SORBONNE UNIVERSITÉ

Advancing Bayesian Deep Learning: Sensible Priors and Accelerated Inference

Ba-Hien Tran

*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in the*

Doctoral School N. 130: Computer Science, Telecommunications and Electronics of
Paris of the Sorbonne University

Committee in charge:

Maurizio Filippone	EURECOM	Advisor
Chris Oates	University of Newcastle	Reviewer
Mark van der Wilk	University of Oxford	Reviewer
Marco Lorenzi	INRIA	Examiner
Serena Villata	CNRS	Examiner
Pietro Michiardi	EURECOM	Examiner (Jury President)

Dedicated to my family . . .

SORBONNE UNIVERSITY

Abstract

Doctoral School N. 130: Computer Science, Telecommunications and
Electronics of Paris

Doctor of Philosophy

Advancing Bayesian Deep Learning: Sensible Priors and Accelerated Inference

by Ba-Hien Tran

Over the past decade, deep learning has witnessed remarkable success in a wide range
of applications, revolutionizing various fields with its unprecedented performance.
However, a fundamental limitation of deep learning models lies in their inability to
accurately quantify prediction uncertainty, posing challenges for applications that
demand robust risk assessment. Fortunately, Bayesian deep learning provides a
promising solution by adopting a Bayesian formulation for neural networks. Despite
significant progress in recent years, there remain several challenges that hinder the
widespread adoption and applicability of Bayesian deep learning. In this thesis, we
address some of these challenges by proposing solutions to choose sensible priors
and accelerate inference for Bayesian deep learning models. The first contribution
of the thesis is a study of the pathologies associated with poor choices of priors for
Bayesian neural networks for supervised learning tasks and a proposal to tackle
this problem in a practical and effective way. Specifically, our approach involves
reasoning in terms of functional priors, which are more easily elicited, and adjust-
ing the priors of neural network parameters to align with these functional priors.
The second contribution is a novel framework for conducting model selection for
Bayesian autoencoders for unsupervised tasks, such as representation learning and
generative modeling. To this end, we reason about the marginal likelihood of these
models in terms of functional priors and propose a fully sample-based approach
for its optimization. The third contribution is a novel fully Bayesian autoencoder
model that treats both local latent variables and the global decoder in a Bayesian
fashion. We propose an efficient amortized MCMC scheme for this model and impose
sparse Gaussian process priors over the latent space to capture correlations between
latent encodings. The last contribution is a simple yet effective approach to improve
likelihood-based generative models through data mollification. This accelerates infer-
ence for these models by allowing accurate density-esimation in low-density regions
while addressing manifold overfitting.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Maurizio Filippone, who deserves my utmost thanks. I am truly fortunate to have had him as my advisor during this journey. He has been supportive, giving me the best opportunities and the freedom to pursue my research interests while being available whenever I needed help. I am grateful to him for his guidance and encouragement throughout my PhD. Furthermore, I express my appreciation to the AXA Research Fund through the Chair of Computational Statistics awarded to Maurizio, making these past three years of research possible.

This thesis would not have been possible without my collaborators, from whom I have learned so much. It is a pleasure, in particular, to thank Dimitrios Milios and Simone Rossi for the numerous inspiring discussions and fruitful collaborations. I extend warm appreciation to Prof. Stephan Mandt and Prof. Babak Shahbaba for hosting me during my research visit at UC Irvine, and to Edwin V. Bonilla for his rigorous questions and insightful comments. I also would like to thank other co-authors, Prof. Pietro Michiardi and Giulio Franzese, for their contributions to our joint works.

I am also grateful to my colleagues and friends at EURECOM, especially to the machine learning group of the data science department, for the stimulating discussions and the fun time we had together.

I extend my sincere thanks to Prof. Chris Oates and Prof. Mark van der Wilk for accepting to review this thesis, and thanks to Prof. Pietro Michiardi, Serena Villata, and Marco Lorenzi for being in the examination committee.

Finally, there is nothing I could do to sufficiently express my gratitude to my parents and my sister. Their love and support have made all of this possible.

Contents

List of Figures

List of Tables

List of Abbreviations

NN	**Neural Network**
CNN	**Convolutional Neural Network**
DNN	**Deep Neural Network**
BNN	**Bayesian Neural Network**
MLP	**Mutilayer Perceptron**
AE	**Autoencoder**
VAE	**Variational Autoencoder**
BAE	**Bayesian Autoencoder**
GP	**Gaussian Process**
DGP	**Deep Gaussian Process**
SGP-BAE	**Sparse Gaussian Process Bayesian Autoencoder**
DSGP-BAE	**Deep Sparse Gaussian Process Bayesian Autoencoder**
DPMM	**Dirichlet Process Mixture Model**
NF	**Normalizing Flow**
MLE	**Maximum Likelihood Estimation**
MAP	**Maximum A Posterior**
VI	**Variational Inference**
MCMC	**Markov Chain Monte Carlo**
HMC	**Hamiltonian Monte Carlo**
SGHMC	**Stochastic Gradient Hamiltonian Monte Carlo**
NLL	**Negative Log Likelihood**
RMSE	**Root Mean Squared Error**
FID	**Fréchet Inception Distance**
ELBO	**Evidence Lower Bound**
KL	**Kullback-Leibler Divergence**
MMD	**Maximum Mean Discrepancy**
RBF	**Radial Basis Function**
ARD	**Automatic Relevance Determination**
RKHS	**Reproducing Kernel Hilbert Space**
OT	**Optimal Transport**
WD	**Wasserstein Distance**
DSWD	**Distributional Siced Wasserstein Distance**
GM	**Generative Model**
DM	**Diffusion Model**
GAN	**Generative Adversarial Network**

List of Symbols

a	Scalar
$\boldsymbol{a}$	Vector
$\boldsymbol{A}$	Matrix
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\mathcal{D}$	Dataset
$\boldsymbol{X}$	Dataset inputs (matrix with N rows, one for each data point)
$\boldsymbol{Y}, \boldsymbol{y}$	Dataset outputs (matrix/vector with N rows, one for each data point)
$\boldsymbol{x}_i$	The i-th input data point (generally vector)
$\boldsymbol{y}_i, y_i$	The i-th output data point (vector, scalar)
$\boldsymbol{\hat{y}}_i, \hat{y}_i$	The i-th prediction of a model (vector, scalar)
p_{data}	The data generating distribution
$\hat{p}_{\text{data}}$	The empirical distribution defined by the training set
$f : \mathcal{X} \rightarrow \mathcal{Y}$	The function f with domain $\mathcal{X}$ and range $\mathcal{Y}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x})$	The evaluation of function $f(\cdot)$ at a single location $\boldsymbol{x}$
$f(\boldsymbol{X})$	The evaluation of function $f(\cdot)$ at each location in $\boldsymbol{X}$
$\boldsymbol{f}$	Shorthand for $f(\boldsymbol{X})$ if there is no ambiguity
$\mathcal{N}$	Gaussian distribution
$\mathcal{MN}$	Matrix Gaussian distribution
$\mathcal{GP}$	Gaussian process
$\mathbb{E}_p[x]$	Expectation of x under distribution p
$\text{KL}\left[\cdot \parallel \cdot\right]$	Kullback-Leibler divergence
$\mathbb{R}$	Real numbers
$\mathcal{O}(\cdot)$	Big-O

Chapter 1

Introduction

1.1   The Undeniable Success of Deep Learning

Deep learning (LeCun et al., 2015) has emerged as a transformative and powerful approach to solving complex problems across various domains. At its core, deep learning relies on artificial neural networks. These networks consist of interconnected layers of neurons, where each neuron processes and passes information to subsequent layers. Unlike classical machine learning algorithms, deep learning models can automatically learn hierarchical representations from raw data, obviating the need for manual feature engineering.

Deep learning has witnessed remarkable success in a wide range of applications, revolutionizing various fields with its unprecedented performance. For example, large language models like OpenAI's GPT (Brown et al., 2020) have demonstrated the ability to generate coherent and contextually relevant text, with applications in chatbots, virtual assistants, and content generation. Another notable area where deep learning has excelled is computer vision. Deep learning has demonstrated exceptional accuracy in tasks such as image classification (Dosovitskiy et al., 2021), object detection (Zhang et al., 2023), image segmentation (Kirillov et al., 2023) and image generation (Dhariwal and Nichol, 2021). These advancements have paved the way for applications like autonomous driving (Grigorescu et al., 2020), medical diagnostics (Litjens et al., 2017), and various other domains.

The success of deep learning can be attributed to three key factors (Goodfellow et al., 2016): (1) Large-scale models with a huge number of parameters, despite their lack of identifiability and interpretability; (2) Abundance of data; (3) High-performance computing, particularly GPUs. Consequently, the prevailing paradigm in deep learning can be summarized as follows: gather a vast dataset, define a cost function with some forms of regularization, devise a complex neural network architecture that enables end-to-end gradient propagation, and employ some variants of stochastic gradient descent to minimize the cost function. As a result, deep learning has become the go-to approach for practitioners seeking simple yet powerful solutions to solve machine learning problems.

1.2   But are Deep Learning Models Really That Good?

Deep Learning, despite its remarkable performance, has notable limitations that warrant consideration including: (1) data-hungriness (Marcus, 2018), (2) miscalibrated predictions (Guo et al., 2017; Minderer et al., 2021); (3) non-robustness to out-of-distribution data (Lee et al., 2018; Hein et al., 2019; Ovadia et al., 2019); (4) poor interpretability (Koh and Liang, 2017; Selvaraju et al., 2017). These limitations pose significant concerns, especially in critical domains like medical diagnostics and autonomous driving, where wrong predictions can have severe consequences.

On the one hand, deep learning models are well-known to suffer from a data-hungry nature, posing a significant challenge in practical applications. This data-hungry nature stems from the high capacity and complexity of deep neural networks, which requires substantial amounts of diverse and representative data to accurately estimate their large number of parameters. For example, modern language models (Brown et al., 2020) are trained on more than 300 billion tokens and image recognition models on 400 million images (Radford et al., 2021). Insufficient data can lead to poor generalization, overfitting, and limited performance. This presents a particular concern in domains where data acquisition is expensive, time-consuming, or restricted, such as in medical research or rare event prediction.

On the other hand, although deep neural network (NN) exhibit impressive performance in terms of accuracy, their prediction uncertainty is often miscalibrated and over-confident (Ovadia et al., 2019). Indeed, in the context of classification tasks, the common practice of interpreting softmax outputs as per-class probabilities lacks a solid statistical foundation (Guo et al., 2017). In addressing this issue, it is important to quantify two distinct types of uncertainty: *aleatoric* and *epistemic uncertainty* (Der Kiureghian and Ditlevsen, 2009; Gal, 2016). Aleatoric uncertainty encapsulates the inherent noise present in the data which remains unaffected by observing additional data. On the other hand, epistemic uncertainty arises from uncertainty in the model parameters, specifically the weights of NNs. This uncertainty reflects our lack of knowledge about which model generated the observed data. While aleatoric uncertainty persists even with an infinite number of samples, epistemic uncertainty can be explained away with sufficient data. Unfortunenately, deep learning models are usually not equipped with a principled mechanism to quantify these uncertainties (Guo et al., 2017).

Moreover, deep learning models typically assume that training and testing data are independent and identically distributed (i.i.d). However, in real-world applications, this assumption is often violated. Once a model is deployed, the distribution of observed data can undergo significant shifts, deviating from the original training data distribution. This is particularly evident in online services where data distribution may vary based on factors like time of day, seasonality, or popular trends. Ensuring robustness under distributional shift and handling out-of-distribution (OOD) inputs

is crucial for the safe deployment of machine learning systems (Amodei et al., 2016). In such contexts, having well-calibrated predictive uncertainty becomes essential as it allows for accurate risk assessment, enables practitioners to gauge the potential degradation in accuracy, and allows the system to refrain from making decisions when confidence is low. Unfortunenately, Hein et al. (2019) demonstrate that rectified linear unit (ReLU) deep NN are always overconfident on out-of-distribution examples. Ovadia et al. (2019) show extensively that deep learning models are not robust to corrupted data. Moreover, Moosavi-Dezfooli et al. (2016) demonstrate that deep learning models are susceptible to adversarial attacks, where small perturbations to the input can cause the model to make wrong predictions with high confidence.

Last but not least, the lack of interpretability in deep learning models poses a significant challenge in their widespread adoption and deployment. This arises from the complex, non-linear nature of deep neural networks and the huge number of parameters involved. As a result, it becomes challenging to understand how and why these models arrive at certain predictions or decisions. In domains where transparency, accountability, and trust are crucial, such as healthcare, finance, and autonomous systems, the black-box nature of deep learning can hinder its practical application. To address this issue, some explicitly interpretable models have been designed (Montavon et al., 2018) aiming to provide explanations for their predictions. Additionally, various techniques have been developed to shed light on the inner workings of neural network predictions. These include gradient-based methods, such as generating "heatmaps" (Selvaraju et al., 2017) that highlight the most influential features, as well as influence-function-based approaches (Koh and Liang, 2017) that quantify the impact of individual training examples on the model's predictions.

1.3 The Promises of Bayesian Deep Learning

The Bayesian treament promises to address limitations of deep learning, i.e. improving robustness, interpretability, and uncertainty quantification. In the past 30 years, numerous compelling arguments have emerged in support of adopting a Bayesian inference to deep learning (MacKay, 1995; Graves, 2011; Blundell et al., 2015; Gal, 2016; Izmailov et al., 2021b). Differently from conventional neural networks, Bayesian neural networks treat the model parameters as random variables and places a prior distribution over them. Subsequently, the posterior distribution over the model parameters is inferred using the Bayes rule to reflect the updated knowledge, balancing prior knowledge with observed data.

By incorporating Bayesian principles into the learning process, Bayesian neural network provides a principled framework to quantify uncertainty for predictions, which can aid decision-making and improve the reliability of systems. Rather than bet everything on one hypothesis — with a single setting of optimized parameters of

neural networks — Bayesian neural networks seek to explore all feasible parameter settings, weighting them by the posterior distribution. This process is known as *Bayesian model averaging* (BMA) (Wilson and Izmailov, 2020), and naturally represents epistemic uncertainty in the model parameters. In principle, when we move far away from the training data, there are many more sets of parameters that are consistent with the observed data, leading to a corresponding growth in our epistemic uncertainty. This could make Bayesian deep learning more robust to out-of-distribution data (Izmailov et al., 2021a; Trinh et al., 2022) and adversarial attacks (Carbone et al., 2020). As a result, Bayesian deep learning has the potential to empower informed decision-making and enable appropriate actions, particularly in critical domains such as healthcare, autonomous systems, and finance (Gal, 2016).

Moreover, Bayesian deep learning offers an elegant mechanism to handle data scarcity and interpretability, by leveraging prior knowledge. Defining a prior distribution is a fundamental aspect of the Bayesian paradigm, allowing for the analysis of task similarity, modeling task noise, and incorporating domain-specific knowledge (see Fortuin, 2022, for a comprehensive review of priors for Bayesian deep learning). By encoding prior knowledge about the task at hand, informative priors can reduce the reliance on large amounts of data, leading to more data-efficient performance (Shwartz-Ziv et al., 2022). Moreover, the Bayesian paradigm offers a function-space view of predictors, which can enhance the interpretability of neural network architectures compared to the weight-space view (see e.g. Khan et al., 2019; Sun et al., 2019). Additionally, the uncertainty quantification provided by the Bayesian framework facilitates the interpretability of predictions through counterfactual explanations. These explanations involve making small changes to an input to decrease the uncertainty assigned to it by the model, thereby enhancing the transparency and interpretability of deep learning models (Antoran et al., 2021).

Lastly, the Bayesian paradigm provides a principled framework for model selection through the marginal likelihood, also known as Bayesian evidence (Immer et al., 2021b; Lotfi et al., 2022). This quantity represents the probability that the data is generated from the model, providing a distinctive approach to the fundamental question of model selection. It automatically incorporates *Occam's razor* principle, favoring simpler explanations of the data that are consistent with the observations. Bayesian model selection for machine learning and neural networks was developed and popularized by Mackay (1992). This mechanism can be used to select the best model to predict which architectures will generalize best, and for automatically setting hyperparameters (Immer et al., 2021b), or impose inductive biases such as learning invariances (van der Wilk et al., 2018; Immer et al., 2022). Notably, this model selection process is based solely on the training data, eliminating the need for a separate validation set that may not be readily available in many real-world applications.

1.4   Challenges for Bayesian Deep Learning

Bayesian deep learning, despite its promises and advantages, also faces several challenges that need to be addressed for its widespread adoption. First of all, Bayesian inference while offering an elegant and principled framework, cannot be solved exactly for these models. The Bayesian quanties of interest, such as the posterior distribution and marginal likelihood, are analytically intractable to compute for most neural network architectures. As a result, approximate inference methods such as Laplace approximation (Daxberger et al., 2021), variational inference (Graves, 2011; Blundell et al., 2015) or Markov chain Monte Carlo (MCMC) (Chen et al., 2014; Zhang et al., 2020) are required. These approximations could decrease the quality of the prediction and uncertainty estimates, and are often computationally expensive (Izmailov et al., 2021b). For example, popular variational inference methods require considerable changes to the training procedure while introducing more parameters to be optimized compared to conventional training. Meanwhile, MCMC methods typically require more gradient evaluations and need to store samples from the posterior distribution, which can be prohibitively expensive for large-scale deep learning models.

On the other hand, choosing sensible priors for Bayesian deep learning is a very challenging task. The first and also one of the most important and difficult steps of the Bayesian workflow is to choose a sensible prior over the parameters to be inferred (Mikkola et al., 2023). However, choosing such a prior for modern neural networks is extremely difficult as these models are characterized by a huge number of parameters, and the choice of these priors has an uncontrolled effect on the induced functional prior, which is the distribution of the functions obtained by sampling the parameters from their prior distribution. As a consequence, most works in Bayesian deep learning utilize a (seemingly) uninformative prior for convenience such as the standard Gaussian prior, $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (see, e.g., Fortuin, 2022, and references therein), which results in serious pathologies. We argue that using such bad priors is a hugely limiting aspect of Bayesian deep learning.

1.5   Outline and Contributions of the Thesis

In this thesis, I proposed solutions to address the aforementioned challenges for Bayesian deep learning. The rest of the thesis is organized as follows:

- Chapter 2 provides a brief introduction to Bayesian machine learning, including Bayesian neural networks and Gaussian processes. The main aims are to describe fundamental Bayesian methodologies, and to provide a background for the rest of the thesis.

- Chapter 3 covers the first contribution of this thesis, where we present the pathologies of choosing priors for Bayesian neural networks and propose a solution to tackle this problem in a practical and effective way. Specifically, our approach involves reasoning in terms of functional priors, which are more easily elicited, and adjusting the priors of neural network parameters to align with these functional priors. To achieve this, we leverage shallow Gaussian processes as an elegant framework for defining prior distributions over functions and propose a novel and robust framework to match their prior with the functional prior of Bayesian neural networks by minimizing their Wasserstein distance (Villani, 2008). Through extensive experimentation in supervised-learning tasks such as classification, regression and active learning, we provide compelling empirical evidence that the integration of these priors with scalable Markov chain Monte Carlo sampling consistently leads to significant performance improvements.

- Chapter 4 shows an extension of the work presented in Chapter 3 to unsupervised learning settings with Bayesian autoencoders, where the parameters of both the encoder and decoder are treated in a Bayesian manner. We introduce a novel framework for conducting model selection for Bayesian autoencoders through prior optimization. To this end, we reason about the marginal likelihood of Bayesian autoencoders in terms of functional priors and propose a fully sample-based approach for its optimization utilizing the distributional sliced Wasserstein distance (Nguyen et al., 2021). The effectiveness of our approach is demonstrated through extensive experiments on various unsupervised learning tasks, with a particular focus on the challenging domain of representation learning and generative modeling for high-dimensional data in scenarios characterized by limited data availability.

- Chapter 5 presents an extension of the work presented in Chapter 4, where we introduce a fully Bayesian autoencoder model that treats both local latent variables and global decoder parameters in a Bayesian fashion. To achieve this, we propose an amortized MCMC approach to by utilizing an implicit stochastic network as an encoder to learn how to sampling from the posterior over local latent variables. The encoder in our model functions analogously to that employed in variational autoencoders (Kingma and Welling, 2014). However, we do not assume any specific form of the posterior distribution of the latent variables. This approach enables flexible priors and posterior approximations while maintaining low computational costs for inference. Furthermore, we propose to incoporate sparse Gaussian process priors over the latent space to capture correlations between latent encodings. We show the strong performance of our approach through a series of experiments focusing on dynamic representation and generative modeling.

- Chapter 6 delves into our final contribution in this thesis, which is inpsired by the work presented in Chapter 3 and the recent success of diffusion models (Ho

et al., 2020; Song et al., 2021). A key feature of score-based diffusion models is to accurately estimate density in low-density regions and address manifold overfitting by employing data mollification through the addition of Gaussian noise. We speculate that this effect bears resemblance to Bayesian autoencoders, wherein the Bayesian treatment empowers these models to effectively behave in the tail regions of the data distribution, resulting in robust performance. We establish a connection between data mollification procedure and Gaussian homotopy, a well-known technique for optimization improvement. Notably, data mollification incurs no additional computational overhead and can be effortlessly implemented with a single line of code in any training loop. We demonstrate that this simple technique can facilitate inference and consistently improve the sample quality of likelihood-based generative models, including variational autoencoders, normalizing flows.

- Chapter 7 gives remarks of the works presented in this thesis and concludes with an outlook on future research directions.

Publications

The works in this thesis were done in collaboration with colleagues and collaborators, and have been mainly peer-reviewed by program committees in A* conferences and journals.

Chapter 3 is based on the following journal publication:

- **Ba-Hien Tran**, Simone Rossi, Dimitrios Milios, Maurizio Filippone. "All You Need is a Good Functional Prior for Bayesian Deep Learning". In *Journal of Machine Learning Research* (2022).

  which is an extension of the following workshop paper:

- **Ba-Hien Tran**, Dimitrios Milios, Simone Rossi, Maurizio Filippone. "Functional Priors for Bayesian Neural Networks through Wasserstein Distance Minimization to Gaussian Processes". In *Symposium on Approximate Bayesian Inference* (2021).

Chapter 4 is based on the following conference publication:

- **Ba-Hien Tran**, Simone Rossi, Dimitrios Milios, Edwin V. Bonilla, Pietro Michiard, Maurizio Filippone. "Model Selection for Bayesian Autoencoders". In *Advances of Neural Information Processing Systems* (2021)

Chapter 5 is based on the following conference publication:

- **Ba-Hien Tran**, Babak Shahbaba, Stephan Mandt, Maurizio Filippone. "Fully Bayesian Autoencoders with Latent Sparse Gaussian Processes". In *International Conference on Machine Learning* (2023).

Chapter 6 is based on the following conference paper:

- **Ba-Hien Tran**, Giulio Franzese, Pietro Michiardi, Maurizio Filippone. "One-Line-of-Code Data Mollification Improves Optimization of Likelihood-based Generative Models". In *Advances on Neural Information Processing Systems* (2023).

  which is an extension of the following workshop paper:

- **Ba-Hien Tran**, Giulio Franzese, Pietro Michiardi, Maurizio Filippone. "Improving Training of Likelihood-based Generative Models with Gaussian Homotopy". In *Workshop on Structured Probabilistic Inference & Generative Modeling* during the International Conference on Machine Learning (2023).

Chapter 2

Probabilistic Methods for Machine Learning

In this chapter, we establish the fundamental background that will underpin the entire thesis. We provide concise introductions to key concepts that play a central role, including probabilistic machine learning, Bayesian neural networks, and Gaussian processes. These concepts lay the groundwork for the subsequent chapters of the thesis.

2.1 A Refresher on Probabilistic Machine Learning

Probabilistic modeling is a cornerstone of modern machine learning toolkits. It offers a principled framework for making coherent inferences, learning from observations and handling uncertainty, through the language of probability theory (Ghahramani, 2015). The key idea behind the probabilistic approach to machine learning is that learning can be thought of as inferring plausible models to explain observed data. At its core, a probabilistic model utilizes probability distributions to express the subjective beliefs or uncertainties surrounding unknown variables within the model, which are encapsulated through the *prior*. Additionally, the model establishes the relationship between these unknown variables and the observed data through the *likelihood*. Through probabilistic inference, often referred to as Bayesian inference, the initial beliefs represented by the prior are transformed into a *posterior* distribution of the unknown variables. This posterior distribution encapsulates the updated beliefs about the unknown variables after taking the observed data into account.

As an example, consider a generic parametric model f parameterized by unknown parameters (variables) $w \in \mathbb{R}^d$, and a dataset comprised of N input-output pairs $\mathcal{D} = \{X, y\} \stackrel{\text{def}}{=} \{(x_i, y_i)\}_{i=1}^N$, with $x_i \in \mathbb{R}^{D_{\text{in}}}$ and $y_i \in \mathbb{R}$. By imposing a prior $p(w)$ on the parameters w, we can obtain the posterior distribution by applying Bayes' rule as

follows:

$$\underbrace{p(\boldsymbol{w}\,|\,\mathcal{D})}_{\text{Posterior}} = \frac{\overbrace{p(\mathcal{D}\,|\,\boldsymbol{w})}^{\text{Likelihood}}\overbrace{p(\boldsymbol{w})}^{\text{Prior}}}{\underbrace{p(\mathcal{D})}_{\text{Model Evidence}}}, \tag{2.1}$$

where the normalization constant $p(\mathcal{D}) = \int p(D\,|\,\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}$ is known as the *model evidence* or *marginal likelihood*.

Likelihood.   The likelihood component $p(\mathcal{D}\,|\,\boldsymbol{w}) = \sum_{i=1}^{N} p(y_i\,|\,f(\boldsymbol{x}_i,\boldsymbol{w}))$ represents the probability density function of the model given the parameters, evaluated at the observed data. Here, we assume a full factorization of the likelihood over the data-points. It quantifies the likelihood of the model f with parameters $\boldsymbol{w}$ generating the observed data. Furthermore, this quantity encapsulates our assumptions about the observed data, such as being continuous and noisy, binary, or count data. Hence, different likelihood distributions are employed depending on the task and data type. For regression tasks with i.i.d noisy data, the Gaussian likelihood $\mathcal{N}(y_i\,|\,f(\boldsymbol{x}_i,\boldsymbol{w}),\sigma_\epsilon^2)$ with a noise variance σ_ϵ^2 is used, whereas for binary classification tasks with a mapping function $\lambda : \mathbb{R} \to [0,1]$, the Bernoulli likelihood $\text{Bern}(y_i\,|\,\lambda(f(\boldsymbol{x}_i,\boldsymbol{w})))$ is utilized.

Prior.   The prior distribution $p(\boldsymbol{w})$ embodies our beliefs about the parameters $\boldsymbol{w}$ before observing the data. A natural question concerning the choice of the prior is the amount of information it conveys. This leads to the classification of priors into *objective* and *subjective* categories. The Jeffrey prior (Jeffreys, 1946) is a well-known example of the former type, defined as $p_{\text{JEFF}}(\boldsymbol{w}) \propto \sqrt{F(\boldsymbol{w})}$, with $F(\boldsymbol{w})$ is the Fisher information. It is considered analogous objective or non-informative prior since it remains invariant to parameterization changes, assigning the same probability to a set of models regardless of the chosen parameterization. In contrast to objective priors, subjective priors align with the roots of the Bayesian approach, as they encode one's beliefs through a prior. Eliciting a prior distribution is a delicate matter, particularly with more complex models where parameters may lack interpretation, and higher-dimensional parameter spaces, making the design of a suitable prior challenging. For a comprehensive review on this topic, refer to Mikkola et al. (2023).

Bayesian model selection.   By leveraging the model evidence, the Bayesian framework offers a principled approach to model selection. Understanding this approach becomes clearer when we explicitly condition the entire inference on a specific model hypothesis $\mathcal{M}_i$, which is part of a finite set of models $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^{M}$. Eq. 2.1 can be rewritten as follows:

$$p(\boldsymbol{w}\,|\,\mathcal{D},\mathcal{M}_i) = \frac{p(\mathcal{D}\,|\,\boldsymbol{w},\mathcal{M}_i)p(\boldsymbol{w}\,|\,\mathcal{M}_i)}{p(\mathcal{D}\,|\,\mathcal{M}_i)} \tag{2.2}$$

The model evidence $p(\mathcal{D} \mid \mathcal{M}_i)$ describes the probability that we would generate a dataset $\mathcal{D}$ with a model $\mathcal{M}_i$ if we randomly sample from a prior over the parameters $p(\boldsymbol{w} \mid \mathcal{M}_i)$. This quantity is referred to as the *marginal likelihood* because it can be viewed as a likelihood function over the space of models, in which the parameters $\boldsymbol{w}$ have been marginalized out:

$$p(\mathcal{D} \mid \mathcal{M}_i) = \int p(\mathcal{D} \mid \boldsymbol{w}, \mathcal{M}_i) p(\boldsymbol{w} \mid \mathcal{M}_i) d\boldsymbol{w}. \tag{2.3}$$

We can futher apply Bayes's rule to condition models on the data as follows:

$$p(\mathcal{M} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathcal{M}) p(\mathcal{M})}{p(\mathcal{D})}, \tag{2.4}$$

where $p(\mathcal{M})$ is a prior distribution over models. Estimating the posterior over the models $p(\mathcal{M} \mid \mathcal{D})$ is very challenging since both the model prior and the marginal likelihood are not obvious. Consequently, the prevalent approach to address the problem of model selection involves maximizing the marginal likelihood, thereby selecting the model with the highest marginal likelihood. This method is commonly known as *type-II maximum likelihood*. In essence, this approach can be seen as a maximum-a-posteriori (MAP) estimate of the posterior in Eq. 2.4, where a uniform prior over models, i.e., $p(\mathcal{M}) \propto 1$, is assumed. In addition, we can compare two models $\mathcal{M}_i$ and $\mathcal{M}_j$ by computing the ratio of their marginal likelihoods, denoted as $p(\mathcal{D} \mid \mathcal{M}_i) / p(\mathcal{D} \mid \mathcal{M}_j)$, which is referred to as the *Bayes factor* (Kass and Raftery, 1995).

2.2 Bayesian Neural Networks

In the previous section, we consider the Bayesian paradigm for the generic form of the parametric model f. In this section, our focus shifts towards exploring the utilization of neural networks for modeling f, along with their corresponding Bayesian treatment.

2.2.1 Deep neural networks

We consider a deep neural network (DNN) consisting of L layers, where the output of the l-th layer $f_l(\boldsymbol{x})$ is a function of the previous layer outputs $f_{l-1}(\boldsymbol{x})$, as follows:

$$f_l(\boldsymbol{x}) = \frac{1}{\sqrt{D_{l-1}}} \left(W_l \varphi(f_{l-1}(\boldsymbol{x})) \right) + b_l, \quad l \in \{1, ..., L\}, \tag{2.5}$$

where φ is a nonlinearity, $b_l \in \mathbb{R}^{D_l}$ is a vector containing the bias parameters for layer l, and $W_l \in \mathbb{R}^{D_l \times D_{l-1}}$ is the corresponding matrix of weights. We shall refer to the

union of weight and bias parameters of a layer l as $\boldsymbol{w}_l = \{W_l, b_l\}$, while the entirety of trainable network parameters will be denoted as $\boldsymbol{w} = \{\boldsymbol{w}_l\}_{l=1}^L$.

In order to simplify the presentation, we focus on fully-connected DNNs; the weight and bias parameters of convolutional neural networks (CNNs) are treated in a similar way, unless stated otherwise. The scheme that involves dividing by $\sqrt{D_{l-1}}$ is known as the *NTK parameterization* (Jacot et al., 2018; Lee et al., 2020), and it ensures that the asymptotic variance neither explodes nor vanishes. For fully-connected layers, D_{l-1} is the dimension of the input, while for convolutional layers D_{l-1} is replaced with the filter size multiplied by the number of input channels.

Training.   Given a dataset with N input-target pairs $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\} \stackrel{\text{def}}{=} \{(\mathbf{x}_i, y_i)\}_{i=1}^N \in (\mathcal{X}, \mathcal{Y})^N$. We consider a loss function $\mathcal{L} : \mathcal{F} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $\mathcal{F}$ is a set of predictors $f : \mathcal{X} \rightarrow \mathcal{Y}$. The goal of training a DNN is to find *the optimal parameters $\boldsymbol{w}^\star$* that minimize the expected risk, which is defined as follows:

$$\mathcal{R}^{\mathcal{L}}(f) \stackrel{\text{def}}{=} \mathbb{E}_{p(\boldsymbol{x}, y)}\big[\mathcal{L}(f(\boldsymbol{x}), y)\big]. \tag{2.6}$$

Due to the intractability of the expected risk, we resort to minimizing the empirical risk, i.e. the average loss over the training data:

$$\hat{\mathcal{R}}_{\mathcal{D}}^{\mathcal{L}}(f) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\boldsymbol{x}_i), y_i). \tag{2.7}$$

Optimizing the objective Eq. 2.7 as-is risks overfitting the training data. Therefore, we need to impose regularization which controls the tradeoff between data fitting and model complexity. This leads us to formulate an optimization problem in the following form:

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \hat{\mathcal{R}}_{\mathcal{D}}^{\mathcal{L}}(f) + \lambda\Omega(\boldsymbol{w}), \tag{2.8}$$

where $\Omega(\boldsymbol{w})$ is a regularization term, such as L_2 regularizer, i.e. $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$; and λ is a regularization coefficient that controls the strength of the regularizer. This objective can be optimized via *backpropagation* (Rumelhart et al., 1986) and using a gradient-based optimization algorithm (Goodfellow et al., 2016).

2.2.2   Bayesian treatment for deep neural networks

The Bayesian treatment of neural networks (MacKay, 1992; Neal, 1996) dictates that a prior distribution $p(\boldsymbol{w})$ is placed over the parameters. We will delve deeper into the role of prior distributions for Bayesian neural network in Chapter 3. The learning problem is now formulated as a transformation of a prior belief into a posterior

distribution by means of Bayes' theorem. The posterior over w is defined as follows:

$$p(w \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid w)p(w)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid w)p(w)}{\int p(\mathcal{D} \mid w)p(w)dw}. \tag{2.9}$$

Once the posterior is infered, we can obtan the *predictive posterior* for unseen data $x^\star$ as follows:

$$p(y^\star \mid x^\star, \mathcal{D}) = \int p(y^\star \mid f(x^\star, w))p(w \mid \mathcal{D})dw. \tag{2.10}$$

The above integral performs a so-called *Bayesian model averaging* operation because rather than betting everything on one hypothesis — with a single setting of parameters — we use every plausible setting of parameters, weighted by their posterior probabilities. This is the main distinction of the Bayesian treatment from the conventional approach where parameters are optimized.

We can see that obtaining the predictive posterior requires computing two integrals, one to calculate the normalizing constant in Bayes' theorem in Eq. 2.9 and one to average our predictions over the posterior distribution in Eq. 2.10. Because the likelihood function $p(\mathcal{D} \mid w)$ is highly non-linear in w for neural networks, these integrals are analytically intractable. As a consequence, *approximate inference* methods are needed for Bayesian neural networks.

2.3 Common Approximation Inference Methods for Bayesian Neural Networks

2.3.1 Laplace approximation

Let $w_{\mathrm{MAP}} \stackrel{\text{def}}{=} \arg\max_w \log p(w \mid \mathcal{D}) = \arg\max_w \log p(\mathcal{D} \mid w) + \log p(w)$ be a local maximum of the posterior, known as the *maximum a posteriori* (MAP) estimate, which can be found by a gradient descent algorithm. By applying Taylor's expansion to the log-unnormalized posterior $g(w) \stackrel{\text{def}}{=} \log p(\mathcal{D} \mid w) + \log p(w)$ around w_{MAP} up to second order, we obtain the following approximation:

$$\log g(w) \approx \log g(w_{\mathrm{MAP}}) - \frac{1}{2}(w - w_{\mathrm{MAP}})^\top \Lambda (w - w_{\mathrm{MAP}}), \tag{2.11}$$

where $\Lambda \stackrel{\text{def}}{=} -\nabla^2 \log g(w)|_{w=w_{\mathrm{MAP}}}$ is the Hessian matrix of the log-unnormalized posterior evaluated at the MAP estimate w_{MAP}.

Integrating Eq. 2.11, we can obtain the analytical form of the normalization constant of the posterior distribution as follows:

$$Z \approx \exp(\log g(\boldsymbol{w}_{\text{MAP}})) \int \exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}})^{\top} \boldsymbol{\Lambda}(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}})\right) d\boldsymbol{w}$$

$$= g(\boldsymbol{w}_{\text{MAP}})(2\pi)^{\frac{d}{2}}(det\boldsymbol{\Lambda})^{-\frac{1}{2}}, \tag{2.12}$$

where d is the dimensionality of the parameter $\boldsymbol{w}$.

From Eq. 2.11 and Eq. 2.12, we can obtain the following approximation of the posterior distribution:

$$p(\boldsymbol{w} \,|\, \mathcal{D}) = \frac{1}{Z}g(\boldsymbol{w}) \approx (2\pi)^{-\frac{d}{2}}(det\boldsymbol{\Lambda})^{\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}})^{\top} \boldsymbol{\Lambda}(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}})\right) \tag{2.13}$$

This is a Gaussian distribution with mean $\boldsymbol{w}_{\text{MAP}}$ and covariance matrix $\boldsymbol{\Lambda}^{-1}$, and is called the *Laplace approximation* of $p(\boldsymbol{w} \,|\, \mathcal{D})$. This method stands out as one of the most cost-effective approximate inference techniques for Bayesian neural networks (Daxberger et al., 2021). It can be applied to any MAP pre-trained neural networks. Additionally, the Hessian matrix only needs to be computed once, which has been made feasible by recent advancements in second-order optimization techniques (Dangel et al., 2020). However, as both the MAP estimate $\boldsymbol{w}_{\text{MAP}}$ and the Hessian matrix $\boldsymbol{\Lambda}$ are local quantities, the Laplace approximation is limited to providing a local approximation around a mode of the posterior distribution $p(\boldsymbol{w} \,|\, \mathcal{D})$.

2.3.2 Variational inference

Another approach to obtaining a parametric approximation of the posterior $p(\boldsymbol{w} \,|\, \mathcal{D})$ is via variational inference (VI). This is a classic tool to tackle intractable Bayesian inference (Jordan et al., 1999; Blei et al., 2017). VI casts the inference problem into an optimization problem to compute a tractable approximation of the posterior distribution. Let $\mathcal{Q} = \{q_{\boldsymbol{\phi}} : \boldsymbol{\phi} \in \boldsymbol{\Phi}\}$ be a set of parametric densities, known as *variational family*, which are usually tractable. The common goal of VI is to minimize the Kullback-Leibler (KL) divergence between the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{w})$ and the true posterior $p(\boldsymbol{w} \,|\, \mathcal{D})$. In other words, we seek the parameter $\boldsymbol{\phi}^{\star}$ such that:

$$\boldsymbol{\phi}^{\star} = \underset{\boldsymbol{\phi} \in \boldsymbol{\Phi}}{\arg\min} \, \text{KL}\left[q_{\boldsymbol{\phi}}(\boldsymbol{w}) \,\|\, p(\boldsymbol{w} \,|\, \mathcal{D})\right] = \underset{\boldsymbol{\phi} \in \boldsymbol{\Phi}}{\arg\min} \int q_{\boldsymbol{\phi}}(\boldsymbol{w}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{w})}{p(\boldsymbol{w} \,|\, \mathcal{D})} d\boldsymbol{w}. \tag{2.14}$$

However, the above integral is intractable in general since it implies that we need to compute the normalization constant of the posterior $p(\boldsymbol{w} \,|\, \mathcal{D})$.

To address this concern, it is possible to explore the upper bound of of the KL divergence. To derive this bound, we introduce and subtract the log-marginal likelihood $p(\mathcal{D})$ to the KL divergence. By recognizing that $p(\boldsymbol{w}, \mathcal{D}) = p(\boldsymbol{w} \,|\, \mathcal{D})p(\mathcal{D}) =$

$p(\mathcal{D} \,|\, w)p(w)$, we can proceed with the derivation as follows:

$$
\begin{aligned}
\mathrm{KL}\left[q_{\phi}(w) \,\|\, p(w \,|\, \mathcal{D})\right] &= \mathbb{E}_{q_{\phi}(w)}\left[\log \frac{q_{\phi}(w)}{p(w \,|\, \mathcal{D})}\right] - \log p(\mathcal{D}) + \log p(\mathcal{D}) \\
&= \mathbb{E}_{q_{\phi}(w)}\left[\log \frac{q_{\phi}(w)}{p(w \,|\, \mathcal{D})p(\mathcal{D})}\right] + \log p(\mathcal{D}) \\
&= -\mathbb{E}_{q_{\phi}(w)}\left[\log \frac{p(\mathcal{D} \,|\, w)p(w)}{q_{\phi}(w)}\right] + \log p(\mathcal{D}) \qquad (2.15) \\
&= -\bigg(\underbrace{\mathbb{E}_{q_{\phi}(w)}[\log p(\mathcal{D} \,|\, w)] - \mathrm{KL}\left[q_{\phi}(w) \,\|\, p(w)\right]}_{:=\mathcal{L}_{\mathrm{ELBO}}(\phi)}\bigg) + \log p(\mathcal{D}).
\end{aligned}
$$

Here, $\mathcal{L}_{\mathrm{ELBO}}(\phi)$ is a lower bound of the log-marginal likelihood $p(\mathcal{D})$, commonly referred to as the *evidence lower bound* (ELBO). Maximizing $\mathcal{L}_{\mathrm{ELBO}}$ is equivalent to minimizing the KL divergence between $q_{\phi}(w)$ and $p(w \,|\, \mathcal{D})$. The first term in $\mathcal{L}_{\mathrm{ELBO}}(\phi)$ corresponds to the expected log-likelihood of the data under the approximate posterior $q_{\phi}(w)$, serving as a measure of how well the model fits the data. The second term introduces a regularization effect by penalizing posteriors that significantly deviate from the prior. Unlike $\mathrm{KL}\left[q_{\phi}(w) \,\|\, p(w \,|\, \mathcal{D})\right]$, obtaining an unbiased estimate of $\mathcal{L}_{\mathrm{ELBO}}(\phi)$ is often computationally feasible. In particular, we can decompose $\mathcal{L}_{\mathrm{ELBO}}$ as follows:

$$
\begin{aligned}
\mathcal{L}_{\mathrm{ELBO}}(\phi) &= \mathbb{E}_{q_{\phi}(w)}[\log p(\mathcal{D} \,|\, w)] - \mathrm{KL}\left[q_{\phi}(w) \,\|\, p(w)\right] \qquad (2.16) \\
&= \sum_{i=1}^{N} \mathbb{E}_{q_{\phi}(w)}[\log p(y_i \,|\, f(x_i, w))] - \mathrm{KL}\left[q_{\phi}(w) \,\|\, p(w)\right] \qquad (2.17) \\
&= \sum_{i=1}^{N} \mathbb{E}_{q_{\phi}(w)}[\log p(y_i \,|\, f(x_i, w))] - \mathbb{E}_{q_{\phi}(w)}\left[\log \frac{q_{\phi}(w)}{p(w)}\right]. \qquad (2.18)
\end{aligned}
$$

Estimations of the expectations in Eq. 2.18 can be done through Monte Carlo sampling, wherein samples are drawn from the approximate posterior $q_{\phi}(w)$. Analytical computation of the KL divergence becomes feasible in some cases, for example, where both $q_{\phi}(w)$ and $p(w)$ are Gaussian. Additionally, exploiting the decomposition of the likelihood term over datapoints allows for the straightforward creation of an unbiased estimate of $\mathcal{L}_{\mathrm{ELBO}}$ using mini-batches of data, enabling scalable variational inference for large datasets. To optimize ϕ, gradient-based optimizers can be employed, while unbiased estimates of $\nabla_{\phi}\mathcal{L}_{\mathrm{ELBO}}$ can be obtained using the reparameterization trick (Kingma and Welling, 2014; Blundell et al., 2015).

Variational inference is a versatile approximate inference technique that encompasses a wide range of variational families $\mathcal{Q}$. Among these, the most commonly employed variational family consists of fully-factorized Gaussian distributions over w. This type of variational inference is commonly referred to as *mean-field variational inference* (MFVI) (Graves, 2011; Blundell et al., 2015). Additionally, more flexible families have

been proposed, including multivariate Gaussian distributions over w, families incorporating matrix-variate Gaussians (Louizos and Welling, 2017), normalizing flows (Rezende and Mohamed, 2015), or implicit distributions (Huszár, 2017; Mescheder et al., 2017; Titsias and Ruiz, 2019) whose densities cannot be directly evaluated. Moreover, instead of using the KL in Eq. 2.14, various alternative divergences and distances are proposed in the literature, such as Rényi divergences (Li and Turner, 2016), χ-divergences (Dieng et al., 2017), f-divergences (Wan et al., 2020), and Wasserstein distances (Ambrogioni et al., 2018).

2.3.3   Markov chain Monte Carlo

Markov chain and stationary distribution.   Consider a sequence of random variables denoted as $w^{(1)}, \ldots, w^{(t)}$. Under the assumption of independence between $w^{(i)}$ and $\{w^{(j)}\}_{j=1}^{i-1}$ given $w^{(i-1)}$ for each $i = 1, \ldots, t$, commonly known as the *Markov assumption*, the joint distribution can be expressed as follows:

$$p(w^{(1)}, \ldots, w^{(t)}) = p(w^{(1)}) \prod_{i=2}^{t} p(w^{(i)} \mid w^{(i-1)}). \qquad (2.19)$$

The sequence $\{w^{(i)}\}_{i=1}^{t}$ is referred to as a *Markov chain*, and the probability $p(w^{(i)} \mid w^{(i-1)})$ is known as the *transition probability*. Once $p(w^{(i)} \mid w^{(i-1)})$ is defined, a Markov chain offers a simple approach to generating a sequence of random variables. Specifically, we can sample $w^{(i)}$ from $p(w^{(i)} \mid w^{(i-1)})$, where $w^{(i-1)}$ is the previous sample.

Consider two random variables in a Markov chain, denoted as $w^{(i)}$ and $w^{(i+1)}$, with a *transition probability* of $p(w^{(i+1)} \mid w^{(i)})$. Let us assume that the distribution of w is represented by $q(w)$. In such a case, we refer to q as the *stationary distribution* of the Markov chain if

$$q(w^{(i+1)}) = \int p(w^{(i+1)} \mid w^{(i)}) q(w^{(i)}) dw^{(i)}. \qquad (2.20)$$

In other words, the stationary distribution is the distribution of w that does not change under the transition probability. Intuitively, once a Markov chain reaches its stationary distribution at a time i, any $w^{(j)}$ with $j \geq i$ is drawn from the same distribution q.

Markov chain Monte Carlo.   The fundamental concept behind Markov chain Monte Carlo (MCMC) methods for Bayesian inference involves constructing a Markov chain. This entails defining the transition function $p(w^{(i+1)} \mid w^{(i)})$, where the stationary distribution of the Markov chain, denoted as q, corresponds to the posterior distribution $p(w \mid \mathcal{D})$. Consequently, once the Markov chain reaches its stationary distribution, running the chain is equivalent to sampling from the true posterior $p(w \mid \mathcal{D})$. The rate at which the Markov chain converges to its stationary distribution, commonly

known as the mixing speed, holds practical significance as it determines the sampling costs involved in obtaining posterior samples.

Hamiltonian Monte Carlo.   Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2011) provides a method for proposing samples of w in a Metropolis-Hastings (MH) franework that efficiently explores the posterior space. These proposals are generated from a Hamiltonian system based on introducing a set of auxiliary momentum variables r. That is, to sample from $p(\theta \mid \mathcal{D})$, HMC considers generating samples from a joint distribution of (w, r), defined as follows:

$$p(w, r) \propto \exp \left(-U(w) - \frac{1}{2} r^\top M^{-1} r \right), \tag{2.21}$$

where $U(w)$ is a *potential energy* function defined by the log-unnormalized posterior, i.e. $U(w) = -\log p(\mathcal{D} \mid w) - \log p(w)$; M is an arbitrary *mass matrix*, and together with r, defines a *kinetic energy* term.

