the problem of enforcing (6) as a reinforcement learning task. In this section, we summarize this interpretation, its connection to neural SDEs, the associated learning objectives, and their relative advantages and disadvantages.

The connection between generative flow networks and diffusion models or SDEs was first made informally by [45] in the distribution-matching setting and by [83] in the maximum-likelihood setting, while the theoretical foundations for continuous GFlowNets were later laid down by [41].

State and action space. To formulate sampling as a sequential decision-making problem, one must define the spaces of states and actions. In the case of sampling by T-step Euler-Maruyama integration, assuming μ_0 is a point mass at $\mathbf{0}$, the state space is

$$\mathcal{S} = \{(\mathbf{0}, 0)\} \cup \left\{(\mathbf{x}, t) : \mathbf{x} \in \mathbb{R}^d, t \in \{\Delta t, 2\Delta t, \ldots, 1\}\right\},$$

with the point $(\mathbf{x}, t)$ representing that the sampling agent is at position $\mathbf{x}$ at time t.

Sampling begins with the *initial state* $\mathbf{x}_0 := (\mathbf{0}, 0)$, proceeds through a sequence of states $(\mathbf{x}_{\Delta t}, \Delta t)$, $(\mathbf{x}_{2\Delta t}, 2\Delta t)$, $\ldots$, and ends at a state $(\mathbf{x}_1, 1)$; states $(\mathbf{x}, t)$ with $t = 1$ are called *terminal states* and their collection is denoted $\mathcal{X}$. From now on, we will often write $\mathbf{x}_t$ in place of the state $(\mathbf{x}_t, t)$ when the time t is clear from context. The sequence of states $\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1$ is called a *complete trajectory*.

The *actions* from a nonterminal state $(\mathbf{x}_t, t)$ correspond to the possible next states $(\mathbf{x}_{t+\Delta t}, t + \Delta t)$ that can be reached from $(\mathbf{x}_t, t)$ by a single step of the Euler-Maruyama integrator.[3]

Forward policy and learning problem. A *(forward) policy* is a collection of continuous distributions over the successor states – states reachable by a single action – of every nonterminal state $(\mathbf{x}, t)$. In our context, this amounts to a collection of conditional probability densities $p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t; \theta)$, representing the density of the transition kernel from $\mathbf{x}_t$ to $\mathbf{x}_{t+\Delta t}$. GFlowNet training optimizes the parameters θ, which may be the weights of a neural network specifying a density over $\mathbf{x}_{t+\Delta t}$ conditioned on $\mathbf{x}_{\Delta t}$.

A policy p_F induces a distribution over complete trajectories $\tau = (\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1)$ via

$$p_F(\tau; \theta) = \prod_{i=0}^{T-1} p_F(\mathbf{x}_{(i+1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \theta).$$

In particular, we get a marginal density over terminal states:

$$p_F^{\top}(\mathbf{x}_1; \theta) = \int p_F(\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1; \theta)\, d\mathbf{x}_{\Delta t} \ldots d\mathbf{x}_{1-\Delta t}. \tag{8}$$

The learning problem solved by GFlowNets is to find the parameters θ of a policy p_F whose terminating density $p_F^{\top}$ is equal to p_{target}, *i.e.*,

$$p_F^{\top}(\mathbf{x}_1; \theta) = \frac{R(\mathbf{x}_1)}{Z} \quad \forall \mathbf{x}_1 \in \mathbb{R}^d. \tag{9}$$

[2] To be precise, the fraction in (7) should be understood as a Radon-Nikodym derivative, which makes sense whether μ_0 is a point mass or a continuous distribution and generalizes to continuous time [8, 62].

[3] Formally, the foundations in [41] require assuming *reference measures* with respect to which the reward and kernel densities are defined. As we deal with Euclidean spaces and assume the Lebesgue measure, readers need not burden themselves with measure theory. We note, however, that this flexibility allows easy generalization to sampling on other spaces, such as any Riemannian manifolds, where other methods do not directly apply.

However, because the integral (8) is intractable and Z is unknown, auxiliary objects must be introduced into optimization objectives to enforce (9), as discussed below.

Notably, if the policy is a Gaussian with mean and variance given by neural networks taking $\mathbf{x}_t$ and t as input, then learning the policy amounts to learning the drift $u(\mathbf{x}_t, t; \theta)$ and diffusion $g(\mathbf{x}_t, t; \theta)$ of a SDE (1), *i.e.*, fitting a neural SDE. **The SDE learning problem in §3.1 is thus the same as that of fitting a GFlowNet with Gaussian policies.**

Backward policy and trajectory balance. A *backward policy* is a collection of conditional probability densities $p_B(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t; \psi)$, representing a probability density of transitioning from $\mathbf{x}_t$ to an ancestor state $\mathbf{x}_{t-\Delta t}$. The backward policy induces a distribution over complete trajectories τ conditioned on their terminal state (cf. (5)):

$$p_B(\tau \mid \mathbf{x}_1; \psi) = \prod_{i=1}^{T} p_B(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \psi),$$

where exceptionally $p_B(\mathbf{x}_0 \mid \mathbf{x}_{\Delta t}) = 1$ as μ_0 is a point mass.

Generalizing a result in the discrete-space setting [44], [41] show that p_F samples from the target distribution (*i.e.*, satisfies (9)) if and only if there exists a backward policy p_B and a scalar Z_θ such that the *trajectory balance conditions* are fulfilled for every complete trajectory $\tau = (\mathbf{x}_0 \to \mathbf{x}_{\Delta t} \to \cdots \to \mathbf{x}_1)$:

$$Z_\theta p_F(\tau; \theta) = R(\mathbf{x}_1) p_B(\tau \mid \mathbf{x}_1; \psi). \tag{10}$$

If these conditions hold, then Z_θ equals the true partition function $Z = \int_{\mathbf{x}} R(\mathbf{x})\, d\mathbf{x}$. The *trajectory balance objective* for a trajectory τ is the squared log-ratio of the two sides of (10), that is:

$$\mathcal{L}_{\text{TB}}(\tau; \theta, \psi) = \left(\log \frac{Z_\theta p_F(\tau; \theta)}{R(\mathbf{x}_1) p_B(\tau \mid \mathbf{x}_1; \psi)} \right)^2. \tag{11}$$

One can thus achieve (9) by minimizing to zero the loss $\mathcal{L}_{\text{TB}}(\tau; \theta, \psi)$ with respect to the parameters θ and ψ, where the trajectories τ used for training are sampled from some *training policy* $\pi(\tau)$. While it is possible to optimize (11) with respect to the parameters of both the forward and backward policies, in some learning problems, one *fixes* the backward policy and only optimizes the parameters of p_F and the estimate of the partition function Z_θ. For example, for most experiments in §5, we fix the backward policy to a discretized Brownian bridge, following past work.

Off-policy optimization. Unlike the KL objective (7), whose gradient involves an expectation over the distribution of trajectories under the current forward process, (11) can be optimized off-policy, *i.e.*, using trajectories sampled from an arbitrary distribution π. Because minimizing $\mathcal{L}_{\text{TB}}(\tau; \theta, \psi)$ to 0 for all τ in the support of π will achieve (9), π can be taken be any distribution with full support, so as to promote discovery of modes of the target distribution. Various choices motivated by reinforcement learning techniques have been proposed, including noisy exploration or tempering [6], replay buffers [16], Thompson sampling [58], and backward traces from terminal states obtained by MCMC [42]. In the continuous case, [45, 41] proposed to simply add a small constant to the policy variance when sampling trajectories for training. Off-policy optimization is a key advantage of GFlowNets over variational methods such as PIS, which require on-policy optimization [45].

However, when $\mathcal{L}_{\text{TB}}$ happens to be optimized on-policy, *i.e.*, using trajectories sampled from the policy p_F itself, we get an unbiased estimator of the gradient of the KL divergence (7) with respect to p_F's parameters up to a constant [61, 45, 89], that is:

$$\mathbb{E}_{\tau \sim p_F(\tau)} \left[\nabla_{\theta'} \mathcal{L}_{\text{TB}}(\tau; \theta, \psi) \right] = 2 \nabla_{\theta'} D_{\text{KL}}(p_F(\tau; \theta) \| p_{\text{target}}(\mathbf{x}_1) p_B(\tau \mid \mathbf{x}_1; \psi)),$$

where $\nabla_{\theta'}$ denotes the gradient with respect to the parameters of p_F, but not Z_θ. This unbiased estimator tends to have higher variance than the reparametrization-based estimator used by PIS. On the other hand, it does not require backpropagation through the simulation of the forward process and can be used to optimize the parameters of both the forward and backward policies.

Other objectives. The trajectory balance objective (11) is not the only possible objective that can be used to enforce (9). A notable generalization is *subtrajectory balance* [SubTB; 43], which involves modeling a scalar *state flow* $f(\mathbf{x}_t; \theta)$ associated with each state $\mathbf{x}_t$ – intended to model the marginal density of the forward process at $\mathbf{x}_t$ – and enforcing *subtrajectory balance* conditions for all partial trajectories $\mathbf{x}_{m\Delta t} \to \mathbf{x}_{(m+1)\Delta t} \to \cdots \to \mathbf{x}_{n\Delta t}$:

$$f(\mathbf{x}_{m\Delta t}; \theta) \prod_{i=m}^{n-1} p_F(\mathbf{x}_{(i+1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \theta) = f(\mathbf{x}_{n\Delta t}; \theta) \prod_{i=m+1}^{n} p_B(\mathbf{x}_{(i-1)\Delta t} \mid \mathbf{x}_{i\Delta t}; \psi), \tag{12}$$

where for terminal states $f(\mathbf{x}_1) = R(\mathbf{x}_1)$. This approach has some computational overhead associated with training the state flow, but has been shown to be effective in discrete-space settings, especially when combined with the *forward-looking* reward shaping scheme proposed by [54]. It has also been tested in the continuous case, but our experimental results suggest that it offers little benefit over the TB objective in the diffusion setting (see §4.1 and §B.1).

It is also worth noting the off-policy VarGrad estimator [52, 61], rediscovered for GFlowNets by [84]. Like TB, VarGrad can be optimized over trajectories drawn off-policy. Rather than enforcing (10) for every trajectory, VarGrad optimizes the empirical *variance* (over a minibatch) of the log-ratio of the two sides of (10). As noted by [45], this is equivalent to minimizing $\mathcal{L}_{\text{TB}}$ first with respect to $\log Z_\theta$ to optimality over the batch, then with respect to the parameters of p_F.

4 Exploration and credit assignment in continuous GFlowNets

The main challenges in training off-policy sampling models are exploration efficiency (discovery of high-reward states) and credit assignment (propagation of reward signals to the actions that led to them). We describe several new and existing methods for addressing these challenges in the context of diffusion-structured GFlowNets. These techniques will be empirically studied and compared in §5.

4.1 Credit assignment methods

Partial energies and subtrajectory-based learning. [85] studied the diffusion sampler learning problem introduced by [41], but replaced the TB learning objective with the SubTB objective.[4] In addition, an inductive bias resembling the geometric interpolation in [46] was used for the state flow function:

$$\log f(\mathbf{x}_t;\theta) = (1-t)\log p_t^{\text{ref}}(\mathbf{x}_t) + t\log R(\mathbf{x}_t) + \text{NN}(\mathbf{x}_t, t;\theta), \tag{13}$$

where NN is a neural network and $p_t^{\text{ref}}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; 0, \sigma^2 t I_d)$ is the marginal density of a Brownian motion with rate σ at $\mathbf{x}_t$. The use of the target density $\log R(\mathbf{x}_t) = -\mathcal{E}(\mathbf{x}_t)$ in the state flow function was hypothesized to provide an effective signal driving the sampler to high-density states at early steps in the trajectory. Such an inductive bias on the state flow was called *forward-looking* (FL) by [54], and we will refer to this method as **FL-SubTB** in §5.

Langevin dynamics inductive bias. [87] proposed an inductive bias on the architecture of the drift of the neural SDE $u(\mathbf{x}_t, t;\theta)$ (in GFlowNet terms, the mean of the Gaussian density $p_F(\mathbf{x}_{t+\Delta t} \mid \mathbf{x}_t;\theta)$) that resembles a Langevin process on the target distribution. One writes

$$u(\mathbf{x}_t, t;\theta) = \text{NN}_1(\mathbf{x}_t, t;\theta) + \text{NN}_2(t;\theta)\nabla\mathcal{E}(\mathbf{x}_t), \tag{14}$$

where NN_1 and NN_2 are neural networks outputting a vector and a scalar, respectively. The second term in (14) is a scaled gradient of the target energy – the drift of a Langevin SDE – and the first term is a learned correction. This inductive bias, which we name the *Langevin parametrization* (**LP**), was shown to improve the efficiency of PIS. We will study its effect on continuous GFlowNets in §5.

The inductive bias (14) placed on policies represents a different way of incorporating the reward signal at intermediate steps in the trajectory and can steer the sampler towards low-energy regions. It contrasts with (13) in that it provides the gradient of the energy directly to the policy, rather than just using the energy to provide a learning signal to policies via the parametrization of the log-state flow (13).

Considerations of the continuous-time limit lead us to conjecture that the Langevin parametrization (14) with NN_1 independent of $\mathbf{x}_t$ is *equivalent* to the forward-looking flow (13) in the limit of small time increments $\Delta t \to 0$, *i.e.*, they induce the same asymptotics of the discrepancy in the SubTB constraints (12) over short partial trajectories. Such theoretical analysis can be the subject of future work.

4.2 A new method for off-policy exploration with local search and replay buffer

Local search with parallel MALA. The FL and LP inductive biases both induce computational overhead: either in the evaluation and optimization of a state flow or in the need to evaluate the energy gradient at every step of sampling (see §C.3). We present an alternative technique that does not induce additional computation cost per training trajectory.

[4] Despite the claimed benefits of FL-SubTB for diffusion samplers, we discovered that [85] modifies critical experimental variables in comparisons and reports irreproducible results; see §B.1.

Table 1: Log-partition function estimation errors for unconditional modeling tasks (mean and standard deviation over 5 runs). The four groups of models are: MCMC-based samplers, simulation-driven variational methods, baseline GFlowNet methods with different learning objectives, and methods augmented with Langevin parametrization and local search. See §C.1 for additional metrics.

Energy →	25GMM ($d = 2$)		Funnel ($d = 10$)		Manywell ($d = 32$)		LGCP ($d = 1600$)	
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\log \hat{Z}$	$\log \hat{Z}^{\text{RW}}$
SMC	0.569±0.010		0.561±0.801		14.99±1.078		*See discussion in §B.1*	
GGNS [42]	0.016±0.042		0.033±0.173		0.292±0.454		N/A	
DIS [8]	1.125±0.056	0.986±0.011	0.839±0.169	0.093±0.038	10.52±1.02	3.05±0.46	299.83±0.67	361.15±6.48
DDS [77]	1.760±0.08	0.746±0.389	0.424±0.049	0.206±0.033	7.36±2.43	0.23±0.05	471.64±1.20	489.30±0.62
PIS [87]	1.769±0.104	1.274±0.218	0.534±0.008	0.262±0.008	3.85±0.03	2.69±0.04	381.14±1.42	414.42±2.06
+ LP [87]	1.799±0.051	0.225±0.583	0.587±0.012	0.285±0.044	13.19±0.82	0.07±0.85	471.45±0.18	487.82±2.26
TB [41]	1.176±0.109	1.071±0.112	0.690±0.018	0.239±0.192	4.01±0.04	2.67±0.02	336.70±56.22	379.50±49.99
TB + Expl. [41]	0.560±0.302	0.422±0.320	0.749±0.015	0.226±0.138	4.01±0.05	2.68±0.06	346.10±55.54	389.21±44.13
VarGrad + Expl.	0.615±0.241	0.487±0.250	0.642±0.010	0.250±0.112	4.01±0.05	2.69±0.06	370.37±0.26	410.37±6.70
FL-SubTB	1.127±0.010	1.020±0.010	0.527±0.011	0.182±0.142	3.98±0.07	2.72±0.05	365.20±6.08	402.65±8.36
+ LP [85]	0.209±0.025	0.011±0.024	0.563±0.021	0.155±0.317	4.23±0.12	2.66±0.22	465.44±1.26	483.90±1.95
TB + Expl. + LS (*ours*)	0.171±0.013	0.004±0.011	0.653±0.025	0.285±0.099	4.57±2.13	0.19±0.29	384.90±0.83	419.55±2.14
TB + Expl. + LP (*ours*)	0.206±0.018	0.011±0.010	0.666±0.615	0.051±0.616	7.46±1.74	1.06±1.11	452.82±1.50	477.62±1.79
TB + Expl. + LP + LS (*ours*)	0.190±0.013	0.007±0.011	0.768±0.052	0.264±0.063	4.68±0.49	0.07±0.17	471.14±0.25	489.03±1.38
VarGrad + Expl. + LP + LS (*ours*)	0.207±0.016	0.015±0.015	0.920±0.118	0.256±0.037	4.11±0.45	0.02±0.21	468.65±0.63	487.34±1.34

Highlight : mean indistinguishable from best in column with $p < 0.05$ under one-sided Welch unpaired t-test.

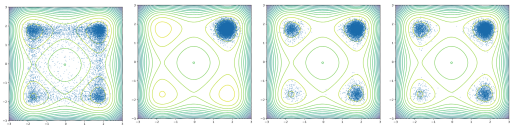


Figure 1: Two-dimensional projections of **Manywell** samples from models trained by different algorithms. Our proposed replay buffer with local search is capable of preventing mode collapse.

To enhance the quality of samples during training, we incorporate local search into the exploration process, motivated by the success of local exploration [82, 32, 39] and replay buffer [*e.g.*, 16] methods for GFlowNets in discrete spaces. Unlike these methods, which define MCMC kernels via the GFlowNet policies, our method leverages parallel Metropolis-adjusted Langevin (MALA) directly in the target space.

In detail, we initially sample M candidates from the sampler: $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\} \sim p_F^{\top}(\cdot)$. Subsequently, we run parallel MALA across M chains over K transitions , with the initial states of the Markov chain being $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\}$. After the $K_{\text{burn-in}}$ burn-in transitions, the accepted samples are stored in a local search buffer $\mathcal{D}_{\text{LS}}$. We occasionally update the buffer using MALA steps and replay samples from it to minimize the computational demands of iterative local search. MALA steps are far more parallelizable than sampler training and need to be made only rarely (as the buffer is much larger than the training batch size), so the overhead of local search is small.

Training with local search and replay buffer. To train samplers with the aid of the buffer, we draw a sample $\mathbf{x}$ from $\mathcal{D}_{\text{LS}}$ (uniformly or using a prioritization scheme, §E), sample a trajectory τ leading to $\mathbf{x}$ from the backward process, and make a gradient update on the objective (*e.g.*, TB) associated with τ.

When training with local search guidance, we alternate two steps, inspired by [42], who alternate training on forward trajectories and backward trajectories initialized at a *fixed* set of MCMC samples. Step A involves training with on-policy or exploratory forward sampling while Step B uses samples drawn from the local search buffer described above. This allows the sampler to explore both diversified samples (Step A) and low-energy samples (Step B). See §E for detailed pseudocode of adaptive-step parallel MALA and local search-guided GFlowNet training.

5 Experiments

We conduct comprehensive benchmarks of various diffusion-structured samplers, encompassing both GFlowNet samplers and methods such as PIS. For the GFlowNet samplers, we investigate a range of techniques, including different exploration strategies and loss functions. Additionally, we examine the efficacy of the Langevin parametrization and the newly proposed local search with buffer.

5.1 Tasks and baselines

We explore two types of tasks, with more details provided in §B: sampling from energy distributions – a 2-dimensional mixture of Gaussians with 25 modes (**25GMM**), the 10-dimensional **Funnel**, the 32-dimensional **Manywell** distribution, and the 1600-dimensional **Log-Gaussian Cox process** –

and *conditional* sampling from the latent posterior of a variational autoencoder (**VAE**; [40, 60]). This allows us to investigate both unconditional and conditional generative modeling techniques.

We evaluate three algorithm categories:

(1) **Traditional sampling methods:** We consider a standard Sequential Monte Carlo (SMC) implementation and a state-of-the-art nested sampling method (GGNS, [42]).
(2) **Simulation-driven variational approaches:** DIS [8], DDS [77], and PIS [87].
(3) **Diffusion-based GFlowNet samplers:** Our evaluation focuses on TB-based training and the enhancements described in §4: the VarGrad estimator (**VarGrad**), off-policy exploration (**Expl.**), Langevin parametrization (**LP**), and local search (**LS**). Additionally, we assess the FL-SubTB-based continuous GFlowNet as studied by [85] for a comprehensive comparison.

For (2) and (3), we employ a consistent neural architecture across methods (details in §D).

Learning problem and fixed backward process. In our main experiments, we borrow the modeling setting from [87]. We aim to learn a Gaussian forward policy p_F that samples from the target distribution in $T = 100$ steps ($\Delta t = 0.01$). Just as in past work [87, 41, 85], the backward process is fixed to a discretized Brownian bridge with a noise rate σ that depends on the domain; explicitly,

$$p_B(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-\Delta t}; \frac{t-\Delta t}{t}\mathbf{x}_t, \frac{t-\Delta t}{t}\sigma^2 \Delta t \mathbf{I}_d\right), \tag{15}$$

understood to be a point mass at $\mathbf{0}$ when $t = \Delta t$. To keep the learning problem consistent with past work, we fix the variance of the forward policy p_F to σ^2. This simplification is justified in continuous time, when the forward and reverse SDEs have the same diffusion rate. However, in §5.3, we will provide evidence that *learning* the forward policy's variance is quite beneficial for shorter trajectories.

Benchmarking metrics. To evaluate diffusion-based samplers, we use two metrics from past work [87, 41], which we restate in our notation. Given any forward policy p_F, we have a variational lower bound on the log-partition function $\log Z = \int_{\mathbb{R}^d} R(\mathbf{x})\, d\mathbf{x}$:

$$\log \int_{\mathbb{R}^d} R(\mathbf{x})\, d\mathbf{x} = \log \mathop{\mathbb{E}}_{\tau=(\cdots\to\mathbf{x}_1)\sim p_F(\tau)}\left[\frac{R(\mathbf{x}_1)p_B(\tau \mid \mathbf{x}_1)}{p_F(\tau)}\right] \geq \mathop{\mathbb{E}}_{\tau=(\cdots\to\mathbf{x}_1)\sim p_F(\tau)}\left[\log \frac{R(\mathbf{x}_1)p_B(\tau \mid \mathbf{x}_1)}{p_F(\tau)}\right].$$

We use a K-sample ($K = 2000$) Monte Carlo estimate of this expectation, $\log \hat{Z}$, as a metric, which equals the true $\log Z$ if p_F and p_B jointly satisfy (10) and thus p_F samples from the target distribution. We also employ an importance-weighted variant, which emphasizes mode coverage over accurate local modeling:

$$\log \hat{Z}^{\mathrm{RW}} := \log \sum_{i=1}^{K}\left[\frac{R(\mathbf{x}_1^{(i)})p_B(\tau^{(i)} \mid \mathbf{x}_1^{(i)})}{p_F(\tau^{(i)})}\right],$$

where $\tau^{(1)}, \ldots, \tau^{(K)}$ are trajectories sampled from p_F and leading to terminal states $\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_1^{(K)}$. The estimator $\log \hat{Z}^{\mathrm{RW}}$ is also a lower bound on $\log Z$ and approaches it as $K \to \infty$ [10]. In the unconditional modeling benchmarks, we compare both estimators to the true log-partition function, which is known analytically for all tasks except LGCP (leading to discrepancies in past work; see §B.1).

In addition, we include a sample-based metric (2-Wasserstein distance); see §C.1.

5.2 Results

Unconditional sampling. We report the metrics for all algorithms and energies in Table 1.

We observe that TB's performance is generally modest without additional exploration and credit assignment mechanisms, except on the **Funnel** task, where variations in performance across methods are negligible. This confirms hypotheses from past work about the importance of off-policy exploration [45, 41] and the importance of improved credit assignment [85]. On the other hand, our results do not show a consistent

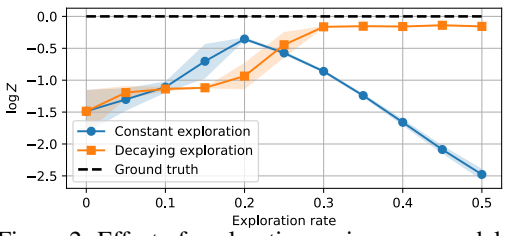


Figure 2: Effect of exploration variance on models trained with TB on the **25GMM** energy. Exploration promotes mode discovery, but should be decayed over time to optimally allocate the modeling power to high-likelihood trajectories.

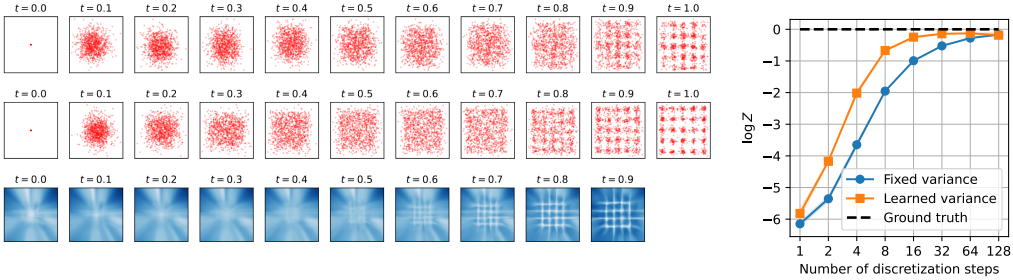


Figure 3: **Left:** Distribution of $\mathbf{x}_0, \mathbf{x}_{0.1}, \ldots, \mathbf{x}_1$ learned by 10-step samplers with fixed (*top*) and learned (*middle*) forward policy variance on the **25GMM** energy. The last step of sampling the *fixed-variance* model adds Gaussian noise of a variance close to that of the components of the target distribution, preventing the the sampler from sharply capturing the modes. The last row shows the policy variance learned as a function of $\mathbf{x}_t$ at various time steps t (white is high variance, blue is low), showing that less noise is added around the peaks near $t = 1$. The two models' log-partition function estimates are -1.67 and -0.62, respectively. **Right:** For varying number of steps T, we plot the $\log \hat{Z}$ obtained by models with fixed and learned variance. **Learning policy variances gives similar samplers with fewer steps.**

and significant improvement of the FL-SubTB
objective used by [85] over TB. Replacing TB with the VarGrad objective yields similar results.

The simple off-policy exploration method of adding variance to the policy notably enhances performance on the **25GMM** task. We investigate this phenomenon in more detail in Fig. 2, finding that exploration that slowly decreases over the course of training is the best strategy.

On the other hand, our local search-guided exploration with a replay buffer (LS) leads to a substantial improvement in performance, surpassing or competing with GFlowNet baselines, non-GFlowNet baselines, and non-amortized sampling methods in most tasks and metrics. This advantage is attributed to efficient exploration and the ability to replay past low-energy regions, thus preventing mode collapse during training (Fig. 1). Further details on LS enhancements are discussed in §E with ablation studies in §E.2.

Incorporating Langevin parametrization (LP) into TB or FL-SubTB results in notable performance improvements (despite being 2-3× slower per iteration), indicating that previous observations [87] transfer to off-policy algorithms. Compared to FL-SubTB, which aims for enhanced credit assignment through partial energy, LP achieves superior credit assignment leveraging gradient information, akin to partial energy in continuous time. LP is either superior or competitive across most tasks and metrics.

In §C.3, we study the scaling of the algorithms with dimension, showing efficiency of the proposed LS.

Table 2: Log-likelihood estimates on a test set for a pretrained VAE decoder on MNIST. The latent being sampled is 20-dimensional. The VAE's training ELBO (Gaussian encoder) was ≈ -101.

Algorithm ↓ Metric →	$\log \hat{Z}$	$\log \hat{Z}^{\text{RW}}$
GGNS [42]	$-82.406_{\pm 0.882}$	
PIS [87]	$-102.54_{\pm 0.437}$	$-47.753_{\pm 2.821}$
+ LP [87]	$-99.890_{\pm 0.373}$	$-47.326_{\pm 0.777}$
TB [41]	$-162.73_{\pm 35.55}$	$-61.407_{\pm 17.83}$
VarGrad	$-102.54_{\pm 0.934}$	$-46.502_{\pm 1.018}$
TB + Expl. [41]	$-148.04_{\pm 4.046}$	$-49.967_{\pm 5.683}$
FL-SubTB	$-147.992_{\pm 22.671}$	$-54.196_{\pm 3.996}$
+ LP [85]	$-111.536_{\pm 1.027}$	$-47.640_{\pm 1.313}$
TB + Expl. + LS (*ours*)	$-245.78_{\pm 13.80}$	$-55.378_{\pm 9.125}$
TB + Expl. + LP (*ours*)	$-112.45_{\pm 0.671}$	$-48.827_{\pm 1.787}$
TB + Expl. + LP + LS (*ours*)	$-117.26_{\pm 2.502}$	$-49.157_{\pm 2.051}$
VarGrad + Expl. (*ours*)	$-103.39_{\pm 0.691}$	$-47.318_{\pm 1.981}$
VarGrad + Expl. + LS (*ours*)	$-105.40_{\pm 0.882}$	$-48.235_{\pm 0.891}$
VarGrad + Expl. + LP (*ours*)	$-99.472_{\pm 0.259}$	$-46.574_{\pm 0.736}$
VarGrad + Expl. + LP + LS (*ours*)	$-99.783_{\pm 0.312}$	$-46.245_{\pm 0.543}$

Conditional sampling. For the VAE task, we observe that the performance of the baseline GFlowNet-based samplers is generally worse than that of the simulation-based PIS (Table 2). While LP and LS improve the performance of TB, they do not close the gap in likelihood estimation; however, with the VarGrad objective, the performance is competitive with or superior to PIS. We hypothesize that this discrepancy is due to the difficulty of fitting the conditional log-partition function estimator, which is required for the TB objective but not for VarGrad, which only learns the policy. (In Fig. D.1 we show decoded samples encoded using the best-performing diffusion encoder.)

5.3 Extensions to general SDE learning problems

Our implementation of diffusion-structured generative flow networks includes several additional options that diverge from the modeling assumptions made in most past work in the field. Notably, it features the ability to:

- **optimize the backward (noising) process** – not only the denoising process – as was done for related learning problems in [11, 62, 78];
- **learn the forward process's diffusion rate** $g(\mathbf{x}_t, t; \theta)$, not only the mean $u(\mathbf{x}_t, t; \theta)$;
- assume a **varying noise schedule** for the backward process, making it possible to train models with standard noising SDEs used for diffusion models for images.

These extensions will allow others to build on our implementation and apply it to problems such as finetuning diffusion models trained on images with a GFlowNet objective.

As noted in §5.1, in the main experiments we fixed the diffusion rate of the learned forward process, an assumption inherited from all past work and justified in the continuous-time limit. However, we perform an experiment to show the importance of extensions such as learning the forward variance in discrete time. Fig. 3 shows the samples of models on the **25GMM** energy following the experimental setup of [42]. We see that when the forward policy's variance is learned, the model can better capture the details of the target distributions, choosing a low variance in the vicinity of the peaks to avoid 'blurring' them through the noise added in the last step of sampling.

In §C.2, we include preliminary results using a variance-preserving backward process, as commonly used in diffusion models, in place of the reversed Brownian motion used in the main experiments.

The ability to model distributions accurately in fewer steps is important for computational efficiency. Future work can consider ways to improve performance in coarse time discretizations, such as non-Gaussian transitions, whose utility in diffusion models trained from data has been demonstrated [81].

6 Conclusion

We have presented a study of diffusion-structured samplers for amortized inference over continuous variables. Our results suggest promising techniques for improving the mode coverage and efficiency of these models. Future work on applications can consider inference of high-dimensional parameters of dynamical systems and inverse problems. In probabilistic machine learning, extensions of this work should study integration of our amortized sequential samplers as variational posteriors in an expectation-maximization loop for training latent variable models, as was recently done for discrete compositional latents by [32], and for sampling Bayesian posteriors over high-dimensional model parameters. The most important direction of theoretical work is understanding the continuous-time limit ($T \to \infty$) of all the algorithms we have studied.

Acknowledgments

We thank Cheng-Hao Liu for assistance with methods from prior work, as well as Julius Berner, Víctor Elvira, Lorenz Richter, Alexander Tong, and Siddarth Venkatraman for helpful discussions and suggestions.

The authors acknowledge funding from UNIQUE, CIFAR, NSERC, Intel, Recursion Pharmaceuticals, and Samsung. The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (`https://alliancecan.ca`), Mila (`https://mila.quebec`), and NVIDIA. The research of M.S. was in part funded by National Science Centre, Poland, 2022/45/N/ST6/03374.

References

[1] Adam, A., Coogan, A., Malkin, N., Legin, R., Perreault-Levasseur, L., Hezaveh, Y., and Bengio, Y. Posterior samples of source galaxies in strong gravitational lenses with score-based priors. *arXiv preprint arXiv:2211.03812*, 2022.

[2] Agrawal, A. and Domke, J. Amortized variational inference for simple hierarchical models. *Neural Information Processing Systems (NeurIPS)*, 2021.

[3] Albergo, M. S., Kanwar, G., and Shanahan, P. E. Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.

[4] Atanackovic, L., Tong, A., Wang, B., Lee, L. J., Bengio, Y., and Hartford, J. DynGFN: Towards bayesian inference of gene regulatory networks with GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2023.

[5] Bandeira, A. S., Maillard, A., Nickl, R., and Wang, S. On free energy barriers in Gaussian priors and failure of cold start MCMC for high-dimensional unimodal distributions. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 381, 2022.

[6] Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

[7] Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[8] Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

[9] Buchner, J. Nested sampling methods. *arXiv preprint arXiv:2101.09675*, 2021.

[10] Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. *International Conference on Learning Representations (ICLR)*, 2016.

[11] Chen, T., Liu, G.-H., and Theodorou, E. A. Likelihood training of Schrödinger bridge using forward-backward SDEs theory. *International Conference on Learning Representations (ICLR)*, 2022.

[12] Chopin, N. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[13] Cornish, R., Caterini, A., Deligiannidis, G., and Doucet, A. Relaxing bijectivity constraints with continuously indexed normalising flows. *International Conference on Machine Learning (ICML)*, 2020.

[14] De Bortoli, V. Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research (TMLR)*, 2022.

[15] Del Moral, P., Doucet, A., and Jasra, A. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

[16] Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

[17] Deleu, T., Nouri, P., Malkin, N., Precup, D., and Bengio, Y. Discrete probabilistic inference as control in multi-path environments. *Uncertainty in Artificial Intelligence (UAI)*, 2024.

[18] Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

[19] Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[20] Föllmer, H. An entropy approach to the time reversal of diffusion processes. pp. 156–163, 1985.

[21] Gao, C., Isaacson, J., and Krause, C. i-flow: High-dimensional integration and sampling with normalizing flows. *Machine Learning: Science and Technology*, 1(4):045023, 2020.

[22] Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

[23] Grenander, U. and Miller, M. I. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

[24] Halton, J. H. Sequential Monte Carlo. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pp. 57–78. Cambridge University Press, 1962.

[25] Harrison, J., Willes, J., and Snoek, J. Variational Bayesian last layers. *International Conference on Learning Representations (ICLR)*, 2024.

[26] Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of Bayesian neural networks. *International Conference on Machine Learning (ICML)*, 2015.

[27] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020.

[28] Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. NeuTralizing bad geometry in Hamiltonian Monte Carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

[29] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. W. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14:1303–1347, 2013.

[30] Hoffman, M. D., Gelman, A., et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1):1593–1623, 2014.

[31] Holdijk, L., Du, Y., Hooft, F., Jaini, P., Ensing, B., and Welling, M. Stochastic optimal control for collective variable free sampling of molecular transition paths. *Neural Information Processing Systems (NeurIPS)*, 2023.

[32] Hu, E. J., Malkin, N., Jain, M., Everett, K., Graikos, A., and Bengio, Y. GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*, 2023.

[33] Hu, E. J., Jain, M., Elmoznino, E., Kaddar, Y., Lajoie, G., Bengio, Y., and Malkin, N. Amortizing intractable inference in large language models. *International Conference on Learning Representations (ICLR)*, 2024.

[34] Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. What are Bayesian neural network posteriors really like? *International Conference on Machine Learning (ICML)*, 2021.

[35] Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., et al. Biological sequence design with gflownets. *International Conference on Machine Learning (ICML)*, 2022.

[36] Jang, H., Kim, M., and Ahn, S. Learning energy decompositions for partial inference of GFlowNets. *International Conference on Learning Representations (ICLR)*, 2024.

[37] Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. Torsional diffusion for molecular conformer generation. *Neural Information Processing Systems (NeurIPS)*, 2022.

[38] Kim, M., Ko, J., Zhang, D., Pan, L., Yun, T., Kim, W., Park, J., and Bengio, Y. Learning to scale logits for temperature-conditional GFlowNets. *arXiv preprint arXiv:2310.02823*, 2023.

[39] Kim, M., Yun, T., Bengio, E., Zhang, D., Bengio, Y., Ahn, S., and Park, J. Local search GFlowNets. *International Conference on Learning Representations (ICLR)*, 2024.

[40] Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[41] Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

[42] Lemos, P., Malkin, N., Handley, W., Bengio, Y., Hezaveh, Y., and Perreault-Levasseur, L. Improving gradient-guided nested sampling for posterior inference. *arXiv preprint arXiv:2312.03911*, 2023.

[43] Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A., Bosc, T., Bengio, Y., and Malkin, N. Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*, 2022.

[44] Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *Neural Information Processing Systems (NeurIPS)*, 2022.

[45] Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

[46] Máté, B. and Fleuret, F. Learning interpolations between Boltzmann densities. *Transactions on Machine Learning Research (TMLR)*, 2023.

[47] Mittal, S., Bracher, N. L., Lajoie, G., Jaini, P., and Brubaker, M. A. Exploring exchangeable dataset amortization for bayesian posterior inference. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

[48] Møller, J., Syversveen, A., and Waagepetersen, R. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998. ISSN 0303-6898.

[49] Nichol, A. and Dhariwal, P. Improved denoising diffusion probabili1stic models. *International Conference on Machine Learning (ICML)*, 2021.

[50] Nicoli, K. A., Nakajima, S., Strodthoff, N., Samek, W., Müller, K.-R., and Kessel, P. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, 101(2):023304, 2020.

[51] Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

[52] Nüsken, N. and Richter, L. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial Differential Equations and Applications*, 2(4):48, 2021.

[53] Øksendal, B. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 2003.

[54] Pan, L., Malkin, N., Zhang, D., and Bengio, Y. Better training of GFlowNets with local credit and incomplete trajectories. *International Conference on Machine Learning (ICML)*, 2023.

[55] Pillai, N. S., Stuart, A. M., and Thiéry, A. H. Optimal scaling and diffusion limits for the langevin algorithm in high dimensions. *The Annals of Applied Probability*, 22(6), December 2012.

[56] Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE transactions on neural networks and learning systems*, 33(4):1452–1466, 2020.

[57] Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. *Artificial Intelligence and Statistics (AISTATS)*, 2014.

[58] Rector-Brooks, J., Madan, K., Jain, M., Korablyov, M., Liu, C.-H., Chandar, S., Malkin, N., and Bengio, Y. Thompson sampling for improved exploration in GFlowNets. *arXiv preprint arXiv:2306.17693*, 2023.

[59] Rezende, D. and Mohamed, S. Variational inference with normalizing flows. *International Conference on Machine Learning (ICML)*, 2015.

[60] Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning (ICML)*, 2014.

[61] Richter, L., Boustati, A., Nüsken, N., Ruiz, F. J. R., and Ömer Deniz Akyildiz. VarGrad: A low-variance gradient estimator for variational inference. *Neural Information Processing Systems (NeurIPS)*, 2020.

[62] Richter, L., Berner, J., and Liu, G.-H. Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*, 2023.

[63] Roberts, G. O. and Rosenthal, J. S. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1): 255–268, 1998.

[64] Roberts, G. O. and Tweedie, R. L. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

[65] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[66] Särkkä, S. and Solin, A. *Applied stochastic differential equations*. Cambridge University Press, 2019.

[67] Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. Towards understanding and improving GFlowNet training. *International Conference on Machine Learning (ICML)*, 2023.

[68] Skilling, J. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833 – 859, 2006. doi: 10.1214/06-BA127. URL https://doi.org/10.1214/06-BA127.

[69] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning (ICML)*, 2015.

[70] Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *Neural Information Processing Systems (NeurIPS)*, 2021.

[71] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2021.

[72] Tiapkin, D., Morozov, N., Naumov, A., and Vetrov, D. Generative flow networks as entropy-regularized RL. *arXiv preprint arXiv:2310.12934*, 2023.

[73] Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Neural Information Processing Systems (NeurIPS)*, 2020.

[74] Tzen, B. and Raginsky, M. Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[75] Tzen, B. and Raginsky, M. Theoretical guarantees for sampling and inference in generative models with latent diffusions. *Conference on Learning Theory (CoLT)*, 2019.

[76] van Krieken, E., Thanapalasingam, T., Tomczak, J., van Harmelen, F., and ten Teije, A. A-NeSI: A scalable approximate method for probabilistic neurosymbolic inference. *Neural Information Processing Systems (NeurIPS)*, 2023.

[77] Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

[78] Vargas, F., Padhy, S., Blessing, D., and Nüsken, N. Transport meets variational inference: Controlled Monte Carlo diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

[79] Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[80] Wu, H., Köhler, J., and Noé, F. Stochastic normalizing flows. *Neural Information Processing Systems (NeurIPS)*, 2020.

[81] Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. *International Conference on Leraning Representations (ICLR)*, 2022.

[82] Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. *International Conference on Machine Learning (ICML)*, 2022.

[83] Zhang, D., Chen, R. T. Q., Malkin, N., and Bengio, Y. Unifying generative models with GFlowNets and beyond. *arXiv preprint arXiv:2209.02606*, 2023.

[84] Zhang, D., Rainone, C., Peschl, M., and Bondesan, R. Robust scheduling with GFlowNets. *International Conference on Learning Representations (ICLR)*, 2023.

[85] Zhang, D., Chen, R. T. Q., Liu, C.-H., Courville, A., and Bengio, Y. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*, 2024.

[86] Zhang, Q. and Chen, Y. Diffusion normalizing flow. *Neural Information Processing Systems (NeurIPS)*, 2021.

[87] Zhang, Q. and Chen, Y. Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

[88] Zhu, Y., Wu, J., Hu, C., Yan, J., Hsieh, C.-Y., Hou, T., and Wu, J. Sample-efficient multi-objective molecular optimization with GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2023.

[89] Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. A variational perspective on generative flow networks. *Transactions on Machine Learning Research (TMLR)*, 2023.

A Code and hyperparameters

Code is available at https://github.com/GFNOrg/gfn-diffusion and will continue to be maintained and extended.

Below are commands to reproduce some of the results on **Manywell** and **VAE** with PIS and GFlowNet models as an example, showing the hyperparameters:

PIS:

```
--mode_fwd pis  --lr_policy 1e-3
```

PIS + Langevin:

```
--mode_fwd pis  --lr_policy 1e-3 --langevin
```

GFlowNet TB:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd tb --lr_policy 1e-3 --lr_flow 1e-1
```

GFlowNet TB + Expl.:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd tb --lr_policy 1e-3 --lr_flow 1e-1
--exploratory --exploration_wd --exploration_factor 0.2
```

GFlowNet VarGrad + Expl.:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd tb-avg --lr_policy 1e-3 --lr_flow 1e-1
--exploratory --exploration_wd --exploration_factor 0.2
```

GFlowNet FL-SubTB:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd subtb --lr_policy 1e-3 --lr_flow 1e-2
--partial_energy --conditional_flow_model
```

GFlowNet FL-SubTB + LP:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd subtb --lr_policy 1e-3 --lr_flow 1e-2
--partial_energy --conditional_flow_model
--langevin --epochs 10000
```

GFlowNet TB + Expl. + LS:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd tb --lr_policy 1e-3 --lr_back 1e-3 --lr_flow 1e-1
--exploratory --exploration_wd --exploration_factor 0.1
--both_ways --local_search
--buffer_size 600000 --prioritized rank --rank_weight 0.01
--ld_step 0.1 --ld_schedule --target_acceptance_rate 0.574
```

GFlowNet TB + Expl. + LP:

```
python train.py
--t_scale 1. --energy many_well --pis_architectures --zero_init --clipping
--mode_fwd tb --lr_policy 1e-3 --lr_flow 1e-1
--exploratory --exploration_wd --exploration_factor 0.2
--langevin --epochs 10000
```

GFlowNet TB + Expl. + LS (VAE):

```
python train.py
--energy vae --pis_architectures --zero_init --clipping
--mode_fwd cond-tb-avg --mode_bwd cond-tb-avg --repeats 5
--lr_policy 1e-3 --lr_flow 1e-1 --lr_back 1e-3
--exploratory --exploration_wd --exploration_factor 0.1
--both_ways --local_search
--max_iter_ls 500 --burn_in 200
--buffer_size 90000 --prioritized rank --rank_weight 0.01
--ld_step 0.001 --ld_schedule --target_acceptance_rate 0.574
```

GFlowNet TB + Expl. + LP + LS (VAE):

```
python train.py
--energy vae --pis_architectures --zero_init --clipping
--mode_fwd cond-tb-avg --mode_bwd cond-tb-avg --repeats 5
--lr_policy 1e-3 --lr_flow 1e-1
--lgv_clip 1e2 --gfn_clip 1e4 --epochs 10000
--exploratory --exploration_wd --exploration_factor 0.1
--both_ways --local_search
--lr_back 1e-3 --max_iter_ls 500 --burn_in 200
--buffer_size 90000 --prioritized rank  --rank_weight 0.01
--langevin
--ld_step 0.001 --ld_schedule --target_acceptance_rate 0.574
```

B Target densities

Gaussian Mixture Model with 25 modes (25GMM). The model, termed as 25GMM, consists of a two-dimensional Gaussian mixture model with 25 distinct modes. Each mode exhibits an identical variance of 0.3. The centers of these modes are strategically positioned on a grid formed by the Cartesian product $\{-10, -5, 0, 5, 10\} \times \{-10, -5, 0, 5, 10\}$, effectively distributing them across the coordinate space.

Funnel [28]. The funnel represents a classical benchmark in sampling techniques, characterized by a ten-dimensional distribution defined as follows: The first dimension, x_0, follows a normal distribution with mean 0 and variance 9, denoted as $x_0 \sim \mathcal{N}(0, 9)$. Conditional on x_0, the remaining dimensions, $x_{1:9}$, are distributed according to a multivariate normal distribution with mean vector $\mathbf{0}$ and a covariance matrix $\exp(x_0)\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. This is succinctly represented as $x_{1:9} \mid x_0 \sim \mathcal{N}(\mathbf{0}, \exp(x_0)\,\mathbf{I})$.

Manywell [51]. The manywell is characterized by a 32-dimensional distribution, which is constructed as the product of 16 identical two-dimensional double well distributions. Each of these two-dimensional components is defined by a potential function, $\mu(x_1, x_2)$, expressed as $\mu(x_1, x_2) = \exp\left(-x_1^4 + 6x_1^2 + 0.5x_1 - 0.5x_2^2\right)$.

VAE [40]. This task involves sampling from a 20-dimensional latent posterior $p(z|x) \propto p(z)p(x|z)$, where $p(z)$ is a fixed prior and $p(x|z)$ is a pretrained VAE decoder, using a conditional sampler $q(z|x)$ dependent on input data (image) x.

LGCP [48]. This density over a 1600-dimensional variable is a Log-Gaussian Cox process fit to a distribution of pine saplings in Finland.

B.1 Discrepancies in past work

Wrong definitions of the Funnel density. As already noted by [77], [87] uses a different variance of the first component in the Funnel density, 1 instead of 9. This apparent bug in the task definition has been propagated to subsequent work, including [41].

Evaluation on LGCP. The LGCP benchmark suffers from the lack of a consistent ground truth $\log Z$ to compare against. Previous work has compared the value of the partition function $\log Z$ against a "long run of Sequential Monte Carlo" [87]. We note that this approach produces noisy estimates of the partition function, especially in high-dimensional problems (indeed, SMC has rarely been used in problems with over a thousand dimensions); therefore, it is unclear how long the SMC needs to be run to produce an accurate estimate. We found that two *different* values are being used in the literature: $\log Z = 512.6$ in one repository and $\log Z = 501.8$ in another.

On FL-SubTB as used in [85]. We make two observations calling into question the main results of [85].

First, the only substantial difference between the algorithm used by [85] and the one from the past work [41] – which first proposed the use of GFlowNet objectives to train diffusion samplers – is the substitution of the FL-SubTB objective [54, 43] for TB [44]. However, [85] elects to compare FL-SubTB *with* the Langevin parameterization to TB *without* the Langevin parameterization. Our results in Table 1 show that while the Langevin parameterization is crucial for the performance of all objectives; FL-SubTB does not provide any consistent benefit over TB or VarGrad.

Second, the results are not reproducible, neither with the published code from [85] run 'out of the box', nor with our reimplementation. In particular, on the LGCP density, the training did not converge within the allotted training time. We have contacted the authors of [85], who confirmed that running their published code does not reproduce the results in the paper but could not provide any further explanation or a working implementation.

C Additional results

C.1 Expanded unconditional sampling results

Table C.1 is an expanded version of Table 1, showing Wasserstein distances between sets of K samples from the true distribution and generated by a trained sampler. (Note that ground truth for LGCP is not available.)

Table C.1: Log-partition function estimation errors and 2-Wasserstein distances for unconditional modeling tasks (mean and standard deviation over 5 runs). The four groups of models are: MCMC-based samplers, simulation-driven variational methods, baseline GFlowNet methods with different learning objectives, and methods augmented with Langevin parametrization and local search.

Energy →	25GMM ($d = 2$)			Funnel ($d = 10$)			Manywell ($d = 32$)		
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$
SMC	0.569±0.010		0.86±0.10	0.561±0.801		50.3±18.9	14.99±1.078		8.28±0.32
GGNS [42]	0.016±0.042		1.19±0.17	0.033±0.173		25.6±4.75	0.292±0.454		6.51±0.32
DIS [8]	1.125±0.056	0.986±0.011	4.71±0.06	0.839±0.169	0.093±0.038	20.7±2.1	10.52±1.02	3.05±0.46	5.98±0.46
DDS [77]	1.760±0.08	0.746±0.389	7.18±0.044	0.424±0.049	0.206±0.033	29.3±9.5	7.36±2.43	0.23±0.05	5.71±0.16
PIS [87]	1.769±0.104	1.274±0.218	6.37±0.65	0.534±0.008	0.262±0.008	22.0±4.0	3.85±0.03	2.69±0.04	6.15±0.02
+ LP [87]	1.799±0.051	0.225±0.583	7.16±0.11	0.587±0.012	0.285±0.044	22.1±4.0	13.19±0.82	0.07±0.85	6.55±0.34
TB [41]	1.176±0.109	1.071±0.112	4.83±0.45	0.690±0.018	0.239±0.192	22.4±4.0	4.01±0.04	2.67±0.02	6.14±0.02
TB + Expl. [41]	0.560±0.302	0.422±0.320	3.61±1.41	0.749±0.015	0.226±0.138	21.3±4.0	4.01±0.05	2.68±0.06	6.15±0.02
VarGrad + Expl.	0.615±0.241	0.487±0.250	3.89±0.85	0.642±0.010	0.250±0.112	22.1±4.0	4.01±0.05	2.69±0.06	6.15±0.02
FL-SubTB	1.127±0.010	1.020±0.010	4.64±0.09	0.527±0.011	0.182±0.142	22.1±4.0	3.98±0.07	2.72±0.05	6.15 ±0.01
+ LP [85]	0.209±0.025	0.011±0.024	1.45±0.29	0.563±0.021	0.155±0.317	22.2±4.0	4.23±0.12	2.66±0.22	6.10±0.02
TB + Expl. + LS (*ours*)	0.171±0.013	0.004±0.011	1.25±0.18	0.653±0.025	0.285±0.099	21.9±4.0	4.57±2.13	0.19±0.29	5.66±0.05
TB + Expl. + LP (*ours*)	0.206±0.018	0.011±0.010	1.29±0.07	0.666±0.615	0.051±0.616	22.3±3.9	7.46±1.74	1.06±1.11	5.73±0.31
TB + Expl. + LP + LS (*ours*)	0.190±0.013	0.007±0.011	1.31±0.07	0.768±0.052	0.264±0.063	21.8±3.9	4.68±0.49	0.07±0.17	5.33±0.03
VarGrad + Expl. + LP + LS (*ours*)	0.207±0.016	0.015±0.015	1.13±0.13	0.920±0.118	0.256±0.037	21.2±4.0	4.11±0.45	0.02±0.21	5.30±0.02

Highlight : mean indistinguishable from minimum in column with $p < 0.05$ under one-sided Welch unpaired t-test.

Table C.2: Log-partition function estimation errors and empirical 2-Wasserstein distances on the 32-dimensional Manywell with Brownian and variance-preserving noising processes.

Backward process →	Brownian			VP		
Objective ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$
TB + Expl. + LP	7.46±1.74	1.06±1.11	5.73±0.31	7.55±2.85	1.49±1.30	5.68±0.42
TB + Expl. + LP + LS	4.68±0.49	0.07±0.17	5.33±0.03	4.52±0.21	1.23±0.07	5.75±0.01
VarGrad + Expl.	4.01±0.05	2.69±0.06	6.15±0.02	4.04±0.05	2.65±0.08	6.17±0.02

C.2 Variance-preserving noising process

Following the recent results by [8, 62, 77], we perform an additional set of experiments with a different successful noise schedule. We replace the Brownian motion by the *variance-preserving SDEs* from Song et al. [71], given by an *Ornstein-Uhlenbeck process:*

$$\sigma(t) := \nu\sqrt{2\beta(t)}\mathbf{I} \quad \text{and} \quad \mu(x,t) := -\beta(t)x \tag{16}$$

with $\nu \in (0, \infty)$.

In particular, we follow the common procedure - use $\nu := 1$ and

$$\beta(t) := (1-t)\beta_{min} + t\beta_{max}, \quad t \in [0,1],$$

with $\beta_{min} = 0.01$ and $\beta_{max} = 4.0$.

We evaluate three representative methods using this variance-preserving backward process. The results, in Table C.2, are similar to those using the Brownian bridge process. We expect that the choice of noising process gains importance in challenging high-dimensional problems.

C.3 Scalability study

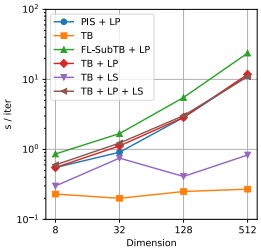
The Manywell energy (§B) is defined in any even number of dimensions and thus allows to study the scaling of the methods with dimension. We evaluate several representative methods in dimension 8, 128, and 512 (in addition to the 32 studied in the main text). All experimental settings are kept the same as as for $d = 32$. Due to the large runtime, some runs in dimensions 128 and 512 had to be limited at 12 hours, while in dimensions 8 and 32 all run in under 3 hours on a RTX8000 GPU.

These results are shown in Table C.3. We observe:

- The overhead of the Langevin parametrization grows with dimension, but is critical to performance.
- The even higher overhead of FL-SubTB as used by [85].
- The relatively high efficiency and low overhead of our newly proposed local search.

Table C.3: Scaling with dimension on Manywell: log-partition function estimation errors and time per training iteration on a RTX8000 GPU.

Dimension →	$d = 8$		$d = 32$		$d = 128$		$d = 512$	
Objective ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$
PIS + LP [87]	0.86	0.14	13.19	0.07	58.0	23.7	251	169
TB [41]	0.95	0.70	4.01	2.68	205.6	119.8	1223	957
FL-SubTB + LP [85]	0.57	0.67	4.23	2.66	48.9	21.7	198	107
TB + LP	0.25	0.04	7.46	1.06	46.4	14.0	259	169
TB + LS	0.44	0.00	4.57	0.19	458.7	139.3	1626	1077
TB + LP + LS	0.25	0.02	4.68	0.07	66.6	14.9	326	209



D Experiment details

Sampling energies. In this section, we detail the hyperparameters used for our experiments. An important parameter is the diffusion coefficient of the forward policy, which is denoted by σ and also used in the definition of the fixed backward process. The base diffusion rate σ^2 (parameter `t_scale`) is set to 5 for **25GMM** and 1 for **Funnel** and **Manywell**, consistent with past work.

For **LGCP**, we found that using too small diffusion rate σ^2 (*e.g.*, $\sigma^2 = 1$) prevents the methods from achieving reasonable results. We tested different values of $\sigma^2 = \{1, 3, 5\}$, and selected $\sigma^2 = 5$, which gives the best results, which follows the findings in Zhang & Chen [87].

For all our experiments, we used a learning rate of 10^{-3}. Additionally, we used a higher learning rate for learning the flow parameterization, which is set as 10^{-1} when using the TB loss and 10^{-2} with the SubTB loss. These settings were found to be consistently stable (unlike those with higher learning rates) and converge within the allotted number of steps (unlike those with lower learning rates).

For the SubTB loss, we experimented with the settings of 10× lower learning rates for both flow and policy models communicated by the authors of [85], but found the results to be inferior both using their published code (and other unstated hyperparameters communicated by the authors) and using our reimplementation.

For models with exploration, we use an exploration factor of 0.2 (that is, noise with a variance of 0.2 is added to the policy when sampling trajectories for training), which decays linearly over the first half of training, consistent with [41].

We train all our models for 25,000 iterations except those using Langevin dynamics, which are trained for 10,000 iterations. This results in approximately equal computation time owing to the overhead from computation of the score at each sampling step.

We use the same neural network architecture for the GFlowNet as one of our baselines [87]. Similar to [87], we also use an initialization scheme with last-layer weights set to 0 at the start of training. Since the SubTB requires the flow function to be conditioned on the current state $\mathbf{x}_t$ and time t, we follow [85] and parametrize the flow model with the same architecture as the Langevin scaling model NN_2 in [87]. Additionally, we perform clipping on the output of the network as well as the score obtained from the energy function, typically setting the clipping parameter of Langevin scaling model to 10^2 and policy network to 10^4, similarly to [77]:

$$f_\theta(k, x) = \text{clip}\Big(\text{NN}_1(k, x; \theta) + \text{NN}_2(k; \theta) \odot \text{clip}(\nabla \ln \pi(x), -10^2, 10^2), -10^4, 10^4\Big). \tag{17}$$

All models were trained with a batch size of 300. In each experiment, we train models on a single NVIDIA A100-Large GPU, if not stated explicitly otherwise.

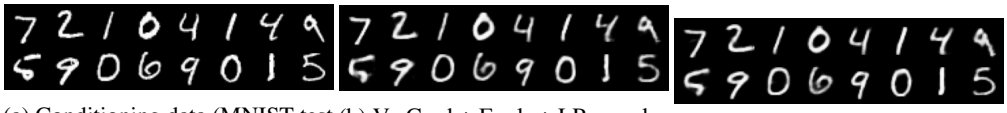
VAE experiment. In the VAE experiment, we used a standard VAE model pretrained for 100 epochs on the MNIST dataset. The encoder $q(z|x)$ contains an input linear layer (784 neurons) followed by hidden linear layer (400 neurons), ReLU activation function, and two linear heads (20 neurons each) whose outputs were reparametrized to be means and scales of multivariate Normal distribution. The decoder consists of 20-dimensional input, one hidden layer (400 neurons), followed by the ReLU activation, and 784-dimensional output. The output is processed by the sigmoid function to be scaled properly into $[0, 1]$.

The goal is to sample conditionally on x the latent z from the unnormalized density $p(z, x) = p(z)p(x \mid z)$ (where $p(z)$ is the prior and $p(x|z)$ is the likelihood computed from the decoder), which

is proportional to the posterior $p(z \mid x)$. We reuse the model architectures from the unconditional sampling experiments, but also provide x as an input to the first layer of the models expressing the policy drift (as well as the flow, for FL-SubTB) and add one hidden layer to process high-dimensional conditions. For models trained with TB, $\log Z_\theta$ also becomes a MLP taking x as input.

The VarGrad and LS techniques require adaptations in the conditional setting. For LS, buffers ($\mathcal{D}_{\text{buffer}}$ and $\mathcal{D}_{\text{LS}}$) must store the associated conditions x together with the samples z and the corresponding unnormalized density $R(z; x)$, *i.e.*, a tuple of $(x, z, R(z; x))$. For VarGrad, because the partition function depends on the conditioning information x, it is necessary to compute variance over many trajectories sharing the same condition. We choose to sample 10 trajectories for each condition occurring in a minibatch and compute the VarGrad loss for each such set of 10 trajectories.

The VAE model was trained on the entire MNIST training set and never updated on the test part of MNIST. In order to evaluate samplers (with respect to the variational lower bound) on a unique set of examples, we chose the first 100 elements of MNIST test data. All of the samplers were trained having access to the MNIST training data and the frozen VAE decoder. For a fair comparison, samplers utilizing the LP were trained for $10,000$, whereas the remaining for $25,000$ iterations. In each iteration, a batch of 300 examples from MNIST was given as conditions. In each experiment, we train models on a single NVIDIA A100-Large GPU, if not stated explicitly otherwise.



(a) Conditioning data (MNIST test set)

(b) VarGrad + Expl. + LP samples decoded

(c) VAE reconstruction

Figure D.1: Our sampler (VarGrad + Expl. + LP) is conditioned by a subset of never-seen data coming from the ground truth distribution (**left**). The conditional samples were then decoded by the the fixed VAE (**middle**). For the comparison, we show the reconstruction of the real data by VAE (**right**). We observed that the decoded samples are visually very similar to the reconstructions making these two pictures almost indistinguishable. Both, decoded samples and reconstruction, are more blurry than the ground truth data, which is caused by a limited capacity of the VAE's latent space.

E Local search-guided GFlowNet

Prioritized sampling scheme. We can use uniform or prioritized sampling to draw samples from the buffer for training. We found prioritized sampling to work slightly better in our experiments (see ablation study in §E.2), although the choice should be investigated more thoroughly in future work.