Samples are then generated based on the Hamiltonian dynamics:

$$dw = M^{-1} r dt, \tag{2.22}$$
$$dr = -\nabla U(w) dt, \tag{2.23}$$

where, the mass matrix M plays the role of a preconditioner.

Simulating the above continuous system can pose computational challenges. To overcome this, a common approach involves discretization techniques such as the *leapfrog* scheme. Additionally, a correction step is typically applied using the classical MH acceptance/rejection framework. In this case, the MH relies on the Hamiltonian function, which is defined as follows:

$$H(w, r) = U(w) + \frac{1}{2} r^\top M^{-1} r. \tag{2.24}$$

Algorithm 1 provides a summary of the HMC algorithm.

Stochastic gradient Hamiltonian Monte Carlo.   Despite its effectiveness in sampling from the posterior, HMC comes with a considerable computational cost, as it necessitates evaluating the gradient of the potential energy function $U(w)$ at every iteration. While evaluating the gradient of the prior component is generally feasible, computing the likelihood gradient can be computationally challenging, particularly for large datasets. To address this issue, one potential approach involves assuming that the likelihood factors on the observations, allowing us to approximate this

---

Algorithm 1: Hamiltonian Monte Carlo

Input: Starting position $\boldsymbol{w}^{(0)}$ and step size ε

1 **for** $t = 1, 2, \ldots$ **do**

    `/* Resample momentum r:                                    */`

2     $\boldsymbol{r}^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$

3     $(\boldsymbol{w}_0, \boldsymbol{r}_0) = (\boldsymbol{w}^{(t)}, \boldsymbol{r}^{(t)})$

    `/* Simulate discretization of Hamiltonian dynamics in` **Eq. 2.22** `and` **Eq. 2.23**`:`
    `*/`

4     $\boldsymbol{r}_0 \leftarrow \boldsymbol{r}_0 - \frac{\varepsilon}{2} \nabla U(\boldsymbol{w}_0)$

5   **for** $i = 1, 2, \ldots, m$ **do**

6         $\boldsymbol{w}^{(i)} \leftarrow \boldsymbol{w}^{(i-1)} + \varepsilon \boldsymbol{M}^{-1} \boldsymbol{r}_{i-1}$

7         $\boldsymbol{r}_i \leftarrow \boldsymbol{r}_{i-1} - \varepsilon \nabla U(\boldsymbol{w}^{(i)})$

8     $\boldsymbol{r}_m \leftarrow \boldsymbol{r}_m - \frac{\varepsilon}{2} \nabla U(\boldsymbol{w}_m)$

9     $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{r}}) = (\boldsymbol{w}_m, \boldsymbol{r}_m)$

    `/* Metropolis-Hastings correction:                         */`

10    $u \sim \text{Uniform}(0, 1)$

11    $\rho = e^{H(\hat{\boldsymbol{w}}, \hat{\boldsymbol{r}}) - H(\boldsymbol{w}^{(t)}, \boldsymbol{r}^{(t)})}$

12   **if** $u < \min(1, \rho)$ **then**

13       $\boldsymbol{w}^{(t+1)} = \hat{\boldsymbol{w}}$

---

quantity using mini-batches of size M:

$$\nabla \tilde{U}(\boldsymbol{w}) \approx -\frac{N}{M} \sum_{i=1}^{M} \nabla \log p(y_i \mid f(\boldsymbol{x}_i, \boldsymbol{w})) - \nabla \log p(\boldsymbol{w}), \tag{2.25}$$

By utilizing mini-batches, we can write the gradients as unbiased estimates of $U(\boldsymbol{w})$ as follows:

$$\nabla \tilde{U}(\boldsymbol{w}) = \nabla U(\boldsymbol{w}) + \mathcal{N}(\boldsymbol{0}, \boldsymbol{V}(\boldsymbol{\theta})), \tag{2.26}$$

where $\boldsymbol{V}(\boldsymbol{\theta})$ represents the covariance of the stochastic gradient noise. It should be noted that this covariance is dependent on various factors, including the parameters themselves.

A naïve approach to implement stochastic gradient Hamiltonian Monte Carlo (SGHMC) would be to simply replace the gradient in Algorithm 1 with Eq. 2.25. However, Chen et al. (2014) prove that the the simulation of the resulting dynamics would make the stationary distribution no longer invariant (i.e., sampling is performed on a distribition which is not the original target distribution). To address this issue, Chen et al. (2014) propose a modification to the HMC algorithm, which involves adding a friction term to the dynamics. The discretized Hamiltonian dynamics are then updated as

follows:

$$\begin{cases} \Delta \boldsymbol{w} &= \varepsilon \boldsymbol{M}^{-1} \boldsymbol{r}, \\ \Delta \boldsymbol{r} &= -\varepsilon \nabla \tilde{U}(\boldsymbol{w}) - \varepsilon \boldsymbol{C} \boldsymbol{M}^{-1} \boldsymbol{r} + \mathcal{N}(0, 2\varepsilon(\boldsymbol{C} - \tilde{\boldsymbol{B}})), \end{cases} \tag{2.27}$$

where ε is an step size, $\boldsymbol{C}$ is an user-defined friction matrix, $\tilde{\boldsymbol{B}}$ is the estimate for the noise of the gradient evaluation.

Hyperparameter selection for SGHMC. Working with the dynamics system Eq. 2.27 remains challenging from a practical point of view due to the need to select the friction term $\boldsymbol{C}$, estimate the noise covariance $\tilde{\boldsymbol{B}}$, choose the mass matrix $\boldsymbol{M}$, and tune the step size ε. The selection of the friction term and step size relies heavily on the specific model and dataset at hand. However, the remaining two quantities can be estimated during the *burn-in phase*, as proposed by Springenberg et al. (2016).

More specifically, Springenberg et al. (2016) leverage the connection between SGHMC and stochastic gradient descent (SGD) to choose the mass matrix $\boldsymbol{M}$. In particular, Duchi et al. (2011) and Tieleman and Hinton (2012) show how normalizing the gradient by its magnitude (estimated over the whole dataset) improves the robustness of SGD. For SGHMC, this is equivalent to choosing the mass matrix $\boldsymbol{M}^{-1} = \text{diag}\left(\hat{V}_w^{-1/2}\right)$, where $\hat{V}_w$ is an estimate of the uncentered variance of the gradient, $\hat{V}_w \approx \mathbb{E}[(\nabla \tilde{U}(\boldsymbol{w}))^2]$. This quantity can be estimated by using exponential moving average as follows:

$$\Delta \hat{V}_w = -\tau^{-1} \hat{V}_w + \tau^{-1} \nabla (\tilde{U}(\boldsymbol{w}))^2, \tag{2.28}$$

where τ is a parameter vector that specifies the moving average windows. This parameter can be automatically chosen by using a procedure similar to the adaptive learning rate for SGD (Tieleman and Hinton, 2012) as follows:

$$\Delta \tau = -g_w^2 \hat{V}_w^{-1} \tau + 1, \quad \text{and,} \quad \Delta g_w = -\tau^{-1} g_w + \tau^{-1} \nabla \tilde{U}(\boldsymbol{w}), \tag{2.29}$$

where g_w is a smoothed estimate of the gradient $\nabla U(\boldsymbol{w})$.

The estimate for the noise of the gradient evaluation $\tilde{\boldsymbol{B}}$ should be ideally an estimate of the empirical Fisher information matrix of $U(\boldsymbol{w})$, which is prohibitively expensive to compute. Therefore, we can use a diagonal approximation, $\tilde{\boldsymbol{B}} = \frac{1}{2}\varepsilon \hat{V}_w$. For the friction matrix, in practice, one can simply set it as $\boldsymbol{C} = C\boldsymbol{I}$, i.e. the same independent noise for each element of $\boldsymbol{w}$.

By substituting these new quantities and $\boldsymbol{v} := \varepsilon \hat{V}_w^{-1/2} \boldsymbol{r}$, the dynamics Eq. 2.27 become

$$\begin{cases} \Delta \boldsymbol{w} &= \boldsymbol{v}, \\ \Delta \boldsymbol{v} &= -\varepsilon^2 \hat{V}_w^{-1/2} \nabla \tilde{U}(\boldsymbol{w}) - \varepsilon C \hat{V}_w^{-1/2} \boldsymbol{v} + \mathcal{N}(0, 2\varepsilon^3 C \hat{V}_w^{-1} - \varepsilon^4 \boldsymbol{I}). \end{cases} \tag{2.30}$$

Springenberg et al. (2016) suggest to choose C such that $\varepsilon C \hat{V}_w^{-1/2} = \alpha I$. This is equivalent to using a constant momentum coefficient of α. The final discretized dynamics are as follows:

$$
\begin{cases}
\Delta w & = v, \\
\Delta v & = -\varepsilon^2 \hat{V}_w^{-1/2} \nabla \tilde{U}(w) - \alpha v + \mathcal{N}(0, 2\varepsilon^2 \alpha \hat{V}_w^{-1/2} - \varepsilon^4 I).
\end{cases}
\tag{2.31}
$$

2.4   Gaussian Processes

In the preceding sections, we discuss neural networks as *parametric* models, where the model's complexity is inherently tied to the number of parameters. In this section, we consider a type of *non-parametric* model for the mapping f, which is Gaussian process (GP) (Rasmussen and Williams, 2006). Unlike parametric models, the complexity of GPs actually grows with the size of data available.

GPs are a simple and general class of models of functions. A GP is any distribution over functions such that any finite set of functions values $f(x_1), f(x_2), \ldots, f(x_N)$ have a joint Gaussian distribution (Rasmussen and Williams, 2006). It is fully characterized by its mean function, $\mu_\theta(x)$, and its covariance function or kernel, $\kappa_\theta(x, x')$, i.e.:

$$
f(x) \sim \mathcal{GP}(\mu_\theta(x), \kappa_\theta(x, x')),
\tag{2.32}
$$

where x and x' represent the input locations where the function $f(\cdot)$ is evaluated, while θ comprises the hyper-parameters of both the mean and covariance functions. Based on the previously stated GP definition, we can formulate the distribution for a finite set of function values as:

$$
p\left(\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix} \middle| \theta\right) = \mathcal{N}\left(\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}; \begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_N) \end{bmatrix}, \begin{bmatrix} \kappa_\theta(x_1, x_1) & \kappa_\theta(x_1, x_2) & \cdots & \kappa_\theta(x_1, x_N)) \\ \kappa_\theta(x_2, x_1) & \kappa_\theta(x_2, x_2) & \cdots & \kappa_\theta(x_2, x_N)) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_\theta(x_N, x_1) & \kappa_\theta(x_N, x_2) & \cdots & \kappa_\theta(x_N, x_N)) \end{bmatrix}\right).
\tag{2.33}
$$

The common assumption for the mean function is often that it is a zero function, denoted as $\mu_\theta(x) = 0$. This choice stems from the rationale that the *prior* knowledge about the function $f(\cdot)$ can be encapsulated in the form of the covariance function and its associated hyper-parameters θ. The selection of the covariance function is determined by the prior insights about the nature of the function, such as its smoothness, oscillations, roughness, or periodicity, see Chapter 2 in Duvenaud (2014) for examples of different covariance functions. A popular covariance function is the radial basis function (RBF) or squared exponential kernel with automatic relevance

determination (ARD) (MacKay, 1996a), which is defined as follows:

$$\kappa_\theta(\boldsymbol{x}, \boldsymbol{x}') = \alpha^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{l_d^2}\right), \tag{2.34}$$

where l_d is the lengthscale for the d-th input dimension and α^2 is the kernel variance, and in this case $\theta = \{\alpha^2\} \cup \{l_d\}_{d=1}^{D}$ are the kernel hyper-parameters.

Consider the case of regression, employing a Gaussian likelihood. Here, every observation y_n is a result of an underlying, unknown function $f(\cdot)$ evaluated at the input $\boldsymbol{x}_n$. This evaluation is perturbed by independent Gaussian noise, with a variance of σ_ϵ^2. A GP can be used to specify the prior over functions $f(\cdot)$ and the corresponding probabilistic model is as follows:

$$f \mid \theta \sim \mathcal{GP}(0, \kappa_\theta(\cdot, \cdot)), \tag{2.35}$$

$$p(\boldsymbol{y} \mid \boldsymbol{f}, \sigma_\epsilon^2) = \prod_{n=1}^{N} \mathcal{N}(y_n; f(\boldsymbol{x}_n), \sigma_\epsilon^2), \tag{2.36}$$

where $\boldsymbol{y}$ is a vector of comporising of all the training outputs, and $\boldsymbol{f}$ is the evaluations of the function $f(\cdot)$ on all the training inputs, i.e. $\boldsymbol{f} = f(\boldsymbol{X}) = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top$.

Prediction. The model allows us to make a prediction of the function value at a new (test) input location $\boldsymbol{x}^\star$, i.e. $f(\boldsymbol{x}^\star)$. Because the joint distribution between the training observations and the test latent functions is a multivariate Gaussian distribution, the posterior can be obtained using the conditional Gaussian distribution property that implies a GP as well (Rasmussen and Williams, 2006):

$$f(\boldsymbol{x}^\star) \mid \boldsymbol{y} \sim \mathcal{GP}(\underbrace{\boldsymbol{K}_{\boldsymbol{x}^\star \boldsymbol{X}}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y}}_{\text{predictive mean follows observations}}, \underbrace{\boldsymbol{K}_{\boldsymbol{x}^\star \boldsymbol{x}^\star} - \boldsymbol{K}_{\boldsymbol{x}^\star \boldsymbol{X}}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{x}^\star}}_{\text{predictive var. shrinks given more data}}), \tag{2.37}$$

where $\boldsymbol{K}_{\boldsymbol{x}^\star \boldsymbol{X}}$, $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}$ are the covariance matrices between the test function values and training function values, and the training function values and themselves, respectively.

Model selection. The aforementioned procedure enables us to obtain the GP posterior and make predictions at test inputs, given a *fixed* set of kernel hyper-parameters θ and noise variance σ_ϵ^2. However, these values are usually not known in advance and often pose a challenge for manual selection. To address this, a fully Bayesian approach can be adopted. This involves specifying priors for the hyper-parameters and deriving the joint posterior distribution, denoted as $p(\boldsymbol{f}, \theta, \sigma_\epsilon^2 \mid \boldsymbol{y})$. However, this procedure is not analytically available and necessitates approximation techniques like MCMC. Alternatively, a common practice is to select the hyper-parameters $\{\theta, \sigma_\epsilon^2\}$ by maximizing the marginal likelihood of the hyper-parameters. In the regression

scenario, the marginal likelihood can be computed analytically, and its logarithm is as follows:

$$\log p(\boldsymbol{y} \,|\, \theta, \sigma_\epsilon^2) = \log \int p(\boldsymbol{y} \,|\, \boldsymbol{f}, \sigma_\epsilon^2) p(\boldsymbol{f} \,|\, \boldsymbol{\theta}) d\boldsymbol{f} \tag{2.38}$$

$$= \log \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{K}_{XX} + \sigma_\epsilon^2 \boldsymbol{I}) \tag{2.39}$$

$$= \underbrace{-\frac{1}{2}\boldsymbol{y}^\top (\boldsymbol{K}_{XX} + \sigma_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{y}}_{\text{encourages fit with data}} - \underbrace{\frac{1}{2} \log |\boldsymbol{K}_{XX} + \sigma_\epsilon^2 \boldsymbol{I}|}_{\text{controls model capacity}} - \underbrace{\frac{N}{2} \log 2\pi}_{\text{constant}}. \tag{2.40}$$

As can be seen from the above objective, the marginal likelihood embodies a trade-off between the fit with the data and the model complexity. Consequently, this approach is anticipated to offer resilience against overfitting. However, it's worth noting that the optimization routine might encounter challenges with regard to local maxima (Rasmussen and Williams, 2006).

Complexity.   As can be seen from Eq. 2.40 and Eq. 2.36 and the computational complexity of the GP for learning hyper-parameters and making predictions is largely dominated by the matrix inversion of $\boldsymbol{K}_{XX} + \sigma_\epsilon^2 \boldsymbol{I}$. This inversion incurs a time complexity of $\mathcal{O}(N^3)$, and during the learning phase, it necessitates repetition multiple times. After this step, subsequent predictions for a test data point can be executed with $\mathcal{O}(N^2)$ time complexity.

Chapter 3

Functional Priors for Bayesian Neural Networks

The Bayesian treatment of neural networks dictates that a prior distribution is specified over their weight and bias parameters. This poses a challenge because modern neural networks are characterized by a large number of parameters, and the choice of these priors has an uncontrolled effect on the induced functional prior, which is the distribution of the functions obtained by sampling the parameters from their prior distribution. We argue that this is a hugely limiting aspect of Bayesian deep learning, and the work presented in this chapter tackles this limitation in a practical and effective way. Our proposal is to reason in terms of functional priors, which are easier to elicit, and to "tune" the priors of neural network parameters in a way that they reflect such functional priors. Gaussian processes offer a rigorous framework to define prior distributions over functions, and we propose a novel and robust framework to match their prior with the functional prior of neural networks based on the minimization of their Wasserstein distance. We provide vast experimental evidence that coupling these priors with scalable Markov chain Monte Carlo sampling offers systematically large performance improvements over alternative choices of priors and state-of-the-art approximate Bayesian deep learning approaches. We consider this work a considerable step in the direction of making the long-standing challenge of carrying out a fully Bayesian treatment of neural networks, including convolutional neural networks, a concrete possibility.

3.1 Introduction

The majority of tasks in machine learning, including classical ones such as classification and regression, can be reduced to estimation of functional representations, and neural networks offer a powerful framework to describe functions of high complexity.

In this work, we focus on the Bayesian treatment of neural networks, which results in a natural form of regularization and allows one to reason about uncertainty in predictions (Tishby et al., 1989; Neal, 1996; Mackay, 2003). Despite the lack of conjugate priors for any Bayesian neural networks (BNNs) of interest, it is possible to generate samples from the posterior distributions over their parameters by means of Markov chain Monte Carlo algorithms (Neal, 1996; Chen et al., 2014).

The concept of prior distribution in Bayesian inference allows us to describe the family of solutions that we consider acceptable, *before* having seen any data. While in some cases selecting an appropriate prior is easy or intuitive given the context (O'Hagan, 1991; Rasmussen and Ghahramani, 2002; Srinivas et al., 2010; Cockayne et al., 2019; Briol et al., 2019; Tran et al., 2021), for nonlinear parametric models with thousands (or millions) of parameters, like deep neural networks (DNNs) and convolutional neural networks (CNNs), this choice is not straightforward. As these models are nowadays accepted as the *de facto standard* in machine learning (LeCun et al., 2015), the community has been actively proposing ways to enable the possibility to reason about the uncertainty in their predictions, with the Bayesian machinery being at the core of many contributions (Graves, 2011; Chen et al., 2014; Gal and Ghahramani, 2016; Liu and Wang, 2016). Despite many advances in the field (Kendall and Gal, 2017; Rossi et al., 2019; Osawa et al., 2019a; Rossi et al., 2020), it is reported that in some cases the predictive posteriors are not competitive to non-Bayesian alternatives, making these models—and Bayesian deep learning, in general—less than ideal solutions for a number of applications. For example, Wenzel et al. (2020) have raised concerns about the quality of BNN posteriors, where it is found that tempering the posterior distribution improves the performance of some deep models.

We argue that observations of this kind should not be really surprising. Bayesian inference is a recipe with exactly three ingredients: the *prior distribution*, the *likelihood* and the *Bayes' rule*. Regarding the Bayes' rule, that is simply a consequence of the axioms of probability. The fact that the posterior might not be useful in some cases should never be attributed to the Bayesian method itself. In fact, it is very easy to construct Bayesian models with poor priors and/or likelihoods, which result in poor predictive posteriors. One should therefore turn to the other two components, which encode model assumptions.

In this work, we focus our discussion and analysis on the prior distribution of BNNs. For such models, the common practice is to define a prior distribution on the network weights and biases, which is often chosen to be Gaussian. A prior over the parameters induces a prior on the functions generated by the model, which also depends on the network architecture. However, due to the nonlinear nature of the model, the effect of this prior on the functional output is not obvious to characterize and control.

Consider the example in Fig. 3.1, where we show the functions generated by sampling the weights of BNNs with a tanh activation from their Gaussian prior $\mathcal{N}(0,1)$. We
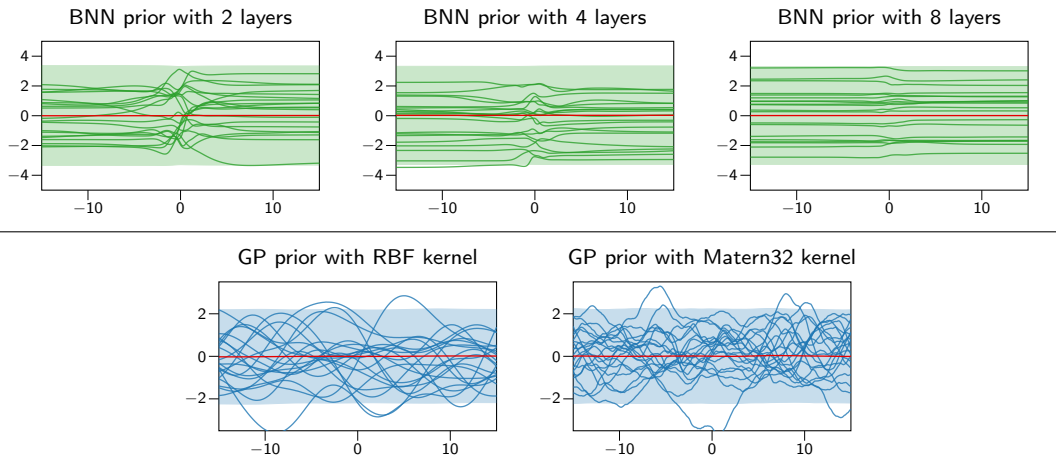
Figure 3.1: (Top) Sample functions of a fully-connected BNN with 2, 4 and 8 layers obtained by placing a Gaussian prior on the weights. (**Bottom**) Samples from a GP prior with two different kernels.

see that as depth is increased, the samples tend to form straight horizontal lines, which is a well-known pathology stemming from increasing model's depth (Neal, 1996; Duvenaud et al., 2014; Matthews et al., 2018). We stress that a fixed Gaussian prior on the parameters is not always problematic, but it can be, especially for deeper architectures. Nonetheless, this kind of generative priors on the functions is very different from shallow Bayesian models, such as Gaussian processes (GPs), where the selection of an appropriate prior typically reflects certain attributes that we expect from the generated functions. A GP defines a distribution over functions which is characterized by a mean and a kernel function κ. The GP prior specification can be more *interpretable* than the one induced by the prior over the weights of a BNN, in the sense that the kernel effectively governs the properties of prior functions, such as shape, variability and smoothness. For example, shift-invariant kernels may impose a certain characteristic length-scale on the functions that can be drawn from the prior distribution.

Contributions

The main research question that we investigate in this work is how to impose functional priors on BNNs. We seek to tune the prior distributions over BNNs parameters so that the induced functional priors exhibit interpretable properties, similar to shallow GPs. While BNN priors induce a regularization effect that penalizes large values for the network weights, a GP-adjusted prior induces regularization directly on the space of functions.

We consider the *Wasserstein distance* between the distribution of BNN functions induced by a prior over their parameters, and a target GP prior. We propose an algorithm that optimizes such a distance with respect to the BNN prior parameters and hyperparameters. An attractive property of our proposal is that estimating the

Wasserstein distance relies exclusively on samples from both distributions, which are easy to generate. We demonstrate empirically that for a wide range of BNN architectures with smooth activations, it is possible to sufficiently capture the function distribution induced by popular GP kernels.

We then explore the effect of GP-induced priors on the predictive posterior distribution of BNNs by means of an extensive experimental campaign. We do this by carrying out fully Bayesian inference of neural network models with these priors through the use of scalable Markov chain Monte Carlo (MCMC) sampling (Chen et al., 2014). We demonstrate systematic performance improvements over alternative choices of priors and state-of-the-art approximate Bayesian deep learning approaches on a wide range of regression and classification problems, as well as a wide range of network architectures including convolutional neural networks; we consider this a significant advancement in Bayesian deep learning.

3.2 Related Work

In the field of BNNs, it is common practice to consider a diagonal Gaussian prior distribution for the network weights (Neal, 1996; Bishop, 2006). Certain issues of these kind of BNN priors have been recently exposed by Wenzel et al. (2020), who show that standard Gaussian priors exhibit poor performance, especially in the case of deep architectures. The authors address this issue by considering a temperate version of the posterior, which effectively reduces the strength of the regularization induced by the prior. Many recent works (Chen et al., 2014; Springenberg et al., 2016) consider a hierarchical structure for the prior, where the variance of the normally-distributed BNN weights is governed by a Gamma distribution. This setting introduces additional flexibility on the space of functions, but it still does not provide much intuition regarding the properties of the prior. A different approach is proposed by Karaletsos and Bui (2020) and Karaletsos and Bui (2020), who consider a GP model for the network parameters that can capture weight correlations.

Bayesian model selection constitutes a principled approach to select an appropriate prior distribution. Model selection is based on the marginal likelihood – the normalizing constant of the posterior distribution – which may be estimated from the training data. This practice is usually used to select hyperparameters of a GP as its marginal likelihood is available in closed form (Rasmussen and Williams, 2006). However, the marginal likelihood of BNNs is generally intractable, and lower bounds are difficult to obtain. Graves (2011) first and Blundell et al. (2015) later used the variational lower bound of the marginal likelihood for optimizing the parameters of a prior, yielding in some cases worse results. Recently, Immer et al. (2021b) extended the Mackay's original proposal (MacKay, 1995) of using the Laplace's method to approximate the marginal likelihood. In this way, one can obtain an estimate of the marginal likelihood

which is scalable and differentiable with respect to the prior hyperparameters, such that they can be optimized together with the BNN posterior.

Many recent attempts in the literature have turned their attention towards defining priors in the space of functions, rather than the space of weights. For example, Nalisnick et al. (2021a) consider a family of priors that penalize the complexity of predictive functions. Hafner et al. (2019) propose a prior that is imposed on training inputs, as well as out-of-distribution inputs. This is achieved by creating pseudo-data by means of perturbing the training inputs; the posterior is then approximated by a variational scheme. Yang et al. (2019) present a methodology to induce prior knowledge by specifying certain constraints on the network output. Pearce et al. (2019) explore DNN architectures that recreate the effect of certain kernel combinations for GPs. This result in an expressive family of network priors that converge to GPs in the infinite-width limit.

A similar direction of research focuses not only on priors but also inference in the space of functions for BNNs. For example, Ma et al. (2019) consider a BNN as an implicit prior in function space and then use GPs for inference. Conversely, Sun et al. (2019) propose a functional variational inference which employs a GP prior to regularize BNNs directly in the function space by estimating the Kullback-Leibler (KL) divergence between these two stochastic processes. However, this method relies on a gradient estimator which can be inaccurate in high dimensions. Khan et al. (2019) follow an alternative route by deriving a GP posterior approximation for neural networks by means of the Laplace and generalized Gauss-Newton (GGN) approximations, leading to an implicit linearization. Immer et al. (2021a) make this linearization explicit and apply it to improve the performance of BNN predictions. In general, these approaches either heavily rely on non-standard inference methods or are constrained to use a certain approximate inference algorithm, such as variational inference or Laplace approximation.

A different line of work focuses on meta-learning by adjusting priors based on the performance of previous tasks (Amit and Meir, 2018). In contrast to these approaches, we aim to define a suitable prior distribution entirely *a priori*. We acknowledge that our choice to impose GP (or hierarchical GP) priors on neural networks is essentially heuristic: there is no particular theory that necessarily claims superiority for this kind of prior distributions. In some applications, it could be preferable to use priors that are tailored to certain kinds of data or architectures, such the *deep weight prior* (Atanov et al., 2019). However, we are encouraged by the empirical success and the interpretability of GP models, and we seek to investigate their suitability as BNN priors on a wide range of regression and classification problems.

Our work is most closely related to a family of works that attempt to map GP priors to BNNs. Flam-Shepherd et al. (2017) propose to minimize the KL between the BNN prior and some desired GP. As there is no analytical form for this KL, the authors rely

on approximations based on moment matching and projections on the observation space. This limitation was later addressed (Flam-Shepherd et al., 2018) by means of a hypernetwork (Ha et al., 2017), which generates the weight parameters of the original BNN; the hypernetwork parameters were trained so that a BNN fits the samples of a GP. In our work, we also pursue the minimization of a sample-based distance between the BNN prior and some desired GP, but we avoid the difficulties in working with the KL divergence, as its evaluation is challenging due to the empirical entropy term. To the best of our knowledge, the Wasserstein distance scheme we propose is novel, and it demonstrates satisfactory convergence for compatible classes of GPs and BNNs.

Concurrently to the release of this work, we have come across another work advocating for the use of GP priors to determine priors for BNNs. Matsubara et al. (2021) rely on the *ridgelet transform* to approximate the covariance function of a GP. Our work is methodologically different, as our focus is to propose a practical framework to impose sensible priors. Most importantly, we present an extensive experimental campaign that demonstrates the impact of functional priors on deep models.

3.3 Preliminaries

In this section, we establish some basic notation on GPs that we follow throughout this chapter. In addiition, we give a brief introduction to the concept of Wasserstein distance, which is the central element of our methodology to impose functional GP priors on BNNs.

3.3.1 Gaussian process priors

GPs constitute a popular modeling choice in the field of Bayesian machine learning (Rasmussen and Williams, 2006), as they allow one to associate a certain class of functional representations with a probability measure. A GP is a stochastic process that is uniquely characterized by a mean function $\mu(x)$ and a covariance function $\kappa(x, x')$. The latter is also known as a kernel function, and it determines the covariance between the realization of the function at pairs of inputs x and x'. For a finite set of inputs X, a GP yields a multivariate Gaussian distribution with mean vector $\mu = \mu(X)$ and covariance matrix $K = \kappa(X, X)$.

There is a significant body of research whose objective is to perform inference for GP models; see Liu et al. (2020) for an extensive review. However, in this work we only treat GPs as a means to define meaningful specifications of priors over functions. Different choices for the kernel result in different priors in the space of functions. A

popular choice in the literature is the radial basis function (RBF) kernel:

$$\kappa_{\alpha,l}(\boldsymbol{x},\boldsymbol{x}') = \alpha^2 \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^\top(\boldsymbol{x}-\boldsymbol{x}')}{l^2}\right), \tag{3.1}$$

which induces functions that are infinitely differentiable, as in Fig. 3.1. The subscripts α, l denote the dependency on hyperparameters: α is the *amplitude*, which controls the prior marginal standard deviation, and l is known as the *lengthscale*, as it controls how rapidly sample functions can vary.

Hierarchical GP priors The most common practice in GP literature is to select values for the hyperparameters that optimize the marginal log-likelihood. We do not recommend such an approach in our setting however, as it introduces additional complexity from a computational perspective. Instead, we opt to consider a hierarchical form for the target prior. Assuming a shift-invariant kernel $\kappa_{\alpha,l}(\boldsymbol{x},\boldsymbol{x}')$ with hyperparameters α and l, we have:

$$\alpha, l \sim \text{LogNormal}(m, s^2), \qquad f \sim \mathcal{N}(0, \kappa_{\alpha,l}(\boldsymbol{x},\boldsymbol{x}')) \tag{3.2}$$

where m and s are user-defined parameters. Samples of the target prior are generated by means of a Gibbs sampling scheme: we first sample the hyperparameters from a log-normal distribution, and then we sample from the corresponding GP. This form of hierarchical GP priors is adopted in the majority of experiments of § 3.6, unless otherwise specified.

3.3.2 Wasserstein distance

The concept of distance between probability measures is central to this work, as we frame the problem of imposing a GP prior on a BNN as a distance minimization problem. We present some known results on the Wasserstein distance that will be used in the sections that follow. Given two *Borel's probability measures* $\pi(\boldsymbol{x})$ and $\nu(\boldsymbol{y})$ defined on the *Polish space* $\mathcal{X}$ and $\mathcal{Y}$ (i.e. any complete separable metric space), the generic formulation of the p-**Wasserstein distance** is defined as follows:

$$W_p(\pi, \nu) = \left(\inf_{\gamma \in \Gamma(\pi,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} D(\boldsymbol{x},\boldsymbol{y})^p \gamma(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{x}d\boldsymbol{y}\right)^{1/p}, \tag{3.3}$$

where $D(\boldsymbol{x},\boldsymbol{y})$ is a proper distance metric between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in the space $\mathcal{X} \times \mathcal{Y}$ and $\Gamma(\pi,\nu)$ is the set of functionals of all possible joint densities γ whose marginals are π and ν.

When the spaces of $\boldsymbol{x}$ and $\boldsymbol{y}$ coincide (i.e. $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \subseteq \mathbb{R}^d$), with $D(\boldsymbol{x},\boldsymbol{y})$ being the Euclidian norm distance, the Wasserstein-1 distance (also known in the literature as

Earth-Mover distance) takes the following shape,

$$W_1(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\| \gamma(x, y) dx dy. \tag{3.4}$$

With the exception of few cases where the solution is available analytically (e.g. π and ν being Gaussians), solving Eq. 3.4 directly or via optimization is intractable. On the other hand, the Wasserstein distance defined in Eq. 3.4 admits the following dual form (Kantorovich, 1942; Kantorovich, 1948),

$$\begin{aligned} W_1(\pi, \nu) &= \sup_{\|\phi\|_L \leq 1} \left[\int \phi(x) \pi(x) dx - \int \phi(y) \nu(y) dy \right] \\ &= \sup_{\|\phi\|_L \leq 1} \mathbb{E}_\pi \phi(x) - \mathbb{E}_\nu \phi(x), \end{aligned} \tag{3.5}$$

where ϕ is a 1-Lipschitz continuous function defined on $\mathcal{X} \to \mathbb{R}$. This is effectively a functional maximization over ϕ on the difference two expectations of ϕ under π and ν. A revised proof of this dual form by Villani (2003) is available in the Supplement.

3.4 Imposing Gaussian Process Priors on Bayesian Neural Networks

The equivalence between function-space view and weight-space view of linear models, like Bayesian linear regression and GPs (Rasmussen and Williams, 2006), is a straightforward application of Gaussian identities, but it allows us to seamlessly switch point of view accordingly to which characteristics of the model we are willing to observe or impose. We would like to leverage this equivalence also for BNNs but the nonlinear nature of such models makes it analytically intractable (or impossible, for non-invertible activation functions). We argue that for BNNs—and Bayesian deep learning models, in general—starting from a prior over the weights is not ideal, given the impossibility of interpreting its effect on the family of functions that the model can represent. We therefore rely on an optimization-based procedure to impose functional priors on BNNs using the Wasserstein distance as a similarity metric between such distributions, as described next.

3.4.1 Wasserstein distance optimization

Assume a prior distribution $p(w; \psi)$ on the weights of a BNN, where ψ is a set of parameters that determine the prior (e.g., $\psi = \{\mu, \sigma\}$ for a Gaussian prior; we discuss more options on the parametrization of BNN priors in the section that follows). This prior over weights induces a prior distribution over functions:

$$p_{nn}(f; \psi) = \int p(f \mid w) p(w; \psi) w, \tag{3.6}$$

where $p(f \mid w)$ is deterministically defined by the network architecture.

In order to keep the notation simple, we consider non-hierarchical GP priors. Hierarchical GPs are treated in the same way, except that samples are generated by the Gibbs sampling scheme of Eq. 3.2. Our target GP prior is $p_{gp}(f \mid \mathbf{0}, K)$, where K is the covariance matrix obtained by computing the kernel function κ for each pair of $\{x_i, x_j\}$ in the training set. We aim at matching these two stochastic processes at a finite number of measurement points $X_{\mathcal{M}} \stackrel{\text{def}}{=} [x_1, ..., x_M]^\top$ sampled from a distribution $q(x)$. To achieve this, we propose a sample-based approach using the 1-Wasserstein distance in Eq. 3.5 as objective:

$$\min_{\psi} \max_{\theta} \mathbb{E}_q \Big[\underbrace{\mathbb{E}_{p_{gp}} [\phi_\theta(f_{\mathcal{M}})] - \mathbb{E}_{p_{nn}} [\phi_\theta(f_{\mathcal{M}})]}_{\mathcal{L}(\psi, \theta)} \Big], \tag{3.7}$$

where $f_{\mathcal{M}}$ denotes the set of random variables associated with the inputs at $X_{\mathcal{M}}$, and ϕ_θ is a 1-Lipschitz function. Following recent literature (Goodfellow et al., 2014; Arjovsky et al., 2017), we parameterize the Lipschitz function by a neural network[1] with parameters θ.

Regarding the optimization of the θ and ψ parameters we alternate between $n_{\text{Lipschitz}}$ steps of maximizing $\mathcal{L}$ with respect to the Lipschitz function's parameters θ and one step of minimizing the Wasserstein distance with respect to the prior's parameters ψ. We therefore use two independent optimizers (RMSprop–see, for example, Tieleman and Hinton, 2012) for θ and ψ. Fig. 3.2 offers a high-level schematic representation of the proposed procedure. Given samples from two stochastic processes, the Wasserstein distance is estimated by considering the inner maximization of Eq. 3.7, resulting in an optimal ϕ^*. This inner optimization step is repeated for every step of the outer optimization loop. Notice that the objective is fully sample-based. As a result, it is not necessary to know the closed-form of the marginal density $p_{nn}(f; \psi)$. One may consider any stochastic process as a target prior over functions, as long as we can draw samples from it (e.g., a hierarchical GP). Finally, we acknowledge that the two training steps could have been optimized jointly in a single loop, as Eq. 3.5 defines a minimax problem. However, this choice allows ϕ_θ to converge enough before a single Wasserstein minimization step takes place. In fact, this is a common trick to make convergence more stable (see e.g., the original Goodfellow et al. (2014) paper, which suggests to allow more training of the discriminator for each step of the generator). In Appendix A.3.6 we further discuss this choice and we show qualitatively the convergence improvements.

Lipschitz constraint. In order to enforce the Lipschitz constraint on ϕ_θ, Arjovsky et al. (2017) propose to clip the weights θ to lie within a compact space $[-c, c]$ such that

---

[1]Details on the 1-Lipschitz function: we used a multilayer perceptron (MLP) with two hidden layers, each with 200 units; the activation function is softplus, which is defined as: softplus$(x) = 1/(1 + \exp(-x))$.
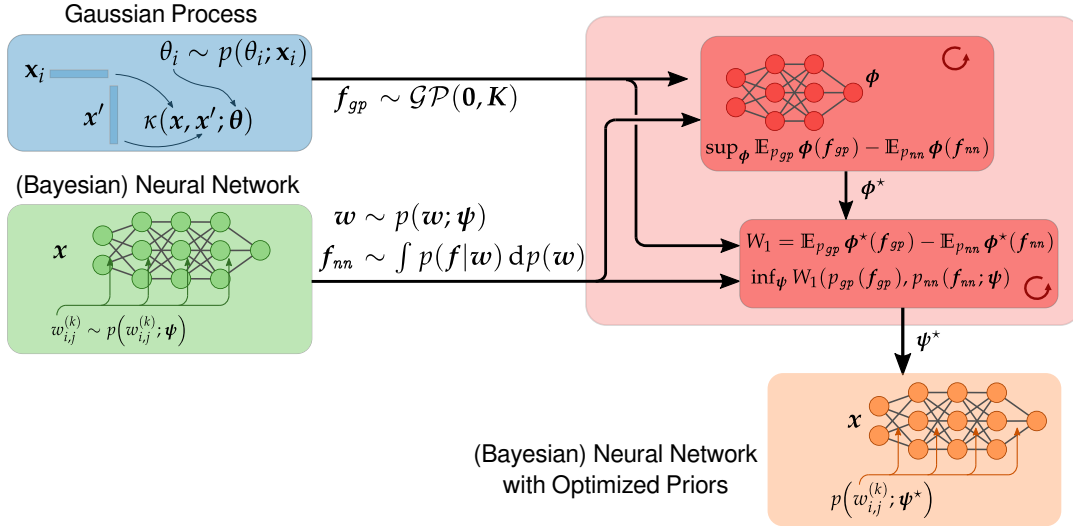
Figure 3.2: Schematic representation of the process of imposing GP priors on BNN via Wasserstein distance minimization.

all functions ϕ_θ are K-Lipschitz. This approach usually biases the resulting ϕ_θ towards a simple function. Based on the fact that a differentiable function is 1-Lipschitz if and only if the norm of its gradient is at most one everywhere, Gulrajani et al. (2017) propose to constrain the gradient norm of the output of the Lipschitz function ϕ_θ with respect to its input. More specifically, the loss of the Lipschitz function is augmented by a regularization term

$$\mathcal{L}_R(\boldsymbol{\psi}, \boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\theta}) + \underbrace{\lambda \mathbb{E}_{p_{\hat{f}}}\left[\left(\left\|\boldsymbol{\nabla}_{\hat{f}} \phi(\hat{f})\right\|_2 - 1\right)^2\right]}_{\text{Gradient penalty}}. \tag{3.8}$$

Here $p_{\hat{f}}$ is the distribution of $\hat{f} = \varepsilon f_{nn} + (1 - \varepsilon) f_{gp}$ for $\varepsilon \sim \mathcal{U}[0, 1]$ and $f_{nn} \sim p_{nn}$, $f_{gp} \sim p_{gp}$ being the sample functions from BNN and GP priors, respectively; λ is a penalty coefficient.

Choice of the measurement set.   In our formulation, we consider finite measurement sets to have a practical and well-defined optimization strategy. As discussed by Shi et al. (2019), there are several approaches to define the measurement set for functional-space inference (Hafner et al., 2019; Sun et al., 2019). For low-dimensional problems, one can simply use a regular grid or apply uniform sampling in the input domain. For high-dimensional problems, one can sample from the training set, possibly with augmentation, where noise is injected into the data. In applications where we know the input region of the test data points, we can set $q(x)$ to include it. We follow a combination of the two approaches: we use the training inputs (or a subset of thereof) as well as additional points that are randomly sampled (uniformly) from the input domain.

3.4.2 Prior parameterization for neural networks

In the previous section, we have treated the parameters of a BNN prior $p_{nn}(f; \psi)$ in a rather abstract manner. Now we explore three different parametrizations of increasing complexity. The only two requirements needed to design a new parametrization are (1) to be able to generate samples and (2) to compute the log-density at any point; the latter is required to be able to draw samples from the posterior over model parameters using most MCMC sampling methods, such as stochastic gradient Hamiltonian Monte Carlo (SGHMC) which we employ in this work.

Gaussian prior on weights.   We consider a layer-wise factorization with two independent zero-mean Gaussian distributions for weights and biases. The parameters to adjust are $\psi = \{\sigma^2_{l_w}, \sigma^2_{l_b}\}^L_{l=1}$, where $\sigma^2_{l_w}$ is the prior variance shared across all weights in layer l, and $\sigma^2_{l_b}$ is the respective variance for the bias parameters. For any weight and bias entries $w_l, b_l \in \boldsymbol{w}_l$ of the l-th layer, the prior is:

$$p(w_l) = \mathcal{N}\left(w_l;\ 0, \sigma^2_{l_w}\right) \quad \text{and} \quad p(b_l) = \mathcal{N}\left(b_l;\ 0, \sigma^2_{l_b}\right).$$

In the experimental section, we refer to this parametrization as the GP-*induced* BNN *prior with Gaussian weights* (GPi-G). Although this simple approach assumes a Gaussian prior on the parameters, in many cases it is sufficient to capture the target GP-based functional priors.

Regarding the implementation of this scheme, there are a few technical choices to discuss. In order to maintain positivity for the standard deviation σ and perform unconstrained optimization, we optimize ρ such that $\sigma = \log(1 + e^\rho)$, which guarantees that σ is always positive. Also, we have to use gradient backpropagation through stochastic variables such as w_l. Thus, in order to treat the parameter w_l in a deterministic manner, instead of sampling the prior distribution directly $w_l \sim \mathcal{N}\left(w_l;\ 0, \sigma^2_{l_w}\right)$, we use the reparameterization trick (Rezende et al., 2014; Kingma and Welling, 2014), and sample from the noise distribution instead,

$$w_l := \sigma_{l_w}\varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1). \tag{3.9}$$

Hierarchical prior.   A more flexible family of priors for BNNs considers a hierarchical structure where the network parameters follow a conditionally Gaussian distribution, and the prior variance for each layer follows an Inverse-Gamma distribution. For the weight and bias variances we have:

$$\sigma^2_{l_w} \sim \Gamma^{-1}(\alpha_{l_w}, \beta_{l_w}) \quad \text{and} \quad \sigma^2_{l_b} \sim \Gamma^{-1}(\alpha_{l_b}, \beta_{l_b})$$

In this case, we have $\psi = \{\alpha_{l_w}, \beta_{l_w}, \alpha_{l_b}, \beta_{l_b}\}^L_{l=1}$, where $\alpha_{l_w}, \beta_{l_w}, \alpha_{l_b}, \beta_{l_b}$ denote the shape and rate parameters of the Inverse-Gamma distribution for the weight and

biases correspondingly for layer l. The conditionally Gaussian prior over the network parameters is given as in the previous section. In the experiments, we refer to this parametrization as the GP-*induced* BNN *prior with Hierarchically-distributed weights* (GPi-H).

Similar to the Gaussian prior, we impose positivity constraints on the shape and rate of the Inverse-Gamma distribution. In addition, we apply the reparameterization trick proposed by Jankowiak and Obermeyer (2018) for the Inverse-Gamma distribution. This method computes an implicit reparameterization using a closed-form approximation of the CDF derivative. We used the corresponding original PyTorch (Paszke et al., 2019) implementation of the method in our experiments.

Beyond Gaussians with Normalizing flows.   Finally, we also consider normalizing flows (NFs) as a family of much more flexible distributions. By considering an invertible, continuous and differentiable function $t : \mathbb{R}^{D_l} \to \mathbb{R}^{D_l}$, where D_l is the number of parameters for l-th layer, a NF is constructed as a sequence of K of such transformations $\mathcal{T}_K = \{t_1, \ldots, t_K\}$ of a simple known distribution (e.g., Gaussian). Sampling from such distribution is as simple as sampling from the initial distribution and then apply the set of transformation $\mathcal{T}_K$. Given an initial distribution $p_0(\boldsymbol{w}_l)$, by denoting $p(\mathcal{T}_K(\boldsymbol{w}_l))$ the final distribution, its log-density can be analytically computed by taking into account to Jacobian of the transformations as follows,

$$\log p(\mathcal{T}_K(\boldsymbol{w}_l)) = \log p_0(\boldsymbol{w}_l) - \sum_{k=1}^{K} \log \left| \det \frac{\partial t_k(\boldsymbol{w}_{l_{k-1}})}{\partial \boldsymbol{w}_{l_{k-1}}} \right|, \qquad (3.10)$$

where $\boldsymbol{w}_{l_{k-1}} = (t_{k-1} \circ \ldots \circ t_2 \circ t_1)(\boldsymbol{w}_l)$ for $k > 1$, and $\boldsymbol{w}_{l_0} = \boldsymbol{w}_l$.

We shall refer to this class of BNN priors as the GP-*induced* BNN *prior, parametrized by normalizing flows* (GPi-NF). We note that NFs are typically used differently in the literature; while previous works showed how to use this distributions for better approximation of the posterior in variational inference (Rezende and Mohamed, 2015; Kingma et al., 2016; Louizos and Welling, 2017) or for parametric density estimation (e.g., Grover et al., 2018), or for enlarging the flexibility of a prior for variational autoencoders (VAEs) (e.g., Chen et al., 2017), as far as we are aware this is the first time that NFs are used to characterize a prior distribution for BNNs.

In our experiments, we set the initial distribution $p_0(\boldsymbol{w}_l)$ to a fully-factorized Gaussian $\mathcal{N}(\boldsymbol{w}_l \,|\, \boldsymbol{0}, \sigma_l^2 \boldsymbol{I})$. We then employ a sequence of four *planar flows* (Rezende and Mohamed, 2015), each defined as

$$t_k(\boldsymbol{w}_{l_{k-1}}) = \boldsymbol{w}_{l_{k-1}} + \boldsymbol{u}_{l_k} h(\boldsymbol{\theta}_{l_k}^{\top} \boldsymbol{w}_{l_{k-1}} + b_{l_k}), \qquad (3.11)$$

where $\boldsymbol{u}_{l_k} \in \mathbb{R}^{D_l}, \boldsymbol{\theta}_{l_k} \in \mathbb{R}^{D_l}, b_{l_k} \in \mathbb{R}$ are trainable parameters, and $h(\cdot) = \tanh(\cdot)$. The log-determinant of the Jacobian of t_k is

$$\log \left| \det \frac{\partial t_k(\boldsymbol{w}_{l_{k-1}})}{\partial \boldsymbol{w}_{l_{k-1}}} \right| = \log \left| 1 + \boldsymbol{u}_{l_k}^\top \boldsymbol{\theta}_{l_k} h'(\boldsymbol{\theta}_{l_k}^\top \boldsymbol{w}_{l_{k-1}} + b_{l_k}) \right|. \tag{3.12}$$

Thus for the *l*-th BNN layer, the parameters to optimize are $\boldsymbol{\psi}_l = \{\sigma_l^2\} \bigcup \{\boldsymbol{u}_{l_k}, \boldsymbol{\theta}_{l_k}, b_{l_k}\}_{k=1}^K$.

Algorithm 2: Wasserstein Distance Optimization

---
1 **Requires**: N_s, number of stochastic process samples; $q(\boldsymbol{x})$, sampling distribution for measurement set; $n_{\text{Lipschitz}}$, number of iterations of Lipschitz function per prior iteration;

2 **while** $\boldsymbol{\psi}$ *has not converged* **do**
3   draw $X_{\mathcal{M}}$ from $q(\boldsymbol{x})$ // `Sample measurement set`;
4   **for** $t = 1, ..., n_{\text{Lipschitz}}$ **do**
5   draw GP functions $\{\boldsymbol{f}_{gp}^{(i)}\}_{i=1}^{N_s} \sim p_{gp}(\boldsymbol{f}; \kappa)$ at $X_{\mathcal{M}}$;
6   draw NN functions $\{\boldsymbol{f}_{nn}^{(i)}\}_{i=1}^{N_s} \sim p_{nn}(\boldsymbol{f}; \boldsymbol{\psi})$ at $X_{\mathcal{M}}$;
7          $\mathcal{L}_R = N_s^{-1} \sum_{i=1}^{N_s} \mathcal{L}_R^{(i)}$ // `Compute Lipschitz objective` $\mathcal{L}_R$ `using` Eq. 3.8;
8          $\boldsymbol{\theta} \leftarrow \text{Optimizer}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \mathcal{L}_R)$ // `Update Lipschitz function` $\phi_{\boldsymbol{\theta}}$;
9   **end**
10   draw GP functions $\{\boldsymbol{f}_{gp}^{(i)}\}_{i=1}^{N_s} \sim p_{gp}(\boldsymbol{f}; \kappa)$ at $X_{\mathcal{M}}$;
11   draw NN functions $\{\boldsymbol{f}_{nn}^{(i)}\}_{i=1}^{N_s} \sim p_{nn}(\boldsymbol{f}; \boldsymbol{\psi})$ at $X_{\mathcal{M}}$;
12      $\widetilde{W}_1 = N_s^{-1} \sum_{i=1}^{N_s} \phi_{\boldsymbol{\theta}}(\boldsymbol{f}_{gp}^{(i)}) - \phi_{\boldsymbol{\theta}}(\boldsymbol{f}_{nn}^{(i)})$ // `Compute Wasserstein-1 distance using` Eq. 3.7;
13      $\boldsymbol{\psi} \leftarrow \text{Optimizer}(\boldsymbol{\psi}, \nabla_{\boldsymbol{\psi}} \widetilde{W}_1)$ // `Update prior` p_{nn};
14 **end**

3.4.3 Algorithm and complexity

Algorithm 2 summarizes our proposed method in pseudocode. The outer loop is essentially a gradient descent scheme that updates the $\boldsymbol{\psi}$ parameters that control the BNN prior. The inner loop is responsible for the optimization of the Lipschitz function $\phi_{\boldsymbol{\theta}}$, which is necessary to estimate the Wasserstein distance. The computational complexity is dominated by the number of stochastic process samples N_s used for the calculation of the Wasserstein distance, and the size $N_{\mathcal{M}}$ of the measurement set $X_{\mathcal{M}}$.

Sampling from a BNN prior does not pose any challenges; N_s samples can be generated in $\mathcal{O}(N_s)$ time. However, sampling from a GP is of cubic complexity, as it requires linear algebra operations such as the Cholesky decomposition. The total complexity of sampling from a hierarchical GP target is $\mathcal{O}(N_s^2 N_{\mathcal{M}}^3)$, as the Cholesky decomposition should be repeated for every sample. For a single step of the outer loop in Algorithm 2, we have to account the $n_{\text{Lipschitz}}$ steps required for the calculation of the distance, resulting in complexity of $\mathcal{O}(n_{\text{Lipschitz}} N_s^2 N_{\mathcal{M}}^3)$ per step. Although our approach introduces an extra computational burden, we note that this is not directly connected to the size of the dataset. We argue that it is worthwhile to invest this

additional cost before the actual posterior sampling phase (via SGHMC), and this is supported by our extensive experimental campaign.

The complexity also depends on the number of parameters in ψ, whose size is a function of the network architecture and the prior parameterization. For the Gaussian and hierarchical parameterizations discussed in § 3.4.2 (i.e. GPi-G and GPi-H), the set $\boldsymbol{\psi}$ grows sub-lineraly with the number of network parameters, as we consider a single weight/bias distribution per layer. The obvious advantage of this arrangement is that our approach can be easily scaled to deep architectures, such as PRERESNET20 and VGG16, as we demonstrate in the experiments.

In the case where BNN weight and bias distributions are represented by normalizing flows, the size of $\boldsymbol{\psi}$ grows linearly with the total number of BNN parameters N_{BNN}. More formally, for a sequence of K transformations, the number of prior parameters that we need to optimize is of order $\mathcal{O}(KN_{\text{BNN}})$. This might be an issue for more complex architectures; in our experiments we apply the GPi-NF configuration for fully connected BNNs only. A more efficient prior parameterization that relies on normalizing flows requires some kind of sparification, which is subject of future work.

3.5 Examples and Practical Considerations

We shall now elaborate on some of the design choices that we have made in this work. First, we visually show the prior one can obtain by using our proposed procedure on a 1D regression (§ 3.5.1) and how the choice of GP priors (in terms of kernel parameters) affects the BNN posterior for 2D classification examples (§ 3.5.2). We then empirically demonstrate that the proposed optimization scheme based on the Wasserstein distance produces a consistent convergence behavior when compared with a KL-based approach (§ 3.5.3).

For these experiments and the rest of the empirical evaluation, we use SGHMC (Springenberg et al., 2016) for posterior inference. The likelihood for regression and classification are set to Gaussian and Bernoulli/multinomial, respectively. Unless otherwise specified, we run four parallel SGHMC chains with a step size of 0.01 and a momentum coefficient of 0.01. We assess the convergence of the predictive posterior based on the $\hat{R}$-statistic (Gelman and Rubin, 1992) over the four chains. In all our experiments, we obtain $\hat{R}$-statistics below 1.1, which indicate convergence to the underlying distribution. To further validate the obtained samples from SGHMC, for a selection of medium-sized datasets we also run a carefully tuned Hamiltonian Monte Carlo (HMC) obtaining similar results (see Table A.8 in the Appendix).
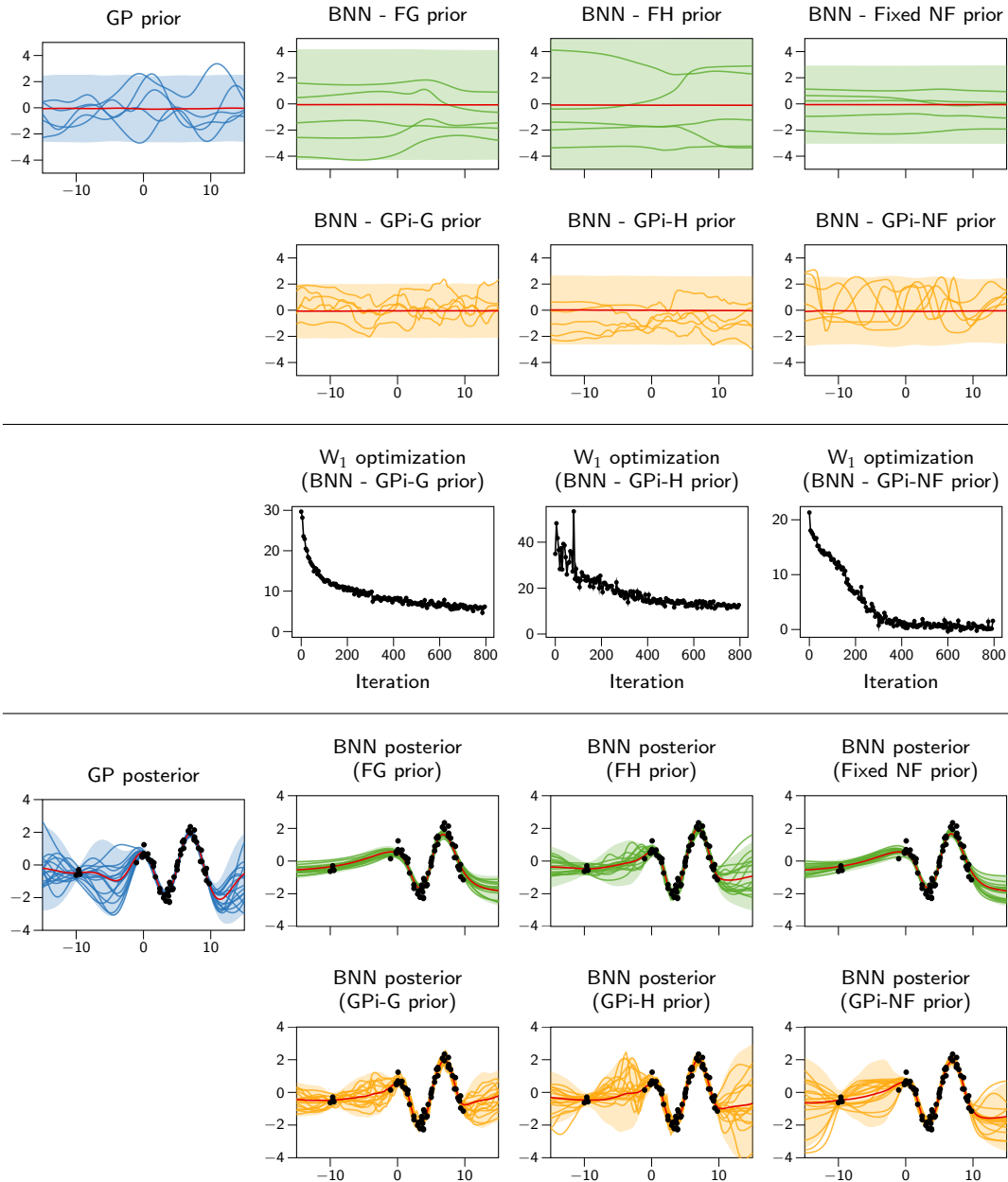
Figure 3.3: Visualization of one-dimensional regression example with a three-layer MLP. The first two rows illustrate the prior sample and distributions, whereas the last two rows show the corresponding posterior distributions. The means and the 95% credible intervals are represented by red lines and shaded areas, respectively. The middle row shows progressions of the prior optimization.

3.5.1 Visualization on a 1D regression synthetic dataset

The dataset used is built as follows: (1) we uniformly sample 64 input locations x in the interval $[-10, 10]$; (2) we rearrange the locations on a defined interval to generate a gap in the dataset; (3) we sample a function f from the GP prior ($l = 0.6, \alpha = 1$) computed at locations x; (4) we corrupt the targets with i.i.d. Gaussian noise ($\sigma_\epsilon^2 = 0.1$). In this example, we consider a three-layer MLP. Fig. 3.3 shows all the results. The
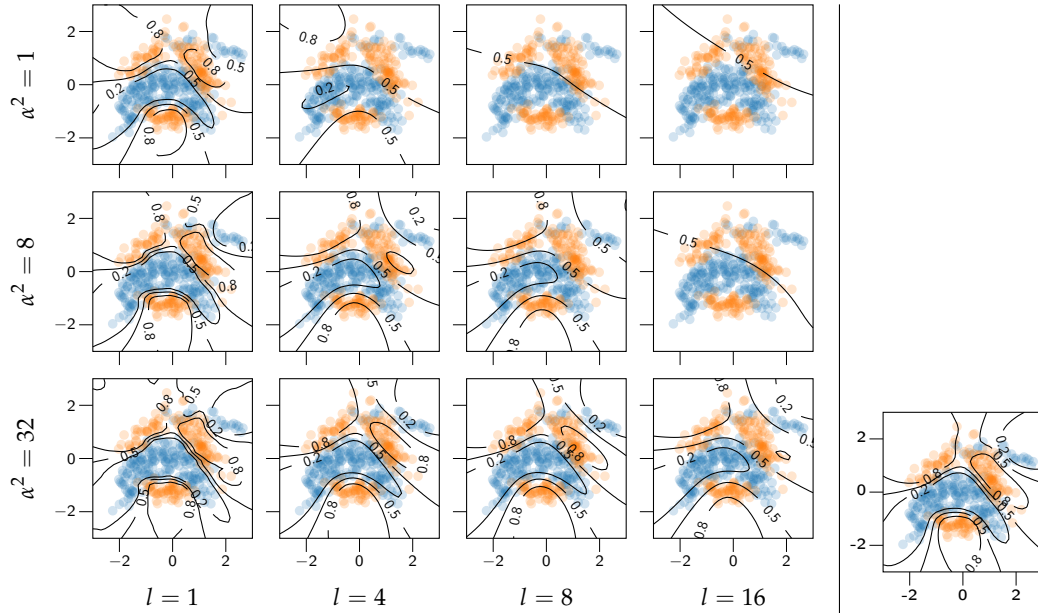
Figure 3.4: (Left) The effect of using different hyperparameters of the RBF kernel of the target GP prior to the predictive posterior. Rows depict increasing the amplitude α, whist columns show increasing the lengthscale l. In each panel the orange and blue dots represent the training points from the two different classes, while the black lines represent decision boundaries at different confidence levels. **(Right)** The predictive posterior with respect to using a target hierarchical-GP prior, in which hyper-priors LogNormal($\log \sqrt{2D}, 1$) and LogNormal($\log 8, 0.3$) are employed on the lengthscales l and variance α^2 respectively, where D is the number of input dimensions.

first two rows illustrate the different choice of priors. For the Wasserstein-based functional priors (GPi-G, GPi-H, GPi-NF), the third row shows the convergence of the optimization procedure. Finally, the last two rows represent the posterior collected by running SGHMC with the corresponding priors.

From the analysis of these plots, we clearly see the benefit of placing a prior on the functions rather than on the parameters. First, the Wasserstein distance plots show satisfactory convergence, with the normalizing flow prior closely matching the GP prior. Second, as expected, the posteriors exhibit similar behavior according to the possible solutions realizable from the prior: classic priors tend to yield degenerate functions resulting in overconfidence in regions without data, while our GP-based priors (GPi-G, GPi-H, GPi-NF) retain information regarding lengthscale and amplitude.

3.5.2 The effects of the GP prior on the BNN posterior

In order to gain insights into the effect of the GP prior (i.e., kernel parameters), we set up an intuitive analysis on the BANANA dataset. We can define the regularization strength of the prior in a sensible way by modifying the hyperparameters of the RBF

kernel. Fig. 3.4 (left) illustrates the predictive posterior of a two-layer BNN, whose prior has been adapted to different target GP priors, featuring different hyperparameters. We observe that the decision boundaries are more complex for smaller lengthscales l and larger amplitudes α, while in the opposite case, we obtain posterior distributions that are too smooth. This behavior reflects the properties of the induced prior.

In a regular GP context, it is possible to tune these hyperparameters by means of marginal likelihood maximization. This is *not* the way we proceed, for two reasons: (1) the overhead to solve the GP and (2) the uselessness of the overall procedure (solving the task with GPs, so to then pick the converged GP prior to solve the BNN inference). As discussed in § 3.3.1, we approach this issue by means of hierarchical GPs. In the rightmost plot of Fig. 3.4, we include the BNN posterior that was adapted to a hierarchical-GP target. Since samples from the target prior can be easily generated using a Gibbs sampling scheme, we can positively impact the expressiveness of the BNN posterior without explicitly worrying which GP prior works best.

3.5.3 Wasserstein distance vs KL divergence

The KL divergence is a popular criterion to measure the similarity between two distributions. In our context, the KL divergence could be used as follows:

$$\mathrm{KL}\left[p_{nn} \parallel p_{gp}\right] = -\int p_{nn}(f;\psi)\log p_{gp}(f)df + \underbrace{\int p_{nn}(f;\psi)\log p_{nn}(f;\psi)df}_{\text{Entropy (intractable)}}, \quad (3.13)$$

This is the form considered by Flam-Shepherd et al. (2017), which propose to minimize the KL divergence between samples of a BNN and a GP. This requires an empirical estimate of the entropy, which is a challenging task for high-dimensional distributions (Delattre and Fournier, 2017). These issues were also reported by Flam-Shepherd et al. (2017), where they propose an early stopping scheme to what is essentially an optimization of the cross-entropy term (i.e., $-\int p_{nn}(f;\psi)\log p_{gp}(f)df$). Instead of computing the entropy, another approach is to estimate its gradient as required by optimization algorithms. This can be carried out by using any methods estimating the log density derivative function of an implicit distribution. For example, Sun et al. (2019) use the spectral Stein gradient estimator (SSGE) (Shi et al., 2018) to obtain an estimate of the gradient of the entropy.

In our experiments, we have found that a scheme based on the Wasserstein distance converges more consistently without the need for additional heuristics. We demonstrate the convergence properties of our scheme against the KL-divergence based optimization with early stopping Flam-Shepherd et al. (2017) and SSGE in Fig. 3.5. In this experiment, following Matthews et al. (2018), we additionally use the kernel two-sample test based on the MMD (Gretton et al., 2012) as an alternative assesment
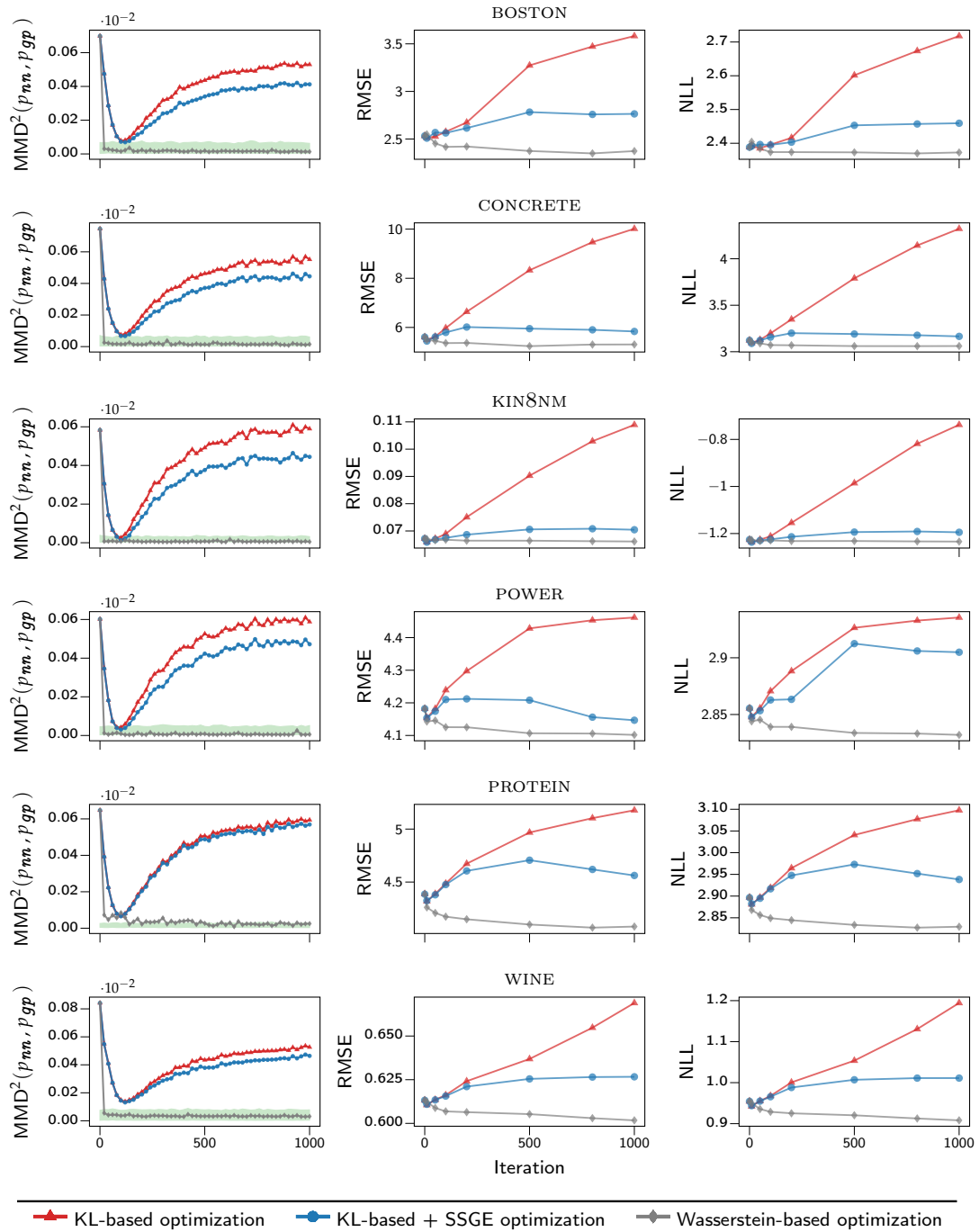
Figure 3.5: Comparison between KL-based and Wasserstein-based optimization. The green shaded area is for calibration and denotes the difference between the squared maximum mean discrepancy (MMD) of the target GP to itself and to another GP with a doubled lengthscale.

of the similarity between BNNs and GPs. A detailed description of estimating this discrepancy and experimental settings are available in Appendix A.2.7. As done by Matthews et al. (2018), we use a target GP prior with a characteristic lengthscale of $l = \sqrt{2D}$, where D is the input dimensionality. We monitor the evolution of squared MMD from the target GP prior and performance metrics for the UCI datasets (test negative log-likelihood (NLL) and root mean squared error (RMSE)). The KL-based approaches offer improvements for the first few iterations, before degrading the quality of the approximation despite using the SSGE for estimating the entropy gradient. Our approach, instead, consistently improves the quality of the approximation to the desired prior. In the Appendix A.3.3 we include a complete account on the convergence of Wasserstein distance for all experiments that follow in the next section.

3.6 Experimental Evaluation

We shall now evaluate whether our scheme offers any competitive advantage in comparison to standard choices of priors. This section is organized as follows: we first summarize the baselines considered in our experimental campaign in § 3.6.1. We then investigate the effect of functional priors on classic UCI benchmark datasets for regression in § 3.6.2 and classification in § 3.6.3. Bayesian CNNs are explored in § 3.6.4, where we also study the benefits of functional priors for handling out-of-distribution data. We next compare against some well-established alternatives to determine prior parameters, such as cross-validation and empirical Bayes in § 3.6.5. We then perform experiments on active learning (§ 3.6.6), where having good and calibrated estimates of uncertainty is critical to achieve fast convergence. Finally, we conclude in § 3.6.7 with a non-Bayesian experiment: we explore the effect of functional priors on maximum-a-posteriori (MAP) estimates, demonstrating that our scheme can also be beneficial as a regularization term in a purely optimization-based setting.

3.6.1 Baselines

In the following experiments, we consider two fixed priors: (1) fixed Gaussian (FG) prior, $\mathcal{N}(0, 1)$; (2) fixed hierarchical (FH) prior where the prior variance for each layer is sampled from an Inverse-Gamma distribution, $\Gamma^{-1}(1, 1)$ (Springenberg et al., 2016); and three GP-induced neural network (NN) priors, namely: (3) GP-induced Gaussian (GPi-G) prior, (4) GP-induced hierarchical (GPi-H) prior, and (5) GP-induced normalizing flow (GPi-NF) prior. Since the computational cost of the GPi-NF prior is high, we only consider this prior in some of the regression experiments. For hierarchical priors, we resample the prior variances using a Gibbs step every 100 iterations.

Table 3.1: Glossary of methods used in the experimental campaign. Here, $p(\boldsymbol{f}) = \int p(\boldsymbol{f} \,|\, \boldsymbol{w}) \mathrm{p}(\boldsymbol{w})$ denotes the induced prior over functions; $\Gamma^{-1}(\alpha, \beta)$ denotes the Inverse-Gamma distribution with shape α, and rate β; $\mathcal{NF}(\mathcal{T}_K)$ indicates a normalizing flow distribution constructed from a sequence of K invertible transformations $\mathcal{T}$; $\widehat{\sigma}^2$, and $(\widehat{\alpha}, \widehat{\beta})$ denote the optimized parameters for the GPi-G and GPi-H priors, respectively. $\widehat{\kappa}$ corresponds to optimized kernel parameters, while $\widehat{\sigma}^2_{\mathrm{LA}}$ shows that the parameters are optimized on the Laplace approximation of the marginal likelihood. References are [a] for Wenzel et al. (2020), [b] for Springenberg et al. (2016), [c] for Lakshminarayanan et al. (2017), [d] for Sun et al. (2019) and, finally, [e] for Immer et al. (2021b).

	Priors			Inference	
Name	$p(\sigma^2)$	$p(\boldsymbol{w} \,|\, \sigma^2)$	$p(\boldsymbol{f})$		Ref.
(━) BNN w/ Fixed Gaussian (FG) prior	–	$\mathcal{N}(0, \sigma^2 \boldsymbol{I}) \;\rightarrow$	?	SGHMC	
(━) BNN w/ Fixed Gaussian prior and TS (FG+TS)	–	$\mathcal{N}(0, \sigma^2 \boldsymbol{I}) \;\rightarrow$	?	Temp. SGHMC	[a]
(━) BNN w/ Fixed hierarchical (FH) prior	$\Gamma^{-1}(\alpha, \beta) \rightarrow$	$\mathcal{N}(0, \sigma^2 \boldsymbol{I}) \;\rightarrow$	?	SGHMC + Gibbs	[b]
(━) Deep ensemble	–	?	?	Ensemble	[c]
(━) Functional BNN w/ variational inference (fBNN)	–	–	$\mathcal{GP}(0, \widehat{\kappa})$	VI	[d]
(━) BNN w/ Laplace GGN approximation (LA-GGN)	–	$\mathcal{N}(0, \widehat{\sigma}^2_{\mathrm{LA}} \boldsymbol{I}) \;\rightarrow$	?	Laplace approx.	[e]
(━) BNN w/ GP-induced Gaussian (GPi-G) prior	–	$\mathcal{N}(0, \widehat{\sigma}^2 \boldsymbol{I}) \;\leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC	**[Ours]**
(━) BNN w/ GP-induced hierarchical (GPi-H) prior	$\Gamma^{-1}(\widehat{\alpha}, \widehat{\beta}) \leftarrow$	$\mathcal{N}(0, \sigma^2 \boldsymbol{I}) \;\leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC + Gibbs	**[Ours]**
(━) BNN w/ GP-induced norm. flow (GPi-NF) prior	–	$\mathcal{NF}(\mathcal{T}_K) \;\leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC	**[Ours]**

Considering the aforementioned settings, we compare BNNs against Deep Ensemble (Lakshminarayanan et al., 2017), arguably one of the state-of-the-art approaches for uncertainty estimation in deep learning (Ashukha et al., 2020; Ovadia et al., 2019). This non-Bayesian method combines solutions that maximize the predictive log-likelihood for multiple neural networks trained with different initializations. We employ an ensemble of 5 neural networks in all experiments. Following Lakshminarayanan et al. (2017), we use Adam optimizer (Kingma and Ba, 2015) to train the individual networks. Furthermore, we compare the results obtained by sampling from the posterior obtained with GP-induced priors against "tempered" posteriors (Wenzel et al., 2020) that use the FG prior and temperature scaling; we refer to this approach as FG+TS. In our experiments, the weight decay coefficient for Deep Ensemble and the temperature value for the "tempered" posterior are tuned by cross-validation.

Additionally, we benchmark our approach against the state-of-the-art variational inference method in function space (Sun et al., 2019), referred to as fBNN. We also evaluate our methodology of imposing priors against an empirical Bayes approach (Immer et al., 2021b), namely LA-GGN, which optimizes the prior based on an approximation of the marginal likelihood by means of the Laplace and GGN approximations. See the Appendix A.2 for implementation details and more detailed hyperparameter settings. Table 3.1 presents an overview of the methods considered in the experiments.
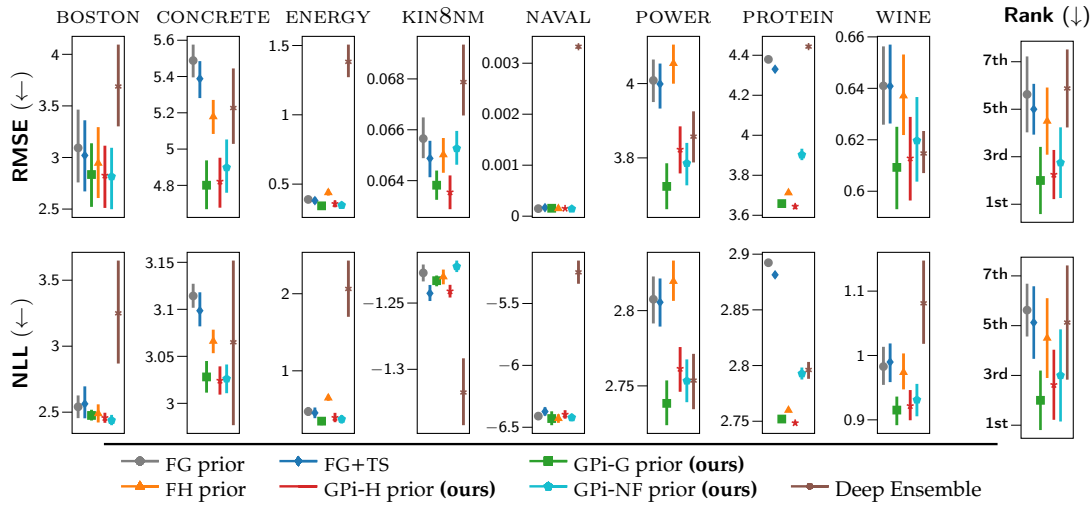
Figure 3.6: UCI regression benchmark results. The dots and error bars represent the means and standard errors over the test splits, respectively. Average ranks are computed across datasets.

3.6.2 UCI regression benchmark

We start our evaluation on real-world data by using regression datasets from the UCI collection (Dua and Graff, 2017). Each dataset is randomly split into training and test sets, comprising of 90% and 10% of the data, respectively. This splitting process is repeated 10 times except for the PROTEIN dataset, which uses 5 splits. We use a two-layer MLP with tanh activation function, containing 100 units for smaller datasets and 200 units for the PROTEIN dataset. We use a mini-batch size of 32 for both the SGHMC sampler and the Adam optimizer for Deep Ensemble.

We map a target hierarchical-GP prior to GPi-G, GPi-H, and GPi-NF priors using our proposed Wasserstein optimization scheme with a mini-batch size of $N_s = 128$. We use an RBF kernel with dimension-wise lengthscales, also known as automatic relevance determination (ARD) (MacKay, 1996b). Hyper-priors LogNormal($\log \sqrt{2D}, 1$) and LogNormal($0.1, 1$) are placed on the lengthscales l and the variance α^2, respectively. Here, D is the number of input dimensions. We use measurement sets having a size of $N_{\mathcal{M}} = 100$, which include 70% random training samples and 30% uniformly random points from the input domain.

Fig. 3.6 illustrates the average test NLL and RMSE. On the majority of datasets, our GP-induced priors provide the best results. They significantly outperform Deep Ensemble in terms of both RMSE and NLL, a metric that considers both uncertainty and accuracy. We notice that tempering the posterior delivers only small improvements for the FG prior. Instead, by using the GPi-G prior, the true posterior's predictive performance is improved significantly.
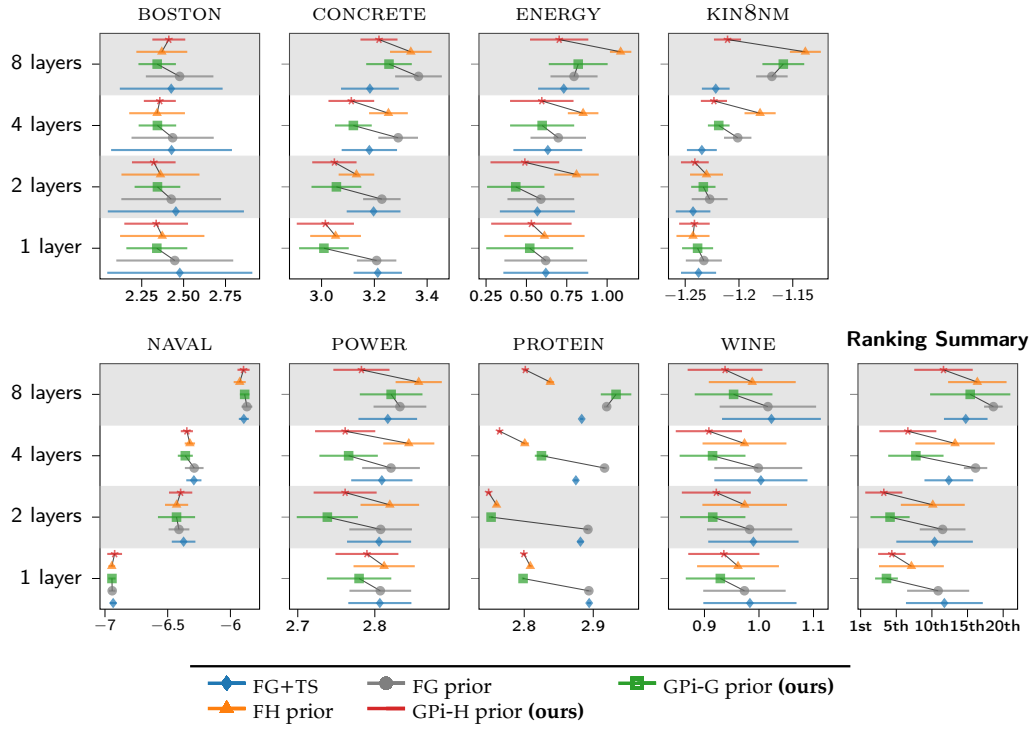
Figure 3.7: Ablation study on the test NLL based on the UCI regression benchmark for different number of hidden layers of MLP. Error bars represent one standard deviation. We connect the fixed and GP-induced priors with a thin black line as an aid for easier comparison. Further to the left is better.

Ablation study on the model capacity.   We further investigate the relation of the model capacity to the prior effect. Fig. 3.7 illustrates the test NLL on the UCI regression benchmark for different number of MLP hidden layers. For most datasets, the GP-induced priors consistently outperform other approaches for all MLP depths. Remarkably, we observe that when increasing the model's capacity, the effect of temperature scaling becomes more prominent. We argue that a tempered posterior is only beneficial for over-parameterized models, as evidenced by pathologically poor results for one-layer MLPs. We further elaborate on this hypothesis in § 3.6.4 with much more complex models such as CNNs.

3.6.3   UCI classification benchmark

Next, we consider 7 classification datasets from the UCI repository. The chosen datasets have a wide variety in size, number of dimensions, and classes. We use a two-layer MLP with tanh activation function, containing 100 units for small datasets (EEG, HTRU2, LETTER, and MAGIC), 200 units for large datasets (MINIBOO, DRIVE, and MOCAP). The experiments have been repeated for 10 random training/test splits. We use a mini-batch size of 64 examples for the SGHMC sampler and the Adam optimizer. Similarly to the previous experiment, we use a target hierarchical-GP prior
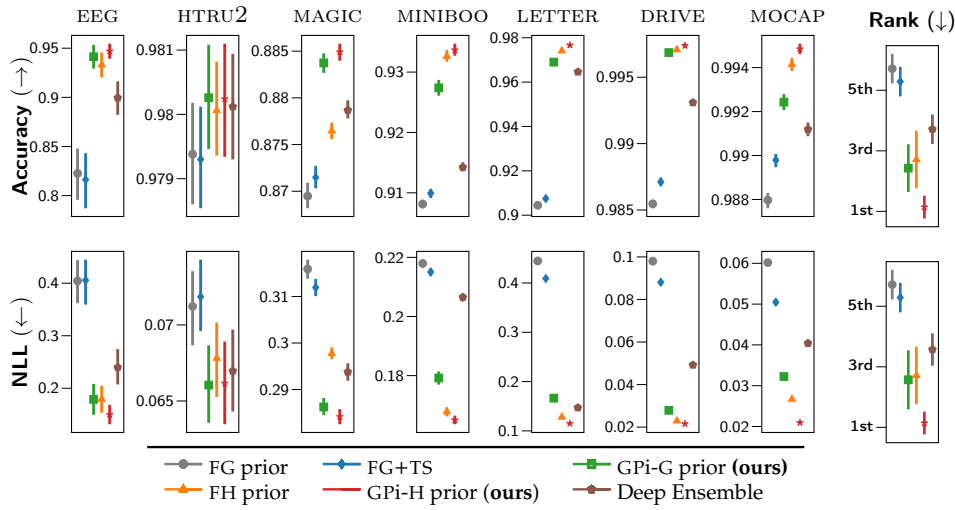
Figure 3.8: UCI classification benchmark results. The dots and error bars represent the means and standard errors over the test splits, respectively. Average ranks are computed across datasets.

with hyper-priors for the lengthscales and the variance are $\mathrm{LogNormal}(\log\sqrt{2D}, 1)$ and $\mathrm{LogNormal}(\log 8, 0.3)$, respectively. We use the same setup of the measurement set as used in the UCI regression experiments.

Fig. 3.8 reports the average test accuracy and NLL. The results for Deep Ensemble are significantly better than those of the FG prior with and without using temperature scaling. Similarly to the previous experiment, the GPi-G prior outranks Deep Ensemble and is comparable with the FH prior, which is a more flexible prior. Once again, the GPi-H prior consistently outperforms other priors across all datasets.

3.6.4 Bayesian convolutional neural networks for image classification

We proceed with the analysis of convolutional neural networks: we first analyze the kind of class priors that are induced by our strategy, and then we move to the CIFAR10 experiment where we also discuss the cases of reduced and corrupted training data.

Analysis on the prior class labels.   As already mentioned, FG is the most popular prior for Bayesian CNNs (Wenzel et al., 2020; Zhang et al., 2020; Heek and Kalchbrenner, 2019). This prior over parameters combined with a structured function form, such as a convolutional neural network, induces a structured prior distribution over functions. However, as shown by Wenzel et al. (2020), this is a poor functional prior because the sample function strongly favors a single class over the entire dataset.

We reproduce this finding for the LENET5 model (Lecun et al., 1998) on the MNIST dataset. In particular, we draw three parameter samples from the FG prior, and we observe the induced prior over classes for each parameter sample (see the three

rightmost columns of Fig. 3.9b). We also visualize the average prior distribution obtained from 200 samples of parameters (see the leftmost column of Fig. 3.9b). Although the average prior distribution is fairly uniform, the distribution for each sample of parameters is highly concentrated on a single class. As illustrated in Fig. 3.9d, the same problem happens for the FH prior.

This pathology does not manifest in our approach, as a more sensible functional prior is imposed. In particular, we choose a target GP prior with an RBF kernel having amplitude $\alpha = 1$, such that the prior distribution for each GP function sample is close to the uniform class distribution (Fig. 3.9a), and a lengthscale $l = 256$. We then map this GP prior to GPi-G and GPi-H priors by using our Wasserstein optimization scheme. Fig. 3.9c and Fig. 3.9e demonstrate that the resulting functional priors are more reasonable as evidenced by the uniformly-distributed prior distributions over all classes.

Deep convolutional neural networks on CIFAR10   We continue the experimental campaign on the CIFAR10 benchmark (Krizhevsky and Hinton, 2009) with a number of popular CNN architectures: LENET5 (Lecun et al., 1998), VGG16 (Simonyan and Zisserman, 2014) and PRERESNET20 (He et al., 2016). Regarding posterior inference with SGHMC, after a burn-in phase of 10,000 iterations, we collect 200 samples with 10,000 simulation steps in between. For a fair comparison, we do not use techniques such as data augmentation or adversarial examples in any of the experiments. Regarding the target hierarchical-GP prior, we place a hyper-prior LogNormal($\log 8, 0.3$) for variance, whereas the hyper-prior for length-scale is LogNormal($\log 512, 0.3$). We use a mini-batch size of $N_s = 128$ and $N_{\mathcal{M}} = 32$ measurement points sampled from the empirical distribution of the training data regarding prior optimization.

Table 3.2 summarizes the results on the CIFAR10 test set with respect to accuracy and NLL. These results demonstrate the effectiveness of the GP-induced priors, as evidenced by the improvements in predictive performance when using GPi-G and GPi-H priors compared to using FG and FH priors, respectively. Noticeably, the GPi-H prior offers the best performance with 76.51%, 87.03%, and 88.20% predictive accuracy on LENET5, VGG16, and PRERESNET20 respectively. We observe that for complex models (e.g., PRERESNET20 and VGG16), FG prior's results are improved by a large margin by tempering the posterior. This is in line with the results showed by Wenzel et al. (2020). By contrast, in the case of LENET5, the predictive performance dramatically degraded when using temperature scaling. In addition to the results in § 3.6.2, this observation supports our conjecture that a "tempered" posterior is only useful for over-parameterized models. Instead, by using GP-induced priors, we consistently obtain the best results in most cases.
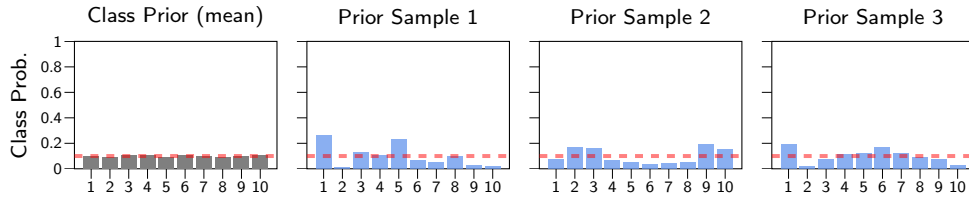
(a) Target GP prior.
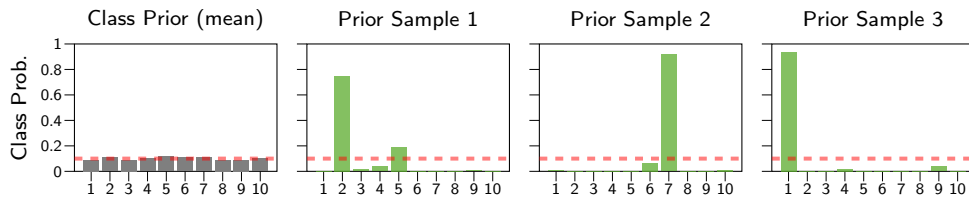
(b) BNN - FG prior.
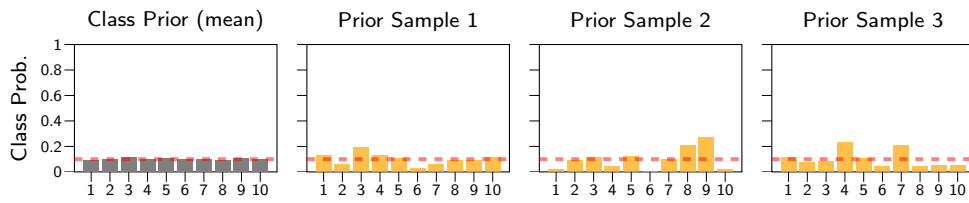
(c) BNN - GPi-G prior.
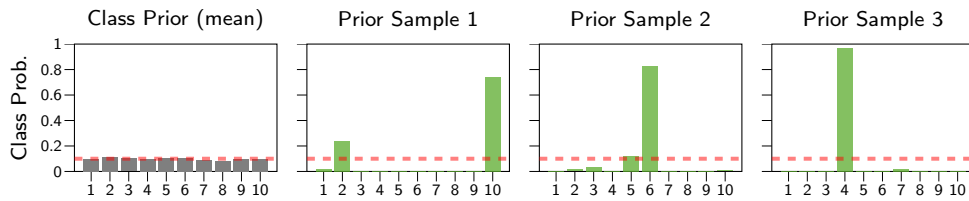
(d) BNN - FH prior.
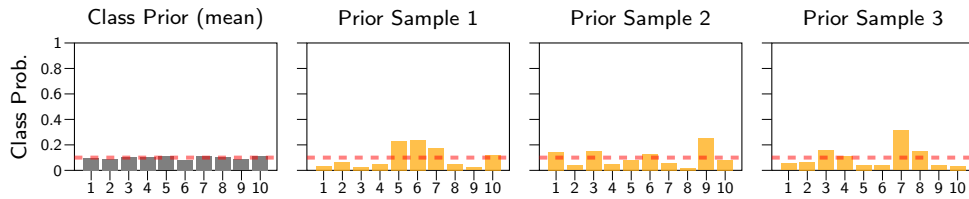
(e) BNN - GPi-H prior.

Figure 3.9: Average class probabilities over all training data of MNIST for three prior samples of parameters (three right columns), and prior distribution averaged over 200 samples of parameters (leftmost column). The GPi-G and GPi-H priors were obtained by mapping from a target GP prior (top row) using our proposed method.

Table 3.2: Results for different convolutional neural networks on the CIFAR10 dataset (errors are ± 1 standard error computed over 4 running times).

Architecture	Method	Accuracy - % ($\uparrow$)	NLL ($\downarrow$)
LENET5	Deep Ensemble	71.13 ± 0.10	0.8548 ± 0.0010
	FG prior	74.65 ± 0.25	0.7482 ± 0.0025
	FG+TS	74.08 ± 0.24	0.7558 ± 0.0024
	GPi-G prior (**ours**)	75.15 ± 0.24	0.7360 ± 0.0024
	FH prior	75.22 ± 0.40	0.7209 ± 0.0040
	GPi-H prior (**ours**)	$\mathbf{76.51} \pm \mathbf{0.21}$	$\mathbf{0.6952} \pm \mathbf{0.0021}$
PRERESNET20	Deep Ensemble	87.77 ± 0.03	0.3927 ± 0.0003
	FG prior	85.34 ± 0.13	0.4975 ± 0.0013
	FG+TS	87.70 ± 0.11	0.3956 ± 0.0011
	GPi-G prior (**ours**)	86.86 ± 0.27	0.4286 ± 0.0027
	FH prior	87.26 ± 0.09	0.4086 ± 0.0009
	GPi-H prior (**ours**)	$\mathbf{88.20} \pm \mathbf{0.07}$	$\mathbf{0.3808} \pm \mathbf{0.0007}$
VGG16	Deep Ensemble	81.96 ± 0.33	0.7759 ± 0.0033
	FG prior	81.47 ± 0.33	0.5808 ± 0.0033
	FG+TS	82.25 ± 0.15	0.5398 ± 0.0015
	GPi-G prior (**ours**)	83.34 ± 0.53	0.5176 ± 0.0053
	FH prior	86.03 ± 0.20	0.4345 ± 0.0020
	GPi-H prior (**ours**)	$\mathbf{87.03} \pm \mathbf{0.07}$	$\mathbf{0.4127} \pm \mathbf{0.0007}$

Robustness to covariate shift. Covariate shift describes a situation where the test input data has a different distribution than the training data. In this experiment, we evaluate the behavior of GP-induced priors under such circumstances. We also compare to Deep Ensemble, which is well-known for its robustness properties under covariate shift (Ovadia et al., 2019).