We use rank-based prioritization [73], which follows a probabilistic approach defined as:

$$p(\mathbf{x}; \mathcal{D}_{\text{buffer}}) \propto \left(k|\mathcal{D}_{\text{buffer}}| + \text{rank}_{\mathcal{D}_{\text{buffer}}}(\mathbf{x})\right)^{-1}, \tag{18}$$

where $\text{rank}_{\mathcal{D}_{\text{buffer}}}(\mathbf{x})$ represents the relative rank of a sample x based on a ranking function $R(\mathbf{x})$ (in our case, the unnormalized target density at sample $\mathbf{x}$). The parameter k is a hyperparameter for prioritization, where a lower value of k assigns a higher probability to samples with higher ranks, thereby introducing a more greedy selection approach. We set $k = 0.01$ for every task. Given that the sampling is proportional to the size of $\mathcal{D}_{\text{buffer}}$, we impose a constraint on the maximum size of the buffer: $|\mathcal{D}_{\text{buffer}}| = 600,000$ with first-in first out (FIFO) data structure for every task, except we use $|\mathcal{D}_{\text{buffer}}| = 90,000$ for VAE task. See the algorithm below for a detailed pseudocode.

Algorithm 1 GFlowNet Training with Local search

Initialize policy parameters θ for P_F, and empty buffers $\mathcal{D}_{\text{buffer}}, \mathcal{D}_{\text{LS}}$
for $i = 1, 2, \ldots, I$ **do**
  if $i\%2 == 0$ **then**
    Sample M trajectories $\{\tau_1, \ldots, \tau_M\} \sim P_F(\cdot|\epsilon\text{-greedy})$
    Update $\mathcal{D}_{\text{buffer}} \leftarrow \mathcal{D}_{\text{buffer}} \cup \{x|\tau \rightarrow x\}$
    Minimize $L(\tau; \theta)$ using $\{\tau_1, \ldots, \tau_M\}$ to update P_F
  else
    if $i\%100 == 0$ **then**
      Sample $\{x_1, \ldots, x_M\} \sim \mathcal{D}_{\text{buffer}}$
      $\mathcal{D}_{\text{LS}} \leftarrow$ Local Search$(\{x_1, \ldots, x_M\}; \mathcal{D}_{\text{LS}})$
    end if
    Sample $\{x'_1, \ldots, x'_M\} \sim p_{\text{buffer}}(\cdots; \mathcal{D}_{\text{LS}})$
    Sample $\{\tau'_1, \ldots, \tau'_M\} \sim P_B(\cdots|x')$
    Minimize $L(\tau'; \theta)$ using $\{\tau'_1, \ldots, \tau'_M\}$ to update P_F
  end if
end for

We use the number of total iterations $I = 25,000$ for every task as default. Note as local search is performed to update $\mathcal{D}_{\text{LS}}$ occasionally that per 100 iterations, the number of local search updates is done $25,000/100 = 250$.

E.1 Local search algorithm

This section describes a detailed algorithm for local search, which provides an updated buffer $\mathcal{D}_{\text{LS}}$, which contains low-energy samples.

Dynamic adjustment of step size η. To enhance local search using parallel MALA, we dynamically select the Langevin step size (η), which governs the MH acceptance rate. Our objective is to attain an average acceptance rate of 0.574, which is theoretically optimal for high-dimensional MALA's efficiency [55]. While the user can customize the target acceptance rate, the adaptive approach eliminates the need for manual tuning.

Computational cost of local search. The computational cost of local search is not significant. Local search for iteration of $K = 200$ requires 6.04 seconds (averaged with five trials in Manywell), where we only occasionally (every 100 iterations) update $\mathcal{D}_{\text{LS}}$ with MALA. The speed is evaluated using the computational resources of the Intel Xeon Scalable Gold 6338 CPU (2.00GHz) and the NVIDIA RTX 4090 GPU.

Algorithm 2 Local search (Parallel MALA)

input Initial states $\{x_1^{(0)}, \ldots, x_M^{(0)}\}$, current buffer $\mathcal{D}_{\text{LS}}$, total steps K, burn in steps $K_{\text{burn-in}}$, initial step size η_0, amplifying factor f_{increase}, damping factor f_{decrease}, unnormalized target density R
output Updated buffer $\mathcal{D}_{\text{LS}}$
  Initialize acceptance counter $a = 0$
  Set $\eta \leftarrow \eta_0$
  for $k = 1 : K$ **do**
    Initialize step acceptance count $a_k = 0$
    for $m = 1 : M$ **do**
      Sample $\sigma \sim \mathcal{N}(0, I)$
      Propose $x_m^* \leftarrow x_m^{(k-1)} + \eta \nabla \log R(x_m^{(k-1)}) + \sqrt{2\eta}\sigma$
      Compute acceptance ratio $r \leftarrow \min\left(1, \frac{R(x_m^*)\exp\left(-\frac{1}{4\eta}\|x_m^{(k-1)} - x_m^* - \eta\nabla \log R(x_m^*)\|^2\right)}{R(x_m^{(k-1)})\exp\left(-\frac{1}{4\eta}\|x_m^* - x_m^{(k-1)} - \eta\nabla \log R(x_m^{(k-1)})\|^2\right)}\right)$
      With probability r, accept the proposal: $x_m^{(k)} \leftarrow x_m^*$ and increment $a_k \leftarrow a_k + 1$
      if $k > K_{\text{burn-in}}$ **then**
        Update buffer: $\mathcal{D}_{\text{LS}} \leftarrow \mathcal{D}_{\text{LS}} \cup \{x_m^*\}$
      end if
    end for
    Compute step acceptance rate $\alpha_k = a_k / M$
    if $\alpha_k > \alpha_{\text{target}}$ **then**
      $\eta \leftarrow \eta \times f_{\text{increase}}$
    else if $\alpha_k < \alpha_{\text{target}}$ **then**
      $\eta \leftarrow \eta \times f_{\text{decrease}}$
    end if
  end for

We adopt default parameters: $f_{\text{increase}} = 1.1$, $f_{\text{decrease}} = 0.9$, $\eta_0 = 0.01$, $K = 200$, $K_{\text{burn-in}} = 100$, and $\alpha_{\text{target}} = 0.574$ for three unconditional tasks. For conditional tasks of VAE, we give more iterations of local search: $K = 500$, $K_{\text{burn-in}} = 200$.

It is noteworthy that by adjusting the inverse temperature β into R^β during the computation of the Metropolis-Hastings acceptance ratio r, we can facilitate a greedier local search strategy aimed at exploring samples with lower energy (*i.e.*, higher density p_{target}). This approach proves advantageous for navigating high-dimensional and steep landscapes, which are typically challenging for locating low-energy samples. For unconditional tasks, we set $\beta = 1$.

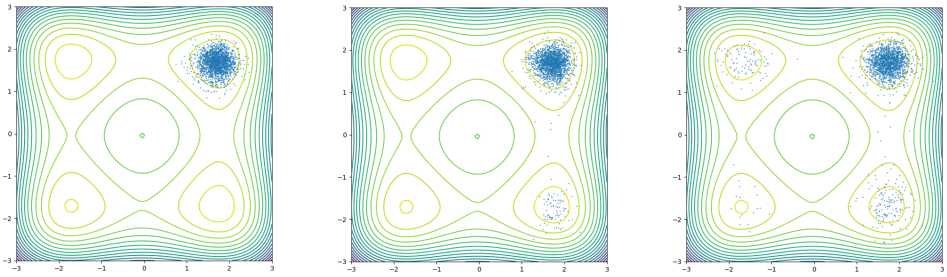
In the context of the VAE task (Table 2), we utilize two GFlowNet loss functions: TB and VarGrad. For local search within TB, we set $\beta = 1$, while for VarGrad, we employ $\beta = 5$. As illustrated in Table 2, employing a local search with $\beta = 1$ fails to enhance the performance of the TB method. Conversely, a local search with $\beta = 5$ results in improvements at the log $\hat{Z}^{\text{RW}}$ metric over the VarGrad + Expl. + LP, even though the performance of VarGrad + Expl. + LP surpasses that of TB substantially. This underscores the importance of selecting an appropriate β value, which is critical for optimizing the exploration-exploitation balance depending on the target objectives.

E.2 Ablation study for local search-guided GFlowNets

Increasing capacity of buffer. The capacity of the replay buffer influences the duration for which it retains past experiences, enabling it to replay these experiences to the policy. This mechanism helps in preventing mode collapse during training. Table E.1 demonstrates that enhancing the buffer's capacity leads to improved sampling quality. Furthermore, Figure 1 illustrates that increasing the buffer's capacity—thereby encouraging the model to recall past low-energy experiences—enhances its mode-seeking capability.

Table E.1: Comparison of the sampling quality of each sampler trained with varying replay buffer capacities in Manywell. Five independent runs have been conducted, with both the mean and standard deviation reported.

Buffer Capacity ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\text{RW}}$	$\mathcal{W}_2^2$
30,000	4.41±0.10	2.73±0.15	6.17±0.02
60,000	4.06±0.05	2.38±0.38	6.14±0.04
600,000	4.57±2.13	0.19±0.29	5.66±0.05



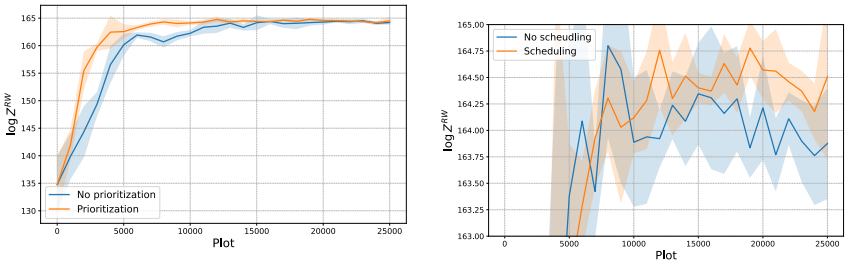
(a) Capacity 30,000 (b) Capacity 60,000 (c) Capacity 600,000

Figure E.1: Illustration of each sampler trained with varying capacities of replay buffers, depicting 2,000 samples. As the capacity of the buffer increases, the number of modes captured by the sampler also increases.

Benefit of prioritization.

Rank-prioritized sampling gives faster convergence compared with no prioritization (uniform sampling), as shown in Fig. E.2a.

Dynamic adjustment of η vs. fixed η = 0.01. As shown in Fig. E.2b, dynamic adjustment to target acceptance rate $\alpha_{\text{target}} = 0.574$ gives better performances than fixed Langevin step size of η showcasing the effectiveness of the dynamic adjustment.



(a) Benefit of prioritization in replay buffer sampling..

(b) Benefit of scheduling η dynamically.

Figure E.2: Ablation study for prioritized replay buffer and step size η scheduling of local search. Mean and standard deviation are plotted based on five independent runs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See theory and experiment sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, see section 5.3 and conclusion, as well as references to appendix material where relevant.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No new theoretical results. For exposition of the mathematical basis for our algorithms, we state the assumptions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See experiment sections and references to appendix material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide code to reproduce nearly all of our experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See the experiment sections and references to appendix material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results tables and plots show standard deviation and indicate significance of the best metric.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See experiment sections and references to appendix material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We believe there are no violations of the CoE.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper studies a ML problem with no immediate societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper studies a ML problem with no immediate application to generation of new image or text content, nor other functions that have the potential for misuse, to the best of our knowledge.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the works introducing all datasets we study.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human studies.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human studies.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

FROM DISCRETE-TIME POLICIES TO CONTINUOUS-TIME DIFFUSION SAMPLERS: ASYMPTOTIC EQUIVALENCES AND FASTER TRAINING

Julius Berner[*]
California Institute of Technology

Lorenz Richter[*]
Zuse Institute Berlin
dida Datenschmiede GmbH

Marcin Sendera[*]
Jagiellonian University
Mila, Université de Montréal

Jarrid Rector-Brooks
Mila, Université de Montréal
Dreamfold

Nikolay Malkin
University of Edinburgh

ABSTRACT

We study the problem of training neural stochastic differential equations, or diffusion models, to sample from a Boltzmann distribution without access to target samples. Existing methods for training such models enforce time-reversal of the generative and noising processes, using either differentiable simulation or off-policy reinforcement learning (RL). We prove equivalences between families of objectives in the limit of infinitesimal discretization steps, linking entropic RL methods (GFlowNets) with continuous-time objects (partial differential equations and path space measures). We further show that an appropriate choice of coarse time discretization during training allows greatly improved sample efficiency and the use of time-local objectives, achieving competitive performance on standard sampling benchmarks with reduced computational cost.

1 INTRODUCTION

We consider the problem of sampling from a distribution on $\mathbb{R}^d$ with density p_{target}, which is described by an unnormalized energy model $p_{\text{target}}(x) = \exp(-\mathcal{E}(x))/Z$ with $Z = \int_{\mathbb{R}^d} \exp(-\mathcal{E}(x))\,\mathrm{d}x$. We have access to $\mathcal{E}$ but not to the normalizing constant Z or to samples from p_{target}. This problem is ubiquitous in Bayesian statistics and machine learning and has been an object of study for decades, with Monte Carlo methods (Duane et al., 1987; Roberts & Tweedie, 1996; Hoffman et al., 2014; Leimkuhler et al., 2014; Lemos et al., 2023) recently being complemented by deep generative models (Albergo et al., 2019; Noé et al., 2019; Gabrié et al., 2021; Midgley et al., 2023; Akhound-Sadegh et al., 2024).

Building upon the success of diffusion models in data-driven generative modeling (Sohl-Dickstein et al., 2015; Ho et al., 2020; Dhariwal & Nichol, 2021; Rombach et al., 2021, *inter alia*), recent work (*e.g.*, Zhang & Chen, 2022; Berner et al., 2022; Vargas et al., 2023; Richter & Berner, 2024; Vargas et al., 2024; Sendera et al., 2024) has proposed solutions to this problem that model generation as the reverse of a diffusion (noising) process in discrete or continuous time (Fig. 1). Thus p_{target} is modeled by gradually transporting samples, by a sequence of stochastic transitions, from a simple prior distribution p_{prior} (*e.g.*, a Gaussian) to the target distribution. When a dataset of samples from p_{target} is given, diffusion models are trained using a score matching objective equivalent to a variational bound on data log-likelihood (Song et al., 2021a). The problem is more challenging

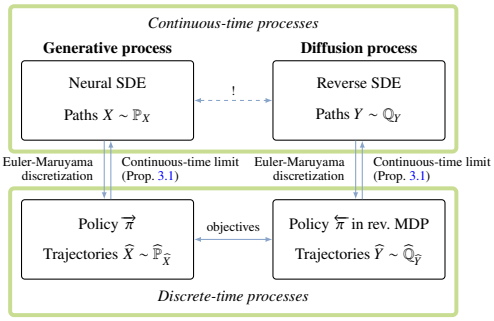


Figure 1: The problem of making continuous-time forward and reverse processes determine the same path space measure is approximated by matching distributions over discrete-time trajectories.

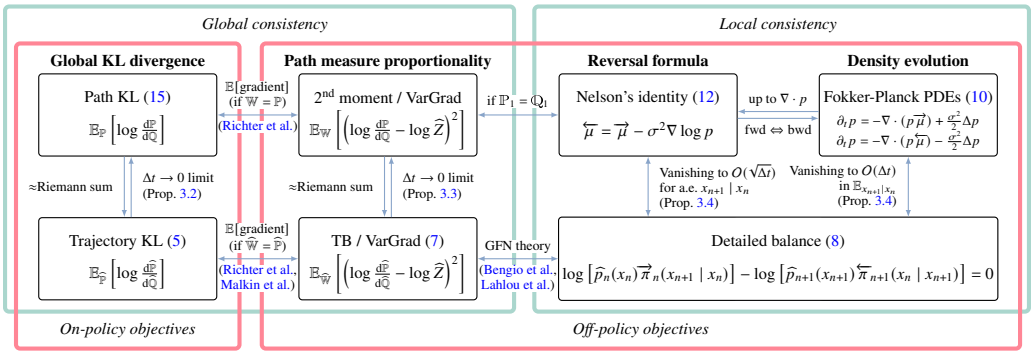


Figure 2: Training objectives for neural SDEs (top row) and their approximations by objectives for discrete-time policies (bottom row). On-policy objectives minimize a divergence by differentiating through SDE integration, while off-policy objectives enforce local or global consistency constraints. Our results explain the behavior of discrete-time objectives as the time discretization becomes finer.

when we have no samples but can only query the energy function, as training methods necessarily involve simulation of the generative process. (We survey additional related work in Appendix A.)

In continuous time, we assume the generative process takes the form of a stochastic differential equation (SDE) (with initial condition p_{prior} and diffusion coefficient σ):

$$\mathrm{d}X_t = \overrightarrow{\mu}(X_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \quad X_0 \sim p_{\text{prior}}. \tag{1}$$

When the drift μ is given by a parametric model, such as a neural network, (1) is called a *neural SDE* (Tzen & Raginsky, 2019; Kidger et al., 2021a; Song et al., 2021b). The goal is to fit the parameters so as to make the distribution of X_1 induced by the initial conditions and the SDE (1) close to p_{target}.

In discrete time, we assume the generative process is described by a Markov chain with transition kernels $\overrightarrow{\pi}_n(\widehat{X}_{n+1} \mid \widehat{X}_n)$, $n = 0, \ldots, N-1$, and initial distribution $\widehat{X}_0 \sim p_{\text{prior}}$. The goal is to learn the transition probabilities $\overrightarrow{\pi}_n$ so as to make the distribution of $\widehat{X}_N$ close to p_{target}. This is the setting of stochastic normalizing flows (Hagemann et al., 2023), which are, in turn, a special case of (continuous) generative flow networks (GFlowNets; Bengio et al., 2021; Lahlou et al., 2023).

Training objectives for both the continuous-time and discrete-time processes are typically based on minimization of a bound on the divergence between the distributions over trajectories induced by the generative process and by the target distribution together with the noising process. These objectives may rely on differentiable simulation of the generative process (Li et al., 2020; Kidger et al., 2021b; Zhang & Chen, 2022) or on off-policy reinforcement learning (RL), which optimizes objectives depending on trajectories obtained through exploration (Nüsken & Richter, 2021; Malkin et al., 2023). Objectives may further be classified as global (involving the entire trajectory) or local (involving a single transition). Common objectives and the relationships among them are summarized in Fig. 2.

Any SDE determines a discrete-time policy when using a time discretization, such as the Euler-Maruyama integration scheme; conversely, in the limit of infinitesimal time steps, the discrete-time policy obtained in this way approaches the continuous-time process (Kloeden & Platen, 1992). The question we study in this paper is how the training objectives for continuous-time and discrete-time processes are related in the limit of infinitesimal time steps. We formally connect RL methods to stochastic control and dynamic measure transport with the following theoretical contributions:

(1) We show that global objectives in discrete time converge to objectives that minimize divergences between path space measures induced by the forward and reverse processes in continuous time (Prop. 3.3).

(2) We show that local constraints enforced by GFlowNet training objectives asymptotically approach partial differential equations that govern the time evolution of the marginal densities of the SDE under the generative and noising processes (Prop. 3.4).

These results motivate the hypothesis that an appropriate choice of time discretization during training can allow for greatly improved sample efficiency. Training with shorter trajectories obtained by

coarse time discretizations would further allow the use of time-local objectives without the computationally expensive bootstrapping techniques that are necessary when training with long trajectories. Confirming this hypothesis, we make the following empirical contribution:

(3) In experiments on standard sampling benchmarks, we show that training with *nonuniform* time discretizations much coarser than those used for inference achieves similar performance to state-of-the-art methods, at a fraction of the computational cost (Fig. 5).

2 Dynamic measure transport in discrete and continuous time

Recall that our goal is to sample from a target distribution $p_{\text{target}} = \frac{1}{Z}\exp(-\mathcal{E}(x))$ given by a continuous energy function $\mathcal{E}\colon \mathbb{R}^d \to \mathbb{R}$. To achieve this goal, we present approaches using discrete-time policies in the framework of Markov decision processes (MDPs) in §2.1 and continuous-time processes in the context of neural SDEs in §2.2. In particular, we will draw similarities between the two approaches and show how time discretizations of neural SDEs give rise to specific policies in MDPs in §2.3. This allows us to rigorously analyze the asymptotic behavior of corresponding distributions and divergences in §3. Note that our general assumptions can be found in Appendix B.1.

Our exposition synthesizes the definitions for MDP policies (Bengio et al., 2023; Lahlou et al., 2023), results on neural SDEs for sampling (Richter & Berner, 2024; Vargas et al., 2024), and PDE perspectives (Máté & Fleuret, 2023; Sun et al., 2024). The results in §3 extend classical results on SDE approximations (see, *e.g.*, Kloeden & Platen (1992)) to objectives for diffusion-based samplers.

2.1 Discrete-time setting: Stochastic control policies

A discrete-time Markovian process $\widehat{X}$ with density $\widehat{\mathbb{P}}(\widehat{X})$ – a distribution over $\mathbb{R}^d$-valued variables $\widehat{X}_0, \dots, \widehat{X}_N$ – can be identified with a policy $\overrightarrow{\pi}$ in the deterministic Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, T, R)$ depicted in Fig. 3, given by

$$\begin{aligned}\overrightarrow{\pi}(a \mid \bullet) &= \widehat{\mathbb{P}}(\widehat{X}_0 = a) = p_{\text{prior}}(a),\\ \overrightarrow{\pi}_n(a \mid (x, t_n)) &= \widehat{\mathbb{P}}(\widehat{X}_{n+1} = a \mid \widehat{X}_n = x). \quad (2)\end{aligned}$$

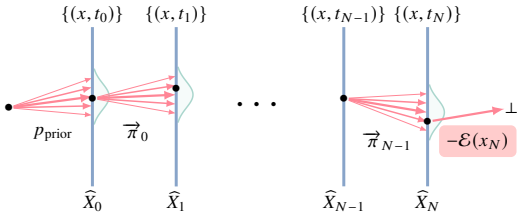


Figure 3: The MDP and policy representing the process $\widehat{\mathbb{P}}$, a distribution over $\widehat{X} = (\widehat{X}_0, \dots, \widehat{X}_N)$.

We sometimes write $\overrightarrow{\pi}_n(\cdot \mid x)$ for $\overrightarrow{\pi}_n(\cdot \mid (x, t_n))$ for convenience. We relegate formal definitions to Appendix B.2; in short, the states are pairs of space and time coordinates (x, t_n) (together with abstract initial and terminal states), actions represent steps from $\widehat{X}_n$ to $\widehat{X}_{n+1}$ (taking action a leads to state (a, t_{n+1})), and the reward for terminating from a state (x, t_N) is set to $-\mathcal{E}(x)$. The learning problem is to find $\overrightarrow{\pi}$ whose induced distribution over $\widehat{X}_N$ is the Boltzmann distribution of the reward.

This learning problem differs from that of standard reinforcement learning, where one wishes to *maximize* the expected reward with respect to a policy in the MDP. However, it is close to the setting of maximum-entropy reinforcement learning (MaxEnt RL; Ziebart, 2010), in which the objective includes the policy entropy as a regularization term. With such regularization, the optimal policy samples the Boltzmann distribution of the negative reward, which is the target distribution in our setting. This observation is the basis for the connection between control (policy optimization) and inference (sampling) (Levine, 2018). (See Eysenbach & Levine (2022) for the connection to KL-constrained or adversarial policy optimization, Zhang & Chen (2022); Domingo-Enrich et al. (2024) for the related derivation of sampling via stochastic control with a diffusion policy, and Tiapkin et al. (2023); Deleu et al. (2024) for the connections between MaxEnt RL and off-policy objectives for training samplers of discrete sequentially constructed objects.)

Distributions over trajectories. The possible trajectories in the MDP starting at $\bullet$ and ending in $\perp$ have the form $\bullet \to (x_{t_0}, t_0) \to \cdots \to (x_{t_N}, t_N) \to \perp$, which we sometimes abbreviate to $x_{t_0} \to x_{t_1} \to \cdots \to x_{t_N}$. Following the policy $\overrightarrow{\pi}$ for $N+1$ steps starting at $\bullet$ yields a distribution

over trajectories $x_{t_0} \to x_{t_1} \to \cdots \to x_{t_N}$, *i.e.*,

$$\widehat{\mathbb{P}}(\widehat{X}) = \widehat{\mathbb{P}}(\widehat{X}_0) \prod_{n=0}^{N-1} \widehat{\mathbb{P}}(\widehat{X}_{n+1} \mid \widehat{X}_n) = p_{\text{prior}}(\widehat{X}_0) \prod_{n=0}^{N-1} \overrightarrow{\pi}_n(\widehat{X}_{n+1} \mid \widehat{X}_n). \tag{3}$$

The same construction is possible in reverse time: a density p_{target} over $\widehat{X}_N$ and a policy $\overleftarrow{\pi}$ (analogously to (2) defining transition probabilities from $\widehat{X}_{n+1}$ to $\widehat{X}_n$) on the reverse MDP yields a Markovian distribution over trajectories $\widehat{\mathbb{Q}}$, given analogously to (3) in reverse time. Given a (forward) policy, the reverse policy generating the same distribution over trajectories can be recovered using the marginal state visitation distributions via the detailed balance formula (8).

Radon-Nikodym derivative and divergences. The distributions $\widehat{\mathbb{P}}$, $\widehat{\mathbb{Q}}$ determined by a pair of policies $\overrightarrow{\pi}$, $\overleftarrow{\pi}$ and densities p_{prior}, p_{target} allow us to develop divergences (losses) for learning the parameters of suitable parametric families of policies. Our goal is to make the forward and reverse processes approximately equal by minimizing a divergence between the distributions over their trajectories. The density ratio of these distributions, also known as *Radon-Nikodym derivative*, is given by

$$\frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X}) = \frac{\widehat{\mathbb{P}}(\widehat{X})}{\widehat{\mathbb{Q}}(\widehat{X})} = \frac{\widehat{\mathbb{P}}(\widehat{X}_0) \prod_{n=0}^{N-1} \widehat{\mathbb{P}}(\widehat{X}_{n+1} \mid \widehat{X}_n)}{\widehat{\mathbb{Q}}(\widehat{X}_N) \prod_{n=0}^{N-1} \widehat{\mathbb{Q}}(\widehat{X}_n \mid \widehat{X}_{n+1})} = \frac{p_{\text{prior}}(\widehat{X}_0) \prod_{n=0}^{N-1} \overrightarrow{\pi}_n(\widehat{X}_{n+1} \mid \widehat{X}_n)}{p_{\text{target}}(\widehat{X}_N) \prod_{n=0}^{N-1} \overleftarrow{\pi}_{n+1}(\widehat{X}_n \mid \widehat{X}_{n+1})}. \tag{4}$$

Using (4), we can write the *Kullback-Leibler* (KL) divergence as

$$\begin{aligned} D_{\text{KL}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}) &\coloneqq \mathbb{E}_{\widehat{X} \sim \widehat{\mathbb{P}}} \left[\log \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X}) \right] \\ &= \mathbb{E}_{\widehat{X} \sim \widehat{\mathbb{P}}} \left[\log p_{\text{prior}}(\widehat{X}_0) + \mathcal{E}(\widehat{X}_N) + \sum_{n=0}^{N-1} \log \frac{\overrightarrow{\pi}_n(\widehat{X}_{n+1} \mid \widehat{X}_n)}{\overleftarrow{\pi}_{n+1}(\widehat{X}_n \mid \widehat{X}_{n+1})} \right] + \log Z. \end{aligned} \tag{5}$$

Since $\log Z$ is constant, this expression can be minimized via gradient descent on the parameters of the policies, for instance by zeroth-order gradient estimation (REINFORCE; Williams (1992)). If the policies allow for a differentiable reparametrization as a function of noise (*e.g.*, if they are conditionally Gaussian) we can use a deep reparametrization trick, amounting to writing the KL as a function of the noises introduced at each step. In particular, by fitting the parameters of $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$ so that the two processes are approximate time-reversals of one another, we also get an approximate solution to the sampling problem, *i.e.*, $\widehat{X}_N$ is approximately distributed as the target distribution p_{target}. This can be motivated by the *data processing inequality*, which yields that

$$D_{\text{KL}}(\widehat{\mathbb{P}}(\widehat{X}_N), p_{\text{target}}(\widehat{X}_N)) \leq D_{\text{KL}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}). \tag{6}$$

We can also consider other divergences between two measures $\widehat{\mathbb{P}}$ and $\widehat{\mathbb{Q}}$. For instance, the *trajectory balance* (TB, also known as *second-moment*, Malkin et al. (2022); Nüsken & Richter (2021)) and related *log-variance* (LV, also known as *VarGrad*, Richter et al. (2020)) divergences are given by

$$D_{\text{TB}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}) = \mathbb{E}_{\widehat{X} \sim \widehat{\mathbb{W}}} \left[\left(\log \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X}) \right)^2 \right] \quad \text{and} \quad D_{\text{LV}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}) = \operatorname{Var}_{\widehat{X} \sim \widehat{\mathbb{W}}} \left[\log \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X}) \right], \tag{7}$$

where the density ratio inside the square is given by (4) and $\widehat{\mathbb{W}}$ is a reference measure. We are free in the choice of reference measure, which allows for exploration in the optimization task (in RL, this is called *off-policy* training). We note that computing the second-moment divergence in (7) requires either knowledge of the normalizing constant Z of p_{target} or a learned approximation, with the LV divergence coinciding with TB when using a batch-level estimate of $\log Z$ (see, *e.g.*, Malkin et al. (2023, §2.3)). While estimators of the two divergences in (7) have different variance (which is related to *baselines* in RL), the expectations of their gradients with respect to the policy of $\widehat{\mathbb{P}}$ coincide when $\widehat{\mathbb{W}} = \widehat{\mathbb{P}}$ and are then, in turn, equal to the gradient of the KL divergence (5) (Richter et al., 2020; Malkin et al., 2023). In §2.2, we will see that one can define analogous concepts in continuous time.

Local divergences. Instead of looking at entire trajectories, we can as well define divergences locally, *i.e.*, on small parts of the trajectories. To this end, one can define the so-called *detailed balance* (DB) divergence as

$$D^{\widehat{\mathbb{W}}}_{\mathrm{DB},n}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) = \mathbb{E}_{\widehat{X}\sim\widehat{\mathbb{W}}}\left[\log\left(\frac{\widehat{p}_n(\widehat{X}_n)\overrightarrow{\pi}(\widehat{X}_{n+1} \mid \widehat{X}_n)}{\widehat{p}_{n+1}(\widehat{X}_{n+1})\overleftarrow{\pi}(\widehat{X}_n \mid \widehat{X}_{n+1})}\right)^2\right], \tag{8}$$

for the time step n, where $\widehat{p}_n$ is a learned estimate of the density of $\widehat{X}_n$ for $0 < n < N$, while $\widehat{p}_0 = p_{\mathrm{prior}}$ and $\widehat{p}_N = p_{\mathrm{target}}$ are fixed. Minimizing the DB divergence enforces that the transition kernels $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$ of $\widehat{\mathbb{P}}$ and $\widehat{\mathbb{Q}}$, respectively, are stochastic transport maps between distributions with densities $\widehat{p}_n$ and $\widehat{p}_{n+1}$, for each n. If the policies and density estimates jointly minimize (8) to 0 for some full-support reference distribution $\widehat{\mathbb{W}}$ and all n, it can be shown that they also minimize the trajectory-level divergences (7); see Bengio et al. (2021) for the discrete case, Lahlou et al. (2023) for the continuous case, Malkin et al. (2023) for the connection to nested variational inference (Buchner, 2021), and Deleu & Bengio (2023) for the connection to detailed balance for Markov chains. The divergence used for training may be a (possibly weighted[1]) sum of the DB divergences (8) for $n = 0, \dots, N-1$. 'Subtrajectory' interpolations between the global TB objective (7) and the local DB objective (8) exist; see Appendix B.4 and Nüsken & Richter (2023).

Uniqueness of solutions. Learning both the generative policy $\overrightarrow{\pi}$ and the time-reversed policy $\overleftarrow{\pi}$ in the general setting as above leads to non-unique solutions. We can achieve uniqueness of the objectives by prescribing $\overleftarrow{\pi}$ (as in diffusion models), adding additional regularizers (as in Schrödinger (half-)bridges), or prescribing the densities $(\widehat{\mathbb{P}}(\widehat{X}_n))_{n=1}^{N-1}$ and imposing constraints on the policies (as in annealing schemes); see Blessing et al. (2024, Tables 6 & 7) and Sun et al. (2024).

2.2 Continuous-time setting: Neural SDEs

We consider neural stochastic differential equations (neural SDEs) with isotropic additive noise, *i.e.*, families of stochastic processes $X = (X_t)_{t\in[0,1]}$ given as solutions of SDEs of the form

$$\mathrm{d}X_t = \overrightarrow{\mu}(X_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \qquad X_0 \sim p_{\mathrm{prior}}, \tag{9}$$

where $\overrightarrow{\mu}\colon \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is the *drift* (also called the *control function*), parametrized by a neural network[2]; $\sigma\colon [0,1] \to \mathbb{R}_{>0}$ is the *diffusion rate*, which in this paper is assumed to be fixed (more generally, it could be a $d \times d$ matrix that depends also on X_t); and W_t is a standard d-dimensional Brownian motion. Using a time discretization, the drift $\overrightarrow{\mu}$, together with the noise given by the diffusion rate and the Brownian motion, can be connected to a policy $\overrightarrow{\pi}$ of a MDP, which can be sampled to approximately simulate the process X (see §2.3).

Distributions over trajectories. Similar to the previous section, we can define a measure on the trajectories of the process X. Since the trajectories $t \mapsto X_t$ are almost surely continuous, the distribution (also known as *law* or *push-forward*) of the process X defines a *path space measure* $\mathbb{P}$, which is a measure on the space $C([0,1], \mathbb{R}^d)$ of continuous functions, representing the distribution of trajectories of X. We will show in §2.3 that such a path measure can be interpreted as the limit of distributions over discrete-time trajectories as in (3) when the step-sizes $t_{n+1} - t_n$ tend to zero.

We can also define the time marginals $p\colon \mathbb{R}^d \times [0,1] \to \mathbb{R}$, where for each time $t \in [0,1]$, $p(\cdot, t)$ gives the density of X_t. In measure-theoretic notation, the time marginals are the densities of the pushforwards of the path measure $\mathbb{P}$ by the evaluation maps $X \mapsto X_t$ sending a continuous function (trajectory) to its value at time t. Thus, we will also denote the distribution of the time marginals by $\mathbb{P}_t$. The evolution of p is governed by the *Fokker-Planck equation* (FPE), which is the partial differential equation (PDE)

$$\partial_t p = -\nabla \cdot (p\overrightarrow{\mu}) + \frac{\sigma^2}{2}\Delta p, \quad p(\cdot, 0) = p_{\mathrm{prior}}, \tag{10}$$

[1] Our result Prop. 3.4 suggests a weighting of $\frac{1}{N\Delta t_n}$, in the notation of §2.3, but our experiments showed no significant difference between such a weighting and a uniform one.

[2] For notational convenience, we do not make the dependence of X on the neural network parameters explicit.

where Δp denotes the Laplacian of p. The Fokker-Planck equation generalizes the *continuity equation* for ordinary differential equations, which corresponds to the case $\sigma = 0$. It expresses the conservation of probability mass when particles distributed with density $p(\cdot, t)$ are stochastically transported by the drift $\overrightarrow{\mu}$ and diffused with scale σ. While such a PDE perspective is only possible in continuous time, in §3 we derive that certain MDPs satisfy FPEs in the limit of finer time discretizations.

Reverse process. As for reverse-time MDPs, we can also define reverse-time SDEs

$$\mathrm{d}X_t = \overleftarrow{\mu}(X_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\overleftarrow{W}_t, \qquad X_1 \sim p_{\text{target}}, \tag{11}$$

where $\overleftarrow{W}_t$ is a reverse-time[3] Brownian motion and $\overleftarrow{\mu}$ is a suitable drift, potentially also parametrized by a neural network. This SDE gives rise to another path space measure $\mathbb{Q}$. While in discrete time (§2.1) local reversibility is given by detailed balance (8), in continuous time one can characterize when the path space measure $\mathbb{Q}$ of the reverse-time SDE in (11) coincides with the path space measure $\mathbb{P}$ of the forward SDE in (9) by a local condition known as Nelson's identity (Nelson (1967), also attributed to Anderson (1982)), which states that $\mathbb{Q} = \mathbb{P}$ if and only if

$$\overleftarrow{\mu} = \overrightarrow{\mu} - \sigma^2 \nabla \log p \quad \text{and} \quad \mathbb{Q}_1 = \mathbb{P}_1, \tag{12}$$

where p denotes the densities of $\mathbb{P}$'s time marginals. It can be shown that substituting this expression into the FPE for the backward process recovers the FPE (10) for the forward process, and similarly that the KL divergence, given by (15) below, between the forward and backward processes is zero.

Radon-Nikodym derivative and divergences. Since we typically cannot compute the time marginals, we cannot directly use Nelson's identity to solve the sampling problem. However, similar to §2.1, we can establish learning problems to infer the parameters of the neural networks $\overrightarrow{\mu}$, $\overleftarrow{\mu}$, so that the induced terminal distribution of the forward SDE (9) is close to the target, $\mathbb{P}_1 \approx p_{\text{target}}$, in some suitable measure of divergence.

The tool to establish such learning problems is Girsanov's theorem, which states the following. Let $\mathbb{P}^{(1)}$ and $\mathbb{P}^{(2)}$ be the path space measures defined by SDEs of the form (9) with drifts $\overrightarrow{\mu}^{(1)}$, $\overrightarrow{\mu}^{(2)}$. Then, for $\mathbb{P}^{(2)}$-almost every $X \in C([0,1], \mathbb{R}^d)$, the Radon-Nikodym derivative is given by

$$\log \frac{\mathrm{d}\mathbb{P}^{(1)}}{\mathrm{d}\mathbb{P}^{(2)}}(X) = \int_0^1 \frac{\|\overrightarrow{\mu}^{(2)}(X_t,t)\|^2 - \|\overrightarrow{\mu}^{(1)}(X_t,t)\|^2}{2\sigma(t)^2}\,\mathrm{d}t + \int_0^1 \frac{\overrightarrow{\mu}^{(1)}(X_t,t) - \overrightarrow{\mu}^{(2)}(X_t,t)}{\sigma(t)^2} \cdot \mathrm{d}X_t. \tag{13}$$

An intuitive explanation of (13) using a discrete-time approximation can be found in Särkkä & Solin (2019, Section 7.4) or in the proof of Lemma B.7. The same result holds for reverse-time processes as in (11) with $\mathrm{d}X_t$ replaced by integration against the reverse-time process $\mathrm{d}\overleftarrow{X}_t$. Using a reversible Brownian motion as a reference path measure (see Léonard (2014; 2013)), we can thus derive the Radon-Nikodym derivative between the path measures $\mathbb{P}$ and $\mathbb{Q}$ of the forward and reverse-time SDEs in (9) and (11) as

$$\begin{aligned}\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X) = \log \frac{p_{\text{prior}}(X_0)}{p_{\text{target}}(X_1)} &+ \int_0^1 \frac{\|\overleftarrow{\mu}(X_t,t)\|^2 - \|\overrightarrow{\mu}(X_t,t)\|^2}{2\sigma(t)^2}\,\mathrm{d}t \\ &+ \int_0^1 \frac{\overrightarrow{\mu}(X_t,t)}{\sigma(t)^2} \cdot \mathrm{d}X_t - \int_0^1 \frac{\overleftarrow{\mu}(X_t,t)}{\sigma(t)^2} \cdot \mathrm{d}\overleftarrow{X}_t,\end{aligned} \tag{14}$$

see Vargas et al. (2024). A related result was derived by Richter & Berner (2024) using the conversion formula $\int_0^1 f(X_t,t) \cdot \mathrm{d}X_t = \int_0^1 f(X_t,t) \cdot \mathrm{d}\overleftarrow{X}_t - \int_0^1 \sigma(t)^2 \nabla \cdot f(X_t,t)\,\mathrm{d}t$. By integrating (14) over $X \sim \mathbb{P}$, it can be derived that the KL divergence is given by an expression analogous to (5):

$$\begin{aligned}D_{\mathrm{KL}}(\mathbb{P},\mathbb{Q}) = \mathbb{E}_{X\sim\mathbb{P}}\Bigg[&\log p_{\text{prior}}(X_0) + \mathcal{E}(X_T) \\ &+ \int_0^1 \left(\frac{\|\overrightarrow{\mu}(X_t,t) - \overleftarrow{\mu}(X_t,t)\|^2}{2\sigma(t)^2} - \nabla \cdot \overleftarrow{\mu}(X_t,t) \right) \mathrm{d}t \Bigg] + \log Z,\end{aligned} \tag{15}$$

[3] We refer to Kunita (2019); Vargas et al. (2024) for details on reverse-time SDEs and backward Itô integration.

Informally, the derivation uses that in expectation over $X \sim \mathbb{P}$, the integral with respect to $\mathrm{d}X_t$ in (14) is the sum of an integral with respect to $\overrightarrow{\mu}(X_t)\,\mathrm{d}t$ and a stochastic integral with zero expectation.

The KL divergence can also be interpreted as the cost of a continuous-time stochastic optimal control problem (Dai Pra, 1991; Berner et al., 2022). Some objectives, such as those in Zhang & Chen (2022), optimize the parameters of the drift defining $\mathbb{P}$ by minimizing variants of the KL divergence (15) approximately: by passing to a time discretization of the SDE (§2.3) and expressing the objective as a function of the Gaussian noises introduced at each step of the SDE integration, amounting to a deep reparametrization trick. For suitable integration schemes (Vargas et al., 2023; 2024), the discretized Radon-Nikodym derivative can be written as a density ratio, so that this approach corresponds to optimizing a discrete-time KL as in (5).

Analogously to the discrete-time setting (7), we can also consider the second-moment or log-variance divergences $D^{\mathbb{W}}_{\mathrm{TB}}(\mathbb{P},\mathbb{Q}) = \mathbb{E}_{X\sim\mathbb{W}}\left[\left(\log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right)^2\right]$ and $D^{\mathbb{W}}_{\mathrm{LV}}(\mathbb{P},\mathbb{Q}) = \mathrm{Var}_{X\sim\mathbb{W}}\left[\log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right]$, where $\mathbb{W}$ is a reference path space measure. These divergences were explored by Nüsken & Richter (2021).

Local time reversal: PDE viewpoint. The continuous-time perspective also offers to employ the PDE framework for learning the dynamical measure transport. Recall that the density p of the process X defined in (9) fulfills the Fokker-Planck equation (10). One can thus aim to learn $\overrightarrow{\mu}$ so as to make it satisfy the FPE, with the boundary values $p(\cdot,0) = p_{\mathrm{prior}}$ and $p(\cdot,1) = p_{\mathrm{target}}$, where p is either prescribed or also learned (as done in Máté & Fleuret (2023)). In Sun et al. (2024) it is shown that when using suitable losses on this problem one recovers a loss equivalent to D_{TB}. When choosing the diffusion loss from Nüsken & Richter (2023), one recovers a continuous-time variant of D_{SubTB} (see Appendix B.4) and thus D_{DB}. In §3, we show that it also works the other way around: we can start with the discrete-time detailed balance divergence and derive PDE constraints in the limit.

2.3 From SDEs to discrete-time Euler-Maruyama policies

Simulation of the process X can be achieved by discretizing time and applying a numerical integration scheme, such as the Euler-Maruyama scheme (Maruyama, 1955). Specifically, one fixes a sequence of time points $0 = t_0 < t_1 < \cdots < t_N = 1$ and defines the discrete-time process $\widehat{X} = (\widehat{X}_n)_{n=0}^N$ by

$$\widehat{X}_0 \sim p_{\mathrm{prior}}, \quad \widehat{X}_{n+1} = \widehat{X}_n + \overrightarrow{\mu}(\widehat{X}_n, t_n)\Delta t_n + \sigma(t_n)\sqrt{\Delta t_n}\,\xi_n, \quad \xi_n \sim \mathcal{N}(0, I_d), \tag{16}$$

where $\Delta t_n \coloneqq t_{n+1} - t_n$. This defines the policy $\overrightarrow{\pi}(a \mid (x,t_n)) = \mathcal{N}(a; x + \overrightarrow{\mu}(x,t_n)\Delta t_n, \sigma(t_n)^2\Delta t_n)$ on an MDP as in (2). It is clear by comparing (2) and (16) that this distribution exactly coincides with the distribution $\widehat{\mathbb{P}}$ in (3) over sequences $(\widehat{X}_0, \widehat{X}_1, \ldots, \widehat{X}_N)$ of the Euler-Maruyama-discretized process $\widehat{X}$. As we will discuss below, with decreasing mesh size, the marginals $\widehat{\mathbb{P}}(X_n)$ of the n-th step of the discretized process converge to the marginals $p(\cdot,t_n)$ of the continuous-time process at time t_n. Based on the Central Limit Theorem, such convergence can also be shown for non-Gaussian policies that satisfy suitable consistency conditions (Kloeden & Platen, 1992, §6.2).

Finally, the same discretization is possible for reverse time: a reverse-time process of the form (11) with drift function $\overrightarrow{\mu}$ together with a target density p_{target} determine a policy $\overleftarrow{\pi}$ on the reverse MDP, corresponding to reverse Euler-Maruyama integration:

$$\widehat{X}_N \sim p_{\mathrm{target}}, \quad \widehat{X}_n = \widehat{X}_{n+1} - \overleftarrow{\mu}(\widehat{X}_{n+1}, t_{n+1})\Delta t_n - \sigma(t_{n+1})\sqrt{\Delta t_n}\,\xi_n, \quad \xi_n \sim \mathcal{N}(0, I_d). \tag{17}$$

However, note that the Euler-Maruyama discretizations of a process and of its reverse-time process defined by (12) do not, in general, coincide. That is, a policy on the reverse MDP can be constructed either by discretizing an SDE to yield a policy on the forward MDP, then reversing it, or by discretizing the reverse SDE to directly obtain a policy on the reverse MDP, possibly with different results. In particular, the Gaussianity of transitions is not preserved under time reversal: the reverse of a discrete-time process with Gaussian increments does not, in general, have Gaussian increments. However, Nelson's identity (12) shows that the two are equivalent in the continuous-time limit.

The discretization allows us to compare the two Radon-Nikodym derivatives: those of the discretizations in (4) and of the continuous-time processes in (14). In particular, in Lemma B.7 we will show that these expressions are equal in the limit.

3 Asymptotic convergence

3.1 Distributions over trajectories

A standard result shows that the discretized process $\widehat{X}$ converges to the continuous counterpart X as the time discretization becomes finer, *i.e.*, as the maximal step size $\max_{n=0}^{N-1} \Delta t_n$ goes to zero (Maruyama, 1955). The precise statement of convergence requires the processes to be embedded in a common probability space. Let ι be the mapping from the observation space of $\widehat{X}$ (discrete-time trajectories) to that of X (continuous-time paths) that takes a sequence $\widehat{X}_0, \dots, \widehat{X}_n$ to the function $f \in C([0,1], \mathbb{R}^d)$ defined by $f(t_n) = \widehat{X}_n$ and linearly interpolating between the t_n (note that ι implicitly depends on the discretization). We then have convergence of $\iota(\widehat{X})$ to X:

Proposition 3.1 (Convergence of Euler-Maruyama scheme)**.** *As* $\max_{n=0}^{N-1} \Delta t_n \to 0$*,* $\iota(\widehat{X})$ *converges weakly and strongly to* X *with order* $\gamma = 1$ *and the path measures* $\iota_* \widehat{\mathbb{P}}$ *converge weakly to* $\mathbb{P}$*.*

We refer the reader to Appendix B.3 for definitions of strong and weak convergence. The result can, *e.g.*, be found in Kloeden & Platen (1992) and we refer to Baldi (2017, Corollary 11.1) and Kloeden & Neuenkirch (2007) for the convergence of path measures. Generally, the Euler-Maruyama scheme has order of strong convergence $\gamma = 1/2$. However, since we consider *additive* noise, *i.e.*, σ not depending on the spatial variable x, the *Milstein scheme* reduces to the Euler-Maruyama scheme and we inherit order $\gamma = 1$ as stated in Prop. 3.1 (Kloeden & Platen, 1992, Section 10.2 and 10.3).

3.2 Radon-Nikodym derivative and divergences

Beyond the convergence of path measures, this section shows – more relevant for practical applications – that commonly used local and global objectives converge their continuous-time counterparts as the time discretization is refined. To this end, we leverage Lemma B.7, which analyzes the convergence of time discretizations of Radon-Nikodym derivatives $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ appearing in (14) to their discrete-time analogs $\frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}$. We note that Vargas et al. (2024, Proposition E.1) shows that, for constant σ, an Euler-Maruyama discretization of $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ can be written as a density ratio as in (4). This also implies that the ratio in the detailed balance divergence in (8) arises from a single-step Euler-Maruyama approximation of the Radon-Nikodym derivative $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ on the subinterval $[t_n, t_{n+1}]$. We present proofs of all results in this Section in Appendix B.6.

Global objectives: Second-moment divergences approach the continuous-time equivalents. The following key result uses convergence of the Radon-Nikodym derivatives (Lemma B.7):

Proposition 3.2 (Convergence of functionals)**.** *If* $\mathbb{P}, \mathbb{Q}, \mathbb{W}$ *are path measures of three forward-time SDEs, the drifts of the SDEs and their derivatives are bounded, and* $f \in C^\infty(\mathbb{R}, \mathbb{R})$ *has at most polynomially growing derivatives, then*

$$\mathbb{E}_{\widehat{X} \sim \widehat{\mathbb{W}}} \left[f\left(\log \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X}) \right) \right] \xrightarrow{\max_n \Delta t_n \to 0} \mathbb{E}_{X \sim \mathbb{W}} \left[f\left(\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X) \right) \right].$$

We now show that the second-moment losses in (7) converge to their continuous-time counterparts.

Proposition 3.3 (Asymptotic consistency of TB and VarGrad)**.** *Under the assumptions of Prop. 3.2, the divergences* $D_{\mathrm{TB}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}})$ *and* $D_{\mathrm{LV}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}})$ *converge to* $D_{\mathrm{TB}}^{\mathbb{W}}(\mathbb{P}, \mathbb{Q})$ *and* $D_{\mathrm{LV}}^{\mathbb{W}}(\mathbb{P}, \mathbb{Q})$*, respectively.*

The convergence holds for the TB divergence with respect to any c, *i.e.*, $\mathbb{E}_{\widehat{\mathbb{W}}}\left[\left(\log \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}} - c \right)^2 \right]$, showing that Prop. 3.3 continues to hold if one uses a learned estimate of the log-partition function $\log Z$ in the TB divergence, as typically done in practice.

Local objectives: Detailed balance approaches the Fokker-Planck PDE. Consider a pair of forward and reverse SDEs with drifts $\overrightarrow{\mu}$ and $\overleftarrow{\mu}$, respectively, defining processes $\mathbb{P}$ and $\mathbb{Q}$, and suppose that $\widehat{p}\colon \mathbb{R}^d \times [0,1] \to \mathbb{R}$ is a density estimate with $\widehat{p}(\cdot, 0) = p_{\text{prior}}$ and $\widehat{p}(\cdot, 1) = p_{\text{target}}$.

For $0 \le t < t' \le 1$, consider any time discretization in which t and t' are adjacent time steps ($t_n = t$ and $t_{n+1} = t'$). The discretization defines a pair of policies $\overrightarrow{\pi}, \overleftarrow{\pi}$ corresponding to Euler-Maruyama discretizations of the two SDEs. Let us define the *detailed balance discrepancy*:

$$\Delta_{t \to t'}(x, x') := \log \frac{\widehat{p}_n(x)\overrightarrow{\pi}_n(x' \mid x)}{\widehat{p}_{n+1}(x')\overleftarrow{\pi}_{n+1}(x \mid x')}, \tag{18}$$

where we set $\widehat{p}_n(x) = \widehat{p}(x, t_n)$. Recalling the definition (8), we have that

$$D^{\widehat{\mathbb{W}}}_{\mathrm{DB},n}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) = \mathbb{E}_{\widehat{Z} \sim \widehat{\mathbb{W}}}\left[\Delta_{t_n \to t_{n+1}}(\widehat{Z}_n, \widehat{Z}_{n+1})^2\right]. \tag{19}$$

The following proposition will show that the two SDEs are time reversals of one another if and only if certain asymptotics of the DB discrepancy vanish. It is proved using a technical lemma (Lemma B.8), which shows that the asymptotics of the discrepancy in h are precisely the errors in the satisfaction of Nelson's identity and the Fokker-Planck equation.

Proposition 3.4 (Asymptotic equality of DB and FPE)**.** *Under the smoothness conditions in Lemma B.8, $\overrightarrow{\mu}, \overleftarrow{\mu}, \widehat{p}$ jointly satisfy Nelson's identity ($\overleftarrow{\mu} = \overrightarrow{\mu} - \sigma^2 \nabla \log \widehat{p}$) at (x_t, t) if and only if*

$$\lim_{h \to 0}\left[\frac{1}{\sqrt{h}}\Delta_{t \to t+h}(x_t, x_{t+h})\right] = 0 \quad \textit{for almost every } z,$$

where $x_{t+h} := x_t + \overrightarrow{\mu}(x_t, t)h + \sigma(t)\sqrt{h}z$. If in addition

$$\lim_{h \to 0} \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}\left[\frac{1}{h}\Delta_{t \to t+h}(x_t, x_{t+h})\right] = 0,$$

then the Fokker-Plank equation is satisfied at (x_t, t). If both conditions hold at all $(x_t, t) \in \mathbb{R}^d \times (0, 1)$, then $\overrightarrow{\mu}, \overleftarrow{\mu}$ define a pair of time-reversed processes with marginal density $\widehat{p}$.

In particular, this result shows that if we *impose* a parametrization of $\overrightarrow{\mu}$ and $\overleftarrow{\mu}$ as two vector fields that differ by $\sigma^2 \nabla \log \widehat{p}$, where $\widehat{p}$ is a fixed or learned marginal density estimate, then asymptotic satisfaction of DB implies that the continuous-time forward and backward processes coincide.

Generalization to processes defined by discrete-time reversal. The generative and diffusion processes play a symmetric role in Prop. 3.4. However, some past work – starting from Zhang & Chen (2022), from which we adopt the experiment settings in §4 – has defined $\overleftarrow{\pi}$ as the reversal of the Euler-Maruyama discretization of a forward SDE, rather than as the Euler-Maruyama discretization of a backward SDE, in a special case where the former happens to have Gaussian increments. To ensure the applicability of the results to the experiment setting, we need a slight generalization:

Proposition 3.5 (DB and FPE for Brownian bridges)**.** *The results of Prop. 3.4 hold if $\sigma(t)$ is constant and $\overleftarrow{\pi}$ is the discrete-time reversal of the Euler-Maruyama discretization of the process*

$$p_{\mathrm{prior}}(x) = \mathcal{N}(x; 0, \sigma_0 I_d), \quad \mathrm{d}X_t = \sigma(t)\,\mathrm{d}W_t. \tag{20}$$

Our theoretical results guarantee that global and local objectives with different discretizations are approximating unique continuous-time objects when $\max_{n=0}^{N-1} \Delta t_n \to 0$. This justifies training and inference of samplers with different discretizations, allowing us to greatly reduce the computational cost of training (see §4). These observations are particularly relevant for diffusion-based samplers which rely on discretization of (partial) trajectories during training. In contrast, for generative modeling, one can use denoising score-matching objectives which can be minimized without any discretization in continuous time.

4 Experiments

We evaluate the effect of time discretization on the training of diffusion samplers using the objectives introduced in §2, targeting several unnormalized densities. In all experiments, we follow the training setting from Sendera et al. (2024), extending their published code with an implementation of variable time discretization (see Appendix C.1 for details). The following objectives are considered:

- **Path integral sampler (PIS)** (Zhang & Chen, 2022): The trajectory-level KL divergence (5), which approximates the path space measure KL (15) is minimized via the deep reparametrization trick (*i.e.*, through differentiable simulation of the generative SDE, hence necessarily on-policy).
- **Trajectory balance (TB)** and **VarGrad**: The trajectory-level divergences of the second-moment type (7), optimized either on-policy or using the off-policy local search technique introduced in Sendera et al. (2024). As TB and VarGrad are found to be nearly equivalent in unconditional sampling settings, we consider VarGrad only for *conditional* sampling (see Fig. 9).
- **Detailed balance (DB)**: The time-local detailed balance divergence (8), and its variant **FL-DB**, which places an inductive bias on the log-density estimates – first used by Wu et al. (2020); Máté & Fleuret (2023) and evaluated in the off-policy RL setting by Zhang et al. (2024); Sendera et al. (2024) – that assumes access to the target energy at intermediate time points (see Appendix B.5).

Each objective is additionally studied with and without the **Langevin parametrization (LP)**, a technique introduced by Zhang & Chen (2022) that parametrizes the generative SDE's drift function via the gradient of the target energy. The assumptions made by each objective are summarized in Table 1.

Table 1: Properties of training objectives. Variants with LP also use the intermediate energy gradient.

Property ↓ Objective →	PIS	TB/VarGrad	DB	FL-DB
Time-local	✗	✗	✓	✓
Off-policy	✗	✓	✓	✓
Use intermediate energy	✗	✗	✗	✓
Use energy gradient	✓	✗	✗	✗

The noising process is always fixed to the reverse of a Brownian motion, following Zhang & Chen (2022) and subsequent work. The following densities are targeted:

- Standard targets **25GMM** (2-dimensional mixture of Gaussians), **Funnel** (10-dimensional funnel-shaped distribution), **Manywell** (32-dimensional synthetic energy), and **LGCP** (1600-dimensional log-Gaussian Cox process) as defined in the benchmarking library of Sendera et al. (2024).
- **VAE**: the conditional task of sampling the 20-dimensional latent z of a variational autoencoder trained on MNIST given an input image x, with target density $p(z \mid x) \propto p(x \mid z)p(z)$.
- Bayesian logistic regression problems for the **German Credit** and **Breast Cancer** datasets (25- and 31-dimensional, respectively), from the benchmark by Blessing et al. (2024).

We use a well-established primary metric: the ELBO of the target distribution computed using the learned sampler and the true log-partition function, estimated using N-step Euler-Maruyama integration. In our notation, the ELBO is $\log \widehat{Z} = \mathbb{E}_{\widehat{X}\sim\widehat{\mathbb{P}}}\left[-\mathcal{E}(\widehat{X}_N) + \log \frac{\widehat{\mathbb{Q}}(\widehat{X}|\widehat{X}_N)}{\widehat{\mathbb{P}}(\widehat{X})}\right]$ (see (33) for details). While recent work on diffusion samplers has used a discretization with uniform-length time intervals for both integration and training, we vary the time discretization. Unless stated otherwise, we evaluate ELBO using $N_{\text{eval}} = 100$ uniform discretization steps. However, during training, we vary the number of time steps N_{train} and their placement:

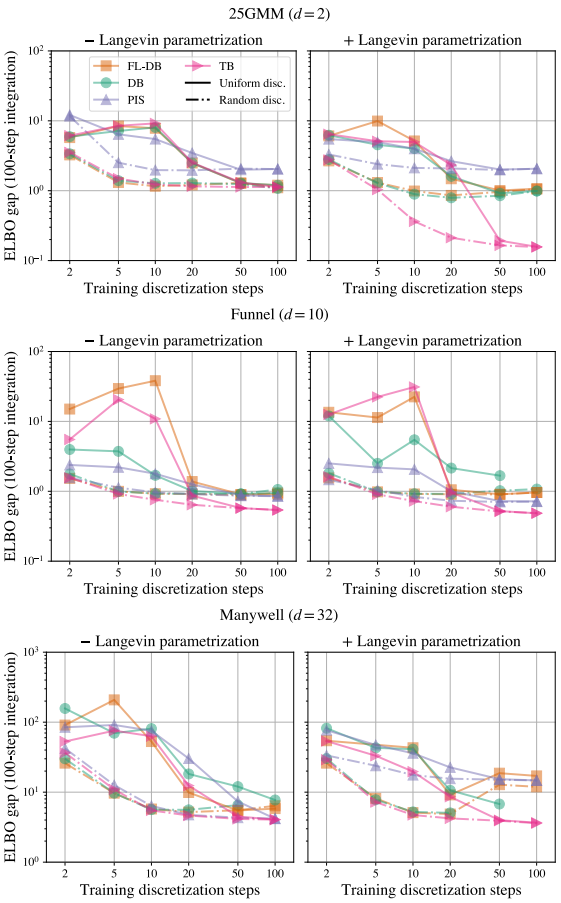


Figure 4: Difference between true $\log Z$ and ELBO as a function of N_{train}, always evaluating with 100-step uniform integration. Additional targets in Fig. 8 and Fig. 9, **Equidistant** results in Fig. 10.

- **Uniform**: Time steps uniformly spaced: $t_i = \frac{i}{N_{\text{train}}}$ for $i = 0, \ldots, N_{\text{train}}$.

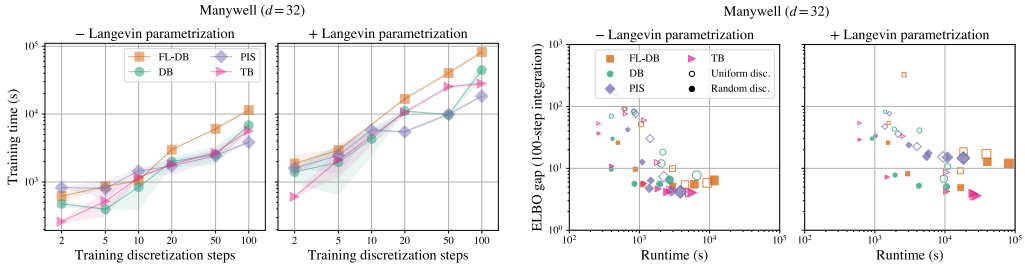


Figure 5: **Left:** Time to train for 25k iterations on **Manywell** as a function of N_{train}, mean and std over 3 runs (note the log-log scale). **Right:** Runtime and ELBO gap, showing that **Random** discretization gives a superior balance of speed and performance. Results for **25GMM** and **Funnel** densities in Fig. 11.