Using the protocol from Ovadia et al. (2019), we train models on CIFAR10 and then evaluate on the CIFAR10C dataset, which is generated by applying 16 different corruptions with 5 levels of intensity for each corruption (Hendrycks and Dietterich, 2019). Our results are summarized in Fig. 3.10 (additional results are available in the appendix). For PRERESNET20, there is a clear improvement in robustness to distribution shift by using the GP-induced priors. Remarkably, the GPi-H prior performs best and outperforms Deep Ensemble at all corruption levels in terms of accuracy and NLL. Meanwhile, the NLL results of SGHMC are significantly better than those of Deep Ensemble. We also notice that the GPi-G prior offers considerable improvements in predictive performance compared to the FG prior.

Performance on small training data For small and high-dimensional datasets, the importance of choosing a sensible prior is more prominent because the prior's influence on the posterior is not overwhelmed by the likelihood. To compare priors in this scenario, we use subsets of the CIFAR10 dataset with different training set sizes, keeping the classes balanced. Fig. 3.11 shows the accuracy and NLL on the test
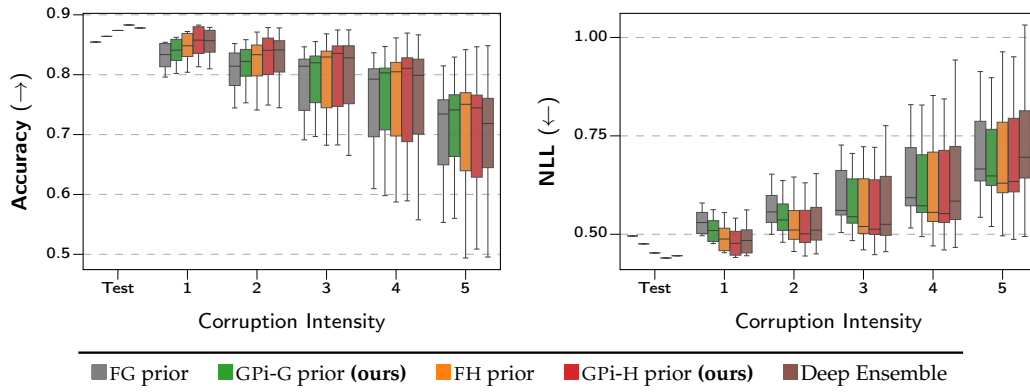
Figure 3.10: Accuracy and NLL on CIFAR10C at varying corruption severities. Here, we use the PRERESNET20 architecture. For each method, we show the mean on the test set and the results on each level of corruption with a box plot. Boxes show the quartiles of performance over each corruption while the error bars indicate the minimum and maximum.
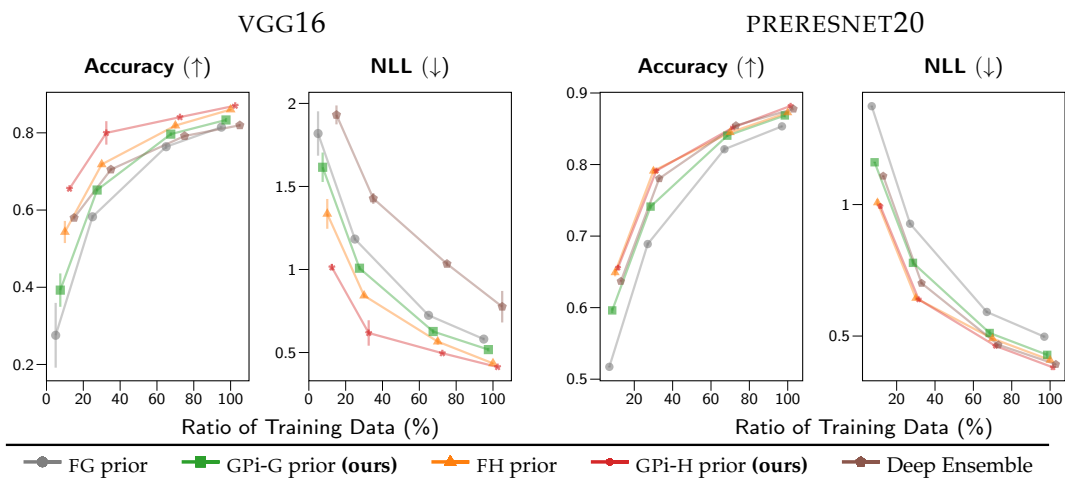
Figure 3.11: Accuracy and negative log-likelihood on CIFAR10 at varying the training set's size. The bars indicate one standard error.

set. The FG prior yields poor predictive performance in small training data cases. Indeed, we observe that the GPi-G prior performs much better than the FG prior in all cases. Besides, the GPi-H prior offers superior predictive performance across all proportions of training/test data. These results again demonstrate the usefulness of the GP-induced priors for the predictive performance of BNNs.

Entropy analysis on out-of-distribution data.   Next, we demonstrate with another experiment that the proposed GP-based priors offer superior predictive uncertainties compared to competing approaches by considering the task of uncertainty estimation on out-of-distribution samples (Lakshminarayanan et al., 2017). Our choice of the target functional prior is reasonable for this type of task because, ideally, the predictive distribution should be uniform over the out-of-distribution classes–which results in maximum entropy–rather than being concentrated on a particular class. Following the experimental protocol from Louizos and Welling (2017), we train LENET5 on the standard MNIST training set, and estimate the entropy of the predictive distribution on both MNIST and NOT-MNIST datasets[2]. The images in the NOT-MNIST dataset have the same size as the MNIST, but represent other characters. For posterior inference with SGHMC, after a burn-in phase of 10,000 iterations, we draw 100 samples with 10,000 iterations in between. We also consider the "tempered" posterior with the FG prior and Deep Ensemble as competitors.

Fig. 3.12 shows the empirical CDF for the entropy of the predictive distributions on MNIST and NOT-MNIST. For the NOT-MNIST dataset, the curves that are closer to the bottom right are preferable, as they indicate that the probability of predicting classes with a high confidence prediction is low. In contrast, the curves closer to the top left are better for the MNIST dataset. As expected, we observe that the uncertainty estimates on out-of-distribution data for the GP-induced priors are better than those obtained by the fixed priors. In line with the results from Louizos and Welling (2017), Deep Ensemble tends to produce overconfident predictions on both in-distribution and out-of-distribution predictions. For tempered posteriors, we can interpret decreasing the temperature as artificially sharpening the posterior by overcounting the training data. This is the reason why a tempered posterior tends to be overconfident.

3.6.5   Optimizing priors with data: cross-validation and empirical Bayes

Although we advocate for functional priors over BNNs, we acknowledge that a prior of this kind is essentially heuristic. A potentially more useful prior might be discovered by traditional means such as cross-validation (CV) or by running an empirical Bayes procedure (a.k.a. type-II maximum likelihood), which maximizes

---

[2]NOT-MNIST   dataset   is   available   at   http://yaroslavvb.blogspot.fr/2011/09/notmnist-dataset.html.
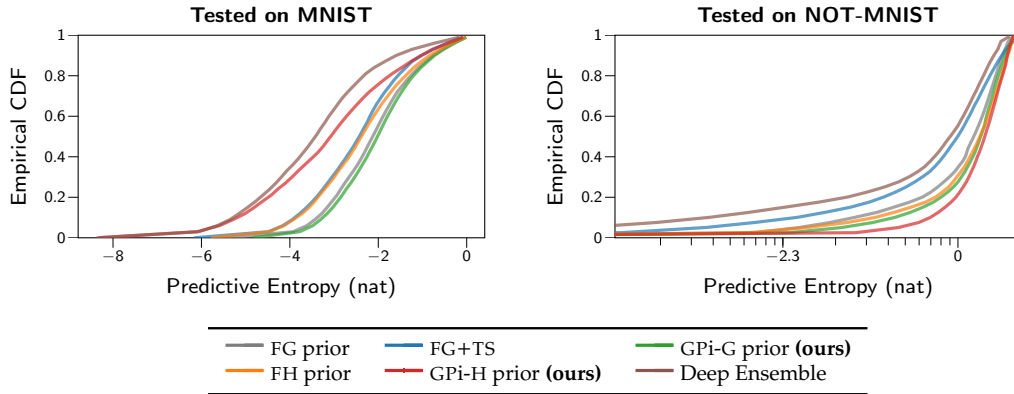
Figure 3.12: Cumulative distribution function plot of predictive entropies when the models trained on MNIST are tested on MNIST (left, the higher the better) and NOT-MNIST (right, the lower the better).
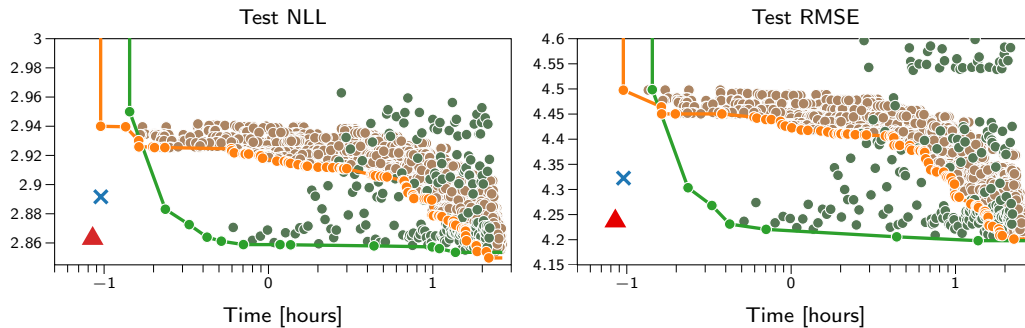
Figure 3.13: A timing comparison between imposing functional prior and cross-validation with either grid-search (▬) or Bayesian optimization (▬). In the plots, each ● corresponds to a run of a single configuration, while ●▬● highlights the Pareto front of the cross validation procedure. The figure also reports the $\mathcal{N}(0,1)$ prior as ✕, while ▲ is our proposal of using functional prior (GPi-G).

the marginal likelihood $p(\mathcal{D}; \boldsymbol{\psi}) = \int p(\mathcal{D} \mid \boldsymbol{w}) p(\boldsymbol{w}; \boldsymbol{\psi}) d\boldsymbol{w}$ w.r.t. the prior parameters. However, these methods present significant challenges: (i) for CV, the number of hyperparameter combinations that needs to be explored becomes exponentially large as the complexity of the neural network grows, or as the exploration grid becomes more fine-grained. (ii) for empirical Bayes, we need to compute the exact marginal likelihood, which is always intractable for BNNs, thus requiring additional approximations like variational inference (VI) or the Laplace approximation. We next demonstrate these issues empirically.

Cross-Validation.   We consider a simple case of a BNN with one hidden layer only; by adopting the simple parameterization of § 3.4.2, we shall have four parameters to optimize in total (i.e. the weight and bias variances of the hidden and the output layer). In Fig. 3.13, we demonstrate how our scheme behaves in comparison with a CV strategy featuring a grid size of 9 (for a total of 6561 configurations). To get results for the cross-validation procedure and to massively exploit all possible parallelization

opportunities, we allocated a cloud platform with 16 server-grade machines, for a total of 512 computing cores and 64 maximum parallel jobs. This required a bit more than one day, although the total CPU time approached 3 months. While grid-based routines are widely adopted by practitioners for cross-validation, we acknowledge that there are more efficient alternatives. To this extent, we also include Bayesian optimization (Močkus, 1975; Snoek et al., 2012; Nogueira, 2014), a classical method for black-box optimization which uses a Gaussian process as the surrogate function to be maximized (or minimized). As expected, CV indeed found marginally better configurations, but the amount of resources and time needed, even for such a small model, is orders of magnitude larger than what required by our scheme, making this procedure computationally infeasible for larger models, like CNNs. To put things into perspective, our Wasserstein-based functional prior could be run on a 4-core laptop in a reasonable time.

Empirical Bayes.   We now discuss state-of-the-art methods for empirical Bayes when using variational inference and Laplace approximation. We demonstrate that our proposal outperforms these approaches through an extensive series of experiments on UCI regression and CIFAR10 benchmarks. More specifically, we evaluate our approach using SGHMC with the GPi-G prior and compare it against fBNN, a method of functional variational inference (Sun et al., 2019) which imposes a GP prior directly over the space of functions of BNNs. The hyperparameters of the GP prior for fBNN are obtained by maximizing the marginal likelihood. As in the original proposal of fBNN, we only consider this baseline in experiments on regression datasets. We consider a comparison with the Gaussian prior obtained by the empirical Bayes approach of Immer et al. (2021b). This method uses the Laplace and GGN methods to approximate the marginal likelihood, and referred to LA-GGN. Here, we use the same parameterization as for the GPi-G prior where we optimize the variance of the Gaussian prior on the weights and biases of each layer individually. The resulting prior obtained by this approach is denoted as LA-MargLik. The details of experimental settings are described in Appendix A.2.8. In Fig. 3.14, we show the results of one-layer MLP with tanh activation function on the UCI regression datasets. Our approach using the SGHMC sampler with the GPi-G prior outperforms the baselines of functional inference on most datasets and across metrics. Moreover, we find that our GPi-G prior is consistently better than the LA-MargLik prior when used together with SGHMC for inference, denoted as "LA-MargLik + SGHMC". These observations are further highlighted in the experiments with Bayesian CNNs on the CIFAR10 benchmark. As can be seen from Fig. 3.15, thanks to using a good prior and a powerful sampling scheme for inference, our proposal consistently achieves the best results in all cases. More comprehensive analyses with Bayesian CNNs are available in Appendix A.3.4.

From a more philosophical point of view, it is worth noting that cross-validating prior
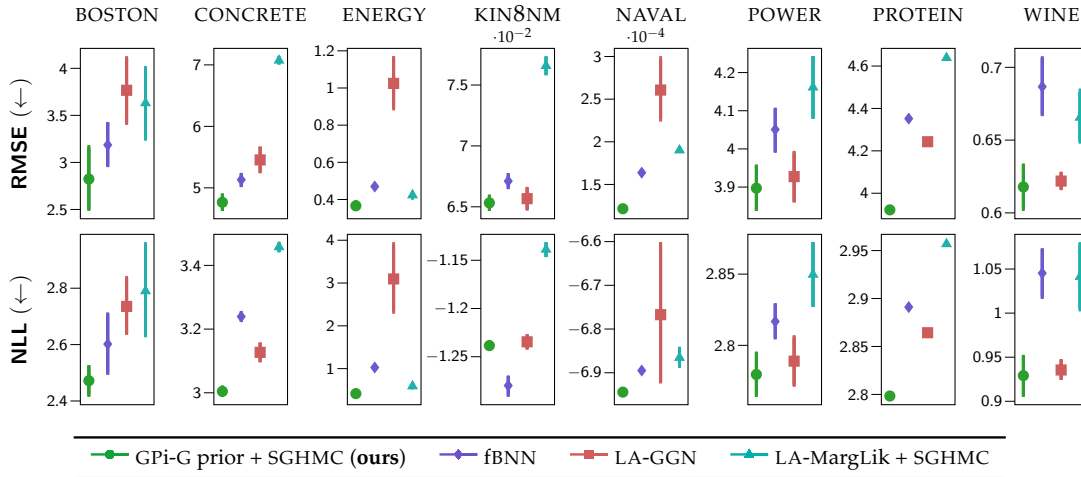
Figure 3.14: Comparison with empirical Bayes and functional inference approaches on the UCI regression datasets. The dots and error bars represent the means and standard errors over the test splits, respectively.
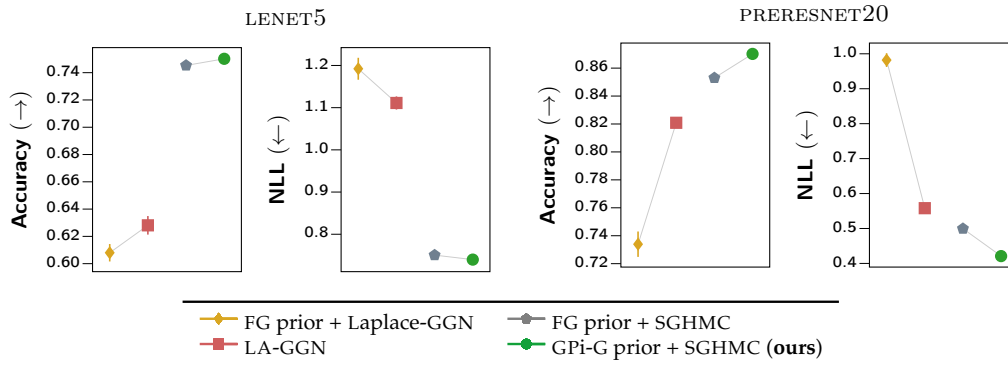
Figure 3.15: Comparison with empirical Bayes and functional inference approaches on the CIFAR10 dataset. A thin black line is used as an aid to see the performance improvement by using the optimized prior instead of the fixed prior (the standard Gaussian prior). The error bars indicate one standard deviation which is estimated by running 4 different random initializations.

parameters, though perfectly legitimate, is not compatible with the classical Bayesian principles. On the other hand, empirical Bayes is widely accepted as a framework to determine prior parameters in terms of a Bayesian context; nevertheless it still has to rely on part of the data. In contrast to both of these alternatives, our procedure returns an appropriate prior *without* having taken any data into consideration.

3.6.6 Active learning

We next perform a series of experiments within an active learning scenario (Settles, 2009). In this type of task, it is crucial to produce accurate estimates of uncertainty to obtain good performance. We use the same network architectures and datasets as used in the UCI regression benchmark. We adopt the experimental setting of Skafte

Data set	FG prior	GPi-G prior (ours)	FH prior	GPi-H prior (ours)
BOSTON	3.199 ± 0.390	2.999 ± 0.382	3.030 ± 0.365	$\mathbf{2.990} \pm 0.384$
CONCRETE	5.488 ± 0.218	5.036 ± 0.239	5.154 ± 0.251	$\mathbf{4.919} \pm 0.299$
ENERGY	$\mathbf{0.442} \pm 0.041$	0.461 ± 0.032	0.458 ± 0.050	0.446 ± 0.025
KIN8NM	0.069 ± 0.001	0.067 ± 0.001	0.068 ± 0.001	$\mathbf{0.066} \pm 0.001$
NAVAL	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
POWER	4.015 ± 0.059	$\mathbf{3.834} \pm 0.068$	4.172 ± 0.051	3.851 ± 0.066
PROTEIN	4.429 ± 0.016	4.036 ± 0.014	4.080 ± 0.018	$\mathbf{3.993} \pm 0.014$
WINE	0.634 ± 0.013	0.617 ± 0.008	0.625 ± 0.010	$\mathbf{0.612} \pm 0.011$

Table 3.3: Results for the active learning scenario. Average test RMSE evaluated at the last step of the iterative data gathering procedure.
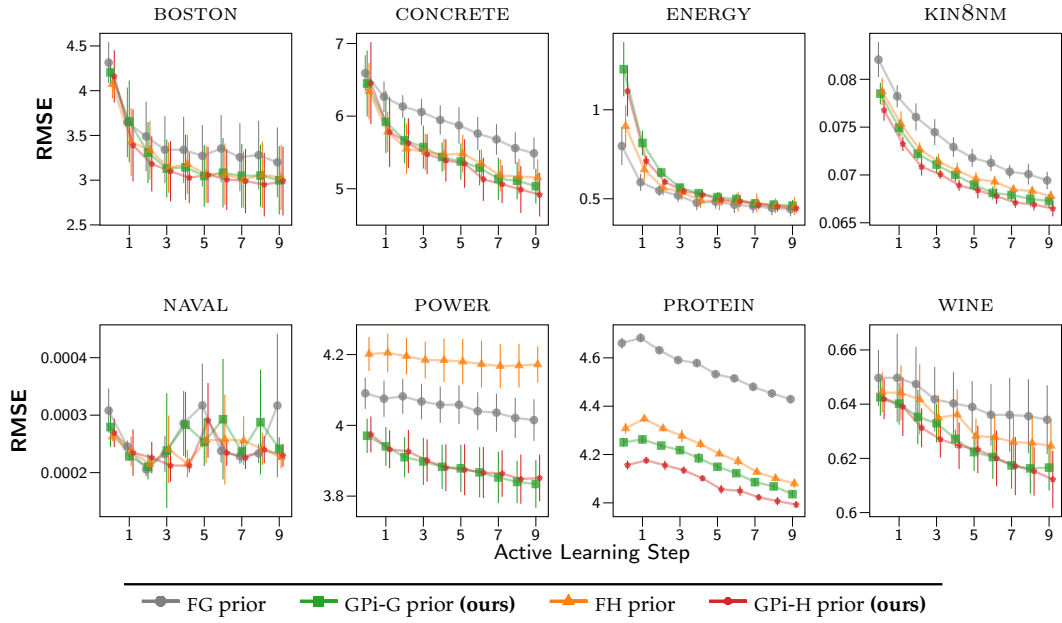
Figure 3.16: The progressions of average test RMSE and standard errors in the active learning experiment.

et al. (2019), where each dataset is split into 20% train, 60% pool, and 20% test sets. For each active learning step, we first train models and then estimate uncertainty for all data instances in the pool set. To actively collect data from the pool set, we follow the information-based approach described by MacKay (1992). More specifically, we choose the n data points with the highest posterior entropy and add them to the training set. Under the assumption of i.i.d. Gaussian noise, this is equivalent to choosing the unlabeled examples with the largest predictive variance (Houlsby et al., 2012). We define $n = 5\%$ of the initial size of the pool set. We use 10 active-learning steps and repeat each experiment 5 times per dataset on random training-test splits to compute standard errors.

Fig. 3.16 shows the progressions of average test RMSE during the data collection process. We observe that, on most datasets (CONCRETE, KIN8NM, POWER, PRO-TEIN, and WINE), the GPi-G and GPi-H priors achieve faster learning than FG and FH priors, respectively. For the other datasets, FH prior is on par with GPi-H, while
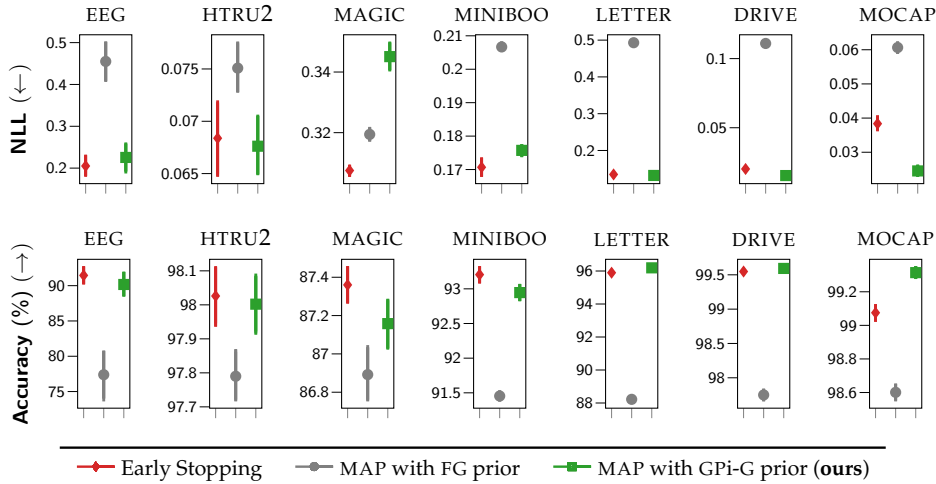
Figure 3.17: Comparison between early stopping and MAP optimization with the FG and GPi-G priors on the UCI classification datasets.

FG consistently results in the worse performance, except in one case (ENERGY). We also report the average test RMSE at the last step in Table 3.3. These results show that the GPi-H prior performs best, while the GPi-G prior outperforms the FG prior in most cases.

3.6.7 Maximum-a-posteriori (MAP) estimation with GP-induced prior

In the last experiment, we demonstrate that the GPi-G prior is useful not only for Bayesian inference but also for MAP estimation. We investigate the impact of the GPi-G priors obtained in the previous experiments and the FG prior on the performance of MAP estimation. We additionally compare to *early stopping*, which is a popular regularization method for neural networks. Compared to early stopping, MAP is a more principled regularization method even though early stopping should exhibit similar behavior to MAP regularization in some cases, such as those involving a quadratic error function (Yao et al., 2007). Regarding the experimental setup, we train all networks for 150 epochs using the Adam optimizer with a fixed learning rate 0.01. For early stopping, we stop training as soon as there is no improvement for 10 consecutive epochs on validation NLL for classification tasks. For the UCI classification datasets, MAP estimation for the GPi-G prior is comparable with early stopping and significantly outperforms the one for the FG prior (Fig. 3.17). For the CNNs, as shown in Fig. 3.18, we observe that the MAP estimations outperform early stopping in most cases. Besides, it is not clear which prior is better. We think this can be attributed to the fact that optimization for very deep nets is non-trivial. As suggested in the literature (Wenzel et al., 2020; Ashukha et al., 2020), one has to use complicated training strategies such as a learning rate scheduler to obtain good performance for deterministic CNNs on high-dimensional data like CIFAR10.
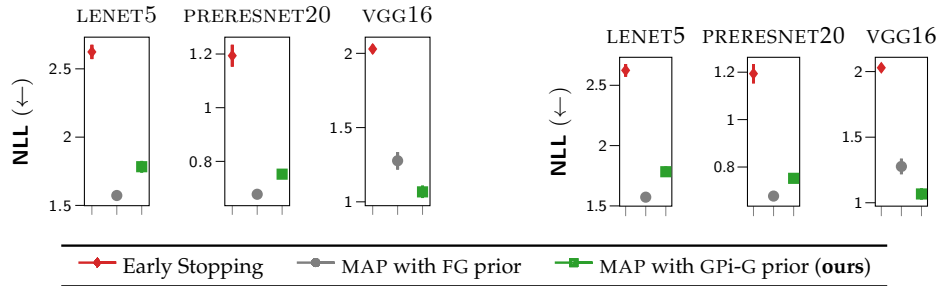
Figure 3.18: Comparison between early stopping and MAP optimization with the FG and GPi-G priors for three different CNN architectures on the CIFAR10 dataset.

3.7 Conclusions

In most machine learning tasks, function estimation is a fundamental and ubiquitous problem. Being able to perform Bayesian inference of neural networks represents a much sought-after objective to equip extremely flexible models with the capability of expressing uncertainty in a sound way (Mackay, 2003; Neal, 1996). Recent advances in MCMC sampling enabling for efficient parameter space exploration, combined with mini-batching (Chen et al., 2014), have turned this long-standing challenge into a concrete possibility. However despite these advances, there have been only few success stories involving the use of Bayesian inference techniques for neural networks (Osawa et al., 2019a; Zhang et al., 2020). We attribute this to the difficulties in specifying sensible priors for thousands/millions of parameters, while being able to understand and control the effect of these choices in the behavior of their output functions (Duvenaud et al., 2014).

The difficulty in reasoning about functional priors for neural networks, made us consider the possibility to enforce these by minimizing their distance to tractable functional priors, effectively optimizing the priors over model parameters so as to reflect these functional specifications. We chose to consider Gaussian processes, as they are a natural and popular choice to construct functional priors, whereby the characteristics of prior functions are determined by the form and parameters of Gaussian process kernel/covariance functions. While previous works attempted this by using the KL divergence between the functional priors (Flam-Shepherd et al., 2017; Flam-Shepherd et al., 2018), the objective proves difficult to work with due to the need to estimate an entropy term based on samples, which is notoriously difficult. In this work, we proposed a novel objective based on the Wasserstein distance, and we showed that this objective offers a tractable and stable way to optimize the priors over model parameters. The attractive property of this objective is that it does not require a closed form for the target functional prior, as long as it is possible to obtain samples from it. We studied different parameterizations of the priors with increasing flexibility, and we showed that more flexibility makes it indeed possible to improve the match to Gaussian process priors, especially when the activation functions are not suitable

to model the target Gaussian processes. It is worth noting that, as far as we know, normalizing flows have never been proposed to model priors for neural networks, and this represents an interesting line of investigation that deserves some attention for future work. We are also planning to investigate our proposal on unsupervised/latent variable models, and study ways to reduce the complexity of the optimization of the Wasserstein distance.

After describing our strategy to optimize the Wasserstein distance, we moved on to show the empirical benefits of choosing sensible priors on a large variety of neural network models, including convolutional neural networks, and modeling tasks such as regression and classification under standard conditions, covariate shift, and active learning. We demonstrated consistent performance improvements over alternatives ways of choosing priors, and we also showed better performance compared to state-of-the-art approximate methods in Bayesian deep learning. In all, this work confirms the hypothesis that choosing sensible priors for deep models matters, and it offers a practical way to do so.

Chapter 4

Model Selection for Bayesian Autoencoders

In this chapter, we develop a novel method for carrying out model selection for Bayesian autoencoders (BAEs) by means of prior hyperparameter optimization. Inspired by the common practice of type-II maximum likelihood optimization and its equivalence to Kullback-Leibler divergence minimization, we propose to optimize the distributional sliced-Wasserstein distance (DSWD) between the output of the autoencoder and the empirical data distribution. The advantages of this formulation are that we can estimate the DSWD based on samples and handle high-dimensional problems. We carry out posterior estimation of the BAE parameters via stochastic gradient Hamiltonian Monte Carlo and turn our BAE into a generative model by fitting a flexible Dirichlet mixture model in the latent space. Consequently, we obtain a powerful alternative to variational autoencoders, which are the preferred choice in modern applications of autoencoders for representation learning with uncertainty. We evaluate our approach qualitatively and quantitatively using a vast experimental campaign on a number of unsupervised learning tasks and show that, in small-data regimes where priors matter, our approach provides state-of-the-art results, outperforming multiple competitive baselines.

4.1   Introduction

The problem of learning useful representations of data that facilitate the solution of downstream tasks such as clustering, generative modeling and classification, is at the crux of the success of many machine learning applications (see, e.g., Bengio et al., 2013b, and references therein). From a plethora of potential solutions to this problem, unsupervised approaches based on autoencoders (Cottrell et al., 1989) are particularly appealing as, by definition, they do not require label information and have proved

effective in tasks such as dimensionality reduction and information retrieval (Hinton and Salakhutdinov, 2006).

Autoencoders are neural network models composed of two parts, usually referred to as the encoder and the decoder. The encoder maps input data to a set of lower-dimensional latent variables. The decoder maps the latent variables back to the observations. The bottleneck introduced by the low-dimensional latent space is what characterizes the compression and representation learning capabilities of autoencoders. It is not surprising that these models have connections with principal component analysis (Baldi and Hornik, 1989), factor analysis and density networks (MacKay and Gibbs, 1999), and latent variable models (Lawrence, 2005).

In applications where quantification of uncertainty is a primary requirement or where data is scarce, it is important to carry out a Bayesian treatment of these models by specifying a prior distribution over their parameters, i.e., the weights of the encoder/decoder. However, estimating the posterior distribution over the parameters of these models, which we refer to as Bayesian autoencoders (BAEs), is generally intractable and requires approximations. Furthermore, the need to specify priors for a large number of parameters, coupled with the fact that autoencoders are not generative models, has motivated the development of variational autoencoders (VAEs) as an alternative that can overcome these limitations (Kingma and Welling, 2014). Indeed, VAEs have found tremendous success and have become one of the preferred methods in modern machine-learning applications (see, e.g., Kingma and Welling, 2019, and references therein).

To recap, three potential limitations of BAEs hinder their widespread applicability in order to achieve a similar or superior adoption to their variational counterpart: (i) lack of generative modeling capabilities; (ii) intractability of inference and (iii) difficulty of setting sensible priors over their parameters. In this work we revisit BAEs and deal with these limitations in a principled way. In particular, we address the first limitation in (i) by employing density estimation in the latent space. Furthermore, we deal with the second limitation in (ii) by exploiting recent advances in Markov chain Monte Carlo (MCMC) and, in particular, stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014). Finally, we believe that the third limitation (iii), which we refer to as the difficulty of carrying out *model selection*, requires a more detailed treatment because choosing sensible priors for Bayesian neural networks is an extremely difficult problem, and this is the main focus of this work.

Contributions. Specifically, in this work we provide a novel, practical, and elegant way of performing model selection for BAEs, which allows us to revisit these models for applications where VAEs are currently the primary choice. We start by considering the common practice of estimating prior (hyper-)parameters via type-II maximum likelihood, which is equivalent to minimizing the Kullback-Leibler (KL) between the

distribution induced by the BAE and the data generating distribution. Because of the intractability of this objective and the difficulty to estimate it through samples, we resort to an alternative formulation where we replace the KL with the distributional sliced-Wasserstein distance (DSWD) (Nguyen et al., 2021) between these two distributions. The advantages of this formulation are that we can estimate the DSWD based on samples and, thanks to the slicing, we can handle large dimensional problems. Once BAE hyperparameters are optimized, we estimate the posterior distribution over the BAE parameters via SGHMC (Chen et al., 2014), which is a powerful sampler that operates on mini-batches and has proven effective for Bayesian deep/convolutional networks (Tran et al., 2022; Zhang et al., 2020; Izmailov et al., 2021b). Furthermore, we turn our BAE into a generative model by fitting a flexible mixture model in the latent space, namely the Dirichlet process mixture model (DPMM). We evaluate our approach qualitatively and quantitatively using a vast experimental campaign on a number of unsupervised learning tasks, with particular emphasis on the challenging task of generative modeling when the number of observations is small.

4.2 Related work

VAEs provide a theoretically-grounded and popular framework for representation learning and deep generative modeling. However, training VAEs poses considerable practical and theoretical challenges yet to be solved. In practice, the learned aggregated posterior distribution of the encoder rarely matches the latent prior, and this hurts the quality of generated samples. Several methods have been proposed to deal with this problem by using a more expressive form of priors on the latent space (Nalisnick and Smyth, 2017; Chen et al., 2017; Tomczak and Welling, 2018; Bauer and Mnih, 2019). Similar to our work, there is a line of research that employs a form of ex-post density estimation on the learned latent space (Dai and Wipf, 2019; Böhm and Seljak, 2020; Ghosh et al., 2020). Wasserstein autoencoders (WAEs) (Tolstikhin et al., 2018) impose a new form of regularization on latent space by reformulating the objective function as an optimal transport (OT) problem. There have been previous attempts to apply the Bayesian approach to VAEs. For example, (Daxberger and Hernández-Lobato, 2019) treats the parameters of VAE's encoder and decoder in a Bayesian manner to deal with out-of-distribution samples. Most of these works focus on imposing prior or regularization on the latent or weight space of autoencoders. In this work, we take a different route, as we aim to impose prior knowledge directly on the output space. Indeed, our work is motivated by recent attempts to rethink prior specification for Bayesian neural networks (BNNs). It is extremely difficult to choose a sensible prior on the parameters of BNNs (see, e.g., Nalisnick, 2018, and references therein) because their effect on the distribution of the induced functions is difficult to characterize. Thus, recent attempts in the literature have turned their attention towards defining priors in the space of functions (Nalisnick et al., 2021b; Yang et al.,

2020; Hafner et al., 2019; Sun et al., 2019). Closest to our work is that of Tran et al. (2022), which matches the functional prior induced by BNNs to Gaussian process (GP) priors by means of the Kantorovich-Rubinstein dual form of the Wasserstein distance. Different from this line of works, we consider a general framework to impose a functional prior for BNNs in an unsupervised learning setting.

4.3   Preliminaries on Bayesian Autoencoders

An autoencoder (AE) is a neural network parameterized by a set of parameters w, which transforms an unlabelled dataset, $x \stackrel{\text{def}}{=} \{x_n\}_{n=1}^{N}$, into a set of reconstructions $f \stackrel{\text{def}}{=} \{f_n\}_{n=1}^{N}$, with $x_n, f_n \in \mathbb{R}^D$. An AE is composed of two parts: (1) an encoder f_{enc} which maps an input sample x_n to a latent code $z_n \in \mathbb{R}^K, K \ll D$; and (2) a decoder f_{dec} which maps the latent code to a reconstructed data point f_n. In short, $f = f(x; w) = (f_{\text{dec}} \circ f_{\text{enc}})(x)$, where we denote $w := \{w_{\text{enc}}, w_{\text{dec}}\}$ the union of parameters of the encoder and decoder.

The Bayesian treatment of AEs dictates that a prior distribution $p(w)$ is placed over all parameters of f_{enc} and f_{dec} and the posterior is inferred using Bayes' rule. For the sake of presentation, we can consider a BAE as a classic BNN for a supervised learning task (Mackay, 1992; Neal, 1996), where we have labels $y \stackrel{\text{def}}{=} \{y_n\}_{n=1}^{N}$ associated with each input point x_n. From this perspective, we can write the posterior on w as follows:

$$p(w \mid y, x) = \frac{p(y \mid w, x)p(w)}{p(y \mid x)}, \tag{4.1}$$

where $p(y \mid w, x)$ is the likelihood defined by the network architecture and the denominator—$p(y \mid x)$—constitutes the marginal likelihood. In order to keep the notation uncluttered, we can drop the explicit dependency on x as input, which leads to:

$$p(w \mid y) = \frac{p(y \mid w)p(w)}{p(y)}, \tag{4.2}$$

We observe that for an AE the labels y are the same as the data points x, meaning that the likelihood is computed in x.

Likelihood model.   Before giving an in-depth treatment on priors for BAEs in the next section, we briefly discuss the likelihood, which can be chosen according to the type of data. Firstly, we assume factorization of the likelihood on the data points, i.e. $p(y \mid w) = \prod_{n=1}^{N} p(y_n \mid w)$. Secondly, given that our experiments mainly focuses on image datasets where pixel values are normalized in the $[0, 1]$ range, we will use the

continuous Bernoulli distribution (Loaiza-Ganem and Cunningham, 2019):

$$p(\boldsymbol{y}_n \,|\, \boldsymbol{w}) = \prod_{i=1}^{D} K(\lambda_i) \lambda_i^{\boldsymbol{y}_{n,i}} (1 - \lambda_i)^{1 - \boldsymbol{y}_{n,i}} := p(\boldsymbol{y}_n \,|\, \boldsymbol{f}_n), \qquad (4.3)$$

where $K(\lambda_i)$ is a properly defined normalization constant (Loaiza-Ganem and Cunningham, 2019) and $\lambda_i = f_i(\boldsymbol{x}_n; \boldsymbol{w}) = f_{n,i} \in [0,1]$ is the i-th output from the BAE given the input $\boldsymbol{x}_n$. We note that, as $\boldsymbol{f}_n$ depends deterministically on $\boldsymbol{w}$, we will use the above expression to refer to both $p(\boldsymbol{y}_n \,|\, \boldsymbol{w})$ and $p(\boldsymbol{y}_n \,|\, \boldsymbol{f}_n)$, where the latter term will be of crucial importance when we define the functional prior induced over the reconstruction f. Finally, we remark that in the Bayesian scheme both the prior and likelihood are modeling choices. In fact, in our experiments we explore an additional likelihood model, namely the *truncated Gaussian*, and we show how the problem of selecting good priors is orthogonal to the choice of the likelihood.

Inference. Although the posterior over the BAE parameters is analytically intractable, it can be approximated by variational methods or using MCMC sampling. Within the large family of approximate Bayesian inference schemes, SGHMC (Chen et al., 2014) allows us to sample from the true posterior by efficiently simulating a Hamiltonian system (Neal, 2011). Unlike more traditional methods, SGHMC can scale up to large datasets by relying on noisy but unbiased estimates of the potential energy function $U(\boldsymbol{w}) = -\sum_{n=1}^{N} \log p(\boldsymbol{y}_n \,|\, \boldsymbol{w}) - \log p(\boldsymbol{w})$. These can be computed by considering a mini-batch of size M of the data and approximating $\sum_{n=1}^{N} \log p(\boldsymbol{y}_n \,|\, \boldsymbol{w}) \approx \frac{N}{M} \sum_{j \in \mathcal{I}_M} \log p(\boldsymbol{y}_j \,|\, \boldsymbol{w})$, where $\mathcal{I}_M$ is a set of M random indices. More details on SGHMC can be found in the Appendix.

Pathologies of standard priors. The choice of the prior is important for the Bayesian treatment of any model as it characterizes the hypothesis space (Mackay, 1992; Murray and Ghahramani, 2005). Specifically for BAEs, one should note that placing a prior on the parameters of the encoder and decoder has an implicit effect on the prior over the network output (i.e. the reconstruction). In addition, the highly nonlinear nature of these models implies that interpreting the effect of the architecture is theoretically intractable and practically challenging. Several works argue that a vague prior such as $\mathcal{N}(0,1)$ is good enough for some tasks and models, like classification with convolutional neural networks (CNNs) (Wilson and Izmailov, 2020).

However, for BAEs this is not enough, as illustrated in Fig. 4.1. The realizations obtained by sampling weights/biases from a $\mathcal{N}(0,1)$ prior indicate that this choice provides poor inductive bias. Meanwhile, by encoding better beliefs via an optimized prior, which is the focus of the next section, the samples can capture main characteristics intrinsic to the data, even when the model is fed with out-of-distribution inputs.
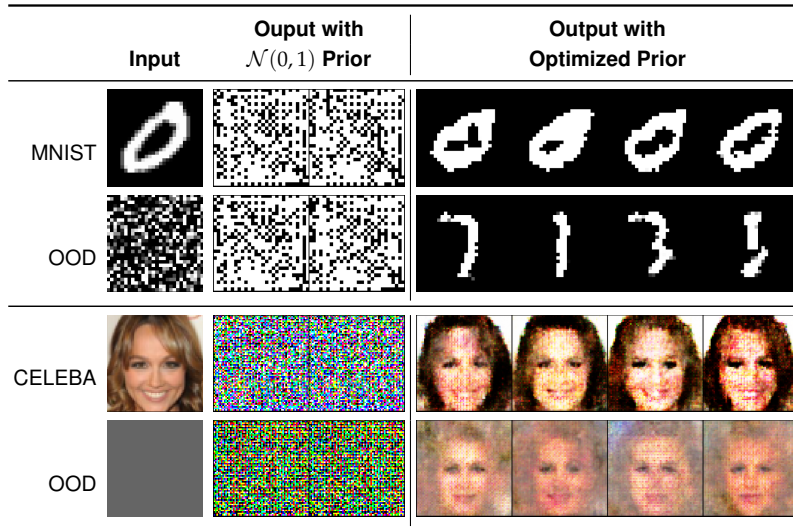
Figure 4.1: Realizations sampled from different priors given an input image. OOD stands for out-of-distribution.

4.4 Model Selection for Bayesian Autoencoders via Prior Optimization

One of the main advantages of the Bayesian paradigm is that we can incorporate prior knowledge into the model in a principled way. Let us assume a prior distribution $p_{\psi}(w)$ on the parameters of the AE network, where now we are explicit on the set of (hyper-)parameters that determine the prior, i.e., ψ. Specifying the prior is easy, e.g., a Gaussian. Determining the effective functional prior, i.e., the prior over the network output f is not trivial due to the complex nonlinear forms of f_{enc} and f_{dec}, which induce a non-trivial effect on the output (functional) prior:

$$p_{\psi}(f) = \int f(x; w) p_{\psi}(w) \mathrm{d}w, \tag{4.4}$$

where, as before, $f = f(x; w)$ is the functional output of the BAE. Although $p_{\psi}(f)$ cannot be evaluated analytically, it is possible to draw samples from it.

Prior parameterization. The only two requirements needed to design a parameterization for the prior are: to be able to (1) draw samples from it and (2) to compute its log-density at any point. The latter is required by many inference algorithms such as SGHMC. We consider a fully-factorized Gaussian prior over weights and biases at layer l:

$$p(w_l) = \mathcal{N}(w_l; \mu_{l_w}, \sigma^2_{l_w}), \quad p(b_l) = \mathcal{N}(b_l; \mu_{l_b}, \sigma^2_{l_b}), \tag{4.5}$$

Notice that, as we shall see in § 4.4.2 and § 4.4.3, in order to estimate our prior hyperparameters, we will require gradient back-propagation through the stochastic

variables w_l and b_l. Thus, we treat these parameters in a deterministic manner by means of the reparameterization trick (Rezende et al., 2014; Kingma and Welling, 2014).

4.4.1 Another route for Bayesian Occam's razor

A common way to estimate hyperparameters (i.e., prior parameters $\boldsymbol{\psi}$) is to rely on the Bayesian Occam's razor (a.k.a. *empirical Bayes*), which dictates that the marginal likelihood $p_{\boldsymbol{\psi}}(\boldsymbol{y})$ should be optimized with respect to $\boldsymbol{\psi}$. There are countless examples where such simple procedure succeeds in practice (see, e.g., Rasmussen and Williams, 2006; Immer et al., 2021b). We note however that marginal likelihood maximization for a large number of hyperparameters can suffer from overfitting (Rasmussen and Williams, 2006; Sebastian et al., 2021). Nevertheless, we do not expect significant overfitting issues in our setting, as we focus on data that are characterized by a high level of structure (i.e. images). As we have seen, regular choices for the prior completely fail to capture the properties of such highly-structured outputs.

The marginal likelihood is obtained by marginalizing out the outputs f and the model parameters w,

$$p_{\boldsymbol{\psi}}(\boldsymbol{y}) = \int p(\boldsymbol{y} \mid f) p_{\boldsymbol{\psi}}(f) \mathrm{d}f, \tag{4.6}$$

where $p(\boldsymbol{y} \mid f)$ and $p_{\boldsymbol{\psi}}(f)$ are given by Eq. 4.3 and Eq. 4.4, respectively. Unfortunately, in our context it is impossible to carry out this optimization due to the intractability of Eq. 4.6.

Classic results in the statistics literature draw parallels between maximum likelihood estimation (MLE) and KL minimization (Akaike, 1973),

$$\arg\max_{\boldsymbol{\psi}} \int \pi(\boldsymbol{y}) \log p_{\boldsymbol{\psi}}(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \arg\min_{\boldsymbol{\psi}} \underbrace{\int \pi(\boldsymbol{y}) \log \frac{\pi(\boldsymbol{y})}{p_{\boldsymbol{\psi}}(\boldsymbol{y})} \mathrm{d}\boldsymbol{y}}_{\mathrm{KL}[\pi(\boldsymbol{y}) \,\|\, p_{\boldsymbol{\psi}}(\boldsymbol{y})]}, \tag{4.7}$$

where $\pi(\boldsymbol{y})$ is the true data distribution. This equivalence provides us with an interesting insight on an alternative view of marginal likelihood optimization as minimization of the divergence between the true data distribution and the marginal $p_{\boldsymbol{\psi}}(\boldsymbol{y})$.

This alternative view allows one to obtain a viable optimization strategy that relies on an empirical estimate of the data distribution $\tilde{\pi}(\boldsymbol{y})$. This presents additional challenges however, as the empirical evaluation and optimization of KL divergences remains a well-known challenging problem (Flam-Shepherd et al., 2017). Although it is possible to evaluate KL (or any other f-divergence) empirically by leveraging results from convex analysis (Nguyen et al., 2010), we have opted to substitute KL

with an alternative metric that is more convenient from a computational perspective. We are inspired by recent works on generative adversarial networks (Arjovsky et al., 2017; Gulrajani et al., 2017) and Bayesian neural networks (Tran et al., 2022), where it is shown that the Wasserstein distance can be estimated efficiently using samples only, even for high-dimensional distributions. We thus employ the Wasserstein distance as a surrogate for KL divergence, so that we avoid the challenges of empirical KL estimation.