- **Random** and **Equidistant**: Two ways of constructing nonuniform partitions of the time interval $[0, 1]$ into N_{train} segments, described in Appendix C.2 and illustrated in Fig. 7.

Results: Training-time discretization. In Fig. 4, we show the ELBO gaps on three of the datasets for different training-time discretizations as a function of N_{train}. We observe that, for all objectives, training with **Random** discretization consistently outperforms **Uniform** discretization with a small number of steps, with the two converging as N_{train} increases to approach $N_{\text{eval}} = 100$. The **Equidistant** discretization performs similarly to **Random** in most cases (see Fig. 10).

Notably, the time-local objectives (DB and FL-DB) perform similarly to the trajectory-level objectives (TB and PIS) when trained with few steps. However, as N_{train} increases, the time-local models' performance typically plateaus or even (on some targets they even diverge with 100 steps). These results suggest that time-local objectives trained with nonuniform discretization and few steps can be a viable alternative to trajectory-level objectives in high-dimensional problems where the memory requirements associated with long trajectories are prohibitive.

Results: Time efficiency. The training time per iteration is expected to scale approximately linearly with the trajectory length N_{train}. Fig. 5 (left) confirms this scaling and illustrates the relative cost of different objectives: FL-DB and methods using the Langevin parametrization are the most expensive, as they require stepwise evaluations of the target energy and its gradient, respectively. Fig. 5 (right) shows the ELBO gap plotted against training time, demonstrating that methods with nonuniform discretization achieve a superior trade-off between training time and sampling performance.

Results: Inference-time discretization. To study the effect of *sampling-time* discretization, we train models with $N_{\text{train}} = 10$ steps (using TB with Langevin parametrization) and different placement of time steps, then evaluate with different $N_{\text{eval}} \in \{1, 2, ..., 100\}$. From Fig. 6, we observe that randomized discretization (**Random** or **Equidistant**) during training leads to smooth ELBO curves as a function of N_{eval}, whereas training with **Uniform** discretization gives unstable behavior with periodic features at multiples of N_{train}, which may be due both to the restricted set of inputs t to the model $\overrightarrow{\mu}(x, t)$ during training and to the harmonic timestep embedding in the model architecture. This result is further evidence that nonuniform discretization during training yields more robust samplers that are less sensitive to the choice of N_{eval}.

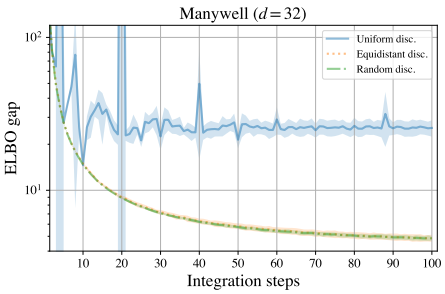


Figure 6: ELBO gaps for models trained with various discretization schemes and $N_{\text{train}} = 10$, then evaluated with various numbers of integration steps N_{eval}. Results on Manywell energy; others shown in Fig. 12.

Additional results. Figures complementing those in the main text appear in Appendices D.2 and D.3, while Appendix D.1 contains more metrics and comparisons in tabular form. In particular, we combine the above objectives with the off-policy local search of Sendera et al. (2024) to achieve near-state-of-the-art results with much coarser (nonuniform) time discretizations during training, whereas local search does not help the performance of methods using coarse **Uniform** schemes (Table 2).

5 Conclusion

We have shown the convergence of off-policy RL objectives used for the training of diffusion samplers to their continuous-time counterparts. Those are Nelson's identity and the Fokker-Planck equation for stepwise objectives and path space measure divergences for trajectory-level objectives. Our experimental results give a first understanding of good practices for training diffusion samplers in coarse time discretizations. We expect that the increased training efficiency and the ability to use local objectives without expensive energy evaluations are especially beneficial in very high-dimensional problems where trajectory length is a bottleneck, noting that trajectory balance was recently used in fine-tuning of diffusion foundation models for text and images (Venkatraman et al., 2024). Future theoretical work could generalize our results to diffusions on general Riemannian manifolds and to non-Markovian continuous-time processes, such as those studied in Daems et al. (2024); Nobis et al. (2023).

Acknowledgments

We thank Alex Tong, Alexandre Adam, and Pablo Lemos for helpful discussions at early stages of this project. J.B. acknowledges support from the Wally Baer and Jeri Weiss Postdoctoral Fellowship. The research of L.R. was partially funded by Deutsche Forschungsgemeinschaft (DFG) through the grant CRC 1114 "Scaling Cascades in Complex Systems" (project A05, project number 235221301). The research of M.S. was funded by National Science Centre, Poland, 2022/45/N/ST6/03374.

The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), and NVIDIA.

Code

Code is available at https://github.com/GFNOrg/gfn-diffusion/tree/stagger.

References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from Boltzmann densities. *International Conference on Machine Learning (ICML)*, 2024.

Michael S Albergo, Gurtej Kanwar, and Phiala E Shanahan. Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Michael Arbel, Alex Matthews, and Arnaud Doucet. Annealed flow transport Monte Carlo. *International Conference on Machine Learning (ICML)*, 2021.

Paolo Baldi. *Stochastic calculus*. Springer, 2017.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Denis Blessing, Xiaogang Jia, Johannes Esslinger, Francisco Vargas, and Gerhard Neumann. Beyond ELBOs: A large-scale evaluation of variational methods for sampling. *arXiv preprint arXiv:2406.07423*, 2024.

Johannes Buchner. Nested sampling methods. *arXiv preprint arXiv:2101.09675*, 2021.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

Rembert Daems, Manfred Opper, Guillaume Crevecoeur, and Tolga Birdal. Variational inference for SDEs driven by fractional noise. *International Conference on Learning Representations (ICLR)*, 2024.

Chenguang Dai, Jeremy Heng, Pierre E Jacob, and Nick Whiteley. An invitation to sequential Monte Carlo samplers. *Journal of the American Statistical Association*, 117(539):1587–1600, 2022.

Paolo Dai Pra. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23(1):313–329, 1991.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Tristan Deleu and Yoshua Bengio. Generative flow networks: a Markov chain perspective. *arXiv preprint arXiv:2307.01422*, 2023.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. *Uncertainty in Artificial Intelligence (UAI)*, 2024.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. *Neural Information Processing Systems (NeurIPS)*, 2021.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Arnaud Doucet, Will Sussman Grathwohl, Alexander GDG Matthews, and Heiko Strathmann. Score-based diffusion meets annealed importance sampling. *Neural Information Processing Systems (NeurIPS)*, 2022.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. *International Conference on Learning Representations (ICLR)*, 2022.

Marylou Gabrié, Grant M Rotskoff, and Eric Vanden-Eijnden. Efficient Bayesian sampling using normalizing flows to assist Markov chain Monte Carlo methods. *arXiv preprint arXiv:2107.08001*, 2021.

Paul Lyonel Hagemann, Johannes Hertrich, and Gabriele Steidl. *Generalized normalizing flows via Markov chains*. Cambridge University Press, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020.

Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1):1593–1623, 2014.

Robert E Kass, Bradley P Carlin, Andrew Gelman, and Radford M Neal. Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural SDEs as infinite-dimensional GANs. *International Conference on Machine Learning (ICML)*, 2021a.

Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural SDEs. *Neural Information Processing Systems (NeurIPS)*, 2021b.

Peter E Kloeden and Andreas Neuenkirch. The pathwise convergence of approximation schemes for stochastic differential equations. *LMS journal of Computation and Mathematics*, 10:235–253, 2007.

Peter E Kloeden and Eckhard Platen. *Stochastic differential equations*. Springer, 1992.

Hiroshi Kunita. *Stochastic flows and jump-diffusions*. Springer, 2019.

Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

Benedict J. Leimkuhler, Charles Matthews, and Michael V. Tretyakov. On the long-time integration of stochastic gradient systems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 470, 2014.

Pablo Lemos, Nikolay Malkin, Will Handley, Yoshua Bengio, Yashar Hezaveh, and Laurence Perreault-Levasseur. Improving gradient-guided nested sampling for posterior inference. *International Conference on Machine Learning (ICML)*, 2023.

Christian Léonard. A survey of the Schrödinger problem and some of its connections with optimal transport. *arXiv preprint arXiv:1308.0215*, 2013.

Christian Léonard. Some properties of path measures. *Séminaire de Probabilités XLVI*, pp. 207–230, 2014.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. *Artificial Intelligence and Statistics (AISTATS)*, 2020.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Neural Information Processing Systems (NeurIPS)*, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Gisiro Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4:48–90, 1955.

Bálint Máté and François Fleuret. Learning interpolations between Boltzmann densities. *Transactions on Machine Learning Research (TMLR)*, 2023.

Alex Matthews, Michael Arbel, Danilo Jimenez Rezende, and Arnaud Doucet. Continual repeated annealed flow transport monte carlo. *International Conference on Machine Learning (ICML)*, 2022.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*, 2023.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

E Nelson. Dynamical theories of Brownian motion. *Press, Princeton, NJ*, 1967.

Gabriel Nobis, Maximilian Springenberg, Marco Aversa, Michael Detzel, Rembert Daems, Roderick Murray-Smith, Shinichi Nakajima, Sebastian Lapuschkin, Stefano Ermon, Tolga Birdal, Manfred Opper, Christoph Knochenhauer, Luis Oala, and Wojciech Samek. Generative fractional diffusion models. *arXiv preprint arXiv:2310.17638*, 2023.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Nikolas Nüsken and Lorenz Richter. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2(4):48, 2021.

Nikolas Nüsken and Lorenz Richter. Interpolating between BSDEs and PINNs: Deep learning for elliptic and parabolic boundary value problems. *Journal of Machine Learning*, 2023.

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of GFlowNets with local credit and incomplete trajectories. *International Conference on Machine Learning (ICML)*, 2023.

Kushagra Pandey, Maja Rudolph, and Stephan Mandt. Efficient integrators for diffusion generative models. *International Conference on Learning Representations (ICLR)*, 2024.

George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. *International Conference on Machine Learning (ICML)*, 2024.

Dominic Phillips and Flaviu Cipcigan. MetaGFN: Exploring distant modes with adapted metadynamics for continuous GFlowNets. *arXiv preprint arXiv:2408.15905*, 2024.

Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco J. R. Ruiz, and Ömer Deniz Akyildiz. VarGrad: A low-variance gradient estimator for variational inference. *Neural Information Processing Systems (NeurIPS)*, 2020.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Lars Ruthotto, Stanley J Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, 2020.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International conference on machine learning*, pp. 1218–1226. PMLR, 2015.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. Improved off-policy training of diffusion samplers. *Neural Information Processing Systems (NeurIPS)*, 2024.

Neta Shaul, Juan Perez, Ricky T. Q. Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. *International Conference on Learning Representations (ICLR)*, 2024.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning (ICML)*, 2015.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Neural Information Processing Systems (NeurIPS)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2021b.

Jingtong Sun, Julius Berner, Lorenz Richter, Marius Zeinhofer, Johannes Müller, Kamyar Azizzadenesheli, and Anima Anandkumar. Dynamical measure transport and neural PDE solvers for sampling. *arXiv preprint arXiv:2407.07873*, 2024.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry Vetrov. Generative flow networks as entropy-regularized RL. *arXiv preprint arXiv:2310.12934*, 2023.

Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken. Transport meets variational inference: Controlled Monte Carlo diffusions. *International Conference on Learning Representations (ICLR)*, 2024.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, et al. Amortizing intractable inference in diffusion models for vision, language, and control. *Neural Information Processing Systems (NeurIPS)*, 2024.

Alexandra Volokhova, Michał Koziarski, Alex Hernández-García, Cheng-Hao Liu, Santiago Miret, Pablo Lemos, Luca Thiede, Zichao Yan, Alán Aspuru-Guzik, and Yoshua Bengio. Towards equilibrium molecular conformation generation with gflownets. *Digital Discovery*, 3:1038–1047, 2024.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Neural Information Processing Systems (NeurIPS)*, 2020.

Dinghuai Zhang, Ricky T. Q. Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with GFlowNets and beyond. *arXiv preprint arXiv:2209.02606*, 2023.

Dinghuai Zhang, Ricky Tian Qi Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*, 2024.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *International Conference on Learning Representations (ICLR)*, 2023.

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Carnegie Mellon University, 2010.

A Additional related work

Classical sampling methods. The gold standard for sampling is often considered *Annealed Importance Sampling* (AIS) (Neal, 2001) and its *Sequential Monte Carlo* (SMC) extensions (Chopin, 2002; Del Moral et al., 2006). The former can be viewed as a special case of our discrete-time setting, where, however, the transition kernels are fixed and not learned, thus requiring careful tuning. For the kernels, often a form of *Markov Chain Monte Carlo* (MCMC), such Langevin dynamics and extensions (*e.g.*, ULA, MALA, and HMC) are considered. While they enjoy asymptotic convergence guarantees, they can suffer from slow mixing times, in particular for multimodal targets (Doucet et al., 2009; Kass et al., 1998; Dai et al., 2022). Alternatives are provided by variational methods that reformulate the sampling problem as an optimization problem, where a parametric family of tractable distributions is fitted to the target. This includes mean-field approximations (Wainwright et al., 2008) as well as normalizing flows (Papamakarios et al., 2021). We note that MCMC can also be interpreted as a variational approximation in an extended state space (Salimans et al., 2015).

Normalizing flows. There exist various versions of combining (continuous-time or discrete-time) normalizing flows with classical sampling methods, such as MCMC, AIS, and SMC (Wu et al., 2020; Arbel et al., 2021; Matthews et al., 2022). Most of these methods rely on the reverse KL divergence that suffers from mode collapse. To combat this issue, the underlying continuity equation (and Hamilton-Jacobi-Bellman equations in case of optimal transport) have been leveraged for the learning problem (Ruthotto et al., 2020; Máté & Fleuret, 2023; Sun et al., 2024). However, in all the above cases, one needs to either restrict model expressivity or rely on costly computations of divergences (in continuous time) or Jacobian determinants (in discrete time). Our Prop. 3.4 shows that, in the stochastic case, the discrepancy in the corresponding Fokker-Planck equation – an expression involving divergences and Laplacians – can be approximated by detailed balance divergences, which require no differentiation.

Diffusion-based samplers. Motivated by (annealed) Langevin dynamics and diffusion models, there is growing interest in the development of SDEs controlled by neural networks, also known as neural SDEs, for sampling. This covers methods based on *Schrödinger (Half-)bridges* (Zhang & Chen, 2022), diffusion models (Vargas et al., 2023; Berner et al., 2022), and annealed flows (Vargas et al., 2024). These methods can be interpreted as special cases of stochastic bridges, aiming at finding a time-reversal between two SDEs starting at the prior and target distributions (Vargas et al., 2024; Richter & Berner, 2024). In particular, this allows to consider general divergences between the associated measures on the SDE trajectories, such as the log-variance divergence (Richter et al., 2020; Nüsken & Richter, 2021). We note that there has also been some work on combining classical sampling methods with diffusion models (Phillips et al., 2024; Doucet et al., 2022).

GFlowNets. GFlowNets are originally defined in discrete space (Bengio et al., 2023), but were generalized to general measure spaces in (Lahlou et al., 2023), who proved the correctness of objectives in continuous time and experimented with using them to train diffusion models as samplers. However, the connection between GFlowNets and diffusion models had already been made informally by Malkin et al. (2023) for samplers of Boltzmann distributions and by Zhang et al. (2023) for maximum-likelihood training, and the latter showed a connection between detailed balance and sliced score matching, which has a similar flavor to our Prop. 3.4. GFlowNets are, in principle, more general than diffusion models with Gaussian noising, as the state space may change between time steps and the transition density does not need to be Gaussian, which has been taken advantage of in some applications (Volokhova et al., 2024; Phillips & Cipcigan, 2024).

Accelerated integrators for diffusion models. We remark that there has been great interest in developing accelerated sampling methods for diffusion models and the related continuous normalizing flows (*e.g.*, Shaul et al., 2024; Pandey et al., 2024). In particular, one can consider higher-order integrators for the associated *probability flow ODE* (Song et al., 2021b) or integrate parts of the SDE analytically (Zhang & Chen, 2023). However, we note that this research is concerned with accelerating *inference*, not training, of diffusion models and thus orthogonal to our research. For generative modeling, one has access to samples from the target distribution, allowing the use of simulation-free denoising score matching for training. For sampling problems without access to samples, diffusion-based methods, such as those outlined in the previous paragraphs, need to rely on

costly simulation-based objectives. However, our findings show that we can significantly accelerate these simulations during training with a negligible drop in inference-time performance.

B Theory details

B.1 Assumptions

Throughout the paper, we assume that all SDEs admit densities of their time marginals (w.r.t. the Lebesgue measure) that are sufficiently smooth such that we have strong solutions to the corresponding Fokker-Planck equations. In particular, we assume that[4] $p_{\text{prior}}, p_{\text{target}} \in C^\infty(\mathbb{R}^d, \mathbb{R}_{>0})$ are bounded. Furthermore, we assume that $\mu \in C^\infty([0,1] \times \mathbb{R}^d, \mathbb{R}^d)$ for all drifts μ, *i.e.*, they are infinitely differentiable, and satisfy a uniform (in time) Lipschitz condition, *i.e.*, there exists a constant C such that for all $x, y \in \mathbb{R}^d$ and $t \in [0,1]$ it holds that

$$\|\mu(x,t) - \mu(y,t)\| \leq C\|x - y\|. \tag{21}$$

Moreover, we assume that the diffusion rate satisfies that $\sigma \in C^\infty([0,1], \mathbb{R}_{>0})$. These conditions guarantee the existence of unique strong solutions to the considered SDEs. They are also sufficient for all considered path measures to be equivalent and for Girsanov's theorem and Nelson's relation to hold. Moreover, they allow the definition of the forward and backward Itô integrals via limits of time discretizations that are independent of the specific sequence of refinements (Vargas et al., 2024). While we use these assumptions to simplify the presentation, we note that they can be significantly relaxed.

B.2 Formal definition of the MDP

We elaborate the definition of the MDP in §2.1.

- The state space is
$$\mathcal{S} = \{\bullet\} \cup \bigcup_{n=0}^{N} \underbrace{\{(x, t_n) : x \in \mathbb{R}^d\}}_{\coloneqq \mathcal{S}_n} \cup \{\bot\}, \tag{22}$$
where $\bullet$ and $\bot$ are abstract initial and terminal states.
- The action space is $\mathcal{A} = \mathbb{R}^d$.
- The transition function $T \colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ describing the deterministic effect of actions is given by
$$T(\bullet, a) = (a, t_0), \quad T((x, t_n), a) = \begin{cases} (a, t_{n+1}) & n < N \\ \bot & n = N \end{cases}, \quad T(\bot, a) = \bot. \tag{23}$$
- The reward is nonzero only for transitions from states in $\mathcal{S}_N$ to $\bot$ and is given by $R(x, t_N) = -\mathcal{E}(x)$.

It is arguably more natural from a control theory perspective to treat the addition of (*e.g.*, Gaussian) noise as stochasticity of the environment, making the policy deterministic. However, we choose to formulate integration as a constrained stochastic policy in a deterministic environment to allow flexibility in the form of the conditional distribution. We also note that the policy at $\bot$ is irrelevant since $\bot$ is an absorbing state.

B.3 Numerical analysis

Definition B.1 (Strong convergence)**.** A numerical scheme $\widehat{X} = (\widehat{X}_n)_{n=0}^N$ is called *strongly convergent* of order γ if

$$\max_{n=0,\dots,N} \mathbb{E}\left[\|\widehat{X}_n - X_{t_n}\|\right] \leq C \left(\max_{n=0}^{N-1} \Delta t_n\right)^\gamma, \tag{24}$$

where $0 < C < \infty$ is independent of $N \in \mathbb{N}$ and the time discretization $0 = t_0 < t_1 < \cdots < t_N = 1$.

[4] Note that we also consider samplers using a Dirac delta prior, which can be treated by relaxing our conditions (Dai Pra, 1991). Under the policy given by (16), we can equivalently consider a (discrete-time) setting on the time interval $[t_1, 1]$ using a Gaussian prior with learned mean and variance $\sigma^2(t_0)\Delta t_0$.

Definition B.2 (Weak convergence)**.** A numerical scheme $\widehat{X} = (\widehat{X}_n)_{n=0}^N$ is called *weakly convergent* of order γ if

$$\max_{n=0,\dots,N} \left\| \mathbb{E}[f(\widehat{X}_n)] - \mathbb{E}[f(X_{t_n})] \right\| \leq C \left(\max_{n=0}^{N-1} \Delta t_n \right)^{\gamma} \tag{25}$$

for all functions f in a suitable test class, where we consider $f \in C^\infty(\mathbb{R}^d, \mathbb{R})$ with at most polynomially growing derivatives. The constant $0 < C < \infty$ is independent of $N \in \mathbb{N}$ and the time discretization $0 = t_0 < t_1 < \cdots < t_N = 1$, but may depend on the class of test functions considered.

Note that if f is globally Lipschitz, then strong convergence implies weak convergence. The converse does not hold.

Let us also consider a continuous version $\iota(\widehat{X})$ of the numerical scheme $\widehat{X} = (\widehat{X}_n)_{n=0}^N$ defined by $\iota(\widehat{X})_{t_n} = \widehat{X}_n$ and linearly interpolating between the t_n, where we note that ι implicitly depends on the discretization. We can then define the pushforward $\iota_* \widehat{\mathbb{P}}$ of the distribution $\widehat{\mathbb{P}}$ of $\widehat{X}$ on the space of continuous functions $C([0,1], \mathbb{R}^d)$. We say that $\iota_* \widehat{\mathbb{P}}$ *converges weakly* to the path measure $\mathbb{P}$ of X if for any bounded, continuous functional $f \colon C([0,1], \mathbb{R}^d) \to \mathbb{R}$ it holds that

$$\mathbb{E}_{X \sim \iota_* \widehat{\mathbb{P}}} [f(X)] \longrightarrow \mathbb{E}_{X \sim \mathbb{P}} [f(X)] \tag{26}$$

as $\max_n \Delta t_n \to 0$.

B.4 Subtrajectory balance

Generalizing trajectory balance (7) and detailed balance (8), we can define divergences for subtrajectories of any length k by multiplying the log-ratios appearing in (8) for several consecutive values of n, which through telescoping cancellation yields a *subtrajectory balance* divergence, defined for any $0 \leq n < n + k \leq N$ by

$$D^{\widehat{\mathbb{W}}}_{\mathrm{SubTB},n,n+k}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) = \mathbb{E}_{\widehat{X} \sim \widehat{\mathbb{W}}} \left[\log \left(\frac{\widehat{p}_n(\widehat{X}_n) \prod_{i=0}^{k-1} \overrightarrow{\pi}(\widehat{X}_{n+i+1} \mid \widehat{X}_{n+i})}{\widehat{p}_{n+k}(\widehat{X}_{n+k}) \prod_{i=0}^{k-1} \overleftarrow{\pi}(\widehat{X}_{n+i} \mid \widehat{X}_{n+i+1})} \right)^2 \right]. \tag{27}$$

The subtrajectory balance (SubTB) divergence generalizes detailed balance and trajectory balance, as one has

$$D^{\widehat{\mathbb{W}}}_{\mathrm{SubTB},n,n+1}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) = D^{n,\widehat{\mathbb{W}}}_{\mathrm{DB}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) \quad \text{and} \quad D^{\widehat{\mathbb{W}}}_{\mathrm{SubTB},0,N}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}, \widehat{p}) = D^{\widehat{\mathbb{W}}}_{\mathrm{TB}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}).$$

The SubTB divergence was introduced for GFlowNets by Malkin et al. (2022) and studied as a learning scheme, in which the divergences with different values of k are appropriately weighted, by Madan et al. (2023). SubTB was tested in the diffusion sampling case by Zhang et al. (2024), although Sendera et al. (2024) found that it is, in general, not more effective than TB while being substantially more computationally expensive.

B.5 Inductive bias on density estimates

We describe the inductive bias on density estimates used in the FL-DB learning objective. While normally one parametrizes the log-density as a neural network taking x and t as input:

$$\log \widehat{p}(x,t) = \mathrm{NN}_\theta(x,t),$$

the inductive bias proposed by Wu et al. (2020); Máté & Fleuret (2023) and studied earlier for GFlowNet diffusion samplers by Zhang et al. (2024); Sendera et al. (2024) writes

$$\log \widehat{p}(x,t) = -t\mathcal{E}(x) + (1-t) \log p_{\mathrm{ref}}(x) + \mathrm{NN}_\theta(x,t),$$

where $p_{\mathrm{ref}}(\cdot, t)$ is the marginal density at time t of the uncontrolled process, *i.e.*, the SDE (1) that sets $\overrightarrow{\mu} \equiv 0$ and has initial condition p_{prior}. Thus a correction is learned to an estimated log-density that interpolates between the prior at $t = 0$ and the target at $t = 1$.

The acronym 'FL-' stands for 'forward-looking', referring to the technique studied for GFlowNets by Pan et al. (2023) and understood as a form of reward-shaping scheme in Deleu et al. (2024).

B.6 Proofs of results from the main text

Proposition B.3 (Convergence of functionals)**.** *If $\mathbb{P}, \mathbb{Q}, \mathbb{W}$ are path measures of three forward-time SDEs, the drifts of the SDEs and their derivatives are bounded, and $f \in C^\infty(\mathbb{R}, \mathbb{R})$ has at most polynomially growing derivatives, then*

$$\mathbb{E}_{\widehat{X}\sim\widehat{\mathbb{W}}}\left[f\left(\log\frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X})\right)\right] \xrightarrow{\max_n \Delta t_n \to 0} \mathbb{E}_{X\sim\mathbb{W}}\left[f\left(\log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)\right)\right].$$

Proof of Prop. 3.2. As shown in the proof of Lemma B.7, $\log\frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(\widehat{X})$ can be viewed as a component of the Euler-Maruyama integration of an Itô process (with space-dependent diffusion) evaluated at time 1. Given our assumptions on the coefficient functions (see also Appendix B.1), the result follows by weak convergence; see, e.g., Kloeden & Platen (1992). □

Proposition B.4 (Asymptotic consistency of TB and VarGrad)**.** *Under the assumptions of Prop. 3.2, the divergences $D_{\mathrm{TB}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}})$ and $D_{\mathrm{LV}}^{\widehat{\mathbb{W}}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}})$ converge to $D_{\mathrm{TB}}^{\mathbb{W}}(\mathbb{P}, \mathbb{Q})$ and $D_{\mathrm{LV}}^{\mathbb{W}}(\mathbb{P}, \mathbb{Q})$, respectively.*

Proof of Prop. 3.3. Immediate from Prop. 3.2, taking $f(x) = x^2$ and $f(x) = x$. □

Proposition B.5 (Asymptotic equality of DB and FPE)**.** *Under the smoothness conditions in Lemma B.8, $\overrightarrow{\mu}, \overleftarrow{\mu}, \widehat{p}$ jointly satisfy Nelson's identity ($\overleftarrow{\mu} = \overrightarrow{\mu} - \sigma^2\nabla\log\widehat{p}$) at (x_t, t) if and only if*

$$\lim_{h\to 0}\left[\frac{1}{\sqrt{h}}\Delta_{t\to t+h}(x_t, x_{t+h})\right] = 0 \quad \textit{for almost every } z,$$

where $x_{t+h} \coloneqq x_t + \overrightarrow{\mu}(x_t, t)h + \sigma(t)\sqrt{h}z$. If in addition

$$\lim_{h\to 0}\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}\left[\frac{1}{h}\Delta_{t\to t+h}(x_t, x_{t+h})\right] = 0,$$

then the Fokker-Plank equation is satisfied at (x_t, t). If both conditions hold at all $(x_t, t) \in \mathbb{R}^d \times (0, 1)$, then $\overrightarrow{\mu}, \overleftarrow{\mu}$ define a pair of time-reversed processes with marginal density $\widehat{p}$.

Proof of Prop. 3.4. We write $\widehat{p}_t(x)$, $\overrightarrow{\mu}_t(x)$, σ_t for $\widehat{p}(x,t)$, $\overrightarrow{\mu}(x,t)$, $\sigma(t)$ for convenience. By Lemma B.8, the first condition implies that for almost all z,

$$\langle z, \sigma_t^2\nabla\log\widehat{p}_t(x_t) - (\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t))\rangle = 0, \tag{28}$$

which implies Nelson's identity at (x_t, t), while the second condition implies that

$$\partial_t\log\widehat{p}_t(x_t) + \left\langle\overrightarrow{\mu}_t(x_t), \nabla\log\widehat{p}_t(x_t)\right\rangle + \langle\nabla, \overleftarrow{\mu}_t(x_t)\rangle + \frac{\sigma_t^2}{2}\left(\Delta\log\widehat{p}_t(x_t) - \left\|\frac{\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)}{\sigma_t^2}\right\|^2\right) = 0. \tag{29}$$

Substituting the expression (28) into (29) and simplifying, we get

$$\partial_t\log\widehat{p}_t(x_t) = -\langle\nabla, \overrightarrow{\mu}_t(x_t)\rangle - \langle\overrightarrow{\mu}_t(x_t), \nabla\log\widehat{p}_t(x_t)\rangle + \frac{\sigma_t^2}{2}\left(\Delta\log\widehat{p}_t(x_t) + \|\nabla\log\widehat{p}_t(x_t)\|^2\right),$$

which gives exactly the logarithmic form of the Fokker-Planck equation. □

Proposition B.6 (DB and FPE for Brownian bridges)**.** *The results of Prop. 3.4 hold if $\sigma(t)$ is constant and $\overleftarrow{\pi}$ is the discrete-time reversal of the Euler-Maruyama discretization of the process*

$$p_{\mathrm{prior}}(x) = \mathcal{N}(x; 0, \sigma_0 I_d), \quad \mathrm{d}X_t = \sigma(t)\,\mathrm{d}W_t. \tag{20}$$

Proof of Prop. 3.5. Using the changes of variables $x \mapsto \sigma x$ followed $t \mapsto t - \sigma_0$, it suffices to show this for $\sigma_0 = 0, \sigma = 1$, making (20) a standard Brownian motion (the change of bounds for t is insubstantial as the conditions are local in time).

Let $\overleftarrow{\pi}$ be the backward policy as originally defined. The reverse drift is $\overleftarrow{\mu}(x,t) = \frac{x}{t}$, so we have

$$\overleftarrow{\pi}(x_t \mid x_{t+h}) = \mathcal{N}\left(x_t; \frac{t}{t+h}x_{t+h}, h\right).$$

Let $\overleftarrow{\pi}'$ be the discrete-time reversal of the forward-discretized Brownian motion. By elementary properties of Gaussians, we have

$$\overleftarrow{\pi}'(x_t \mid x_{t+h}) = \mathcal{N}\left(x_t; \frac{t}{t+h}x_{t+h}, \frac{t}{t+h}h\right).$$

Let $\Delta_{t\to t+h}(x_t, x_{t+h})$ and $\Delta'_{t\to t+h}(x_t, x_{t+h})$ be the discrepancies (18)) defined using $\overleftarrow{\pi}$ and $\overleftarrow{\pi}'$, respectively. We will show that replacing Δ by Δ' does not affect the asymptotics in Prop. 3.4.

We have

$$\begin{aligned}
\Delta_{t\to t+h}(x_t, x_{t+h}) - \Delta'_{t\to t+h}(x_t, x_{t+h}) &= \log \overleftarrow{\pi}'(x_t \mid x_{t+h}) - \log \overleftarrow{\pi}(x_t \mid x_{t+h}) \\
&= \frac{-1}{2}\left[d \log \frac{t}{t+h} + \left\|x_t - \frac{t}{t+h}x_{t+h}\right\|^2 \left(\frac{1}{\frac{t}{t+h}h} - \frac{1}{h}\right)\right] \\
&= \frac{-1}{2}\left[d \log\left(1 - \frac{h}{t+h}\right) + \frac{1}{t}\left\|x_t - x_{t+h} + \frac{h}{t+h}x_{t+h}\right\|^2\right].
\end{aligned}$$

Setting $x_{t+h} = x_t + \overrightarrow{\mu}_t(x_t)h + \sqrt{h}z$, the above becomes

$$\frac{-1}{2}\left[-\frac{h}{t}d + O(h^2) + \frac{1}{t}\left(h\|z\|^2 + O(h^{3/2})\right)\right].$$

For fixed z, the $\sqrt{h}$-order asymptotics of this expression vanish. In expectation over $z \sim \mathcal{N}(0, I_d)$, the h-order asymptotics vanish because $\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}\left[\|z\|^2\right] = d$. □

B.7 Technical lemmas

Lemma B.7 (Convergence of Radon-Nikodym derivatives). *(a) Let $\mathbb{P}^{(1)}$ and $\mathbb{P}^{(2)}$ be the path space measures defined by SDEs of the form (9) with initial conditions $p^{(1),(2)}_{\text{prior}}$ and drifts $\overrightarrow{\mu}^{(1),(2)}$. Let $\widehat{\mathbb{P}}^{(1),(2)}$ be the Euler-Maruyama-discretized measures with respect to a time discretization $(t_n)_{n=0}^N$. For $\mathbb{P}^{(2)}$-almost every $X \in C([0,1], \mathbb{R}^d)$, $\frac{\mathrm{d}\widehat{\mathbb{P}}^{(1)}}{\mathrm{d}\widehat{\mathbb{P}}^{(2)}}(X_{t_0,\ldots,t_N}) \to \frac{\mathrm{d}\mathbb{P}^{(1)}}{\mathrm{d}\mathbb{P}^{(2)}}(X)$ as $\max_n \Delta t_n \to 0$, where $X_{t_0,\ldots,t_N}$ is the restriction of X to the times $t_0, \ldots, t_N$.*

(b) The same is true for a path space measure $\mathbb{P}$ defined by a forward SDE with initial conditions and a measure $\mathbb{Q}$ defined by a reverse SDE with terminal conditions: if $\widehat{\mathbb{P}}$ and $\widehat{\mathbb{Q}}$ are the discrete-time processes given by Euler-Maruyama and reverse Euler-Maruyama integration, respectively, then for $\mathbb{Q}$-almost every $X \in C([0,1], \mathbb{R}^d)$, as $\max_n \Delta t_n \to 0$, $\frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(X_{t_0,\ldots,t_N}) \to \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X)$.

Proof. We first show (a). We have

$$\begin{aligned}
\log \frac{\mathrm{d}\widehat{\mathbb{P}}^{(1)}}{\mathrm{d}\widehat{\mathbb{P}}^{(2)}}(X_{t_0,\ldots,t_N}) &= \log \frac{p^{(1)}_{\text{prior}}(X_0)\prod_{n=0}^{N-1}\overrightarrow{\pi}_n(X_{t_{n+1}} \mid X_{t_n})}{p^{(2)}_{\text{prior}}(X_0)\prod_{n=0}^{N-1}\overrightarrow{\pi}_n(X_{t_{n+1}} \mid X_{t_n})} \\
&= \log \frac{p^{(1)}_{\text{prior}}(X_0)}{p^{(2)}_{\text{prior}}(X_0)} + \sum_{n=0}^{N-1} \log \frac{\mathcal{N}(X_{t_{n+1}}; X_{t_n} + \overrightarrow{\mu}^{(1)}(X_{t_n}, t_n)\Delta t_n, \sigma(t_n)^2\Delta t_n)}{\mathcal{N}(X_{t_{n+1}}; X_{t_n} + \overrightarrow{\mu}^{(2)}(X_{t_n}, t_n)\Delta t_n, \sigma(t_n)^2\Delta t_n)} \\
&= \log \frac{p^{(1)}_{\text{prior}}(X_0)}{p^{(2)}_{\text{prior}}(X_0)} + \sum_{n=0}^{N-1}\Bigg[-\frac{\|\overrightarrow{\mu}^{(1)}(X_{t_n}, t_n)\|^2 - \|\overrightarrow{\mu}^{(2)}(X_{t_n}, t_n)\|^2}{2\sigma(t_n)^2}\Delta t_n \\
&\qquad + \frac{\overrightarrow{\mu}^{(1)}(X_{t_n}, t_n) - \overrightarrow{\mu}^{(2)}(X_{t_n}, t_n)}{\sigma(t_n)^2} \cdot (X_{t_{n+1}} - X_{t_n})\Bigg]. \quad (30)
\end{aligned}$$

This is precisely the (Riemann) sum for the integral defining the continuous-time Radon-Nikodym derivative (13); by continuity and our assumptions in Appendix B.1, the sum approaches the integral as $\max_n \Delta t_n \to 0$.

We now show (b) assuming (a). Let $\mathbb{P}^0$ be the path measure defined by Gaussian $\mathcal{N}(0, I)$ initial conditions and drift 0 and $\widehat{\mathbb{P}}^0$ its discretization. Similarly, let $\mathbb{Q}^0$ be defined by Gaussian terminal conditions and zero reverse drift and let $\widehat{\mathbb{Q}}^0$ be its reverse-time discretization. By absolute continuity, we have

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(X) = \frac{\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{P}^0(X)}{\mathrm{d}\mathbb{Q}/\mathrm{d}\mathbb{Q}^0(X)} \frac{\mathrm{d}\mathbb{P}^0}{\mathrm{d}\mathbb{Q}^0}(X), \quad \frac{\mathrm{d}\widehat{\mathbb{P}}}{\mathrm{d}\widehat{\mathbb{Q}}}(X_{t_0,\dots,t_N}) = \frac{\mathrm{d}\widehat{\mathbb{P}}/\mathrm{d}\widehat{\mathbb{P}}^0(X_{t_0,\dots,t_N})}{\mathrm{d}\widehat{\mathbb{Q}}/\mathrm{d}\widehat{\mathbb{Q}}^0(X_{t_0,\dots,t_N})} \frac{\mathrm{d}\widehat{\mathbb{P}}^0}{\mathrm{d}\widehat{\mathbb{Q}}^0}(X_{t_0,\dots,t_N}).$$

By (a), $\mathrm{d}\widehat{\mathbb{P}}/\mathrm{d}\widehat{\mathbb{P}}^0(X_{t_0,\dots,t_N}) \to \mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{P}^0(X)$, and similarly for $\mathbb{Q}$. It remains to show that $\log \mathrm{d}\widehat{\mathbb{P}}^0/\mathrm{d}\widehat{\mathbb{Q}}^0(X_{t_0,\dots,t_N}) \to \log \mathrm{d}\mathbb{P}^0/\mathrm{d}\mathbb{Q}^0(X) = \log \mathcal{N}(X_0; 0, I) - \log \mathcal{N}(X_1; 0, I)$. Indeed, we have

$$\begin{aligned}
\log \mathrm{d}\widehat{\mathbb{P}}^0/\mathrm{d}\widehat{\mathbb{Q}}^0(X_{t_0,\dots,t_N}) &= \log \frac{\mathcal{N}(X_0; 0, I)}{\mathcal{N}(X_1; 0, I)} + \sum_{n=1}^{N} \log \frac{\mathcal{N}(X_{t_n}; X_{t_{n-1}}, \sigma(t_{n-1})\Delta t_{n-1})}{\mathcal{N}(X_{t_{n-1}}; X_{t_n}, \sigma(t_n)\Delta t_{n-1})} \\
&= \log \frac{\mathcal{N}(X_0; 0, I)}{\mathcal{N}(X_1; 0, I)} + \sum_{n=1}^{N} \left[\frac{\|X_{t_n} - X_{t_{n-1}}\|^2}{2\Delta t_{n-1}} \left(\frac{1}{\sigma(t_n)^2} - \frac{1}{\sigma(t_{n-1})^2} \right) \right. \\
&\qquad \left. + d \log \frac{\sigma(t_n)}{\sigma(t_{n-1})} \right] \\
&\xrightarrow{\text{a.s.}} \log \frac{\mathcal{N}(X_0; 0, I)}{\mathcal{N}(X_1; 0, I)} + d \log \frac{\sigma(1)}{\sigma(0)} + \int_0^1 \underbrace{\frac{d\sigma(t)^2}{2} \mathrm{d}\sigma(t)^{-2}}_{=-\mathrm{d}(d \log \sigma(t))} \\
&= \log \frac{\mathcal{N}(X_0; 0, I)}{\mathcal{N}(X_1; 0, I)}. \qquad (31)
\end{aligned}$$

which coincides with the continuous-time Radon-Nikodym derivative. □

Lemma B.8 (Continuous-time asymptotics of the DB discrepancy)**.** *Let us define the abbreviations $\widehat{p}_t(x)$, $\overrightarrow{\mu}_t(x)$, σ_t to refer to $\widehat{p}(x,t)$, $\overrightarrow{\mu}(x,t)$, $\sigma(t)$. Suppose that $\overleftarrow{\mu}_t$ and $\overleftarrow{\mu}_t$ are continuously differentiable in x and once in t and that $\log \widehat{p}_t$ is continuously differentiable once in t and twice in x.*

(a) For a given z, the asymptotics of the DB discrepancy at (x_t, t) are of order $\sqrt{h}$ and are given by

$$\lim_{h\to 0} \left[\frac{1}{\sqrt{h}} \Delta_{t\to t+h}(x_t, x_t + \overrightarrow{\mu}_t(x_t)h + \sigma_t z) \right] = \sigma_t^{-1} \langle z, \sigma_t^2 \nabla \log \widehat{p}_t(x_t) - (\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)) \rangle.$$

*(b) The expectation of the DB discrepancy over the forward policy (*i.e.*, over $z \sim \mathcal{N}(0, I)$) is asymptotically of order h, with leading term*

$$\begin{aligned}
&\lim_{h\to 0} \mathbb{E}_{x_{t+h} \sim \overrightarrow{\pi}(x_{t+h}|x_t)} \left[\frac{1}{h} \Delta_{t\to t+h}(x_t, x_{t+h}) \right] \\
&\quad = \partial_t \log \widehat{p}_t(x_t) + \left\langle \overrightarrow{\mu}_t(x_t), \nabla \log \widehat{p}_t(x_t) \right\rangle + \langle \nabla, \overleftarrow{\mu}_t(x_t) \rangle \\
&\qquad + \frac{\sigma_t^2}{2} \left(\Delta \log \widehat{p}_t(x_t) - \left\| \frac{\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)}{\sigma_t^2} \right\|^2 \right).
\end{aligned}$$

Similarly, the expectation over the backward policy is

$$\begin{aligned}
&\lim_{h\to 0} \mathbb{E}_{x_{t-h} \sim \overleftarrow{\pi}(x_{t-h}|x_t)} \left[\frac{1}{h} \Delta_{t-h\to t}(x_{t-h}, x_t) \right] \\
&\quad = \partial_t \log \widehat{p}_t(x_t) + \left\langle \overleftarrow{\mu}_t(x_t), \nabla \log \widehat{p}_t(x_t) \right\rangle + \langle \nabla, \overrightarrow{\mu}_t(x_t) \rangle \\
&\qquad - \frac{\sigma_t^2}{2} \left(\Delta \log \widehat{p}_t(x_t) - \left\| \frac{\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)}{\sigma_t^2} \right\|^2 \right).
\end{aligned}$$

Proof. We will simultaneously show (a) and the first part of (b). The second part of (b) is symmetric, by reversing time.

Identifying x_{t+h} with $x_t + \overrightarrow{\mu}_t(x_t)h + \sigma_t\sqrt{h}z$, we will analyze the leading asymptotics of the DB discrepancy, i.e., $\Delta_{t\to t+h}(x_t, x_{t+h}) = \sqrt{h}\langle z, \dots\rangle + h(\dots) + O(h^{3/2})$. The coefficient of $\sqrt{h}$ will be the scalar product of z with a term that is independent of z and equals the expression on the right side in (a), and thus vanishes in expectation over z. The coefficient of h, in expectation over z, will equal the expression on the right side in (b).

We can show using Taylor expansions that

$$\log\frac{\widehat{p}_{t+h}(x_{t+h})}{\widehat{p}_t(x_t)} = \sqrt{h}\,\langle z, \sigma_t\nabla\log\widehat{p}_t(x_t)\rangle + h\left[\partial_t\log\widehat{p}_t(x_t) + \left\langle\overrightarrow{\mu}_t(x_t), \nabla\log\widehat{p}_t(x_t)\right\rangle + \frac{1}{2}\sigma_t^2\langle z, \nabla^2\log\widehat{p}_t(x_t)z\rangle\right] + O(h^{3/2}). \tag{32}$$

Now we are going to analyze the second part of (18), which involves the policies. We have

$$\begin{aligned}
&\log\frac{\overleftarrow{\pi}(x_t \mid x_{t+h})}{\overrightarrow{\pi}(x_{t+h} \mid x_t)}\\
&= \frac{-1}{2}\left[\frac{\|x_t - x_{t+h} + \overleftarrow{\mu}_{t+h}(x_{t+h})h\|^2}{\sigma_{t+h}^2 h} - \frac{\|x_{t+h} - x_t - \overrightarrow{\mu}_t(x_t)h\|^2}{\sigma_t^2 h} + d\log\frac{2\pi\sigma_{t+h}^2}{2\pi\sigma_t^2}\right]\\
&= \frac{-1}{2}\left[\frac{\|x_t - x_{t+h} + \overleftarrow{\mu}_{t+h}(x_{t+h})h\|^2}{\sigma_{t+h}^2 h}\right] + \frac{\|\sigma_t\sqrt{h}z\|^2}{2\sigma_t^2 h} - d\log\frac{\sigma_{t+h}}{\sigma_t}.\\
&= \frac{-1}{2}\left[\frac{\|x_t - x_{t+h} + \overleftarrow{\mu}_{t+h}(x_{t+h})h\|^2}{\sigma_{t+h}^2 h}\right] + \frac{\|z\|^2}{2} - dh\partial_t(\log\sigma_t) + O(h^2).
\end{aligned}$$

Next we will write

$$\begin{aligned}
x_t - x_{t+h} + \overleftarrow{\mu}_{t+h}(x_{t+h})h &= x_t - x_{t+h} + \overrightarrow{\mu}_t(x_t)h - (\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_{t+h}(x_{t+h}))h\\
&= -\sigma_t\sqrt{h}z - (\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_{t+h}(x_{t+h}))h
\end{aligned}$$

and substitute this into the first term above, yielding

$$\begin{aligned}
&\frac{-1}{2}\left[\frac{\|x_t - x_{t+h} + \overleftarrow{\mu}_{t+h}(x_{t+h})h\|^2}{\sigma_{t+h}^2 h}\right] + \frac{\|z\|^2}{2} - dh\partial_t(\log\sigma_t) + O(h^2)\\
&= \frac{-1}{2}\left[\frac{\|-\sigma_t\sqrt{h}z - (\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_{t+h}(x_{t+h}))h\|^2}{\sigma_{t+h}^2 h}\right] + \frac{\|z\|^2}{2} - dh\partial_t(\log\sigma_t) + O(h^2)\\
&= -\frac{\|\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_{t+h}(x_{t+h})\|^2}{2\sigma_{t+h}^2}h - \frac{\langle\sigma_t z, \overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_{t+h}(x_{t+h})\rangle}{\sigma_{t+h}^2}\sqrt{h}\\
&\quad - \frac{\sigma_t^2\|z\|^2}{2\sigma_{t+h}^2} + \frac{\|z\|^2}{2} - dh\partial_t(\log\sigma_t) + O(h^2)\\
&= \sqrt{h}\left[\left\langle z, -\sigma_t^{-1}(\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t))\right\rangle + \frac{\sigma_t\langle z, \sigma_t\sqrt{h}\nabla\overleftarrow{\mu}_t(x_t)z\rangle}{\sigma_{t+h}^2}\right]\\
&\quad + h\left[-\frac{\|\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)\|^2}{2\sigma_t^2} - d\partial_t(\log\sigma_t)\right] + \frac{\|z\|^2}{2}\underbrace{\left(1 - \frac{\sigma_t^2}{\sigma_{t+h}^2}\right)}_{=2\partial_t(\log\sigma_t)h + O(h^2)} + O(h^{3/2})\\
&= \sqrt{h}\left\langle z, -\sigma_t^{-1}(\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t))\right\rangle\\
&\quad + h\left[-\frac{\|\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)\|^2}{2\sigma_t^2} + \langle z, \nabla\overleftarrow{\mu}_t(x_t)z\rangle - \left(\|z\|^2 - d\right)\partial_t(\log\sigma_t)\right] + O(h^{3/2}).
\end{aligned}$$

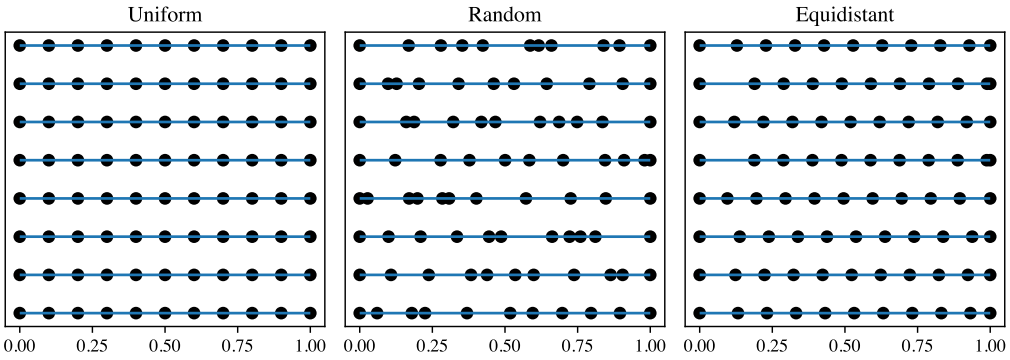


Figure 7: Sampled 10-step discretizations of the unit interval using the three schemes considered.

Combining with the terms in (32), we get that the coefficient of $\sqrt{h}$ is exactly as desired. For the coefficient of h, and the terms of the form $\langle z, \dots \rangle$ and $\|z\|^2 - d$ vanish in expectation over z. For the terms that are quadratic in z, Hutchinson's formula implies that

$$\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\langle z, \nabla\overleftarrow{\mu}_t(x_t)z\rangle\right] = \langle\nabla, \overleftarrow{\mu}_t(x_t)\rangle,$$
$$\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\langle z, \nabla^2 \log\widehat{p}_t(x_t)z\rangle\right] = \Delta\log\widehat{p}_t(x_t).$$

Putting these identities together, we obtain that

$$\lim_{h\to 0}\mathbb{E}_{x_{t+h}\sim\overrightarrow{\pi}(x_{t+h}|x_t)}\left[\frac{1}{h}\Delta_{t\to t+h}(x_t, x_{t+h})\right]$$
$$= \partial_t\log\widehat{p}_t(x_t) + \left\langle\overrightarrow{\mu}_t(x_t), \nabla\log\widehat{p}_t(x_t)\right\rangle + \frac{1}{2}\sigma_t^2\Delta\log\widehat{p}_t(x_t) - \frac{\|\overrightarrow{\mu}_t(x_t) - \overleftarrow{\mu}_t(x_t)\|^2}{2\sigma_t^2} + \langle\nabla, \overleftarrow{\mu}_t(x_t)\rangle,$$

which is equivalent to the expression in (b). □

C Experiment details

C.1 Training settings

All models are trained for 25,000 steps using settings identical to those suggested by Sendera et al. (2024) (https://github.com/GFNOrg/gfn-diffusion). For DB, we use the same learning rates as for SubTB (10^{-3} for the drift and 10^{-2} for the flow function), and for PIS, 10^{-3} or 10^{-4} depending on its stability in the specific case.

Training times are measured by wall time of execution on a large shared cluster, primarily on RTX8000 GPUs. Although all runs were assigned by the same job scheduler, some variability in results is inevitable due to inconsistent hardware.

C.2 Discretization schemes

In this section, we define the two nonuniform discretization schemes used in the experiments (see Fig. 7 for illustration):

- **Random:** We sample i.i.d. numbers $z_0, \dots, z_{N_{\text{train}}-1} \sim \mathcal{U}([1, c])$, where c is a sufficiently large constant (we take $c = 10$). We then define

$$\Delta t_i = \frac{z_i}{\sum_{j=0}^{N_{\text{train}}-1} z_j}, \quad t_i = \sum_{j=0}^{i-1}\Delta t_j.$$

  Thus, the interval lengths sum to 1, and no two have a ratio of lengths greater than c. (Note that we also tested setting the t_i ($0 < i < N_{\text{train}}$) to be i.i.d. random values sampled from $\mathcal{U}([0, 1])$ sorted in increasing order, but this caused numerical instability when very short intervals were present.)

- **Equidistant:** We sample $t_1 \sim \mathcal{U}([\epsilon, 2/N_{\text{train}} - \epsilon])$, where for us $\epsilon = 10^{-4}$, then set

$$t_i = t_1 + \frac{i-1}{N_{\text{train}}}$$

for $i = 1, \ldots, N_{\text{train}} - 1$. Thus $\Delta t_i = \frac{1}{N_{\text{train}}}$ for all $1 < i < N_{\text{train}} - 1$, *i.e.*, all intervals are of equal length except possibly the first and last.

D Additional results

D.1 Additional metrics and objectives

In Table 2, we show extended results on the four unconditional sampling benchmarks from Sendera et al. (2024), reporting the ELBO $\log \widehat{Z}$ and importance-weighted ELBO $\log \widehat{Z}^{\text{RW}}$. Specifically, the two are computed as

$$\begin{aligned}
\log \widehat{Z} &\coloneqq \frac{1}{K}\sum_{i=1}^{K}\left[-\mathcal{E}(\widehat{X}_N^{(i)}) + \log \frac{\widehat{\mathbb{Q}}(\widehat{X}^{(i)} \mid \widehat{X}_N^{(i)})}{\widehat{\mathbb{P}}(\widehat{X}^{(i)})}\right] = \log Z + \frac{1}{K}\sum_{i=1}^{K}\left[\log \frac{\widehat{\mathbb{Q}}(\widehat{X}^{(i)})}{\widehat{\mathbb{P}}(\widehat{X}^{(i)})}\right], \\
\log \widehat{Z}^{\text{RW}} &\coloneqq \log \frac{1}{K}\sum_{i=1}^{K}\exp\left[-\mathcal{E}(\widehat{X}_N^{(i)}) + \log \frac{\widehat{\mathbb{Q}}(\widehat{X}^{(i)} \mid \widehat{X}_N^{(i)})}{\widehat{\mathbb{P}}(\widehat{X}^{(i)})}\right] = \log Z + \log \frac{1}{K}\sum_{i=1}^{K}\left[\frac{\widehat{\mathbb{Q}}(\widehat{X}^{(i)})}{\widehat{\mathbb{P}}(\widehat{X}^{(i)})}\right],
\end{aligned} \tag{33}$$

where $\widehat{X}^{(1)}, \ldots \widehat{X}^{(K)} \sim \widehat{\mathbb{P}}$ and we note that $\mathbb{E}[\log \widehat{Z}] = \log Z - D_{\text{KL}}(\widehat{\mathbb{P}}, \widehat{\mathbb{Q}}) \leq \log Z$ and $\mathbb{E}[Z^{\text{RW}}] = Z$. We take $K = 2000$ samples and report the difference between the ground truth $\log Z$ and the ELBO when $\log Z$ is known.

These results are consistent with the conclusions in the main text. Notably, when combined with local search, coarse nonuniform discretizations continue to show results comparable to those of 100-step training discretization in most cases. Table 3 shows results on two additional target energies and on the conditional VAE task.

Table 2: ELBOs and IS-ELBOs on **25GMM**, **Funnel**, and **Manywell** (absolute error from the true value).

25GMM ($d = 2$)

Training discretization →	10-step random				10-step equidistant				10-step uniform				100-step uniform	
Evaluation steps →	10		100		10		100		10		100		100	
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$
PIS	2.40±0.10	1.02±0.09	1.56±0.10	0.93±0.16	2.39±0.11	0.97±0.10	1.51±0.09	1.01±0.09	2.43±0.12	0.85±0.58	5.62±0.32	1.03±0.14	1.65±0.30	1.12±0.20
TB	2.10±0.05	1.02±0.05	1.23±0.03	1.03±0.03	2.10±0.04	0.96±0.14	1.22±0.03	1.04±0.03	2.10±0.03	0.99±0.11	8.77±0.69	1.02±0.96	1.13±0.01	1.02±0.01
TB + LS	1.71±0.06	0.02±0.17	0.47±0.06	**0.002±0.04**	1.71±0.04	0.16±0.07	0.42±0.03	0.03±0.02	1.67±0.06	0.05±0.02	10.38±2.78	1.87±0.77	0.16±0.01	**0.0004**±0.01
VarGrad	2.12±0.04	1.04±0.04	1.22±0.01	1.04±0.01	2.09±0.03	1.04±0.01	1.19±0.03	1.03±0.01	2.12±0.02	1.02±0.04	9.13±0.87	0.92±1.19	1.12±0.01	1.02±0.01
VarGrad + LS	1.68±0.07	0.04±0.09	0.37±0.06	0.02±0.02	1.67±0.01	0.07±0.07	**0.33±0.07**	0.02±0.01	1.62±0.04	0.06±0.07	8.25±0.95	1.11±0.24	**0.15**±0.004	0.01±0.01
PIS + LP	2.80±0.07	1.02±0.17	1.98±0.06	0.10±0.42	2.77±0.10	1.00±0.21	1.94±0.03	0.05±0.30	2.77±0.08	1.00±0.20	3.49±0.08	0.14±1.24	1.76±0.02	0.43±0.45
TB + LP	1.57±0.05	0.03±0.18	0.32±0.02	0.02±0.05	1.56±0.03	0.01±0.16	0.36±0.06	0.03±0.03	2.70±2.33	0.11±0.33	5.30±0.80	0.43±0.47	0.16±0.01	0.01±0.01
TB + LS + LP	1.78±0.10	0.02±0.08	0.41±0.06	0.02±0.04	1.82±0.01	0.08±0.06	0.43±0.05	0.07±0.08	1.68±0.09	0.05±0.02	8.37±1.50	1.50±0.46	0.16±0.01	0.01±0.01
VarGrad + LP	1.59±0.04	0.03±0.08	0.35±0.06	**0.01±0.02**	1.46±0.005	0.07±0.06	0.32±0.04	0.04±0.01	1.53±0.01	0.01±0.01	5.52±0.80	0.53±0.54	**0.15**±0.01	**0.003**±0.01
VarGrad + LS + LP	1.68±0.09	0.02±0.08	0.26±0.02	**0.01±0.01**	1.69±0.05	0.07±0.06	**0.24±0.01**	**0.01±0.01**	1.64±0.06	0.04±0.07	7.07±1.50	0.90±0.85	0.16±0.01	0.01±0.005

Funnel ($d = 10$)

Training discretization →	10-step random				10-step equidistant				10-step uniform				100-step uniform	
Evaluation steps →	10		100		10		100		10		100		100	
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$
PIS	1.11±0.01	0.59±0.03	**0.72±0.02**	0.09±0.50	1.11±0.01	0.59±0.03	**0.72±0.02**	**0.02±0.58**	1.11±0.01	0.58±0.02	8.63±4.20	1.65±0.74	**0.52**±0.01	0.08±0.54
TB	1.09±0.02	0.51±0.04	0.76±0.02	0.48±0.04	1.09±0.02	0.47±0.10	0.74±0.01	0.45±0.03	1.07±0.01	0.42±0.11	10.86±5.22	2.29±1.35	0.54±0.01	0.26±0.06
TB + LS	1.46±0.02	0.66±0.03	1.13±0.03	0.40±0.02	1.40±0.09	0.62±0.08	1.11±0.18	0.46±0.09	1.41±0.02	0.62±0.07	268.47±327.21	29.70±46.66	1.01±0.03	0.36±0.04
VarGrad	1.09±0.02	0.50±0.05	0.76±0.02	0.42±0.05	1.11±0.01	0.36±0.24	0.76±0.01	0.46±0.06	1.07±0.02	0.46±0.04	9.97±4.49	2.41±1.20	0.53±0.01	0.17±0.18
VarGrad + LS	1.68±0.11	0.65±0.04	1.48±0.21	0.37±0.16	1.58±0.07	0.32±0.22	1.28±0.02	0.45±0.06	1.51±0.06	0.59±0.02	78.04±90.93	3.93±6.23	1.11±0.05	**0.02**±0.56
PIS + LP	1.11±0.01	0.56±0.07	0.71±0.01	**0.28±0.09**	1.10±0.01	0.56±0.04	**0.69±0.02**	0.29±0.05	1.10±0.02	0.57±0.02	8.85±2.48	1.80±0.74	0.50±0.03	**0.13**±0.17
TB + LP	1.08±0.02	0.40±0.12	0.72±0.03	0.37±0.03	1.54±0.51	0.50±0.12	0.91±0.21	0.44±0.11	1.07±0.02	0.38±0.11	30.07±22.61	9.56±13.27	**0.48**±0.005	0.25±0.03
TB + LS + LP	1.30±0.02	0.46±0.05	0.90±0.04	0.30±0.05	1.27±0.01	0.45±0.09	0.86±0.04	0.32±0.03	1.26±0.03	0.43±0.03	149.16±187.71	14.23±19.54	0.82±0.04	0.25±0.09
VarGrad + LP	1.08±0.02	0.46±0.17	0.72±0.02	0.37±0.02	1.10±0.01	0.43±0.08	0.74±0.02	0.38±0.04	1.07±0.01	0.43±0.13	48.10±42.22	21.80±30.37	**0.48**±0.01	0.23±0.04
VarGrad + LS + LP	1.39±0.04	0.46±0.04	0.99±0.05	0.33±0.03	1.44±0.04	0.44±0.08	1.09±0.18	0.36±0.06	1.32±0.05	0.44±0.04	162.54±189.06	10.68±12.41	0.77±0.07	0.25±0.05

Manywell ($d = 32$)

Training discretization →	10-step random				10-step uniform				100-step uniform	
Evaluation steps →	10		100		10		100		100	
Algorithm ↓ Metric →	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$	$\Delta \log Z$	$\Delta \log Z^{\mathrm{RW}}$
PIS (lr = 10^{-3})	14.08±0.14	2.70±0.30	**4.74±0.15**	2.77±0.05	14.08±0.13	2.97±0.37	69.72±13.41	33.84±11.79	**3.87±0.03**	2.69±0.03
PIS (lr = 10^{-4})	14.34±0.28	3.23±0.54	6.37±0.08	2.80±0.20	14.16±0.27	2.86±0.73	75.30±1.89	35.65±1.45	4.17±0.04	2.62±0.06
TB	14.96±0.22	2.92±1.10	5.49±0.43	2.70±0.11	14.81±0.17	2.55±2.05	62.95±10.12	30.07±5.79	4.05±0.05	2.75±0.01
TB + LS	15.24±0.62	1.54±0.77	7.24±0.46	**0.55±0.43**	14.86±0.60	0.45±0.89	51.08±4.27	16.82±3.08	4.52±0.91	**0.37±0.14**
VarGrad	14.94±0.28	2.79±1.35	5.64±0.56	2.77±0.05	14.80±0.14	2.86±1.61	71.71±18.54	35.53±11.51	4.04±0.11	2.78±0.04
VarGrad + LS	16.02±0.26	2.84±0.15	7.03±0.56	2.00±0.46	16.08±0.75	3.26±1.10	69.14±12.35	28.45±13.46	6.53±3.56	4.43±2.70
PIS + LP (lr = 10^{-3})	13.97±0.18	2.15±0.28	4.34±0.25	1.69±0.41	*diverging*				3.60±0.06	1.37±0.22
PIS + LP (lr = 10^{-4})	31.98±0.09	4.46±3.45	17.55±0.26	1.39±0.64	31.87±0.21	5.26±3.39	35.96±0.34	8.42±1.61	14.71±0.07	0.50±0.75
TB + LP	14.87±0.36	3.02±1.23	4.72±0.27	2.66±0.03	14.62±0.21	3.27±1.19	19.66±1.49	4.20±0.63	3.66±0.25	2.42±0.32
TB + LS + LP	13.88±0.58	0.60±0.23	**2.40±0.39**	**0.00±0.20**	13.67±0.44	0.81±0.51	24.32±1.02	2.10±0.43	1.81±0.05	**0.03±0.07**
VarGrad + LP	14.79±0.39	3.11±1.11	4.68±0.34	2.71±0.03	14.63±0.20	3.15±0.02	20.72±3.32	3.89±0.72	3.41±0.10	2.09±0.27
VarGrad + LS + LP	16.24±0.70	1.31±0.75	5.12±0.68	0.32±0.21	14.22±0.22	0.35±0.08	22.89±4.12	1.71±1.87	**1.77±0.06**	0.05±0.06

Table 3: ELBOs with different numbers of training and integration steps on **Credit**, **Cancer**, the conditional **VAE**, and **LGCP**. Training on **LGCP** was often unstable, consistent with findings of prior work, so fewer methods are reported.