To summarize: (1) we would like to do prior selection by carrying out type-II MLE; (2) the MLE objective is analytically intractable but the connection with KL minimization allows us to (3) swap the divergence with the Wasserstein distance, yielding a practical framework for choosing priors.

4.4.2 Matching the marginal distribution to the data distribution via Wasserstein distance minimization

Given the two probability measures π and p_{ψ}, both defined on $\mathbb{R}^D$ for simplicity, the p-Wasserstein distance between π and p_{ψ} is given by

$$W_p^p(\pi, p_{\psi}) = \inf_{\gamma \in \Gamma(\pi, p_{\psi})} \int \|\boldsymbol{y} - \boldsymbol{y}'\|^p \gamma(\boldsymbol{y}, \boldsymbol{y}') \mathrm{d}\boldsymbol{y} \mathrm{d}\boldsymbol{y}', \tag{4.8}$$

where $\Gamma(\pi, p_{\psi})$ is the set of all possible distributions $\gamma(\boldsymbol{y}, \boldsymbol{y}')$ such that the marginals are $\pi(\boldsymbol{y})$ and $p_{\psi}(\boldsymbol{y}')$ (Villani, 2008). While usually analytically unavailable or computationally intractable, for $D = 1$ the distance has a simple closed form solution, that can be easily estimated using samples only (Kolouri et al., 2019).

The distributional sliced-Wasserstein distance (DSWD) takes advantage of this result by projecting the estimation of distances for high-dimensional distributions into simpler estimation of multiple distances in one dimension. The projection is done using the Radon transform $\mathcal{R}$, an operator that maps a generic density function φ defined in $\mathbb{R}^D$ to the set of its integrals over hyperplanes in $\mathbb{R}^D$,

$$\mathcal{R}\varphi(t, \boldsymbol{\theta}) := \int \varphi(\boldsymbol{r})\delta(t - \boldsymbol{r}^\top \boldsymbol{\theta})\mathrm{d}\boldsymbol{r}, \quad \forall t \in \mathbb{R}, \ \ \forall \boldsymbol{\theta} \in \mathbb{S}^{D-1}, \tag{4.9}$$

where $\mathbb{S}^{D-1}$ is the unit sphere in $\mathbb{R}^D$ and $\delta(\cdot)$ is the Dirac delta (Helgason, 2010). Using the Radon transform, for a given direction (or *slice*) $\boldsymbol{\theta}$ we can project the two densities π and p_{ψ} into one dimension and we can solve the optimal transport problem in this projected space. Furthermore, to avoid unnecessary computations, instead of considering all possible directions in $\mathbb{S}^{D-1}$, DSWD proposes to find the optimal probability measure of slices $\sigma(\boldsymbol{\theta})$ on the unit sphere $\mathbb{S}^{D-1}$,

$$DSW_p(\pi, p_{\psi}) := \sup_{\sigma \in \mathbb{M}_C} \left(\mathbb{E}_{\sigma(\boldsymbol{\theta})} W_p^p\big(\mathcal{R}\pi(t, \boldsymbol{\theta}), \mathcal{R}p_{\psi}(t, \boldsymbol{\theta})\big) \right)^{1/p}, \tag{4.10}$$

where, for $C > 0$, $\mathbb{M}_C$ is the set of probability measures σ such that $\mathbb{E}_{\theta,\theta' \sim \sigma}[\theta^\top \theta'] \leq C$ (a constraint that aims to avoid directions to lie in only one small area). The direct computation of DSW_p in Eq. 4.10 is still challenging but admits an equivalent dual form,

$$\sup_{h \in \mathcal{H}} \left\{ \left(\mathbb{E}_{\bar{\sigma}(\theta)}\left[W_p^p(\mathcal{R}\pi(t, h(\theta)), \mathcal{R}p_\psi(t, h(\theta))) \right] \right)^{1/p} - \lambda_C \mathbb{E}_{\theta,\theta' \sim \bar{\sigma}}\left[\left| h(\theta)^\top h(\theta') \right| \right] \right\} + \lambda_C C,$$

(4.11)

where $\bar{\sigma}$ is a uniform distribution in $\mathbb{S}^{D-1}$, $\mathcal{H}$ is the set of functions $h : \mathbb{S}^{D-1} \to \mathbb{S}^{D-1}$ and λ_C is a regularization hyperparameter. The formulation in Eq. 4.11 is obtained by employing the Lagrangian duality theorem and by reparameterizing $\sigma(\theta)$ as push-forward transformation of a uniform measure in $\mathbb{S}^{D-1}$ via h. Now, by parameterizing h using a deep neural network with parameters ϕ, defined as h_ϕ, Eq. 4.11 becomes an optimization problem with respect to the network parameters. The final step is to approximate the analytically intractable expectations with Monte Carlo integration,

$$\max_{\phi} \left\{ \left[\frac{1}{K} \sum_{i=1}^{K} \left[W_p^p(\mathcal{R}\pi(t, h_\phi(\theta_i)), \mathcal{R}p_\psi(t, h_\phi(\theta_i))) \right] \right]^{1/p} - \frac{\lambda_C}{K^2} \sum_{i,j=1}^{K} |h_\phi(\theta_i)^\top h_\phi(\theta_j)| \right\} + \lambda_C C,$$

with $\theta_i \sim \bar{\sigma}(\theta)$. Finally, we can use stochastic gradient methods to update ϕ and then use the resulting optima for the estimation of the original distance. We encourage the reader to check the detailed explanation of this formulation, including its derivation and some practical considerations for implementation, available in the Appendix.

4.4.3 Summary

We aim at learning the prior on the BAE parameters by optimizing the marginal $p_\psi(y)$ obtained after integrating out the weights from the joint $p_\psi(y, w)$. The connection with *empirical Bayes* and KL minimization suggests that we can find the optimal $\psi^\star$ by minimizing the KL between the true data distribution $\pi(y)$ and the marginal $p_\psi(y)$. However, matching these two distributions is non-trivial due to their high dimensionality and the unavailability of their densities. To overcome this problem, we propose a sample-based approach using the distributional sliced 2-Wasserstein distance (Eq. 4.11) as objective:

$$\psi^\star = \arg\min_{\psi} \left[DSW_2(p_\psi(y), \pi(y)) \right].$$

(4.12)

This objective function is flexible and does not require the closed-form of either $p_\psi(y)$ or $\pi(y)$. The only requirement is that we can draw samples from these two distributions. Note that we can sample from $p_\psi(y)$, by first computing f after sampling from $p_\psi(w)$ and then perturbing the generated f by sampling from the

likelihood $p(y \mid f)$. For the continuous Bernoulli likelihood this operation can be implemented by using the reparameterization form that allows to backpropagate gradients (Loaiza-Ganem and Cunningham, 2019).

4.5 Experiments

Competing approaches. We compare our proposal with a wide selection of methods from the literature. For autoencoding methods, we choose the vanilla **VAE** (Kingma and Welling, 2014), the **β-VAE** (Higgins et al., 2017) and **WAE** (Wasserstein AE) (Tolstikhin et al., 2018). In addition, we consider models with more complex encoders (**VAE + Sylvester flows** (Berg et al., 2018)), generators (**2-stage VAE** (Dai and Wipf, 2019)), and priors (**VAE + VampPrior** (Tomczak and Welling, 2018)). For CELEBA we also include a comparison with generative adversarial networks (GANs), with the vanilla setup of **NS-GAN** (Goodfellow et al., 2014; Lucic et al., 2018) and the more recent **DiffAugment-GAN** (Zhao et al., 2020; Karras et al., 2020). Finally, we also compare against BAE with the standard $\mathcal{N}(0,1)$ prior. Unless otherwise stated, all models—including ours—share the same latent dimensionality ($K = 50$). We defer a more detailed description of these models and architectures to the Appendix.

Generative process. Differently from VAEs and other methods, deterministic and Bayesian AEs are not generative models. To generate new samples with BAEs we employ ex-post density estimation over the learned latent space, by fitting a density estimator $p_{\vartheta}(z)$ to $\{z_i = \mathbb{E}_{p(w_{\text{enc}} \mid y)}[f_{\text{enc}}(x_i; w_{\text{enc}})]\}$. In this work, we employ a non-parametric model for density estimation based on Dirichlet process mixture model (DPMM) (Blei and Jordan, 2006), so that its complexity is automatically adapted to the data; see also (Bengio et al., 2013a) for alternative ways to turn AEs into generative models. After estimating $p_{\vartheta}(z)$, a new sample can be generated by drawing z_{new} from $p_{\vartheta}(z)$ and $f_{\text{new}} = \mathbb{E}_{p(w_{\text{dec}} \mid y)}[f_{\text{dec}}(z_{\text{new}}; w_{\text{dec}})]$.

Evaluation metrics. To evaluate the reconstruction quality, we use the test log-likelihood (LL), which tells us how likely the test targets are generated by the corresponding model. The predictive log-likelihood is a proper scoring rule that depends on both the accuracy of predictions and their uncertainty (Gneiting and Raftery, 2007). To assess the quality of the generated images, instead, we employ the widely used Fréchet Inception Distance (FID) (Heusel et al., 2017). We note that, as GANs are not inherently equipped with an explicit likelihood model, we only report their FID scores. Finally, all our experiments and evaluations are repeated four times, with different random training splits.
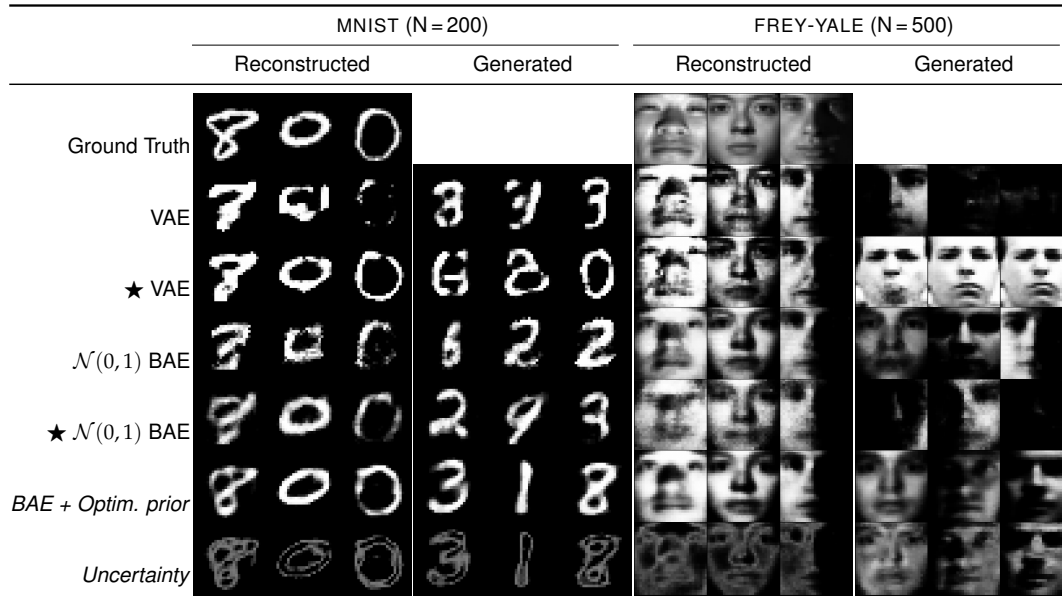
Figure 4.2: Qualitative evaluation for MNIST and YALE. Here, ★ indicates using the union of the training data and the data used to optimize prior to train the model. The last row depicts standard deviation of reconstructed/generated images estimated by BAE using the optimized prior.

4.5.1 Analysis of the effect of the prior

To demonstrate the effect of our model selection strategy, we consider scenarios in the small-data regime where the prior might not be necessarily tuned on the training set. In this way we are able to impose inductive bias beyond what is available in the training data. We investigate two cases:

- MNIST (Lecun et al., 1998): We use 100 examples of the 0 digits to tune the prior. The training set consists of examples of 1-9 digits, whereas the test set contains 10 000 instances of all digits. We aim to demonstrate the ability of our approach to incorporate prior knowledge about completely unseen data with different characteristics into the model.
- FREY-YALE (Dai et al., 2015): We use 1 956 examples of FREY faces to optimize the prior. The training set and test set are comprised of YALE faces. We demonstrate the benefit of using a different dataset but from the same domain (e.g. face images) to specify the prior distribution.

Visual inspection. Fig. 4.2 shows some qualitative results (additional images are available in the Appendix), while Fig. 4.3 shows the convergence of the Wasserstein distance during prior optimization in our proposal. From a visual inspection we see that, on MNIST, by encoding knowledge about the "0" digit into the prior, the BAE can reconstruct this digit fairly well although we only use "1" to "9" digits for inference (differently from the BAE with standard prior). Similarly, on FREY-YALE, we see that by encoding knowledge from another dataset in the same domain, the
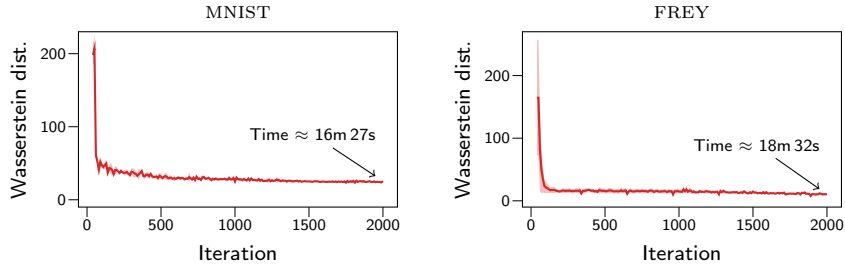
Figure 4.3: Convergence of the proposed Wasserstein minimization scheme.
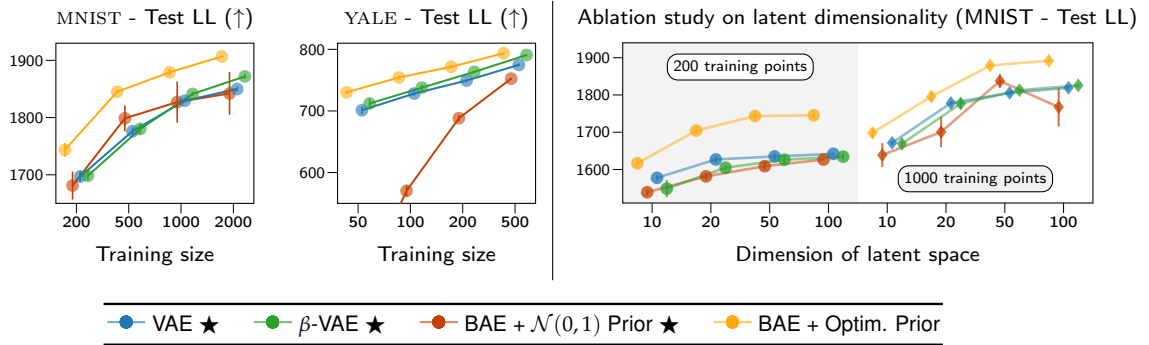
Figure 4.4: Test log-likelihood (LL) of MNIST and YALE. *Left:* test LL as a function of training size; *Right:* test LL as a function of latent dimensionality.

optimized prior can impose a softer constraint compared to using directly this dataset for inference. In addition, if we use directly the union of FREY and YALE faces for training (methods denoted with a ★), VAE yields images that are similar to FREY instead of YALE faces, while generated images from BAE with $\mathcal{N}(0,1)$ prior are of lower quality. This again highlights the advantage of our approach to specifying an informative prior compared to using that data for training. Another important benefit of our Bayesian treatment of AEs is that we can quantify the *uncertainty* for both reconstructed and generated images. The last row of Fig. 4.2 illustrates the uncertainty estimate corresponding to the BAE with optimized prior on MNIST and YALE datasets. Our model exhibits increased uncertainty for semantically and visually challenging pixels such as the left part of the second "0" digit image in the MNIST example. We also observe that the uncertainty is greater for generated images compared to reconstructed images as illustrated in the YALE example. This is reasonable because the reconstruction process is guided by the input data rather than synthesizing new data according to a random latent code.

Visualization of inductive bias on MNIST. To have an intuition of the inductive bias induced by the optimized prior, we visualize a low-dimensional projection of parameters sampled from the prior and the posterior (Izmailov et al., 2019). As we see in Fig. 4.5, the hypothesis space induced by the $\mathcal{N}(0,1)$ prior is huge, compared to where the true solution should lie. Effectively this is another visualization of the

famous Bayesian Occam's razor plot by Mackay (2003), where the model has very high complexity and poor inductive biases. On the other hand, by considering our proposal to do *model selection*, the hypothesis space of the optimized prior is reduced to regions close to the full posterior. Additional visualizations are available in the Appendix.
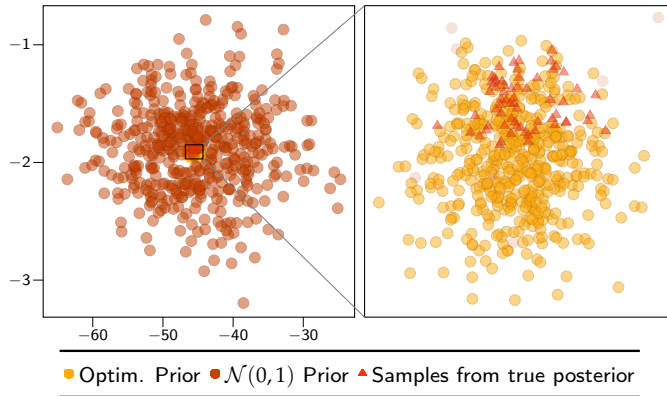
Figure 4.5: Visualization in 2D of samples from priors and posteriors of BAE's parameters. The setup is the same as before with MNIST.

Quantitative evaluation. For a quantitative analysis we rely on Fig. 4.4, where we study the effect on the reconstruction quality of different training sizes (on the *left*) and different latent dimensions (on the *right*). Since we observed that the results of VAE variants are not significantly different, we only show the results for β-VAE and we leave the extended results to the Appendix. From this experiment we can draw important conclusions. The BAE with optimized priors clearly outperforms the competing methods (and the BAE with standard prior) in the inference task for all training sizes, with slightly diminishing effect for larger sets, as expected. This pattern also holds when looking at different latent dimensions (Fig. 4.4, *right*), where regardless of the dimensionality of the latent space, BAEs with optimized priors achieve the best performance.

4.5.2 Reconstruction and generation of CELEBA

We now look at a more challenging benchmark, the CELEBA dataset (Liu et al., 2015). For our proposal, we use 1 000 examples that are randomly chosen from the original training set to learn the prior distribution. The test set consists of about 20 000 images. The goal of this experiment is to evaluate whether sacrificing part of the training data to specify a good prior is beneficial when compared to using that data for training the model. Fig. 4.6 shows qualitative results for the competing methods, their corresponding test LLs and FIDs for different training dataset sizes. In terms of test log-likelihoods (LLs) (Fig. 4.6, *top right*), we observe two clear patterns: (i) that BAE approaches perform considerably better than other methods and (ii) the
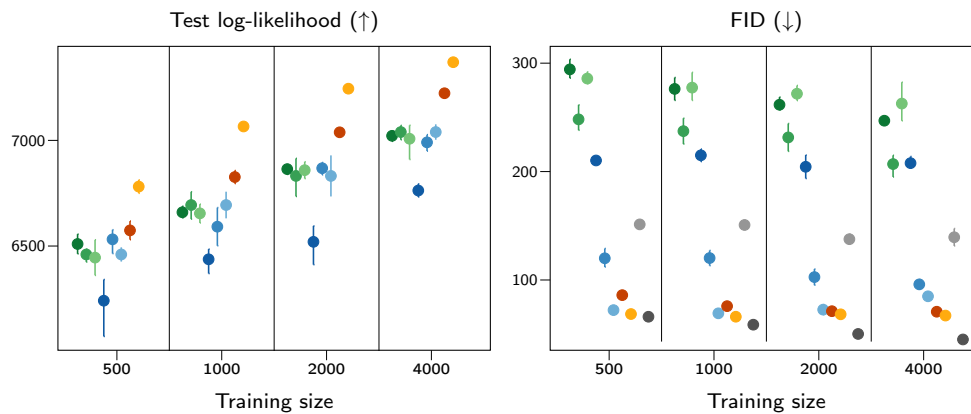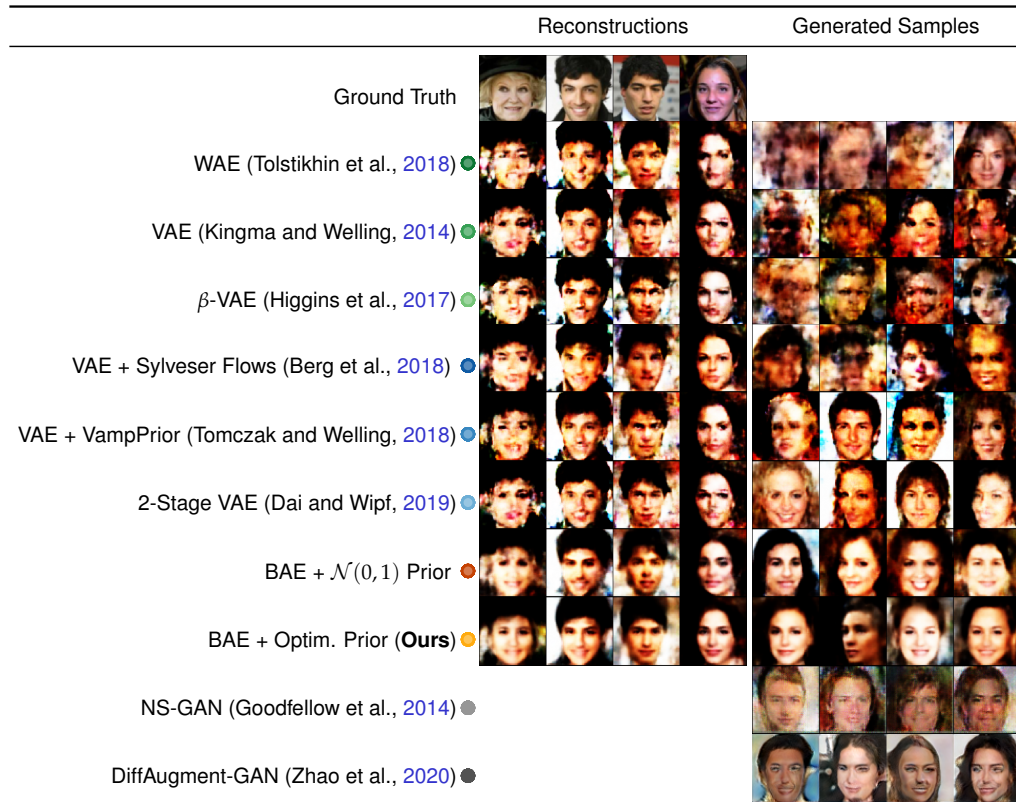
Figure 4.6: Qualitative (*top*) and quantitative evaluation (*bottom*) on CELEBA. The markers and bars represent the means and one standard deviations, respectively. In the (*top*) figure, the sizes of training data and the data for optimizing prior are 500 and 1000, respectively. The higher the log-likelihood (LL) and the lower FID the better.

VAE with Sylvester flows performs consistently poor across dataset sizes. This latter observation indicates that having a more expressive posterior for the encoder is not helpful when considering the small training sizes used in our experiments. More importantly, we see that the BAE using the optimized prior significantly outperforms other methods despite using less data for inference. These results largely agree with the quality of the reconstructions (first column of images in Fig. 4.6, *left*) in that BAE methods provide more visually appealing reconstructions when compared to other approaches.

We now evaluate the quality of the generated images (second column of images in Fig. 4.6, *left*) along with their FID scores (Heusel et al., 2017). Visually, it is clear that images generated from VAEs (standard, β, Sylvester and WAE) are very poor. This failure may originate from the fact that the aggregated posterior distribution of the encoder is not aligned with the prior on the latent space. This problem is more prominent in the case of small training data, where the encoder is not well-trained. The VampPrior tackles this problem by explicitly modeling the aggregated posterior, while 2-stage VAE uses another VAE to estimate the density of the learned latent space. By reducing the effect of the aggregated posterior mismatch, these strategies improve the quality of the generated images remarkably. These results are consistent with their corresponding FID scores (Fig. 4.6, *bottom right*) where we also see that BAE using the optimized prior consistently outperforms all variants of VAEs and NS-GAN. Finally, we see that DiffAugment-GAN, with the exception of using a training size of 500, yields better FID scores. However, this is not surprising as this model uses much more complex network architectures (Karras et al., 2020), combined with a powerful differentiable augmentation scheme. More importantly, it is clear that with few training samples our method generates more semantically meaningful images then all other approaches, including DiffAugment-GAN.

The effect of the likelihood. As previously mentioned, in the Bayesian paradigm, the likelihood represents a modeling choice to be made in addition to the choice of the prior. Our empirical evaluation was predominantly conducted with the continuous Bernoulli likelihood. This likelihood maybe not be an ideal choice for colored images because it biases pixel values to the extremes, which results in saturated images. For the CELEBA dataset, we additionally consider the truncated Gaussian likelihood (Burkardt, 2014), which is another valid alternative for $[0, 1]$-valued data (results summarized in Fig. 4.7). We observe that indeed the colors in the images generated by this likelihood are more realistic and less saturated compared to those generated by the continuous Bernoulli. Still the problem of selecting a good prior is present, as it can be seen for methods like e.g. VAE. For our proposed approach we haven't made strong assumptions on the likelihood (just the ability to sample from it) and as such it is flexible and not tied to a specific choice. These results show that our method is not only quantitatively but also qualitatively better than the competing
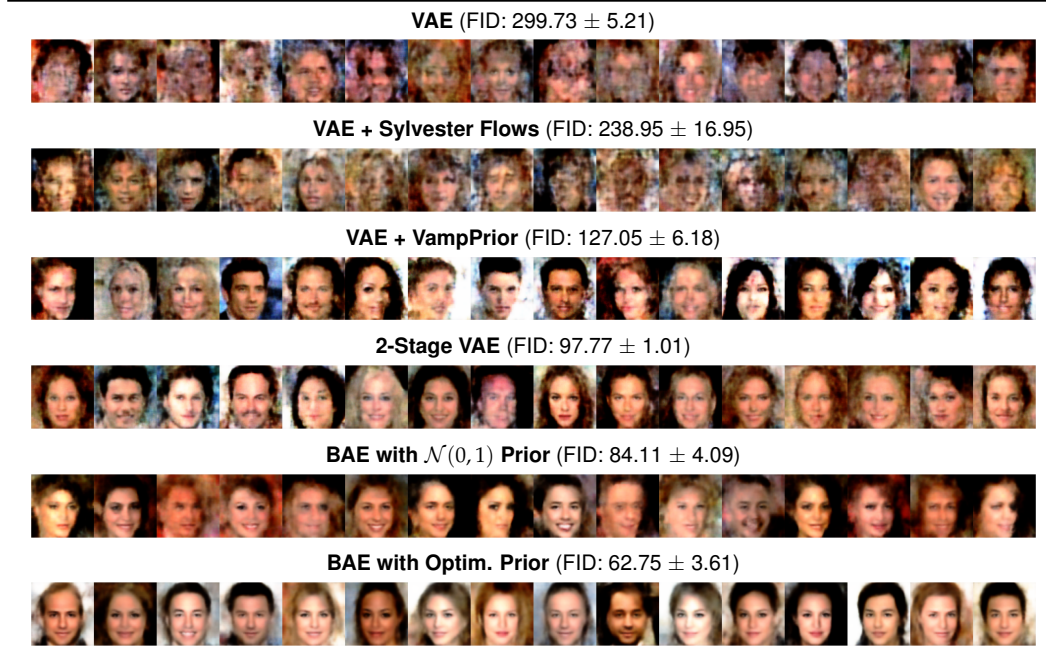
Figure 4.7: Qualitative and quantitative evaluation of generated samples with the *truncated Gaussian likelihood* (Burkardt, 2014). Here, we use 500 CELEBA samples for inference.

approaches and confirm that the benefits from our framework are independent of the choice of likelihood. In Appendix, we show more qualitative and numerical results of reconstruction and generation with the truncated Gaussian likelihood.

4.5.3   Prior adjustment versus posterior tempering

We have shown that the proposed framework for adjusting the prior is compatible with standard Bayesian practices, as it emulates type-II maximum likelihood. In other words, the distribution fitting that we induce by means of Wasserstein distance minimization relates to the marginal output of BAEs, very much in the same spirit of marginal likelihood maximization. The distribution is fit considering *all* possible functions, when marginalized through the likelihood, creating an implicit regularization effect. Our scheme does not give more weight to particular training instances, but it simply restricts the hypothesis space. This is unlike *posterior tempering* (Zhang et al., 2018b; Izmailov et al., 2019; Wenzel et al., 2020; Aitchison, 2021; Zeno et al., 2021), which is commonly defined as $p_\tau(\boldsymbol{w} \,|\, \boldsymbol{y}) \propto p(\boldsymbol{y} \,|\, \boldsymbol{w})^{1/\tau} p(\boldsymbol{w})$, where $\tau > 0$ is a *temperature* value. With $\tau < 1$, tempering is known to improve performance in the case of small training data and using miss-specified priors, but it corresponds to artificially sharpening the posterior by over-counting the data τ times.
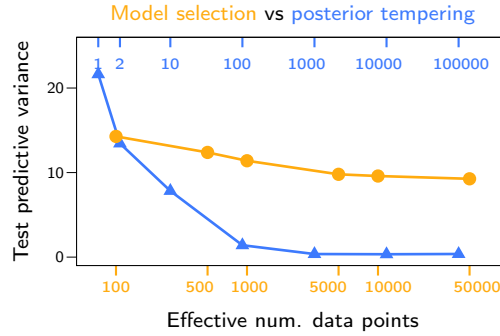
Figure 4.8: Average test predictive variance as a function of the number of data points used to optimize the prior, and the temperature (i.e. how many times the data points are over-counted).
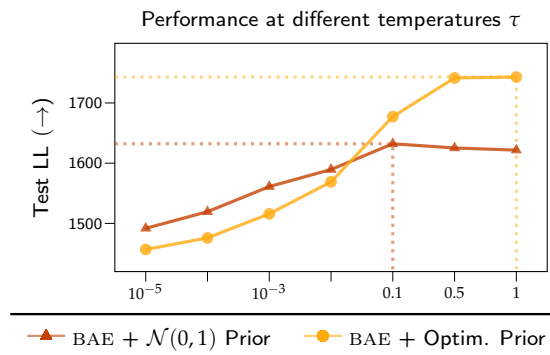
Figure 4.9: Test performance for temperature scaling with different priors. The dotted lines indicate the best performance.

To demonstrate the differences with our proposal, we setup a comparison on MNIST. In the empirical comparison of Fig. 4.8, we consider different temperatures and different sets of data points used to optimize the prior. As expected, the tempered posterior quickly collapses on the mode, while the posterior after our treatment retains a sufficiently constant variance, regardless of the number of data points used. It is also interesting to notice that with the $\mathcal{N}(0,1)$ prior, the best temperature is $\tau = 0.1$, while for our approach that optimizes the prior is $\tau = 1$, further confirming that the model now is well specified (Fig. 4.9).

4.6 Conclusions

In this work, we have reconsidered the Bayesian treatment of autoencoders (AE) in light of recent advances in Bayesian neural networks. We have addressed the main challenge of BAEs, so that they can be rendered as viable alternative to generative models such as VAEs. More specifically, we have found that the main limitation of BAEs lies in the difficulty of specifying meaningful priors in the context of highly-structured data, which is ubiquitous in modern machine learning applications. Consequently, we have proposed to specify priors over the autoencoder weights by

means of a novel optimization of prior hyperparameters. Inspired by connections
with marginal likelihood optimization, we derived a practical and efficient optimiza-
tion framework, based on the minimization of the distributional sliced-Wasserstein
distance between the distribution induced by the BAE and the data generating dis-
tribution. The resulting hyperparameter optimization strategy leads to a novel way
to perform model selection for BAEs, and we showed its advantages in an extensive
experimental campaign.

Limitations and ethical concerns.   Even if theoretically justified and empirically
verified with extensive experimentation, our proposal for model selection still remains
a *proxy* to the true marginal likelihood maximization. The DSWD formulation has
nice properties of asymptotic convergence and computational tractability, but it
may represent only one of the possible solutions. At the same time, we stress that
the current literature does not cover this problem of BAEs at all, and we believe
our approach is a considerable step towards the development of practical Bayesian
methods for representation learning in modern applications characterized by large-
scale structured data (including tabular and graph data, which are currently not
covered). At the same time, the accessibility to these models to a wider audience
and different kind of data might help to widespread harmful applications, which is a
concern shared among all generative modeling approaches. An ethical analysis of
the consequences of Bayesian priors in unsupervised learning scenarios is also worth
an in-depth investigation, which goes beyond the scope of this work.

Chapter 5

Fully Bayesian Autoencoders with Latent Sparse Gaussian Processes

In this chapter, we present a fully Bayesian autoencoder model that treats both local latent variables and global decoder parameters in a Bayesian fashion. This approach allows for flexible priors and posterior approximations while keeping the inference costs low. To achieve this, we introduce an amortized MCMC approach by utilizing an implicit stochastic network to learn sampling from the posterior over local latent variables. Furthermore, we extend the model by incorporating a Sparse Gaussian Process prior over the latent space, allowing for a fully Bayesian treatment of inducing points and kernel hyperparameters and leading to improved scalability. Additionally, we enable Deep Gaussian Process priors on the latent space and the handling of missing data. We evaluate our model on a range of experiments focusing on dynamic representation learning and generative modeling, demonstrating the strong performance of our approach in comparison to existing methods that combine Gaussian Processes and autoencoders.

5.1   Introduction

The problem of learning representations of data that are useful for downstream tasks is a crucial factor in the success of many machine learning applications (Bengio et al., 2013b). Among the numerous proposed solutions, modeling approaches that evolved from autoencoders (AEs) (Cottrell et al., 1989) are particularly appealing, as they do not require annotated data and have proven effective in unsupervised learning tasks, such as data compression and generative modeling (Tomczak, 2022; Yang et al., 2022). AEs are neural networks consisting of an encoder that maps input data to a set of lower-dimensional latent codes and a decoder that maps the latent codes back to the observations.

In applications where data is scarce or uncertainty quantification is crucial, it is beneficial to treat these models in a Bayesian manner (Mackay, 1992; Neal, 1996; Wilson and Izmailov, 2020; Izmailov et al., 2021b) by imposing meaningful prior distributions over both the parameters of the encoder and decoder (Tran et al., 2021; Miani et al., 2022). A parallel development are variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende and Mohamed, 2015) that treat the latent space of an autoencoder in a Bayesian fashion, enabling scalable inference over a large number of local (per-data point) latent variables using amortized variational inference. Note that these models typically treat the encoder and decoder as deterministic neural networks. In the related works section, we further elaborate on the differences between several versions of AE models and the way Bayesian inference is carried out.

A critical limitation of standard VAEs is their utilization of factorized priors, which is inadequate for modeling correlations between latent encodings. However, capturing latent correlations is often necessary to model structured data. For example, in autonomous driving or medical imaging applications, high-dimensional images are correlated in time. Spatio-temporal dependencies between samples are also common in environmental and life sciences datasets. To address this limitation, several works (Pearce, 2019; Casale et al., 2018) have attempted to introduce Gaussian process (Gaussian process (GP)) priors over the latent space of VAEs that capture correlations between pairs of latent variables through a kernel function. While GP priors outperform conventional priors on many tasks, they also introduce computational challenges such as $\mathcal{O}(N^3)$ complexity for GP inference, where N is the number of data instances. Recently, Jazbec et al. (2021) proposed the SVGP-VAE model to tackle this computational issue by relying on sparse approximations; these summarize the dataset into a set of so-called inducing points (Quiñonero-Candela and Rasmussen, 2005).

Although the SVGP-VAE model (Jazbec et al., 2021) has achieved promising results, it has several significant drawbacks. First, similarly to VAEs, SVGP-VAE is strongly tied to a variational inference (VI) formulation (Jordan et al., 1999; Zhang et al., 2018a), which can lead to poor approximations due to VI making strong assumptions on both the factorization and functional form of the posterior. Second, the SVGP-VAE model follows the common practice in the sparse GP literature of optimizing the inducing points and kernel hyperparameters based on the marginal likelihood. However, this approach does not account for uncertainty in the inducing inputs and hyperparameters. It is well known that this can result in biased estimates and underestimated predictive uncertainties (Rossi et al., 2021; Lalchand et al., 2022b). In this work, we propose a novel Sparse Gaussian Process Bayesian Autoencoder (SGP-BAE) model that addresses these issues by providing scalable and flexible inference through a fully Bayesian treatment without relying on VI.

Contributions. Specifically, **first**, we develop a fully Bayesian autoencoder (BAE) model (§ 5.3), where we adopt a Bayesian treatment for both the local (per-data point) latent variables and the global decoder parameters. This approach differs from VAEs in that it allows specifying any prior over the latent space while decoupling the model from the inference. As a result, we can rely on powerful alternatives to VI to carry out inference, and we adopt stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) as a scalable solution. To achieve this, we propose an amortized Markov chain Monte Carlo (MCMC) approach for our Bayesian autoencoder (BAE) model by using an implicit stochastic network as the encoder to learn to draw samples from the posterior of local latent variables. Our approach addresses the prohibitively expensive inference cost induced by the local latent variables and avoids making strong assumptions on the form of the posterior. **Second**, when imposing a GP prior over the latent space, we propose a novel scalable SGP-BAE model (§ 5.4) in which the inducing points and kernel hyperparameters of the sparse GP prior on the latent space are treated in a fully Bayesian manner. This model offers attractive features such as high scalability, richer modeling capability, and improved prediction quality. **Third**, we extend the SGP-BAE model to allow one to impose deep GP priors on the latent space (Damianou and Lawrence, 2013) and to handle missing data. To the best of our knowledge, this is the first work to consider deep GP priors for AEs. **Finally**, we conduct a rigorous evaluation of our SGP-BAE model through a variety of experiments on dynamic representation learning and generative modeling. The results demonstrate excellent performance compared to existing methods of combining GPs and AEs (§ 5.5).

5.2 Related Work

Autoencoders. AEs (Cottrell et al., 1989) are powerful models for representation learning which operate by projecting data onto a lower-dimensional latent space through an encoder and by mapping latent representations back into the original data by means of a decoder. VAEs (Kingma and Welling, 2014; Rezende et al., 2014) elegantly combine AEs with variational inference enabling the model to generate new data and allowing for the specification of any prior on the latent space. To improve the performance of VAEs, recent works have attempted to employ flexible priors such as mixtures (Dilokthanakul et al., 2016; Tomczak and Welling, 2018; Bauer and Mnih, 2019), normalizing flows (Chen et al., 2017), nonparametric models such as Stick-breaking processes (Nalisnick and Smyth, 2017), or Gaussian processes (Casale et al., 2018).

Recently, Tran et al. (2021) and Miani et al. (2022) explored a Bayesian treatment of the encoder and decoder parameters in standard AEs, demonstrating superior performance. However, this approach lacks a mechanism to impose priors on the latent space. In the seminal work on VAE, Kingma and Welling (2014) already explored

a fully Bayesian treatment of VAEs by introducing a prior on the decoder. Variational inference is employed to infer the decoder and the latent variables. Daxberger and Hernández-Lobato (2019) suggested employing SGHMCs for sampling decoder parameters, but this method uses the evidence lower bound (ELBO) of VAEs as the sampling objective, which may lead to suboptimal approximations. Following Glazunov and Zarras (2022) we use the term Bayesian variational autoencoder (BVAE) to refer to this set of models. To avoid confusion with these models, hereafter, we use the term Bayesian autoencoders (BAEs) to indicate our proposed approach, where both the latent variables and decoder are treated in a fully Bayesian way, and inference uses our amortized SGHMC scheme.

Gaussian process priors for AE models. The earliest attempts to combine AE models with GPs are the GP prior VAE (GPPVAE) (Casale et al., 2018) and GP-VAE (Pearce, 2019). Both these models lack scalability for generic kernel choices and data types. GPPVAE is restricted to a specialized GP product kernel and employs a Taylor approximation for GPs, while GP-VAE relies on exact GP inference. Recently, Fortuin et al. (2020) and Zhu et al. (2022) propose GP-VAE and Markovian-GPVAE, respectively, that are indeed scalable (linear in N time complexity) by exploiting the Markov assumption, but they are applicable only on time-series data. Most closely to our method is the approach of Jazbec et al. (2021) (SVGP-VAE), Ashman et al. (2020) (SGP-VAE) and Ramchandran et al. (2021), where they utilize inducing point methods (Titsias, 2009; Hensman et al., 2013) to make GPs scalable. However, all these methods strongly rely on VAEs and a variational formulation for GPs. In this work, we take a completely different route, as we aim to treat sparse GPs and AEs in a fully Bayesian way while enjoying scalability thanks to recent advances in stochastic-gradient MCMC sampling. Table 5.1 compares our proposed models with relevant related works.

5.3 Imposing Distributions over the Latent Space of Bayesian Autoencoders

We are interested in unsupervised learning problems with a high-dimensional dataset consisting of N data points $Y \overset{\text{def}}{=} [y_1, \cdots, y_N]^\top \in \mathbb{R}^{N \times P}$. Each data point has a corresponding low-dimensional auxiliary data entry, summarized as $X \overset{\text{def}}{=} [x_1, \cdots, x_N]^\top \in \mathbb{R}^{N \times D}$. For instance, y_i could be a video frame and x_i the corresponding time stamp. As another example, consider electronic health record (EHR) data, where the auxiliary data could include patient covariate information, such as age, height, weight, sex, and time since remission. Finally, we denote by $Z \overset{\text{def}}{=} [z_1, \cdots, z_N]^\top \in \mathbb{R}^{N \times C}$ the low-dimensional latent representation of the data, meaning that each latent variable z_i lives in a C-dimensional latent space. We aim to build a model that is able to (1)

Table 5.1: A summary of related methods. Here, $\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{S}$ refer to GP hyper-parameters, inducing variables and inducing inputs, respectively. N and B are the number of data points and the mini-batch size, whereas $M, H \ll N$ are the number of inducing points and the low-rank matrix factor, respectively. The notation ✗ indicates the nonexistence of a specific feature (column) within a given model (row), whereas the symbol - denotes that the model is not employed with a GP prior. References are [a] for Kingma and Welling (2014), [b] for Sohn et al. (2015), [c] for Casale et al. (2018), [d] for Pearce (2019), [e] for Fortuin et al. (2020), [f] for Ashman et al. (2020) and [g] for Jazbec et al. (2021).

Model	Scalable	Non i.i.d. data	Free-form posterior	GP complexity	Arbitrary kernel & data type	Learnable GP	Inference $\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{S}$	Ref.
(◉) VAE	✓	✗	✗	-	-	-	-	[a]
(◉) CVAE	✓	✗	✗	-	-	-	-	[b]
(◉) GPPVAE	✓	✓	✗	$\mathcal{O}(NH^2)$	✗	✗	✗	[c]
(◉) GPVAE	✗	✓	✗	$\mathcal{O}(N^3)$	✓	✗	✗	[d]
(◉) GP-VAE	✓	✓	✗	$\mathcal{O}(N)$	✗	✗	✗	[e]
(◉) SGP-VAE	✓	✓	✗	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✗	[f]
(◉) SVGP-VAE	✓	✓	✗	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✗	[g]
(◉) BAE	✓	✗	✓	-	-	-	-	**Ours**
(◉) GP-BAE	✗	✓	✓	$\mathcal{O}(N^3)$	✓	✗	✗	**Ours**
(◉) SGP-BAE	✓	✓	✓	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✓	**Ours**

generate $\boldsymbol{Y}$ based on the auxiliary data $\boldsymbol{X}$, and (2) provide useful and interpretable low-dimensional representations of $\boldsymbol{Y}$.

Model setup. In this work, we consider a model based on AEs, and we aim to treat this in a fully Bayesian manner. This treatment promises improved predictions, reliable uncertainty estimates, and increased robustness under data sparsity (Mackay, 1992; Izmailov et al., 2021b). One difficulty in doing so is that the prior distribution over the latent variables would be determined by the prior over the weights of the encoder and not a distribution of interest. In many applications, including the ones considered here, this is undesirable, and the goal is to impose a certain prior distribution over the latent representation in a similar vein as VAEs and their variants. Therefore, we propose to treat the entire AE in a fully Bayesian manner except for the encoder and to design the encoder in such a way that it maps data $\boldsymbol{y}_i$ into corresponding codes $\boldsymbol{z}_i$ while allowing these mappings to be compatible with posterior samples over the latent codes. For the encoder, as we will elaborate on shortly, we will employ a so-called stochastic inference network to learn to draw posterior samples of latent variables $\boldsymbol{z}_i$ given high-dimensional inputs $\boldsymbol{y}_i$, while for the decoder and the latent variables we employ scalable MCMC techniques.

Bayesian treatment of the latent space and the decoder. In order to retain a fully Bayesian treatment of the latent space and the decoder, we impose a prior $p(\boldsymbol{\varphi})$ over the decoder's parameters, $\boldsymbol{\varphi}$. In addition, another prior $p(\boldsymbol{Z} \mid \boldsymbol{X}; \boldsymbol{\theta})$ is imposed on the
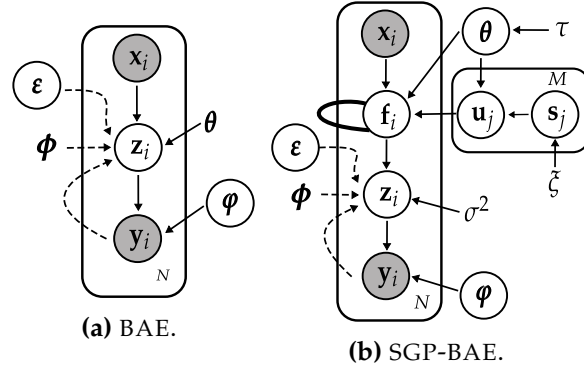
(a) BAE.

(b) SGP-BAE.

Figure 5.1: The graphical models of vanilla BAE **(a)**, and the proposed SGP-BAE with a fully Bayesian sparse GP prior imposed on the latent space. Here, we treat the decoder's parameters φ in a Bayesian way while the encoder's parameters ϕ are considered deterministically. The solid lines denote the generative part, whereas the dashed lines denote the encoding part. The SGP-BAE is treated in a fully Bayesian manner by imposing priors on the decoder's parameters φ as well as the GP kernel's parameters θ, inducing locations s. The cyclic thick line represents that the latent GP correlates with every latent code.

latent variables. This prior is conditioned on the auxiliary data X and characterized by a set of hyperparameters θ. For example, one may employ an uninformative prior such as an isotropic Gaussian commonly used in standard VAEs, but in the next section we will consider structured priors such as GPs and their deep version as well. We assume that the observed data Y is fully factorized and conditional on the latent variables Z and a decoder network with parameters φ, i.e., $p(Y \mid Z, \varphi) = \prod_{i=1}^{N} p(y_i \mid z_i, \varphi)$. The full joint distribution of the model is as follows:

$$p(\varphi, Z, Y \mid X) = p(\varphi)p(Z \mid X; \theta)p(Y \mid Z, \varphi). \tag{5.1}$$

We wish to infer the posterior over the latent variables and decoder parameters, which is given by Bayes' rule:

$$p(\varphi, Z \mid Y, X) = \frac{p(\varphi, Z, Y \mid X)}{p(Y \mid X)}, \tag{5.2}$$

where $p(Y \mid X) = \int p(\varphi, Z, Y \mid X)d\varphi dZ$ is the marginal likelihood. The generative process of data samples from this BAE model is illustrated in Fig. 5.1a.