Credit ($d = 25$)

Training discretization →	10-step random		10-step equidistant		10-step uniform		100-step uniform
Algorithm ↓ Evaluation steps →	10	100	10	100	10	100	100
PIS	-1174.23±14.07	-671.68±8.14	-1181.62±17.17	**-667.03±21.25**	-1171.35±14.59	-1130.57±20.69	**-606.61±0.65**
TB	-1301.50±9.68	-911.04±16.74	-1318.14±22.13	-898.98±24.18	-1281.31±9.74	-1179.87±30.61	-634.08±2.88
VarGrad	-1279.95±14.36	-847.65±22.65	-1288.40±10.49	-838.67±14.12	-1264.02±15.67	-1172.46±32.20	-631.84±3.20
PIS + LP	-1175.46±14.14	-671.60±12.01	-1183.60±17.90	**-669.30±16.34**	-1174.25±17.00	-1114.56±43.56	**-608.29±2.12**
TB + LP	-1342.96±6.77	-943.63±18.37	-1360.68±32.84	-956.97±4.12	-1300.17±8.29	-1165.11±25.76	-666.49±2.79
VarGrad + LP	-1303.67±15.11	-876.12±10.70	-1323.16±3.03	-933.40±50.79	-1281.15±6.49	-1186.95±150.69	-651.98±0.18

Cancer ($d = 31$)

Training discretization →	10-step random		10-step equidistant		10-step uniform		100-step uniform
Algorithm ↓ Evaluation steps →	10	100	10	100	10	100	100
PIS	-6.60±1.60	**9.51±3.13**	-7.73±0.63	9.15±1.45	-8.94±4.87	-4933.64±986.02	**17.64±12.51**
TB	-48.57±23.39	-28.02±18.77	-59.77±45.25	-29.81±18.81	-35.42±8.76	-1096.80±530.21	5.32±6.03
VarGrad	-28.97±6.03	-5.84±0.98	-31.83±2.58	-11.76±5.90	-30.09±3.76	-966.70±357.24	9.41±1.77
PIS + LP	-12.27±2.99	**7.30±1.92**	-16.87±3.26	6.35±2.27	-11.51±1.76	-3649.25±629.76	**19.47±1.87**
TB + LP	-25.79±3.04	-4.33±2.77	-41.52±28.79	-12.60±16.39	-24.33±1.48	-2738.75±344.22	11.56±0.59
VarGrad + LP	-30.55±0.14	-1.69±1.94	-28.16±4.40	-6.05±4.59	-26.36±1.95	-978.60±140.28	13.41±2.19

VAE ($d = 20$)

Training discretization →	10-step random		10-step equidistant		10-step uniform		100-step uniform
Algorithm ↓ Evaluation steps →	10	100	10	100	10	100	100
PIS	-117.83±1.25	-104.52±0.36	-117.68±1.29	**-104.29±0.58**	-117.74±1.12	-154.88±6.51	**-102.71±0.52**
TB	-161.97±1.26	-149.86±4.93	-162.72±4.85	-149.76±0.75	-160.49±0.56	-161.90±5.63	-142.88±5.14
VarGrad	-122.04±1.62	-109.45±1.40	-170.51±4.78	-159.71±7.46	-120.98±0.96	-133.39±4.98	-104.16±0.67
PIS + LP	-115.90±0.64	-100.20±0.33	-115.81±0.31	**-100.13±0.06**	-115.83±0.82	-120.61±1.41	-99.34±0.40
TB + LP	-140.41±2.18	-114.80±1.07	-140.72±1.10	-114.81±1.39	-137.54±2.51	-136.64±2.96	-109.25±1.68
VarGrad + LP	-118.52±1.47	-102.24±0.27	-138.51±0.70	-113.49±1.39	-117.35±0.99	-122.22±0.70	**-99.01±0.27**

LGCP ($d = 1600$)

Training discretization →	10-step random		10-step uniform		100-step uniform
Algorithm ↓ Evaluation steps →	10	100	10	100	100
PIS	-1471.16±6.83	**-1467.85±2.59**	-1471.49±11.66	-1729.56±103.09	**-1465.14±20.76**
TB	-1618.86±3.01	-1617.35±1.34	-1617.33±6.54	-1666.37±13.78	-1619.89±6.56
TB + LS	-1878.87±23.04	-1880.52±13.07	-1877.13±18.69	-1705.60±36.86	-1891.62±4.77
PIS + LP	343.46±0.31	472.24±0.68	343.18±0.33	-211.79±293.49	**473.74±1.14**
TB + LP	332.16±0.42	461.53±1.16	337.37±0.12	-1931.42±2636.38	468.68±4.13
TB + LS + LP	341.53±0.36	**472.43±0.42**	341.65±0.16	-77.64±77.72	451.89±3.28

D.2 ADDITIONAL FIGURES

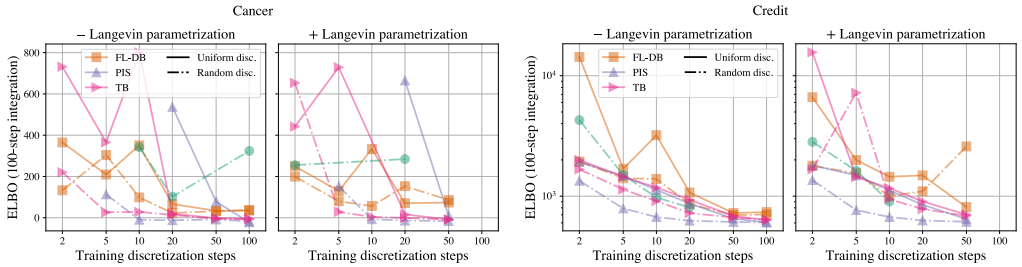


Figure 8: Results extending main text Fig. 4: Credit and Cancer densities.

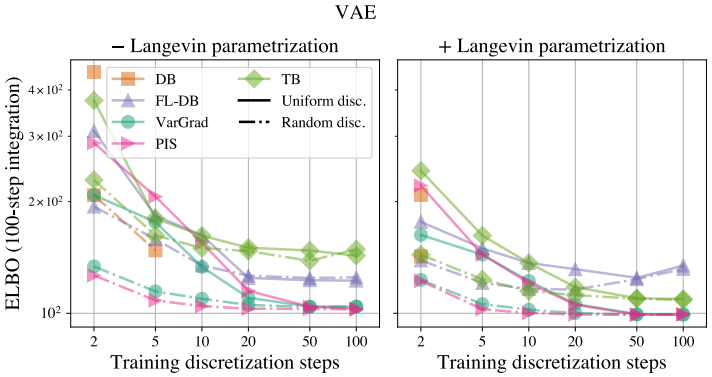


Figure 9: Results extending main text Fig. 4 on the conditional VAE target density.

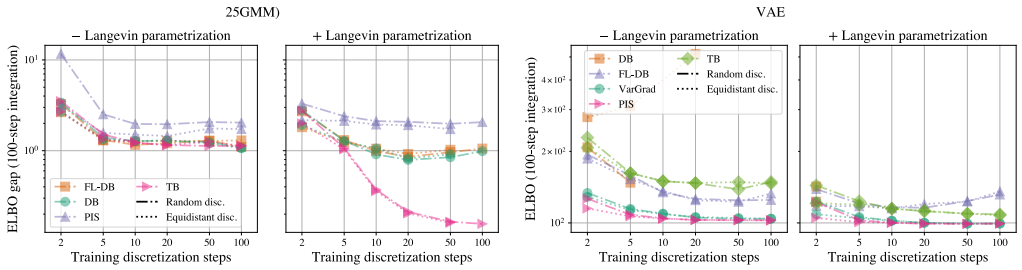


Figure 10: Comparison of **Random** and **Equidistant** distretizations on the **25GMM** (unconditional) and **VAE** (conditional) targets.

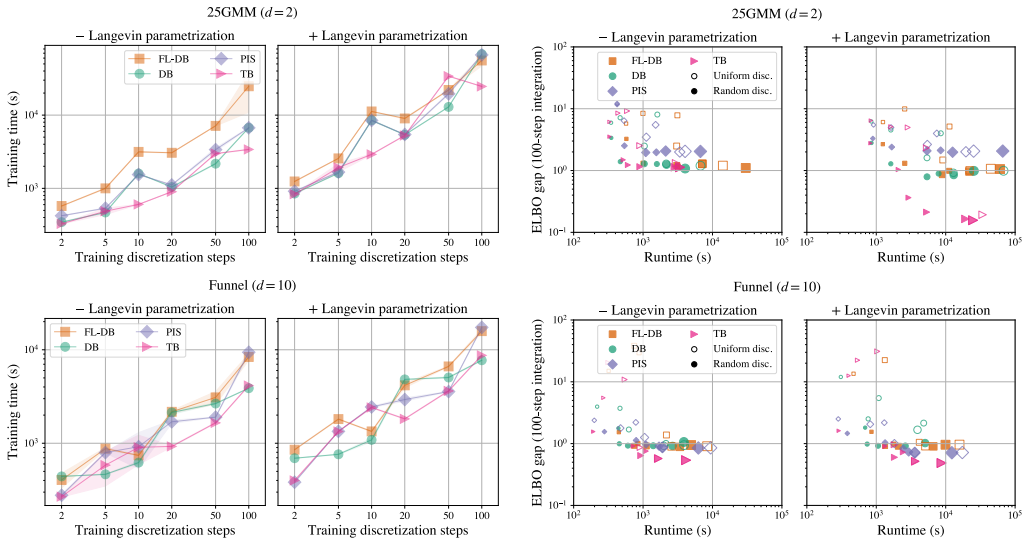


Figure 11: Results extending main text Fig. 5: Efficiency of nonuniform coarse discretizations on Funnel and 25GMM densities.

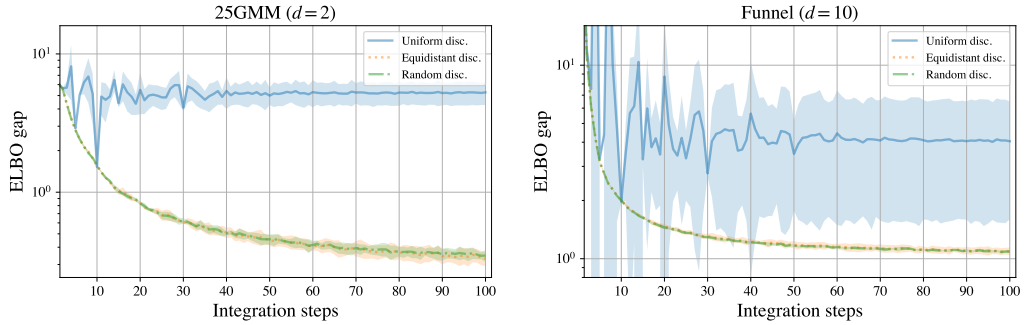


Figure 12: Results extending main text Fig. 6. Evaluation of models trained with $N_{\text{train}} = 10$ steps using varying numbers of integration steps.

D.3 VAE RECONSTRUCTIONS

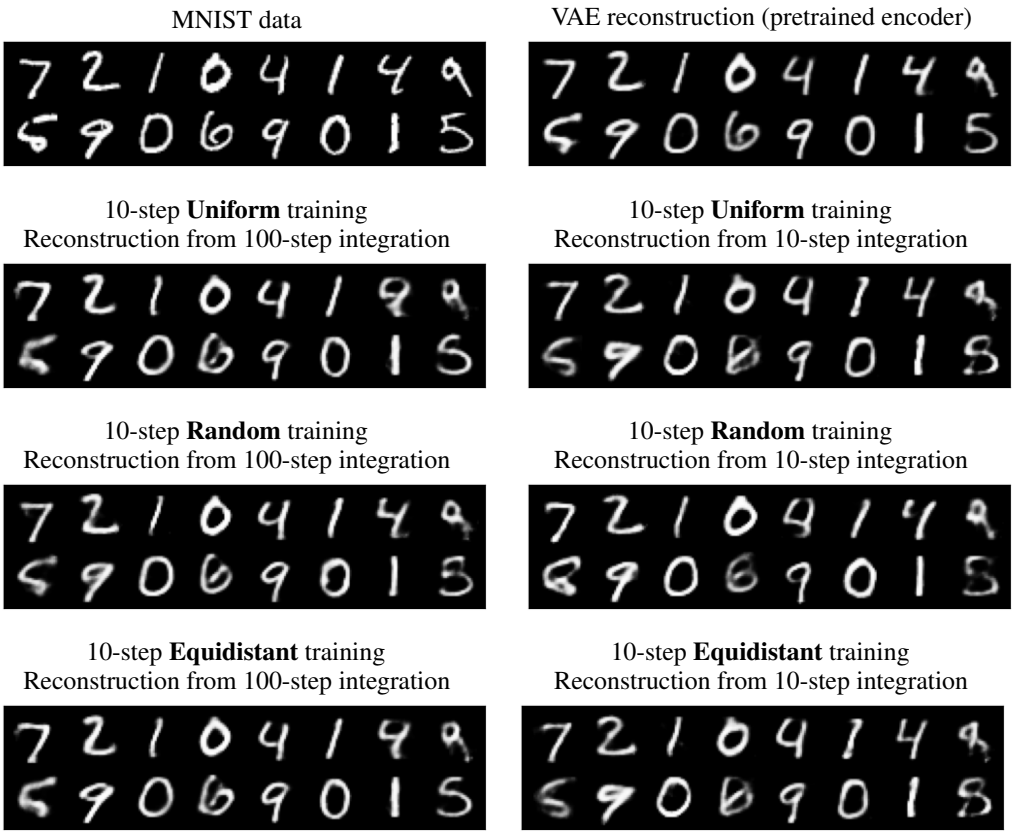


Figure 13: Mode of decoder $p(\cdot \mid z)$ evaluated on encoded latents z for the VAE experiment: input data x and reconstruction using z sampled from the pretrained VAE encoder (top row) and reconstructions using z sampled from diffusion encoders (next three rows).

Non-Gaussian Gaussian Processes for Few-Shot Regression

Marcin Sendera *
Jagiellonian University

Jacek Tabor
Jagiellonian University

Aleksandra Nowak
Jagiellonian University

Andrzej Bedychaj
Jagiellonian University

Massimiliano Patacchiola
University of Cambridge

Tomasz Trzcinski
Jagiellonian University,
Warsaw University of Technology,
Tooploox

Przemysław Spurek
Jagiellonian University

Maciej Zieba
Wrocław University of
Science and Technology,
Tooploox

Abstract

Gaussian Processes (GPs) have been widely used in machine learning to model distributions over functions, with applications including multi-modal regression, time-series prediction, and few-shot learning. GPs are particularly useful in the last application since they rely on Normal distributions and enable closed-form computation of the posterior probability function. Unfortunately, because the resulting posterior is not flexible enough to capture complex distributions, GPs assume high similarity between subsequent tasks – a requirement rarely met in real-world conditions. In this work, we address this limitation by leveraging the flexibility of Normalizing Flows to modulate the posterior predictive distribution of the GP. This makes the GP posterior locally non-Gaussian, therefore we name our method Non-Gaussian Gaussian Processes (NGGPs). We propose an invertible ODE-based mapping that operates on each component of the random variable vectors and shares the parameters across all of them. We empirically tested the flexibility of NGGPs on various few-shot learning regression datasets, showing that the mapping can incorporate context embedding information to model different noise levels for periodic functions. As a result, our method shares the structure of the problem between subsequent tasks, but the contextualization allows for adaptation to dissimilarities. NGGPs outperform the competing state-of-the-art approaches on a diversified set of benchmarks and applications.

1 Introduction

Gaussian Processes (GPs) [33, 46] are one of the most important probabilistic methods, and they have been widely used to model distributions over functions in a variety of applications such as multi-modal regression [56], time-series prediction [3, 27] and meta-learning [29, 45]. Recent works propose to use GPs in the *few-shot learning* scenario [4, 29, 39, 49], where the model is trained to solve a supervised task with only a few labeled samples available. This particular application is well-fitted to GPs since they can determine the posterior distribution in closed-form from a small set of data samples [29].

*Corresponding author: marcin.sendera@doctoral.uj.edu.pl

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

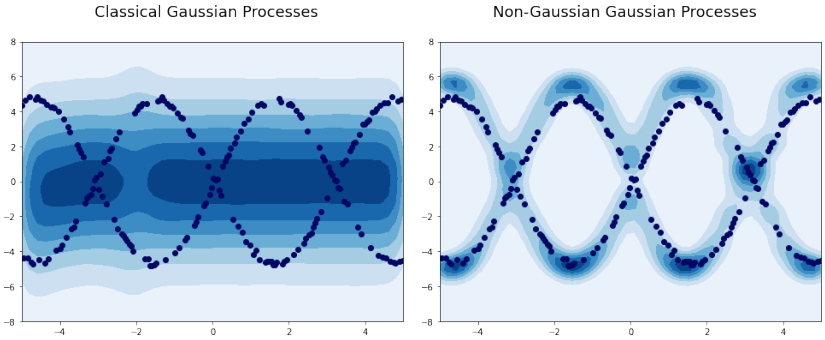


Figure 1: Results of Deep Kernels with classical GP (left) and NGGP (right). The one-dimensional samples were generated randomly from $\sin(x)$ and $-\sin(x)$ functions with additional noise. NGGP, compared to GP, does not have an assumption of Gaussian prior, which allows for modeling a multi-modal distribution.

However, the generalization capabilities of GPs come at the price of reduced flexibility when the modeled distributions are complex, *e.g.*, they have high skewness or heavy tails. Furthermore, GPs assume a high similarity between subsequent tasks. This condition is rarely met in real-world applications where tasks can vary during time, as is the case in heteroscedastic regression. These limitations of GPs also extend to multi-modal learning or, more generally, to multi-label regression [56].

In this work, we address those drawbacks by modeling the GPs posterior predictive distributions with a local non-Gaussian approximation. We do so by introducing a new method that we have named *Non-Gaussian Gaussian Processes (NGGPs)*. In NGGPs, we leverage the flexibility of Continuous Normalizing Flows (CNF) [16] to model arbitrary probability distributions. In particular, we propose an invertible ODE-based mapping that operates on each component of the random variable vectors. This way, we can compute a set of CNFs parameters shared across all vectors, with the resulting mapping incorporating the information of the context to model different noise for periodic functions. Figure 1 shows how NGGPs are able to capture the overall structure of a problem, whereas standard GPs fail. NGGPs are able to reconstruct a multi-modal sine function while adapting to local dissimilarities thanks to the contextualization provided by the ODE-based mapping. We provide empirical evidence that NGGPs outperform competitive state-of-the-art approaches on a diversified set of benchmarks and applications in a few-shot learning scenario; the code is released with an open-source license[2].

The contributions of our work can be summarized as follows:

- We introduce Non-Gaussian Gaussian Processes (NGGPs), a new probabilistic method for modeling complex distributions through locally non-Gaussian posteriors.
- We show how invertible ODE-based mappings can be coupled with GPs to process the marginals of multivariate random variables resulting in more flexible models.
- We extensively test NGGPs on a variety of few-shot learning benchmarks, achieving state-of-the-art performances in most conditions.

2 Related Work

The related work section is divided into three parts. First, we present a general Few-Shot Learning problem. Then, we discuss GPs, focusing on models, which use flow architectures. Finally, in the third paragraph, we describe existing approaches to Few-Shot Learning, which use Gaussian Processes.

Few-Shot Learning Few-Shot Learning aims at solving problems in which the number of observations is limited. Some of the early methods in this domain have applied a two-phase approach by pre-training on the base set of training tasks and then fine-tuning the parameters to the test tasks [4, 28]. An alternative approach is given by non-parametric metric-learning algorithms, which aim at optimizing a metric, that is then used to calculate the distance between the target observations

[2] `https://github.com/gmum/non-gaussian-gaussian-processes`

and the support set items [48, 38, 42]. Another popular approach to few-shot learning is Model Agnostic Meta-Learning (MAML) [9] and its variants [12, 24, 32, 54, 14, 52, 6]. MAML aims at finding a set of joined task parameters that can be easily fine-tuned to new test tasks via few gradient descent updates. MAML can also be treated as a Bayesian hierarchical model [10, 15, 18]. Bayesian MAML [55] combines efficient gradient-based meta-learning with non-parametric variational inference in a principled probabilistic framework. A few algorithms have been focusing exclusively on regression tasks. An example is given by ALPaCA [17], which uses a dataset of sample functions to learn a domain-specific encoding and prior over weights.

Gaussian Processes GPs have been applied to numerous machine learning problems, such as spatio-temporal density estimation [7], robotic control [53], or dynamics modeling in transcriptional processes in the human cell [21]. The drawback of GP lies in the computational cost of the training step, which is $O(n^3)$ (where n denotes the number of observations in the training sample).

In [41], the authors extend the flexibility of GPs by processing the targets with a learnable monotonic mapping (the warping function). This idea is further extended in [22], which shows that it is possible to place the prior of another GP on the warping function itself. Our method is different from these approaches, since the likelihood transformation is obtained by the use of a learnable CNF mapping.

In [26], the authors present the Transformed Gaussian Processes (TGP), a new flexible family of function priors that use GPs and flow models. TGPs exploit Bayesian Neural Networks (BNNs) as input-dependent parametric transformations. The method can match the performance of Deep GPs at a fraction of the computational cost.

The methods discussed above are trained on a single dataset, that is kept unchanged. Therefore, it is not trivial to adapt such methods to the the few-shot setting.

Few-Shot Learning with Gaussian Processes When the number of observations is relatively small, GPs represent an interesting alternative to other regression approaches. This makes GPs a good candidate for meta-learning and few-shot learning, as shown by recent publications that have explored this research direction. For instance, Adaptive Deep Kernel Learning (ADKL) [45] proposes a variant of kernel learning for GPs, which aims at finding appropriate kernels for each task during inference by using a meta-learning approach. A similar approach can be used to learn the mean function [11]. In [37], the authors presented a theoretically principled PAC-Bayesian framework for meta-learning. It can be used with different base learners (e.g., GPs or BNNs). Topics related to kernel tricks and meta-learning have been explored in [47]. The authors propose to use nonparametric kernel regression for the inner loop update. In [43], the authors introduce an information-theoretic framework for meta-learning by using a variational approximation to the information bottleneck. In their GP-based approach, to account for likelihoods other than Gaussians, they propose approximating the non-Gaussian terms in the posterior with Gaussian distributions (by using amortized functions), while we use CNFs to increase the flexibility of the GPs.

In [29], the authors present Deep Kernel Transfer (DKT): a Bayesian treatment for the meta-learning inner loop through the use of deep kernels, which has achieved state-of-the-art results. In DKT, the deep kernel and the parameters of the GP are shared across all tasks and adjusted to maximize the marginal log-likelihood, which is equivalent to Maximum-Likelihood type II (ML-II) learning. DKT is particularly effective in the regression case since it is able to capture prior knowledge about the data through the GP kernel. However, in many settings, prior assumptions could be detrimental if they are not met during the evaluation phase. This is the case in few-shot regression, where there can be a significant difference between the tasks seen at training time and the tasks seen at evaluation time. For instance, if we are given few-shot tasks consisting of samples from periodic functions but periodicity is violated at evaluation time, then methods like DKT may suffer in terms of predictive accuracy under this domain shift. In this work, we tackle this problem by exploiting the flexibility of CNFs.

3 Background

Gaussian Processes. The method proposed in this paper strongly relies on Gaussian Processes (GPs) and their applications in regression problems. GPs are a well-established framework for principled uncertainty quantification and automatic selection of hyperparameters through a marginal likelihood objective [35]. More formally, a GP is a collection of random variables such that the joint distribution of every finite subset of random variables from this collection is a multivariate Gaussian [31]. We

denote Gaussian Process as $f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$, where $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and covariance functions. When prior information is not available, a common choice for μ is the zero constant function. The covariance function must impose a valid covariance matrix. This is achieved by restricting k to be a kernel function. Examples of such kernels include the Linear kernel, Radial Basis Function (RBF) kernel, Spectral Mixture (Spectral) kernel [50], or Cosine-Similarity kernel [33]. Kernel functions can also be directly modeled as inner products defined in the feature space imposed by a feature mapping $\psi : X \to V$:

$$k(x, x') = \langle \psi(x), \psi(x') \rangle_V \tag{1}$$

An advantage of the formulation above is that it can be easily implemented by modeling ψ through a neural network. Throughout this work, we call this technique the NN Linear kernel (sometimes called Deep Kernel [29]). Since every kernel can be described in terms of Equation (1), such an approach may be desired if no prior information about the structure of the kernel function is available.

Gaussian Processes provide a method for modeling probability distributions over functions. Consider a regression problem:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i, \text{ for } i = 1, \ldots, m, \tag{2}$$

where ϵ_i are i.i.d. noise variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. Let $\mathbf{X}$ be the matrix composed of all samples $\mathbf{x}_i$ and let $\mathbf{y}$ be the vector composed of all target values $\mathbf{y}_i$. Assuming that $f(\cdot) \sim \mathcal{GP}\left(0, k\left(\cdot,\cdot\right)\right)$, we obtain:

$$\mathbf{y}|\mathbf{X} \sim \mathcal{N}(0, \mathbf{K} + \sigma\mathbb{I}), \tag{3}$$

where $k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Analogously, inference over the unknown during the training samples is obtained by conditioning over the normal distribution. Let $(\mathbf{y}, \mathbf{X})$ be the train data and let $(\mathbf{y}_*, \mathbf{X}_*)$ be the test data. Then the distribution of $\mathbf{y}_*$ given $\mathbf{y}, \mathbf{X}, \mathbf{X}_*$ is also a Gaussian distribution [34]:

$$\mathbf{y}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_* \sim \mathcal{N}(\mu_*, \mathbf{K}_*), \tag{4}$$

where:

$$\boldsymbol{\mu}_* = \mathbf{K}\left(\mathbf{X}_*, \mathbf{X}\right)\left(\mathbf{K}\left(\mathbf{X}, \mathbf{X}\right) + \sigma^2\mathbb{I}\right)^{-1}\mathbf{y}$$
$$\mathbf{K}_* = \mathbf{K}\left(\mathbf{X}_*, \mathbf{X}_*\right) + \sigma^2\mathbb{I} - \mathbf{K}\left(\mathbf{X}_*, \mathbf{X}\right)\left(\mathbf{K}\left(\mathbf{X}, \mathbf{X}\right) + \sigma^2\mathbb{I}\right)^{-1}\mathbf{K}\left(\mathbf{X}, \mathbf{X}_*\right)$$

Continuous Normalizing Flows. Normalizing Flows (NF) [36] are gaining popularity among generative models thanks to their flexibility and the ease of training via direct negative log-likelihood (NLL) optimization. Flexibility is given by the change-of-variable technique that maps a latent variable $\mathbf{z}$ with know prior $p(\mathbf{z})$ to $\mathbf{y}$ from some observed space with unknown distribution. This mapping is performed through a series of (parametric) invertible functions: $\mathbf{y} = \mathbf{f}_n \circ \cdots \circ \mathbf{f}_1(\mathbf{z})$. Assuming known prior $p(\mathbf{z})$ for $\mathbf{z}$, the log-likelihood for $\mathbf{y}$ is given by:

$$\log p(\mathbf{y}) = \log p(\mathbf{z}) - \sum_{n=1}^{N} \log \left| \det \frac{\partial \mathbf{f}_n}{\partial \mathbf{z}_{n-1}} \right|, \tag{5}$$

where $\mathbf{z} = \mathbf{f}_1^{-1} \circ \cdots \circ \mathbf{f}_n^{-1}(\mathbf{y})$ is a result of the invertible mapping. The biggest challenge in normalizing flows is the choice of the invertible functions $\mathbf{f}_n, \ldots, \mathbf{f}_1$. This is due to the fact that they need to be expressive while guaranteeing an efficient calculation of the Jacobian determinant, which usually has a cubic cost. An alternative approach is given by CNF models [16]. CNFs use continuous, time-dependent transformations instead of sequence of discrete functions $\mathbf{f}_n, \ldots, \mathbf{f}_1$. Formally, we introduce a function $\mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)$ that models the dynamics of $\mathbf{z}(t)$, $\frac{\partial \mathbf{z}(t)}{\partial t} = \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)$, parametrized by $\boldsymbol{\beta}$. In the CNF setting, we aim at finding a solution $\mathbf{y} := \mathbf{z}(t_1)$ for the differential equation, assuming the given initial state $\mathbf{z} := \mathbf{z}(t_0)$ with a known prior. As a consequence, the transformation function $\mathbf{f}_{\boldsymbol{\beta}}$ is defined as:

$$\mathbf{y} = \mathbf{f}_{\boldsymbol{\beta}}(\mathbf{z}) = \mathbf{z} + \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)dt. \tag{6}$$

The inverted form of the transformation can be easily computed using the formula:

$$\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = \mathbf{y} - \int_{t_0}^{t_1} \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{z}(t), t)dt. \tag{7}$$

The log-probability of $\mathbf{y}$ can be computed by:

$$\log p(\mathbf{y}) = \log p(\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y})) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}}{\partial \mathbf{z}(t)}\right) dt \quad \text{where} \quad \mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = \mathbf{z}. \tag{8}$$

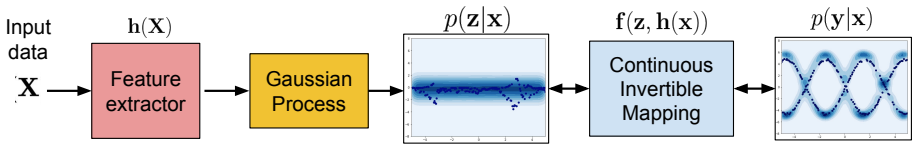


Figure 3: The general architecture of our approach. The input data are embedded by the feature extractor $\mathbf{h}(\cdot)$ and then used to create a kernel for the GP. Next, the output $\mathbf{z}$ of the GP is adjusted using an invertible mapping $\mathbf{f}(\cdot)$ which is conditioned on the output of the feature extractor. This allows us to model complex distributions of the target values $\mathbf{y}$.

4 Non-Gaussian Gaussian Processes

In this work, we introduce Non-Gaussian Gaussian Processes (NGGPs) to cope with the significant bottlenecks of Gaussian Processes for Few-Shot regression tasks: reduced flexibility and assumption about the high similarity between the structure of subsequent tasks. We propose to model the posterior predictive distribution as non-Gaussian on each datapoint. We are doing so by incorporating the flexibility of CNFs. However, we do not stack the CNF on GP to model the multidimensional distribution over $\mathbf{y}$. Instead, we attack the problem with an invertible ODE-based mapping that can utilize each component of the random variable vector and create the specific mapping for each datapoint (see Figure 2).

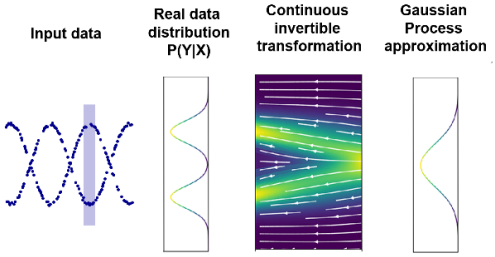


Figure 2: General idea of NGGP. A complex multimodal distribution can be modelled by exploiting a continuous invertible transformation to fit the Normal distribution used by the GP. Image inspired by Figure 1 in [16].

The general overview of our method is presented in Figure 3. Consider the data matrix $\mathbf{X}$, which stores the observations $\mathbf{x}_i$ for a given task. Each element is processed by a feature extractor $\mathbf{h}(\cdot)$ to create the latent embeddings. Next, we model the distribution of the latent variable $\mathbf{z}$ with a GP. Further, we use an invertible mapping $\mathbf{f}(\cdot)$ in order to model more complex data distributions. Note that the transformation is also conditioned on the output of the feature extractor $\mathbf{h}(\cdot)$ to include additional information about the input.

The rest of this section is organized as follows. In Section 4.1, we demonstrate how the marginal can be calculated during training. In Section 4.2, we demonstrate how to perform an inference stage with the model. Finally, in Section 4.3, we show how the model is applied to the few-shot setting.

4.1 Training objective

Consider the GP with feature extractor $\mathbf{h}_{\boldsymbol{\phi}}(\cdot)$ parametrized by $\boldsymbol{\phi}$ and any kernel function $k_{\boldsymbol{\theta}}(\cdot, \cdot)$ parametrized by $\boldsymbol{\theta}$. Assuming the given input data $\mathbf{X}$ and corresponding output values $\mathbf{z}$, we can define the marginal log-probability for the GP:

$$\log p(\mathbf{z}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{z}^{\mathrm{T}}(\mathbf{K} + \sigma^2\mathbb{I})^{-1}\mathbf{z} - \frac{1}{2}\log|\mathbf{K} + \sigma^2\mathbb{I}| - \frac{D}{2}\log(2\pi), \tag{9}$$

where D is the dimension of $\mathbf{y}$, $\mathbf{K}$ is the kernel matrix, and $k_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_i), \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_j))$.

Taking into account Equation (8) we can express the log marginal likelihood as follows:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \log p(\mathbf{z}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial \mathbf{g}_{\boldsymbol{\beta}}}{\partial \mathbf{z}(t)}\right) dt, \tag{10}$$

where $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = \mathbf{z}$, $p(\mathbf{z}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta})$ is the marginal defined by Equation (9) and $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\cdot)$ is the transformation given by Equation (6). In the next stage of the pipeline, we propose to apply the flow transformation $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\cdot)$ independently to each one of the marginal elements in $\mathbf{y}$, that is $\mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}) = [f_{\boldsymbol{\beta}}^{-1}(y_1), \ldots, f_{\boldsymbol{\beta}}^{-1}(y_D)]^{\mathrm{T}}$, with $f_{\boldsymbol{\beta}}^{-1}(\cdot)$ sharing its parameters across all components. In other words, while the GP captures the dependency across the variables, the flow operates independently on the marginal components of $\mathbf{y}$. Additionally, the flow is conditioned on the information

Algorithm 1 NGGP in the few-shot setting, train and test functions.

Require: $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^N$ train dataset and $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{Q}_*\}$ test task.
Parameters: $\boldsymbol{\theta}$ kernel hyperparameters, $\boldsymbol{\phi}$ feature extractor parameters, $\boldsymbol{\beta}$ flow transformation parameters.
Hyperparameters: α, η, γ: step size hyperparameters for the optimizers.

function TRAIN($\mathcal{D}$, α, η, γ, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\beta}$)
  while not done **do**
    Sample task $\mathcal{T} = (\mathbf{X}, \mathbf{y}) \sim \mathcal{D}$
    $\mathcal{L} = -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\beta})$ ▷ See Equation (13)
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\nabla_\theta\mathcal{L}$, ▷ Updating kernel hyperparameters
      $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta\nabla_\phi\mathcal{L}$, ▷ Updating feature extractor parameters
      $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \gamma\nabla_\beta\mathcal{L}$ ▷ Updating flow transformation parameters
  end while
  return $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\beta}$
end function
function TEST($\mathcal{T}_*$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\beta}$)
  Assign support $\mathcal{S}_* = (\mathbf{X}_{*,s}, \mathbf{y}_{*,s})$ and query $\mathcal{Q}_* = (\mathbf{X}_{*,q}, \mathbf{y}_{*,q})$
  return $p(\mathbf{y}_{*,q}|\mathbf{X}_{*,q}, \mathbf{y}_{*,s}, \mathbf{X}_{*,s}, , \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\beta})$ ▷ See Equation (14)
end function

encoded by the feature extractor, such that it can account for the context information $\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_d)$ from the corresponding input value $\mathbf{x}_d$:

$$y_d = f_{\boldsymbol{\beta}}(z_d, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_d)) = z_d + \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z_d(t), t, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_d))dt. \tag{11}$$

The inverse transformation can be easily calculated with the following formula:

$$f_{\boldsymbol{\beta}}^{-1}(y_d) = y_d - \int_{t_0}^{t_1} g_{\boldsymbol{\beta}}(z_d(t), t, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_d))dt \tag{12}$$

The final marginal log-likelihood can be expressed as:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \log p(\mathbf{z}^{\mathbf{h}}|\mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}) - \sum_{d=1}^{D} \int_{t_0}^{t_1} \frac{\partial g_{\boldsymbol{\beta}}}{\partial z_d(t)} dt, \tag{13}$$

where $\mathbf{z}^{\mathbf{h}} = \mathbf{f}_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}))$ is the vector of inverse functions $f_{\boldsymbol{\beta}}(z_d, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{x}_d))$ given by Equation (12).

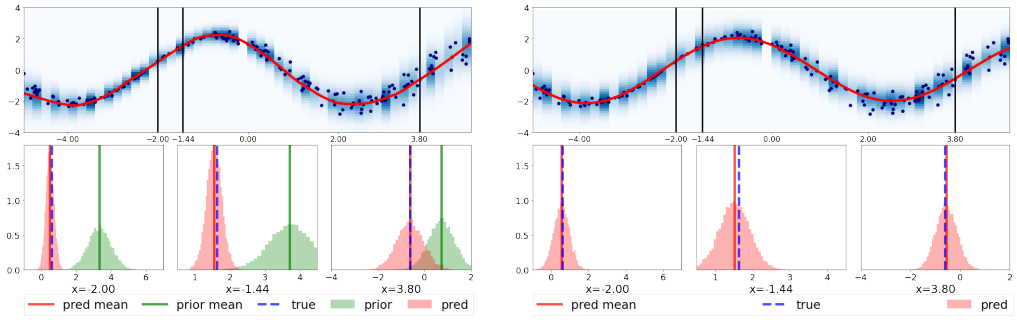
The transformation described above can be paired with popular CNF models. Here we choose Ffjord [16], which has showed to perform better on low-dimensional data when compared against discrete flows like RealNVP [5] or Glow [19]. Note that, the CNF is applied independently on the components of the GP outputs and shared across them. Therefore, we do not have any issue with the estimation of the Jacobian, since this corresponds to the first-order derivative of the output w.r.t. the scalar input.

4.2 Inference with the model

At inference time, we estimate the posterior predictive distribution $p(\mathbf{y}_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\beta})$, where we have access to training data $(\mathbf{y}, \mathbf{X})$ and model the probability of D_* test outputs $\mathbf{y}_*$ given the inputs $\mathbf{X}_*$. The posterior has a closed expression (see Section 3). Since the transformation given by Equation (11) operates independently on the outputs, we are still able to model the posterior in closed form:

$$\log p(\mathbf{y}_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \log p(\mathbf{z}_*^{\mathbf{h}}|\mathbf{X}, \mathbf{z}^{\mathbf{h}}, \mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta}) - \sum_{d=1}^{D_*} \int_{t_0}^{t_1} \frac{\partial g_{\boldsymbol{\beta}}}{\partial z_d(t)} dt, \tag{14}$$

where $\mathbf{z}_*^{\mathbf{h}} = f_{\boldsymbol{\beta}}^{-1}(\mathbf{y}_*, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}_*))$, $\mathbf{z}^{\mathbf{h}} = f_{\boldsymbol{\beta}}^{-1}(\mathbf{y}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{X}))$ are the inverted transformations for test and train data, and $p(\mathbf{z}_*^{\mathbf{h}}|\mathbf{X}_*, \mathbf{z}^{\mathbf{h}}, \mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta})$ is the GP posterior described in Equation (4).



(a) NGGP + NN Linear

(b) DKT + Spectral

Figure 4: The results for the sines dataset with mixed-noise for the best performing kernels for NGGP (NN Linear) and DKT (Spectral). The top plot in each figure represents the estimated density (blue hue) and predicted curve (red line), as well as the true test samples (navy blue dots). For three selected input points (denoted by black vertical lines), we plot the obtained marginal densities in the bottom images (red color). In addition, for the NGGP method, we also plot the marginal priors (in green) for each of these three points. It may be observed that NGGP is more successful in modeling the marginal for varying noise levels.

4.3 Adaptation for few-shot regression

In few-shot learning, we are given a meta-dataset of tasks $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$ where each task $\mathcal{T}_n$ contains a support set $\mathcal{S}_n$, and a query set $\mathcal{Q}_n$. At training time, both support and query contain input-output pairs $(\mathbf{X}, \mathbf{y})$, and the model is trained to predict the target in the query set given the support. At evaluation time, we are given a previously unseen task $\mathcal{T}_* = (\mathcal{S}_*, \mathcal{Q}_*)$, and the model is used to predict the target values of the unlabeled query points. We are interested in few-shot regression, where inputs are vectors and outputs are scalars.

We follow the paradigm of Deep Kernel Transfer (DKT) introduced in [29] and propose the following training and testing procedures (see Algorithm 1). During the training stage, we randomly sample the task, calculate the loss defined by Equation (13) and update all the parameters using gradient-based optimization. During testing, we simply identify the query and support sets and calculate the posterior given by Equation (14).

5 Experiments

In this section, we provide an extensive evaluation of our approach (NGGP) on a set of challenging few-shot regression tasks. We compare the results with other baseline methods used in this domain. As quantitative measures, we use the standard mean squared error (*MSE*) and, when applicable, the negative log-likelihood (*NLL*).

Sines dataset We start by comparing NGGP to other few-shot learning algorithms in a simple regression task defined on sines functions. To this end, we adapt the dataset from [9] in which every task is composed of points sampled from a sine wave with amplitude in the range $[0.1, 5.0]$, phase in the range $[0, \pi]$, and Gaussian noise $\mathcal{N}(0, 0.1)$. The input points are drawn uniformly at random from the range $[-5, 5]$. We consider 5 support and 5 query points during the training and 5 support and 200 query points during inference. In addition, following [29], we also consider an *out-of-range* scenario, in which the range during the inference is extended to $[-5, 10]$. We also perform a variation of sines experiment in which we inject input-dependent noise. The target values in this setting are modeled by $A \sin(x + \varphi) + |x + \varphi|\epsilon$, where the amplitude, phase, input, and noise points are drawn from the same distributions as in the standard setup described before. We refer to this dataset ablation as *mixed-noise sines*. For more information about the training regime and architecture, refer to Supplementary Materials A. Table 1 presents the results of the experiments. We use the DKT method as a reference since it provides state-of-the-art results for the few-shot sines dataset [29]. For a report with more baseline methods, please refer to Supplementary Materials B.

Both DKT and our NGGP perform very well when paired with the Spectral Mixture Kernel, achieving the same performance on *in-range* data. However, our approach gives superior results in the *out-of-*

Table 1: The *MSE* and *NLL* results for the inference tasks on sines datasets in the *in-range* and *out-range* settings. Lowest results in bold (the lower the better).

Method	sines				mixed-noise sines			
	in-range		out-of-range		in-range		out-of-range	
	MSE	*NLL*	*MSE*	*NLL*	*MSE*	*NLL*	*MSE*	*NLL*
DKT + RBF	1.36±1.64	-0.76±0.06	2.94±2.70	-0.69±0.06	1.60±1.63	0.48 ± 0.22	2.99± 2.37	2.01 ± 0.59
DKT + Spectral	**0.02±0.01**	**-0.83±0.03**	0.04±0.03	-0.70±0.14	**0.18 ± 0.12**	0.37±0.16	1.33 ± 1.10	1.58 ± 0.40
DKT + NN Linear	**0.02±0.02**	-0.73±0.11	6.61±31.63	38.38±40.16	**0.18±0.11**	0.45 ± 0.23	5.85 ± 12.10	8.64 ± 6.55
NGGP + RBF	1.02±1.40	-0.74±0.07	3.02±2.53	-0.65±0.08	1.30±1.36	0.33 ± 0.16	3.90 ± 2.60	1.83 ± 0.53
NGGP + Spectral	**0.02±0.01**	**-0.83±0.05**	**0.03±0.02**	**-0.80±0.07**	0.22 ± 0.14	0.44 ± 0.19	**1.14 ± 0.90**	**1.35 ± 0.38**
NGGP + NN Linear	0.04±0.03	-0.73±0.10	7.34±12.85	29.86±27.97	0.20 ± 0.12	**0.17 ± 0.15**	4.74 ± 6.29	2.92 ± 1.93

range scenario, confirming that NGGP is able to provide a better estimate of the predictive posterior for the unseen portions of the task. It is also worth noting that in all settings, NGGP consistently achieves the best *NLL* results. This is particularly evident for the *in-range* mixed-noise sines dataset. We analyze this result in Figure 4, where NGGP successfully models the distribution of the targets, predicting narrow marginals for the more centralized points and using wider distributions for the points with larger noise magnitude. This is in contrast with DKT, which fails to capture different noise levels within the data. These observations confirm our claim that the NGGP is able to provide a good estimate in the case of heteroscedastic data.

Head-pose trajectory In this experiment, we use the Queen Mary University of London multiview face dataset [13]. This dataset is composed of grayscale face images of 37 people (32 train, 5 test). There are 133 facial images per person, covering a viewsphere of $\pm 90^\circ$ in yaw and $\pm 30^\circ$ in tilt at 10° increment. We follow the evaluation procedure provided in [29]. Each task consists of randomly sampled trajectories taken from this discrete manifold. The *in-range* scenario includes the full manifold, while the *out-of-range* scenario includes only the leftmost 10 angles. At evaluation time, the inference is performed over the full manifold with the goal of predicting the tilt. The results are provided in Table 2. In terms of *MSE*, our NGGP method is competitive with other approaches, but it achieves significantly better *NLL* results, especially in the *out-of-range* setting. This suggests that NGGPs are indeed able to adapt to the differences between the tasks seen at training time and tasks seen at evaluation time by providing a probability distribution that accurately captures the true underlying data.

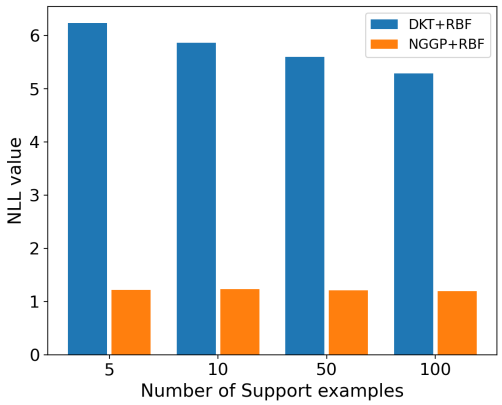
Table 2: Quantitative results for Queen Mary University of London for *in-range* and *out-of-range* settings, taking into account *NLL* and *MSE* measures.

Method	in-range		out-of-range	
	MSE	*NLL*	*MSE*	*NLL*
Feature Transfer/1	0.25±0.04	-	0.20±0.01	-
Feature Transfer/100	0.22±0.03	-	0.18±0.01	-
MAML (1 step)	0.21±0.01	-	0.18±0.02	-
DKT + RBF	0.12±0.04	0.13±0.14	0.14±0.03	0.71±0.48
DKT + Spectral	0.10±0.01	0.03±0.13	0.07±0.05	0.00±0.09
DKT + NN Linear	0.04±0.03	-0.12±0.12	0.12±0.05	0.30±0.51
NGGP + NN Linear	0.02±0.02	-0.47±0.32	0.06±0.05	0.24±0.91
NGGP + Spectral	**0.03±0.03**	**-0.68±0.23**	**0.03±0.03**	**-0.62±0.24**

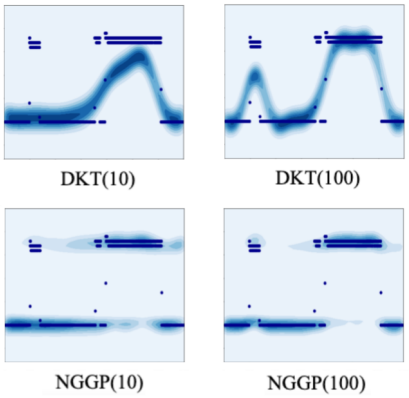
Object pose prediction We also study the behavior of NGGP in a pose prediction dataset introduced in [54]. Each task in this dataset consists of 30 gray-scale images with resolution 128×128, divided evenly into support and query. The tasks are created by selecting an object from the Pascal 3D [51] dataset, rendering it in 100 random orientations, and sampling out of it 30 representations. The goal is to predict the orientation relative to a fixed canonical pose. Note that 50 randomly selected objects are used to create the meta-training dataset, while the remaining 15 are utilized to create a distinct meta-test set. Since the number of objects in meta-training is small, a model could memorize the canonical pose of each object and then use it to predict the target value, completely disregarding the support points during the inference. This would lead to poor performance on the unseen objects in the meta-test tasks. This special case of overfitting is known as the *memorization problem* [54].

We analyze the performance of GP-based models in this setting by evaluating the performance of DKT and NGGP models[3]. We compare them against the methods used in [54], namely MAML [9],

[3]Information about architecture and training regime is given in Supplementary Materials A.



(a) NLL results DKT vs. NGGP.



(b) Single day comparison DKT vs. NGGP.

Figure 5: The results for the Power dataset experiment: **(a)** The quantitative comparison between DKT and NGGP considering different numbers of support examples. **(b)** The power consumption for a single day randomly selected from the test data. We compare DKT vs. NGGP (with RBF kernel) considering 10 and 100 support points. NGGP captures multi-modality and thus better adjusts to the data distribution.

Conditional Neural Processes (CNP) [12] and their meta-regularized versions devised to address the memorization problem — MR-MAML and MR-CNP [54]. In addition, we also include the fine-tuning (FT) baseline and CNP versions with standard regularization techniques such as Bayes-by-Backprop (BbB) [2] and Weight Decay [20]. The results are presented in Table 3.

Both GP-related approaches: NGGP and DKT are similar or usually outperform the standard and meta-regularized methods, which indicates that they are less prone to memorization and therefore benefit from a better generalization. The *NLL* is significantly lower for NGGP than for DKT, confirming that NGGP is better at inferring complex data distributions.

Table 3: Quantitative results for the object pose prediction task. We report the mean and standard deviation over 5 trials. The lower the better. Asterisks (*) denote values reported in [54].

Method	*MSE*	*NLL*
MAML*	5.39 ± 1.31	-
MR-MAML*	2.26 ± 0.09	-
CNP*	8.48 ± 0.12	-
MR-CNP*	2.89 ± 0.18	-
FT*	7.33 ± 0.35	-
FT + Weight Decay*	6.16 ± 0.12	-
CNP + Weight Decay*	6.86 ± 0.27	-
CNP + BbB*	7.73 ± 0.82	-
DKT + RBF	1.82 ± 0.17	1.35 ± 0.10
DKT + Spectral	$\mathbf{1.79 \pm 0.15}$	1.30 ± 0.06
NGGP + RBF	1.98 ± 0.27	$\mathbf{0.22 \pm 0.08}$
NGGP + Spectral	2.34 ± 0.28	0.86 ± 0.45

Power Dataset In this series of experiments, we use the Power [1] dataset and define an experimental setting for the few-shot setting. We treat each time series composed of 1440 values (60 minutes $\times$ 24 hours) that represents the daily power consumption (*sub_metering_3*) as a single task. We train the model using the tasks from the first 50 days, randomly sampling 10 points per task, while validation tasks are generated by randomly selecting from the following 50 days.

Quantitative and qualitative analysis are provided in Figure 5. We use only *NLL* to assess the results due to the multi-modal nature of the data and analyze the value of the criterion for different numbers of support examples. NGGP better adjusts to the true data distribution, even in the presence of very few support examples during inference. This experiment supports the claim that NGGPs are well-suited for modeling multi-modal distributions and step functions.

NASDAQ and EEG datasets In order to test the performance of our methods for real-world time series prediction, we used two datasets - NASDAQ100 [30] and EEG [8]. For an extensive description of the datasets and evaluation regime of this experiment, see Supplementary Materials A. Quantitative results are presented in Table 4. Our experiments show that NGGP outperforms the baseline DKT method across all datasets. The improvement is especially visible for the *out-of-range* NASDAQ100 when both methods use the RBF kernel. The results suggest that NGGPs can be successfully used to model real-world datasets, even when the data does not follow a Gaussian distribution.

Table 4: Quantitative results for NASDAQ and EEG datasets.

(a) NASDAQ100

in-range		
Method	*MSE · 100*	*NLL*
NGGP + RBF	**0.012 ± 0.014**	**-3.092 ± 0.255**
NGGP + NN Linear	0.023 ± 0.044	-2.567 ± 1.235
DKT + NN Linear	0.027 ± 0.032	-2.429 ± 0.271
DKT + RBF	0.022 ± 0.042	-2.878 ± 0.706
out-of-range		
Method	*MSE · 100*	*NLL*
NGGP + RBF	0.016 ± 0.034	-2.978 ± 0.571
NGGP + NN Linear	**0.003 ± 0.004**	**-2.998 ± 0.260**
DKT + NN Linear	0.005 ± 0.006	-2.612 ± 0.059
DKT + RBF	0.181 ± 0.089	1.049 ± 2.028

(b) EEG

in-range		
Method	*MSE · 100*	*NLL*
NGGP + RBF	**0.222 ± 0.181**	**-1.715 ± 0.282**
NGGP + NN Linear	0.361 ± 0.223	-1.387 ± 0.273
DKT + NN Linear	0.288 ± 0.169	-1.443 ± 0.188
DKT + RBF	0.258 ± 0.218	-1.640 ± 0.237
out-of-range		
Method	*MSE · 100*	*NLL*
NGGP + RBF	0.463 ± 0.415	**-1.447 ± 0.221**
NGGP + NN Linear	**0.452 ± 0.578**	-1.046 ± 0.624
DKT + NN Linear	0.528 ± 0.642	-1.270 ± 0.622
DKT + RBF	0.941 ± 0.917	-1.242 ± 0.685

6 Conclusions

In this work, we introduced NGGP – a generalized probabilistic framework that addresses the main limitations of Gaussian Processes, namely its rigidity in modeling complex distributions. NGGP leverages the flexibility of Normalizing Flows to modulate the posterior predictive distribution of GPs. Our approach offers a robust solution for few-shot regression since it finds a shared set of parameters between consecutive tasks while being adaptable to dissimilarities and domain shifts. We have provided an extensive empirical validation of our method, verifying that it can obtain state-of-the-art performance on a wide range of challenging datasets. In future work, we will focus on applications of few-shot regression problems needing the estimation of exact probability distribution (*e.g.*, continuous object-tracking) and settings where there is a potential discontinuity in similarity for subsequent tasks (*e.g.*, continual learning).

Limitations The main limitation of NGGP s is the costs of learning flow-based models, that could be more expensive than using a standard DKT when the data come from a simple distribution. In such a case, other methods like DKT could be more efficient. Moreover, GPs are expensive for tasks with a large number of observations, making NGGP a better fit for few-shot learning rather than bigger settings. Finally, in some cases, it can be more challenging to train and fine-tune NGGP than DKT because the number of parameters and hyper-parameters is overall larger (e.g. the parameters of the flow).

Broader Impact Gaussian Processes for regression already have had a huge impact on various real-world applications [7, 53, 21, 25]. NGGPs make it possible to apply *a priori* knowledge and expertise to even more complex real-world systems, providing fair and human-conscious solutions, *i.e.*, in neuroscience or social studies (see experiments on individual power consumption, EEG, and NASDAQ datasets from section 5). The proposed method is efficient and represents a great tool for better uncertainty quantification. Careful consideration of possible applications of our method must be taken into account to minimize any possible societal impact. For instance, the use of NGGP in object-tracking could be harmful if deployed with malevolent and unethical intents in applications involving mass surveillance.

Acknowledgments

This research was funded by Foundation for Polish Science (grant no POIR.04.04.00-00-14DE/18-00 carried out within the Team-Net program co-financed by the European Union under the European Regional Development Fund) and National Science Centre, Poland (grant no 2020/39/B/ST6/01511). The work of M. Zieba was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463. The work of P. Spurek was supported by the National Centre of Science (Poland) Grant No. 2019/33/B/ST6/00894. This research was funded by the Priority Research Area Digiworld under the program Excellence Initiative – Research University at the Jagiellonian University in Kraków. The authors have applied a CC BY license to any Author Accepted Manuscript (AAM) version arising from this submission, in accordance with the grants' open access conditions.

References

[1] Individual household electric power consumption data set. `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`. Accessed: 2021-05-25.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[3] Sofiane Brahim-Belhouari and Amine Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[6] Yingjun Du, Haoliang Sun, Xiantong Zhen, Jun Xu, Yilong Yin, Ling Shao, and Cees G. M. Snoek. Metakernel: Learning variational random features with limited labels, 2021.

[7] Vincent Dutordoir, Hugh Salimbeni, Marc Deisenroth, and James Hensman. Gaussian process conditional density estimation, 2018.

[8] SM Fernandez-Fraga, MA Aceves-Fernandez, JC Pedraza-Ortega, and JM Ramos-Arreguin. Screen task experiments for eeg signals based on ssvep brain computer interface. *International Journal of Advanced Research*, 6(2):1718–1732, 2018. Accessed: 2021-05-25.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[10] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[11] Vincent Fortuin, Heiko Strathmann, and Gunnar Rätsch. Meta-learning mean functions for gaussian processes. *arXiv preprint arXiv:1901.08098*, 2019.

[12] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[13] Shaogang Gong, Stephen McKenna, and John J Collins. An investigation into face pose distributions. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 265–270. IEEE, 1996.

[14] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018.

[15] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[16] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[17] James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 318–337. Springer, 2018.

[18] Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. *arXiv preprint arXiv:1812.06080*, 2018.

[19] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[20] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[21] Neil Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using gaussian processes. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[22] Miguel Lázaro-Gredilla. Bayesian warped gaussian processes. *Advances in Neural Information Processing Systems*, 25:1619–1627, 2012.

[23] Miguel Lázaro-Gredilla. Bayesian warped gaussian processes. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[24] Yan Li, Ethan X Fang, Huan Xu, and Tuo Zhao. International conference on learning representations 2020. In *International Conference on Learning Representations 2020*, 2020.

[25] Zhu Li, Adrian Perez-Suay, Gustau Camps-Valls, and Dino Sejdinovic. Kernel dependence regularizers and gaussian processes with applications to algorithmic fairness, 2019.

[26] Juan Maroñas, Oliver Hamelijnck, Jeremias Knoblauch, and Theodoros Damoulas. Transforming gaussian processes with normalizing flows. In *International Conference on Artificial Intelligence and Statistics*, pages 1081–1089. PMLR, 2021.

[27] Fatemeh Najibi, Dimitra Apostolopoulou, and Eduardo Alonso. Enhanced performance gaussian process regression for probabilistic short-term solar output forecast. *International Journal of Electrical Power & Energy Systems*, 130:106916, 2021.

[28] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[29] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

[30] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction, 2017. Accessed: 2021-05-25.

[31] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[32] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. *arXiv preprint arXiv:1909.04630*, 2019.

[33] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[34] Carl Edward Rasmussen and C Williams. Gaussian processes for machine learning the mit press. *Cambridge, MA*, 2006.

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[37] Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pages 9116–9126. PMLR, 2021.

[38] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[39] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each p\'olya-gamma augmented gaussian processes. *arXiv preprint arXiv:2007.10417*, 2020.

[40] Edward Snelson, Zoubin Ghahramani, and Carl Rasmussen. Warped gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[41] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped gaussian processes. *Advances in neural information processing systems*, 16:337–344, 2004.

[42] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[43] Michalis K Titsias, Sotirios Nikoloutsopoulos, and Alexandre Galashov. Information theoretic meta learning with gaussian processes. *arXiv preprint arXiv:2009.03228*, 2020.

[44] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[45] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

[46] Martin Trapp, Robert Peharz, Franz Pernkopf, and Carl Edward Rasmussen. Deep structured mixtures of gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 2251–2261. PMLR, 2020.

[47] Arun Venkitaraman, Anders Hansson, and Bo Wahlberg. Task-similarity aware meta-learning through nonparametric kernel regression. *arXiv preprint arXiv:2006.07212*, 2020.

[48] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[49] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[50] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075. PMLR, 2013.

[51] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014.

[52] Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam Kosiorek, and Yee Whye Teh. Metafun: Meta-learning with iterative functional updates. In *International Conference on Machine Learning*, pages 10617–10627. PMLR, 2020.

[53] Dit-Yan Yeung and Yu Zhang. Learning inverse dynamics by gaussian process begrression under the multi-task learning framework. In *The Path to Autonomous Robots*, pages 1–12. Springer, 2009.

[54] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

[55] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[56] Maciej Zięba, Marcin Przewięźlikowski, Marek Śmieja, Jacek Tabor, Tomasz Trzcinski, and Przemysław Spurek. Regflow: Probabilistic flow-based regression for future prediction. *arXiv preprint arXiv:2011.14620*, 2020.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5 and Supplementary Materials A.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Supplementary Materials A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5 and Supplementary Materials A.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5 and Supplementary Materials A.
   (b) Did you mention the license of the assets? [Yes] See Supplementary Materials A.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Supplementary Materials A.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Training Regime

A.1 Implementation of the GPs

We use the `GPyTorch`[4] package for the computations of GPs and their kernels. The NN linear kernel is implemented in all experiments as a 1-layer MLP with ReLU activations and hidden dimension 16. For the Spectral Mixture Kernel, we use 4 mixtures.

A.2 Sines Dataset

For the first experiments on sines functions, we use the dataset from [9]. For each task, the input points x are sampled from the range $[-5, 5]$, and the target values y are obtained by applying $y = A \sin(x - \varphi) + \epsilon$, where the amplitude A and phase φ are drawn uniformly at random from ranges $[0.1, 5]$ and $[0, \pi]$, respectively. The noise values ϵ are modeled by a normal distribution with zero mean and standard deviation equal to 0.1.

During the training, we use 5 support and 5 query points. The inference is performed over 500 tasks, each consisting of 200 query points and 5 support points. The models are trained for 50000 iterations with batch size 1 (one task per each parameters update) and learning rate 0.001 using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The feature extractor for this experiment is implemented by a 2-layer MLP with ReLU activations and hidden dimension 40, which follows the setting of [9]. The last hidden layer is used as the representation for the DKT[5] and NGGP methods in the Gaussian Process framework.

The CNF component for our model was inspired by FFJORD. Our implementation is based on the original code provided by the authors[6]. We use two stacked blocks of CNFs, each composed of two hidden *concatsquash* layers, 64 units each, with *tanh* activation. We adjusted *concatsquash* layers for the conditional variant of CNF by feeding them with an additional conditioning factor - the 40 dim output from the feature extractor.

We use the same settings for the *in-range* heterogeneous noise experiment, but we train the NGGP method for 10000 iterations instead of 50000 since we have noticed that this is enough for the model to converge.

A.3 Head-pose trajectory

For the head-pose trajectory task, we use the same setting as proposed in [29] with the same feature extractor - convolution neural network with 3 layers, each with 36 output channels, stride 2, and dilation 2. The NN Linear kernel in this experiment is implemented by a 1-layer MLP with ReLU activations and hidden dimension 16.

During the training phase, we use a meta-batch size equal to 5, the learning rate 0.001, and the Adam optimizer with the same configuration as in the sines experiment. Models were trained for 100 iterations. We use 5 support and 5 query points during the train. During the inference, we use 5 points as the support and the remaining samples of the trajectory as the query. We perform the inference over 10 different tasks.

For NGGP, we use the same CNF component architecture as in for the sines dataset. However, we also add Gaussian noise from the Normal distribution $\mathcal{N}(0, 0.1)$ to the head-pose orientations. Adding noise allows for better performance when learning with the CNF component.

A.4 Object pose prediction

In order to verify the extend of memorization in NGGP, we consider so-called *non-mutually exclusive* tasks. In this setting, the tasks are constructed in such a way that a single model can solve all tasks zero-shot. In particular, we follow the procedure of the pose prediction task introduced in [54]. The few-shot regression dataset is based on the Pascal 3D[7] data [51] and was recreated based on the code

[4] https://gpytorch.ai/, available on the MIT Licence

[5] For the DKT implementation we use the code provided at `https://github.com/BayesWatch/deep-kernel-transfer`

[6] `https://github.com/rtqichen/ffjord`

[7] `ftp://cs.stanford.edu/cs/cvgl/PASCAL3D+_release1.1.zip`

from the original research paper [8]. Firstly, the objects were randomly split into the meta-training set (50) and meta-testing (15), then the MuJoCo [44] library was used to render the instances of objects on a table, setting them random orientations. The observation is a tuple consisting of a 128×128 gray-scale image and its label - orientation relative to a fixed canonical pose. Every task consists of 30 positions sampled from the 100 renderings and divided randomly into *support* and *query*.

During the training, we use a meta-batch of 10 tasks. The NGGP and DKT models were trained over 1000 iterations, with learning rates equal to 0.01 for the kernel parameters, 0.01 for the feature extractor parameters, and 0.001 for the ODE-mapping component. We used the Adam optimizer with the same β configuration as in the sines experiment. We also use the same CNF component architecture as in the sines dataset. Similarly, as in the head-pose trajectory experiment, we add Gaussian noise from $\mathcal{N}(0, 0.1)$ to the orientations for better performance . The inference is performed over 100 tasks, which also consist of 15 support and 15 query points. As the feature extractor, we use one of the architectures tested in the original research paper [54] - the convolutional encoder with five layers stacked as follows: 2 convolutional layers with stride 2 and output dimensions 32 and 48; *max pooling layer* with kernel 2×2; convolutional layer with output dimension 64; *flatten layer* and *linear layer* with output dimension equal to 64.

For this dataset, we tested NGGP and DKT models with RBF and Spectral kernels only. This choice was due to the similarity between head-pose trajectory and object pose prediction settings, and the results show that these two kernels performed the best on such tasks.

A.5 Power Dataset

The Power Dataset[9] is an UCI benchmark that describes individual household electric power consumption. The original data is composed of 7 time-dependent attributes, but we focus only on the *sub_metering_3* attribute in our experiments. We split the dataset into tasks, where each of the tasks corresponds to daily electricity consumption and is represented by 1440 measurements (in minutes). We train the model using the first 50 days and validate it using the next 50 days. We used the same architecture as for the sines dataset in our experiments, except the feature extractor returns $1D$ embedding.

A.6 NASDAQ100 and EEG Datasets

The NASDAQ100[10] dataset consists of 81 major stocks under the NASDAQ 100 index. We decided to use the NASDAQ100 dataset with padding that includes 390 points per day over a 105 days interval.

We use 70% of the initial data points of the NDX100 index for the creation of meta-train tasks. The *in-range* meta-tasks were obtained from the last 30% of the data, while the *out-of-range* inference was obtained from the whole time-series of a different stock index. For this purpose, we utilize the time-series given by the YHOO index, which was not used during the training.

The EEG[11] dataset contains raw time series of brainwave signals sampled at 128Hz for 14 electrodes placed at different areas of the patient scalp. Particular patients had been stimulated for various periods, so the time series had different lengths.

The meta-training tasks were obtained form patient $A001SB1\_1$ and electrode $AF4$ from the first 70% of that time-series data points. Same as in NASDAQ100, meta-test tasks were for the *in-range* scenario were obtained from the last 30% of the same data. The *out-of-range* inference tasks were computed on different patient time-series of EEG data points - we used the $A003SB1\_1$ patient.

For both models, we used the same backbone architecture with Adam optimizer parameters set to the same values as in the experiment on the sines dataset with a learning rate set to 0.001. During the training and testing, we used 5 support and 5 query points. The support and query points where

[8] `https://github.com/google-research/google-research/tree/master/meta_learning_without_memorization`, on Apache-2.0 License

[9] `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`, made available under the "Creative Commons Attribution 4.0 International (CC BY 4.0)" license.

[10] `https://cseweb.ucsd.edu/~yaq007/NASDAQ100_stock_data.html`

[11] `https://archive.ics.uci.edu/ml/datasets/EEG+Steady-State+Visual+Evoked+Potential+Signals`, UCI repository dataset

sampled as an random interval of 10 consecutive points. Models were trained with a batch size 1 for 1000 iterations.

B Additional Results: Sines Regression

In addition to the GP-based methods reported in the main text, we also summarize the performance of other baseline algorithms on the sines dataset with standard Gaussian noise. The results are presented in Table 5. It may be observed that the DKT and NGGP significantly outperform other approaches. Therefore we only provide a comparison between those two methods in section 5 in the main paper.

Table 5: The *MSE* and *NLL* results for the inference tasks on sines datasets in the *in-range* and *out-range* settings. The lowest results in bold. Asterisks (*) and (**) denote values reported in [45] and [29], respectively. The lower the result, the better.

Method	**in-range**		**out-of-range**	
	MSE	*NLL*	*MSE*	*NLL*
ADKL*	0.14	-	-	-
R2-D2*	0.46	-	-	-
ALPaCA**	0.14±0.09	-	5.92±0.11	-
Feature Transfer/1**	2.94±0.16	-	6.13±0.76	-
Feature Transfer/100**	2.67±0.15	-	6.94±0.97	-
MAML (1 step)**	2.76±0.06	-	8.45±0.25	-
DKT + RBF	1.36±1.64	-0.76±0.06	2.94±2.70	-0.69±0.06
DKT + Spectral	**0.02±0.01**	**-0.83±0.03**	0.04±0.03	-0.70±0.14
DKT + NN Linear	0.02±0.02	-0.73±0.11	6.61±31.63	38.38±40.16
NGGP + RBF	1.02±1.40	-0.74±0.07	3.02±2.53	-0.65±0.08
NGGP + Spectral	**0.02±0.01**	-0.83±0.05	**0.03±0.02**	**-0.80±0.07**
NGGP + NN Linear	0.04±0.03	-0.73±0.10	7.34±12.85	29.86±27.97

C Additional Results: Classical Regression Tasks

Our main goal was to show improvement of NGGP over standard GPs in the case of a few-shot regression task. Albeit, we test our method also in classical regression task setting. Intuition is that NGGP may be superior to standard GPs in a simple regression setting for datasets with non-gaussian characteristics, but do not expect any improvement otherwise.

C.1 Classical Regression Tasks

Following the experiments from [23, 40], we decided to run NGGP on regular regression tasks. In this setting, we trained models over 10000 iterations on samples containing 100 points from a given dataset. Averaged results on 500 test samples containing 40 points that were not seen during the training - are presented in 6.

Table 6: Results on classical regression tasks on proposed datasets are inconclusive. One may see that results of methods performance vary between datasets.

Dataset	*abalone*		*Ailerons*		*creeprupt*	
	MSE	NLL	MSE	NLL	MSE	NLL
GP + RBF	1.26 ± 0.68	**-1.47 ± 0.20**	1.28 ± 0.66	-1.47 ± 0.19	1.26 ± 0.68	**-1.47 ± 0.19**
DKT + RBF	**1.18 ± 0.28**	-1.41 ± 0.09	1.41 ± 0.39	**-1.49 ± 0.12**	1.24 ± 0.39	-1.44 ± 0.12
NGGP + RBF	1.23 ± 0.29	-1.44 ± 0.09	**1.25 ± 0.31**	-1.44 ± 0.10	**1.10 ± 0.25**	-1.41 ± 0.08

C.2 Sines

Table 7: One may observe that addition of CNF significantly improves results of the classical GP with RBF kernel in such setting.

	GP + RBF	DKT + RBF	NGGP + RBF
MSE	1.06 ± 0.24	0.72 ± 0.32	**0.34 ± 0.22**
NLL	-0.98 ± 0.10	-1.20 ± 0.15	**-1.33 ± 0.13**

We ran additional experiments on a synthetic dataset of 2d sine waves (as in the setting from Figure 1). The data was generated by randomly sampling either $\sin(x)$ or $-\sin(x)$ for a given point x, together with adding uniform noise from $(0.1, 0.5)$. Models were trained for 10000 iterations over samples from the range $(-5.0, 5.0)$ with 100 points in one sample. The prediction was done for samples from the interval $(5.0, 10.0)$ - MSE and NLL were averaged on 500 test samples. We present the quantitative results in Table 7.

HyperShot: Few-Shot Learning by Kernel HyperNetworks

Marcin Sendera [†1] Marcin Przewięźlikowski[†1] Konrad Karanowski[2]
Maciej Zięba[2,3] Jacek Tabor[1] Przemysław Spurek[1]

[1]Faculty of Mathematics and Computer Science, Jagiellonian University
6 Łojasiewicza Street, 30-348 Kraków, Poland

[2]Department of Artificial Intelligence, University of Science and Technology
Wyb. Wyspiańskiego 27, 50-370, Wrocław, Poland

[3]Tooploox
Tęczowa 7, 53-601, Wrocław, Poland
marcin.{sendera, przewiezlikowski}@doctoral.uj.edu.pl

Abstract

Few-shot models aim at making predictions using a minimal number of labeled examples from a given task. The main challenge in this area is the one-shot setting where only one element represents each class. We propose HyperShot - the fusion of kernels and hypernetwork paradigm. Compared to reference approaches that apply a gradient-based adjustment of the parameters, our model aims to switch the classification module parameters depending on the task's embedding. In practice, we utilize a hypernetwork, which takes the aggregated information from support data and returns the classifier's parameters handcrafted for the considered problem. Moreover, we introduce the kernel-based representation of the support examples delivered to hypernetwork to create the parameters of the classification module. Consequently, we rely on relations between embeddings of the support examples instead of direct feature values provided by the backbone models. Thanks to this approach, our model can adapt to highly different tasks. [*]

1. Introduction

Current Artificial Intelligence techniques cannot rapidly generalize from a few examples. This common inability stems from the fact that most deep neural networks must be trained on large-scale data. In contrast, humans can learn new tasks quickly by utilizing what they learned in the past. **Few-shot learning** models try to fill this gap by *learning how to learn* from a limited number of examples. Few-shot learning is the problem of making predictions based on a small number of labeled examples. The goal of few-shot learning is not to recognize a fixed set of labels but to learn how to quickly adapt to new tasks with a small amount of training data. After training, the model can classify new data using only a few training examples.

Two new Few-shot learning techniques have recently emerged. The first one is based on the kernel method and Gaussian processes [20, 31, 39]. The universal deep kernel has enough data to generalize well to unseen tasks without over-fitting. The second technique makes use of the Hypernetworks [10, 22, 21, 40, 43, 44], which allow to aggregate information from the support set and produce dedicated network weights for new tasks.

The above approaches give promising results but also have some limitations. Kernel-based methods are not flexible enough, since they use Gaussian processes on top of the models. Moreover, it is not trivial to use Gaussian processes for classification tasks. On the other hand, Hypernetworks must aggregate information from the support set, and it is hard to model the relation between classes as opposed to classical feature extraction.

This paper combines the Hypernetworks paradigm with kernel methods to realize a new strategy that mimicks the human way of learning. First, we examine the entire support set and extract the information in order to distinguish objects of each class. Then, based on the relations between their features, we create the decision rules.

[*]The source code is available at: https://github.com/gmum/few-shot-hypernets-public.

[†]Denotes equal contribution.

Kernel methods realize the first part of the process. For each of the few-shot tasks, we extract the features from the support set through the backbone architecture and calculate kernel values between them. Then we use a Hypernetwork architecture [10, 39] – a neural network that takes kernel representation and produces decision rules in the form of a classifier. In our approach, the Hypernetwork aggregates the information from the support set and produces weights of the target model dedicated to the specific task, classifying the query set.

Our model, dubbed HyperShot, inherits the flexibility from Hypernetworks and the ability to learn the relation between objects from the kernel-based methods.

We perform an extensive experimental study of our approach by benchmarking it on various one-shot and few-shot image classification tasks. We find that HyperShot demonstrates high accuracy in all tasks, performing comparably or better than the other recently proposed methods. Moreover, HyperShot shows a strong ability to generalize, as evidenced by its performance on cross-domain classification tasks.

The contributions of this work are three-fold:

- In this paper, we propose a model which realizes the *learn how to learn* paradigm by modeling learning rules which are not based on gradient optimization and can produce completely different decision strategies.
- We propose a new approach to solve the few-shot learning problem by aggregating information from the support set by kernel methods and directly producing weights from the neural network dedicated to the query set.
- We propose HyperShot, which combines the Hypernetworks paradigm with kernel methods to produce the weights dedicated for each task.

2. HyperShot: Hypernetwork for few-shot learning

In this section, we present our HyperShot model for few-shot learning.

2.1. Background

Few-shot learning The terminology describing the few-shot learning setup is dispersive due to the colliding definitions used in the literature. For a unified taxonomy, we refer the reader to [4, 38]. Here, we use the nomenclature derived from the meta-learning literature, which is the most prevalent at the time of writing. Let:

$$\mathcal{S} = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^{L} \tag{1}$$

be a support-set containing input-output pairs, with L examples with the equal class distribution. In the *one-shot* scenario, each class is represented by a single example, and $L = K$, where K is the number of the considered classes in the given task. Whereas, for *few-shot* scenarios, each class usually has from 2 to 5 representatives in the support set $\mathcal{S}$. Let:

$$\mathcal{Q} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M} \tag{2}$$

be a query-set (sometimes referred to in the literature as a target-set), with M examples, where M is typically one order of magnitude greater than K. For ease of notation, the support and query sets are grouped in a task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\}$. During the training stage, the models for few-shot applications are fed by randomly selected examples from training set $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$, defined as a collection of such tasks.

During the inference stage, we consider task $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$, where $\mathcal{S}_*$ is a support set with the known class values for a given task, and $\mathcal{X}_*$ is a set of query (unlabeled) inputs. The goal is to predict the class labels for query inputs $\mathbf{x} \in \mathcal{X}_*$, assuming support set $\mathcal{S}_*$ and using the model trained on $\mathcal{D}$.

Hypernetwork In the canonical work [10], hypernetworks are defined as neural models that generate weights for a separate target network solving a specific task. The authors aim to reduce the number of trainable parameters by designing a hyper-network with a smaller number of parameters than the target network. Making an analogy between hyper-networks and generative models, the authors of [32] use this mechanism to generate a diverse set of target networks approximating the same function.

2.2. HyperShot - overview

We introduce HyperShot – a model that utilizes hypernetworks for few-shot problems. The main idea of the proposed approach is to predict the values of the parameters for a classification network that makes predictions on the query images given the information extracted from support examples for a given task. Thanks to this approach, we can switch the classifier's parameters between completely different tasks based on the support set. The information about the current task is extracted from the support set using a parameterized kernel function that operates on embedding space. Thanks to this approach, we use relations among the support examples instead of taking the direct values of the embedding values as an input to the hypernetwork. Consequently, this approach is robust to the embedding values for new tasks far from the feature regions observed during training. The classification of the query image is also performed using the kernel values calculated with respect to the support set.

The architecture of HyperShot is provided in Fig. 1. We aim to predict the class distribution $p(\mathbf{y}|S, \mathbf{x})$, given a query

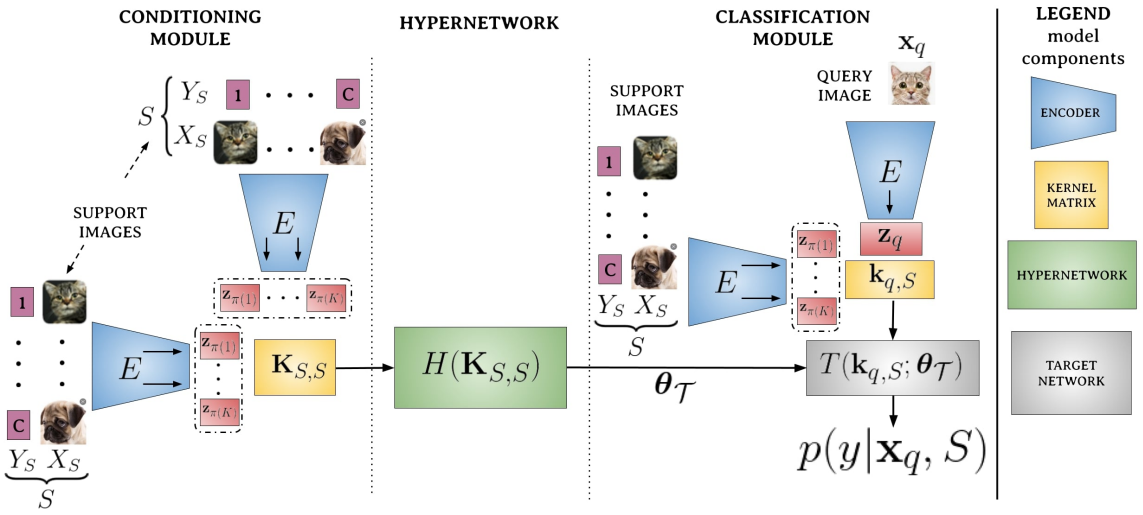


Figure 1. The general architecture of HyperShot. First, the examples from a support set are sorted according to the corresponding class labels and transformed by encoding network $E(\cdot)$ to obtain the matrix of ordered embeddings of the support examples, $\mathbf{Z}_S$. The low-dimensional representations stored in $\mathbf{Z}_S$ are further used to compute kernel matrix $\mathbf{K}_{S,S}$. The values of the kernel matrix are passed to the hypernetwork $H(\cdot)$ that creates the parameters $\boldsymbol{\theta}_T$ for the target classification module $T(\cdot)$. The query image $\mathbf{x}$ is processed by encoder $E(\cdot)$, and the vector of kernel values $\mathbf{k}_{\mathbf{x},S}$ is calculated between query embedding $\mathbf{z}_\mathbf{x}$ and the corresponding representations of support examples, $\mathbf{Z}_S$. The kernel vector $\mathbf{k}_{\mathbf{x},S}$ is further passed to target model $T(\cdot)$ to obtain the probability distribution for the considered classes.

image $\mathbf{x}$ and set of support examples $S = \{(\mathbf{x}_l, y_l)\}_{l=1}^{K}$. First, all images from the support set are grouped by their corresponding class values. Next, each of the images $\mathbf{x}_l$ from the support set is transformed using encoding network $E(\cdot)$, which creates low-dimensional representations of the images, $E(\mathbf{x}_l) = \mathbf{z}_l$. The constructed embeddings are sorted according to class labels and stored in the matrix $\mathbf{Z}_S = [\mathbf{z}_{\pi(1)}, \ldots, \mathbf{z}_{\pi(K)}]^{\mathrm{T}}$, where $\pi(\cdot)$ is the bijective function, that satisfies $y_{\pi(l)} \leq y_{\pi(k)}$ for $l \leq k$.

In the next step we calculate the kernel matrix $\mathbf{K}_{S,S}$, for vector pairs stored in rows of $\mathbf{Z}_S$. To achieve this, we use the parametrized kernel function $k(\cdot,\cdot)$, and calculate $k_{i,j}$ element of matrix $\mathbf{K}_{S,S}$ in the following way:

$$k_{i,j} = k(\mathbf{z}_{\pi(i)}, \mathbf{z}_{\pi(j)}). \tag{3}$$

The kernel matrix $\mathbf{K}_{S,S}$ represents the extracted information about the relations between support examples for a given task. The matrix $\mathbf{K}_{S,S}$ is further reshaped to the vector format and delivered to the input of the hypernetwork $H(\cdot)$. The role of the hypernetwork is to provide the parameters $\boldsymbol{\theta}_T$ of target model $T(\cdot)$ responsible for the classification of the query object. Thanks to that approach, we can switch between the parameters for entirely different tasks without moving via the gradient-controlled trajectory, like in some reference approaches like MAML.

The query image $\mathbf{x}$ is classified in the following manner. First, the input image is transformed to low-dimensional feature representation $z_\mathbf{x}$ by encoder $E(\mathbf{x})$. Further, the kernel vector $\mathbf{k}_{\mathbf{x},S}$ between the query embedding and sorted support vectors $\mathbf{Z}_S$ is calculated in the following way:

$$\mathbf{k}_{\mathbf{x},S} = [k(\mathbf{z}_\mathbf{x}, \mathbf{z}_{\pi(1)}), \ldots, k(\mathbf{z}_\mathbf{x}, \mathbf{z}_{\pi(K)})]^{\mathrm{T}}. \tag{4}$$

The vector $\mathbf{k}_{\mathbf{x},S}$ is further provided on the input of target model $T(\cdot)$ that is using the parameters $\boldsymbol{\theta}_T$ returned by hypernetwork $H(\cdot)$. The target model returns the probability distribution $p(\mathbf{y}|S, \mathbf{x})$ for each class considered in the task.

The function $\pi(\cdot)$ enforces some ordering of the input delivered to $T(\cdot)$. Practically, any other permutation of the classes for the input vector $\mathbf{k}_{\mathbf{x},S}$. In such a case, the same permutation should be applied to rows and columns of $\mathbf{K}_{S,S}$. As a consequence, the hypernetwork is able to produce the dedicated target parameters for each of the possible permutations. Although this approach does not guarantee the permutation invariance for real-life scenarios, thanks to dedicated parameters for any ordering of the input, it should be satisfied for major cases.

2.3. Kernel function

One of the key components of our approach is a kernel function $k(\cdot,\cdot)$. In this work we consider the dot product of the transformed vectors given by:

$$k(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{f}(\mathbf{z}_1)^{\mathrm{T}}\mathbf{f}(\mathbf{z}_2), \tag{5}$$

where $\mathbf{f}(\cdot)$ can be a parametrized transformation function, represented by MLP model, or simply an identity operation, $\mathbf{f}(\mathbf{z}) = \mathbf{z}$. In Euclidean space this criterion can be expressed as $k(\mathbf{z}_1, \mathbf{z}_2) = ||\mathbf{f}(\mathbf{z}_1)|| \cdot ||\mathbf{f}(\mathbf{z}_2)|| \cos\alpha$, where α is an angle between vectors $\mathbf{f}(\mathbf{z}_1)$ and $\mathbf{f}(\mathbf{z}_2)$. The main feature of this function is that it considers the vectors' norms, which can be problematic for some tasks that are outliers regarding the

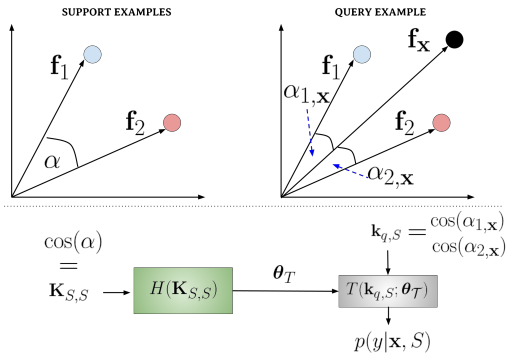


Figure 2. Simple 2D example illustrating the application of cosine kernel for HyperShot. We consider the two support examples from different classes represented by vectors $\mathbf{f_1}$ and $\mathbf{f_2}$. For this simple scenario, the input of hypernetwork is represented simply by the cosine of α, which is an angle between vectors $\mathbf{f}_1$ and $\mathbf{f}_2$. We aim at classifying the query example $\mathbf{x}$ represented by a vector $\mathbf{f}_\mathbf{x}$. Considering our approach, we deliver to the target network $T(\cdot)$ the cosine values of angles between first ($\alpha_{\mathbf{x},1}$) and second ($\alpha_{\mathbf{x},2}$) support vectors and classify the query example using the weights $\boldsymbol{\theta}_T$ created by hypernetwork $H(\cdot)$ from $\cos\alpha$ (remaining components on the diagonal of $\mathbf{K}_{S,S}$ are constant for cosine kernel).

representations created by $\mathbf{f}(\cdot)$. Therefore, we consider in our experiments also the cosine kernel function given by:

$$k_c(\mathbf{z}_1, \mathbf{z}_2) = \frac{\mathbf{f}(\mathbf{z}_1)^\mathrm{T}\mathbf{f}(\mathbf{z}_2)}{||\mathbf{f}(\mathbf{z}_1)|| \cdot ||\mathbf{f}(\mathbf{z}_2)||}, \tag{6}$$

that represents the normalized version dot product. Considering the geometrical representation, $k_c(\mathbf{z}_1, \mathbf{z}_2)$ can be expressed as $\cos\alpha$ (see the example given by Fig. 2). The support set is represented by two examples from different classes, $\mathbf{f}_1$ and $\mathbf{f}_2$. The target model parameters $\boldsymbol{\theta}_T$ are created based only on the cosine value of the angle between vectors $\mathbf{f}_1$ and $\mathbf{f}_2$. During the classification stage, the query example is represented by $\mathbf{f_x}$, and the classification is applied on the cosine values of angles between $\mathbf{f_x}$ and $\mathbf{f}_1$, and $\mathbf{f_x}$ and $\mathbf{f}_2$, respectively.

2.4. Training and prediction

The training procedure assumes the following parametrization of the model components. The encoder $E := E_{\boldsymbol{\theta}_E}$ is parametrized by $\boldsymbol{\theta_E}$, the hypernetwork $H = H_{\boldsymbol{\theta}_H}$ by $\boldsymbol{\theta}_H$, and the kernel function k by $\boldsymbol{\theta}_k$. We assume that training set $\mathcal{D}$ is represented by tasks $\mathcal{T}_i$ composed of support $\mathcal{S}_i$ and query $\mathcal{Q}_i$ examples. The training is performed by optimizing the cross-entropy criterion:

$$L = -\sum_{\mathcal{T}_i \in \mathcal{D}} \sum_{m=1}^{M} \sum_{k=1}^{K} y_{i,m}^k \log p(y_{i,m}^k | \mathcal{S}_i, \mathbf{x}_{i,m}), \tag{7}$$

where $(\mathbf{x}_{i,n}, \mathbf{y}_{i,n})$ are examples from query set $\mathcal{Q}_i$, where $Q_i = \{(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})\}_{m=1}^M$. The distribution for currently considered classes $p(\mathbf{y}|\mathcal{S}, \mathbf{x})$ is returned by target network T of HyperShot. During the training, we jointly optimize the parameters $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, and $\boldsymbol{\theta}_E$, minimizing the L loss.

Algorithm 1 HyperShot - training and prediction functions

Require: Training set $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^N$, and $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$ test task.
Parameters: $\boldsymbol{\theta}_H$ - parameters , $\boldsymbol{\theta}_k$ - kernel parameters, and $\boldsymbol{\theta}_E$ - encoder parameters
Hyperparameters: N_{train} - number of training iterations, N_{tune} number of tuning iterations, α- step size.

function TRAIN($\mathcal{D}$, α, N_{train}, $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_E$)
  while $n \leq N_{train}$ **do**
    Sample task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\} \sim \mathcal{D}$
    Assign support $\mathcal{S} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^M$
    $L = -\sum_{m=1}^M \sum_{k=1}^K y_m^k \log p(y_m^k | \mathcal{S}_i, \mathbf{x}_m, \boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E)$
    Update: $\boldsymbol{\theta}_E \leftarrow \boldsymbol{\theta}_E - \alpha\nabla_{\boldsymbol{\theta}_E}\mathcal{L}$,
      $\boldsymbol{\theta}_H \leftarrow \boldsymbol{\theta}_H - \alpha\nabla_{\boldsymbol{\theta}_H}\mathcal{L}$,
      $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \alpha\nabla_{\boldsymbol{\theta}_k}\mathcal{L}$
    $n = n + 1$
  end while
  return $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_E$
end function
function PREDICT($\mathcal{T}_*$, α, N_{tune}, $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_E$)
  Create tuning task: $\mathcal{T}_t = \{\mathcal{S}_*, \mathcal{S}_*\}$
  Adapt $\hat{\boldsymbol{\theta}}_H, \hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_E$ = TRAIN($\mathcal{T}_t$, α, N_{tune}, $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_E$)
  for each $\mathbf{x} \in \mathcal{X}_*$ **do**
    return $\arg\max_\mathbf{y} p(\mathbf{y}|\mathcal{S}_*, \mathbf{x}, \hat{\boldsymbol{\theta}}_H, \hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_E)$
  end for
end function

During the inference stage, we consider the task $\mathcal{T}_*$, composed of a set of labeled support examples $\mathcal{S}_*$ and a set of unlabelled query examples represented by input values $\mathcal{X}_*$ that the model should classify. We can simply take the probability values $p(\mathbf{y}|\mathcal{S}_*, \mathbf{x})$ assuming the given support set $\mathcal{S}_*$ and single query observation $\mathbf{x}$ from $\mathcal{X}_*$, using the model with trained parameters $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, and $\boldsymbol{\theta}_E$. However, we observe that slightly better results are obtained while adapting the model's parameters on the considered task. We do not have access to labels for query examples. Therefore we imitate the query set for this task simply by taking support examples and creating the adaptation task $\mathcal{T}_i = \{\mathcal{S}_*, \mathcal{S}_*\}$ and updating the parameters of the model using several gradient iterations. The detailed presentation o training and prediction procedures are provided by Algorithm 1.

2.5. Adaptation to few-shot scenarios

The proposed approach uses the ordering function $\pi(\cdot)$ that keeps the consistency between support kernel matrix $\mathbf{K}_{S,S}$ and the vector of kernel values $\mathbf{k}_{\mathbf{x},S}$ for query example $\mathbf{x}$. For few-shot scenarios, each class has more than one representative in the support set. As a consequence, there are various possibilities to order the feature vectors in the

support set inside the considered class. To eliminate this issue, we follow [44] and propose to apply the aggregation function to the embeddings $\mathbf{z}$ considering the support examples from the same class. Thanks to this approach, the kernel matrix is calculated based on the aggregated values of the latent space of encoding network E, making our approach independent of the ordering among the embeddings from the same class. In experimental studies, we examine the quality of *mean* aggregation operation (**averaged**) against simple class-wise concatenation of the embeddings (**fine-grained**) in ablation studies.

3. Related Work

Feature transfer [45] is a baseline procedure for few-shot learning and consists of pre-training the neural network and a classifier. During meta-validation, the classifier is fine-tuned to the novel tasks. [4] extend this idea by using cosine distance between the examples.

In recent years, a variety of meta-learning methods [3, 11, 30] have been proposed to tackle the problem of few-shot learning. The various meta-learning architectures for few-shot learning can be roughly categorized into several groups:

Memory-based methods [15, 16, 17, 27, 29, 42] are based on the idea to train a meta-learner with memory to learn novel concepts.

Metric-based methods meta-learn a deep representation with a metric in feature space, such that distance between examples from the support and query set with the same class have a small distance in such space. Some of the earliest works exploring this notion are Matching Networks [36] and Prototypical Networks [33], which form *prototypes* based on embeddings of the examples from the support set in the learned feature space and classify the query set based on the distance to those prototypes. Numerous subsequent works aim to improve the expressiveness of the prototypes through various techniques. [19] achieve this by conditioning the network on specific tasks, thus making the learned space task-dependent. [12] transform embeddings of support and query examples in the feature space to make their distributions closer to Gaussian. [35] propose Relation Nets, which learn the metric function instead of using a fixed one, such as Euclidean or cosine distance. Similar to the above methods, HyperShot uses a kernel function that predicts the relations between the examples in a given task. The key difference is that instead of performing a nearest-neighbor classification based on the kernel values, in HyperShot, the kernel matrix is classified by a task-specific classifier generated by the hypernetwork.

Optimization-based methods follow the idea of an optimization process over support set within the meta-learning framework like MetaOptNet [13], Model-Agnostic Meta-Learning (MAML), and its extensions [5, 18, 23, 25, 6, 18]. Those techniques aim to train general models, which can adapt their parameters to the support set at hand in a small number of gradient steps. Similar to such techniques, HyperShot also aims to produce task-specific models but utilizes a hypernetwork instead of optimization to achieve that goal.

Gaussian processes [26] possess many properties useful in few-shot learning, such as natural robustness to the limited amounts of data and the ability to estimate uncertainty. When combined with meta-learned deep kernels, [20], Gaussian processes were demonstrated to be a suitable tool for few-shot regression and classification, dubbed Deep Kernel Transfer (DKT). The assumption that such a universal deep kernel has enough data to generalize well to unseen tasks has been challenged in subsequent works. [39] introduced a technique of learning dense Gaussian processes by inducing variables. This approach achieves substantial performance improvement over the alternative methods. Similarly, HyperShot also depends on learning a model that estimates task-specific functions' parameters. However, HyperShot employs a hypernetwork instead of a Gaussian process to achieve that goal.

Hypernetworks [10] have been proposed as a solution to few-shot learning problems in a number of works but have not been researched as widely as the approaches mentioned above. Multiple works proposed various variations of hyper-networks that predict a shallow classifier's parameters given the support examples [2, 7, 22]. Subsequent works have extended those models by calculating cosine similarity between the query examples and the generated classifier weights [8] and utilizing a probabilistic model that predicts a distribution over the parameters suitable for the given task [9]. More recently, [21, 43, 44] explored generating all of the parameters of the target network with a transformer-based hypernetwork, but found that for larger target networks, it is sufficient to generate only the parameters of the final classification layer. A particularly effective approach is to use Transformer-based hypernetworks as set-to-set functions which make the generated classifier more discriminative [40]. A key characteristic of the above approaches is that during inference, the hypernetwork predicts weights responsible for classifying each class independently, based solely on the examples of that class from the support set. This property makes such solutions agnostic to the number of classes in a task, useful in practical applications. However, it also means that the hypernetwork does not take advantage of the inter-class differences in the task at hand.

In contrast, HyperShot exploits those differences by utilizing kernels, which helps improve its performance.

4. Experiments

In the typical few-shot learning setting, making a valuable and fair comparison between proposed models is often complicated because of the existence of the significant differences in architectures and implementations of known methods. In order to limit the influence of the deeper backbone (feature extractor) architectures, we follow the unified procedure proposed by [4].

In this section, we describe the experimental analysis and performance of the HyperShot in the large variety of few-shot benchmarks. Specifically, we consider both classification (see Section 4.1) and cross-domain adaptation (see Section 4.2) tasks. Whereas the classification problems are focused on the most typical few-shot applications, the latter cross-domain benchmarks check the ability of the models to adapt to out-of-distribution tasks. Additionally, we perform an ablation study of the possible adaptation procedures of HyperShot to few-shot scenarios, as well as architectural choices – presented in Section 4.3.

In all of the reported experiments, the tasks consist of 5 classes (5-way) and 1 or 5 support examples (1 or 5-shot). Unless indicated otherwise, all compared models use a known and widely utilized backbone consisting of four convolutional layers (each consisting of a 2D convolution, a batch-norm layer, and a ReLU non-linearity; each layer consists of 64 channels) [4] and have been trained from scratch.

We report the performance of two variants of HyperShot:

- **HyperShot** - models generated by the hypernetworks for each task.
- **HyperShot + adaptation** - models generated by hypernetworks adapted to the support examples of each task for 10 training steps[†].

In all cases, we observe a modest performance boost thanks to adapting the hypernetwork.

Comprehensive details for each training procedure are reported in the Appendix.

4.1. Classification

Firstly, we consider a classical few-shot learning scenario, where all the classification tasks (both training and inference) come from the same dataset. The main aim of the proposed classification experiments is to find the ability of the few-shot models to adapt to never-seen tasks from the same data distribution.

We benchmark the performance of the HyperShot and other methods on two challenging and widely considered datasets: Caltech-USCD Birds (**CUB**) [37] and **mini-ImageNet** [27]. The following experiments are in the most popular setting, 5-way, consisting of 5 random classes. In all experiments, the query set of each task consists of 16 samples for each class (80 in total). We provide the additional training details in the Appendix. We compare HyperShot to a vast pool of the state-of-the-art algorithms, including the canonical methods (like Matching Networks [36], Prototypical Networks [33], MAML [5], and its extensions) as well as the recently popular Bayesian methods mostly build upon the Gaussian Processes framework (like DKT [20]).

We consider the more challenging 1-shot classification task, as well as the 5-shot setting and report the results in Table 1. The additional comparing methods on larger backbones, as well as a bigger number of baselines, are included in Appendix.

In the 1-shot scenario, HyperShot achieves the second and third-best accuracies in the **CUB** dataset with and without utilizing an adapting procedure (66.13% with adapting, 65.27% without) and performs better than any other model, except for FEAT [40] (68.87%). In the **mini-ImageNet** dataset, our approach is among the top approachess (53.18%), slightly losing with DFSVLwF [8] (56.20%).

Considering the 5-shot scenario, HyperShot is the third-best model achieving 80.07% in the **CUB** dataset and 69.62% in the **mini-ImageNet**, whereas the best model, FEAT [40], achieves 82.90% and 71.66% on the mentioned datasets, respectively.

The obtained results clearly show that HyperShot achieves results comparable to state-of-the-art models on the standard set of few-shot classification settings.

4.2. Cross-domain adaptation

In the cross-domain adaptation setting, the model is evaluated on tasks coming from a different distribution than the one it had been trained on. Therefore, such a task is more challenging than standard classification and is a plausible indicator of a model's ability to generalize. In order to benchmark the performance of HyperShot in cross-domain adaptation, we merge data from two datasets so that the training fold is drawn from the first dataset and validation and testing fold – from another one. Specifically, we test HyperShot on two cross-domain classification tasks:
mini-ImageNet $\rightarrow$ **CUB** (model trained on **mini-ImageNet** and evaluated on **CUB**) and **Omniglot** $\rightarrow$ **EMNIST** in the 1-shot and 5-shot settings. We report the results in Table 2. In most settings, HyperShot achieves the highest accuracy, except for 1-shot **mini-ImageNet** $\rightarrow$ **CUB** classification, where its accuracy is on par with the accuracy achieved by DKT [20] (40.14% and 40.03% achieved by DKT and HyperShot, respectively). We note that just like

[†]In the case of the adapted hypernetworks, we tune a copy of the hypernetwork on the support set separately for each validation task. This way, we ensure that our model does not take unfair advantage of the validation tasks.

Table 1. The classification accuracy results for the inference tasks on **CUB** and **mini-ImageNet** datasets in the 1-shot and 5-shot settings. The highest results are in bold and second-highest in italic (the larger, the better).

	CUB		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
Feature Transfer [45]	46.19 ± 0.64	68.40 ± 0.79	39.51 ± 0.23	60.51 ± 0.55
Baseline++ [4]	61.75 ± 0.95	78.51 ± 0.59	47.15 ± 0.49	66.18 ± 0.18
MatchingNet [36]	60.19 ± 1.02	75.11 ± 0.35	48.25 ± 0.65	62.71 ± 0.44
ProtoNet [33]	52.52 ± 1.90	75.93 ± 0.46	44.19 ± 1.30	64.07 ± 0.65
RelationNet [35]	62.52 ± 0.34	78.22 ± 0.07	48.76 ± 0.17	64.20 ± 0.28
DKT + BNCosSim [20]	62.96 ± 0.62	77.76 ± 0.62	49.73 ± 0.07	64.00 ± 0.09
PPA [22]	–	–	54.53 ± 0.40	–
MAML [5]	56.11 ± 0.69	74.84 ± 0.62	45.39 ± 0.49	61.58 ± 0.53
MAML++ [1]	–	–	52.15 ± 0.26	68.32 ± 0.44
Bayesian MAML [41]	55.93 ± 0.71	–	53.80 ± 1.46	64.23 ± 0.69
Meta-SGD [14]	–	–	50.47 ± 1.87	64.03 ± 0.94
PAMELA [24]	–	–	53.50 ± 0.89	*70.51 ± 0.67*
FEAT [40]	$\mathbf{68.87 \pm 0.22}$	$\mathbf{82.90 \pm 0.15}$	*55.15 ± 0.20*	$\mathbf{71.61 \pm 0.16}$
DFSVLwF [8]	–	–	$\mathbf{56.20 \pm 0.86}$	–
HyperShot	65.27 ± 0.24	79.80 ± 0.16	52.42 ± 0.46	68.78 ± 0.29
HyperShot+ adaptation	*66.13 ± 0.26*	*80.07 ± 0.22*	53.18 ± 0.45	69.62 ± 0.20

in the case of regular classification, adapting the hypernetwork on the individual tasks consistently improves its performance.

4.3. Ablation study

In order to investigate different architectural choices in adapting HyperShot to the specific task, we provide a comprehensive ablation study. We focused mostly on the four major components of the HyperShot design, i.e., the method of processing multiple support examples per class, number of neck layers, the number of head layers, and the size of the hidden layers, presented in Tables 3, 4, and 5. In case of the experiments focusing on aggregating the number of support examples in the 5-way 5-shot setting, we perform the benchmarks on **CUB** and **mini-ImageNet**, using a 4-layer convolutional backbone. In the remaining experiments we tested HyperShot on the **CUB** dataset in the 5-way 1-shot setting with ResNet-10 backbone.

Aggregating support examples in the 5-shot setting In HyperShot, the hypernetwork generates the weights of the information about the support examples, expressed through the support-support kernel matrix. In the case of 5-way 1-shot classification, each task consists of 5 support examples, and therefore, the size of the kernel matrix is (5×5), and the input size of the hypernetwork is 25. However, with a growing number of the support examples, increasing the size of the kernel matrix would be impractical and could lead to overparametrization of the hypernetwork.

Since hypernetworks are known to be sensitive to large input sizes [10], we consider a way to maintain a constant input size of HyperShot, independent of the number of support examples of each class by using means of support embeddings of each class for kernel calculation, instead of individual embeddings. Prior works suggest that when there are multiple examples of a class, the averaged embedding of such class represents it sufficiently in the embedding space [33].

To verify this approach, in the 5-shot setting, we train HyperShot with two variants of calculating the inputs to the kernel matrix:

- **fine-grained** – utilizing a hypernetwork that takes as an input a kernel matrix between each of the embeddings of the individual support examples. This kernel matrix has a shape of (25×25).
- **averaged** – utilizing a hypernetwork where the kernel matrix is calculated between the **means** of embeddings of each class. The kernel matrix in this approach has a shape of (5×5).

We benchmark both variants of HyperShot on the 5-shot classification task on **CUB** and **mini-ImageNet** datasets, as well as the task of cross-domain **Omniglot** $\rightarrow$ **EMNIST** classification. We report the accuracies in Table 3. It is evident that averaging the embeddings before calculating the kernel matrix yields superior results.

Hidden size: Firstly, as presented in Table 4, we compare different sizes of hidden layers. The results agree with the intuition that the wider layers, the better results. However, we also observe that some hidden sizes (e.g., 8188) could be too large to learn effectively. Because of that, we propose to use hidden sizes of 2048 or 4096 as the standard.

Neck and head layers: Then, we compared the influence of the number of neck layers and head layers of HyperShot

Table 2. The classification accuracy results for the inference tasks on cross-domain tasks (**Omniglot→EMNIST** and **mini-ImageNet→CUB**) datasets in the 1-shot setting. The highest results are bold and second-highest in italic (the larger, the better).

	Omni→EMNIST		mini-ImageNet→CUB	
Method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Feature Transfer	64.22 ± 1.24	86.10 ± 0.84	32.77 ± 0.35	50.34 ± 0.27
Baseline++ [4]	56.84 ± 0.91	80.01 ± 0.92	39.19 ± 0.12	57.31 ± 0.11
MatchingNet [36]	75.01 ± 2.09	87.41 ± 1.79	36.98 ± 0.06	50.72 ± 0.36
ProtoNet [33]	72.04 ± 0.82	87.22 ± 1.01	33.27 ± 1.09	52.16 ± 0.17
MAML [5]	72.68 ± 1.85	83.54 ± 1.79	34.01 ± 1.25	48.83 ± 0.62
RelationNet [35]	75.62 ± 1.00	87.84 ± 0.27	37.13 ± 0.20	51.76 ± 1.48
DKT [20]	75.40 ± 1.10	*90.30 ± 0.49*	$\mathbf{40.14 \pm 0.18}$	56.40 ± 1.34
Bayesian MAML [41]	63.94 ± 0.47	65.26 ± 0.30	33.52 ± 0.36	51.35 ± 0.16
OVE PG GP + Cosine (ML) [34]	68.43 ± 0.67	86.22 ± 0.20	39.66 ± 0.18	55.71 ± 0.31
OVE PG GP + Cosine (PL) [34]	77.00 ± 0.50	87.52 ± 0.19	37.49 ± 0.11	57.23 ± 0.31
HyperShot	*78.06 ± 0.24*	89.04 ± 0.18	39.09 ± 0.28	*57.77 ± 0.33*
HyperShot + adaptation	$\mathbf{80.65 \pm 0.30}$	$\mathbf{90.81 \pm 0.16}$	*40.034 ± 0.41*	$\mathbf{58.86 \pm 0.38}$

Table 3. The classification *accuracy* results for HyperShot in the 5-shot setting with two variants of the support embeddings aggregation. The performance measured on **Omniglot→EMNIST**, **CUB**, and **mini-ImageNet→CUB** tasks. The larger, the better.

	Omni→EMNIST	CUB	mini-ImageNet
HyperShot (fine-grained)	87.55 ± 0.19	78.05 ± 0.20	67.07 ± 0.47
HyperShot (averaged)	89.04 ± 0.18	79.80 ± 0.16	69.62 ± 0.28

Table 4. Comparison between various hidden sizes in the HyperShot's layers. The classification *accuracy* results on **CUB** task and 5-way 1-shot setting. The larger, the better.

hidden size	**accuracy**
256	70.16 ± 0.45
512	71.70 ± 0.46
1024	70.89 ± 0.62
2048	$\mathbf{72.43 \pm 0.59}$
4096	71.99 ± 0.70
8188	72.05 ± 0.33

Table 5. Comparison between various HyperShot's architectures (different number of **neck layers** and **head layers**). The classification *accuracy* results on **CUB** task and 5-way 1-shot setting. The larger, the better.

neck layers	**head layers**	**accuracy**
1	**3**	$\mathbf{73.00 \pm 0.55}$
2	1	71.53 ± 0.33
2	2	68.06 ± 0.59
2	3	71.99 ± 0.70
3	3	70.81 ± 0.39

for the achieved results, as presented in Table 5. We observed that the most critical is the number of head layers - specific for each target network's layers. Because of that, we propose using the standard number of 3 head layers and using various neck layers - tuning them to the specific task.

5. Conclusion

In this work, we introduced HyperShot — a new framework that uses kernel methods combined with hypernetworks. Our method uses the kernel-based representation of the support examples and a hypernetwork paradigm to create the query set's classification module. We concentrate on relations between embeddings of the support examples instead of direct feature values. Thanks to this approach, our model can adapt to highly different tasks.

We evaluate the HyperShot model on various one-shot and few-shot image classification tasks. HyperShot demonstrates high accuracy in all tasks, performing comparably or better to state-of-the-art solutions. Furthermore, the model has a strong ability to generalize, as evidenced by its performance on cross-domain classification tasks.

Limitations The main limitation of HyperShot is the considerable (up to 10000 epochs) training time. This could possibly be reduced by employing a pre-training scheme similar to [28, 40]. We are planning to investigate this in the future.

Impact The results of this research show that Few-Shot methods which utilize kernels and hypernetworks achieve good performance in most benchmarks, particularly cross-domain classification. Thus, HyperShot and other similar models offer a promising direction in the research towards better generalization of Few-Shot models.

Acknowlegdements

The work of J. Tabor was supported by the National Centre of Science (Poland) Grant No. 2019/33/B/ST6/00894. The work of P. Spurek and M.Przewięźlikowski was supported by the National Centre of Science (Poland) Grant No. 2021/43/B/ST6/01456. The work of M. Zięba and M. Sendera was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463.

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2018.

[2] Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartlomiej Świątkowski, Bernhard Schölkopf, and Richard E. Turner. Discriminative k-shot learning using probabilistic models, 2017.

[3] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. 1992.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[6] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9537–9548, 2018.

[7] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 10–15 Jul 2018.

[8] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting, 2018.

[9] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2018.

[10] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[11] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

[12] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based few-shot learning, 2021.

[13] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[14] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017.

[15] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[16] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

[17] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, pages 3664–3673. PMLR, 2018.

[18] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[19] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[20] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

[21] Toby Perrett, Alessandro Masullo, Tilo Burghardt, Majid Mirmehdi, and Dima Damen. Temporal-relational crosstransformers for few-shot action recognition. *CoRR*, abs/2101.06184, 2021.

[22] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Yuille. Few-shot image recognition by predicting parameters from activations, 2017.

[23] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[24] Jathushan Rajasegaran, Salman H. Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Meta-learning the learning trends shared across tasks. *CoRR*, abs/2010.09291, 2020.

[25] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*, 32:113–124, 2019.

[26] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[27] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[28] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization, 2019.

[29] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[30] Jürgen Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, 01 1992.

[31] Marcin Sendera, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzciński, Przemysław Spurek, and Maciej Zięba. Non-gaussian gaussian processes for few-shot regression, 2021.

[32] Abdul-Saboor Sheikh, Kashif Rasul, Andreas Merentitis, and Urs Bergmann. Stochastic maximum likelihood optimization via hypernetworks. *arXiv preprint arXiv:1712.01141*, 2017.

[33] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[34] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. In *International Conference on Learning Representations*, 2020.

[35] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[36] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[37] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[38] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning, 2020.

[39] Ze Wang, Zichen Miao, Xiantong Zhen, and Qiang Qiu. Learning to learn dense gaussian processes for few-shot learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[40] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

[41] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[42] Xiantong Zhen, Ying-Jun Du, Huan Xiong, Qiang Qiu, Cees Snoek, and Ling Shao. Learning to learn variational semantic memory. In *NeurIPS*, 2020.

[43] Andrey Zhmoginov, Mark Sandler, and Max Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning, 2022.

[44] Zhenxi Zhu, Limin Wang, Sheng Guo, and Gangshan Wu. A closer look at few-shot video classification: A new baseline and benchmark, 2021.

[45] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.

HyperShot: Few-Shot Learning by Kernel HyperNetworks – supplementary material

Marcin Sendera [†1] Marcin Przewięźlikowski[†1] Konrad Karanowski[2]
Maciej Zięba[2,3] Jacek Tabor[1] Przemysław Spurek[1]

[1]Faculty of Mathematics and Computer Science, Jagiellonian University
6 Łojasiewicza Street, 30-348 Kraków, Poland

[2]Department of Artificial Intelligence, University of Science and Technology
Wyb. Wyspiańskiego 27, 50-370, Wrocław, Poland

[3]Tooploox
Tęczowa 7, 53-601, Wrocław, Poland
marcin.{sendera, przewiezlikowski}@doctoral.uj.edu.pl

A. Additional results

This section provides additional results in the 5-way (1-shot and 5-shot) classification tasks for models using larger backbones, namely ResNet-10 and ResNet-12 [11], as well as and expanded version of Table 1 from the main text, containing more baselines . We provide the results for ResNet-10 on the **CUB** and **mini-ImageNet** datasets in Table 1, for ResNet-12 on **mini-ImageNet** dataset in Table 2 and for Conv4 on the **CUB** and **mini-ImageNet** datasets in Table 3. It should be noted that the results for ResNet-10 on **mini-ImageNet** for all methods were obtained by us using a unified codebase [3, 27]. On the other hand, for benchmarks of ResNet-10 on **CUB** and ResNet-12 on **mini-ImageNet** we report the accuracies of methods other than HyperShot as reported in [27] and [45], respectively.

ResNet-10 – CUB and mini-ImageNet In the **CUB** dataset classification tasks (see Table 1), HyperShot is amongst the state-of-the-art models achieving classification accuracy often equal within the variance to the best models. Considering the 5-shot scenario, the highest classification result across the evaluated methods ($86.38\% \pm 0.15$) obtained the GPLDLA model based on the Gaussian Processes framework. However, the HyperShot performance, $86.28\% \pm 0.29$, is the second-best but even lies within the variance of the best model. In the 1-shot setting, ProtoNet obtains the highest result ($73.22\% \pm 0.92$), whereas HyperShot is the third one ($71.99\% \pm 0.70$) but still equal according to the variances.

In the **mini-ImageNet** classification task with the ResNet-10 backbone, HyperShot achieves the second-best accuracy in both 1-shot and 5-shot settings*. In the 1-shot setting, the DKT model [27] achieved the best result, with HyperShot being a close second, with only 0.04 pp difference. In the 5-shot setting, the baseline++ approach outperforms all others by a large margin [3], whereas HyperShot and ProtoNet [35] achieve similar, second-best results. We observe that apart from HyperShot, which achieves second-best results in both settings, models which perform well in one setting are outperformed by others in the second and vice versa.

ResNet-12 – mini-ImageNet In the **mini-ImageNet** classification task with the ResNet-12 backbone (see Table 2), HyperShot ranks relatively low in terms of accuracy (63.30% and 76.21% in the 1-shot and 5-shot settings, respectively), as compared to recently proposed approaches such as RENet [14] and FEAT [45], which outperform it by a large margin. Nevertheless, we note that for the experiments on ResNet-12 we used the same set of hyperparameters as in the ResNet-10 experiments, which may not be an optimal choice for a larger backbone.

[†]Denotes equal contribution.

*In the case of the **mini-ImageNet** classification with ResNet10, we benchmarked all of the listed models ourselves. To our best knowledge, previously, there were no reported benchmarks on this dataset with the ResNet-10 backbone.

It is worth noticing that HyperShot without adaptation steps performances sometimes slightly better than the same with adaptation. We even observe that a few first steps of adaptation procedure result in an unnoticeable increase of accuracy of the basic model. However, the usual 10 steps result in this setting in slightly worse performance, so one should use it cautiously. We decided to report the results after the standard adaptation procedure only.

B. Training details

In this section, we present in detail the architecture and hyperparameters of HyperShot.

Architecture overview From a high-level perspective, the architecture of HyperShot consists of three parts:

- backbone - a convolutional feature extractor.
- neck - a sequence of zero or more fully-connected layers with ReLU nonlinearities in between.
- heads - for each parameter of the target network, a sequence of one or more linear layers, which predicts the values of that parameter. All heads of HyperShot have identical lengths, hidden sizes, and input sizes that depend on the generated parameter's size.

The target network generated by HyperShot re-uses its backbone. We outline this architecture in Figure 1.

Backbone For each experiment described in the main body of this work, we follow [27] in using a shallow backbone (feature extractor) for HyperShot as well as referential models. This backbone consists of four convolutional layers, each consisting of a convolution, batch normalization, and ReLU nonlinearity. Apart from the first convolution, which has the number of input size equal to the number of image channels, each convolution has an input and output size of 64. We apply max-pooling between each convolution, which decreases by half the resolution of the processed feature maps. The output of the backbone is flattened so that the further layers can process it.

We perform additional experiments described in Appendix A where instead of the above backbone, we utilize ResNet-10 [11].

Datasets For the purpose of making a fair comparison, we follow the procedure presented in, e.g., [27, 3]. In the case of the **CUB** dataset [41], we split the whole amount of 200 classes (11788 images) across train, validation, and test consisting of 100, 50, and 50 classes, respectively [3]. The **mini-ImageNet** dataset [32] is created as the subset of **ImageNet** [34], which consists of 100 different classes represented by 600 images for each one. We followed the standard procedure and divided the **mini-ImageNet** into 64 classes for the train, 16 for the validation set, and the remaining 20 classes for the test. The well-known **Omniglot** dataset [17] is a collection of characters from 50 different languages. The **Omniglot** contains 1623 white and black characters in total. We utilize the standard procedure to include the examples rotated by 90° and increase the size of the dataset to 6492, from which 4114 were further used in training. Finally, the **EMNIST** dataset [4] collects the characters and digits coming from the English alphabet, which we split into 31 classes for the test and 31 for validation.

Data augmentation We apply data augmentation during model training in all experiments, except **Omniglot** $\rightarrow$ **EMNIST** cross-domain classification. The augmentation pipeline is identical to the one used by [27] and consists of the random crop, horizontal flip, and color jitter steps.

C. Hyperparameters

Below, we outline the hyperparameters of architecture and training procedures used in each experiment.

We use cosine similarity as a kernel function and averaged support embeddings aggregation in all experiments. HyperShot is trained with the learning rate of 0.001 with the Adam optimizer [16] and no learning rate scheduler. Task-specific adaptation is also performed with the Adam optimizer and the learning rate of 0.0001.

For the natural image tasks (**CUB**, **mini-ImageNet**, **mini-ImageNet** $\rightarrow$ **CUB** classification), we use a hypernetwork with the neck length o 2, head lengths of 3, and a hidden size of 4096, which produce a target network with a single fully-connected layer. We perform training for 10000 epochs.

For the simpler **Omniglot** $\rightarrow$ **EMNIST** character classification task, we train a smaller hypernetwork with the neck length of 1, head lengths of 2, and the hidden size of 512, which produces a target network with two fully-connected layers and a hidden size of 128. We train this hypernetwork for a shorter number of epochs, namely 2000.

We summarize all the above hyperparameters in Table 4.

D. Source code

The source code required for running the experiments is available at `https://github.com/gmum/few-shot-hypernets-public`.

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2018.

[2] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form

Table 1. The classification accuracy results for the inference tasks in the **CUB** and **mini-ImageNet** dataset in the 5-way (1-shot and 5-shot) scenarios. We consider models using the ResNet-10 backbone. The highest results are bold and second-highest in italic (the larger, the better).