Characterizing the posterior distribution over φ, Z is analytically intractable and requires approximations. Given the success of scalable MCMC techniques to obtain samples from the posterior of model parameters in deep learning models (Zhang et al., 2020; Tran et al., 2022), in this work, we propose to follow this practice to obtain samples from $p(\varphi, Z \mid Y, X)$. In particular, we employ SGHMC (Chen et al., 2014), which can scale up to large datasets by relying on noisy unbiased estimates of the energy function (log-posterior) $U(\varphi, Z; X, Y) \overset{\text{def}}{=} -\log p(\varphi, Z, Y \mid X)$ and without the need to evaluate the energy function over the entire dataset. More precisely, when the prior over the latent codes is fully factorized, we can approximate this energy

function using mini-batches of size B as follows:

$$U(\boldsymbol{\varphi}, \boldsymbol{Z}; \boldsymbol{X}, \boldsymbol{Y}) \approx \tilde{U}(\boldsymbol{\varphi}, \boldsymbol{Z}; \boldsymbol{X}, \boldsymbol{Y}) = -\log p(\boldsymbol{\varphi}) - \qquad (5.3)$$
$$- \frac{N}{B} \sum_{i \in \mathcal{I}_B} \left[\log p(\boldsymbol{z}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) + \log p(\boldsymbol{y}_i \mid \boldsymbol{z}_i, \boldsymbol{\varphi}) \right]$$

where $\mathcal{I}_B$ is a set of B random indices.

Encoder as a stochastic inference network. SGHMC can be challenging to implement on probabilistic models with many latent variables due to the high computational burden of iteratively refining the approximate posterior for each latent variable. Additionally, it can be difficult to evolve the latent variables for each new test sample. To address these challenges, we propose using a stochastic neural network as an inference network to efficiently generate latent codes similar to those generated by the posterior distribution, inspired by amortized inference techniques (Kingma and Welling, 2014; Wang and Liu, 2016; Feng et al., 2017; Shi et al., 2019) and MCMC distillation (Korattikara Balan et al., 2015; Wang et al., 2018; Li et al., 2017).

More specifically, instead of storing every latent code, we use an inference network $\boldsymbol{z}_i = f_{\boldsymbol{\phi}}(\boldsymbol{y}_i; \boldsymbol{\varepsilon})$ with parameters $\boldsymbol{\phi}$ that generates a corresponding latent code $\boldsymbol{z}_i$ given an input $\boldsymbol{y}_i$ and a random seed $\boldsymbol{\varepsilon}$. The random seed $\boldsymbol{\varepsilon}$ is drawn from a distribution $q(\boldsymbol{\varepsilon})$ that is easy to sample from, such as a uniform or standard Gaussian distribution. The inference network $f_{\boldsymbol{\phi}}$ serves as an encoder by generating posterior samples of the latent code $\boldsymbol{z}_i$ given the observed input $\boldsymbol{y}_i$. It is essential to note that the encoder in our model approximates the posterior distribution of latent variables, which is similar to the approach used in VAEs. However, the primary distinction is that we do not assume any specific form of the posterior distribution of latent variables $\boldsymbol{z}_i$. Our encoder is trained to draw posterior samples of latent variables, rather than serving as a parametric variational distribution.

We incrementally refine the encoder $f_{\boldsymbol{\phi}}$ such that its outputs mimic the SGHMC dynamics. Specifically, after every K iterations of sampling the decoder parameters and the latent codes using SGHMC, we adjust the encoder parameters $\boldsymbol{\phi}$ based on the following objective:

$$\mathcal{L}\left(\{\boldsymbol{z}_i, \boldsymbol{y}_i\}_{i \in \mathcal{I}_B}; \boldsymbol{\phi}\right) \overset{\text{def}}{=} \sum_{i \in \mathcal{I}_B} \left\| f_{\boldsymbol{\phi}}(\boldsymbol{y}_i; \boldsymbol{\varepsilon}_i) - \boldsymbol{z}_i^{(k)} \right\|_2^2, \qquad (5.4)$$

where $\boldsymbol{z}_i^{(k)}$ is the k-th posterior sample from SGHMC of the latent variable $\boldsymbol{z}_i$, and it is used as label to update $\boldsymbol{\phi}$. As the analytic solution of Eq. 5.4 is intractable, we perform J steps of gradient descent to update $\boldsymbol{\phi}$ using an optimizer such as Adam (Kingma and Ba, 2015). It is worth mentioning that our objective to train the inference network is a modeling choice. Conditional generative models such as diffusion models (Ho et al., 2020) or generative adversarial networks (GANs) (Goodfellow et al., 2014) can be

considered as alternatives. However, we must exercise caution to ensure that our goal of learning low-dimensional representations of the data is met. In our experiments, we used a similar network architecture for the inference network as in VAEs for fair comparisons. The inference procedure for our BAEs is described in Algorithm 3.

Algorithm 3: Inference for BAEs with SGHMC

Input: Dataset $\{X, Y\}$, mini-batch size B, # SGHMC iterations K, # encoder interations J

1 Initialize the autoencoder parameters ϕ and φ
2 **while** *φ, Z have not converged* **do**
3   Sample a mini-batch of B random indices $\mathcal{I}_B$
4   Sample random seed $\{\varepsilon_i\}_{i=1}^{B} \sim q(\varepsilon)$
5   Initialize the latent codes from the encoder $\{z_i\}_{i=1}^{B} = f_{\phi}(\{y_i\}_{i\in\mathcal{I}_B}, \{\varepsilon_i\}_{i=1}^{B})$
6   **for** *K iterations* **do**
7   Compute energy function $\tilde{U}$ using Eq. 5.3
8   Sample from posterior $p(\varphi, Z \mid Y, X)$: $\varphi, \{z\}_{i=1}^{B} \leftarrow \text{SGHMC}(\varphi, \{z\}_{i=1}^{B}; \nabla_{\varphi,z}\tilde{U})$
9   **for** *J iterations* **do**
10   Compute objective function $\mathcal{L}$ using Eq. 5.4
11   Update encoder: $\phi \leftarrow \text{Optimizer}(\phi; \nabla_{\phi}\mathcal{L})$

5.4 Scalable Gaussian Process Prior for Bayesian Autoencoders

In the previous section, we introduced our novel version of a BAE where we imposed a simple fully-factorized prior over the latent space, such as isotropic Gaussians. However, in many applications, such priors are incapable of appropriately modeling the correlation nature of the data. For example, it is sensible to model structured data evolving over time with a BAE with a prior over the latent space in the form of a GP with the auxiliary data as input. In this section, we consider precisely these scenarios by introducing GP priors in the latent space, which allows us to model sample covariances as a function of the auxiliary data. We then discuss the scalability issues induced by the use of GP priors, and we propose Sparse Gaussian Process Bayesian Autoencoders (SGP-BAEs) where we recover scalability thanks to sparse approximations to GPs (Quiñonero-Candela and Rasmussen, 2005; Rossi et al., 2021). In this model, we carry out fully Bayesian inference of the decoder, as well as the inducing inputs and covariance hyperparameters of the sparse GPs, while we optimize the stochastic inference network implementing the encoder.

5.4.1 Gaussian process prior

We assume C independent latent functions $f^{[1]}, ..., f^{[C]}$, which results in each z_i being evaluated at the corresponding x_i, i.e., $z_i = \left[f^{[1]}(x_i), \cdots, f^{[C]}(x_i)\right]$. We assume

that each function is drawn from a zero-mean GP prior with a covariance function $\kappa(x, x'; \theta)$:

$$p(Z \mid X; \theta) = \prod_{c=1}^{C} \mathcal{N}\left(z_{1:N}^{[c]} \mid \mathbf{0}, K_{xx \mid \theta}\right), \tag{5.5}$$

where the c-th latent channel of all latent variables, $z_{1:N}^{[c]} \in \mathbb{R}^N$ (the c-th column of Z), has a correlated Gaussian prior with covariance $K_{xx \mid \theta} \in \mathbb{R}^{N \times N}$ obtained by evaluating $\kappa(x_i, x_j; \theta)$ over all input pairs of X. Here, the latent function values are informed by all y values according to the covariance of the corresponding auxiliary input x. One can recover the fully factorized $\mathcal{N}(\mathbf{0}, I)$ prior on the latent space by simply setting $K_{xx \mid \theta} = I$.

This GP prior over the latent space of BAEs introduces fundamental scalability issues. First, we have to compute the inverse and log-determinant of the kernel matrix $K_{xx \mid \theta}$, which results in $\mathcal{O}(N^3)$ time complexity. This is only possible when N is of moderate size. Second, it is impossible to employ a mini-batching inference method like SGHMC since the energy function $U(\varphi, Z; X, Y)$ does not decompose as a sum over all the observations.

5.4.2 Bayesian sparse Gaussian processes

In order to keep the notation uncluttered, we focus on a single channel and suppress the superscript index c. Given a set of latent function evaluations over the dataset, $f = [f_1, \cdots, f_N]^\top$, we assume that the latent codes are stochastic realizations based on f and additive Gaussian noise, i.e., $\mathcal{N}(Z \mid f, \sigma^2 I)$. Sparse GPs (Quiñonero-Candela and Rasmussen, 2005) are a family of approximate models that address the scalability problem by introducing a set of $M \ll N$ inducing points $u = (u_1, \cdots, u_M)$ at corresponding inducing inputs $S = \{s_1, \cdots, s_M\}$ such that $u_i = f(s_i)$. We assume that these inducing variables follow the same GP as the original process, resulting in the following joint prior:

$$p(f, u) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{XX \mid \theta} & K_{XS \mid \theta} \\ K_{SX \mid \theta} & K_{SS \mid \theta} \end{bmatrix}\right), \tag{5.6}$$

where the covariance matrices $K_{SS \mid \theta}$ and $K_{XS \mid \theta}$ are computed between the elements in S and $\{X, S\}$, respectively.

Fully Bayesian sparse GPs. The fully Bayesian treatment of sparse GPs requires priors $p_\tau(\theta)$ and $p_\xi(S)$ over covariance hyperparameters and inducing inputs, respectively, with τ and ξ as prior hyperparameters. With these assumptions, we term this model as Bayesian sparse Gaussian process autoencoder (SGP-BAE), and the corresponding generative model is illustrated in Fig. 5.1b.

By defining $\boldsymbol{\Psi} \stackrel{\text{def}}{=} \{\boldsymbol{\varphi}, \boldsymbol{u}, \boldsymbol{S}, \boldsymbol{\theta}\}$, we can rewrite the full joint distribution of parameters in SGP-BAE:

$$p(\boldsymbol{\Psi}, \boldsymbol{f}, \boldsymbol{Z}, \boldsymbol{Y} \mid \boldsymbol{X}) = \underbrace{p(\boldsymbol{\Psi})}_{\substack{\text{Priors on inducing} \\ \text{inputs \& variables, decoder}}} \underbrace{p(\boldsymbol{f} \mid \boldsymbol{u}, \boldsymbol{X}, \boldsymbol{S}, \boldsymbol{\theta}) p(\boldsymbol{Z} \mid \boldsymbol{f}; \sigma^2 \boldsymbol{I})}_{\text{Sparse GP prior on latent space}} \underbrace{p(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\varphi})}_{\substack{\text{Likelihood of} \\ \text{observed data}}}, \qquad (5.7)$$

where $p(\boldsymbol{\Psi}) = p(\boldsymbol{\varphi}) p_\tau(\boldsymbol{\theta}) p_\xi(\boldsymbol{S}) p(\boldsymbol{u} \mid \boldsymbol{S}, \boldsymbol{\theta})$. Here, $p(\boldsymbol{u} \mid \boldsymbol{S}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{0}, K_{SS \mid \boldsymbol{\theta}})$, and $p(\boldsymbol{f} \mid \boldsymbol{u}, \boldsymbol{X}, \boldsymbol{S}, \boldsymbol{\theta}) = \mathcal{N}(K_{XS \mid \boldsymbol{\theta}} K_{SS \mid \boldsymbol{\theta}}^{-1} \boldsymbol{u}, K_{XX \mid \boldsymbol{\theta}} - K_{XS \mid \boldsymbol{\theta}} K_{SS \mid \boldsymbol{\theta}}^{-1} K_{SX \mid \boldsymbol{\theta}})$. We assume a factorization $p(\boldsymbol{Z} \mid \boldsymbol{f}; \sigma^2 \boldsymbol{I}) = \prod_{i=1}^{N} p(\boldsymbol{z}_i \mid \boldsymbol{f}_i; \sigma^2)$ and make no further assumptions about the other distribution.

Scalable inference objective. We wish to infer the set of variables $\boldsymbol{\Psi}$ and the latent codes $\boldsymbol{Z}$. To do so, we have to marginalize out the latent process $\boldsymbol{f}$ from the full joint distribution above. In particular, we have:

$$\log p(\boldsymbol{\Psi}, \boldsymbol{Z}, \boldsymbol{Y} \mid \boldsymbol{X}) = \log p(\boldsymbol{\Psi}) + \log \int p(\boldsymbol{f} \mid \boldsymbol{\Psi}, \boldsymbol{X}) p(\boldsymbol{Z} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I}) d\boldsymbol{f} + \log p(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\varphi}). \quad (5.8)$$

This objective should be decomposed over observations to sample from the posterior over all the latent variables using a scalable method such as SGHMC. As discussed by Rossi et al. (2021), this can be done effectively by imposing independence in the conditional distribution (Snelson and Ghahramani, 2005), i.e., by parameterizing $p(\boldsymbol{f} \mid \boldsymbol{\Psi}, \boldsymbol{X}) = \mathcal{N}(K_{xs \mid \boldsymbol{\theta}} K_{ss \mid \boldsymbol{\theta}}^{-1} \boldsymbol{u}, \text{diag}[K_{xx \mid \boldsymbol{\theta}} - K_{xs \mid \boldsymbol{\theta}} K_{ss \mid \boldsymbol{\theta}}^{-1} K_{sx \mid \boldsymbol{\theta}}])$. With this approximation, the log-joint marginal becomes as follows:

$$\log p(\boldsymbol{\Psi}, \boldsymbol{Z}, \boldsymbol{Y} \mid \boldsymbol{X}) \approx \log p(\boldsymbol{\Psi}) + \sum_{n=1}^{N} \left\{ \log \mathbb{E}_{p(f_n \mid \boldsymbol{\Psi}, \boldsymbol{X})} [p(\boldsymbol{z}_n \mid f_n; \sigma^2)] + \log p(\boldsymbol{y}_n \mid \boldsymbol{z}_n, \boldsymbol{\varphi}) \right\}$$

$$\stackrel{\text{def}}{=} -U(\boldsymbol{\Psi}, \boldsymbol{Z}; \boldsymbol{X}, \boldsymbol{Y}). \qquad (5.9)$$

We can now carry out inference for this SGP-BAE model by plugging a mini-batching approximation of this energy function into Line 7 and Line 8 of Algorithm 3. By using this sparse approximation, we reduce the computational complexity of evaluating the GP prior down to $\mathcal{O}(BM^2)$, and SGP-BAE can be readily applied to a generic dataset and arbitrary GP kernel function.

Extension to deep Gaussian processes. We can easily extend SGP-BAE to deep Gaussian process priors (Damianou and Lawrence, 2013) to model much more complex functions in the latent space of BAEs. To the best of our knowledge, the use of deep GPs has not been considered in previous work. We assume a deep Gaussian process prior $f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$, where each $f^{(l)}$ is a GP. Each layer is associated with a set of kernel hyperparameters $\boldsymbol{\theta}^{(l)}$, inducing inputs $\boldsymbol{S}^{(l)}$ and inducing variables $\boldsymbol{u}^{(l)}$. The set of variables to be inferred is $\boldsymbol{\Psi} = \{\boldsymbol{\varphi}\} \cup \{\boldsymbol{u}^{(l)}, \boldsymbol{S}^{(l)}, \boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$. The joint

distribution is as follows:

$$p(\boldsymbol{\Psi}, \{\boldsymbol{f}^{(l)}\}_{l=1}^{L}, \boldsymbol{Z}, \boldsymbol{Y} \mid \boldsymbol{X}) = p(\boldsymbol{\Psi}) \underbrace{\prod_{l=1}^{L} p(\boldsymbol{f}^{(l)} \mid \boldsymbol{f}^{(l-1)}, \boldsymbol{\Psi}) p(\boldsymbol{Z} \mid \boldsymbol{f}^{(L)}; \sigma^2 \boldsymbol{I})}_{\text{Deep GP prior}} p(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\varphi}), \quad (5.10)$$

where we omit the dependency on $\boldsymbol{X}$ for notational brevity. To perform inference, the hidden layers $\boldsymbol{f}^{(l)}$ have to be marginalized and propagated up to the final layer L (Salimbeni and Deisenroth, 2017). The marginalization can be approximated by quadrature (Hensman et al., 2015b) or through Monte Carlo sampling (Bonilla et al., 2019). Detailed derivations of this extension can be found in Appendix C.2.
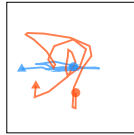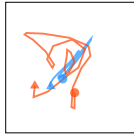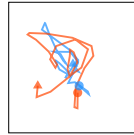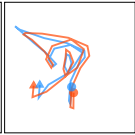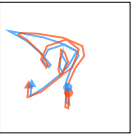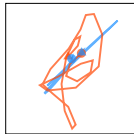
Extension for missing data. In practice, real-world data may be sparse, with many missing and few overlapping dimensions across the entire dataset. We can easily extend SGP-BAE to handle such datasets. We assume that any possible permutation of observed features is potentially missing, such that each high-dimensional observation $\boldsymbol{y}_n = \boldsymbol{y}_n^o \cup \boldsymbol{y}_n^u$ contains a set of observed features $\boldsymbol{y}_n^o$ and unobserved features $\boldsymbol{y}_n^u$. The likelihood term of the inference objective (Eq. 5.9) factorizes across data points and dimensions, so there is no major modification in this objective, as the summation of the likelihood term should be done only over the non-missing dimensions, i.e.:

$$p(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\varphi}) = \sum_{n=1}^{N} \log p(\boldsymbol{y}_n^o \mid \boldsymbol{z}_n, \boldsymbol{\varphi}). \quad (5.11)$$

5.5 Experiments

In this section, we provide empirical evidence that our SGP-BAE outperforms alternatives of combination between GP priors and AE models on synthetic data and real-world high-dimensional benchmark datasets. Throughout all experiments, unless otherwise specified, we use the radial basis function (RBF) kernel with automatic relevance determination (ARD) with marginal variance and independent lengthscales λ_i per feature (MacKay, 1996a). We place a lognormal prior with unit variance and means equal to 1 and 0.05 for the lengthscales and variance, respectively. Since the auxiliary data of most of the considered datasets are timestamps, we impose a non-informative uniform prior on the inducing inputs. We observe that this prior works well in our experiments. We set the hyperparameters of the number of SGHMC and optimization steps to $J = 30$, and $K = 50$, respectively. The details for all experiments are available in Appendix C.3.

Table 5.2: Reconstructions of the latent trajectories of moving ball. In the first column, frames of each test video are overlayed and shaded by time. Ground truth trajectories are illustrated in **orange**, while predicted trajectories are depicted in **blue**. We use $M = 10$ inducing points for the methods employed with sparse GPs.

5.5.1 Synthetic moving ball data

We begin our empirical evaluation by considering the moving ball dataset proposed by Pearce (2019). This dataset comprises grayscale videos showing the movement of a ball. The two-dimensional trajectory of the ball is simulated from a GP characterized by an RBF kernel. Our task is to reconstruct the underlying trajectory in the 2D latent space from the frames in pixel space. Unlike Jazbec et al. (2021), we generate a fixed number of 35 videos for training and another 35 videos for testing. It is still possible to perform full GP inference on such a small dataset. For this experiment, we consider full GP-based methods, such as Gaussian Process Bayesian Autoencoder (GP-BAE) and GP-VAE (Pearce, 2019), as oracles for the sparse variants. Because the dataset is quite small, we perform full-batch training/inference.e.

Benefits of moving away from variational inference   In this experiment, we show that, by relaxing strong assumptions on the posterior of latent space and taking advantage of a powerful scalable MCMC method, BAEs consistently outperform VAEs. Fig. 5.2 illustrates the performance of the considered methods in terms of root mean squared error (RMSE). The results show that our GP-BAE model performs much better than GP-VAE (Pearce, 2019) though both models use the same full GP priors. In addition, by treating inducing inputs and kernel hyperparameters of sparse GPs in a Bayesian fashion, SGP-BAE offers a rich modeling capability. This is evident in the improved performance of SGP-BAE compared to sparse variational Gaussian process VAE (SVGP-VAE) (Jazbec et al., 2021). SGP-BAE is able to approach the performance of GP-BAE despite using a small number of inducing points. These numerical results align with the qualitative evaluation of the reconstructed trajectories shown in Table 5.2. As expected, the standard BAE and VAE endowed with a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ prior on the latent space completely fail to model the trajectories faithfully. In contrast, GP-BAE and SGP-BAE are able to match them closely. In Appendix C.5, we further
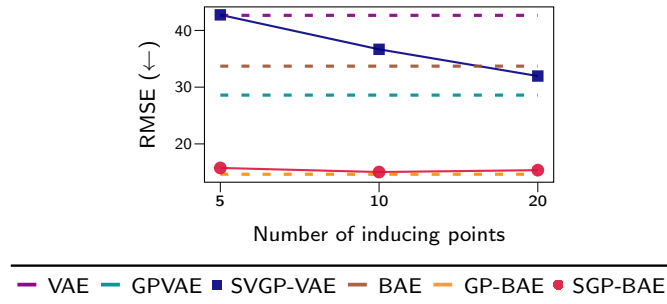
Figure 5.2: Performance of autoencoder models as a function of the number of inducing points.

show an ablation study on alternatives of Bayesian treatments for AEs and AE-style models with GP prior. This study ultimately demonstrates that our proposal offers superior performance.

Benefits of being fully Bayesian. Our SGP-BAE model has the same advantage as the SVGP-VAE (Jazbec et al., 2021) in that it allows for an arbitrary GP kernel, and the kernel hyperparameters and inducing inputs can be inferred jointly during training or inference. In contrast, other methods either use fixed GP priors (Pearce, 2019) or employ a two-stage approach (Casale et al., 2018), where the GP hyperparameters are optimized separately from the AEs. SVGP-VAE optimizes these hyperparameters using the common practice of maximization of the marginal likelihood, ML-II. This results in a point estimate of the hyperparameters but may be prone to overfitting, especially when the training data is small while there are many hyperparameters. A distinct advantage of SGP-BAE over SVGP-VAE is that it is fully Bayesian for the GP hyperparameters and inducing points. This not only improves the quality of the predictions but also offers sensible uncertainty quantification. Fig. 5.3 illustrates the posterior of the lengthscale. By using a sufficient number of inducing points and operating in a Bayesian way, SGP-BAE obtains a distribution over lengthscales that is compatible with observed data. When using too few inducing points, the model tends to estimate a larger lengthscale. This is expected as the effective lengthscale of the observed process in the subspace spanned by these few inducing points is larger. We also observe that the more inducing points are used, the more confident the posterior over the lengthscale is. Our method also produces a sensible posterior distribution on the inducing inputs, as shown in Fig. 5.4. The estimated inducing inputs are evenly spaced over the time dimension, which is reasonable since the latent trajectories are generated from stationary GPs.

5.5.2 Conditional generation of rotated MNIST

In the next experiment, we consider a large-scale benchmark of conditional generation. We follow the experimental setup from (Casale et al., 2018; Jazbec et al., 2021), where
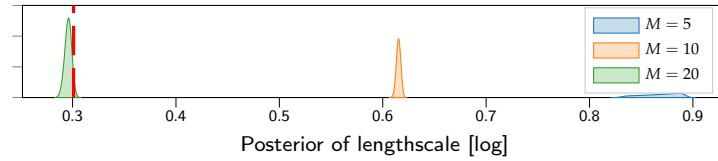
Figure 5.3: The posterior of the lengthscale corresponding to using a different number of inducing points, M. The **red** line denotes the true lengthscale.
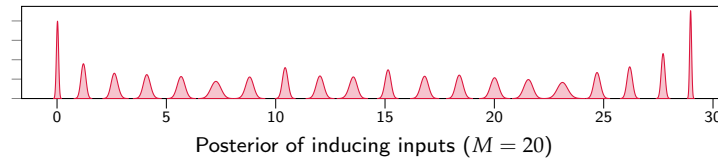
Figure 5.4: The posterior of the inducing position.

they use a rotated MNIST dataset ($N = 4050$). İn particular, we are given a set of images of digit three that have been rotated at different angles in $[0, 2\pi)$. Our goal is to generate a new image rotated at an unseen angle. As it is not trivial to apply full GP for such a large dataset, we omit GP-VAE (Pearce, 2019) and GP-BAE baselines. We consider the GPPVAE model Casale et al. (2018), which employs a low-rank approximation for the GP, and the SVGP-VAE model (Jazbec et al., 2021) as baselines. For both SVGP-VAE and SGP-BAE, we use a number of inducing points of $M = 32$ and a mini-batch size of $B = 256$. We also compare our method with the Conditional Variational Autoencoder (CVAE) of Sohn et al. (2015), which allows conditional generation tasks. Following Jazbec et al. (2021), we consider an extension of the sparse GP model (Hensman et al., 2013), named DEEP-SVIGP, that utilizes a deep likelihood parameterized by a neural network. As shown in Table 5.3, our SGP-BAE model generates images that are more visually appealing and most faithful to the ground truth compared to other approaches.

Table 5.3: Conditionally generated MNIST images. The right most column depicts the epistemic uncertainty obtained by our SGP-BAE model.

GT IMAGE	DEEP-SVIGP	(●) CVAE	(●) GPPVAE	(●) SVGP-VAE	(●) SGP-BAE	(●) VAR.
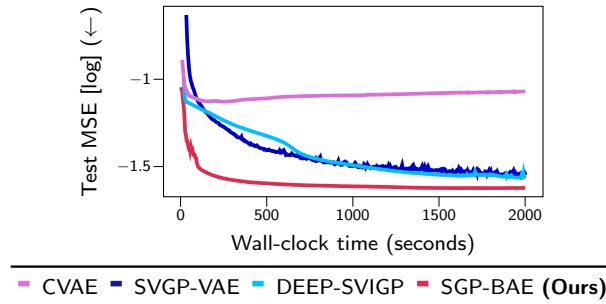
Figure 5.5: Comparison of test mean squared error (MSE) on the rotated MNIST dataset as function of training time.

Table 5.4: Results on the rotated MNIST digit 3 dataset. Here, we report the mean and standard deviation computed from 4 runs.

MODEL	MSE ($\downarrow$)
CVAE (Sohn et al., 2015)	0.0819 ± 0.0027
GPPVAE (Casale et al., 2018)	0.0351 ± 0.0005
SVGP-VAE (Jazbec et al., 2021)	0.0257 ± 0.0004
SGP-BAE (OURS)	$\mathbf{0.0228 \pm 0.0004}$
DEEP-SVIGP (Jazbec et al., 2021)	0.0236 ± 0.0010

Performance on conditional generation. Table 5.4 presents the quantitative evaluation of the conditionally generated images in terms of MSE. Our SGP-BAE model clearly outperforms the other competing methods. It is worth noting that DEEP-SVIGP (Hensman et al., 2013) does not use an amortization mechanism, and its performance is considered to be an upper bound for that of SVGP-VAE. As discussed by Jazbec et al. (2021), DEEP-SVIGP can be used for conditional generation tasks, where the goal is to impose a single GP over the entire dataset, and therefore amortization is not necessary. However, this model cannot be used in tasks where inference has to be amortized across multiple GPs, such as learning interpretable data representations.

Computational efficiency. Similarly to the competing methods that use sparse approximations such as SVGP-VAE and DEEP-SVIGP, each iteration of GP-BAE involves the computation of the inverse covariance matrix, resulting in a time complexity of $\mathcal{O}(M^3)$. Fig. 5.5 shows the convergence in terms of test MSE for the competing methods and our SGP-BAE, trained for a fixed training time budget. We omit GPPVAE (Casale et al., 2018) from this comparison, as reported by Jazbec et al. (2021), which is significantly slower than the sparse methods. This result demonstrates that SGP-BAE converges remarkably faster in terms of wall-clock time while achieving better final predictive performance.

Epistemic uncertainty quantification. Unlike the competing methods, our SGP-BAE model can capture the epistemic uncertainty of the decoder thanks to the
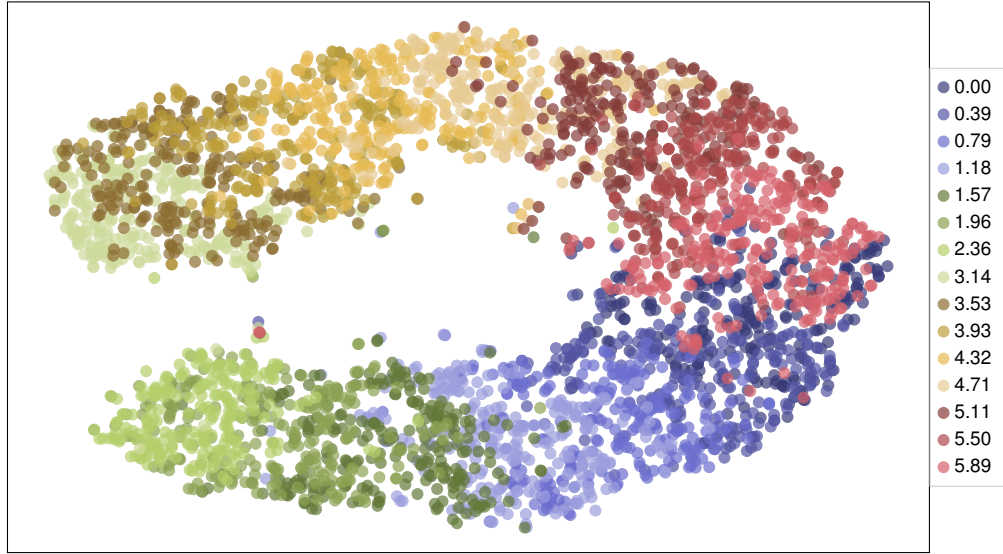
Figure 5.6: Visualization of t-SNE embeddings (Maaten and Hinton, 2008) of SGP-BAE latent vectors on the training data of the rotated MNIST. Each image embedding is colored with respect to its associated angle. Here, we use a perplexity parameter of 50 for t-SNE.

Table 5.5: A comparison between methods of multi-output GP models and GP autoencoders on the EEG and JURA datasets.

DATASET	METRIC	IGP	GPAR	(●) SGP-VAE	(●) **SGP-BAE**	**DSGP-BAE**
EEG	SMSE ($\downarrow$)	1.70 ± 0.14	0.28 ± 0.00	0.52 ± 0.05	$\mathbf{0.22 \pm 0.01}$	$\mathbf{0.21 \pm 0.01}$
	NLL ($\downarrow$)	2.59 ± 0.02	$\mathbf{1.68 \pm 0.01}$	1.98 ± 0.02	1.96 ± 0.08	2.25 ± 0.13
JURA	MAE ($\downarrow$)	0.57 ± 0.00	$\mathbf{0.42 \pm 0.01}$	0.54 ± 0.01	$\mathbf{0.45 \pm 0.03}$	$\mathbf{0.44 \pm 0.02}$
	NLL ($\downarrow$)	1563.42 ± 166.55	17.81 ± 1.06	1.02 ± 0.01	$\mathbf{0.91 \pm 0.04}$	$\mathbf{0.85 \pm 0.04}$

Bayesian treatment and the use of powerful inference methods. This can improve the quality of uncertainty quantification on reconstructed or generated images. As shown in the rightmost column of Table 5.3, SGP-BAE can provide sensible epistemic uncertainty quantification. Our model exhibits increased uncertainty for semantically and visually challenging pixels, such as the boundaries of the digits.

Latent space visualization. Fig. 5.6 illustrates a two-dimensional t-SNE (Maaten and Hinton, 2008) visualization of latent vectors ($C = 16$) for the rotated MNIST data obtained by our SGP-BAE model. It is evident that the clusters of embeddings are organized in a structured manner according to the angles they represent. Specifically, the embeddings of rotated images are arranged in a continuous sequence from 0 to 2π in a clockwise direction.

5.5.3 Missing data imputation

Next, we consider the task of imputing missing data on multi-output spatio-temporal datasets. We compare our method against Sparse GP-VAE (SGP-VAE) (Ashman et al.,
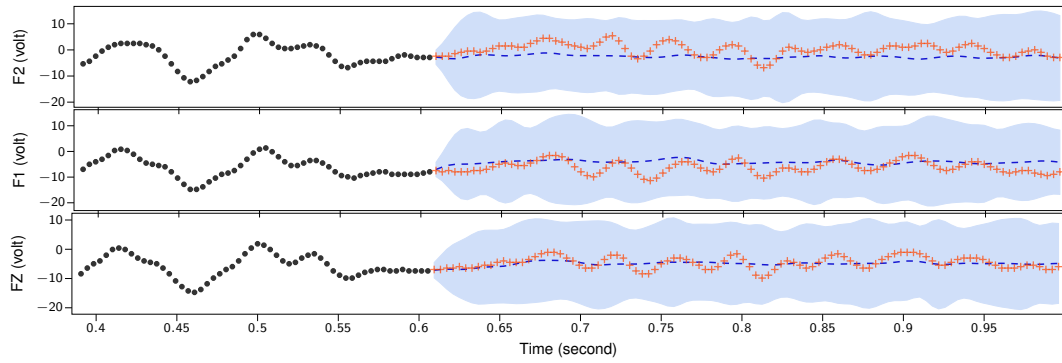
(a) Independent Gaussian Processes (IGP)
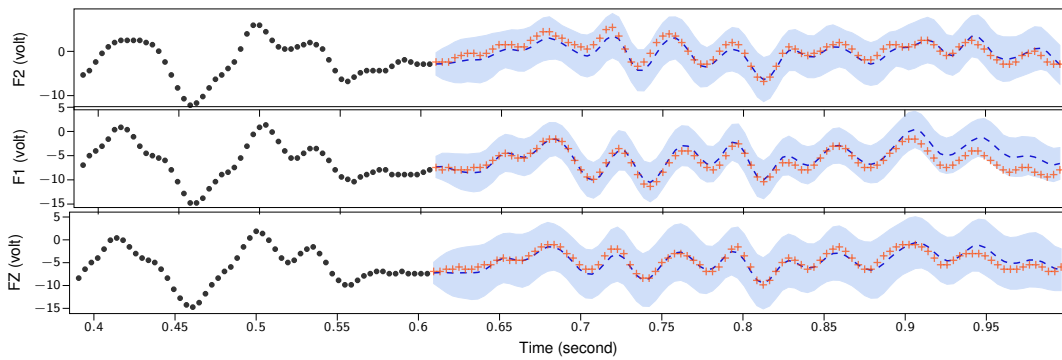
(b) Gaussian Process Autoregressive Model (GPAR)
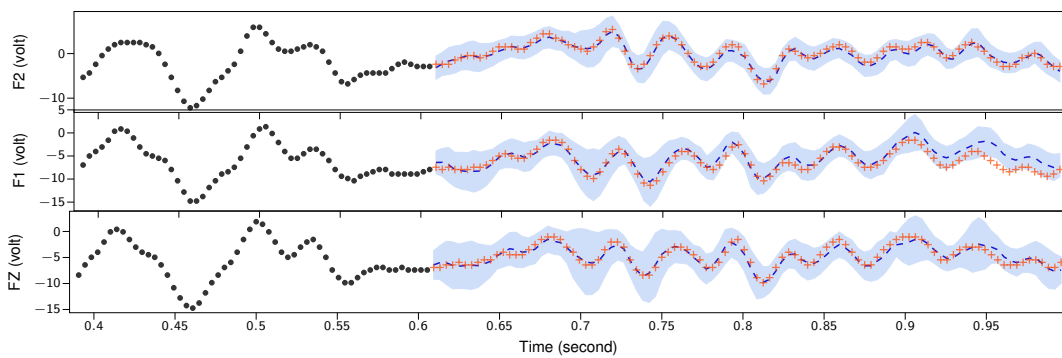
(c) SGP-BAE

Figure 5.7: Visualization of predictions for missing data of the EEG dataset. Each panel shows one of the three channels with missing data (orange crosses) and observed data (black points).

2020) and multi-output GP methods such as Independent GPs (IGP) and Gaussian Process Autoregressive Regression (GPAR) (Requeima et al., 2019). Additionally, we consider the DSGP-BAE model, which is an extension of our SGP-BAE model endowed with 3-layer deep GP prior. For a fair comparison, we treat partially missing data as zeros to feed into the inference network (encoder) for SGP-VAE and our SGP-BAE model. We leave the adoption of partial inference networks (Ashman et al., 2020) to our model for future work. We follow the experimental setup of Requeima et al. (2019) and Ashman et al. (2020) and use two standard datasets for this comparison.

Electroencephalogram (EEG). This dataset comprises 256 measurements taken over one second. Each measurement consists of seven electrodes, FZ and F1-F6, placed on the patient's scalp ($x_n \in \mathbb{R}^1$, $y_n \in \mathbb{R}^7$). The goal is to predict the last 100 samples for electrodes FZ, F1, and F2, given that the first 156 samples of FZ, F1, and F2 and the whole signals of F3-F6 are observed. Performance is measured with the standardized mean squared error (SMSE) and negative log-likelihood (NLL).

Jura. This is a geospatial dataset consisting of 359 measurements of the topsoil concentrations of three metals — Nickel, Zinc, and Cadmium — collected from a region of Swiss Jura ($x_n \in \mathbb{R}^2$, $y_n \in \mathbb{R}^3$). The dataset is split into a training set comprised of Nickel and Zinc measurements for all 359 locations and Cadmium measurements for just 259 locations. The task is to predict the Cadmium measurements at the remaining 100 locations conditioned on the observed training data. Performance is evaluated with the mean absolute error (MAE) and negative log-likelihood.

Table 5.5 compares the performance of SGP-BAE to the competing methods. As mentioned in Ashman et al. (2020), GPAR is the state-of-the-art method for these datasets. We find that our SGP-BAE and DSGP-BAE models perform comparably with GPAR on the EEG dataset but better on the JURA dataset. A significant advantage of GP autoencoder methods is that they model P outputs using only C latent GPs, while GPAR uses P GPs. This can be beneficial when the dimensionality of the data, P, is very high. Similarly to the previous experiments, SGP-BAE consistently performs better than variational approximation-based methods such as SGP-VAE (Ashman et al., 2020). As expected, IGP is the worst-performing method due to its inability to model the correlations between output variables. For completeness, we show the predictive mean and uncertainty estimation for the missing values of the EEG datase in Fig. 5.7.

We find that the utilization of deep Gaussian process (DGP) priors yields only marginal improvement on the EEG and JURA datasets, as shown in Table 5.5. In addition, we observe that DSGP-BAE just performs comparably to SGP-BAE on the moving ball dataset. This can be attributed to the fact that the correlation between the latent variables of these datasets is sufficiently simple to be modeled using shallow

GPs. For instance, in the experiments conducted on the moving ball dataset, the data is generated from a GP, following the setup used in the previous work (Pearce, 2019; Jazbec et al., 2021). Nevertheless, when dealing with more complex datasets, we believe that the flexibility offered by DGPs can prove beneficial for modeling intricate patterns (e.g., discontinuities or strong non-stationarities) of the latent process.

5.6 Conclusions

We have introduced our novel SGP-BAE that integrates fully Bayesian sparse GPs on the latent space of Bayesian autoencoders. Our proposed model is generic, as it allows an arbitrary GP kernel and deep GPs. The inference for this model is carried out by a powerful and scalable SGHMC sampler. Through a rigorous experimental campaign, we have demonstrated the excellent performance of SGP-BAE on a wide range of representation learning and generative modeling tasks.

Limitations and future works. While our model's ability to learn disentangled representations has been demonstrated through empirical evidence, there is a need to establish a theoretical grounding for the disentanglement of its latent space. Furthermore, it would be useful to study the amortization gap (Cremer et al., 2018; Krishnan et al., 2018; Marino et al., 2018) of our model. Additionally, exploring more informative priors for the decoder's parameters (Tran et al., 2021), beyond the isotropic Gaussian prior used in this work, is also worthwhile. There are potential avenues for future research. One of them is the fully Bayesian treatment of the auxiliary data X. This approach can be beneficial when the auxiliary data is unavailable or contains missing values. In addition, it would be interesting to apply the model to downstream applications where modeling correlations between data points and uncertainty quantification are required, such as in bioinformatics and climate modeling.

Chapter 6

Improving Training of Likelihood-based Generative Models with Data Mollification

Generative models have attracted considerable attention due to their tremendous success in various domains, such as computer vision where they are capable to generate impressive realistic-looking images. Likelihood-based generative models (GMs) are attractive due to the possibility to generate new data by a single model evaluation. However, they typically achieve lower sample quality compared to state-of-the-art score-based diffusion models (DMs). The work presented in this chapter provides a significant step in the direction of addressing this limitation. The idea is to borrow one of the strengths of score-based DMs, which is the ability to perform accurate density estimation in low-density regions and to address manifold overfitting by means of data mollification. We connect data mollification through the addition of Gaussian noise to Gaussian homotopy, which is a well-known technique to improve optimization. Data mollification can be implemented by adding one line of code in the optimization loop, and we demonstrate that this provides a boost in generation quality of likelihood-based GMs, without computational overheads. We report results on image datasets with popular likelihood-based GMs, including variants of variational autoencoders and normalizing flows, showing large improvements in FID score.

6.1   Introduction

GMs have attracted considerable attention recently due to their tremendous success in various domains, such as computer vision, graph generation, physics and

reinforcement learning (see e.g., Kingma and Welling, 2019; Kobyzev et al., 2021, and references therein). Given a set of data points, GMs attempt to characterize the distribution of such data so that it is then possible to draw new samples from this. Popular approaches include variational autoencoders (VAEs), normalizing flows (NFs), generative adversarial networks (GANs), and score-based diffusion models (DMs).

In general, the goal of any GMs is similar to that of density estimation with the additional aim to do so by constructing a parametric mapping between an easy-to-sample-from distribution p_s and the desired data distribution p_{data}. While different GMs approaches greatly differ in their optimization strategy and formulation, the underlying objectives share some similarity due to their relation to the optimal transport problem, defined as $\arg\min_\pi \int \|x - y\|^2 d\pi(x, y)$. Here π is constrained to belong to the set of joint distributions with marginals p_s, p_{data}, respectively (Genevay et al., 2017; Liutkus et al., 2019). This unified perspective is explicitly investigated for GANs and VAEs (Genevay et al., 2017) for example, whereas other works study NFs (Onken et al., 2021). Similarly, DMs can be connected to Schrodinger Bridges (Chen et al., 2022), which solve the problem of *entropy-regularized* optimal transport (Pavon et al., 2021). Given that extensions of the regularized optimal transport case are available also for other generative models (Sanjabi et al., 2018; Reshetova et al., 2021), we should expect that, in principle, any technique should allow generation of samples with similar quality, provided it is properly tuned. However, this is not true in practice. The different formulations lead to a variety of properties associated with GMs, and pros and cons of each formulation can be understood through the so-called GM tri-lemma (Xiao et al., 2022). The three desirable properties of GMs are high sample quality, mode coverage, and fast sampling, and it has been argued that such goals are difficult to be satisfied simultaneously (Xiao et al., 2022) .

The state-of-the-art is currently dominated by score-based DMs, due to their ability to achieve high sample quality and good mode coverage. However, generating new samples is computationally expensive due to the need to simulate stochastic differential equations. Likelihood-based GMs are complementary, in that they achieve lower sample quality, but sampling requires one model evaluation per sample and it is therefore extremely fast. While some attempts have been made to bridge the gap by combining GANs with DMs (Xiao et al., 2022) or training GANs with diffusions (Wang et al., 2023), these still require careful engineering of architectures and training schedules. The observation that all GMs share a common underlying objective indicates that we should look at what makes DMs successful at optimizing their objective. Then, the question we address in this work is: can we borrow the strengths of score-based DMs to improve likelihood-based GMs, without paying the price of costly sample generation?

One distinctive element of score-based DMs is data mollification, which is typically achieved by adding Gaussian noise (Song and Ermon, 2019) or, in the context of

image datasets, by blurring (Rissanen et al., 2023). A large body of evidence points to the *manifold hypothesis* (Roweis and Saul, 2000), which states that the intrinsic dimensionality of image datasets is much lower than the dimensionality of their input. Density estimation in this context is particularly difficult because of the degeneracy of the likelihood for any density concentrated on the manifold where data lies (Loaiza-Ganem et al., 2022b). Under the manifold hypothesis, or even when the target density is multi-modal, the Lipschitz constant of GMs has to be large, but regularization, which is necessary for robustness, is antagonist to this objective (Salmona et al., 2022; Cornish et al., 2020). As we will study in detail in this work, the process of data mollification gracefully guides the optimization mitigating manifold overfitting and enabling a desirable tail behavior, yielding accurate density estimation in low-density regions. In likelihood-based GMs, data mollification corresponds to smoothing the loss landscape. In the optimization literature, mollification (a.k.a. Gaussian homotopy or graduated optimization) has been proven to accelerate optimization. The literature provides results for stochastic non-convex problems (Hazan et al., 2016), which is the setting we consider here. Another recent work in the same setting gives some continuity results on the optimum under small increments in the level of mollification (Gargiani et al., 2020).

Strictly speaking, data mollification in score-based DMs and likelihood-based GMs are slightly different. In the latter, the amount of noise injected in the data is continuously annealed throughout training. At the beginning, the equivalent loss landscape seen by the optimizer is much smoother, due to the heavy perturbation of the data, and a continuous reduction of the noise level allows optimization to be gracefully guided until the point where the level of noise is zero (Hazan et al., 2016; Gargiani et al., 2020). DMs, instead, are trained at each step of the optimization process by considering **all** noise levels simultaneously, where complex amortization procedures, such as self-attention (Song et al., 2021), allow the model to efficiently share parameters across different perturbation levels. It is also worth mentioning that score-based DMs possess another distinctive feature in that they perform gradient-based density estimation (Song and Ermon, 2019; Hyvärinen, 2005). It has been conjectured that this can be helpful to avoid manifold overfitting by allowing for the modeling of complex densities while keeping the Lipschitz constant of score networks low (Salmona et al., 2022). In this work, we attempt to verify the hypothesis that data mollification is heavily responsible for the success of score-based DMs. We do so by proposing data mollification for likelihood-based GMs, and provide theoretical arguments and experimental evidence that data mollification consistently improves their optimization. Crucially, this strategy yields better sample quality and it is extremely easy to implement, as it requires adding very little code to any existing optimization loop.

We consider a large set of experiments involving VAEs and NFs and some popular image datasets. These provide a challenging test for likelihood-based GMs due to the large dimensionality of the input space and to the fact that density estimation

needs to deal with data lying on manifolds. The results show systematic, and in some cases dramatic, improvements in sample quality, indicating that this is a simple and effective strategy to improve optimization of likelihood-based GMs models. This chapter is organized as follows: in § 6.2 we illustrate the challenges associated with generative modeling when data points lie on a manifold, particularly with density estimation in low-density regions and manifold overfitting; in § 6.3 we propose data mollification to address these challenges; § 6.4 reports the experiments with a discussion of the limitations and the broader impact, while § 6.5 presents related works, and § 6.6 concludes the chapter.

6.2 Challenges in Training Deep Generative Models

We are interested in unsupervised learning, and in particular on the task of density estimation. Given a dataset $\mathcal{D}$ consisting of N i.i.d samples $\mathcal{D} \stackrel{\Delta}{=} \{x_i\}_{i=1}^N$ with $x_i \in \mathbb{R}^D$, we aim to estimate the unknown continuous generating distribution $p_{\text{data}}(x)$. In order to do so, we introduce a model $p_\theta(x)$ with parameters θ and attempt to estimate θ based on the dataset $\mathcal{D}$. A common approach to estimate θ is to maximize the likelihood of the data, which is equivalent to minimizing the following objective:

$$\mathcal{L}(\theta) \stackrel{\Delta}{=} -\mathbb{E}_{p_{\text{data}}(x)} \left[\log p_\theta(x) \right]. \tag{6.1}$$

There are several approaches to parameterize the generative model $p_\theta(x)$. In this work, we focus on two widely used likelihood-based generative models (GMs), which are normalizing flows (NFs) (Papamakarios et al., 2021; Kobyzev et al., 2021) and variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende and Mohamed, 2015). Although NFs and VAEs are among the most popular deep GMs, they are characterized by a lower sample quality compared to GANs and score-based DMs. In this section, we present two major reasons behind this issue by relying on the manifold hypothesis.

6.2.1 The manifold hypothesis and density estimation in low-density regions

The manifold hypothesis is a fundamental concept in manifold learning (Roweis and Saul, 2000; Tenenbaum et al., 2000; Bengio et al., 2013b) stating that real-world high-dimensional data tend to lie on a manifold $\mathcal{M}$ characterized by a much lower dimensionality compared to the one of the input space (ambient dimensionality) (Narayanan and Mitter, 2010). This has been verified theoretically and empirically for many applications and datasets (Ozakin and Gray, 2009; Narayanan and Mitter, 2010; Pope et al., 2021; Tempczyk et al., 2022). For example, Pope et al. (2021)
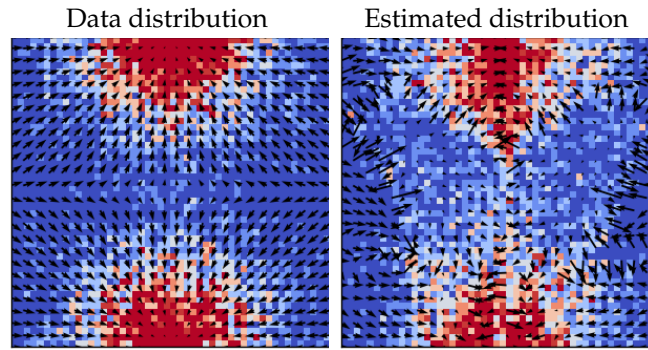
Figure 6.1: Left: Histogram of samples from data distribution $p_{\text{data}}(\boldsymbol{x})$ and its true scores $\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})$; **Right:** Histogram of of samples from the estimated distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and its scores $\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$. In the low density regions, the model is unable to capture the true density and scores.

report extensive evidence that natural image datasets have indeed very low intrinsic dimension relative to the high number of pixels in the images.

The manifold hypothesis suggests that density estimation in the input space is challenging and ill-posed. In particular, data points on the manifold should be associated with high density, while points outside the manifold should be considered as lying in regions of nearly zero density (Meng et al., 2021). This implies that the target density in the input space should be characterized by high Lipschitz constants. The fact that data is scarce in regions of low density makes it difficult to expect that models can yield accurate density estimation around the tails. These pose significant challenges for the training of deep GMs (Cornish et al., 2020; Meng et al., 2021; Song and Ermon, 2019). Recently, diffusion models (Song and Ermon, 2019; Ho et al., 2020; Song et al., 2021) have demonstrated the ability to mitigate this problem by gradually transforming a Gaussian distribution, whose support spans the full input space, into the data distribution. This observation induces us to hypothesize that the data mollification mechanism in score-based DMs is responsible for superior density estimation in low-density regions.

To demonstrate the challenges associated with accurate estimation in low-density regions, we consider a toy experiment where we use a REAL-NVP flow (Dinh et al., 2017a) to model a two-dimensional mixture of Gaussians, which is a difficult test for NFs in general. Details on this experiment are provided in the Appendix. Fig. 6.1 depicts the true and estimated densities, and their corresponding scores, which are the gradient of the log-density function with respect to the data (Hyvärinen, 2005). Note that the use of "score" here is slightly different from that from traditional statistics where score usually refers to the gradient of the log-likelihood with respect to model parameters. As it can be seen in the figure, in regions of low data density, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is completely unable to model the true density and scores. This problem is due to the lack of data samples in these regions and may be more problematic under the manifold hypothesis and for high-dimensional data such as images. In § 6.4, we will

demonstrate how it is possible to considerably mitigate this issue by means of data mollification.

6.2.2   Manifold overfitting

The manifold hypothesis suggests that overfitting on a manifold can occur when the model $p_\theta(x)$ assigns an arbitrarily large likelihood in the vicinity of the manifold, even if the distribution does not accurately capture the true distribution $p_{\text{data}}(x)$ (Dai and Wipf, 2019; Loaiza-Ganem et al., 2022b). This issue is illustrated in Fig. 2 of Loaiza-Ganem et al. (2022b) and it will be highlighted in our experiment (§ 6.4.1), where the true data distribution $p_{\text{data}}(x)$ is supported on a one-dimensional curve manifold $\mathcal{M}$ in two-dimensional space $\mathbb{R}^2$. Even when the model distribution $p_\theta(x)$ poorly approximates $p_{\text{data}}(x)$, it may reach a high likelihood value by concentrating the density around the correct manifold $\mathcal{M}$. If $p_\theta(x)$ is flexible enough, any density defined on $\mathcal{M}$ may achieve infinite likelihood and this might be an obstacle for retrieving $p_{\text{data}}(x)$.

A theoretical formalization of the problem of manifold overfitting appears in (Loaiza-Ganem et al., 2022b) and it is based on the concept of Riemannian measure (Pennec, 2006). The Riemannian measure on manifolds holds an analogous role to that of the Lebesgue measure on Euclidean spaces. To begin, we establish the concept of smoothness for a probability measure on a manifold.

Definition 1. *Let $\mathcal{M}$ be a finite-dimensional manifold, and p be a probability measure on $\mathcal{M}$. Let $\mathfrak{g}$ be a Riemannian metric on $\mathcal{M}$ and $\mu_\mathcal{M}^{(\mathfrak{g})}$ the corresponding Riemannian measure. We say that p is* smooth *if $p \ll \mu_\mathcal{M}^{(\mathfrak{g})}$ and it admits a continuous density $p : \mathcal{M} \to \mathbb{R}_{>0}$ with respect to $\mu_\mathcal{M}^{(\mathfrak{g})}$.*

We now report Theorem 1 from Loaiza-Ganem et al. (2022b) followed by a discussion on its implications for our work.

Theorem 1. *(Loaiza-Ganem et al., 2022b). Let $\mathcal{M} \subset \mathbb{R}^D$ be an analytic d-dimensional embedded submanifold of $\mathbb{R}^d$ with $d < D$, μ_D is the Lebesgue measure on $\mathbb{R}^D$, and $p^\dagger$ a smooth probability measure on $\mathcal{M}$. Then there exists a sequence of probability measures $\{p_\theta^{(t)}\}_{t=0}^\infty$ on $\mathbb{R}^D$ such that:*

1. $p_\theta^{(t)} \to p^\dagger$ *as* $t \to \infty$.

2. $\forall t \geq 0, p_\theta^{(t)} \ll \mu_D$ *and* $p_\theta^{(t)}$ *admits a density* $p_\theta^{(t)} : \mathbb{R}^D \to \mathbb{R}_{>0}$ *with respect to* μ_D *such that:*

   (a) $\lim_{t\to\infty} p_\theta^{(t)}(x) = \infty, \forall x \in \mathcal{M}$.

   (b) $\lim_{t\to\infty} p_\theta^{(t)}(x) = 0, \forall x \notin cl(\mathcal{M})$, *where $cl(\cdot)$ denotes closure in $\mathbb{R}^D$.*

Theorem 1 holds for any smooth probability measure supported in $\mathcal{M}$. This is an important point because this includes the desired p_{data}, provided that this is smooth too. The key message in Loaiza-Ganem et al. (2022b) is that, a-priori, there is no reason to expect that for a likelihood-based model to converge to p_{data} out of all

the possible $p^\dagger$. Their proof is based on convolving $p^\dagger$ with a Gaussian kernel with variance σ_t^2 that decreases to zero as $t \to \infty$, and then verify that the stated properties of $p_{\boldsymbol{\theta}}^{(t)}$ hold. Our analysis, while relying on the same technical tools, is instead constructive in explaining why the proposed data mollification allows us to avoid manifold overfitting. The idea is as follows: at time step $t = 0$, we select the desired p_{data} convolved with a Gaussian kernel with a large, but finite, variance $\sigma^2(0)$ as the target distribution for the optimization. Optimization is performed and $p_{\boldsymbol{\theta}}^{(0)}$ targets this distribution, without any manifold overfitting issues, since the dimensionality of the corrupted data is non-degenerate. At the second step, the target distribution is obtained by convolving p_{data} with the kernel with variance $\sigma^2(1) < \sigma^2(0)$, and again manifold overfitting is avoided. By iteratively repeating this procedure, we can reach the point where we are matching a distribution convolved with an arbitrarily small variance $\sigma^2(t)$, without ever experiencing manifold overfitting. We argue that when removing the last bit of perturbation, the parametric distribution will match p_{data}, without overfitting; this is equivalent to a good initialization method. Under a different perspective, the annealing procedure introduces a memory effect in the optimization process which is important for the success of mollification. Indeed, as mentioned in Loaiza-Ganem et al. (2022b) and verified by ourselves in earlier investigations, a small constant amount of noise does not provide any particular benefits over the original scheme.

6.3 Generative Models with Data Mollification

Motivated by the aforementioned problems with density estimation in low-density regions and manifold overfitting, we propose a simple yet effective approach to improve likelihood-based GMs. Our method involves mollifying data using Gaussian noise, gradually reducing its variance, until recovering the original data distribution $p_{\text{data}}(\boldsymbol{x})$. This mollification procedure is similar to the reverse process of diffusion models, where a prior noise distribution is smoothly transformed into the data distribution (Song and Ermon, 2019; Ho et al., 2020; Song et al., 2021). As already mentioned, data mollification alleviates the problem of manifold overfitting and it induces a memory effect in the optimization which improves density estimation in regions of low density.

Algorithm 4: Gaussian mollification

1 **for** $t \leftarrow 1, 2, ..., T$ **do**
2    $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$ // `Sample training data`
3    $\tilde{\boldsymbol{x}}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\varepsilon}$ // `Mollify data with` $\alpha_t, \sigma_t^2 \leftarrow \gamma(t/T)$ `and` $\varepsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
4    $\boldsymbol{\theta}_t \leftarrow \text{UPDATE}(\boldsymbol{\theta}_{t-1}, \tilde{\boldsymbol{x}}_t)$ // `Train model`

$$\tilde{\mathbf{x}}_0 \qquad\qquad\qquad\qquad\qquad\qquad \tilde{\mathbf{x}}_T$$

Training Iteration
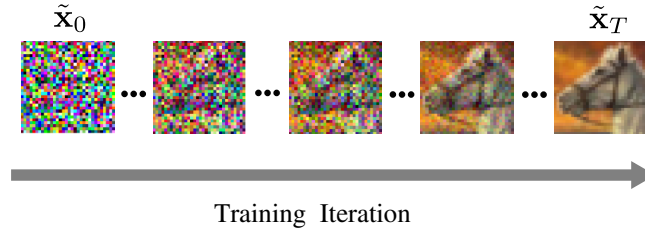
Figure 6.2: Illustration of Gaussian mollification, where $\tilde{x}_t$ is the mollified data at iteration t.

Gaussian Mollification.   Given that we train the model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ for T iterations, we can create a sequence of progressively less smoothed versions of the original data $\boldsymbol{x}$, which we refer to as mollified data $\tilde{x}_t$. Here, t ranges from $t = 0$ (the most mollified) to $t = T$ (the least mollified). For any $t \in [0, T]$, the distribution of the mollified data $\tilde{x}_t$, conditioned on $\boldsymbol{x}$, is given as follows:

$$q(\tilde{x}_t \mid \boldsymbol{x}) = \mathcal{N}(\tilde{x}_t; \alpha_t \boldsymbol{x}, \sigma_t^2 \boldsymbol{I}), \tag{6.2}$$

where α_t and σ_t^2 are are positive scalar-valued functions of t. In addition, we define the signal-to-noise ratio $\mathrm{SNR}(t) = \alpha_t^2/\sigma_t^2$. and we assume that it monotonically increases with t, i.e., $\mathrm{SNR}(t) \leq \mathrm{SNR}(t+1)$ for all $t \in [0, T-1]$. In other words, the mollified data $\tilde{x}_t$ is progressively less smoothed as t increases. In this work, we adopt the *variance-preserving* formulation used for diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Kingma et al., 2021), where $\alpha_t = \sqrt{1 - \sigma_t^2}$ and $\sigma_t^2 = \gamma(t/T)$. Here, $\gamma(\cdot)$ is a monotonically decreasing function from 1 to 0 that controls the rate of mollification. Intuitively, this procedure involves gradually transforming an identity-covariance Gaussian distribution into the distribution of the data. Algorithm 4 summarizes the proposed Gaussian mollification procedure, where the red line indicates a simple additional step required to mollify data compared with vanilla training using the true data distribution.

Noise schedule.   The choice of the noise schedule $\gamma(\cdot)$ has an impact on the performance of the final model. In this work, we follow common practice in designing the noise schedule based on the literature of score-based DMs (Nichol and Dhariwal, 2021; Hoogeboom et al., 2023; Chen, 2023). In particular, we adopt a sigmoid schedule (Jabri et al., 2022), which has recently been shown to be more effective in practice compared to other choices such as linear (Ho et al., 2020) or cosine schedules (Nichol and Dhariwal, 2021).

The sigmoid schedule $\gamma(t/T)$ (Jabri et al., 2022) is defined through the sigmoid function:

$$\mathrm{sigmoid}\left(-\frac{t/T}{\tau}\right), \tag{6.3}$$

where τ is a temperature hyper-parameter. This function is then scaled and shifted to ensure that $\gamma(0) = 1$ and $\gamma(1) = 0$. We encourage the reader to check the
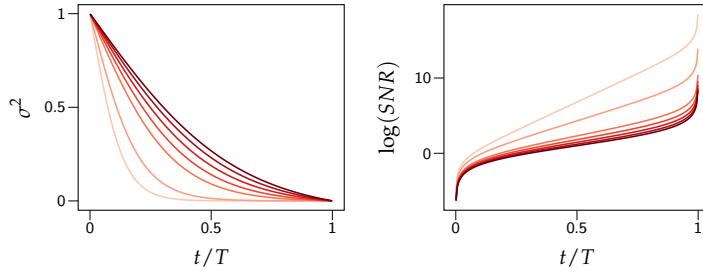
Figure 6.3: Illustration of sigmoid schedule and the corresponding log(SNR). The temperature values from 0.2 to 0.9 are progressively shaded, with the lighter shade corresponding to lower temperatures.

implementation of this schedule, available in Appendix. Fig. 6.3 illustrates the sigmoid schedule and the corresponding log(SNR) with different values of τ. We use a default temperature of 0.7 as it demonstrates consistently good results in our experiments.

6.4 Experiments

In this section, we demonstrate emprically the effectiveness of our proposal through a wide range of experiments on synthetic data and some popular real-world image datasets. The Appendix contains a detailed description of each experiment to guarantee reproducibility.

6.4.1 2D Synthetic Data Sets

We begin our experimental campaign with two 2D synthetic datasets. The two-dimensional nature of these datasets allows us to demonstrate the effectiveness of Gaussian mollification in mitigating the challenges associated with density estimation in low-density regions and manifold overfitting. Here, we consider $p_\theta(x)$ to be a REAL-NVP flow (Dinh et al., 2017a), which comprises five coupling bijections, each consisting of a two-hidden layer multilayer perceptron (MLP). To assess the capability of $p_\theta(x)$ to recover the true data distribution, we use maximum mean discrepancy (MMD) (Gretton et al., 2012) with a radial basis function (RBF) kernel on a held-out set. In these experiments, we employ the Gaussian mollification strategy presented in the previous section and compare the estimated density with the *vanilla* approach where we use the original training data without any mollification.

Mixture of Gaussians.   First, we consider a target distribution that is a mixture of two Gaussians, as depicted in Fig. 6.4. As discussed in § 6.2.1, the vanilla training procedure fails to accurately estimate the true data distribution and scores, particularly in the low-density regions. The estimated densities and the mollified data during
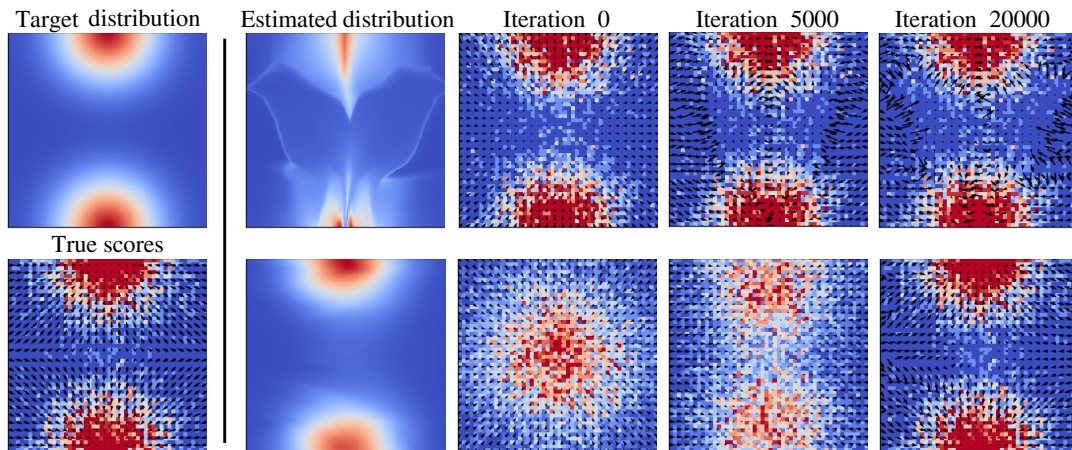
Figure 6.4: The first column shows the target distribution and the true scores. The second column depicts the estimated distributions of the Gaussian mixture model (GMM) , which yield MMD^2 of 15.5 and 2.5 for the vanilla (top) and mollification (bottom) training, respectively. The remaining columns show histogram of samples from the true (**top row**) and mollified data (**bottom row**), and estimated scores.
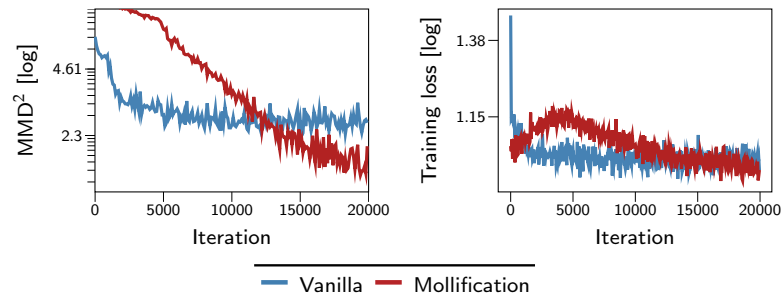
Figure 6.5: The learning curves of the GMM experiments.
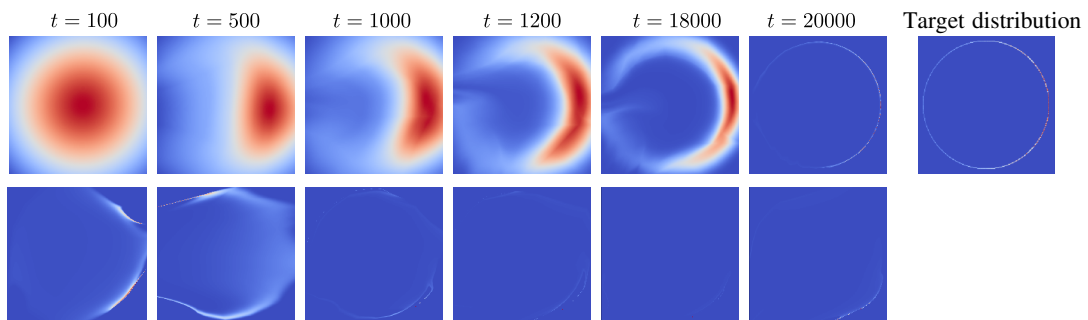
Figure 6.6: The progression of the estimated densities for the von Mises distribution from the vanilla (**bottom row**) and our mollification (**top row**) approaches.

Table 6.1: FID score on CIFAR10 and CELEBA dataset (*lower is better*). The small colored numbers indicate improvement or degration of the mollification training compared to the vanilla training.

MODEL	CIFAR10			CELEBA		
	VANILLA	GAUSSIAN MOLLIFCATION	BLURRING MOLLIFICATION	VANILLA	GAUSSIAN MOLLIFICATION	BLURRING MOLLIFICATOIN
REAL-NVP	131.15	121.75 ↓7.17%	120.88 ↓7.83%	81.25	79.68 ↓1.93%	85.40 ↑5.11%
GLOW	74.62	64.87 ↓13.07%	66.70 ↓10.61%	97.59	70.91 ↓27.34%	74.74 ↓23.41%
VAE	191.98	155.13 ↓19.19%	175.40 ↓8.64%	80.19	72.97 ↓9.00%	77.29 ↓3.62%
VAE-IAF	193.58	156.39 ↓19.21%	162.27 ↓16.17%	80.34	73.56 ↓8.44%	75.67 ↓5.81%
IWAE	183.04	146.70 ↓19.85%	163.79 ↓10.52%	78.25	71.38 ↓8.78%	76.45 ↓2.30%
β-VAE	112.42	93.90 ↓16.47%	101.30 ↓9.89%	67.78	64.59 ↓4.71%	67.08 ↓1.03%
HVAE	172.47	137.84 ↓20.08%	147.15 ↓14.68%	74.10	72.28 ↓2.46%	77.54 ↑4.64%

the training are depicted in Fig. 6.4. Initially, the mollification process considers a simpler coarse-grained version of the target density, which is easy to model. This is demonstrated by the low training loss at the beginning of the optimization, as depicted in Fig. 6.5. Subsequently, the method gradually reduces the level of noise allowing for a progressive refinement of the estimated versions of the target density. This process uses the solution from one level of mollification as a means to guiding optimization for the next. As a result, Gaussian mollification facilitates the recovery of the modes and enables effective density estimation in low-density regions. The vanilla training procedure, instead, produces a poor estimate of the target density, as evidenced by the trace-plot of the MMD^2 metric in Fig. 6.5 and the visualization of the scores in Fig. 6.4.

Von Mises distribution. We proceed with an investigation of the von Mises distribution on the unit circle, as depicted in Fig. 6.6, with the aim of highlighting the issue of manifold overfitting (Loaiza-Ganem et al., 2022b). In this experiment, the data lies on a one-dimensional manifold embedded in a two-dimensional space. The vanilla training procedure fails to approximate the target density effectively, as evidenced by the qualitative results and the substantially high value of MMD^2 (≈ 383.44) shown in Fig. 6.6. In contrast, Gaussian mollification gradually guides the estimated density towards the target, as depicted in Fig. 6.6, leading to a significantly lower MMD^2 (≈ 6.13). Additionally, the mollification approach enables the estimated model not only to accurately learn the manifold but also to capture the mode of the density correctly.

6.4.2 Image Experiments

Setup. We evaluate our method on image generation tasks on CIFAR10 (Krizhevsky and Hinton, 2009) and CELEBA 64 (Liu et al., 2015) datasets, using a diverse set of likelihood-based GMs. The evaluated models include the vanilla VAE Kingma and Welling, 2014, the β-VAE (Higgins et al., 2017), and the VAE-IAF (Kingma et al.,

2016) which employs an expressive inverse autoregressive flow for the approximate posterior. To further obtain flexible approximations of the posterior of latent variables as well as a tight evidence lower bound (ELBO), we also select the Hamiltonian-VAE (HVAE) (Caterini et al., 2018) and the importance weighted VAE (IWAE) (Burda et al., 2016). For flow-based models, we consider the REAL-NVP (Dinh et al., 2017a) and GLOW (Kingma and Dhariwal, 2018) models in our benchmark. We found that that further training the model on the original data after the mollification procedure leads to better performance. Hence, in our approach we apply data mollification during the first half of the optimization phase, and we continue optimize the model using the original data in the second half. Nevertheless, to ensure a fair comparison, we adopt identical settings for the vanilla and the proposed approaches, including random seed, optimizer, and the total number of iterations.

Blurring mollification.   Even though Gaussian mollification is motivated by the manifold hypothesis, it is not the only way to mollify the data. Indeed, Gaussian mollification does not take into account certain inductive biases that are inherent in natural images, including their multi-scale nature. Recently, Rissanen et al. (2023), Hoogeboom and Salimans (2023), and Daras et al. (2023) have proposed methods that incorporate these biases in diffusion-type generative models. Their approach involves stochastically reversing the heat equation, which is a partial differential equation (PDE) that can be used to erase fine-scale information when applied locally to the 2D plane of an image. During training, the model first learns the coarse-scale structure of the data, which is easier to learn, and then gradually learns the finer details. It is therefore interesting to assess whether this form of data mollification is effective in the context of this work compared to the Gaussian homotopy. Note, however, that under the manifold hypothesis, this type of mollification produces the opposite effect to Gaussian homotopy in that at time $t = 0$ mollified images lie on a 1D manifold and they are gradually transformed to span the dimension of the data manifold; more details on blurring mollification can be found in the Appendix.

Image generation.   We evaluate the quality of the generated images using the popular Fréchet Inception Distance (FID) score (Heusel et al., 2017) computed on 50K samples from the trained model using the `pytorch-fid`[1] library. The results, reported in Table 6.1, indicate that the proposed data mollification consistently improves model performance compared to vanilla training across all datasets and models. Additionally, mollification through blurring, which is in line with recent results from diffusion models (Rissanen et al., 2023), is less effective than Gaussian mollification, although it still enhances the vanilla training in most cases. We also show intermediate samples in Fig. 6.8 illustrating the progression of samples from pure random noise or completely blurred images to high-quality images. Furthermore, we observe that

---

[1] https://github.com/mseitzer/pytorch-fid
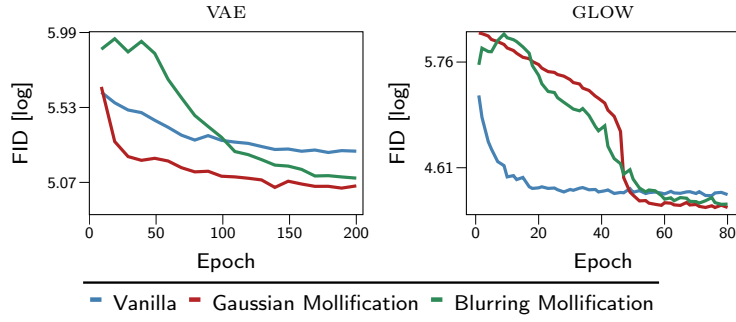
Figure 6.7: The progression of FID on CIFAR10 dataset.
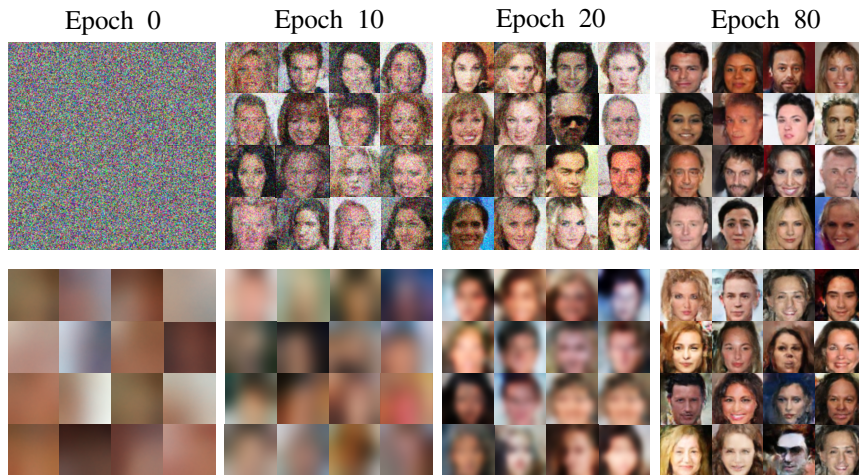
Figure 6.8: Intermediate samples generated from REAL-NVP flows (Dinh et al., 2017a), which are trained on CELEBA dataset employed with Gaussian (**top row**) and blurring mollification (**bottom row**).

Gaussian mollification leads to faster convergence of the FID score for VAE-based models, as shown in Fig. 6.7. We provide additional results in the Appendix. As a final experiment, we consider a recent large-scale VAE model for the CIFAR10 dataset, which is a deep hierarchical VAE (NVAE) (Vahdat and Kautz, 2020). By applying Gaussian mollification without introducing any additional complexity, e.g., step-size annealing, we improve the FID score from 53.64 to 52.26.

Ablation study on noise schedule.   We ablate on the choice of noise schedule for Gaussian mollification. Along with the sigmoid schedule, we also consider the linear (Ho et al., 2020) and cosine (Nichol and Dhariwal, 2021) schedules, which are also popular for diffusion models. As shown in Table Table 6.2, our method consistently outperforms the vanilla approach under all noise schedules. We also observe that the sigmoid schedule consistently produced good results. Therefore, we chose to use the sigmoid schedule in all our experiments.

Comparisons with the two-step approach in Loaiza-Ganem et al. (2022b).   Manifold overfitting in likelihood-based GMs has been recently analyzed in Loaiza-Ganem

Table 6.2: FID score on CIFAR10 w.r.t. different choices of noise schedule.

MODEL	VANILLA	SIGMOID	COSINE	LINEAR
REAL-NVP	191.98	121.75	118.71	123.93
GLOW	74.62	64.87	71.90	74.36
VAE	191.98	155.13	154.71	156.47
β-VAE	112.42	93.90	92.86	93.14
IWAE	183.04	146.70	146.49	149.16

et al. (2022b), which provides a two-step procedure to mitigate the issue. The first step maps inputs into a low-dimensional space to handle the intrinsic low dimensionality of the data. This step is then followed by likelihood-based density estimation on the resulting lower-dimensional representation. This is achieved by means of a generalized autoencoder, which relies on a certain set of explicit deep GMs, such as VAEs. Here, we compare our proposal with this two-step approach; results are reported in Table 6.3. To ensure a fair comparison, we use the same network architecture for our VAE and their generalized autoencoder, and we rely on their official implementation [2]. Following Loaiza-Ganem et al. (2022b), we consider a variety of density estimators in the low-dimensional space such as NFs, autoregressive models (ARMs) (Uria et al., 2013), adversarial variational Bayes (AVB) (Mescheder et al., 2017) and energy-based models (EBMs) (Du and Mordatch, 2019). We observe that Gaussian mollification is better or comparable with these variants. In addition, our method is extremely simple and readily applicable to any likelihood-based GMs without any extra auxiliary models or the need to modify training procedures.

Table 6.3: Comparisons of FID scores on CIFAR10 between mollitication and two-step methods.

VANILLA VAE	MOLLIFICATION		TWO-STEP			
	VAE+GAUSSIAN	VAE + BLURRING	VAE+NF	VAE+EBM	VAE+AVB	VAE+ARM
191.98	155.13	175.40	208.80	166.20	153.72	203.32

Limitations. One limitation is that this work focuses exclusively on likelihood-based GMs and on image data. The improvements in FID score indicate that the performance boost is generally substantial, but still far from being comparable with state-of-the-art DMs. While this may give an impression of a low impact, we believe that this work is important in pointing to one of the successful aspects characterizing DMs and show how this can be easily integrated in the optimization of likelihood-based GMs. A second limitation is that, in line with the literature on GMs where models are extremely costly to train and evaluate, we did not provide error bars on the results reported in the tables in the experimental section. Having said that,

---

[2]https://github.com/layer6ai-labs/two_step_zoo

the improvements reported in the experiments have been shown on a variety of models and on two popular image datasets. Furthermore, the results are supported theretically and experimentally by a large literature on Gaussian homotopy.

Broader impact.   This work provides an efficient way to improve a class of GMs. While we focused on images, the proposed method can be applied to other types of data. Like other works in this literature, the proposed method can have both positive (e.g., synthesizing new data automatically or anomaly detection) and negative (e.g., deep fakes) impacts on society depending on the application.

6.5   Related work

Our work is positioned within the context of improving GMs through the introduction of noise to the data. One popular approach is the use of denoising autoencoders (Vincent et al., 2008), which are trained to reconstruct clean data from noisy samples. Building upon this, Bengio et al., 2014 proposed a framework for modeling a Markov chain whose stationary distribution approximates the data distribution. In addition, Vincent, 2011 showed a connection between denoising autoencoders and score matching, which is an objective closely related to recent diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020). More recently, Meng et al., 2021 introduced a two-step approach to improve autoregressive generative models, where a smoothed version of the data is first modeled by adding a fixed level of noise, and then the original data distribution is recovered through an autoregressive denoising model. In a similar vein, Loaiza-Ganem et al., 2022a recently attempted to use Tweedie's formula (Robbins, 1956) as a denosing step, but surprisingly found that it does not improve the performance of NFs and VAEs. Our work is distinct from these approaches in that Gaussian mollification guides the estimated distribution towards the true data distribution in a progressive manner by means of annealing instead of fixing a noise level. Moreover, our approach does not require any explicit denoising step, and it can be applied off-the-shelf to the optimization of any likelihood-based GMs without any modifications.

6.6   Conclusion

Inspired by the enormous success of score-based DMs, in this work we hypothesized that data mollification is partially responsible for their impressive performance in generative modeling tasks. In order to test this hypothesis, we introduced data mollification within the optimization of likelihood-based GMs, focusing in particular on NFs and VAEs. Data mollification is extremely easy to implement and it has nice theoretical properties due to its connection with Gaussian homotopy, which

is a well-known technique to improve optimization. We applied this idea to challenging generative modeling tasks involving imaging data and relatively large-scale architectures as a means to demonstrate systematic gains in performance in various conditions and input dimensions. We measured performance in quality of generated images through the popular FID score. While we are far from closing the gap with DMs in achieving competing FID score, we are confident that this work will serve as the basis for future works on performance improvements in state-of-the-art models mixing DMs and likelihood-based GMs, and in alternative forms of mollification to improve optimization of state-of-the-art GMs. For example, it would be interesting to study how to apply data mollification to improve the training of GANs; preliminary investigations show that the strategy proposed here does not offer significant performance improvements, and we believe this is due to the fact that data mollification does not help in smoothing the adversarial objective. Also, while our study shows that the data mollification schedule is not critical, it would be interesting to study whether it is possible to derive optimal mollification schedules, taking inspiration, e.g., from Iwakiri et al. (2022). We believe it would also be interesting to consider mixture of likelihood-based GMs to counter problems due to the union of manifolds hypothesis, whereby the intrinsic dimension changes over the input (Brown et al., 2023). Finally, it would be interesting to investigate other data, such as 3D point cloud data (Yang et al., 2019) and extend this work to deal with other tasks, such as supervised learning.

Chapter 7

Final Remarks and Outlooks

So far in this thesis, we have discussed several challenges in Bayesian deep learning and proposed several solutions to address them. Theses challenges encompass the intricate task of choosing sensible priors as well as the development of efficient inference methodologies. In this chapter, we summarize the main contributions of this thesis and discuss some open questions as well as future research directions.

7.1 Summary of Contributions

This thesis makes contributions that follow four different themes:

- **Functional priors for Bayesian neural networks** (Chapter 3)

  In this chapter, we addressed the challenge of selecting priors for Bayesian neural networks (BNNs) in supervised-learning settings through a practical and efficient solution. Our approach involves adopting functional priors, which are more easily elicited, and aligning neural network parameter priors with these functional priors. To realize this, we employed Gaussian processes (GPs) as a powerful tool for defining prior distributions over functions. We introduced a novel framework to match their priors with the functional priors of BNNs by minimizing their Wasserstein distance. Rigorous experimentation across supervised-learning tasks demonstrated compelling empirical support for the integration of these priors with scalable Markov chain Monte Carlo (MCMC) sampling, consistently yielding considerable performance enhancements.

  Open questions: The main open question is how we can choose the target stochastic processes to map into BNN. Our proposed framework is general and supports any stochastic process as long as we can sample from it. In this work, we mainly considered shallow and hierarchical GPs as the target stochastic processes. It would be interesting to explore the impact of other stochastic processes, such as deep Gaussian processs (DGPs) or convolutional GPs (van der Wilk et al., 2017) on the performance of BNNs.

- **Model selection for Bayesian autoencoders** (Chapter 4)

  In this chapter, we introduced a novel framework for performing model selection in Bayesian autoencoders via prior optimization. We approached this by introducing a proxy for type-II maximum likelihood where we replace the marginal likelihood with the Wasserstein distance between the data distribution and the one modeled by the Autoencoder a priori. Our approach's efficacy is showcased through comprehensive experiments encompassing diverse *unsupervised learning* tasks such as representation learning and generative modeling for high-dimensional data, particularly in situations constrained by limited data availability.

  Open questions: Utilizing the marginal likelihood for selecting the prior distribution stands as a notably principled strategy, although its estimation poses a considerable difficulty, particularly for BNNs. In this work, we proposed a powerful and flexible framework to optimize this objective by leveraging the Wasserstein distance but only focused on unsupervised learning tasks. The open question is how to extend this framework to BNNs in supervised learning settings.

- **Sparse Gaussian process priors for fully Bayesian autoencoders** (Chapter 5)

  In this chapter, we introduced a fully Bayesian autoencoder model that extends the Bayesian treatment to both local latent variables and global decoder parameters. Our approach involves an amortized MCMC method utilizing an implicit stochastic network as an encoder. This enables flexible priors and posterior approximations while maintaining low computational costs for inference. Additionally, we proposed incorporating sparse Gaussian process priors over the latent space to capture correlations between latent encodings. Our approach showcased impressive performance across dynamic representation and generative modeling tasks, as evident by a wide range of experiments.

  Open questions: Although our novel fully Bayesian autoencoder model allows us to specify any priors on the latent space and the decoder parameters, in this work, we only employed a simple isotropic Gaussian prior for the decoder parameters. Exploring more informative priors for the decoder parameters is worthwhile in future work to further improve the performance of our model.

- **Improving training of likelihood-based generative models with data mollification** (Chapter 6)

  In this chapter, we introduced a simple yet effective technique to improve the training of likelihood-based generative models through data mollification. This was inspired by the recent success of diffusion models. A key feature of score-based diffusion models is to accurately estimate density in low-density regions and address manifold overfitting by employing data mollification through

the addition of Gaussian noise. Remarkably, data mollification introduces no additional computational burden and can be seamlessly integrated into any training loop with just a single line of code. We demonstrated that this simple technique can significantly facilitate inference and consistently enhances the sample quality of likelihood-based generative models, including variational autoencoders (VAEs) and normalizing flows.

Open questions: It would be interesting to study how to apply our data mollification technique to implicit generative models such as generative adversarial networks (GANs); preliminary investigations show that the strategy proposed here does not offer significant performance improvements. Moreover, we adopted some data mollification schedules from diffusion models and demonstrated their efficacy in enhancing the training of likelihood-based generative models. Nevertheless, there remains the potential to derive optimal mollification schedules tailored specifically to these models.

7.2 Future Directions

One of the main research hypotheses of this thesis is that *with appropriate priors, Bayesian deep learning can offer data-efficiency, accurate and reliable solutions which enable us to solve many challenging practical problems*. In this thesis, we successfully developed fundamental methods to specify sensible priors for Bayesian deep learning models. However, inspired by the "*no-free-lunch*" theorem (Wolpert, 1996), we believe that there is no universally preferred prior as different tasks require their own inductive biases. Thus, our next goals are: (1) to develop novel approaches to incorporate prior knowledge and impose inductive biases into various network architectures for tasks for which this has yet to be done; and (2) to test these priors on practical and important applications where high-quality uncertainty estimates and robust predictions are needed. To this end, we identify two promising research directions, which we will discuss in the following.

Unsupervised Learning Informative Priors for Bayesian Neural Nets

Deep learning models offer powerful tools to learn from data, but models usually require a huge amount of labeled data to be successful. This is one of the biggest obstacles to applying deep learning models in practical settings such as medical imaging or, in more general, life sciences, where the cost of annotating data is expensive. Meanwhile, unlabeled data are increasingly available due to advances in sensor technologies. As a result, the idea of exploiting additional data, e.g., unlabeled data from the same task, or labeled data from different but related tasks, has been considered a practical way to reduce the amount of labeled training data required

to learn a task. This is the main motivation for transfer, meta, semi, self-supervised learning, and recent foundation models (Bommasani et al., 2021) for a large number of applications of deep learning in recent years.

Even though the Bayesian framework offers a principled approach to incorporating prior knowledge into our model, there is still a lack of sound and effective methods to exploit extra unlabeled data to define informative priors for Bayesian neural networks for supervised learning tasks. Recently, there have been some attempts to utilize extra data to specify priors for Bayesian neural networks. For example, Atanov et al. (2019) firstly train a convolutional neural network (CNN) on related tasks and then learn a generative model using a VAE for the learned filter weights. This generative model is then used as a prior for convolutional filters of a Bayesian neural network for downstream tasks. Meanwhile, Shwartz-Ziv et al. (2022) proposed to use a re-scaled Bayesian posterior from the first task as a pre-trained prior for the downstream task. However, these works mainly consider leveraging auxiliary labeled data and are cast into settings such as domain adaptation or homogeneous transfer learning where the source and target tasks have similar features and label spaces but differ in their marginal distributions. Unfortunately, the transfer of these approaches from the supervised to unsupervised context is not trivial because of the huge amount of model parameters to fit and the absence of an objective quantity to optimize in this case.

Inspired by the success of our recent work on Bayesian autoencoders (BAEs) (Tran et al., 2021), the first future direction is to develop a novel method based on this model that enables unsupervised learning of informative foundation priors for supervised learning tasks from massive unlabeled data. We demonstrated that, given a good prior, BAEs outperform variants of VAEs (Kingma and Welling, 2014) in terms of quality of generated samples but also on representation learning. Thanks to the distributional sliced Wasserstein distance (Nguyen et al., 2021), we proposed a principled approach to optimize the prior for this model bypassing computationally involved procedures and the curse of dimensionality. We also found that the resulting prior over the parameters of the encoder induces a meaningful and reusable prior in the latent space. In addition, the main distinction of BAEs from VAEs is that we impose prior directly on the parameters of the network instead of on the latent space. Thus the resulting prior for the encoder can be adapted easily to the downstream supervised tasks. We also endeavor to investigate strategies to constrain the latent space and refine this prior to impose the inductive bias effectively while retaining the model's flexibility for the downstream task. Additionally, it is crucial to validate this framework not only on standard benchmarks but also in critical applications like medical image classification and segmentation, where annotated data is scarce and robust uncertainty estimation is highly desirable.

Informative Priors for Bayesian Graph Neural Networks

Graph data is ubiquitous in the real world, including social networks, traffic networks, and molecules, to just name a few. Recently, graph neural networks (GNNs) (Kipf and Welling, 2017) have attracted significant attention in dealing with such data due to the ability to learn effectively graph representations based on message-passing. In the last few years, these models have achieved remarkable accuracy in a wide range of important applications, such as learning dynamics of physical systems, learning multiagent communications, and protein structure predictions.

Unfortunately, recent works (Teixeira et al., 2019; Wang et al., 2021; Hsu et al., 2022) reveal that GNNs are not trustworthy, and more importantly, GNNs tend to be under-confident in their predictions, which is very different from other modern deep learning models that are often over-confident. Furthermore, state-of-the-art calibration methods for deep learning models are not able to fix this issue (Teixeira et al., 2019). There are some recent works have been proposed to tackle this problem in a post-hoc manner. However, these require leaving out part of the data to perform post-recalibration, which can be problematic in applications where obtaining labeled data is difficult or expensive. In addition, Ovadia et al., 2019 showed that post-hoc calibration gives good results in independent and identically distributed (i.i.d.) regimes, but it usually fails under even a mild distributional shift of the data.

The first part of the second future research direction aims at revisiting the scalable Bayesian treatment of Graph Neural Networks and at designing informative priors for these models, which enable data-efficient learning and reliable uncertainty estimation. Some recent works (Elinas et al., 2020; Hasanzadeh et al., 2020) demonstrate that Bayesian GGNs are more robust than their deterministic counterpart in dealing with challenging cases, such as learning in the absence of an input graph and dealing with highly-effective adversarial perturbations of the graph. However, the literature on evaluating the uncertainty estimation of Bayesian GNNs and choosing sensible priors for these models is far from mature. Besides, priors for Bayesian GNNs are still overlooked and mainly built in an ad-hoc manner (Elinas et al., 2020). There are two possible steps to solve theses problems. The first step is to perform a comprehensive evaluation of uncertainty estimation from scalable Bayesian methods for GGNs not only in i.i.d. regimes but also under dataset shift. The second step is to design novel topology-aware priors to further increase the robustness and data-efficiency of Bayesian GNNs by encoding our high-level beliefs on the graph data. For example, the novel priors should reflect our knowledge of topology factors that influence the calibration of GNNs (Hsu et al., 2022), such as diversity of node distributions, neighborhood similarity, etc. Another example in the context of modeling of molecules, is incorporating the belief that molecules are invariant to rotations.

The second part of this research direction is intended to apply Bayesian GNNs endowed with good priors to Bayesian optimization for scientific problems such as

optimization of chemical properties of molecules. Bayesian opimization (BO) is well-suited for global optimization of expensive and black-box functions and has been widely used in many contexts where experiments or simulations can be extremely costly and time-consuming. The key element of BO is a surrogate model which captures a distribution over potential functions to incorporate uncertainty in its prediction. GPs are typically chosen as the surrogate model for BO but these models have several bottlenecks, such as scaling poorly with the number of data points and often struggling in high dimensional search spaces. BNNs therefore have been proposed as an alternative to GPs to tackle complex BO tasks with high-dimensional structured data (Kim et al., 2022). In addition, Wang et al. (2022) recently showed that choosing a sensible prior over functions plays an important role for BO but only considered the case of GPs. Our aim, therefore, is to extend our framework for choosing functional priors (Tran et al., 2022) for BNNs and combine them with novel topology-aware priors for Bayesian optimization problems on graph data. By imposing sensible inductive biases on the graph and constraints on the function space, these priors are expected to significantly facilitate the Bayesian optimization process in terms of sample efficiency. This is extremely helpful for applications of chemical optimization such as drug design and material optimization, where the space of synthesizable molecules is extremely large and complex while the synthesis cost is very large.

Appendix A

Appendix for Chapter 3

A.1 A primer on Wasserstein Distance

Given two *Borel's probability measures* $\pi(\boldsymbol{x})$ and $\nu(\boldsymbol{y})$ defined on the *Polish space* $\mathcal{X}$ and $\mathcal{Y}$ (i.e. any complete separable metric space such as a subset of $\mathbb{R}^d$), the p-**Wasserstein distance** is defined as follows

$$W_p(\pi, \nu) = \left(\inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} D(\boldsymbol{x}, \boldsymbol{y})^p \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \right)^{1/p}, \tag{A.1}$$

where $D(\boldsymbol{x}, \boldsymbol{y})$ is a proper distance metric between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in the space $\mathcal{X} \times \mathcal{Y}$ and $\Gamma(\pi, \nu)$ is the set of functionals of all possible joint densities whose marginals are indeed π and ν.

When the space of $\boldsymbol{x}$ and $\boldsymbol{y}$ coincides (i.e. $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \subseteq \mathbb{R}^d$), the most used formulation is the 1-Wasserstein distance with Euclidian norm as distance,

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{y}\| \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}, \tag{A.2}$$

This is also known in the literature as the Earth-Mover distance. Intuitively, here γ measures how much mass must be transported from $\boldsymbol{x}$ to $\boldsymbol{y}$ in order to transform the distributions π into the distribution ν. Solving the Wasserstein distance means computing the minimum mass that needs to be moved. The question "How?" is answered by looking at the optimal transport plan (not the focus of these notes).

The remaining part of these notes will be dedicated to the proof of the dual formulation for Eq. A.2. It is well known in the literature of optimization that linear programming problem with convex constrains admits a dual formulation. Kantorovich introduced the dual formulation of the Wasserstein distance in 1942.

Theorem 2. *On the same setup as before, the Wasserstein distance defined as*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{y}\| \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}, \tag{A.3}$$

admits the following dual form

$$W(\pi, \nu) = \sup_{\|f\|_L \leq 1} \int_{\mathcal{X}} f(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} - \int_{\mathcal{X}} f(\boldsymbol{y}) \nu(\boldsymbol{y}) d\boldsymbol{y} \tag{A.4}$$

where f is a 1-Lipschitz continuous function defined on $\mathcal{X} \rightarrow \mathbb{R}$.