	CUB		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
Feature Transfer	63.64 ± 0.91	81.27 ± 0.57	–	–
Baseline++ [3]	69.55 ± 0.89	85.17 ± 0.50	54.35 ± 0.34	$\mathbf{75.26 \pm 0.16}$
MatchingNet [40]	71.29 ± 0.87	83.47 ± 0.58	54.18 ± 0.09	67.71 ± 0.20
ProtoNet [35]	$\mathbf{73.22 \pm 0.92}$	85.01 ± 0.52	53.28 ± 0.17	73.04 ± 0.15
MAML [6]	70.32 ± 0.99	80.93 ± 0.71	–	–
RelationNet [38]	70.47 ± 0.99	83.70 ± 0.55	51.88 ± 0.45	67.21 ± 0.16
DKT + CosSim [27]	70.81 ± 0.52	83.26 ± 0.50	–	–
DKT + BNCosSim [27]	$\mathit{72.27 \pm 0.30}$	85.64 ± 0.29	$\mathbf{56.03 \pm 0.50}$	71.28 ± 0.12
SimpleShot [42]	53.78 ± 0.21	71.41 ± 0.17	–	–
GPLDLA [15]	71.30 ± 0.16	$\mathbf{86.38 \pm 0.15}$	–	–
HyperShot	71.99 ± 0.70	86.28 ± 0.29	55.36 ± 0.64	$\mathit{73.06 \pm 0.30}$
HyperShot + adaptation	71.60 ± 0.59	$\mathit{86.22 \pm 0.30}$	$\mathit{55.99 \pm 0.63}$	72.87 ± 0.33

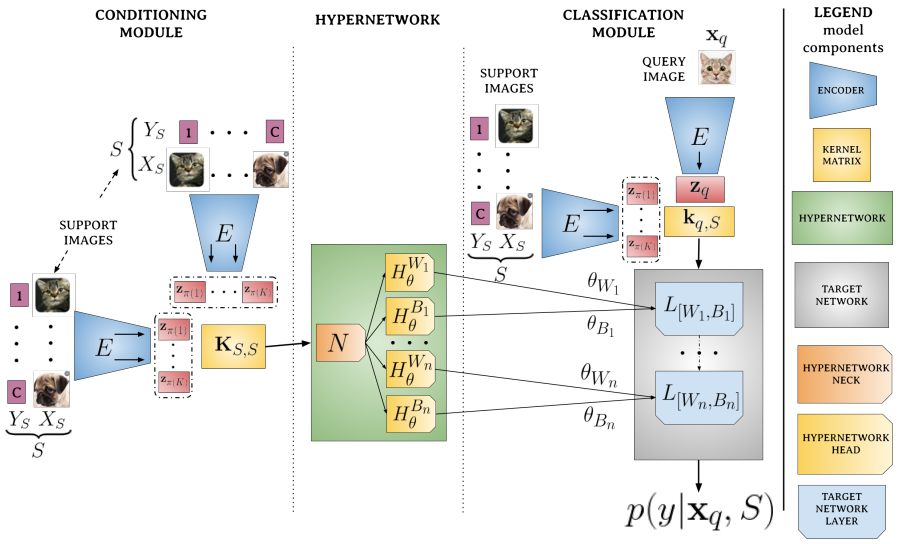


Figure 1. A detailed outline of the architecture of HyperShot, with the denoted flow of parameters generated by the hypernetwork heads.

solvers. In *International Conference on Learning Representations*, 2018.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[4] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters (2017). *arXiv preprint arXiv:1702.05373*, 2017.

[5] Chen Fan, Parikshit Ram, and Sijia Liu. Sign-maml: Efficient model-agnostic meta-learning by signsgd. *CoRR*, abs/2109.07497, 2021.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[7] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9537–9548, 2018.

[8] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting, 2018.

[9] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2018.

[10] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.

Table 2. The classification accuracy results for the inference tasks in the **mini-ImageNet** dataset in the 5-way (1-shot and 5-shot) scenarios. We consider models using the ResNet-12 backbone. The highest results are bold and second-highest in italic (the larger, the better).

Method	1-shot	5-shot
cosine classifier [3]	55.43 ± 0.81	77.18 ± 0.61
TADAM [26]	58.50 ± 0.30	76.70 ± 0.30
Shot-Free [33]	59.04	77.64
TPN [21]	59.46	75.65
MTL [37]	61.20 ± 1.80	75.50 ± 0.80
RFS-simple [39]	62.02 ± 0.63	79.64 ± 0.44
ProtoNet [35]	62.39 ± 0.21	80.53 ± 0.14
MetaOptNet [18]	62.64 ± 0.82	78.63 ± 0.46
MatchingNet [40]	63.08 ± 0.80	75.99 ± 0.60
MAML [6]	63.11 ± 0.92	–
PPA [28]	59.60	73.34
CAN [12]	63.85 ± 0.48	79.44 ± 0.34
NegMargin [20]	63.85 ± 0.81	81.57 ± 0.56
DeepEMD [47]	65.91 ± 0.82	*82.41 ± 0.56*
FEAT [45]	*66.78 ± 0.20*	82.05 ± 0.14
RENet [14]	**67.60 ± 0.44**	**82.58 ± 0.30**
HyperShot	59.12 ± 0.26	76.53 ± 0.22
HyperShot + adaptation	60.30 ± 0.31	76.21 ± 0.20

Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification, 2019.

[13] Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9122–9133, 2019.

[14] Dahyun Kang, Heeseung Kwon, Juhong Min, and Minsu Cho. Relational embedding for few-shot classification, 2021.

[15] Minyoung Kim and Timothy Hospedales. Gaussian process meta few-shot classifier learning via linear discriminant laplace approximation. *arXiv preprint arXiv:2111.05392*, 2021.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[17] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[18] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[19] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017.

[20] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification, 2020.

[21] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning, 2019.

[22] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[23] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

[24] Cuong Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Uncertainty in model-agnostic meta-learning using variational inference. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3090–3100, 2020.

[25] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[26] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[27] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

[28] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Yuille. Few-shot image recognition by predicting parameters from activations, 2017.

[29] Jathushan Rajasegaran, Salman H. Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Meta-learning the learning trends shared across tasks. *CoRR*, abs/2010.09291, 2020.

[30] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*, 32:113–124, 2019.

[31] Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2018.

[32] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[33] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training, 2020.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[35] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Table 3. The classification accuracy results for the inference tasks on **CUB** and **mini-ImageNet** datasets in the 1-shot and 5-shot settings. The highest results are in bold and second-highest in italic (the larger, the better). This is an expanded version of Table 1 from the main text.

	CUB		**mini-ImageNet**	
Method	1-shot	5-shot	1-shot	5-shot
ML-LSTM [32]	–	–	43.44 ± 0.77	60.60 ± 0.71
SNAIL [22]	–	–	45.10	55.20
LLAMA [10]	–	–	49.40 ± 1.83	
VERSA [9]	–	–	48.53 ± 1.84	67.37 ± 0.86
Amortized VI [9]	–	–	44.13 ± 1.78	55.68 ± 0.91
Meta-Mixture [13]	–	–	49.60 ± 1.50	64.60 ± 0.92
SimpleShot [42]	–	–	49.69 ± 0.19	66.92 ± 0.17
Feature Transfer [49]	46.19 ± 0.64	68.40 ± 0.79	39.51 ± 0.23	60.51 ± 0.55
Baseline++ [3]	61.75 ± 0.95	78.51 ± 0.59	47.15 ± 0.49	66.18 ± 0.18
MatchingNet [40]	60.19 ± 1.02	75.11 ± 0.35	48.25 ± 0.65	62.71 ± 0.44
ProtoNet [35]	52.52 ± 1.90	75.93 ± 0.46	44.19 ± 1.30	64.07 ± 0.65
RelationNet [38]	62.52 ± 0.34	78.22 ± 0.07	48.76 ± 0.17	64.20 ± 0.28
DKT + CosSim [27]	63.37 ± 0.19	77.73 ± 0.26	48.64 ± 0.45	62.85 ± 0.37
DKT + BNCosSim [27]	62.96 ± 0.62	77.76 ± 0.62	49.73 ± 0.07	64.00 ± 0.09
VAMPIRE [24]	–	–	51.54 ± 0.74	64.31 ± 0.74
PLATIPUS [7]	–	–	50.13 ± 1.86	–
ABML [31]	49.57 ± 0.42	68.94 ± 0.16	45.00 ± 0.60	–
OVE PG GP + Cosine (ML) [36]	63.98 ± 0.43	77.44 ± 0.18	50.02 ± 0.35	64.58 ± 0.31
OVE PG GP + Cosine (PL) [36]	60.11 ± 0.26	79.07 ± 0.05	48.00 ± 0.24	67.14 ± 0.23
Reptile [25]	–	–	49.97 ± 0.32	65.99 ± 0.58
R2-D2 [2]	–	–	48.70 ± 0.60	65.50 ± 0.60
VSM [48]	–	–	54.73 ± 1.60	68.01 ± 0.90
PPA [28]	–	–	54.53 ± 0.40	–
MAML [6]	56.11 ± 0.69	74.84 ± 0.62	45.39 ± 0.49	61.58 ± 0.53
MAML++ [1]	–	–	52.15 ± 0.26	68.32 ± 0.44
iMAML-HF [30]	–	–	49.30 ± 1.88	–
SignMAML [5]	–	–	42.90 ± 1.50	60.70 ± 0.70
Bayesian MAML [46]	55.93 ± 0.71		53.80 ± 1.46	64.23 ± 0.69
Unicorn-MAML [43]	–	–	54.89	–
Meta-SGD [19]	–	–	50.47 ± 1.87	64.03 ± 0.94
MetaNet [23]	–	–	49.21 ± 0.96	–
PAMELA [29]	–	–	53.50 ± 0.89	*70.51 ± 0.67*
FEAT [44]	$\mathbf{68.87 \pm 0.22}$	$\mathbf{82.90 \pm 0.15}$	*55.15 ± 0.20*	$\mathbf{71.61 \pm 0.16}$
DFSVLwF [8]	–	–	$\mathbf{56.20 \pm 0.86}$	–
HyperShot	65.27 ± 0.24	79.80 ± 0.16	52.42 ± 0.46	68.78 ± 0.29
HyperShot+ adaptation	*66.13 ± 0.26*	*80.07 ± 0.22*	53.18 ± 0.45	69.62 ± 0.20

[36] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. In *International Conference on Learning Representations*, 2020.

[37] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning, 2019.

[38] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[39] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need?, 2020.

[40] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[41] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Re-

Table 4. Hyperparameters

hyperparameter	**CUB**	**mini-ImageNet**	**mini-ImageNet → CUB**	**Omniglot → EMNIST**
kernel function	cosine similarity	cosine similarity	cosine similarity	cosine similarity
learning rate	0.001	0.001	0.001	0.001
hypernetwork's head layers no.	3	3	3	2
hypernetwork's neck layers no.	2	2	2	1
hypernetwork layers' hidden dim	4096	4096	4096	512
support embeddings aggregation	averaged	averaged	averaged	averaged
taskset size	1	1	1	1
target network layers no.	1	1	1	2
target network activation	ReLU	ReLU	ReLU	ReLU
adaptation epochs (if used)	10	10	10	10
adaptationlearning rate	0.0001	0.0001	0.0001	0.0001
optimizer	Adam	Adam	Adam	Adam
epochs no.	10000	10000	10000	2000

port CNS-TR-2011-001, California Institute of Technology, 2011.

[42] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[43] Han-Jia Ye and Wei-Lun Chao. How to train your maml to excel in few-shot classification, 2021.

[44] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

[45] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions, 2021.

[46] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[47] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning, 2020.

[48] Xiantong Zhen, Ying-Jun Du, Huan Xiong, Qiang Qiu, Cees Snoek, and Ling Shao. Learning to learn variational semantic memory. In *NeurIPS*, 2020.

[49] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.

Machine Vision and Applications (2023) 34:53
https://doi.org/10.1007/s00138-023-01403-4

ORIGINAL PAPER

The general framework for few-shot learning by kernel HyperNetworks

Marcin Sendera[1,4] · Marcin Przewięźlikowski[1,4,5] · Jan Miksa[1] · Mateusz Rajski[1] · Konrad Karanowski[2] · Maciej Zięba[2,3] · Jacek Tabor[1] · Przemysław Spurek[1]

Received: 17 March 2023 / Revised: 17 April 2023 / Accepted: 18 April 2023 / Published online: 20 May 2023

Abstract
Few-shot models aim at making predictions using a minimal number of labeled examples from a given task. The main challenge in this area is the *one-shot* setting, where only one element represents each class. We propose the general framework for few-shot learning via kernel HyperNetworks—the fusion of kernels and hypernetwork paradigm. Firstly, we introduce the classical realization of this framework, dubbed HyperShot. Compared to reference approaches that apply a gradient-based adjustment of the parameters, our models aim to switch the classification module parameters depending on the task's embedding. In practice, we utilize a hypernetwork, which takes the aggregated information from support data and returns the classifier's parameters handcrafted for the considered problem. Moreover, we introduce the kernel-based representation of the support examples delivered to hypernetwork to create the parameters of the classification module. Consequently, we rely on relations between the support examples' embeddings instead of the backbone models' direct feature values. Thanks to this approach, our model can adapt to highly different tasks. While such a method obtains very good results, it is limited by typical problems such as poorly quantified uncertainty due to limited data size. We further show that incorporating Bayesian neural networks into our general framework, an approach we call BayesHyperShot, solves this issue.

Keywords Few-shot learning · Meta-learning · HyperNetworks · Kernel methods · Bayesian neural networks

1 Introduction

Current artificial intelligence techniques cannot rapidly generalize from a few examples. This common inability stems from the fact that most deep neural networks must be trained on large-scale data. In contrast, humans can learn new tasks quickly by utilizing what they learned in the past. Few-shot learning models try to fill this gap by *learning how to learn* from a limited number of examples. Few-shot learning is the problem of making predictions based on a few labeled examples. The goal of few-shot learning is not to recognize a fixed set of labels but to quickly adapt to new tasks with a small amount of training data. After training, the model can classify new data using only a few training examples.

Marcin Sendera and Marcin Przewięźlikowski have contributed equally to this work.

Marcin Sendera
marcin.sendera@doctoral.uj.edu.pl

Marcin Przewięźlikowski
marcin.przewiezlikowski@doctoral.uj.edu.pl

Jan Miksa
jan.miksa@student.uj.edu.pl

Mateusz Rajski
mateusz.rajski@student.uj.edu.pl

Konrad Karanowski
254533@student.pwr.edu.pl

Maciej Zięba
maciej.zieba@pwr.edu.pl

Jacek Tabor
jacek.tabor@uj.edu.pl

Przemysław Spurek
przemyslaw.spurek@uj.edu.pl

[1] Faculty of Mathematics and Computer Science, Jagiellonian University, Lojasiewicza 6, Kraków, Poland

[2] Department of Artificial Intelligence, Wroclaw University of Science and Technology, Wyb. Wyspianskiego 27, Wrocław, Poland

[3] Tooploox, Wrocław, Poland

[4] Doctoral School of Exact and Natural Sciences, Jagiellonian University, Lojasiewicza 11, Kraków, Poland

[5] IDEAS NCBR, Chmielna 69, Warsaw, Poland

Two novel few-shot learning techniques have recently emerged. The first is based on the kernel methods and Gaussian processes [1–3]. The universal deep kernel has enough data to generalize well to unseen tasks without overfitting. The second technique makes use of the Hypernetworks [4–9], which allow to aggregate information from the support set and produce dedicated network weights for new tasks.

The above approaches give promising results but also have some limitations. Kernel-based methods are not flexible enough, since they use Gaussian processes on top of the models. Moreover, it is not trivial to use Gaussian processes for classification tasks. On the other hand, Hypernetworks must aggregate information from the support set, and it is hard to model the relation between classes as opposed to classical feature extraction.

This paper introduces a general framework that combines the Hypernetworks paradigm with kernel methods to realize a new strategy that mimics the human way of learning. First, we examine the entire support set and extract the information in order to distinguish objects of each class. Then, based on the relations between their features, we create the decision rules.

Kernel methods realize the first part of the process. For each of the few-shot tasks, we extract the features from the support set through the backbone architecture and calculate kernel values between them. Then we use a Hypernetwork architecture [3, 4]—a neural network that takes kernel representation and produces decision rules in the form of a classifier. In our framework, the Hypernetwork aggregates the information from the support set and produces weights or adaptation parameters of the target model dedicated to the specific task, classifying the query set.

The models we propose inherit the flexibility from Hypernetworks and the ability to learn the relation between objects from kernel-based methods.

We consider two alternative approaches to the realization of our framework—the classical one, which we dubbed HyperShot, and the Bayesian version called BayesHyperShot. Firstly, we started with the classical approach.

We perform an extensive experimental study of HyperShot by benchmarking it on various one-shot and few-shot image classification tasks. We find that HyperShot demonstrates high accuracy in all tasks, performing comparably or better than the other recently proposed methods. Moreover, HyperShot shows a strong ability to generalize, as evidenced by its performance on cross-domain classification tasks.

Unfortunately HyperShot, similar to other few-shot algorithms, suffers from limited data size, which may result in drawbacks such as poorly quantified uncertainty. To solve this problem, we extend our method by incorporating Bayesian neural network and Hypernetwork paradigms, which results in a Bayesian version of our general framework —BayesHyperShot. In practice, hypernetwork produces parameters of probability distribution on weights. In this paper, we consider Gaussian prior, but our framework can be used for modeling even more complex priors.

The contributions of this work are fourfold:

- In this paper, we propose a general framework that realizes the *learn how to learn* paradigm by modeling learning rules which are not based on gradient optimization and can produce completely different decision strategies.
- We propose a new approach to solve the few-shot learning problem by aggregating information from the support set by kernel methods and directly producing weights from the neural network dedicated to the query set.
- We propose HyperShot model, which combines the Hypernetworks paradigm with kernel methods classically, to produce the weights dedicated for each task.
- We show that our model can be generalized to Bayesian version BayesHyperShot, which allows for solving problems with poorly quantified uncertainty.

2 Hypernetworks for few-shot learning–a general framework

In this section, we present our general framework for utilizing the Hypernetworks and kernel-based methods for few-shot learning. In the beginning, we give a quick recap of the necessary background. Then, we introduce our framework by presenting the HyperShot. Finally, we show how to incorporate the Bayesian approach in this general framework and present the BayesHyperShot.

2.1 Background

Few-shot learning The terminology describing the few-shot learning setup is dispersive due to the colliding definitions used in the literature. For a unified taxonomy, we refer the reader to [10, 11]. Here, we use the nomenclature derived from the meta-learning literature, which is the most prevalent at the time of writing. Let:

$$\mathcal{S} = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^{L} \tag{1}$$

be a support set containing input–output pairs, with L examples with the equal class distribution. In the *one-shot* scenario, each class is represented by a single example, and $L = K$, where K is the number of the considered classes in the given task. For *few-shot* scenarios, each class usually has from 2 to 5 representatives in the support set $\mathcal{S}$. Let:

$$\mathcal{Q} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M} \tag{2}$$

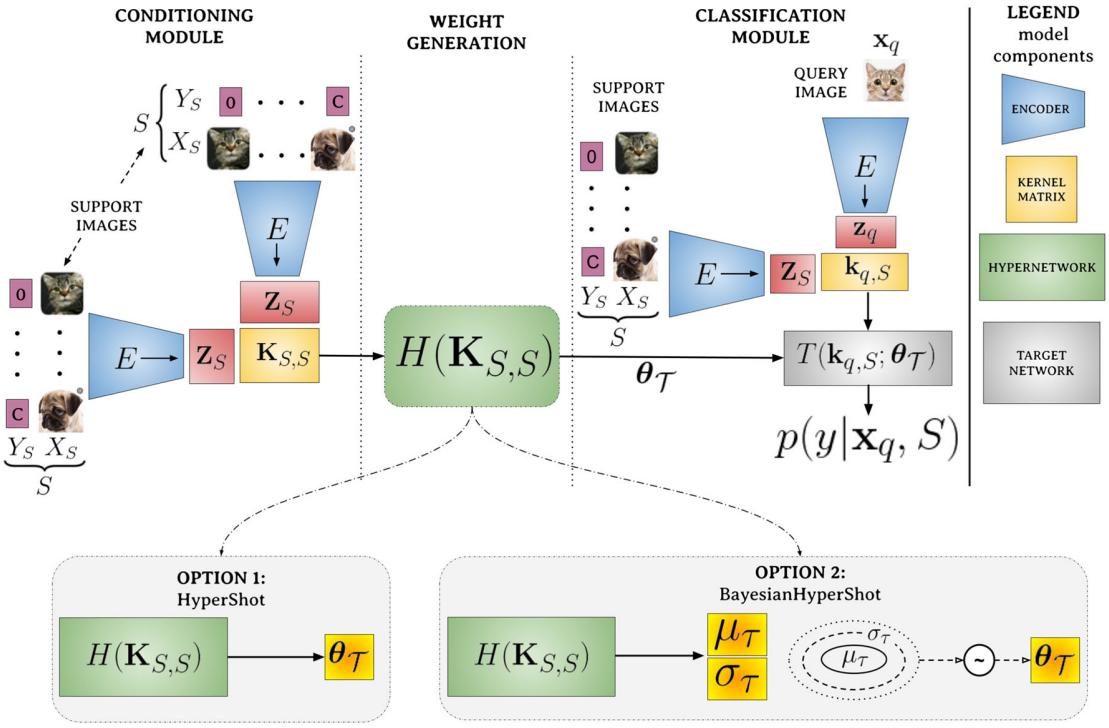


Fig. 1 General architecture of our framework. First, the examples from a support set are sorted according to the corresponding class labels and transformed by encoding network $E(\cdot)$ to obtain the matrix of ordered embeddings of the support examples, $\mathbf{Z}_S$. The low-dimensional representations stored in $\mathbf{Z}_S$ are further used to compute kernel matrix $\mathbf{K}_{S,S}$. The values of the kernel matrix are passed to the hypernetwork $H(\cdot)$ that creates the parameters $\boldsymbol{\theta}_\tau$ for the target classification module $T(\cdot)$. We identify two options for generating $\boldsymbol{\theta}_\tau$: (**i**) HyperShot—hypernetwork H generates $\boldsymbol{\theta}_\tau$ straightforwardly; (**ii**) BayesHyperShot—H generates parameters μ_τ and σ_τ of a Gaussian distribution, from which $\boldsymbol{\theta}_\tau$ can then be sampled. The query image $\mathbf{x}$ is processed by encoder $E(\cdot)$, and the vector of kernel values $\mathbf{k}_{\mathbf{x},S}$ is calculated between query embedding $\mathbf{z}_\mathbf{x}$ and the corresponding representations of support examples, $\mathbf{Z}_S$. The kernel vector $\mathbf{k}_{\mathbf{x},S}$ is further passed to target model $T(\cdot)$ to obtain the probability distribution for the considered classes

be a query set (sometimes referred to in the literature as a target set), with M examples, where M is typically one order of magnitude greater than K. For ease of notation, the support and query sets are grouped in a task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\}$. During the training stage, the models for few-shot applications are fed by randomly selected examples from training set $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^N$, defined as a collection of such tasks.

During the inference stage, we consider task $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$, where $\mathcal{S}_*$ is a support set with the known class values for a given task, and $\mathcal{X}_*$ is a set of query (unlabeled) inputs. The goal is to predict the class labels for query inputs $\mathbf{x} \in \mathcal{X}_*$, assuming support set $\mathcal{S}_*$ and using the model trained on $\mathcal{D}$.

Hypernetwork In the canonical work [4], hypernetworks are defined as neural models that generate weights for a separate target network solving a specific task. The authors aim to reduce the number of trainable parameters by designing a hypernetwork with a smaller number of parameters than the target network. Making an analogy between hypernetworks and generative models, the authors of [12] use this mechanism to generate a diverse set of target networks approximating the same function.

2.2 HyperShot overview

We introduce HyperShot—a model that utilizes hypernetworks for few-shot problems. The main idea of the proposed approach is to predict the values of the parameters for a classification network that makes predictions on the query images given the information extracted from support examples for a given task. Thanks to this approach, we can switch the classifier's parameters between completely different tasks based on the support set. The information about the current task is extracted from the support set using a parameterized kernel function that operates on embedding space. Thanks to this approach, we use relations among the support examples instead of taking the direct values of the embedding values as an input to the hypernetwork. Consequently, this approach is robust to the embedding values for new tasks far from the feature regions observed during training. The classification of the query image is also performed using the kernel values calculated with respect to the support set.

The architecture of HyperShot is provided in Fig. 1. We aim to predict the class distribution $p(\mathbf{y} \mid S, \mathbf{x})$, given a query image $\mathbf{x}$ and set of support examples $S = \{(\mathbf{x}_l, y_l)\}_{l=1}^K$.

First, all images from the support set are grouped by their corresponding class values. Next, each of the images $\mathbf{x}_l$ from the support set is transformed using encoding network $E(\cdot)$, which creates low-dimensional representations of the images, $E(\mathbf{x}_l) = \mathbf{z}_l$. The constructed embeddings are sorted according to class labels and stored in the matrix $\mathbf{Z}_S = [\mathbf{z}_{\pi(1)}, \ldots, \mathbf{z}_{\pi(K)}]^{\mathrm{T}}$, where $\pi(\cdot)$ is the bijective function, that satisfies $y_{\pi(l)} \leq y_{\pi(k)}$ for $l \leq k$.

In the next step, we calculate the kernel matrix $\mathbf{K}_{S,S}$, for vector pairs stored in rows of $\mathbf{Z}_S$. To achieve this, we use the parametrized kernel function $k(\cdot, \cdot)$, and calculate $k_{i,j}$ element of matrix $\mathbf{K}_{S,S}$ in the following way:

$$k_{i,j} = k(\mathbf{z}_{\pi(i)}, \mathbf{z}_{\pi(j)}). \tag{3}$$

The kernel matrix $\mathbf{K}_{S,S}$ represents the extracted information about the relations between support examples for a given task. The matrix $\mathbf{K}_{S,S}$ is further reshaped to the vector format and delivered to the input of the hypernetwork $H(\cdot)$. The role of the hypernetwork is to provide the parameters $\boldsymbol{\theta}_T$ of target model $T(\cdot)$ responsible for the classification of the query object. Thanks to that approach, we can switch between the parameters for entirely different tasks without moving via the gradient-controlled trajectory, like in some reference approaches like MAML.

We base the architecture of the Hypernetwork on a multilayer perceptron (MLP). Specifically, the HyperNetwork first processes the kernel matrix $\mathbf{K}_{S,S}$ through an MLP (dubbed the "neck"), whose output is then processed by "heads"—separate MLPs dedicated to producing the weights of each target network layer—see Fig. 4 for a visualization. We summarize the parameters of Hypernetworks and target networks used in our experiments in Table 8.

The query image $\mathbf{x}$ is classified in the following manner. First, the input image is transformed to low-dimensional feature representation $z_{\mathbf{x}}$ by encoder $E(\mathbf{x})$. Further, the kernel vector $\mathbf{k}_{\mathbf{x},S}$ between the query embedding and sorted support vectors $\mathbf{Z}_S$ is calculated in the following way:

$$\mathbf{k}_{\mathbf{x},S} = [k(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\pi(1)}), \ldots, k(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\pi(K)})]^{\mathrm{T}}. \tag{4}$$

The vector $\mathbf{k}_{\mathbf{x},S}$ is further provided on the input of target model $T(\cdot)$ that is using the parameters $\boldsymbol{\theta}_T$ returned by hypernetwork $H(\cdot)$. The target model returns the probability distribution $p(\mathbf{y} \mid S, \mathbf{x})$ for each class considered in the task.

The function $\pi(\cdot)$ enforces some ordering of the input delivered to $T(\cdot)$. Practically, any other permutation of the classes for the input vector $\mathbf{k}_{\mathbf{x},S}$. In such a case, the same permutation should be applied to rows and columns of $\mathbf{K}_{S,S}$. As a consequence, the hypernetwork is able to produce the dedicated target parameters for each of the possible permutations. Although this approach does not guarantee the permutation invariance for real-life scenarios, thanks to dedicated parameters for any ordering of the input, it should be satisfied for major cases.

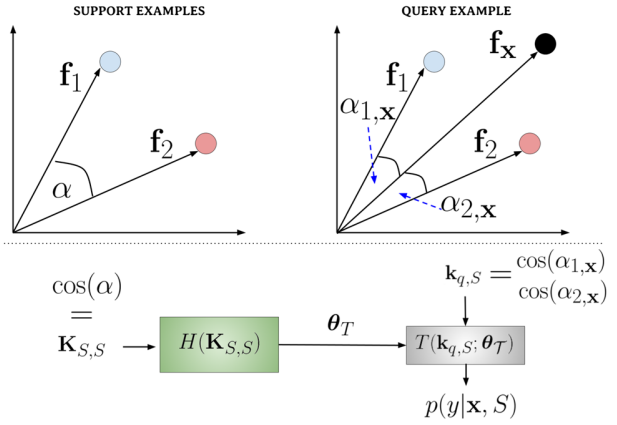


Fig. 2 Simple 2D example illustrating the application of cosine kernel for HyperShot. We consider the two support examples from different classes represented by vectors $\mathbf{f_1}$ and $\mathbf{f_2}$. For this simple scenario, the input of hypernetwork is represented simply by the cosine of α, which is an angle between vectors $\mathbf{f}_1$ and $\mathbf{f}_2$. We aim at classifying the query example $\mathbf{x}$ represented by a vector $\mathbf{f}_{\mathbf{x}}$. Considering our approach, we deliver to the target network $T(\cdot)$ the cosine values of angles between first ($\alpha_{\mathbf{x},1}$) and second ($\alpha_{\mathbf{x},2}$) support vectors and classify the query example using the weights $\boldsymbol{\theta}_T$ created by hypernetwork $H(\cdot)$ from $\cos\alpha$ (remaining components on the diagonal of $\mathbf{K}_{S,S}$ are constant for cosine kernel)

2.3 Kernel function

One of the key components of our approach is a kernel function $k(\cdot, \cdot)$. In this work, we consider the dot product of the transformed vectors given by:

$$k(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{f}(\mathbf{z}_1)^{\mathrm{T}} \mathbf{f}(\mathbf{z}_2), \tag{5}$$

where $\mathbf{f}(\cdot)$ can be a parametrized transformation function, represented by MLP model, or simply an identity operation, $\mathbf{f}(\mathbf{z}) = \mathbf{z}$. In Euclidean space, this criterion can be expressed as $k(\mathbf{z}_1, \mathbf{z}_2) = ||\mathbf{f}(\mathbf{z}_1)|| \cdot ||\mathbf{f}(\mathbf{z}_2)|| \cos\alpha$, where α is an angle between vectors $\mathbf{f}(\mathbf{z}_1)$ and $\mathbf{f}(\mathbf{z}_2)$. The main feature of this function is that it considers the vectors' norms, which can be problematic for some tasks that are outliers regarding the representations created by $\mathbf{f}(\cdot)$. Therefore, we consider in our experiments also the cosine kernel function given by:

$$k_c(\mathbf{z}_1, \mathbf{z}_2) = \frac{\mathbf{f}(\mathbf{z}_1)^{\mathrm{T}} \mathbf{f}(\mathbf{z}_2)}{\mathbf{f}(\mathbf{z}_1) \cdot \mathbf{f}(\mathbf{z}_2)}, \tag{6}$$

that represents the normalized version dot product. Considering the geometrical representation, $k_c(\mathbf{z}_1, \mathbf{z}_2)$ can be expressed as $\cos\alpha$ (see the example given by Fig. 2). The support set is represented by two examples from different classes, $\mathbf{f}_1$ and $\mathbf{f}_2$. The target model parameters $\boldsymbol{\theta}_T$ are cre-

Algorithm 1 HyperShot - training and prediction functions

Require: Training set $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$, and $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$ test task.
Parameters: $\boldsymbol{\theta}_H$ - parameters , $\boldsymbol{\theta}_k$ - kernel parameters, and $\boldsymbol{\theta}_E$ - encoder parameters
Hyperparameters: N_{train} - number of training iterations, N_{tune} number of tuning iterations, α- step size.

function TRAIN($\mathcal{D}$, α, N_{train}, $\boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E$)
  while $n \leq N_{train}$ **do**
    Sample task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\} \sim \mathcal{D}$
    Assign support $\mathcal{S} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M}$
    $L = -\sum_{m=1}^{M}\sum_{k=1}^{K} y_m^k \log p(y_m^k \mid \mathcal{S}_i, \mathbf{x}_m, \boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E)$
    Update: $\boldsymbol{\theta}_E \leftarrow \boldsymbol{\theta}_E - \alpha\nabla_{\theta_E}\mathcal{L}$,
      $\boldsymbol{\theta}_H \leftarrow \boldsymbol{\theta}_H - \alpha\nabla_{\theta_H}\mathcal{L}$,
      $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \alpha\nabla_{\theta_k}\mathcal{L}$
    $n = n + 1$
  end while
  return $\boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E$
end function
function PREDICT($\mathcal{T}_*$, α, N_{tune}, $\boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E$)
  Create tuning task: $\mathcal{T}_t = \{\mathcal{S}_*, \mathcal{S}_*\}$
  Adapt $\hat{\boldsymbol{\theta}}_H, \hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_E$ = TRAIN($\mathcal{T}_t$, α, N_{tune}, $\boldsymbol{\theta}_H, \boldsymbol{\theta}_k, \boldsymbol{\theta}_E$)
  for each $\mathbf{x} \in \mathcal{X}_*$ **do**
    return $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathcal{S}_*, \mathbf{x}, \hat{\boldsymbol{\theta}}_H, \hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_E)$
  end for
end function

ated based only on the cosine value of the angle between vectors $\mathbf{f}_1$ and $\mathbf{f}_2$. During the classification stage, the query example is represented by $\mathbf{f}_\mathbf{x}$, and the classification is applied on the cosine values of angles between $\mathbf{f}_\mathbf{x}$ and $\mathbf{f}_1$, and $\mathbf{f}_\mathbf{x}$ and $\mathbf{f}_2$, respectively.

2.4 Training and prediction

The training procedure assumes the following parametrization of the model components. The encoder $E := E_{\boldsymbol{\theta}_E}$ is parametrized by $\boldsymbol{\theta}_E$, the hypernetwork $H = H_{\boldsymbol{\theta}_H}$ by $\boldsymbol{\theta}_H$ and the kernel function k by $\boldsymbol{\theta}_k$. We assume that training set $\mathcal{D}$ is represented by tasks $\mathcal{T}_i$ composed of support $\mathcal{S}_i$ and query $\mathcal{Q}_i$ examples. The training is performed by optimizing the cross-entropy criterion:

$$L = \sum_{\mathcal{T}_i \in \mathcal{D}} l_{\mathcal{T}_i} = \sum_{\mathcal{T}_i \in \mathcal{D}} \sum_{m=1}^{M} \sum_{k=1}^{K} y_{i,m}^k - \log p(y_{i,m}^k \mid \mathcal{S}_i, \mathbf{x}_{i,m}), \tag{7}$$

where $(\mathbf{x}_{i,n}, \mathbf{y}_{i,n})$ are examples from query set $\mathcal{Q}_i$, where $Q_i = \{(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})\}_{m=1}^{M}$. The distribution for currently considered classes $p(\mathbf{y} \mid \mathcal{S}, \mathbf{x})$ is returned by target network T of HyperShot. During the training, we jointly optimize the parameters $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_E$, minimizing the L loss.

During the inference stage, we consider the task $\mathcal{T}_*$, composed of a set of labeled support examples $\mathcal{S}_*$ and a set of unlabeled query examples represented by input values $\mathcal{X}_*$ that the model should classify. We can simply take the probability values $p(\mathbf{y} \mid \mathcal{S}_*, \mathbf{x})$ assuming the given support set $\mathcal{S}_*$ and single query observation $\mathbf{x}$ from $\mathcal{X}_*$, using the model with trained parameters $\boldsymbol{\theta}_H$, $\boldsymbol{\theta}_k$, and $\boldsymbol{\theta}_E$. However, we observe that slightly better results are obtained while adapting the model's parameters on the considered task. We do not have access to labels for query examples. Therefore, we imitate the query set for this task simply by taking support examples and creating the adaptation task $\mathcal{T}_i = \{\mathcal{S}_*, \mathcal{S}_*\}$ and updating the parameters of the model using several gradient iterations. The detailed presentation o training and prediction procedures is provided by Algorithm 1.

2.5 Adaptation to few-shot scenarios

The proposed approach uses the ordering function $\pi(\cdot)$ that keeps the consistency between support kernel matrix $\mathbf{K}_{S,S}$ and the vector of kernel values $\mathbf{k}_{\mathbf{x},S}$ for query example $\mathbf{x}$. For few-shot scenarios, each class has more than one representative in the support set. As a consequence, there are various possibilities to order the feature vectors in the support set inside the considered class. To eliminate this issue, we follow [8] and propose to apply the aggregation function to the embeddings $\mathbf{z}$ considering the support examples from the same class. Thanks to this approach, the kernel matrix is calculated based on the aggregated values of the latent space of encoding network E, making our approach independent of the ordering among the embeddings from the same class. In experimental studies, we examine the quality of *mean* aggregation operation (averaged) against simple class-wise concatenation of the embeddings (fine-grained) in ablation studies.

2.6 Extension to Bayesian setting—BayesHyperShot

Finally, we introduce the Bayesian extension of HyperShot, where the distribution over the parameters is represented by Gaussian prior:

$$p(\boldsymbol{\theta}_\mathcal{T}) = \mathcal{N}(\boldsymbol{\theta}_\mathcal{T} \mid 0, I).$$

Thanks to that assumption, the target model $\mathcal{T}$ serves probabilistic and can be used as a Bayesian network during inference.

In order to achieve that, we postulate to use of variational amortized posterior:

$$q(\boldsymbol{\theta}_\mathcal{T} \mid \mathcal{S}_i, \boldsymbol{\theta}_H) = \mathcal{N}(\boldsymbol{\theta}_\mathcal{T} \mid \mu_{\boldsymbol{\theta}_H}(\mathcal{S}_i), \sigma_{\boldsymbol{\theta}_H}(\mathcal{S}_i)),$$

where the Gaussian parameters $\mu(\mathcal{S}_i)$ and $\sigma(\mathcal{S}_i)$ are returned by hypernetwork $H_{\boldsymbol{\theta}_H}$ for a given support set $\mathcal{S}_i$, i.e., $(\mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)) = H_{\boldsymbol{\theta}_H}(\mathcal{S}_i, \boldsymbol{\theta}_H)$. Compared to the basic HyperShot model (option 1 Fig. 1) that predicts deterministic

target weight, the proposed extension delivers the distribution over parameters for a given support set option 2 Fig. 1).

We train the BayesHyperShot using the procedure given by Algorithm 1 with the modified training objective, given by:

$$\mathcal{L}_B = \sum_{\mathcal{T}_i \in \mathcal{D}} \left[\frac{1}{P} \sum_{p=1}^{P} \left[l_{\mathcal{T}_i}(\boldsymbol{\theta}^p_{\mathcal{T}_i}) - \gamma KL(q(\boldsymbol{\theta}^p_{\mathcal{T}_i} \mid \mathcal{S}_i, \boldsymbol{\theta}_H) \mid \mathcal{N}(0, I)) \right] \right],$$

where $\boldsymbol{\theta}^1_{\mathcal{T}_i}, \ldots, \boldsymbol{\theta}^P_{\mathcal{T}_i} \sim q(\boldsymbol{\theta}_{\mathcal{T}} \mid \mathcal{S}_i, \boldsymbol{\theta}_H)$ are the target parameters sampled by variational posterior modeled using the hypernetwork, $l_{\mathcal{T}_i}(\boldsymbol{\theta}^p_{\mathcal{T}_i})$ is cross-entropy loss function calculated using target model with weights $\boldsymbol{\theta}^p_{\mathcal{T}_i}$ and $KL(\cdot, \cdot)$ is Kullback–Leibler divergence between the posterior and standard Gaussian. In order to stabilize the training procedure, we use hyperparameter γ that controls the trade-off between cross-entropy loss and a regularization term. During training, we apply an annealing scheme [13], for which γ grows from zero to a fixed constant during training. The final value $\gamma_{\max}$ is a hyperparameter of the model.

During the inference stage, we sample the set of target parameters $\boldsymbol{\theta}^1_{\mathcal{T}_i}, \ldots, \boldsymbol{\theta}^P_{\mathcal{T}_i} \sim q(\boldsymbol{\theta}_{\mathcal{T}} \mid \mathcal{S}_i, \boldsymbol{\theta}_H)$. The final prediction is made simply by averaging among sampled target models, $\frac{1}{P} \sum_{p=1}^{P} p(\mathbf{y} \mid \mathcal{S}_*, \mathbf{x}, \hat{\boldsymbol{\theta}}_H, \hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_E, \boldsymbol{\theta}^p_{\mathcal{T}_i})$.

3 Related work

In recent years, various meta-learning methods [14–16] have been proposed to tackle the problem of few-shot learning. The various meta-learning architectures for few-shot learning can be roughly categorized into several groups, which we below divide into non-Bayesian and Bayesian families of approaches.

3.1 Non-Bayesian approaches

In non-Bayesian few-shot learning, the goal is typically to learn a set of parameters that can be used to classify new examples based on a limited amount of training data.

Transfer learning [17] is a simple yet effective baseline procedure for few-shot learning which consists of pre-training the neural network and a classifier on all of the classes available during meta-training. During meta-validation, the classifier is then fine-tuned to the novel tasks. In [10], the authors proposed Baseline++, an extension of this idea that uses cosine distance between the examples.

Metric-based methods meta-learn a deep representation with a metric in feature space, such that distance between examples from the support and query set with the same class have a small distance in such space. Some of the earliest works exploring this notion are matching networks [18] and prototypical networks [19], which form *prototypes* based on embeddings of the examples from the support set in the learned feature space and classify the query set based on the distance to those prototypes. Numerous subsequent works aim to improve the expressiveness of the prototypes through various techniques. Oreshkin et al. [20] achieve this by conditioning the network on specific tasks, thus making the learned space task-dependent. Hu [21] transform embeddings of support and query examples in the feature space to make their distributions closer to Gaussian. Sung et al. [22] propose Relation Nets, which learn the metric function instead of using a fixed one, such as Euclidean or cosine distance.

Optimization-based methods follow the idea of an optimization process over support set within the meta-learning framework like MetaOptNet [23], Meta-SGD [24] or model-agnostic meta-learning (MAML) [25] and its extensions [26–31]. Those techniques aim to train general models, which can adapt their parameters to the support set at hand in a small number of gradient steps. Similar to such techniques, HyperShot (the classical realization of our framework) also aims to produce task-specific models but utilizes a hypernetwork instead of optimization to achieve that goal.

Hypernetworks-based methods [4] have been proposed as a solution to few-shot learning problems in a number of works but have not been researched as widely as the approaches mentioned above. Multiple works proposed various variations of hypernetworks that predict a shallow classifier's parameters given the support examples [5, 32, 33]. More recently, [7–9] explored generating all of the parameters of the target network with a transformer-based hypernetwork, but found that for larger target networks, it is sufficient to generate only the parameters of the final classification layer. A particularly effective approach is to use transformer-based hypernetworks as set-to-set functions which make the generated classifier more discriminative [6]. A key characteristic of the above approaches is that during inference, the hypernetwork predicts weights responsible for classifying each class independently, based solely on the examples of that class from the support set. This property makes such solutions agnostic to the number of classes in a task, useful in practical applications. However, it also means that the hypernetwork does not take advantage of the inter-class differences in the task at hand.

In contrast, models in our framework (in particular HyperShot) exploit those differences by utilizing kernels, which helps improve its performance.

3.2 Bayesian approaches

There are many few-shot learning methods that utilizes Bayesian approach to estimate the parameters of a model. We grouped them in three categories:

Bayesian optimization-based methods reformulate MAML as a hierarchical Bayesian model [34–39]. Contrary to this group of methods, the BayesHyperShot does not utilize a bi-level optimization scheme, such as MAML. Moreover, we look at the adaptation of the target network's weights not at the optimization process itself.

Probabilistic weight generation methods focus on predicting a distribution over the parameters suitable for the given task [40]. Similarly, our BayesHyperShot also predicts the probability over the parameters of the target network performing the few-shot classification. The key difference is that in BayesHyperShot, the target network combines such an approach with a kernel mechanism.

Gaussian processes-based methods [41] possess many properties useful in few-shot learning, such as natural robustness to the limited amounts of data and the ability to estimate uncertainty. When combined with meta-learned deep kernels, In [1], Gaussian processes were demonstrated to be a suitable tool for few-shot regression and classification, dubbed deep kernel transfer (DKT). The assumption that such a universal deep kernel has enough data to generalize well to unseen tasks has been challenged in subsequent works. [3] introduced a technique of learning dense Gaussian processes by inducing variables. This approach achieves substantial performance improvement over the alternative methods. Similarly, Bayesian version of our model (BayesHyperShot) also depends on learning a model that estimates task-specific functions' parameters. However, BayesHyperShot employs a hypernetwork instead of a Gaussian process to achieve that goal.

4 Experiments

In the typical few-shot learning setting, making a valuable and fair comparison between proposed models is often complicated because of the existence of significant differences in architectures and implementations of known methods. In order to limit the influence of the deeper backbone (feature extractor) architectures, we follow the unified procedure proposed by [10].

In this section, we describe the experimental analysis and performance of the proposed methods—HyperShot and BayesHyperShot—in a large variety of few-shot benchmarks. Specifically, we consider both classification (see Sect. 4.1) and cross-domain adaptation (see Sect. 4.2) tasks. Whereas the classification problems are focused on the most typical few-shot applications, the latter cross-domain benchmarks check the ability of the models to adapt to out-of-distribution tasks. We compare the classical realization of our framework (HyperShot) against the non-Bayesian approaches and the Bayesian version (BayesHyperShot) against Bayesian approaches only. We limit our comparison to models designed for the standard inductive few-shot setting. Notably, we exclude models which utilize anything besides the support set for fitting to the task at hand, such as methods that are transductive [42] or which utilize additional unlabeled data [43]. Additionally, in Sect. 4.3 we conduct an experiment showing the uncertainty estimation via BayesHyperShot. Finally, we perform an ablation study of the possible adaptation procedures of HyperShot to few-shot scenarios, as well as architectural choices—presented in Sect. 4.4.

In all of the reported experiments, the tasks consist of 5 classes (5 ways) and 1 or 5 support examples (1 or 5 shots). Unless indicated otherwise, all compared models use a known and widely utilized backbone consisting of four convolutional layers (each consisting of a 2D convolution, a batch-norm layer, and a ReLU nonlinearity; each layer consists of 64 channels) [10] and have been trained from scratch.

We report the performance of two versions of our general framework—classical (HyperShot) and Bayesian (BayesHyperShot). Note that each of them has two variants:

- HyperShot/BayesHyperShot–models generated by the hypernetworks for each task.
- HyperShot/BayesHyperShot + adaptation–models generated by hypernetworks adapted to the support examples of each task for 10 training steps.[1]

In all cases, we observe a modest performance boost thanks to adapting the hypernetwork. Comprehensive details and hyperparameters for each training procedure are reported in Appendices B and C.

4.1 Classification

Firstly, we consider a classical few-shot learning scenario, where all the classification tasks (both training and inference) come from the same dataset. The main aim of the proposed classification experiments is to find the ability of the few-shot models to adapt to never-seen tasks from the same data distribution.

We benchmark the performance of our models and other methods on two challenging and widely considered datasets: Caltech-USCD Birds (CUB) [44] and mini-ImageNet [45]. The following experiments are in the most popular setting, 5 ways, consisting of 5 random classes. In all experiments, the query set of each task consists of 16 samples for each class (80 in total).

HyperShot We start with a non-Bayesian perspective and compare HyperShot to a vast pool of state-of-the-art algo-

[1] In the case of the adapted hypernetworks, we tune a copy of the hypernetwork on the support set separately for each validation task. This way, we ensure that our model does not take unfair advantage of the validation tasks.

Table 1 Classification accuracy results for the non-Bayesian approaches

Method	CUB		Mini-ImageNet	
	One-shot	Five-shot	One-shot	Five-shot
Feature transfer [17]	46.19 ± 0.64	68.40 ± 0.79	39.51 ± 0.23	60.51 ± 0.55
Baseline++ [10]	61.75 ± 0.95	78.51 ± 0.59	47.15 ± 0.49	66.18 ± 0.18
ProtoNet [19]	52.52 ± 1.90	75.93 ± 0.46	44.19 ± 1.30	64.07 ± 0.65
RelationNet [22]	62.52 ± 0.34	78.22 ± 0.07	48.76 ± 0.17	64.20 ± 0.28
FO-MAML [26]	–	–	48.70 ± 1.84	63.11 ± 0.92
Reptile [26]	–	–	49.97 ± 0.32	65.99 ± 0.58
FEAT [6]	$\mathbf{68.87 \pm 0.22}$	$\mathbf{82.90 \pm 0.15}$	$\mathbf{55.15 \pm 0.20}$	$\mathbf{71.61 \pm 0.16}$
MAML [25]	56.11 ± 0.69	74.84 ± 0.62	45.39 ± 0.49	61.58 ± 0.53
MAML++ [27]	–	–	52.15 ± 0.26	68.32 ± 0.44
iMAML-HF [30]	–	–	49.30 ± 1.88	–
SignMAML [28]	–	–	42.90 ± 1.50	60.70 ± 0.70
Unicorn-MAML [29]	–	–	54.89	–
Meta-SGD [24]	–	–	50.47 ± 1.87	64.03 ± 0.94
PAMELA [46]	–	–	*53.50 ± 0.89*	*70.51 ± 0.67*
HyperMAML [31]	66.11 ± 0.28	78.89 ± 0.19	51.84 ± 0.57	66.29 ± 0.43
HyperShot [47]	65.27 ± 0.24	79.80 ± 0.16	52.42 ± 0.46	68.78 ± 0.29
HyperShot + adaptation [47]	*66.13 ± 0.26*	*80.07 ± 0.22*	53.18 ± 0.45	69.62 ± 0.20

We consider the inference tasks on *CUB* and *mini-ImageNet* datasets in the one-shot and five-shot settings. The highest results are in bold and the second-highest in italic (the larger, the better)

rithms, including the canonical methods (like matching networks [18], prototypical networks [19], MAML [25], and its extensions) as well as the recently popular: Unicorn-MAML [29] PAMELA [46].

We consider the more challenging one-shot classification task, as well as the five-shot setting and report the results in Table 1.

In the one-shot scenario, HyperShot achieves the second-best accuracy in the CUB dataset with adapting procedure (66.13% with adapting, 65.27% without) and performs better than any other model, except for FEAT [6] (68.87%). In the mini-ImageNet dataset, our approach is among the top approaches (53.18%), slightly losing with FEAT [6] (55.15%). Considering the five-shot scenario, HyperShot is the second-best model achieving 80.07% in the CUB dataset and 69.62% in the mini-ImageNet, whereas the best model, FEAT [6], achieves 82.90% and 71.61% on the mentioned datasets, respectively.

The obtained results clearly show that HyperShot achieves results comparable to state-of-the-art non-Bayesian models on the standard set of few-shot classification settings.

BayesHyperShot Then, we compare the Bayesian version of our general framework—BayesHyperShot against the state-of-the-art Bayesian methods. These Bayesian models are mostly built upon the Gaussian Processes framework (like DKT [1]). We consider the more challenging one-shot classification task, as well as the five-shot setting and report the results in Table 2. In the one-shot scenario, BayesHyperShot achieves the third-best accuracy in the CUB dataset despite the adapting procedure (66.30% in both cases) and performs similarly to the best model—BayesHMAML [36] (respectively, 66.92% with adaptation and 66.57% without). In the mini-ImageNet dataset, our Bayesian approach is among the top methods (51.11%), slightly losing with Bayesian MAML [34] (53.80%) and BayesHMAML (52.69%).

Considering the five-shot scenario, BayesHyperShot is the best model achieving 80.60% in the CUB dataset and 67.21% in the mini-ImageNet, whereas the most significant competitor, BayesHMAML [36] achieves 80.47% and 68.24% on the mentioned datasets, respectively.

According to the results, the performance of BayesHyperShot is comparable to or better than other state-of-the-art Bayesian models on the standard set of few-shot classification settings.

4.2 Cross-domain adaptation

In the cross-domain adaptation setting, the models are evaluated on tasks coming from a different distribution than the one they had been trained on. Therefore, such a task is more challenging than standard classification and is a plausible indicator of a model's ability to generalize. In order to benchmark the performance of our framework in cross-domain adaptation, we merge data from two datasets so that the training fold is drawn from the first dataset and validation and testing fold—from another one. Specifically, we test HyperShot on two cross-domain classification tasks:

Table 2 Classification accuracy results for the Bayesian approaches

Method	CUB		Mini-ImageNet	
	One-shot	Five-shot	One-shot	Five-shot
LLAMA [35]	–	–	49.40 ± 1.83	–
VERSA [40]	–	–	48.53 ± 1.84	67.37 ± 0.86
Amortized VI [40]	–	–	44.13 ± 1.78	55.68 ± 0.91
Meta-Mixture [39]	–	–	49.60 ± 1.50	64.60 ± 0.92
DKT + BNCosSim [1]	62.96 ± 0.62	77.76 ± 0.62	49.73 ± 0.07	64.00 ± 0.09
VAMPIRE [38]	–	–	51.54 ± 0.74	64.31 ± 0.74
ABML [37]	49.57 ± 0.42	68.94 ± 0.16	45.00 ± 0.60	–
Bayesian MAML [34]	55.93 ± 0.71		$\mathbf{53.80 \pm 1.46}$	64.23 ± 0.69
BayesHMAML [36]	$\mathit{66.57 \pm 0.47}$	79.86 ± 0.31	52.54 ± 0.46	$\mathit{67.39 \pm 0.35}$
BayesHMAML + adaptation [36]	$\mathbf{66.92 \pm 0.38}$	$\mathit{80.47 \pm 0.38}$	$\mathit{52.69 \pm 0.38}$	$\mathbf{68.24 \pm 0.47}$
BayesHyperShot	66.30 ± 0.42	80.43 ± 0.34	50.81 ± 0.33	66.51 ± 0.71
BayesHyperShot + adaptation	66.30 ± 0.42	$\mathbf{80.60 \pm 0.37}$	51.11 ± 0.35	67.21 ± 0.72

We consider the inference tasks on *CUB* and *mini-ImageNet* datasets in the one-shot and five-shot settings. The highest results are in bold and the second-highest in italic (the larger, the better)

Table 3 Classification accuracy results for the non-Bayesian approaches

Method	Omniglot→EMNIST		mini-ImageNet→CUB	
	One-shot	Five-shot	One-shot	Five-shot
Feature transfer [17]	64.22 ± 1.24	86.10 ± 0.84	32.77 ± 0.35	50.34 ± 0.27
Baseline++ [10]	56.84 ± 0.91	80.01 ± 0.92	$\mathit{39.19 \pm 0.12}$	57.31 ± 0.11
ProtoNet [19]	72.04 ± 0.82	87.22 ± 1.01	33.27 ± 1.09	52.16 ± 0.17
RelationNet [22]	75.62 ± 1.00	87.84 ± 0.27	37.13 ± 0.20	51.76 ± 1.48
MAML [25]	74.81 ± 0.25	83.54 ± 1.79	34.01 ± 1.25	48.83 ± 0.62
HyperMAML [31]	79.07 ± 1.09	$\mathit{89.22 \pm 0.78}$	36.32 ± 0.61	49.43 ± 0.14
HyperShot [47]	78.06 ± 0.24	89.04 ± 0.18	39.09 ± 0.28	$\mathit{57.77 \pm 0.33}$
HyperShot + adaptation [47]	$\mathbf{80.65 \pm 0.30}$	$\mathbf{90.81 \pm 0.16}$	$\mathbf{40.03 \pm 0.41}$	$\mathbf{58.86 \pm 0.38}$

We consider the inference tasks on cross-domain tasks (Omniglot→EMNIST and mini-ImageNet→CUB) datasets in the one-shot and five-shot setting. The highest results are bold and second-highest in italic (the larger, the better)

mini-ImageNet → CUB (model trained on mini-ImageNet and evaluated on CUB) and Omniglot → EMNIST in the one-shot and five-shot settings. Similarly to the previous experiment, we treated Bayesian and non-Bayesian approaches separately.

HyperShot We start with the non-Bayesian approaches and report the results in Table 3. In every setting, HyperShot achieves the highest accuracy and, as such, is much better than any other non-Bayesian approach. We note that just like in the case of regular classification, adapting the hypernetwork on the individual tasks consistently improves its performance.

BayesHyperShot The results among the Bayesian methods are reported in Table 4. We observer that in every setting BayesHyperShot is comparable but slightly worse than the best models. The difference is usually between 1 and 3 percent points, making the BayesHyperShot among three or four best Bayesian methods. It is worth noting that in this setting, performing the adaptation procedure on BayesHyperShot could result in worse performance than not adapting at all.

4.3 Uncertainty quantification

The most important feature of the Bayesian realization of our general framework is that we have the full insight into the model's uncertainty. In fact, due to the BayesHyperShot's probabilistic construction, we can quantify the level of model's certainty and answer the question if the model is sure about the specific prediction. Moreover, this property, enable us to say if the given sample comes from the known distribution—the distribution related to the current few-shot task or if it is out-of-distribution example.

In the following experiment, we present the uncertainty quantification of the BayesHyperShot model. We consider a

Table 4 Classification accuracy results for the Bayesian approaches

Method	Omniglot→EMNIST One-shot	Five-shot	Mini-ImageNet→CUB One-shot	Five-shot
DKT [1]	75.40 ± 1.10	$\mathbf{90.30 \pm 0.49}$	$\mathbf{40.14 \pm 0.18}$	*56.40 ± 1.34*
OVE PG GP + Cosine (ML) [48]	68.43 ± 0.67	86.22 ± 0.20	*39.66 ± 0.18*	55.71 ± 0.31
OVE PG GP + Cosine (PL) [48]	77.00 ± 0.50	87.52 ± 0.19	37.49 ± 0.11	$\mathbf{57.23 \pm 0.31}$
Bayesian MAML [34]	63.94 ± 0.47	65.26 ± 0.30	33.52 ± 0.36	51.35 ± 0.16
BayesHMAML [36]	*80.95 ± 0.46*	89.21 ± 0.27	36.90 ± 0.34	49.24 ± 0.38
BayesHMAML + adaptation [36]	$\mathbf{81.05 \pm 0.47}$	*89.76 ± 0.26*	37.23 ± 0.44	50.79 ± 0.59
BayesHyperShot	79.71 ± 0.23	89.54 ± 0.25	37.85 ± 0.22	51.37 ± 0.69
BayesHyperShot + adaptation	79.65 ± 0.33	89.70 ± 0.12	37.02 ± 0.36	51.62 ± 0.72

We consider the inference tasks on cross-domain tasks (Omniglot→EMNIST and mini-ImageNet→CUB) datasets in the one-shot and five-shot setting. The highest results are bold and second-highest in italic (the larger, the better)

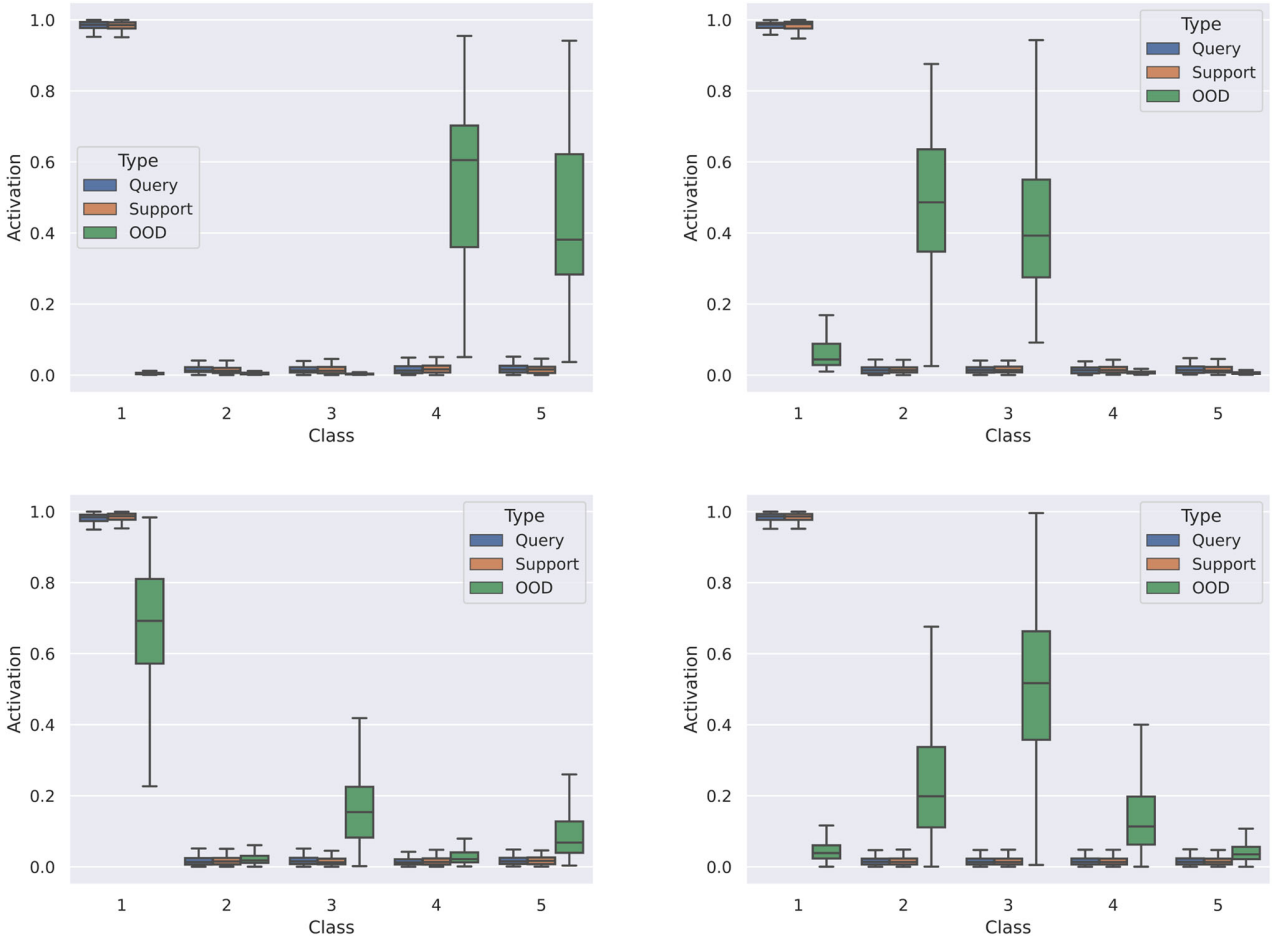


Fig. 3 We visualize box plots of distributions of activations produced by the sampled models for the four different sets of support/query/out-of-distribution images. As we can see, BayesHyperShot always yields similar predictions for elements from both the support and query sets. On the other hand, we observe a high variance of activations when processing out-of-distribution images, which indicates the high uncertainty of the model in such cases

model trained on the specific dataset and setting (here, it was Omniglot → EMNIST). Then, we have taken three different types of samples to quantify the level of uncertainty of our model's predictions. Specifically, we test the images from:

Table 5 Classification *accuracy* results for HyperShot in the five-shot setting with two variants of the support embeddings aggregation

	Omni→EMNIST	CUB	Mini-ImageNet
HyperShot (fine-grained)	87.55 ± 0.19	78.05 ± 0.20	67.07 ± 0.47
HyperShot (averaged)	89.04 ± 0.18	79.80 ± 0.16	69.62 ± 0.28

The performance measured on Omniglot→EMNIST, CUB, and mini-ImageNet→CUB tasks. The larger, the better

- Support set;
- Query set;
- Coming from the same dataset, but with class not present in the support set (out of distribution).

We observe that the examples coming from the support set or query set are classified with very high certainty. The model classifies such images as from a given class (label 1) or not (label 0). Note that label 1 appears for only one class, and label 0 for the rest.

However, the most important is the result obtained for the samples of classes not present in the support set. We observe that the model is uncertain about the classification—it usually gives nonzero probabilities for each of the possible classes. The present increase in the entropy of probabilities allows us to classify such examples as out-of-distribution samples. The results are presented in Fig. 3.

4.4 Ablation study

In order to investigate different architectural choices in adapting our framework to the specific task, we provide a comprehensive ablation study. In this ablation study, we consider the HyperShot model only for better clarity and apply our findings when selecting the parameters of BayesHyperShot. We focused mostly on the four major components of the HyperShot design, i.e., the method of processing multiple support examples per class, the number of neck layers, the number of head layers and the size of the hidden layers, presented in Tables 5, 6 and 7. In the case of the experiments focusing on aggregating the number of support examples in the five-way five-shot setting, we perform the benchmarks on CUB and mini-ImageNet, using a four-layer convolutional backbone. In the remaining experiments, we tested HyperShot on the CUB dataset in the five-way one-shot setting with ResNet-10 backbone.

Aggregating support examples in the five-shot setting

In HyperShot, the hypernetwork generates the weights of the information about the support examples, expressed through the support–support kernel matrix. In the case of five-way one-shot classification, each task consists of 5 support examples, and therefore, the size of the kernel matrix is (5×5), and the input size of the hypernetwork is 25. However, with a growing number of the support examples, increasing the size of the kernel matrix would be impractical and could lead to overparametrization of the hypernetwork.

Since hypernetworks are known to be sensitive to large input sizes [4], we consider a way to maintain a constant input size of HyperShot, independent of the number of support examples of each class by using means of support embeddings of each class for kernel calculation, instead of individual embeddings. Prior works suggest that when there are multiple examples of a class, the averaged embedding of such class represents it sufficiently in the embedding space [19].

To verify this approach, in the five-shot setting, we train HyperShot with two variants of calculating the inputs to the kernel matrix:

- Fine-grained—utilizing a hypernetwork that takes as an input a kernel matrix between each of the embeddings of the individual support examples. This kernel matrix has a shape of (25×25).
- Averaged—utilizing a hypernetwork where the kernel matrix is calculated between the means of embeddings of each class. The kernel matrix in this approach has a shape of (5×5).

We benchmark both variants of HyperShot on the five-shot classification task on CUB and mini-ImageNet datasets, as well as the task of cross-domain Omniglot → EMNIST classification. We report the accuracies in Table 5. It is evident that averaging the embeddings before calculating the kernel matrix yields superior results.

Hidden size: Firstly, as presented in Table 6, we compare different sizes of hidden layers. The results agree with the intuition that the wider the layers, the better the results. However, we also observe that some hidden sizes (e.g., 8188) could be too large to learn effectively. Because of that, we propose to use hidden sizes of 2048 or 4096 as the standard.

Neck and head layers: Then, we compared the influence of the number of neck layers and head layers of HyperShot for the achieved results, as presented in Table 7. We observed that the most critical is the number of head layers—specific for each target network's layers. Because of that, we propose using the standard number of 3 head layers and using various neck layers—tuning them to the specific task.

Table 6 Comparison between various hidden sizes in the HyperShot's layers

Hidden size	Accuracy
256	70.16 ± 0.45
512	71.70 ± 0.46
1024	70.89 ± 0.62
2048	$\mathbf{72.43 \pm 0.59}$
4096	71.99 ± 0.70
8188	72.05 ± 0.33

The classification *accuracy* results on CUB task and five-way one-shot setting. The larger, the better

Table 7 Comparison between various HyperShot's architectures (different number of neck layers and head layers)

Neck layers	Head layers	Accuracy
1	**3**	$\mathbf{73.00 \pm 0.55}$
2	1	71.53 ± 0.33
2	2	68.06 ± 0.59
2	3	71.99 ± 0.70
3	3	70.81 ± 0.39

The classification *accuracy* results on CUB task and five-way one-shot setting. The larger, the better

5 Conclusion

In this work, we introduced a novel general framework that uses kernel methods combined with hypernetworks. Our method directly relies on the kernel-based representations of the support examples and a hypernetwork paradigm to create the query set's classification module. We concentrate on relations between embeddings of the support examples instead of direct feature values. Thanks to this approach, models that realize our framework can adapt to highly different tasks.

Specifically, in this paper, we propose a classical (HyperShot) and a Bayesian (BayesHyperShot) realization of our framework. We evaluate both models on various one-shot and few-shot image classification tasks. Both HyperShot and BayesHyperShot demonstrate high accuracy in all tasks, performing comparably or better to state-of-the-art solutions. Moreover, both models have a strong ability to generalize, as evidenced by their performances on cross-domain classification tasks.

Finally, we demonstrate the ability of the Bayesian version of our framework to properly quantify the uncertainty of the model's prediction. As such, BayesHyperShot is able to recognize an out-of-distribution samples and return the level of certainty of the classification.

Acknowledgements This research was funded in part by National Science Centre, Poland, 2022/45/N/ST6/03374. The work of M. Sendera was supported by the National Centre of Science (Poland) Grant No. 2022/45/N/ST6/03374. The work of J. Tabor was supported by the National Centre of Science (Poland) Grant No. 2019/33/B/ST6/00894. The work of P. Spurek and M. Przewięźlikowski was supported by the National Centre of Science (Poland) Grant No. 2021/43/B/ST6/01456. The work of M. Zięba was supported by the National Centre of Science (Poland) Grant No. 2020/37/B/ST6/03463.

Appendix A. Comparison to the original HyperShot publication

This work is an extension of "HyperShot: Few-Shot Learning by Kernel Hypernetworks," originally published at the WACV 2023 Conference [47]. In this section, we address the points raised by the reviewers of the original publication, and outline which parts of this work are extensions of the original publication.

A.1 Addressing the WACV 2023 reviews

Hypernetwork architecture The reviewers raised a concern about a lack of a detailed description of our Hypernetwork architecture. While the general architecture had been visualized as a part of Fig. 4, we also expanded Sect. 2.2 to include a more detailed description of the architecture.

Comparisons to related work The reviewers suggested several additional works to which we could compare HyperShot. While we added several of them to the tables reported in our experiments [49, 50], we chose to omit others [42, 43, 51] as the settings of the few-shot problems they solve differ from the standard inductive few-shot classification.

Application to other tasks besides image classification An important point raised by the reviewers was the lack of experiments on different problems besides few-shot image classification. While we acknowledge that there are various different computer vision problems for which data-efficient solutions would be desirable, we felt that such experiments are out of the scope of our work, as HyperShot and BayesHyperShot were tailor-made for the classification problem, which is by far the most popular few-shot benchmark. An application of kernel Hypernetworks to other few-shot problems could be an interesting point of future work.

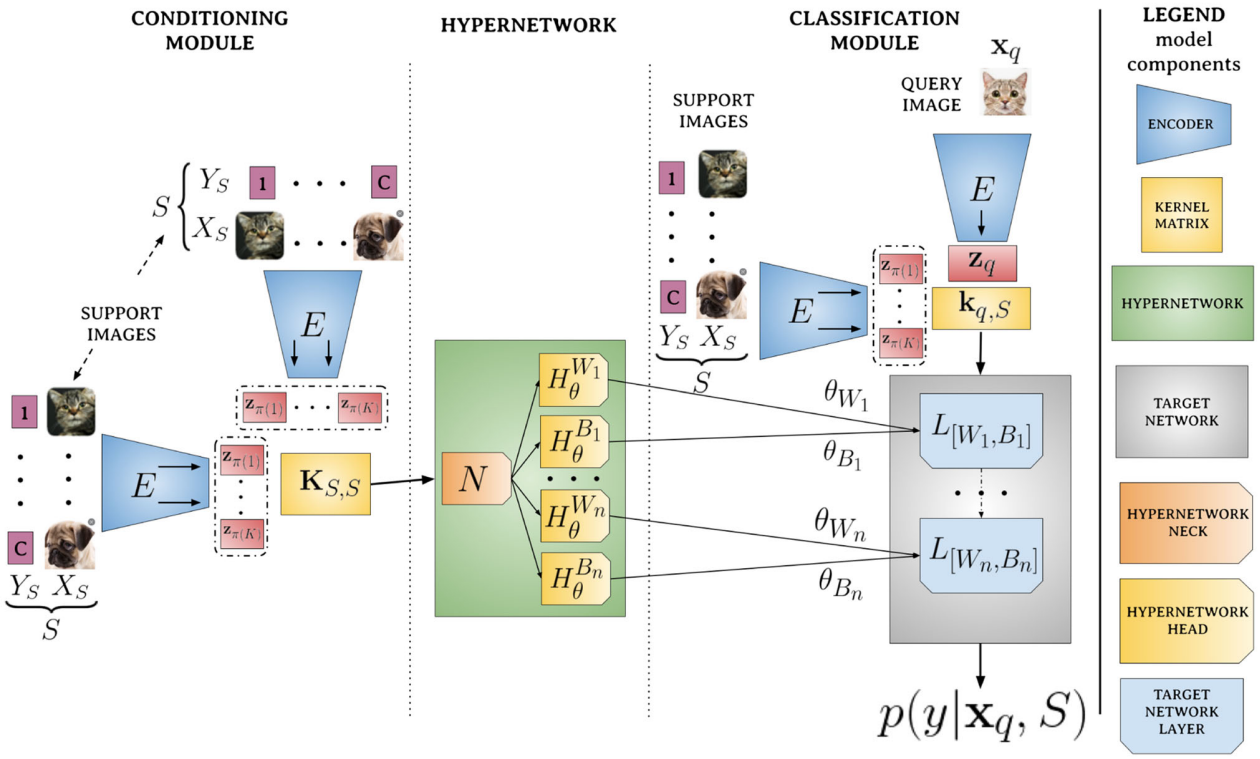


Fig. 4 A detailed outline of the architecture of HyperShot, with the denoted flow of parameters generated by the hypernetwork heads

A.2 Extensions compared to the WACV 2023 paper

The most important difference of this work compared to the WACV 2023 publications is the generalization of the previously introduced HyperShot method to a Bayesian framework, resulting in a model which we call BayesHyperShot. We conduct a series of experiments on the HyperShot model and show that apart from learning from small amounts of data, it is capable of modeling uncertainty of predictions.

Appendix B. Training details

In this section, we present in detail the architecture and hyperparameters of HyperShot.

Architecture overview

From a high-level perspective, the architecture of HyperShot consists of three parts:

- Backbone—a convolutional feature extractor.
- Neck—a sequence of zero or more fully connected layers with ReLU nonlinearities in between.
- Heads—for each parameter of the target network, a sequence of one or more linear layers, which predicts the values of that parameter. All heads of HyperShot have identical lengths, hidden sizes and input sizes that depend on the generated parameter's size.

The target network generated by both HyperShot and BayesHyperShot reuses its backbone. We outline this architecture in Fig. 4.

Backbone For each experiment described in the main body of this work, we follow [1] in using a shallow backbone (feature extractor) for HyperShot as well as referential models. This backbone consists of four convolutional layers, each consisting of a convolution, batch normalization and ReLU nonlinearity. Apart from the first convolution, which has the input size equal to the number of image channels, each convolution has an input and output size of 64. We apply max-pooling between each convolution, which decreases by half the resolution of the processed feature maps. The output of the backbone is flattened so that the further layers can process it.

Datasets For the purpose of making a fair comparison, we follow the procedure presented in, e.g., [1, 10]. In the case of the CUB dataset [44], we split the whole amount of 200 classes (11788 images) across train, validation and test consisting of 100, 50 and 50 classes, respectively [10]. The mini-ImageNet dataset [45] is created as the subset of ImageNet [52], which consists of 100 different classes represented by 600 images

Table 8 Hyperparameters

Hyperparameter	CUB	Mini-ImageNet	Mini-ImageNet → CUB	Omniglot → EMNIST
Kernel function	Cosine similarity	Cosine similarity	Cosine similarity	Cosine similarity
Learning rate	0.001	0.001	0.001	0.001
Hypernetwork's head layers no	3	3	3	2
Hypernetwork's neck layers no	2	2	2	1
Hypernetwork layers' hidden dim	4096	4096	4096	512
Support embeddings aggregation	Averaged	Averaged	Averaged	Averaged
Task set size	1	1	1	1
Target network layers no	1	1	1	2
Target network activation	ReLU	ReLU	ReLU	ReLU
Adaptation epochs (if used)	10	10	10	10
Adaptation learning rate	0.0001	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam	Adam
Epochs no	10000	10000	10000	2000

for each one. We followed the standard procedure and divided the mini-ImageNet into 64 classes for the train, 16 for the validation set and the remaining 20 classes for the test. The well-known Omniglot dataset [53] is a collection of characters from 50 different languages. The Omniglot contains 1623 white and black characters in total. We utilize the standard procedure to include the examples rotated by 90° and increase the size of the dataset to 6492, from which 4114 were further used in training. Finally, the EMNIST dataset [54] collects the characters and digits coming from the English alphabet, which we split into 31 classes for the test and 31 for validation.

Data augmentation We apply data augmentation during model training in all experiments, except Omniglot → EMNIST cross-domain classification. The augmentation pipeline is identical to the one used by [1] and consists of the random crop, horizontal flip and color jitter steps.

Appendix C. Hyperparameters

Below, we outline the hyperparameters of architecture and training procedures used in each experiment.

We use cosine similarity as a kernel function and averaged support embeddings aggregation in all experiments. HyperShot is trained with the learning rate of 0.001 with the Adam optimizer [55] and no learning rate scheduler. Task-specific adaptation is also performed with the Adam optimizer and the learning rate of 0.0001.

For the natural image tasks (CUB, mini-ImageNet, mini-ImageNet → CUB classification), we use a hypernetwork with the neck length of 2, head lengths of 3 and a hidden size of 4096, which produce a target network with a single fully connected layer. We perform training for 10000 epochs.

For the simpler Omniglot → EMNIST character classification task, we train a smaller hypernetwork with the neck length of 1, head lengths of 2 and the hidden size of 512, which produces a target network with two fully connected layers and a hidden size of 128. We train this hypernetwork for a shorter number of epochs, namely 2000.

We summarize all the above hyperparameters in Table 8.

Appendix D. Source code

The source code required for running the experiments is available at https://github.com/gmum/few-shot-hypernets-public.

References

1. Patacchiola, M., Turner, J., Crowley, E.J., O'Boyle, M., Storkey, A.J.: Bayesian meta-learning for the few-shot setting via deep kernels. Adv. Neural Inf. Process. Syst. **33**, 16108–16118 (2020)
2. Sendera, M., Tabor, J., Nowak, A., Bedychaj, A., Patacchiola, M., Trzcinski, T., Spurek, P., Zieba, M.: Non-gaussian gaussian processes for few-shot regression. Adv. Neural Inf. Process. Syst. **34**, 10285–10298 (2021)
3. Wang, Z., Miao, Z., Zhen, X., Qiu, Q.: Learning to learn dense gaussian processes for few-shot learning. Adv. Neural Inf. Process. Syst. **34**, 13230–13241 (2021)
4. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. In: International Conference on Learning Representations (2017). https://openreview.net/forum?id=rkpACe1lx
5. Qiao, S., Liu, C., Shen, W., Yuille, A.L.: Few-shot image recognition by predicting parameters from activations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7229–7238 (2018)
6. Ye, H.-J., Hu, H., Zhan, D.-C., Sha, F.: Few-shot learning via embedding adaptation with set-to-set functions. In: IEEE/CVF

Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8808–8817 (2020)
7. Zhmoginov, A., Sandler, M., Vladymyrov, M.: Hypertransformer: Model generation for supervised and semi-supervised few-shot learning. In: International Conference on Machine Learning, pp. 27075–27098. PMLR (2022)
8. Zhu, Z., Wang, L., Guo, S., Wu, G.: A Closer Look at Few-Shot Video Classification: A New Baseline and Benchmark. arXiv (2021). https://doi.org/10.48550/ARXIV.2110.12358. arXiv:2110.12358
9. Perrett, T., Masullo, A., Burghardt, T., Mirmehdi, M., Damen, D.: Temporal-relational crosstransformers for few-shot action recognition. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 475–484 (2021). https://doi.org/10.1109/CVPR46437.2021.00054
10. Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C.F., Huang, J.-B.: A closer look at few-shot classification. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=HkxLXnAcFQ
11. Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M.: Generalizing from a few examples: A survey on few-shot learning. ACM Comput. Surv. (csur) **53**(3), 1–34 (2020)
12. Sheikh, A.-S., Rasul, K., Merentitis, A., Bergmann, U.: Stochastic maximum likelihood optimization via hypernetworks. arXiv preprint arXiv:1712.01141 (2017)
13. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S.: Generating sentences from a continuous space. In: 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, pp. 10–21. Association for Computational Linguistics (ACL) (2016)
14. Bengio, S., Bengio, Y., Cloutier, J., Gescei, J.: On the optimization of a synaptic learning rule. In: Optimality in Biological and Artificial Networks, pp. 281–303 (2013)
15. Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A.: Meta-learning in neural networks: A survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(9), 5149–5169 (2021)
16. Schmidhuber, J.: Learning to control fast-weight memories: an alternative to dynamic recurrent networks. Neural Comput. **4**(1), 131–139 (1992). https://doi.org/10.1162/neco.1992.4.1.131
17. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proc. IEEE (2020). https://doi.org/10.1109/JPROC.2020.3004555
18. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. Adv. Neural. Inf. Process. Syst. **29**, 3630–3638 (2016)
19. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Adv. Neural Inf. Process. Syst. **30** (2017)
20. Oreshkin, B., Rodríguez López, P., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31 (2018). https://proceedings.neurips.cc/paper_files/paper/2018/file/66808e327dc79d135ba18e051673d906-Paper.pdf
21. Hu, Y., Gripon, V., Pateux, S.: Leveraging the feature distribution in transfer-based few-shot learning. In: Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30, pp. 487–499. Springer (2021)
22. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208 (2018)
23. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10657–10665 (2019)
24. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
25. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. PMLR (2017)
26. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
27. Antoniou, A., Edwards, H., Storkey, A.: How to train your MAML. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=HJGven05Y7
28. Fan, C., Ram, P., Liu, S.: Sign-maml: Efficient model-agnostic meta-learning by signsgd. In: 5th Workshop on Meta-Learning at NeurIPS 2021 (2021)
29. Ye, H.-J., Chao, W.-L.: How to train your MAML to excel in few-shot classification. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=49h_IkpJtaE
30. Rajeswaran, A., Finn, C., Kakade, S.M., Levine, S.: Meta-learning with implicit gradients. Adv. Neural. Inf. Process. Syst. **32**, 113–124 (2019)
31. Przewięźlikowski, M., Przybysz, P., Tabor, J., Zięba, M., Spurek, P.: Hypermaml: Few-shot adaptation of deep models with hypernetworks. arXiv preprint arXiv:2205.15745 (2022)
32. Bauer, M., Rojas-Carulla, M., Świątkowski, J.B., Schölkopf, B., Turner, R.E.: Discriminative k-shot learning using probabilistic models. arXiv preprint arXiv:1706.00326 (2017)
33. Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D., Eslami, S.M.A.: Conditional neural processes. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1704–1713 (2018)
34. Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., Ahn, S.: Bayesian model-agnostic meta-learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 7343–7353 (2018)
35. Grant, E., Finn, C., Levine, S., Darrell, T., Griffiths, T.: Recasting gradient-based meta-learning as hierarchical bayes. In: International Conference on Learning Representations (2018)
36. Borycki, P., Kubacki, P., Przewięźlikowski, M., Kuśmierczyk, T., Tabor, J., Spurek, P.: Hypernetwork approach to Bayesian maml. arXiv preprint arXiv:2210.02796 (2022)
37. Ravi, S., Beatson, A.: Amortized Bayesian meta-learning. In: International Conference on Learning Representations (2018)
38. Nguyen, C., Do, T.-T., Carneiro, G.: Uncertainty in model-agnostic meta-learning using variational inference. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 3090–3100 (2020)
39. Jerfel, G., Grant, E., Griffiths, T.L., Heller, K.: Reconciling meta-learning and continual learning with online mixtures of tasks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 9122–9133 (2019)
40. Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., Turner, R.: Meta-learning probabilistic inference for prediction. In: International Conference on Learning Representations (2018)
41. Rasmussen, C.E.: Gaussian processes in machine learning. In: Summer School on Machine Learning, pp. 63–71. Springer (2003)
42. Shen, X., Xiao, Y., Hu, S.X., Sbai, O., Aubry, M.: Re-ranking for image retrieval and transductive few-shot classification. In: Advances in Neural Information Processing Systems, vol. 34, pp. 25932–25943 (2021)
43. Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., Cord, M.: Boosting few-shot visual learning with self-supervision. In: Proceedings of the IEEE International Conference on Computer Vision (2019)

44. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology (2011)
45. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR (2017)
46. Rajasegaran, J., Khan, S.H., Hayat, M., Khan, F.S., Shah, M.: Meta-learning the learning trends shared across tasks. CoRR **abs/2010.09291** (2020)
47. Sendera, M., Przewięźlikowski, M., Karanowski, K., Zięba, M., Tabor, J., Spurek, P.: Hypershot: Few-shot learning by kernel hypernetworks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2469–2478 (2023)
48. Snell, J., Zemel, R.: Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. In: International Conference on Learning Representations (2020)
49. Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J.B., Isola, P.: Rethinking few-shot image classification: a good embedding is all you need? In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16, pp. 266–282. Springer (2020)
50. Rizve, M.N., Khan, S., Khan, F.S., Shah, M.: Exploring complementary strengths of invariant and equivariant representations for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10836–10846 (2021)
51. Jian, Y., Torresani, L.: Label hallucination for few-shot classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 7005–7014 (2022)
52. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
53. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 33 (2011)
54. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: Extending mnist to handwritten letters. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2921–2926 (2017). https://doi.org/10.1109/IJCNN.2017.7966217
55. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2014)

Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations

Marcin Sendera[*1,2] **Amin Sorkhei** **Tomasz Kuśmierczyk**[*1]

[1]Jagiellonian University
[2]Mila, Université de Montréal

Abstract

Gaussian Processes (GPs) provide a convenient framework for specifying function-space priors, making them a natural choice for modeling uncertainty. In contrast, Bayesian Neural Networks (BNNs) offer greater scalability and extendability but lack the advantageous properties of GPs. This motivates the development of BNNs capable of replicating GP-like behavior. However, existing solutions are either limited to specific GP kernels or rely on heuristics.
We demonstrate that trainable activations are crucial for effective mapping of GP priors to wide BNNs. Specifically, we leverage the closed-form 2-Wasserstein distance for efficient gradient-based optimization of reparameterized priors and activations. Beyond learned activations, we also introduce trainable periodic activations that ensure global stationarity by design, and functional priors conditioned on GP hyperparameters to allow efficient model selection.
Empirically, our method consistently outperforms existing approaches or matches performance of the heuristic methods, while offering stronger theoretical foundations.

1 INTRODUCTION

Function-space priors for BNNs offer a better way of specifying beliefs on data modeled by the model. Unlike priors on model parameters (i.e. weights), which lack interpretability and are hard to specify, they ultimately lead to more intuitive and meaningful representation of prior knowledge [Sun et al., 2019]. Function-space priors can be conveniently specified in terms of GPs. Instead of specifying complex distributions over model parameters, GPs allow for defining priors over entire functions, through the choice of kernel capturing requirements such as smoothness or periodicity.

Finding a GP that matches a wide BNN is straightforward. A classic result by Neal [1996], Williams [1996], later extended to deep neural networks by Lee et al. [2017], Matthews et al. [2018], shows that wide layers in neural networks behave *a priori* like GPs. This is achieved by identifying GP kernels matching the covariances of (B)NNs.

In contrast, finding BNNs that exhibit the desired behavior of GPs is a notoriously difficult problem, with solutions or approximations existing for only a few GP kernels. For example, Meronen et al. [2020] recently found a solution for the popular Matérn kernel using the Wiener-Khinchin theorem. However, the proposed BNN activation assumes binary white noise priors on the model weights, which severely hinders posterior learning. Consequently, an approximate solution with Gaussian priors must be used, ultimately leading to suboptimal performance, as we demonstrate in Sec. 4.3.

The challenge of imposing function-space *a priori* behavior on BNNs previously has been addressed using gradient-based optimization. In particular, Flam-Shepherd et al. [2017], Flam-Shepherd et al. [2018], and especially Tran et al. [2022] attempted to solve this by learning priors on parameters. However, to achieve sufficient fidelity with simple priors (Gaussian or hierarchical), their approach requires deep networks, which presents a challenge for posterior learning due to the lack of effective algorithms for learning posteriors in deep and wide BNNs. On the other hand, by using normalizing flows to model weight priors they often can match target GP prior (see Sec. 4.2). Nevertheless, imposing different priors for each weight invalidates the theoretical assumptions regarding BNN convergence, rendering this approach merely a heuristic.

Fitting only weight (and biases) priors is insufficient for matching BNN and GP priors. Hence, we propose learning parametric activations as well. We draw inspiration from previous work demonstrating that function-space priors can be adjusted by altering activations [Neal, 1996] and also from

[*]Equal contribution to implementation.

research utilizing activations for improved uncertainty estimation [Morales-Alvarez et al., 2020, Postels et al., 2020]. In fact, the same GP-like behavior can often be realized by different combinations of parameter priors and activations, as seen in the BNN covariance formulation (Eq. 1) and learning both jointly offers a greater flexibility. In particular, our method (see Fig. 1) transfers priors from GPs to BNNs by matching their function-space distributions using the closed-form 2-Wasserstein loss and by learning activations in addition to priors on weights. It achieves faithful function-space priors using just shallow BNNs. In contrast to Tran et al. [2022], it is backed by theoretical results ensuring that the learned BNNs asymptotically converge to GPs, and compared to Meronen et al. [2020], it can work with arbitrary kernels. Finally, it relies on simple weight priors, enabling efficient posterior inference.

Within the proposed framework, we propose two novel extensions. First, by conditioning the BNN's functional priors (weights and activation) on GP hyperparameters – implemented via hypernetworks – we facilitate efficient model selection, e.g., allow prior hyperparameters optimization. Second, building on the result of Meronen et al. [2021] that periodic activations induce stationarity, we propose trainable periodic activations and show how such activations can be learned in practice, ensuring global stationarity in BNNs.

Our contributions extend the existing literature on BNN–GP equivalence in several ways: (1) we introduce a practical approach for imposing GP-like function-space priors on BNNs; (2) we employ hypernetworks to condition these priors on GP hyperparameters; (3) we introduce trainable periodic activations; and (4) we empirically demonstrate that even shallow BNNs can achieve the desired properties, offering an alternative to baselines using deep models. Note that although parametric activations and hypernetworks have been used previously, we are the first to employ and combine them for this particular task. Moreover, although our design of trainable periodic activations builds on the previous result that periodic activations induce stationarity, it is novel. Finally, despite our focus on the specification of priors, we also validate our pre-trained priors by learning posteriors, employing both Hamiltonian Monte Carlo (HMC) and scalable Stochastic Variational Inference (SVI) to demonstrate empirical improvements in predictive performance and uncertainty quantification.

In the Appendix, the results presented in the paper are supplemented with a description of related work, a discussion of computational advantages of our approach, a detailed overview of the experimental settings, and additional plots and numerical data.

Our code we made publicly available [1].

[1] https://github.com/gmum/bnn-functional-priors

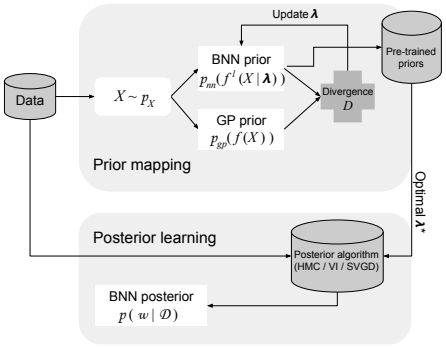


Figure 1: Schematic view of our approach.

2 PRELIMINARIES

Behavior akin to GPs has been identified in wide neural networks with a single hidden layer by Neal [1996], Williams [1996] and later generalized for deep architectures by Lee et al. [2017], Matthews et al. [2018] who considered stacking multiple such the wide layers.
Let's consider a network defined as:

$$f_i^0(x) = \sum_j^I w_{ij}^0 x_j + b_i^0, \quad i = 1, \ldots, H_0;$$

$$f^l(x) = \sum_j^{H_0} w_{ij}^l \phi(f_j^0(x)) + b^l$$

where x denotes an I-dimensional input, and $f^l(x)$ represents output at the l-th layer. In a BNN, $z_{ij}(x) = w_{ij}^l \phi(f_j^0(x))$ is a random variable. Assuming that the weights w_{ij} share a common prior and also the biases b_i have independent but common prior, the output $f^l(x)$ becomes a sum of independent and identically distributed (i.i.d.) random variables. By the Central Limit Theorem, this results in $f^l(x)$ converging to a Gaussian distribution, implying that the BNN converges to a GP in the limit of infinite width.

The covariance between two inputs, x and x', for the BNN at the (output) layer l is given by:

$$\mathrm{Cov}(f^l(x), f^l(x')) = {\sigma_b^l}^2 + {\sigma_w^l}^2 \mathbb{E}_{f_j^0}[\phi(f_j^0(x))\phi(f_j^0(x'))], \tag{1}$$

where σ_b^l and σ_w^l are respectively the variances of the biases and weights at layer l and the expectation is taken over distributions for w^0 and b^0. Hence, the BNN functional prior corresponds to a GP with kernel $\kappa_f^l(x, x') = \mathrm{Cov}(f^l(x), f^l(x'))$.

In this work, we assume zero-centered GPs. Therefore, for the distributions on weights and biases, we take $\mathbb{E}[w] = 0$ and $\mathbb{E}[b] = 0$. To ensure that the variance remains stable as the layer width increases we scale $\mathrm{Var}[w_{ij}^l] = \frac{{\sigma_w^l}^2}{H_0}$.

3 METHOD

3.1 TRANSFER OF PRIORS FROM GP TO BNN

Finding a GP equivalent to a pre-specified BNN can be easily done by a Monte Carlo estimate of Eq. 1. However, we focus on the significantly more challenging inverse problem of imposing behavior akin to a GP on a BNN. The task is to *identify appropriate priors on w^l, b^l, w^0, b^0, and the activation ϕ in Eq. 1 so that the covariance induced by the BNN aligns with the desired GP kernel κ*. Ideally, we aim to have $f^l \sim \text{GP}(0, \kappa)|_{\mathcal{X}}$, meaning that the distribution of the BNN outputs (pre-likelihood) would closely match that of the GP across the input space $\mathcal{X}$, i.e., $p_{nn}(f^l) = p_{gp}(f^l)$.

In practice, however, we settle for approximate matching over finite index sets [Shi et al., 2018, Sun et al., 2019], where the densities over BNN and GP outputs should approximately align as $p_{nn}(f^l(X)) \approx p_{gp}(f^l(X))$. $X \sim p_X$ ($X \in \mathcal{X}$) can be understood as sampling sets of inputs $\{x\}$ at which we observe the functions.

The matching between BNN and GP, we formulate as an optimization task:

$$p^*_{nn} = \text{argmin}_{p_{nn}} \frac{1}{S} \sum_{X \sim p_X} D(p_{nn}(f^l(X)), p_{gp}(f^l(X))),$$

where D is an *arbitrary*, differentiable divergence measure, and S is a number of samples. In practice, we perform gradient-based optimization with $S = 1$ sample used for each step.

3.2 DIFFERENTIABLE PRIORS AND ACTIVATIONS

The results presented in Sec. 2 hold for distributions with finite variances and light tails. We use the basic zero-centered factorized Gaussians with learned variances, e.g., $p(w|\sigma^2_w)$ and $p(b|\sigma^2_b)$, as prior distributions on model weights and biases. To enable gradient-based optimization, such as gradient propagation through weight and bias samples, we reparameterize the distributions using the reparameterization trick [Kingma and Welling, 2014]. Additionally, *we propose to employ parametric and differentiable activations $\phi(\cdot|\eta)$ to enhance the network's flexibility.* This previously overlooked idea allows us to model complex functional priors within a single hidden layer of a BNN, even with simple prior distributions on weights and biases.

Given the reparameterized distributions on weights and biases, as well as the parametric activation, p_{nn} prior is fully characterized by the parameters $\lambda = \{\sigma^0_b, \sigma^0_w, \sigma^l_b, \sigma^l_w, \eta\}$. The optimization objective can be then expressed as:

$$\lambda^* = \text{argmin}_\lambda \frac{1}{S} \sum_{X \sim p_X} D\left(p_{nn}\left(f^l(X|\lambda)\right), p_{gp}\left(f^l(X)\right)\right), \tag{2}$$

where S denotes the number of input samples. Eq. 2 is solved through gradient-based optimization w.r.t. λ.

Eq. 2 requires deciding on the divergence measure D and a model for the activation function ϕ. The problem of learning activations for neural networks involves designing functions that enable networks to capture complex and non-linear relationships effectively. In principle, any function $\phi : \mathcal{R} \to \mathcal{R}$ can serve as an activation, but the choice of ϕ significantly impacts both the network's expressiveness and its training dynamics.

We explore several models for ϕ, including Rational (Pade) activations [Molina et al., 2019], Piecewise Linear (PWL) activations[2], and activations implemented as a neural network with a single narrow (with only 5 neurons) hidden layer, using ReLU/SiLU own activations. These functions are computationally efficient and introduce desirable non-linear properties, striking a balance between efficiency and the capacity to model intricate patterns.

Our choice of a NN-based activation with a single hidden layer consisting of 5-10 neurons and using ReLU/SiLU activations was informed by empirical studies (see Fig. 2 and Tables 3 and 4 in Section B) comparing various learnable functions. These experiments, including ablations on the number of layers and width of the NN activations, show that this configuration offers a good balance of flexibility, fast convergence, and quality of prior fit. While deeper or wider NN activations can marginally improve fidelity in some cases, they also increase convergence times and can make optimization harder.

3.3 PERIODIC ACTIVATIONS FOR STATIONARY GPS

The optimization in Eq. 2 acts on range of inputs X, where the BNN's behavior increasingly resembles the GP as training progresses. However, there are no guarantees about the BNN's behavior outside this subset, particularly far from it. This limitation is exacerbated by *local stationarity* in BNNs: although GPs with stationary kernels are *globally* stationary, BNNs usually only exhibit stationarity within a limited input range. Outside this range, the covariance induced by the BNN may diverge from the desired GP behavior. This localized training can cause uncertainty quantification issues. In regions far from the data, a BNN may either become overly confident or overly uncertain, depending on the priors and the mismatch between the BNN's architecture and the GP's true behavior. It is challenging to enforce global properties, like stationarity or smoothness, across the entire input space, as the BNN's learned covariance structure is overly dependent on the inputs' range.

We address the local stationarity challenge by *introducing trainable activations designed to exhibit periodic behavior.*

[2] https://pypi.org/project/torchpwl/

This approach is motivated by the result of Meronen et al. [2021], who showed that periodic activation functions induce stationarity in BNNs. Based on this theoretical result, we propose a practical solution. Our proposed activations

$$\phi(x|\psi, A) = \sum_{i=1}^{K} A_i \cos(2\pi\psi_i x) + \sum_{j=1}^{K} A_j \sin(2\pi\psi_j x)$$

are inspired by Fourier analysis and can be trained to fit a desired functional prior. They rely on the variational parameters $\eta = \{\psi_i, A_i, \psi_j, A_j\}$ steering respectively frequencies and amplitudes. For experiments, we used $K = 5$ components.

3.4 CONDITIONAL PRIORS AND ACTIVATIONS

Researchers previously, when fitting BNN priors, were limited to upfront fixing hyperparameters (such as length-scale) of target GPs or alternatively, they would employ workarounds like hierarchical kernels with hyperparameters sampled from hyperpriors (see, e.g., Tran et al. [2022]). Instead, *we propose to condition the priors on weights, biases, and activations in BNNs on the hyperparameters* of a GP kernel. By incorporating GP hyperparameters directly into BNNs, we *bridge the equivalence gap between GPs and BNNs*, finally enabling BNNs to fully replicate the behavior of GPs. In particular, integrating hyperparameters directly within BNNs allows for their optimization, for example, using marginal likelihood (evidence). This facilitates effective model selection, which is a significant novelty compared to previous approaches.

Although our work focuses on conditioning on GP hyperparameters, the proposed conditioning framework is however more general and could be extended to priors dependent on other factors. For example, conditioning on input data could enhance model robustness against input range shifts by allowing the BNN to adapt its priors to the input range. This adaptation could alleviate issues with the locality of the matching BNN to a GP, as discussed in Section3.3.

The conditioning we implemented using hypernetworks [Ha et al., 2017, Chauhan et al., 2024], as $[\sigma, \eta] := \text{hnet}(\gamma|\theta)$, where σ and η are the sets of parameters for the priors on weights and activations, respectively. Here, γ represents the set of conditioning hyperparameters, which are transformed by the hypernetwork hnet. The hypernetwork has its own parameters θ, which are now optimized in Eq. 2 as $\lambda = \{\theta\}$ instead of $\{\sigma, \eta\}$. For a fixed architecture of hnet, σ and η are now fully determined by γ along with θ.

We are the first to test hypernetworks for generating activation parameters and using them for conditioning with hyperparameters. This poses a technical challenge in network design. For example, no architectures other than those based on RBFs showed promising results. Ultimately, we employed an MLP with three hidden layers, each using RBF activations. On top of that, we applied separate linear layers to map to the appropriate outputs: one for each of the prior variances σ and one for the parameters η of the trainable activation.

3.5 LOSS

The loss D measures the divergence between two distributions over functions. While p_{gp} can be sampled and evaluated (for fixed inputs GPs act like Gaussians), p_{nn} is implicitly defined by a BNN, meaning it can be sampled from, but not evaluated directly. Due to the implicit nature of p_{nn}, D must be specified for samples $\{f^l\}$ and approximated using Monte Carlo.

We discovered that the standard losses fail for our task. For example, estimating the empirical entropy term $(\int p_{nn}(f) \log(p_{nn}(f))\, df)$ for KL presents a numerical challenge. Consequently, we followed Tran et al. [2022] and relied on the Wasserstein distance instead (see Tran et al. [2022] for details):

$$\begin{aligned} D &= \left(\inf_{\varsigma \in \Gamma(p_{nn}, p_{gp})} \int_{\mathcal{F}\times\mathcal{F}} d(f, f')^p \varsigma(f, f') df df' \right)^{1/p} \\ &= \sup_{|\Psi|_L \leq 1} \mathbb{E}_{p_{nn}}[\Psi(f)] - \mathbb{E}_{p_{gp}}[\Psi(f)] \end{aligned} \tag{3}$$

However, unlike Tran et al. [2022], we propose to use the 2-Wasserstein metric, which for Multivariate Gaussians (applicable in the case of GPs and wide BNNs – at least approximately) has a *closed-form solution* [Mallasto and Feragen, 2017]:

$$D = ||\mu_1 - \mu_2||_2^2 + \text{Tr}\left(\Sigma_1 + \Sigma_2 - 2\sqrt{\sqrt{\Sigma_1}\Sigma_2\sqrt{\Sigma_1}}\right),$$

where $\mu_{1/2}$, $\Sigma_{1/2}$ are respectively expectations and covariance matrices estimated for p_{nn} and p_{gp} from samples $\{f^l(X)\}$ (we used 512 or 1024 reparameterized samples) evaluated for inputs X. Not only we avoid the internal optimisation due to $\sup_{|\psi|}$, but additionally D can be efficiently computed based on results by Buzuti and Thomaz [2023]. For experiments, we report numerical values of D normalized by number of elements in X.

3.6 OUTPUT STRUCTURE

The optimization objective in Eq. 2 is defined over functions f. The functions are passed through likelihoods to form model outputs y. For regression tasks, a Gaussian likelihood is typically used; for binary classification, Bernoulli; and for multiclass classification, a Categorical likelihood with multivariate f transformed via softmax. Note that the *prior transfer is independent of a likelihood*, meaning it does not rely on how the latent functions f relate to the outputs y. Then, as explained in Sec. 3.5, the optimization can be performed efficiently between Gaussians. If these assumptions

were not satisfied—e.g., if the desired priors are not expressible via a GP – we can revert to the nested optimization for $|\Psi|_L$ as implied by Eq. 3. Overall, we explain our method in terms of GPs and for brevity, we denote the target functional priors by p_{gp}. However, it is important to note that the method is *applicable to arbitrarily specified functional priors*.

For completeness, we follow to briefly discuss also the transfer of priors from GPs to BNNs for multivariate/multi-class-output settings; however, such extensions fall outside of the scope of this paper. In particular, the multivariate models [Rasmussen and Williams, 2005, Alvarez et al., 2012] can be approached using various architectural strategies, depending on the nature of the dependencies among the output dimensions. One approach is to construct independent BNNs for each output dimension, *stacking them side-by-side*, where each BNN is pretrained separately with priors obtained from independent single-output GPs. This approach is appropriate when output dimensions are assumed to be independent, leading to an ensemble of independently trained BNNs. Alternatively, *shared hidden layer* BNN can be used, where a common hidden representation is shared across all output dimensions, thereby capturing potential dependencies between outputs. The shared hidden layer structure reflects the idea of a multi-output or multi-task GP, where dependencies among outputs can be modeled explicitly using coregionalization techniques [Goovaerts, 1997, Bonilla et al., 2008]. The shared hidden layer BNN can thus be endowed with priors derived from these multi-output GPs, ensuring that both spatial and output correlations are incorporated into the BNN model.

3.7 OPTIMISATION CHALLENGES

The problem of finding BNNs behaving like GPs (e.g. inverting covariance equation) in *unidenfiable*. There exist multiple solutions assuring similar quality of the final match. Activations ϕ solving Eq.(2) are not unique. For example:

- The solutions are symmetric w.r.t activity values (y-axis), i.e., for $\phi'(f) = -\phi(f)$, values of covariance given by Eq.(1) are not changed, simply because $(-1)^2 = 1$.
- For $p(f_i^0(x))$ symmetric around 0 (for example, Gaussians), activations with flipped arguments $\phi'(f) = \phi(-f)$ result in the same covariances.
- Scaling activations $\phi'(f) = \alpha\phi(f)$ leads to the same covariances as scaling variances of the output weights as $(\sigma_w^l)' = \alpha \cdot \sigma_w^l$

Given a sufficiently flexible model (like a neural network itself), one can learn to approximate any target activation function to an arbitrary degree of accuracy on a compact domain. This is in line with *the universal approximation theorem*. However, in practice, there are multiple limitations and challenges. A model may require an impractically large number of parameters to approximate certain complex functions to a desired level of accuracy. More complex models may be harder to fit and require more training data. Some functions might require high numerical precision to be approximated effectively, and even if a model can fit a target activation function on a compact set, it might not generalize well outside the training domain. Overall, multiple factors may lead gradient-based optimization to fail for our task. However, such the *poor outcomes will be reflected in low values of the final loss* (Eq. 3). A practical remedy is to rerun optimization, once poorly performing outliers are noticed.

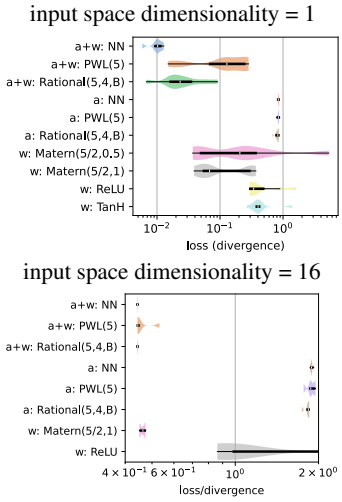


Figure 2: Quality of matching BNNs to the prior of a GP with a Matérn kernel ($\nu = 5/2, \ell = 1$) for 1D (top) and 16D inputs (bottom). We evaluate models with trained parameter priors (denoted by *w*), activations (denoted by *a*), and both (denoted by *a+w*). Each label specifies whether a fixed activation (e.g., ReLU) or a specific activation model (e.g., Rational) was employed. Gaussian parameter priors were used by default. If not trained, we set variances to 1., and for the hidden layer, we normalized the variance by its width. The label *w: Matérn* refers to a BNN with the closed-form (fixed) activation as derived by Meronen et al. [2020].

4 FINDINGS

4.1 DOES LEARNING ACTIVATIONS IMPROVE LEARNING OF FUNCTION-SPACE PRIORS?

To map GP function-space priors to a BNN, we can: 1) train its priors on parameters, 2) train both priors and activation, or 3) learn just the activation function. We empirically show that learning activations in addition to priors provide better solutions to the considered problem.

Fig. 2 illustrates the results of an extensive study in which we compare the quality of the GP prior fit for various mod-

Table 1: Results for UCI regression task (Boston dataset) with prior transferred from a GP; comparison between the baseline (using a deep BNN; [Tran et al., 2022]) and ours (single hidden layer BNN with varying widths). *Periodic* activation (Sec. 3.3) and an activation realized by a NN with SiLU own activation (*default*) were used.

Method →	our ($width = 128$, *Periodic act.*)		our ($width = 128$)		our ($width = 1000$)		Baseline
Metric ↓	- regularization	+ regularization	- regularization	+ regularization	- regularization	+ regularization	—
RMSE	2.9067±0.8257	2.8967±0.8258	2.8643±0.8386	**2.8348±0.8371**	2.9189±0.8408	3.0059±0.9068	2.8402±0.8986
NLL	2.5057±0.1870	2.5072±0.1904	2.4937±0.1798	2.4862±0.1763	2.5122±0.1871	2.5971±0.1206	**2.4778±0.1481**

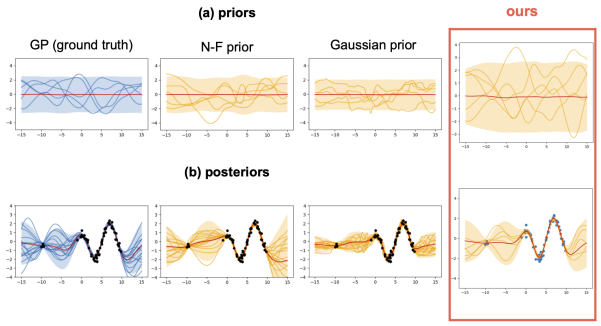


Figure 3: Prior **(a)** and posterior **(b)** predictive distributions for a BNN with trained parameters priors and activations (ours; 4th column), and for Tran et al. [2022] approach with different prior realizations (Gaussian (3 hidden layers; 3rd column) and NF (2 hidden layers; 2nd). The first column illustrates the ground truth (GP). Numerical results complementing the figures we provide in the Appendix.

els of parameter priors and activations. The target was $\mathrm{GP}(0, \text{Matérn}(\nu = 5/2, l = 1))$. For each configuration, we conducted several training iterations and measured the final (converged) loss multiple times. We observe that learning activations alone is not sufficient for good fits, but when combined with learning priors, it significantly improves the results.

Fig. 11 in the Appendix presents the results of a similar experiment conducted for a GP with the Periodic kernel ($\ell = 1, p = 1$). The best performance is observed for the activation introduced in Sec. 3.3, however, increasing the number of layers or neurons can also improve the fit for the activation modeled by an MLP.

We supplement the figure with additional plots showing samples from runs where poor convergence was observed. As explained in Sec. 3.7, due to unfortunate initialization or the insufficient modeling capacity of ϕ, gradient-based optimization may fail to capture a GP functional prior. In such cases, the optimization gets stuck in a local optimum, and the observed final loss differs significantly from the values obtained in other scenarios.

4.2 DO EXPRESSIVE FUNCTION-SPACE PRIORS REQUIRE NETWORKS TO BE DEEP?

Function-space priors require expressive models, which can be achieved either through a multi-layer neural network architecture or with a shallow BNN with more flexible (*e.g.,* learnable) activations. [Tran et al., 2022] explored the former, focusing on multi-layer networks. Instead of considering BNNs with wide layers – where the theoretical results in Sec. 2 apply – they *postulated* that a deep network can model a functional prior after tuning the network's parameter priors to match a target functional prior.

[Tran et al., 2022] investigated several types of priors for BNN weights, including Gaussian, hierarchical, and Normalizing Flow (NF)-based priors. They argue that only the NF prior is flexible enough to capture complex distributions. While NFs [Rezende and Mohamed, 2015] can model intricate priors across all weights, by assigning a distinct prior to each weight they violate the assumptions of the Central Limit Theorem (CLT), preventing the BNN from converging to a GP (*see* Sec. 2). Although [Tran et al., 2022] achieved their best results using this heuristic NF prior, among the considered priors, the theoretical guarantees hold only for the Gaussian and hierarchical priors, and only in the context of wide BNNs.

Beyond the lack of theoretical justification, using complex priors such as NFs or requiring deep networks introduces additional challenges in posterior inference. MCMC-based methods [Chen et al., 2014, Del Moral et al., 2006] become computationally expensive, while approximate inference techniques [Hoffman et al., 2013, Ritter et al., 2018] may lack expressiveness.

In Fig. 4 we compare our approach against the behavior of their methods for varying numbers of layers and activations. Specifically, we evaluate posterior in a 1D regression task (Fig. 3), both on training data and in out-of-distribution (OOD) regions. The results show that multiple hidden layers are required for the Gaussian prior. In contrast, for the NF priors, increasing depth leads to worse performance in OOD regions – illustrating an example of the heuristic's failure. For both methods, we note high sensitivity to activation change. NFs perform best for one or two layers, but only with RBF activation. For Swish/ReLU they fail to capture the data (high RMSE/NLL). Please see also Sec. E.

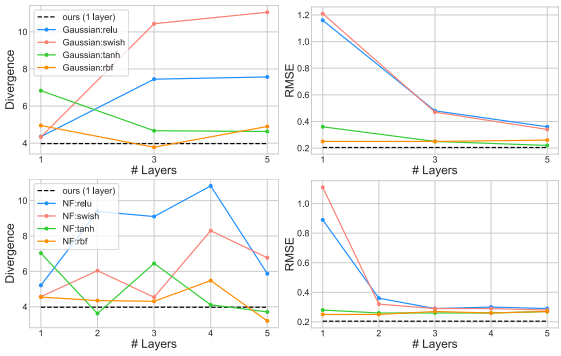


Figure 4: Comparison of posterior predictive quality across the methods from [Tran et al., 2022] for the problem from Fig. 3, considering different priors, activation functions, and numbers of hidden layers. We evaluate both *out-of-distribution* (in-between region) performance (**left**, Wasserstein divergence) and *in-distribution* performance (**right**, RMSE). For the Gaussian prior (**top**), the results align with the assumption by the authors – adding more layers improves the GP approximation. Contrary, for the Normalizing Flow prior (**bottom**), the best performance is observed with one or two hidden layers, while deeper networks suffer from instability, highlighting failure of the heuristic. Our method (**black line**) consistently outperforms the baselines.

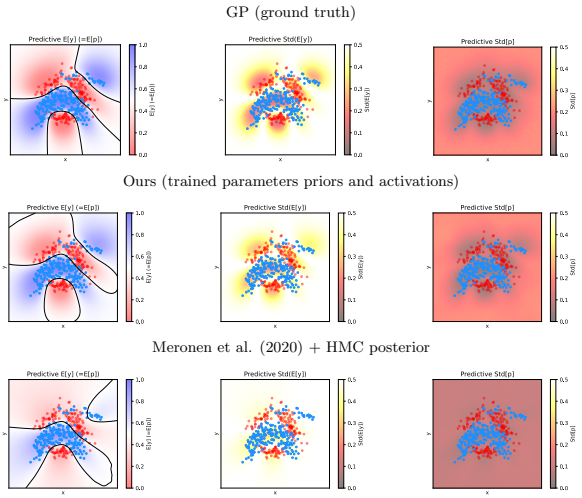


Figure 5: Posterior predictive distributions for a BNN with trained parameter priors and activations (ours; 2nd row) and for a BNN with analytically derived activations (3rd row). The first column illustrates class probabilities, the second column shows the total variance in class predictions, and the last column depicts the epistemic uncertainty component of the total uncertainty. Here, we show posteriors obtained using HMC, while in Sec. D.4 of the Appendix, we provide an extended version of the figure that includes results obtained using the original code from Meronen et al. [2020] (which uses MC-Dropout instead of MCMC), along with additional numerical results.

For further evaluation, we compare our approach with Tran et al. [2022] on a range of regression tasks, including both synthetic and real-world datasets (e.g., the Boston dataset). The results are shown in Fig.3 and Tab.1, respectively. Additional discussion and results for several other regression tasks are provided in the Appendix. We closely follow the settings picked by Tran et al. [2022] and still observe that our method performs comparably or better than the baseline, while requiring only a single-hidden-layer BNN.

Additionally, we checked the findings from [Wu et al., 2024] that moment matching helps biasing optimisation towards better solutions. For the UCI regression task, we included an additional moment matching regularization term: $\mathcal{R}(p_{nn}, p_{gp}) = \big(\mathbb{E}_{p_{nn}}[var(f)] - \mathbb{E}_{p_{gp}}[var(f)]\big)^2 + \big(\mathbb{E}_{p_{nn}}[kurtosis(f)] - \mathbb{E}_{p_{gp}}[kurtosis(f)]\big)^2 + \big(\mathbb{E}_{p_{nn}}[skeweness(f)] - \mathbb{E}_{p_{gp}}[skeweness(f)]\big)^2$. However, we have *not* observed any significant improvements.

4.3 CAN LEARNED ACTIVATIONS MATCH PERFORMANCE OF CLOSED-FORM ONES?

Deriving suitable neural activations to match BNNs to GPs is a formidable challenge. For example, Meronen et al. [2020] derived recently an analytical activation function for the popular Matérn kernel. However, their solution relies on the assumption that the priors on the BNN weights follow binary white noise. Such the prior hinders existing posterior learning algorithms, making the proposed solution impractical. Nevertheless, Meronen et al. [2020] shows that their solution can work with more convenient Gaussian priors, albeit as a suboptimal approximation. In this work, we propose a general gradient-based alternative that can also be applied to the GPs with the Matérn kernel and we demonstrate that our black-box approach can match or even surpass the performance of the aforementioned approximation.

The empirical evaluation we performed for the 2D data classification problem following the setting of Meronen et al. [2020]. For our method, we used a BNN with Gaussian priors with trained variances, where the activation function was modeled by an NN with a single hidden layer consisting of 5 neurons using SiLU activation. For the baseline, we trained just variances and used the fixed activation. Posterior distributions were generally obtained using a HMC sampler.

The results presented in Fig. 5 (and extended results provided in Sec. D.4 in the Appendix) demonstrate that our method enhances the match between the posteriors of a BNN and the desired target GP. Note that Meronen et al. [2020] originally used MC Dropout to obtain the posteriors, achieving much worse results than those obtained with HMC. For fairness in comparison, we tested both MC Dropout and HMC. In terms of performance, our model not only *captures class probabilities accurately but also*

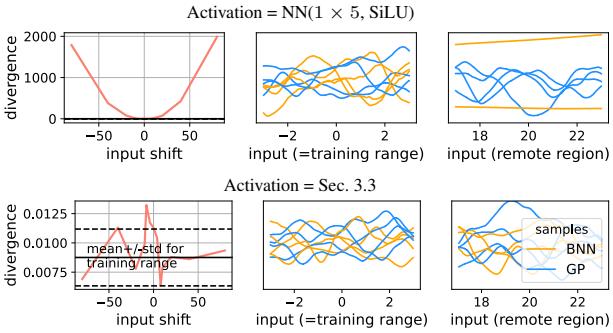


Figure 6: Sensitivity of priors transferred from a GP to a BNN with respect to shifts in the input range for learned activations: (top) activation realized by a NN with a single hidden layer containing 5 neurons and SiLU own activation; (bottom) periodic activation introduced in Sec. 3.3, which is robust to input range shifts. Priors were trained on random inputs $X \in [-3, 3]$. The first column shows the mismatch between the desired target GP and the trained BNN prior for shifted input ranges, measured by the 2-Wasserstein loss (lower is better). The remaining columns present samples from the priors for two ranges: one matching the training range and one far from it.

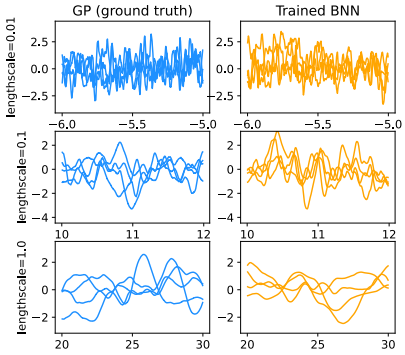


Figure 7: Comparison of samples from a GP (left) and a trained BNN (right) using priors and activations (periodic activation as in Sec. 3.3) conditioned on the GP's hyperparameter (e.g., lengthscale). The BNN was trained on random inputs $X \in [-3, 3]$. Regardless of input range and the hyperparameter's value, samples from both models appear *indistinguishable*, i.e., the BNN imitates the GP perfectly.

adeptly handles the total variance in class predictions and the epistemic uncertainty component, which are crucial for robust decision-making under uncertainty.

4.4 CAN STATIONARITY BE INDUCED WITH LEARNED PERIODIC ACTIVATIONS?

Trained priors for BNNs are inherently local, effectively mimicking GP behavior only within the range of training inputs X. As illustrated in Fig. 6 (top), outside this range, the BNN's learned prior may diverge from the desired behavior, and properties such as stationarity or smoothness may not be preserved beyond the observed input domain. We however can not only detect this issue by noting high values of the divergence D, but also mitigate it by using activations with appropriate structural biases. Specifically, the periodic activation introduced in Sec. 3.3 proves to be robust to input domain translations, as demonstrated in Fig. 6 (bottom).

4.5 IS MODEL SELECTION FOR BNNS WITH TRANSFERRED PRIORS POSSIBLE?

A BNN with a functional prior learned from a GP essentially inherits the properties dictated by the GP, including those determined by specific values of the GP's hyperparameters. While the hyperparameters of the source GP can be tuned and optimized, the BNN does not inherently possess this capability, requiring researchers to either pre-optimize the hyperparameters or look for suitable workarounds [Tran et al., 2022]. Conditioning the priors on weights, biases, and activation, as explained in Sec. 3.4, addresses this limitation, ultimately closing the BNN-GP equivalence gap. Fig. 7 shows results for a prior trained to adapt to a varying lengthscale of a Matérn kernel. The prior closely follows the behavior of the ground truth model.

To validate the usefulness of this method, we performed marginal likelihood maximization on the data from Fig. 3, using gradient-based optimization for both the GP with the Matérn kernel and our BNN. The optimal lengthscale found for the BNN ($\ell = 1.65$) closely matches the one found for the GP ($\ell = 1.84$).

4.6 DOES IT SCALE?

Learning posteriors for Gaussian Processes (GPs) is challenging, as exact inference scales cubically with the number of data points Rasmussen and Williams [2005]. To address this, scalable approximations such as Stochastic Variational Gaussian Processes (SVGP) [Hensman et al., 2015] have been proposed for practical applications. SVGP employs stochastic variational gradients to optimize a fixed set of inducing points, typically (and also here) 1024.

On the other hand, there exist a number of posterior inference methods for BNNs. For the experiments presented in the previous sections, we utilized HMC. However, Variational Inference (VI) with Normalizing Flows [Rezende and Mohamed, 2015] offers a scalable alternative, albeit with challenges related to both the accuracy of posterior approximation and selection of hyperparameters. However, the former limitation can be mitigated by employing RealNVPs [Dinh et al., 2016] with appropriate capacity for approximating posteriors. Agrawal and Domke [2024] showed

Table 2: Mean test-set NLL and RMSE for the HouseElectric dataset. The optimal prior lengthscale ℓ^* and additive Gaussian noise were *learned* by maximizing $\mathcal{L}$ with $\beta = 1$. Results for $\ell = 10\ell^*$ and $\ell = 0.1\ell^*$ highlight the importance of prior hyperparameter selection. Additionally, using a model of activation function with more neurons improves performance. The best results are achieved by combining these modifications with $\beta = 0.4$.

	Ours with $\beta = 1$				Ours with $\beta = 0.4$		$\pm$	SVGP	Exact GP
	learned	$\ell = 0.1 \times \ell^*$	$\ell = 10 \times \ell^*$	$\ell = \ell^*$ & improved activation	$\ell = \ell^*$	$\ell = \ell^*$ & improved activation	std.	(baseline)	(BBMM)
$\downarrow$ Test RMSE	0.0549	0.0550	0.0560	0.0550	0.0537	**0.0535**	≤ 0.0007	0.0566	0.055
$\downarrow$ Test NLL	-1.4925	-1.4850	-1.4550	-1.4980	-1.5107	**-1.5155**	≤ 0.0064	-1.4600	$-0.152^\dagger$

† Due to lack of available implementation, we failed to reproduce results from Wang et al. [2019] and provide here values copied directly from the paper.

that RealNVPs can exhibit sufficient fidelity to match posteriors for models with complex priors.

For VI learning is performed using gradients of

$$\mathcal{L} = \mathbb{E}_{q(w|\zeta)} \left[\log p(\mathcal{D}|w)\right] - \beta \cdot \mathrm{KL}\left[q(w|\zeta) \,\|\, p(w)\right]$$
$$\approx \frac{1}{S} \sum_{w \sim q(w|\zeta)} \left(\frac{|\mathcal{D}|}{|\mathcal{B}|} \log p(\mathcal{B}|w) + \beta \cdot (p(w) - q(w|\zeta)) \right)$$

where we estimate the Evidence Lower Bound (ELBO) objective for Stochastic Variational Inference (SVI) [Hoffman et al., 2013] using minibatches $\mathcal{B} \subset \mathcal{D}$ with $|\mathcal{B}| = 10{,}240$. The expectation is approximated with $S = 128$ reparameterized Monte Carlo samples, where $q(w|\zeta)$ is the posterior approximation modeled by a RealNVP.

Tab. 2 summarizes the test set performance on the UCI Household Power Consumption (HouseElectric) regression dataset. This dataset contains approximately 2M rows, split into training (7/9) and test (2/9) subsets, and is the largest dataset used to evaluate GPs [Wang et al., 2019].

The regression model for the data has two hyperparameters: the prior *lengthscale* ℓ of the Matérn ($\nu = 3/2$) kernel and the additive *observation noise* scale in the Gaussian likelihood. We selected these hyperparameters by maximizing the ELBO $\mathcal{L}$, which for $\beta = 1$ provides a proper lower bound for the evidence. This was possible thanks to employing the conditional priors introduced in Sec. 3.4.

Variational Inference (VI) often performs better with $\beta < 1$ [Higgins et al., 2017], which also holds in our case - setting $\beta = 0.4$ (a typical value) improved both RMSE and NLL. However, as indicated by discrepancies with the Exact GP results, this improvement is likely due to the reduced influence of the learned GP prior, which may be suboptimal for this particular dataset.

To analyze the importance of prior selection, we also computed posteriors for lengthscales reduced by a factor of ten ($\ell = 0.1 \times \ell^*$) and increased tenfold ($\ell = 10 \times \ell^*$). In both cases, we observed a drop in NLL, highlighting the sensitivity of performance to choice of the prior.

We further tested a learned prior with slightly *improved activation* model. We used a neural network with two hidden layers of width 10, instead of the default single layer with five neurons. This resulted in a 0.15% improvement in the 2-Wasserstein loss. Although this represents a minor enhancement in prior fidelity, it still led to noticeable improvements in posterior quality.

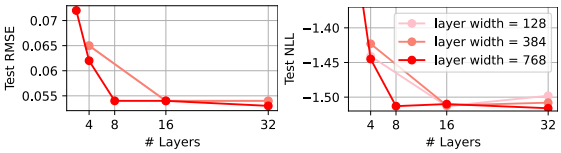


Figure 8: Impact of the architecture of $q(w|\zeta)$ parameterized by *RealNVP* [Dinh et al., 2016]: (1) although adding more layers improves performance, the gains become marginal for > 16 layers; (2) wider flows require fewer layers to achieve performance saturation.

Finally, we investigated the capacity of the RealNVP used for posterior approximation. Fig. 8 compares test set performance for varying the number of flow layers and the width of each layer. We observed that increasing the number of layers improves performance, but the gains become marginal beyond 16 layers, and that wider flows require fewer layers to achieve performance saturation.

5 CONCLUSION

In this paper, we addressed the problem of transferring functional priors for wide Bayesian Neural Networks to replicate desired a priori properties of Gaussian Processes. Previous approaches typically focused on learning distributions over weights and biases, often requiring deep BNNs for sufficient flexibility. We proposed an alternative approach by also learning activations, providing greater adaptability for shallow models and eradicating the need for task-specific architectural designs. To the best of our knowledge, we are the first to explore learning activations in this context. Moreover, to further enhance adaptability of these transferred priors, we came up with the idea of conditioning them with hypernetworks, which opens a new interesting future research direction. To demonstrate the flexibility and effectiveness of the proposed methods, we conducted a comprehensive experimental study validating our ideas.

Author Contributions

Marcin Sendera actively participated in shaping the project across all its stages, contributing both during discussions and through hands-on development. He explored and assessed multiple variants of the Wasserstein loss, as well as a range of learnable activation functions proposed in the literature, including piecewise linear (PWL), Padé, and others. He proposed to use the closed-form solution for the Wasserstein metric. He was primarily responsible for implementing and executing the experiments reported in Figures 3 and 4, and in Table 1, ensuring the reproducibility and robustness of the results. He contributed to writing of the manuscript, particularly in sections related to empirical results.

Amin Sorkhei participated in discussions at the initial stage of the project. He implemented and conducted the initial experiments involving the mapping of Gaussian Process priors to Bayesian Neural Networks using the Kullback-Leibler divergence and a learnable activation parameterized by a neural network.

Tomasz Kuśmierczyk conceived the project and supervised its development across all stages, including the core theoretical contributions and experimental design. He introduced the idea of learning activations and proposed the use of Wasserstein-based loss and regularizations. He proposed to use neural networks to parameterize learnable activations. He developed and implemented the periodic (Section 3.3) and conditioned activations (Section 3.4). He was responsible for implementing and executing the experiments presented in Figures 2, 5, 6, 7, 8, and Table 2. He was responsible for the majority of the manuscript writing as well as for shaping its final form.

Acknowledgements

We thank Nikolay Malkin and Jacek Tabor for helpful discussions at early stages of this project and revising the manuscript.

This research is part of the project No. **2022/45/P/ST6/02969** co-funded by the National Science Centre and the European Union Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie grant agreement No. 945339. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

This research was in part funded by National Science Centre, Poland, **2022/45/N/ST6/03374**.

We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. **PLG/2023/016302**.

References

Abhinav Agrawal and Justin Domke. Disentangling impact of capacity, objective, batchsize, estimators, and step-size on flow VI. *arXiv preprint arXiv:2412.08824*, 2024.

Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4 (3):195–266, 2012. doi: 10.1561/2200000036.

Edwin V Bonilla, Ying Chai, and Christopher K Williams. Multi-task Gaussian Process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2008.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

Lucas F Buzuti and Carlos E Thomaz. Fréchet autoencoder distance: a new approach for evaluation of generative adversarial networks. *Computer Vision and Image Understanding*, 235:103768, 2023.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57 (9):1–29, 2024.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pages 1683–1691. PMLR, 2014.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo Samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68 (3):411–436, 05 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping Gaussian Process priors to Bayesian Neural Networks. In *NIPS Bayesian deep learning workshop*, volume 3, 2017.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Characterizing and warping the function space of Bayesian Neural Networks. In *NeurIPS Workshop on Bayesian Deep Learning*, page 18, 2018.

Vincent Fortuin. Priors in bayesian deep learning: A review. *International Statistical Review*, 90(3):563–591, 2022. doi: https://doi.org/10.1111/insr.12502.

Pierre Goovaerts. *Geostatistics for natural resources evaluation*. Oxford University Press on Demand, 1997. ISBN 978-0195115383.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *ICLR*. International Conference on Representation Learning, 2017.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA, 09–12 May 2015. PMLR.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR2017, Conference Track Proceedings*, 2017.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14:1303–1347, 2013.

Jiri Hron, Yasaman Bahri, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Exact posterior distributions of wide Bayesian Neural Networks. *arXiv preprint arXiv:2006.10541*, 2020.

Jiri Hron, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Wide Bayesian neural networks have a simple weight posterior: theory and accelerated sampling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8926–8945. PMLR, 17–23 Jul 2022.

Theofanis Karaletsos and Thang D Bui. Hierarchical Gaussian Process priors for Bayesian Neural Network weights. *Advances in Neural Information Processing Systems*, 33: 17141–17152, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian Processes. *arXiv preprint arXiv:1711.00165*, 2017.

Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Anton Mallasto and Aasa Feragen. Learning from uncertain curves: The 2-Wasserstein metric for Gaussian Processes. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Takuo Matsubara, Chris J Oates, and François-Xavier Briol. The ridgelet prior: A covariance function approach to prior specification for Bayesian Neural Networks. *The Journal of Machine Learning Research*, 22(1):7045–7101, 2021.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process behaviour in wide deep Neural Networks. *ICLR*, 2018.

Lassi Meronen, Christabella Irwanto, and Arno Solin. Stationary activations for uncertainty calibration in deep learning. *Advances in Neural Information Processing Systems*, 33:2338–2350, 2020.

Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. *Advances in Neural Information Processing Systems*, 34:1673–1685, 2021.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations*, 2019.

Pablo Morales-Alvarez, Daniel Hernández-Lobato, Rafael Molina, and José Miguel Hernández-Lobato. Activation-level uncertainty in deep neural networks. In *International Conference on Learning Representations*, 2020.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

Tim Pearce, Russell Tsuchida, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in Bayesian Neural Networks: Kernel combinations and periodic functions. In *Uncertainty in Artificial Intelligence*, pages 134–144. PMLR, 2020.

Janis Postels, Hermann Blum, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. The hidden uncertainty in a neural network's activations. In *Proceedings of the Bayesian Deep Learning Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, 2015.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th international conference on learning representations, ICLR 2018-conference track proceedings*, volume 6. International Conference on Representation Learning, 2018.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pages 4644–4653. PMLR, 2018.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 5690–5701, 2019.

Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022.

Russell Tsuchida, Fred Roosta, and Marcus Gallagher. Richer priors for infinitely wide multi-layer perceptrons. *arXiv preprint arXiv:1911.12927*, 2019.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact Gaussian Processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

Veit David Wild, Robert Hu, and Dino Sejdinovic. Generalized variational inference in function spaces: Gaussian measures meet bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:3716–3730, 2022.

Christopher Williams. Computing with infinite networks. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

Mengjing Wu, Junyu Xuan, and Jie Lu. Functional wasserstein bridge inference for bayesian deep learning. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Revisiting the Equivalence of Bayesian Neural Networks and Gaussian Processes: On the Importance of Learning Activations (Supplementary Material)

Marcin Sendera[*1,2] **Amin Sorkhei** **Tomasz Kuśmierczyk**[*1]

[1]Jagiellonian University
[2]Mila, Université de Montréal

We supplement the main text here with a description of the related work, a discussion of computational costs, and additional details of the experimental setup, followed by additional numerical results. Our code we made publicly available [1].

A RELATED WORK

The relationship between (B)NNs and GPs has been a subject of significant research interest. Much of this work has been motivated by the desire to better understand NNs through their connection to GPs. A seminal result by Neal [1996], Williams [1996], later extended to deep NNs by Lee et al. [2017], Matthews et al. [2018], shows that infinitely wide layers in NNs exhibit behavior akin to GPs. While we explore a similar setting, our focus is the inverse: implementing function-space priors, specifically GP-like, within BNNs.

This problem was previously addressed by Meronen et al. [2020], who proposed an activation function for BNNs that corresponds to the popular Matérn kernel. Due to the complexity of deriving analytical solutions, other works, such as Flam-Shepherd et al. [2017, 2018], Tran et al. [2022], have explored gradient-based optimization to learn priors on weights and biases. However, their approach requires deep networks with complex distributions and additionally, presents a challenge in learning posterior. In contrast, we achieve high-fidelity functional priors in shallow BNNs by matching *both* priors on parameters and activations.

Finding accurate posteriors for deep and complex models such as BNNs is notoriously challenging due to their high-dimensional parameter spaces and complex likelihood surfaces. However, for single-hidden-layer wide BNNs, it has been recently shown that posterior sampling via Markov Chain Monte Carlo can be performed efficiently [Hron et al., 2022]. Moreover, research has demonstrated that the exact posterior of a wide BNN weakly converges to the posterior corresponding to the GP that matches the BNN's prior [Hron et al., 2020]. This allows BNNs to inherit GP-like properties while preserving their advantages.

An alternative line of work touching efficient learning for BNNs in function space has explored variational inference. Sun et al. [2019] introduced functional variational BNNs (fBNNs) that maximize an Evidence Lower Bound defined over functions. However, Burt et al. [2020] highlighted issues with using the Kullback-Leibler divergence in function space, leading to ill-defined objectives. To overcome this, methods like Gaussian Wasserstein inference [Wild et al., 2022] and Functional Wasserstein Bridge Inference [Wu et al., 2024] leverage the Wasserstein distance to define well-behaved variational objectives. These approaches try to incorporate functional priors into variational objective for deep networks. In contrast, in our method transfer of priors to shallow BNNs remains independent and fully separated from an inference method.

Finally, other works worth mentioning include Meronen et al. [2021], who explored periodic activation functions in BNNs to connect network weight priors with translation-invariant GP priors. Pearce et al. [2020] derived BNN architectures to mirror GP kernel combinations, showcasing how BNNs can produce periodic kernels. Karaletsos and Bui [2020] introduced a hierarchical model using GP for weights to encode correlated weight structures and input-dependent weight priors, aimed at

[1]https://github.com/gmum/bnn-functional-priors

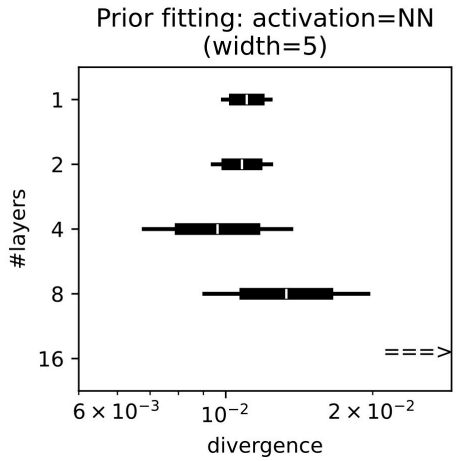


Figure 9: Prior matching quality: fixed width.

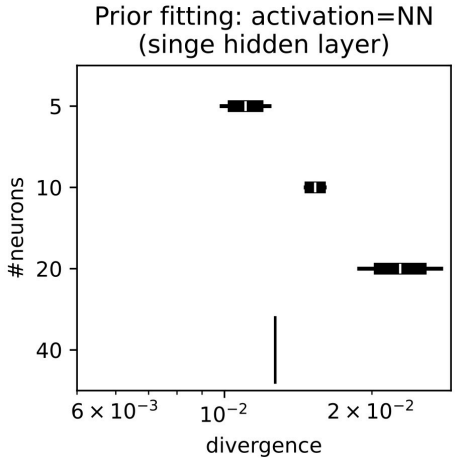


Figure 10: Prior matching quality: fixed depth.

regularizing the function space. Matsubara et al. [2021] proposed using ridgelet transforms to approximate GP function-space distributions with BNN weight-space distributions, providing a non-asymptotic analysis with finite sample-size error bounds. Finally, Tsuchida et al. [2019] extended the convergence of NN function distributions to GPs under broader conditions, including partially exchangeable priors. A more detailed discussion of related topics can be found for example, in Sections 2.3 and 4.2 of [Fortuin, 2022].

B COMPUTATIONAL AND ARCHITECTURAL ADVANTAGES

Our method involves an initial prior fitting step, which precedes the posterior inference step. The prior fitting step takes approximately one hour for the HouseElectric dataset (Section 4.6); however, one can alternatively use a prior from a library of pre-trained priors, thereby amortizing the cost of this step.

Our experiments show that substituting fixed activations (ReLU/TanH, ~ 0.5s/iteration) with learnable NN activations (single hidden layer, 5 neurons, ~ 2.3s/iteration) or periodic activations (~ 2.7s/iteration) increases the time per iteration by approximately a factor of 4 on our 16-core CPU setup. On the other hand, learning parameter priors alongside activations has only a marginal effect on computation time compared to fixing priors and only learning activations.

Additionally, we conducted a study measuring both the quality of prior fit and the runtime for a prior modeled by a neural network with varying numbers of layers and widths. The respective results can be found in Figures 9, 10 and Tables 3, 4. These results suggest that increasing the number of layers up to a certain depth (e.g., 4 layers in this case) may marginally improve the fidelity of the solutions, whereas increasing the network width tends to make convergence harder. In either case, using a larger neural network consistently leads to longer convergence times.

Table 3: Prior matching time (activation=NN, width=5)

#layers	iteration time [s]
1	2.34
2	3.10
4	4.57
8	7.38
16	15.03

Table 4: Prior matching time (activation=NN, single hidden layer)

#neurons	iteration time [s]
5	2.34
10	2.59
20	3.95
40	7.59

Once the prior is established (or a pre-trained one is used), posterior inference can be performed. Our approach offers here computational advantages, particularly for large datasets where exact GP inference is prohibitive. For example, RealNVP-based variational inference (Section 4.6) converges in about 2 hours on a single GPU for HouseElectric, while exact GP

inference can take over three days on a single GPU, or 1.5 hours when parallelized on 8 GPUs using methods such as BBMM [Wang et al., 2019]. Furthermore, during posterior inference using HMC for the data in Fig. 3, we observed no significant differences in step time ($\sim$ 1s) between fixed and learnable activations, though this may vary with implementation and hardware.

Finally, beyond computational performance, BNNs provide greater modularity and extensibility. Single-layer BNNs can serve as components in larger neural architectures, for example, as Bayesian last layers (with remaining layers trained point-wise). This enables leveraging architectural innovations from deep learning in ways that GPs cannot easily accommodate. Our method could replace existing approaches in models like SNGP [Liu et al., 2020], potentially offering improved prior control (as random Fourier feature GPs are limited to specific stationary kernels), more off-the-shelf end-to-end training options (e.g., SVGD), and potentially enhanced performance, as suggested by our comparisons against SVGP in Section 4.6.

C LEARNED GP KERNELS

In this paper, we show that our approach allows for inducing a GP-like behavior onto single-layer BNNs without being restricted to a family of GP kernels. In particular, we induced the following GP kernel behavior, while performing the mentioned experiments:

- Matérn (3/2) - house electric regression;
- Matérn (5/2) - inducing stationarity, ablation on learning activations and priors, 2d classification;
- RBF - 1d regression;
- RBF + ARD - UCI regression;
- Periodic - ablation on learning activations and priors.

D EXPERIMENTAL DETAILS

D.1 CAN LEARNED ACTIVATIONS ACHIEVE MORE FAITHFUL FUNCTION-SPACE PRIORS? – ADDITIONAL FIGURES

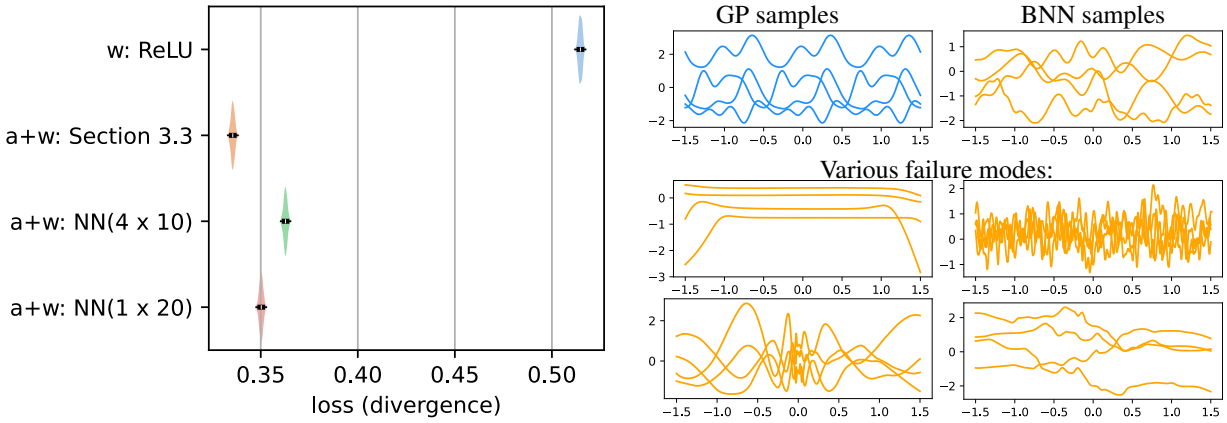


Figure 11: Quality of matching BNNs to the prior of a GP with the Periodic kernel ($\ell = 1$, $p = 1$). (Left): Evaluation of models with trained parameter priors (denoted by w) and with trained both weights priors and activations (denoted by a+w). (Right): Samples from the ground truth GP, a well-fitted BNN and BNN outliers with poor final loss. As explained in Sec. 3.7, or because of the insufficient modeling capacity of ϕ, gradient-based optimization may fail to capture a GP functional prior.

D.2 1-DIMENSIONAL REGRESSION - COMPARISON WITH Tran et al. [2022]

In order to illustrate the abilities of our approach on a standard regression task and for a fair comparison with another method optimizing the Wasserstein distance, we followed the exact experimental setting presented in Tran et al. [2022]. We used a GP with an RBF kernel (*length scale* ℓ=0.6, *amplitude*=1.0) and a Gaussian likelihood (*noise variance*=0.1) as the ground truth.

For a baseline, we utilized the approach by Tran et al. [2022], consisting of a BNN with 3 layers and 50 neurons each, using TanH activation function. We considered all three prior realizations presented in that work: a simple Gaussian prior, a hierarchical prior, and a prior given by a Normalizing Flow. On the other hand, our model configuration consisted of a BNN with Gaussian priors centered at $\mathbf{0}$ and trained variances, where the activation function was modeled by a neural network with a single hidden layer with 5 neurons and SiLU activation. Posterior distributions were obtained using an HMC sampler.

We found that our approach allows for achieving the same or better results than the baseline by Tran et al. [2022], both in visual and numerical comparisons. We obtained better results in terms of distributional metrics as presented in Tab. 5 and Tab. 6 and getting better (visually) posterior distributions (see Fig. 3) without utilizing any computationally heavy prior realization (like, *e.g.,* Normalizing Flow).

Table 5: Comparison of trained (function-space) priors in a 1D regression task: [Tran et al., 2022] vs. our method. Whereas Tran et al. [2022] considered three different priors on parameters (including Gaussian, Hierarchical, and Normalizing Flow) for deep BNNs, our implementation consists of a single-hidden-layer BNN with standard Gaussian priors with learned variances and trained activation. The methods were compared using a set of distributional metrics against the ground truth provided by a GP. We compared prior distributions of functions over a range of X covering regions with and without data (to account also for overconfidence far from data). The lower, the better.

Setting	1-Wasserstein	2-Wasserstein	Linear-MMD	Poly-MMD $\times 10^3$	RBF-MMD	Mean-MSE	Mean-L2	Mean-L1	Median-MSE	Median-L2	Median-L1
Priors:											
Gaussian prior	7.52	7.77	-6.08	1.158	0.036	0.003	0.058	0.045	0.004	0.063	0.052
Hierarchical prior	8.14	8.37	0.22	1.421	0.027	0.004	0.064	0.059	0.008	0.090	0.074
Normalizing Flow prior	7.61	8.09	-8.192	0.393	0.019	0.004	0.061	0.051	0.004	0.066	0.052
ours	**6.86**	**7.06**	**-9.40**	**-0.016**	**0.008**	**0.001**	**0.032**	**0.027**	**0.002**	**0.048**	**0.041**

Table 6: Comparison of trained (function-space) posteriors in a 1D regression task: [Tran et al., 2022] vs. our method. Whereas Tran et al. [2022] considered three different priors on parameters (including Gaussian, Hierarchical, and Normalizing Flow) for deep BNNs, our implementation consists of a single-hidden-layer BNN with standard Gaussian priors with learned variances and trained activation. The methods were compared on RMSE and NLL of test data.

Setting	RMSE	NLL
Posteriors:		
Gaussian prior	0.2559	**0.2456**
Hierarchical prior	0.2461	0.2613
Normalizing Flow prior	0.2534	0.2550
ours	**0.2045**	0.3044

D.3 UCI REGRESSION - COMPARISON WITH Tran et al. [2022]

In addition, we compare against Tran et al. [2022] also on UCI regression tasks, which have more input dimensions d – between 8 and 12, while having 1-dimensional output. We use 10-split of training datasets as in typical in this direction of research.

For the baseline, we followed the exact experimental setting presented in Tran et al. [2022]. We used a GP with an RBF kernel (*length scale* $\ell = \sqrt{2.0 * input\_dim}$ and *amplitude*=1.0, ARD) and a Gaussian likelihood as the ground truth. We set the noise variance value and architectures for the specific datasets as in the mentioned baseline paper.

On the other hand, our model configuration consisted of a BNN with Gaussian priors centered at $\mathbf{0}$ and trained variances, where the activation function was modeled by a neural network with a single hidden layer with 5 neurons and SiLU activation. Posterior distributions were obtained using an HMC sampler.

We found that our approach allows for achieving the same or better results than the baseline, in terms of numerical values (*RMSE* and *NLL*) as presented in Tab. 7.

Table 7: Extended results for a set of UCI regression tasks (various input dimensionality d) with prior transferred from a GP; comparison between the baseline (using a deep BNN; [Tran et al., 2022]) and ours (single hidden layer BNN with 128 neurons; w/o regularization) and an activation realized by a NN with SiLU own activation were used.

Dataset →	Boston ($d = 12$)		Concrete ($d = 8$)		Energy ($d = 8$)		Protein ($d = 9$)		Wine ($d = 11$)	
Method ↓ Metric →	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*	*RMSE*	*NLL*
baseline	2.8402$\pm$0.8986	2.4778$\pm$0.1481	4.4628$\pm$0.7511	2.9728$\pm$0.0876	0.3431$\pm$0.0613	0.3482$\pm$0.1607	0.5038$\pm$0.0093	0.7447$\pm$0.0142	0.4736$\pm$0.0420	0.8725$\pm$0.0285
ours	2.8643$\pm$0.8386	2.4937$\pm$0.1798	4.9143$\pm$0.7528	3.0201$\pm$0.1011	0.3692$\pm$0.0566	0.4064$\pm$0.1347	0.4957$\pm$0.0058	0.7251$\pm$0.0094	0.4833$\pm$0.0398	0.8673$\pm$0.0255

D.4 CAN LEARNED ACTIVATIONS MATCH PERFORMANCE OF CLOSED-FORM ONES? – COMPARISON WITH Meronen et al. [2020]

For the comparison against Meronen et al. [2020], we closely followed their experimental setting and used a GP with a Matérn kernel ($\nu = 5/2$, ℓ=1) as the ground truth. For a baseline, we utilized the approach by Meronen et al. [2020], where the authors derived analytical activations to match the Matérn kernel. Our model configuration consists of a BNN with Gaussian priors and trained variances, where the activation function was modeled by a neural network with a single hidden layer consisting of 5 neurons using SiLU activation. Posterior distributions were generally obtained using an HMC sampler, except in the case of Meronen et al. [2020], where the original implementation employed MC Dropout to approximate the posterior. However, for completeness and fairness in comparison, we also generated results using an HMC-derived posterior for the model by Meronen et al. [2020]. Additional tests (see Tab. 8) were conducted on BNNs with fixed parameter priors (*Default*=Gaussian priors with a variance of 1, and *Normal*=Gaussian priors normalized by the hidden layer width) and with activation functions including ReLU and TanH.

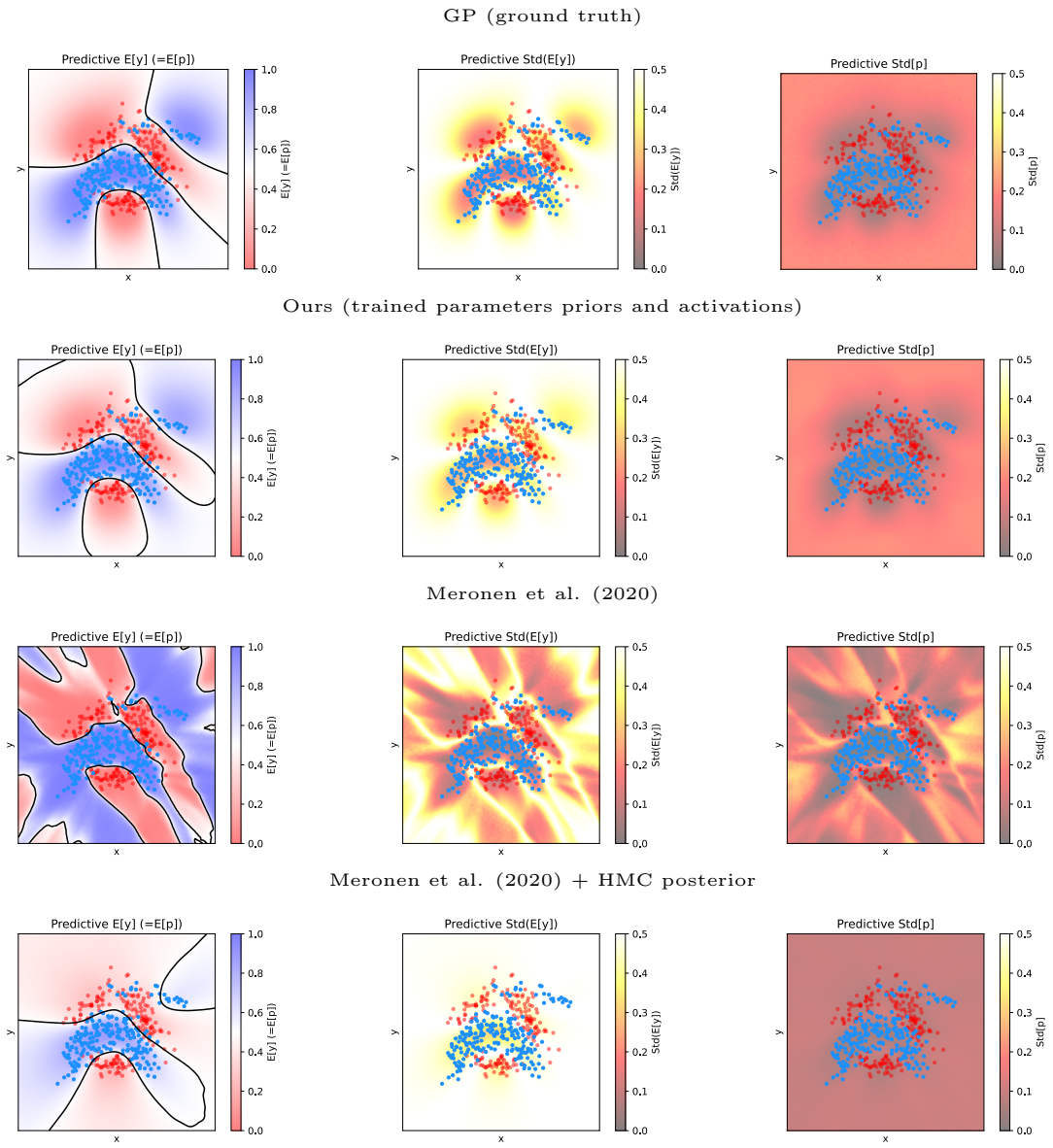


Figure 12: Posterior predictive distributions for a BNN with trained parameters priors and activations (ours; 2nd row), and for a BNN with analytically derived activation (3rd and 4th row). The first column illustrates class probabilities, the second column shows the total variance in class predictions, and the last column depicts the epistemic uncertainty component of the total uncertainty.

Table 8: Similarity of the posterior predictive distributions of BNNs with a single wide hidden layer to the posterior of a GP, considered as the ground truth. Calculations were performed for a 2D test grid X (100×100) spanning the area visible in Fig. 12, which includes regions both with and without training data to account for overconfidence far from the data. The posteriors were obtained using the HMC sampler in all cases, except for [Meronen et al., 2020], where the original code was used (however, results for this model with an HMC-derived posterior are also included below). Lower values indicate better performance. For our method, we present results using weights and priors pretrained for 1D inputs, as well as those trained on functional priors for 1D/2D inputs matched to the task on X.

Weights and activation	1-Wasserstein	2-Wasserstein	Linear-MMD	Mean-L1	Mean-L2	Mean-MSE	Median-L1	Median-L2	Median-MSE	Poly-MMD$\times 10^3$	RBF-MMD
Default with ReLU	39	39	667	0.23	0.26	0.067	0.3	0.33	0.11	4518	0.16
Default with TanH	38	38	595	0.22	0.24	0.058	0.3	0.32	0.11	4181	0.15
Normal with ReLU	31	32	603	0.21	0.25	0.061	0.22	0.26	0.069	3093	0.47
Normal with TanH	25	25	314	0.14	0.18	0.031	0.15	0.18	0.034	1721	0.55
[Meronen et al., 2020]	36	36	777	0.24	0.28	0.078	0.28	0.32	0.1	7311	0.24
Normal + HMC	21	21	72	0.07	0.09	0.007	0.071	0.094	0.009	435	0.24
Default + HMC	42	42	284	0.14	0.16	0.026	0.32	0.34	0.12	1867	0.11
Ours: pretrained 1D	25	25	73	0.06	0.09	0.008	0.076	0.1	0.011	771	0.07
Ours: trained 1D	23	23	6	0.03	0.03	0.001	0.029	0.035	0.001	45	0.03
Ours: trained 2D	23	23	13	0.03	0.03	0.001	0.028	0.035	0.001	74	0.02

D.5 PERIODIC AND CONDITIONAL ACTIVATIONS

BNNs with periodic activations were trained on functions sampled from a GP with a Matérn kernel with $\nu = \frac{5}{2}$ and lengthscales ℓ of values ranging from 0.01 to 10.0. The inputs were randomly sampled from the interval $[-3\ell, 3\ell]$. Each input batch comprised 8 sets X, each containing 512 values x. For each set, 128 functions were sampled. Adam optimizer, with a learning rate of 0.01, was employed over 4000 iterations.

For the conditioning of priors and activations, a MLP hypernetwork was utilized. This hypernetwork consisted of 3 hidden layers with respective widths of 128, 32, and 8, and RBF activations. On top of the hidden layers, separate dense heads were deployed for each output—a 4-dimensional head for producing variances and two 10-dimensional heads for generating $\{\psi\}$ and $\{A\}$. Each head projected the output of the last hidden layer to the requested dimensions.

E EXTENSIVE COMPARISON AGAINST Tran et al. [2022]

As is well known, a single-hidden-layer BNN with infinite width approximates a Gaussian Process (GP). We leverage this fundamental result in the context of optimal transport (minimizing the Wasserstein metric) to induce a GP with a desired kernel behavior onto a BNN.

A related approach, formulated as the dual problem of optimal transport, was explored in [Tran et al., 2022]. However, their method utilizes the complex Normalizing Flow priors as default, and as such, does not incorporate any mathematical property that guarantees the resulting model is a GP. From this perspective, the approach in [Tran et al., 2022] can be seen as a heuristic that yields favorable performance – measured by RMSE and NLL – within the training data range, but without any formal assurance that the model ultimately corresponds to a GP. Their method relies on a multiple-layer BNN with narrow layers, where the number of layers is chosen empirically based on the specific problem. Although employing a Gaussian prior may encourage GP-like behavior, this property does not necessarily hold when using Normalizing Flow priors.

In the following experiments, we empirically demonstrate that this approach generally fails to converge to a GP or satisfy the theoretical requirements due to the use of narrow layers and Normalizing Flow priors.

E.1 USAGE OF WIDER BNNS

First, we investigated the effect of increasing the width of a single-hidden-layer BNN from 50 neurons – following the typical choice in [Tran et al., 2022] – to 1024 neurons. Specifically, we compared two types of priors: Gaussian priors and Normalizing Flow priors. Additionally, we examined four different activation functions commonly used in BNNs and evaluated these settings on a 1D regression task.

Numerical results for RMSE, NLL, and the 1-Wasserstein ($\mathcal{W}_1^1$) and 2-Wasserstein ($\mathcal{W}_2^2$) metrics are presented in Tab.9 and Tab.10 for Gaussian and Normalizing Flow priors, respectively. We observe that increasing the layer width generally improves the Wasserstein metrics while maintaining similar or slightly worse RMSE and NLL when using Gaussian priors. This improvement is likely due to better adherence to theoretical assumptions (*see* Tab. 9). However, even with a wider layer and a fixed activation function, achieving true GP behavior remains elusive.

In contrast, using wider layers with Normalizing Flow priors significantly degrades performance across all considered metrics (*see* Tab.10). We hypothesize that this is due to the lack of theoretical foundations for such priors. Furthermore, we visualize the posterior distributions of narrow and wide BNN layers for two activation functions: *ReLU*, which performs poorly even with a Gaussian prior, and *TanH* (*see* Fig.13). These results indicate that Normalizing Flow priors – despite being uniquely assigned to each neuron – fail to capture the increased capacity of wider networks, regardless of the activation function used. This serves as a clear example of where the heuristic proposed by [Tran et al., 2022] collapses.

Table 9: Comparison of Tran et al. [2022] using a narrow and a wide hidden layer in a BNN with a set Gaussian prior. We observe that the wider hidden layer usually helps in achieving better Wasserstein metrics and similar or slightly worse RMSE and NLL when using Gaussian priors.

$width \rightarrow$	50				1024			
$activation \downarrow metric \rightarrow$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$
TanH	0.36	0.42	7.57	7.83	0.53	0.86	8.04	8.26
RBF	0.25	0.24	7.27	7.61	0.24	0.25	6.96	7.16
ReLU	1.16	4.10	13.67	13.82	1.19	4.35	12.45	12.62
Swish	1.21	4.52	13.18	13.34	1.22	4.59	12.61	12.78

Table 10: Comparison of Tran et al. [2022] using a narrow and a wide hidden layer in a BNN with a set Normalizing Flow prior. We notice that using wider layers with Normalizing Flow priors significantly lowers the result on each of the considered metrics.

$width \rightarrow$	50				1024			
$activation \downarrow metric \rightarrow$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$	*RMSE*	*NLL*	$\mathcal{W}_1^1$	$\mathcal{W}_2^2$
TanH	0.28	0.27	6.22	6.53	0.89	2.32	10.41	10.52
RBF	0.25	0.24	5.48	5.75	0.35	0.41	6.00	6.20
ReLU	0.89	2.34	9.53	9.67	1.10	3.67	12.29	12.44
Swish	1.11	3.64	13.67	13.82	1.18	4.19	13.71	13.86

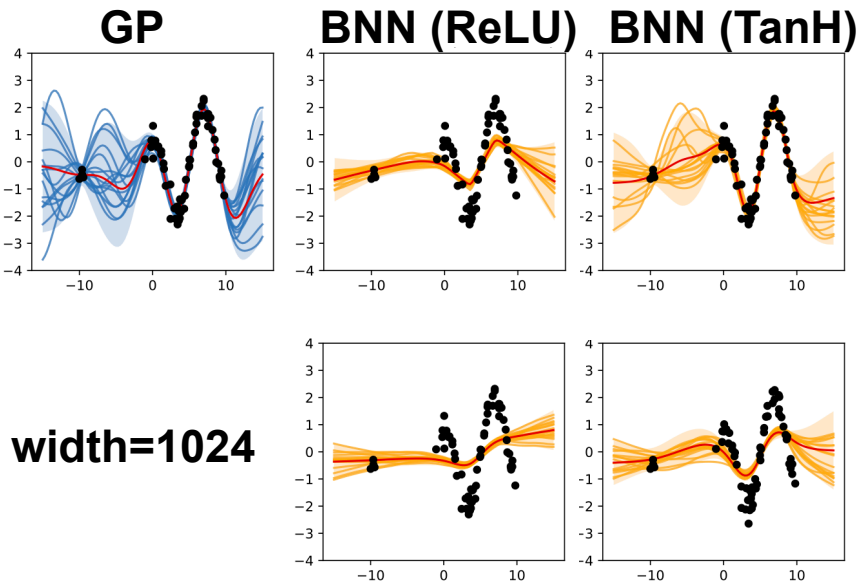


Figure 13: Comparison of Normalizing Flow prior with a one-layer BNN having a varying number of neurons and ReLU (**middle column**) or TanH (**3rd rcolumn**) activation function. We provide a BNN with a narrow hidden layer (50 neurons) as the baseline and compare it against a BNN with a wider hidden layer (1024 neurons). We observe that even wider hidden layer does not provide a better GP approximation due to the lack of theoretical guarantees when using non-Gaussian priors.

E.2 INFLUENCE OF NUMBER OF HIDDEN LAYERS IN BNNS

Next, we examine whether deeper BNNs within the [Tran et al., 2022] framework lead to improved results.

Specifically, we compare BNNs with multiple hidden layers (*i.e.,* $\{2, 3, 4, 5\}$) against single-hidden-layer baselines, evaluating four distinct activation functions – ReLU, TanH, Swish, and RBF – on RMSE and NLL, as shown in Fig. 14.

Additionally, in Fig. 15, we visualize the posterior distributions for two activation functions, ReLU and TanH, under the Normalizing Flow prior to provide further qualitative insights.

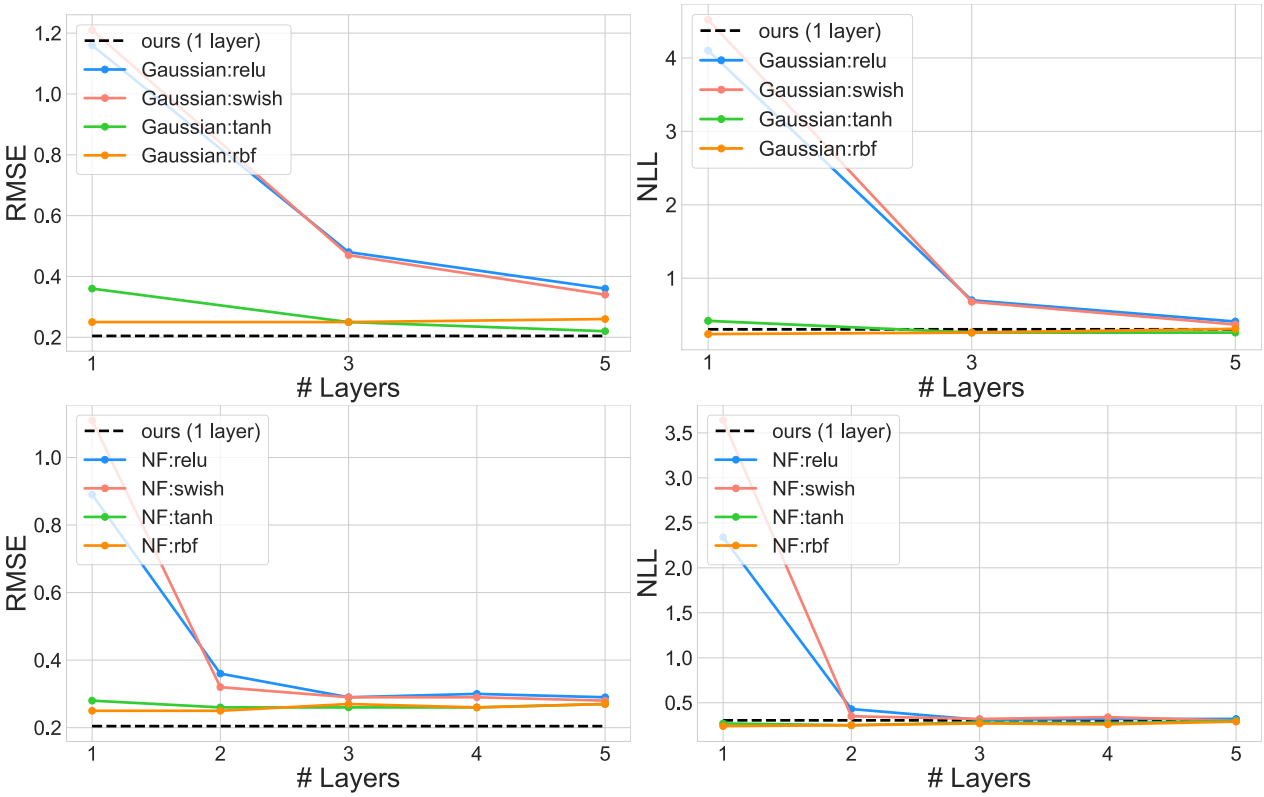


Figure 14: Comparison of different priors, activation functions and BNNs depth in Tran et al. [2022] method in terms of RMSE (**left column**) and NLL **right column** on test data. We observe that this method with Gaussian prior (**top row**) usually get better results with a larger number of hidden layers. Contrary, using Normalizing Flow prior get better results only for some (*worse*) activations. The largest increase of quality might be observed for the worst activation functions. We find that our approach (**dashed black line**) has better, or similar results in terms of both NLL and RMSE.

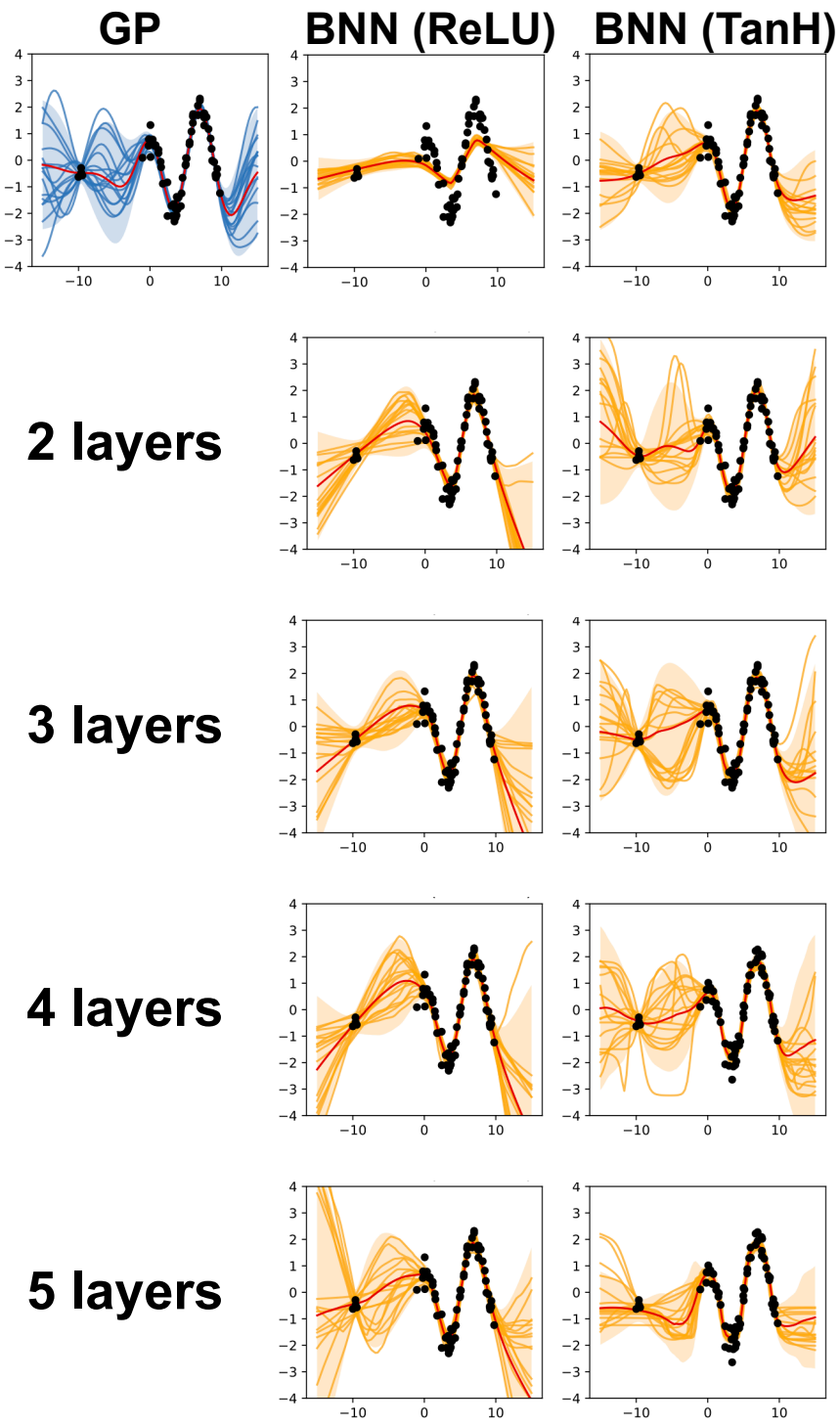


Figure 15: Comparison of Normalizing Flow prior with a BNN having a varying number of hidden layers and ReLU (**middle column**) or TanH (**3rd column**) activation function. We select a single hidden layer (50 neurons) BNN as a baseline (**top row**), which we compare against a multiple hidden layers BNNs ($2, 3, 4,$ or 5 layers). We notice that using a larger number of hidden layers destroys the posteriors for a BNN with TanH activation function, but for some limit helps when using poor ReLU activation.

E.3 OUT OF DISTRIBUTION EVALUATION

Finally, we evaluate the [Tran et al., 2022] method with different priors and varying numbers of hidden layers on an out-of-distribution (OOD) task. Since the true data distribution in OOD regions is unknown, we rely on optimal transport metrics—1-Wasserstein ($\mathcal{W}_1^1$) and 2-Wasserstein ($\mathcal{W}_2^2$). Additionally, we compare all these configurations against our single-hidden-layer method.

To ensure a comprehensive comparison across various OOD scenarios, we define multiple distinct regions of x, where different methods may exhibit different behaviors. These regions are illustrated in Fig. 16.

The results, presented in Fig. 17, 18, 19, and 20, lead to several key observations regarding the performance on OOD tasks:

- **Gaussian prior:** A greater number of hidden layers generally improves performance when using appropriate activation functions (*TanH, RBF*) but degrades it for others (*ReLU, Swish*).
- **Normalizing Flow prior:** In many cases, the best results are achieved with single- or two-hidden-layer BNNs, while deeper architectures significantly degrade performance.
- **Sensitivity to depth:** The Normalizing Flow prior is highly sensitive to the number of hidden layers, providing clear evidence of the heuristic's instability.
- **Our method:** Due to its grounding in theoretical properties, our approach consistently outperforms all configurations of the [Tran et al., 2022] method.

These findings highlight the limitations of heuristic-based approaches and reinforce the importance of theoretical guarantees in achieving robust generalization.

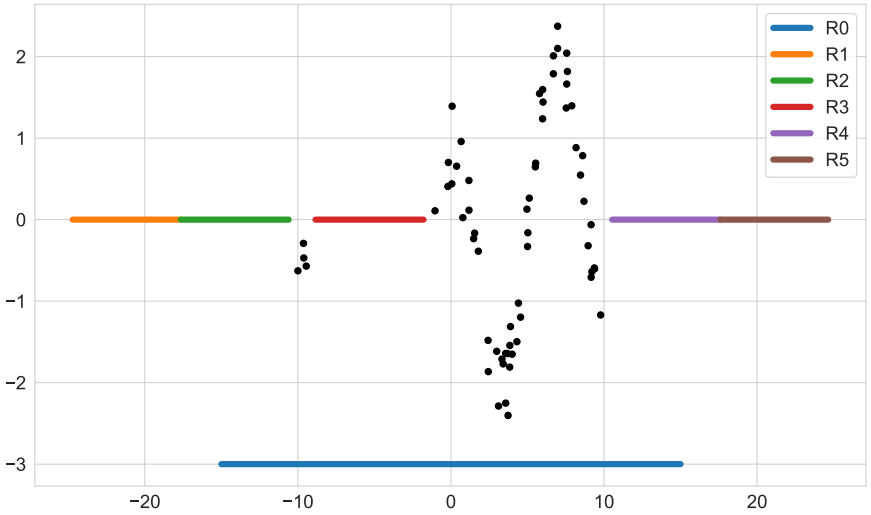


Figure 16: We compare our method with Tran et al. [2022] on the out of distribution data. For this aim, we provide a wide experimental setting comparing both methods on a few different ranges of data (x). We present these ranges as different colour lines in this plot. Firstly, we consider the original test range (**R0**), then we focus on the data between the given blobs of points (**R3**). Finally, we consider the out of distribution ranges – ones being closer to the train data (**R2**, **R4**, and **R2** $\bigcup$ **R4**) or further (**R1**, **R5**, and **R1** $\bigcup$ **R5**). The densities of evaluation points meets the density of the original data on each range despite **R0**.

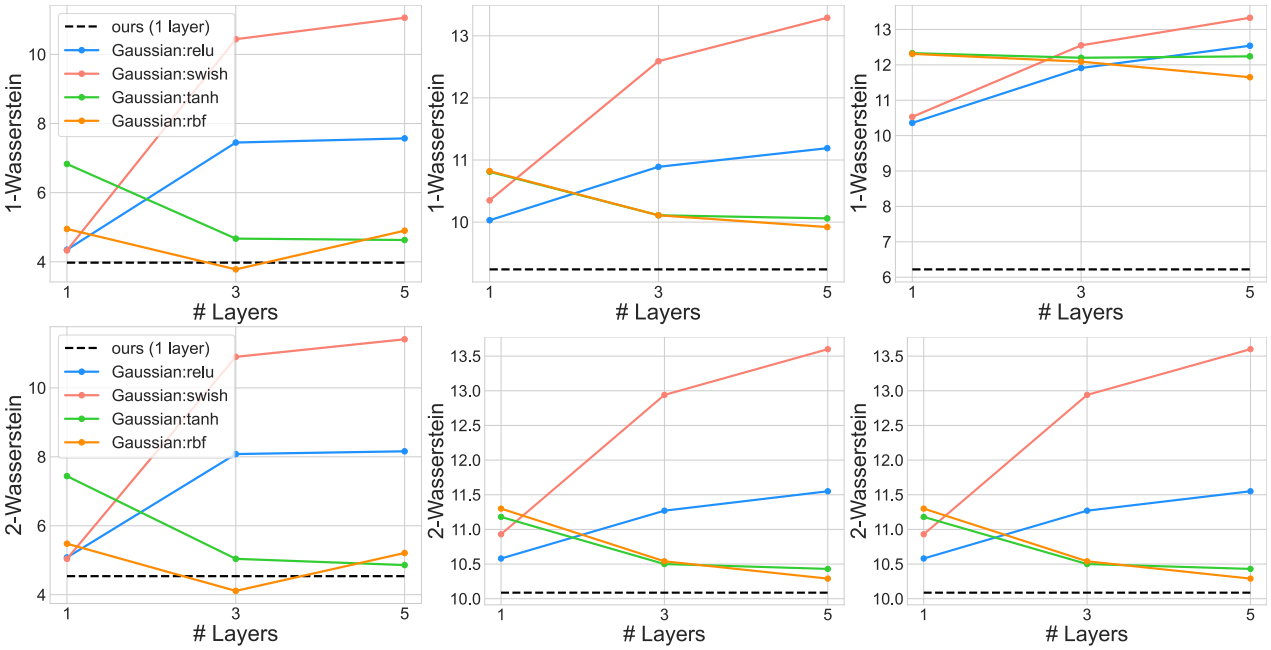


Figure 17: Comparison of Tran et al. [2022] method with a Gaussian prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R3**, **R4** $\bigcup$ **R2**, and **R1** $\bigcup$ **R5** from left to right side. The appropriate regions might be found in Fig. 16.

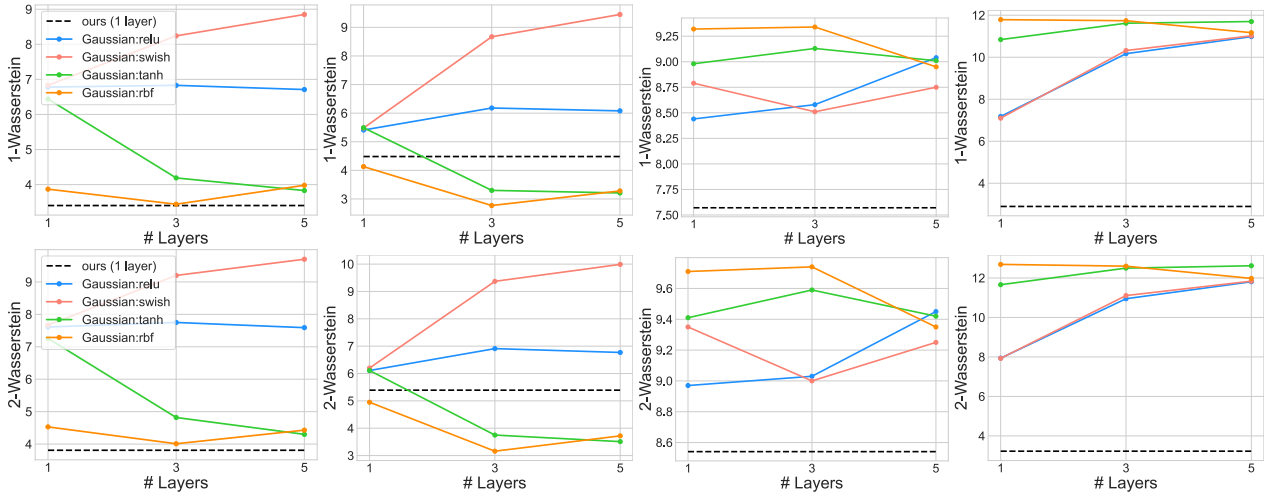


Figure 18: Comparison of Tran et al. [2022] method with a Gaussian prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R1**, **R2**, **R4**, and **R5** from left to right side. The appropriate regions might be found in Fig. 16.

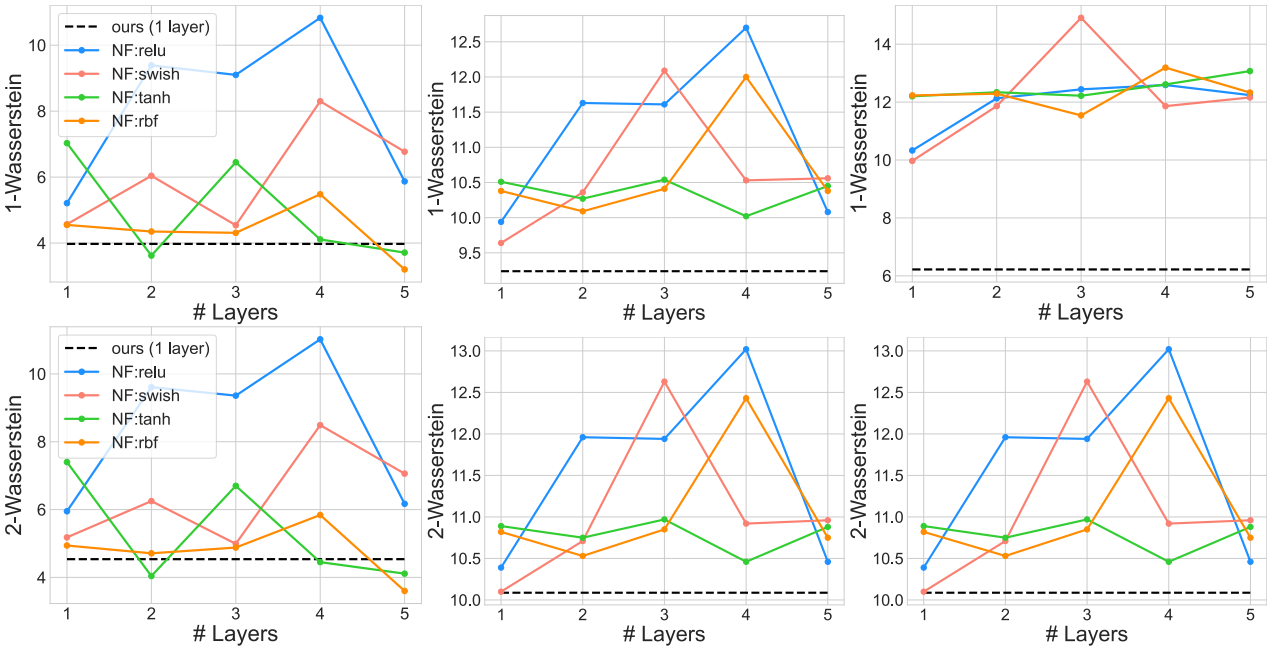


Figure 19: Comparison of Tran et al. [2022] method with a Normalizing Flow prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R3**, **R4** $\bigcup$ **R2**, and **R1** $\bigcup$ **R5** from left to right side. The appropriate regions might be found in Fig. 16.

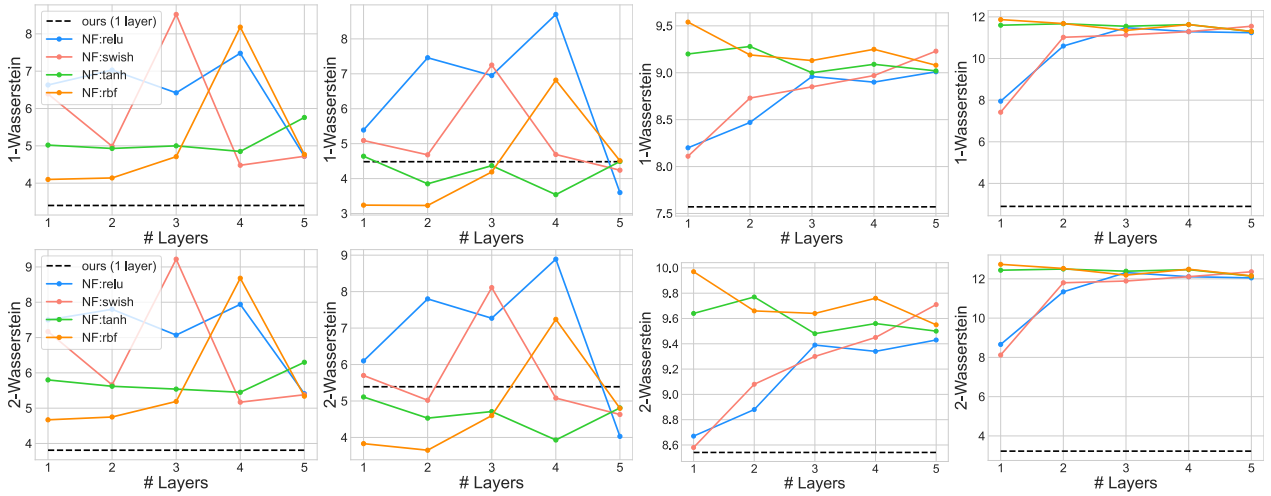


Figure 20: Comparison of Tran et al. [2022] method with a Normalizing Flow prior and different number of hidden layers. In this plot, we present $\mathcal{W}_1^1$ (**top row**) and $\mathcal{W}_2^2$ (**bottom row**) for the regions **R1**, **R2**, **R4**, and **R5** from left to right side. The appropriate regions might be found in Fig. 16.