Step 1: Kantorovich duality

First of all we start with the **Kantorovich duality**, which defines a dual form for the generic 1-Wasserstein.

Theorem 3. *Given a nonnegative measurable function $D : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, the 1-Wasserstein is computed as follows,*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int D(\boldsymbol{x}, \boldsymbol{y}) \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}, \tag{A.5}$$

The Kantorovich duality proves that this is equal to the following constrained optimization problem,

$$W(\pi, \nu) = \sup_{\substack{f, g \\ f(\boldsymbol{x}) + g(\boldsymbol{y}) \leq D(\boldsymbol{x}, \boldsymbol{y})}} \int f(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} + \int g(\boldsymbol{y}) \nu(\boldsymbol{y}) d\boldsymbol{y}. \tag{A.6}$$

We define $\iota_\Gamma(\gamma)$ the following quantity

$$\iota_\Gamma(\gamma) = \sup_{f, g} \left[\int f(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} + \int g(\boldsymbol{y}) \nu(\boldsymbol{y}) d\boldsymbol{y} - \iint \left[f(\boldsymbol{x}) + g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \right]$$

and we observe that

$$\iota_\Gamma(\gamma) = \begin{cases} 0 & \text{if } \gamma \in \Gamma(\pi, \nu), \\ +\infty & \text{otherwise}. \end{cases}$$

This is true because given the definition of Γ, if $\gamma \in \Gamma(\pi, \nu)$ then $\pi(\boldsymbol{x}) = \int \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}$ and $\nu(\boldsymbol{y}) = \int \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x}$. By substituiting these quantities, it follows that

$$\int f(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} + \int g(\boldsymbol{y}) \nu(\boldsymbol{y}) d\boldsymbol{y} = \int f(\boldsymbol{x}) \int \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} d\boldsymbol{x} + \int g(\boldsymbol{y}) \int \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}$$
$$= \iint \left[f(\boldsymbol{x}) + g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}.$$

In other cases, f and g can be chosen such that the supremum becomes $+\infty$. Given this property and the constrain on γ, we can add $\iota_\Gamma(\gamma)$ to the formulation of the

Wasserstein distance in Eq. A.3,

$$W(\pi,\nu) = \inf_{\gamma \in \Gamma(\pi,\nu)} \left[\int D(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} \right] + \iota_\Gamma(\gamma) =$$

$$= \inf_\gamma \left[\int D(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} + \sup_{f,g} \left[\int f(\boldsymbol{x})\pi(\boldsymbol{x})d\boldsymbol{x} + \int g(\boldsymbol{y})\nu(\boldsymbol{y})d\boldsymbol{y} - \right. \right. \qquad \text{(A.7)}$$
$$\left. \left. \iint \left[f(\boldsymbol{x}) + g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} \right] \right] ,$$

Now, the original integral of the Wasserstein distance does not depend on f and g; therefore the supremum can be moved in front,

$$W(\pi,\nu) = \inf_\gamma \sup_{f,g} \Upsilon(\gamma,(f,g)) \qquad \text{(A.8)}$$

$$\Upsilon(\gamma,(f,g)) \stackrel{\text{def}}{=} \int D(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} + \int f(\boldsymbol{x})\pi(\boldsymbol{x})d\boldsymbol{x} + \int g(\boldsymbol{y})\nu(\boldsymbol{y})d\boldsymbol{y} -$$
$$\iint \left[f(\boldsymbol{x}) + g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y}$$

Under certain conditions stated by the *minimax theorem*, i.e. $\Upsilon(\gamma,(f,g))$ is convex-concave function (Υ is concave for fixed (f,g) while convex for fixed γ), we can swap the infinum and the supremum and rewrite the definition as follows,

$$W(\pi,\nu) = \sup_{f,g} \inf_\gamma \int \left[D(\boldsymbol{x},\boldsymbol{y}) - f(\boldsymbol{x}) - g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} + \int f(\boldsymbol{x})\pi(\boldsymbol{x})d\boldsymbol{x} + \int g(\boldsymbol{y})\nu(\boldsymbol{y})d\boldsymbol{y}$$

Proofs that the hypothesis used for the minimax theorem hold for this case are presented in Theorem 1.9 of "Topics in Optimal Transport" (Villani, 2003). Focusing on the infimum part, we can write

$$\inf_\gamma \int \left[D(\boldsymbol{x},\boldsymbol{y}) - f(\boldsymbol{x}) - g(\boldsymbol{y}) \right] \gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} = \begin{cases} 0 & \text{if } f(\boldsymbol{x}) + g(\boldsymbol{y}) \leq D(\boldsymbol{x},\boldsymbol{y}), \\ -\infty & \text{otherwise}. \end{cases}$$

If the function $\zeta(\boldsymbol{x},\boldsymbol{y}) = D(\boldsymbol{x},\boldsymbol{y}) - (f(\boldsymbol{x}) + g(\boldsymbol{y}))$ takes a negative value at some point $(\boldsymbol{x}_0,\boldsymbol{y}_0)$, then by choosing $\gamma = \lambda\delta(\boldsymbol{x}_0,\boldsymbol{y}_0)$ with $\lambda \to +\infty$ (i.e. a Dirac delta in $(\boldsymbol{x}_0,\boldsymbol{y}_0)$), we see that the infimum is infinite. On the other hand, is $\zeta(\boldsymbol{x},\boldsymbol{y})$ is nonnegative, then the infimum is obtained for $\gamma = 0$. Finally, this constrains can be added to the previous conditions making thus recovering the formulation in Eq. A.4.

Step 2: D-Transforms

The next challenge is to find f and g such that we can easily recover the constrain optimization above. We approach this problem by supposing to have chosen some $f(\boldsymbol{x})$. This means that the objective is to find a good $g(\boldsymbol{y})$ that for all $\boldsymbol{x},\boldsymbol{y}$ satisfy the

condition

$$f(x) + g(y) \leq D(x, y).$$

The trivial solution is $g(y) \leq D(x, y) - f(x)$. This must be true for all x, also in the worst case (when we take the infimum),

$$g(y) \leq \inf_x [D(x, y) - f(x)].$$

At this point, we observe that for a given f, if we want the supremum in Eq. 5 we cannot get a better g then taking the equality,

$$\bar{f}(y) := \inf_x [D(x, y) - f(x)].$$

We therefore have the following formulation of the Wasserstein distance,

$$W(\pi, \nu) = \sup_f \left[\int f(x)\pi(x)dx + \int \bar{f}(y)\nu(y)dy \right]$$

If now we suppose to choose g, by following the same reasoning the best f that we can get is defined

$$\bar{\bar{f}}(x) = \bar{g}(x) := \inf_y [D(x, y) - g(y)].$$

If we replace $g(y)$ with Eq. 17 we have yet another recursive definition of the Wasserstein distance,

$$W(\pi, \nu) = \sup_f \left[\int \bar{\bar{f}}(x)\pi(x)dx + \int \bar{f}(y)\nu(y)dy \right]$$

If we constrain f to be D-concave, then $\bar{\bar{f}} = f$.

Step 2.1: Euclidean distance

It's worth mentioning that this formulation is valid for any nonnegative measurable function D. For the Euclidian distance this simplify even further.

Theorem 4. *When $D(x, y) = \|x - y\|$ and f is 1-Lipschitz, f is D-concave if and only if $\bar{f} = -f$*

We prove the necessity condition of such result. First of all, we observe that if f is 1-Lipschitz then $\bar{f}$ is 1-Lipschitz too. This is true because for any given x

$$\bar{f}_x(y) = \|x - y\| - f(x)$$

is 1-Lipschitz and therefore the infimum of $\bar{f}(y) = \inf_x \|x - y\| - f(x)$ is 1-Lipschitz. Since $\bar{f}$ is 1-Lipschitz, for all x and y we have

$$|\bar{f}(y) - \bar{f}(x)| \leq \|y - x\|$$
$$\implies -\bar{f}(x) \leq \|x - y\| - \bar{f}(y)$$

Since this is true for all y,

$$-\bar{f}(x) \leq \inf_y \|x - y\| - \bar{f}(y)$$
$$-\bar{f}(x) \leq \underbrace{\inf_y \|x - y\| - \bar{f}(y)}_{\bar{\bar{f}} \equiv f} \leq -\bar{f}(x)$$

where the right inequality follows by choosing $y = x$ in the infimum. We know that $\bar{\bar{f}} \equiv f$. This means that $-\bar{f}(x)$ must be equal to $f(x)$ for the last equation to hold.

Step 3. Putting everything together

We started our discussion by proving the Kantovich duality, which states that

$$\inf_{\gamma \in \Gamma(\pi, \nu)} \int D(x, y)\gamma(x, y)dxdy = \sup_{\substack{f, g \\ f(x) + g(y) \leq D(x, y)}} \int f(x)\pi(x)dx + \int g(y)\nu(y)dy,$$

We then proved that

$$\sup_{\substack{f, g \\ f(x) + g(y) \leq D(x, y)}} \left[\int f(x)\pi(x)dx + \int g(y)\nu(y)dy \right] =$$
$$= \sup_{\substack{f \\ \bar{f} = \inf_x D - f}} \left[\int f(x)\pi(x)dx + \int \bar{f}(y)\nu(y)dy \right],$$

Finally, given $D(x, y)$ to be the Euclidean distance, we discussed the shape of $\bar{f}$ when we restrict f to be 1-Lipschitz, showing that $\bar{f} = -f$. Putting everything together, we obtain the dual 1-Wasserstein distance in Eq. A.4,

$$W(\pi, \nu) = \sup_{\|f\|_L \leq 1} \int f(x)\pi(x)dx - \int f(y)\nu(y)dy$$

A.2 Implementation and experimental details

In this section, we present details on implementation and hyperparameters used in our experimental campaign. Our implementation is mainly in PyTorch (Paszke

et al., 2019). We follow the standard protocol of training, validation and testing. The hyperparameters are selected according to the negative log-likelihood (NLL) performance on a validation set, which is created by randomly choosing 20% of the data points from the training set. We standardize all the input features and the outputs using the statistics of the training set. Regarding prior optimization, unless otherwise specified, for the inner loop of Algorithm 2, we use the Adagrad optimizer (Duchi et al., 2011) with a learning rate of 0.02, a Lipschitz regularization coefficient $\lambda = 10$, and a number of Lipschitz iterations $n_{\text{Lipschitz}} = 200$. Whereas, for the outer loop of Algorithm 2 we use the RMSprop optimizer (Tieleman and Hinton, 2012) with a learning rate of 0.05 for the experiments on the UCI and BANANA datasets, and a learning rate of 0.01 for the rest. See Appendix A.3.3 for the progressions of prior optimization.

A.2.1   Deep Ensemble

Deep Ensemble (Lakshminarayanan et al., 2017) averages the predictions across networks trained independently starting from different initializations. In our experiments, we use an ensemble of 5 neural networks. Every member of the ensemble is trained with the L_2-regularized objective

$$\mathcal{L}(\boldsymbol{w}) := -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2, \tag{A.9}$$

where N is the size of training data, λ is the weight decay coefficient, $\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w})$ is the log likelihood evaluated at the data point $(\boldsymbol{x}_i, y_i)$. Following Lakshminarayanan et al. (2017), for regression task, in order to capture predictive uncertainty, we use a network that outputs the predicted mean $\mu_w(\boldsymbol{x})$ and variance $\sigma_w^2(\boldsymbol{x})$. Assume that the observed value follows a heteroscedastic Gaussian distribution, the log likelihood is then

$$\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) = -\frac{1}{2}\log \sigma_w^2(\boldsymbol{x}_i) - \frac{(y_i - \mu_w(\boldsymbol{x}_i))^2}{2\sigma_w^2(\boldsymbol{x}_i)} + \text{const.} \tag{A.10}$$

For the classification task, the log likelihood is simply the softmax cross-entropy loss.

We use the Adam optimizer (Kingma and Ba, 2015) to train all the networks. For multilayer perceptrons (MLPs), we use a fixed learning rate 0.01 and total epochs of 50. Whereas convolutional neural networks (CNNs) are trained for 200 epochs. The learning rate starts from 10^{-2} and decays to $(10^{-3}, 10^{-4}, 10^{-5})$ at epochs $(50, 100, 150)$. The L_2 regularization strength is tuned over a grid $\lambda \in \{10^k \mid k \text{ from -8 to -1}\}$.

A.2.2   Likelihoods for BNNs

Similarly to the prior, the likelihood for Bayesian neural networks (BNNs) is a modeling choice. It is a function of the model predictions $\hat{\boldsymbol{y}}$ and the correct targets

y. For multi-class *C*-way classification, the neural network (NN) have *C* output units over which a softmax function is applied, hence the network outputs class probabilities. The likelihood is commonly chosen as a multinomial distribution, $p(\mathcal{D} \mid \boldsymbol{w}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \hat{y}_{n,c}^{y_{n,c}}$, for *C* classes, where $\hat{y} \in [0,1]$ denotes predicted probability, and $y_{n,c}$ is the true targets.

For regression, one usually models output noise as a zero-mean Gaussian: $\epsilon \sim \mathcal{N}(0, \sigma_{\epsilon}^2)$, where σ_{ϵ}^2 is the variance of the noise. The likelihood is then the Gaussian $p(\mathcal{D} \mid \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y} \mid \hat{\boldsymbol{y}}, \sigma_{\epsilon}^2)$. Notice that the noise variance σ_{ϵ}^2 is treated as a hyperparameter. We do choose this hyperparameter over the grid $\sigma_{\epsilon}^2 \in \left\{ 5^k, 10^k \mid k \text{ from -3 to -1} \right\}$. The optimal values are selected according to the NLL result of the predictive posterior. Table A.4 and Table A.6 present σ_{ϵ}^2 used in the UCI regression experiments.

Experimental configurations. In all experiments, unless otherwise specified, we use a momentum coefficient $\alpha = 0.01$, and a step size $\varepsilon = 0.01$. For the UCI regression experiments, we sample four independent chains; for each chain, the number of collected samples after thinning is 30 except for the large dataset (PROTEIN), where a number of 60 samples is used. The thinning intervals are 2000 and 5000 iterations for the small and large datasets, respectively. The burn-in period lasts 2000 iterations for the BOSTON, CONCRETE, ENERGY, WINE datasets, and 5000 iterations for the rest. For the UCI classification experiments, we also use four chains, in which the number of burn-in iterations are 2000 for small datasets (EEG, HTRU2, MAGIC, and MOCAP) and 5000 for large datasets (MINIBOO, LETTER, and DRIVE). We draw 30 samples for each chain with a thinning interval of 2000 iterations for the small datasets and 5000 iterations for the large datasets. In the experiments with CNNs on CIFAR10, after a burn-in phase of 10,000 iterations, we collect 200 samples with a thinning interval of 10,000 iterations.

A.2.3 Tempered posterior

We follow the approach of Wenzel et al. (2020) for tempering the posterior as follows

$$p(\boldsymbol{w} \mid \mathcal{D}) \propto \exp(-U(\boldsymbol{w})/T), \qquad (A.11)$$

where $U(\boldsymbol{w}) = -\log p(\mathcal{D} \mid \boldsymbol{w}) - \log p(\boldsymbol{w})$ is the potential energy, and *T* is the temperature value. As suggested by Wenzel et al. (2020), we only study the "cold" posterior, where a temperature $T < 1$ is used. In this case, we artificially sharpen the posterior by overcounting the training data by a factor of $1/T$ and rescaling the prior as $p(\boldsymbol{w})^{\frac{1}{T}}$. As a result, the posterior distribution is more concentrated around solutions with high likelihood. In our experiments, we do grid-search over temperature values $T \in \left\{ 0.5, 0.1, 10^{-2}, 10^{-3}, 10^{-4} \right\}$.

A.2.4   Details on the sampling scheme for BNN hierarchical priors

As mentioned in § 3.4.2, for the BNN hierarchical priors, we firstly place a Gaussian prior on the network parameters. For simplicity, let's consider only the weights in *l*-th layer. We have

$$w_l^{(1)}, ..., w_l^{(N_l)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{l_w}^2), \tag{A.12}$$

where $w_l^{(i)}$ is the *i*-th weight, N_l is the number of weights in layer *l*. We further place an Inverse-Gamma prior on the variance:

$$\sigma_{l_w}^2 \sim \Gamma^{-1}(\alpha_{l_w}, \beta_{l_w}). \tag{A.13}$$

We aim to generate samples from the posterior $p\left(\sigma_{l_w}^2, \{w_l^{(i)}\}_{i=1}^{N_l} \mid \mathcal{D}\right)$. As done by Chen et al. (2014), the sampling procedure is carried out by alternating the following steps:

(i) Sample weights from $p\left(\{w_l^{(i)}\}_{i=1}^{N_l} \mid \sigma_{l_w}^2, \mathcal{D}\right)$ using the stochastic gradient Hamiltonian Monte Carlo (SGHMC) sampler. We sample the weights for *K* steps before resampling the variance.

(ii) Sample the variance from $p\left(\sigma_{l_w}^2 \mid \{w_l^{(i)}\}_{i=1}^{N_l}\right)$ using a Gibbs step.

Assume we observed the weights $\{w_l^{(i)}\}_{i=1}$ after the step (i), the posterior for the variance can be obtained in a closed form as follows

$$
\begin{aligned}
p\left(\sigma_{l_w}^2 \mid \{w_l^{(i)}\}_{i=1}^{N_l}\right) &\propto \left(\prod_{i=1}^{N_l} p\left(w_l^{(i)} \mid \sigma_{l_w}^2\right)\right) p\left(\sigma_{l_w}^2 \mid \alpha_{l_w}, \beta_{l_w}\right) \\
&\propto \left(\prod_{i=1}^{N_l} (\sigma_{l_w}^2)^{-1/2} \exp\left\{-\frac{1}{2\sigma_{l_w}^2}\left(w_l^{(i)}\right)^2\right\}\right) (\sigma_{l_w}^2)^{-\alpha_{l_w}-1} \exp\left\{-\frac{1}{\sigma_{l_w}^2}\beta_{l_w}\right\} \\
&= (\sigma_{l_w}^2)^{-(\alpha_{l_w}+N_l/2)-1} \exp\left\{-\frac{1}{\sigma_{l_w}^2}\left(\beta_{l_w} + \frac{1}{2}\sum_{i=1}^{N_l}\left(w_l^{(i)}\right)^2\right)\right\} \\
&\propto \Gamma^{-1}\left(\alpha_{l_w} + \frac{N_l}{2}, \beta_{l_w} + \frac{1}{2}\sum_{i=1}^{N_l}\left(w_l^{(i)}\right)^2\right). \tag{A.14}
\end{aligned}
$$

As a default, in our experiments, we set the resampling interval $K = 100$ except for the experiment on the 1D synthetic dataset (§ 3.5.1), in which we use $K = 20$.

A.2.5   MAP estimation with Gaussian prior

For completeness, we describe the maximum-a-posteriori (MAP) estimation for the case of Gaussian prior used in § 3.6.7. This derives interpretation of the regularization effect from the prior for deterministic networks. We aim at finding a point estimate

that maximizes the posterior:

$$
\begin{aligned}
w_{\mathrm{MAP}} &= \arg\max_{w} p(w \mid \mathcal{D}) \\
&= \arg\max_{w} p(\mathcal{D} \mid w) p(w) \\
&= \arg\max_{w} \{\log p(\mathcal{D} \mid w) + \log p(w)\}.
\end{aligned}
\tag{A.15}
$$

If the prior is a Gaussian distribution, $p(w) = \mathcal{N}(\mu, \Sigma)$, we have

$$
w_{\mathrm{MAP}} = \arg\max_{w} \left\{ \log p(\mathcal{D} \mid w) - \frac{1}{2}(w - \mu)^{\top} \Sigma^{-1}(w - \mu) \right\}.
\tag{A.16}
$$

In our experiments, the prior covariance is set as isotropic, $\Sigma = \sigma_{\mathrm{prior}}^2 I$, and prior mean is zero, $\mu = 0$. Thus, we have

$$
w_{\mathrm{MAP}} = \arg\max_{w} \left\{ \log p(\mathcal{D} \mid w) - \frac{1}{2\sigma_{\mathrm{prior}}^2} \|w\|_2^2 \right\}
\tag{A.17}
$$

Here, we use the same likelihoods $p(\mathcal{D}|w)$ as in Appendix A.2.2.

Regression task. For the Gaussian likelihood $p(\mathcal{D} \mid w) = \mathcal{N}(y \mid \hat{y}, \sigma_{\epsilon}^2)$, the MAP estimation is then

$$
w_{\mathrm{MAP}} = \arg\max_{w} \left\{ -\frac{1}{2\sigma_{\epsilon}^2} \|\hat{y} - y\|_2^2 - \frac{1}{2\sigma_{\mathrm{prior}}^2} \|w\|_2^2 \right\}.
\tag{A.18}
$$

This is equivalent to minimizing the L_2-regularized squared-error objective:

$$
w_{\mathrm{MAP}} = \arg\min_{w} \left\{ \sum_{n=1}^{N} (\hat{y}_n - y_n)^2 + \frac{\sigma_{\epsilon}^2}{\sigma_{\mathrm{prior}}^2} \|w\|_2^2 \right\}.
\tag{A.19}
$$

Here, we can interpret that the term $\frac{\sigma_{\epsilon}^2}{\sigma_{\mathrm{prior}}^2}$ controls the regularization strength.

Classification task. For the multinomial likelihood $p(\mathcal{D} \mid w) = \prod_{n=1}^{N} \prod_{c=1}^{C} \hat{y}_{n,c}^{y_{n,c}}$, estimating MAP is equivalent to minimizing the L_2-regularized cross-entropy objective:

$$
w_{\mathrm{MAP}} = \arg\min_{w} \left\{ -\sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c}) + \frac{1}{2\sigma_{\mathrm{prior}}^2} \|w\|_2^2 \right\},
\tag{A.20}
$$

where $\frac{1}{\sigma_{\mathrm{prior}}^2}$ is the regularization coefficient.

Layer	Dimensions
Conv2D	$3 \times 6 \times 5 \times 5$
Conv2D	$6 \times 16 \times 5 \times 5$
Linear-ReLU	400×120
Linear-ReLU	120×84
Linear-Softmax	84×10

Table A.1: LENET5 architecture

Layer	Dimensions	
Conv2D	$3 \times 16 \times 3 \times 3$	
Residual Block	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix}$	$\times 3$
Residual Block	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix}$	$\times 3$
Residual Block	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	$\times 3$
AvgPool	8×8	
Linear-Softmax	64×10	

Table A.2: PRE-RESNET20 architecture

Layer	Dimensions
Conv2D	$3 \times 32 \times 3 \times 3$
Conv2D	$32 \times 32 \times 3 \times 3$
MaxPool	2×2
Conv2D	$32 \times 64 \times 3 \times 3$
Conv2D	$64 \times 64 \times 3 \times 3$
MaxPool	2×2
Conv2D	$64 \times 128 \times 3 \times 3$
Conv2D	$128 \times 128 \times 3 \times 3$
Conv2D	$128 \times 128 \times 3 \times 3$
MaxPool	2×2
Conv2D	$128 \times 256 \times 3 \times 3$
Conv2D	$256 \times 256 \times 3 \times 3$
Conv2D	$256 \times 256 \times 3 \times 3$
MaxPool	2×2
Conv2D	$256 \times 256 \times 3 \times 3$
Conv2D	$256 \times 256 \times 3 \times 3$
Conv2D	$256 \times 256 \times 3 \times 3$
MaxPool	2×2
Linear-ReLU	256×256
Linear-ReLU	256×256
Linear-Softmax	256×10

Table A.3: VGG16 architecture

A.2.6 Network architectures

As previously mentioned in **??**, we employ the NTK parameterization (Jacot et al., 2018; Lee et al., 2020) for MLPs and CNNs. We initialize the weights $w_l \sim \mathcal{N}(0, 1)$ and $b_l = 0$ for both fully-connected and convolutional layers. Tables A.1 to A.3 show details on the CNNs architectures used in our experimental campaign. These networks are adapted to the CIFAR10 dataset. The parameters of batch normalization layers of PRERESNET20 are treated as constants. In particular, we set the scale and shift parameters to 1 and 0, respectively.

A.2.7 Measuring similarity between GPs and BNNs using maximum mean discrepancy

In § 3.5.3, we adopted the approach of Matthews et al. (2018) to measure the similarity between Gaussian processes (GPs) and BNNs using a kernel two-sample test based on maximum mean discrepancy (MMD) Gretton et al. (2012). The MMD between two

distributions p_{gp} and p_{nn} is defined as follows

$$\text{MMD}(p_{gp}, p_{nn}) = \sup_{\|h\|_{\mathcal{H}} \leq 1} \left[\mathbb{E}_{p_{gp}}[h] - \mathbb{E}_{p_{nn}}[h] \right], \tag{A.21}$$

where $\mathcal{H}$ denotes a reproducing kernel Hilbert space (RKHS) induced by a characteristic kernel K. Similarly to the Wasserstein distance, MMD is an integral probability metric (Müller, 1997). The main difference is the choice of class functions $\mathcal{H}$ as we consider the class of 1-Lipschitz functions for the Wasserstein distance. In fact, under some mild conditions, these metrics are equivalent.

By considering two stochastic processes p_{gp} and p_{nn} at a finite number of measurement points $X_{\mathcal{M}}$, we can obtain the closed form of MMD as follows

$$\text{MMD}^2(p_{gp}, p_{nn}) = \mathbb{E}_{f_{\mathcal{M}}, f'_{\mathcal{M}} \sim p_{gp}}[K(f_{\mathcal{M}}, f'_{\mathcal{M}})] + \mathbb{E}_{f_{\mathcal{M}}, f'_{\mathcal{M}} \sim p_{nn}}[K(f_{\mathcal{M}}, f'_{\mathcal{M}})] \tag{A.22}$$
$$- 2\mathbb{E}_{f_{\mathcal{M}} \sim p_{gp}, f'_{\mathcal{M}} \sim p_{nn}}[K(f_{\mathcal{M}}, f'_{\mathcal{M}})],$$

which can be estimated by using samples from p_{nn} and p_{gp} evaluated at $X_{\mathcal{M}}$ (Gretton et al., 2012). For the MMD estimate, we use an radial basis function (RBF) kernel with a characteristic lengthscale of $l = \sqrt{2D}$, where D is the number of dimensions of the input features, and 5000 samples from p_{nn} and p_{gp}. The measurement set is comprised of 500 test points.

A.2.8 Details on the experiments with functional BNNs and empirical Bayes

In the experiments with fBNN, we keep the same settings as used in Sun et al. (2019)[1]. In particular, we use a GP with RBF kernels for small UCI datasets with less than 2000 data points, while a GP with Neural Kernel Network (NKN) kernels is employed for large UCI datasets.

In the experiments with the empirical Bayes approach (Immer et al., 2021b), following the Authors' repository[2], we use the *Laplace* library (Daxberger et al., 2021) for the implementation. We use the Kronecker-factored Laplace for Hessian approximation. We follow the same experimental protocol of (Immer et al., 2021b) including the optimizer, the early stopping scheme and the frequency of updating the prior.

---

[1] https://github.com/ssydasheng/FBNN
[2] https://github.com/AlexImmer/marglik

A.3   Additional results

A.3.1   Additional results on MAP estimation with GP-induced priors

Fig. A.1 illustrates the comparison between early stopping, and MAP estimation with the FG and GPi-G priors on the UCI regression datasets. We use the same setup as in § 3.6.7. We observe that the predictive performance obtained by MAP with the GPi-G prior outperforms those of early stopping and MAP with the FG prior in most cases.

A.3.2   Tabular results on the UCI benchmarks

Detailed results on the UCI regression and classification datasets are reported in Tables A.4 to A.7.

A.3.3   Convergence of Wasserstein optimization

Figs. A.4 to A.6 depict the progressions of Wasserstein optimization in the UCI regression, UCI classification, and CIFAR10 experiments, respectively.

A.3.4   Additional comparisons with the empirical Bayes approach

We complement the results presented in § 3.6.5 with different scenarios of optimizing the prior and carrying out the inference. In particular, we evaluate our GPi-G prior when employed with the scalable Laplace approximation (LA) approach (Immer et al., 2021a) for inference, refered as "GPi-G prior + LA-GGN". As shown in Fig. A.2, the GPi-G prior still outperforms the fixed prior (FG prior). In addition, we consider the case where the prior optimized on the approximated marginal likelihood (Immer et al., 2021b) is used together with SGHMC. We denote this approach "LA-MargLik + SGHMC". As it can be seen from the results, this prior is not helpful and even worse than the fixed prior when employed with the SGHMC. This is reasonable because the LA-MargLik prior is closely tied with the LA-GGN inference method; the marginal likelihood is optimized jointly with the approximate posterior, and the same optimized hyperparameters might not work just as well for a different posterior approximation.

Data set	N	D	σ_ϵ^2	*Depth*	FG prior	FG+TS	GPi-G prior	FH prior	GPi-H prior	Deep Ensemble
BOSTON	506	13	0.1	1	3.124 ± 1.065	3.065 ± 0.964	$\mathbf{2.823} \pm 0.960$	2.949 ± 1.041	2.850 ± 1.007	3.764 ± 1.122
				2	3.093 ± 1.001	3.020 ± 0.938	2.835 ± 0.922	2.945 ± 0.996	$\mathbf{2.826} \pm 0.909$	3.688 ± 1.147
				4	3.120 ± 0.961	2.975 ± 0.906	$\mathbf{2.869} \pm 0.881$	2.941 ± 0.944	2.931 ± 0.875	3.540 ± 1.166
				8	3.228 ± 0.924	$\mathbf{2.973} \pm 0.849$	2.976 ± 0.957	3.078 ± 1.004	3.110 ± 0.950	3.542 ± 1.068
CONCRETE	1030	8	0.1	1	5.442 ± 0.263	5.419 ± 0.250	$\mathbf{4.765} \pm 0.386$	4.930 ± 0.390	4.781 ± 0.443	5.632 ± 0.563
				2	5.488 ± 0.253	5.388 ± 0.296	$\mathbf{4.801} \pm 0.416$	5.179 ± 0.280	4.822 ± 0.396	5.226 ± 0.631
				4	5.651 ± 0.262	5.326 ± 0.337	5.024 ± 0.321	5.557 ± 0.245	$\mathbf{4.946} \pm 0.384$	5.011 ± 0.560
				8	5.839 ± 0.311	5.289 ± 0.365	5.515 ± 0.339	5.757 ± 0.274	5.184 ± 0.315	$\mathbf{5.124} \pm 0.517$
ENERGY	768	8	0.001	1	0.395 ± 0.071	0.392 ± 0.071	$\mathbf{0.366} \pm 0.080$	0.393 ± 0.074	0.370 ± 0.076	2.252 ± 0.241
				2	0.389 ± 0.062	0.381 ± 0.068	$\mathbf{0.343} \pm 0.071$	0.439 ± 0.063	0.358 ± 0.071	1.382 ± 0.348
				4	0.422 ± 0.051	0.402 ± 0.061	0.396 ± 0.063	0.428 ± 0.061	$\mathbf{0.394} \pm 0.063$	1.049 ± 0.340
				8	0.457 ± 0.052	$\mathbf{0.418} \pm 0.063$	0.475 ± 0.056	0.467 ± 0.055	0.437 ± 0.058	1.041 ± 0.323
KIN8NM	8192	8	0.1	1	0.066 ± 0.002	0.066 ± 0.002	$\mathbf{0.065} \pm 0.002$	$\mathbf{0.065} \pm 0.002$	$\mathbf{0.065} \pm 0.002$	0.071 ± 0.004
				2	0.066 ± 0.002	0.065 ± 0.002	$\mathbf{0.064} \pm 0.002$	0.065 ± 0.002	$\mathbf{0.064} \pm 0.002$	0.068 ± 0.004
				4	0.067 ± 0.002	0.065 ± 0.002	0.065 ± 0.002	0.069 ± 0.002	$\mathbf{0.064} \pm 0.002$	0.070 ± 0.003
				8	0.069 ± 0.002	$\mathbf{0.065} \pm 0.002$	0.070 ± 0.002	0.072 ± 0.002	$\mathbf{0.065} \pm 0.002$	0.071 ± 0.003
NAVAL	11934	16	0.001	1	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	0.004 ± 0.000
				2	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	0.003 ± 0.000
				4	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	$\mathbf{0.000} \pm 0.000$	0.003 ± 0.000
				8	$\mathbf{0.001} \pm 0.000$	$\mathbf{0.001} \pm 0.000$	$\mathbf{0.001} \pm 0.000$	$\mathbf{0.001} \pm 0.000$	$\mathbf{0.001} \pm 0.000$	0.004 ± 0.000
POWER	9568	4	0.05	1	4.003 ± 0.162	4.000 ± 0.164	$\mathbf{3.897} \pm 0.177$	4.022 ± 0.159	3.936 ± 0.170	4.008 ± 0.182
				2	4.008 ± 0.168	3.999 ± 0.170	$\mathbf{3.723} \pm 0.183$	4.054 ± 0.155	3.823 ± 0.179	3.857 ± 0.191
				4	4.064 ± 0.163	4.014 ± 0.165	3.835 ± 0.173	4.163 ± 0.147	$\mathbf{3.814} \pm 0.177$	3.826 ± 0.186
				8	4.105 ± 0.160	4.042 ± 0.165	4.062 ± 0.188	4.205 ± 0.149	3.895 ± 0.167	$\mathbf{3.854} \pm 0.179$
PROTEIN	45730	9	0.5	1	4.374 ± 0.019	4.376 ± 0.015	$\mathbf{3.922} \pm 0.011$	3.973 ± 0.019	3.926 ± 0.019	4.376 ± 0.019
				2	4.379 ± 0.019	4.330 ± 0.024	3.658 ± 0.021	3.713 ± 0.021	$\mathbf{3.644} \pm 0.025$	4.443 ± 0.020
				4	4.509 ± 0.015	4.321 ± 0.019	4.082 ± 0.055	3.976 ± 0.035	$\mathbf{3.774} \pm 0.021$	3.854 ± 0.038
				8	4.530 ± 0.020	4.362 ± 0.014	4.593 ± 0.108	4.148 ± 0.031	$\mathbf{3.980} \pm 0.022$	3.997 ± 0.027
WINE	1599	11	0.5	1	0.637 ± 0.042	0.636 ± 0.044	0.618 ± 0.045	0.633 ± 0.044	0.622 ± 0.045	$\mathbf{0.612} \pm 0.020$
				2	0.641 ± 0.044	0.641 ± 0.044	$\mathbf{0.609} \pm 0.046$	0.637 ± 0.044	0.613 ± 0.046	0.615 ± 0.025
				4	0.650 ± 0.045	0.649 ± 0.046	0.608 ± 0.046	0.637 ± 0.044	$\mathbf{0.602} \pm 0.048$	$\mathbf{0.602} \pm 0.031$
				8	0.662 ± 0.049	0.660 ± 0.049	0.632 ± 0.046	0.646 ± 0.046	0.621 ± 0.048	$\mathbf{0.609} \pm 0.026$

Table A.4: Average test root mean squared error (RMSE) on UCI regression datasets (errors are ± 1 standard error). Bold results indicate the best performance. Here, N is the size of dataset, D is the number of input dimensions, σ_ϵ^2 is the noise variance, and *Depth* is the number of hidden layers of the MLP.
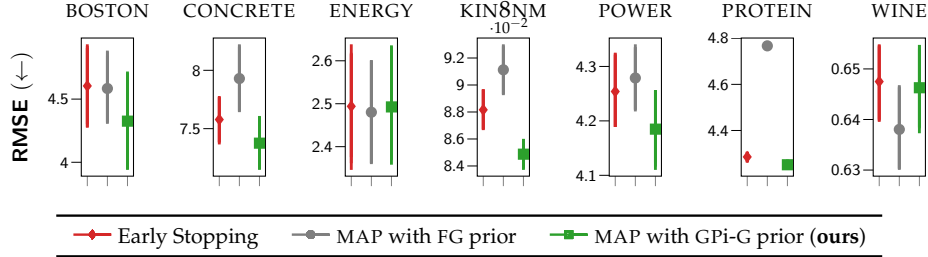
Figure A.1: Comparison between early stopping and MAP estimations with respect to the FG and GPi-G priors on the UCI regression datasets.
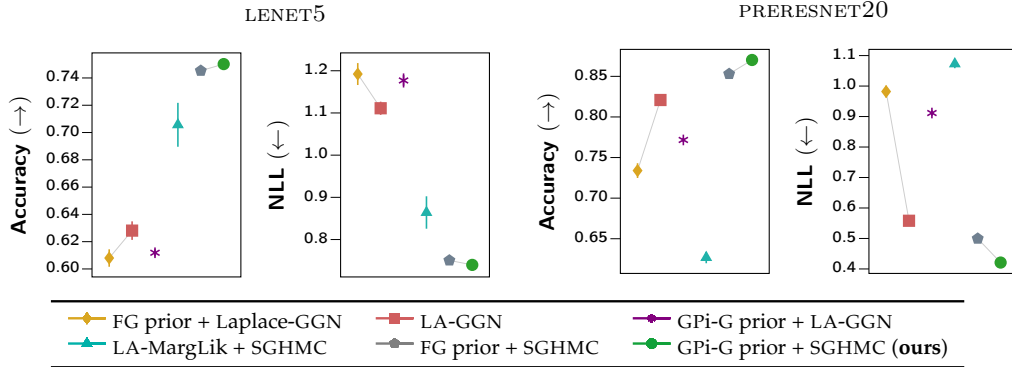
Figure A.2: Comparison with empirical Bayes and functional inference methods on CIFAR10 dataset.

Data set	Classes	N_{train}	N_{test}	D	FG prior	FG+TS	GPi-G prior	FH prior	GPi-H prior	Deep Ensemble
EEG	2	10980	4000	14	82.26 ± 7.17	81.63 ± 8.09	94.13 ± 1.96	93.31 ± 3.67	$\mathbf{94.69} \pm 2.17$	89.94 ± 4.98
HTRU2	2	12898	5000	8	97.94 ± 0.23	97.93 ± 0.24	$\mathbf{98.03} \pm 0.24$	98.01 ± 0.20	98.02 ± 0.26	98.01 ± 0.24
MAGIC	2	14020	5000	10	86.95 ± 0.39	87.15 ± 0.34	88.37 ± 0.29	87.65 ± 0.25	$\mathbf{88.49} \pm 0.26$	87.87 ± 0.27
MINIBOO	2	120064	10000	50	90.81 ± 0.22	90.99 ± 0.21	92.74 ± 0.39	93.26 ± 0.28	$\mathbf{93.37} \pm 0.27$	91.42 ± 0.21
LETTER	26	15000	5000	16	90.45 ± 0.41	90.75 ± 0.37	96.90 ± 0.29	97.41 ± 0.26	$\mathbf{97.67} \pm 0.20$	96.46 ± 0.27
DRIVE	11	48509	10000	48	98.55 ± 0.10	98.71 ± 0.09	99.69 ± 0.04	99.71 ± 0.04	$\mathbf{99.74} \pm 0.05$	99.31 ± 0.06
MOCAP	5	68095	10000	37	98.80 ± 0.10	98.98 ± 0.09	99.24 ± 0.10	99.41 ± 0.08	$\mathbf{99.49} \pm 0.07$	99.12 ± 0.09

Table A.5: Average test accuracy (%) on UCI classification datasets (errors are ± 1 standard error). Bold results indicate the best performance. Here, *Classes* is the number of classes, N_{train}, N_{test} is the sizes of training set and test set, respectively; D is the number of input dimensions.

Data set	N	D	σ_ϵ^2	Depth	FG prior	FG+TS	GPi-G prior	FH prior	GPi-H prior	Deep Ensemble
BOSTON	506	13	0.1	1	2.558 ± 0.294	2.582 ± 0.365	2.472 ± 0.153	2.498 ± 0.212	$\mathbf{2.469} \pm 0.160$	3.177 ± 1.188
				2	2.541 ± 0.251	2.563 ± 0.343	2.475 ± 0.115	2.489 ± 0.196	$\mathbf{2.458} \pm 0.110$	3.249 ± 1.111
				4	2.548 ± 0.207	2.542 ± 0.304	2.475 ± 0.095	$\mathbf{2.473} \pm 0.140$	2.486 ± 0.080	3.448 ± 1.483
				8	2.581 ± 0.170	2.541 ± 0.259	$\mathbf{2.474} \pm 0.094$	2.496 ± 0.128	2.529 ± 0.083	3.004 ± 0.915
CONCRETE	1030	8	0.1	1	3.104 ± 0.039	3.106 ± 0.048	$\mathbf{3.004} \pm 0.050$	3.027 ± 0.051	3.007 ± 0.057	3.113 ± 0.214
				2	3.114 ± 0.037	3.099 ± 0.054	3.028 ± 0.050	3.066 ± 0.036	$\mathbf{3.024} \pm 0.044$	3.065 ± 0.259
				4	3.145 ± 0.040	3.091 ± 0.055	3.060 ± 0.037	3.127 ± 0.039	$\mathbf{3.056} \pm 0.046$	3.034 ± 0.251
				8	3.184 ± 0.047	3.092 ± 0.058	3.128 ± 0.046	3.169 ± 0.042	3.109 ± 0.037	$\mathbf{3.054} \pm 0.189$
ENERGY	768	8	0.001	1	0.496 ± 0.216	0.496 ± 0.222	$\mathbf{0.417} \pm 0.227$	0.489 ± 0.210	0.425 ± 0.210	2.076 ± 0.500
				2	0.471 ± 0.174	0.454 ± 0.196	$\mathbf{0.347} \pm 0.150$	0.648 ± 0.116	0.392 ± 0.180	2.062 ± 1.014
				4	0.558 ± 0.145	0.506 ± 0.180	0.478 ± 0.168	0.681 ± 0.080	$\mathbf{0.476} \pm 0.170$	1.935 ± 0.981
				8	0.636 ± 0.123	0.585 ± 0.134	0.657 ± 0.154	0.867 ± 0.056	$\mathbf{0.562} \pm 0.152$	1.713 ± 0.736
KIN8NM	8192	8	0.1	1	-1.233 ± 0.018	-1.238 ± 0.017	-1.238 ± 0.015	-1.243 ± 0.016	-1.241 ± 0.015	$\mathbf{-1.317} \pm 0.061$
				2	-1.227 ± 0.018	-1.243 ± 0.017	-1.233 ± 0.012	-1.230 ± 0.016	-1.241 ± 0.014	$\mathbf{-1.317} \pm 0.076$
				4	-1.201 ± 0.013	-1.235 ± 0.015	-1.219 ± 0.011	-1.180 ± 0.015	-1.223 ± 0.013	$\mathbf{-1.256} \pm 0.074$
				8	-1.169 ± 0.015	-1.222 ± 0.014	-1.159 ± 0.020	-1.138 ± 0.015	-1.211 ± 0.013	$\mathbf{-1.264} \pm 0.070$
NAVAL	11934	16	0.001	1	-6.943 ± 0.028	-6.935 ± 0.028	-6.944 ± 0.031	$\mathbf{-6.946} \pm 0.028$	-6.923 ± 0.062	-5.172 ± 0.227
				2	-6.410 ± 0.087	-6.373 ± 0.099	$\mathbf{-6.430} \pm 0.156$	-6.429 ± 0.097	-6.397 ± 0.098	-5.248 ± 0.274
				4	-6.289 ± 0.079	-6.291 ± 0.064	$\mathbf{-6.359} \pm 0.063$	-6.323 ± 0.043	-6.347 ± 0.051	-5.122 ± 0.259
				8	-5.869 ± 0.046	-5.893 ± 0.042	-5.886 ± 0.040	-5.886 ± 0.041	$\mathbf{-5.895} \pm 0.051$	-4.934 ± 0.428
POWER	9568	4	0.05	1	2.807 ± 0.042	2.807 ± 0.043	$\mathbf{2.780} \pm 0.044$	2.812 ± 0.042	2.790 ± 0.043	2.799 ± 0.045
				2	2.808 ± 0.043	2.806 ± 0.044	$\mathbf{2.738} \pm 0.042$	2.819 ± 0.040	2.761 ± 0.043	2.754 ± 0.053
				4	2.821 ± 0.039	2.809 ± 0.042	2.766 ± 0.040	2.844 ± 0.035	2.762 ± 0.041	$\mathbf{2.738} \pm 0.059$
				8	2.833 ± 0.036	2.817 ± 0.040	2.821 ± 0.043	2.857 ± 0.032	2.783 ± 0.038	$\mathbf{2.753} \pm 0.037$
PROTEIN	45730	9	0.5	1	2.894 ± 0.004	2.894 ± 0.003	2.798 ± 0.002	2.809 ± 0.003	2.799 ± 0.004	$\mathbf{2.753} \pm 0.009$
				2	2.892 ± 0.004	2.881 ± 0.005	2.752 ± 0.004	2.760 ± 0.004	$\mathbf{2.748} \pm 0.004$	2.796 ± 0.016
				4	2.916 ± 0.003	2.875 ± 0.004	2.825 ± 0.011	2.801 ± 0.007	2.764 ± 0.004	$\mathbf{2.606} \pm 0.039$
				8	2.919 ± 0.004	2.883 ± 0.003	2.933 ± 0.025	2.838 ± 0.006	2.802 ± 0.004	$\mathbf{2.658} \pm 0.013$
WINE	1599	11	0.5	1	0.973 ± 0.080	0.983 ± 0.090	$\mathbf{0.929} \pm 0.067$	0.962 ± 0.079	0.936 ± 0.069	1.008 ± 0.162
				2	0.983 ± 0.082	0.990 ± 0.087	$\mathbf{0.915} \pm 0.063$	0.974 ± 0.082	0.922 ± 0.067	1.081 ± 0.193
				4	0.999 ± 0.085	1.004 ± 0.090	0.915 ± 0.064	0.973 ± 0.081	$\mathbf{0.908} \pm 0.064$	1.774 ± 0.468
				8	1.016 ± 0.093	1.023 ± 0.095	0.953 ± 0.075	0.988 ± 0.084	0.938 ± 0.072	$\mathbf{0.927} \pm 0.100$

Table A.6: Average test NLL in nats on UCI regression datasets (errors are ± 1 standard error). Bold results indicate the best performance. Here, N is the size of dataset, D is the number of input dimensions, σ_ϵ^2 is the noise variance, and *Depth* is the number of hidden layers of the MLP.

A.3.5 Additional results with full-batch Hamiltonian Monte Carlo

Table A.8 shows a comparison between full-batch Hamiltonian Monte Carlo (HMC) and SGHMC using the FG and our GPi-G priors on small UCI regression datasets. We use the no-u-turn sampler (NUTS) extension (Hoffman and Gelman, 2014) of HMC with the NumPyro's implementation (Phan et al., 2019). NUTS adaptively sets the trajectory length of HMC, which along with the adaptation of the mass matrix and the step size. We have simulated 4 chains with a burn-in phase of 200 iterations and 200 collected samples for each chain. We see that SGHMC performs remarkably similar to a carefully tuned HMC algorithm, despite the discretization error.

A.3.6 Additional discussion on the optimization of Wasserstein distance

In the Algorithm 2, we have opted to separate the two optimization procedures for the Lipschitz function ϕ_θ and the Wasserstein distance. We acknowledge that the two could have been optimized jointly in a single loop, as Eq. 3.7 defines a minimax problem. However, our choice allows ϕ_θ to be stabilized before a single Wasserstein minimization step takes place. In fact, this is a common trick to make convergence more stable (see e.g., the original Goodfellow et al. (2014) paper, which suggests to

Data set	Classes	N_{train}	N_{test}	D	FG prior	FG+TS	GPi-G prior	FH prior	GPi-H prior	Deep Ensemble
EEG	2	10980	4000	14	0.404 ± 0.120	0.406 ± 0.129	0.179 ± 0.046	0.179 ± 0.075	$\mathbf{0.150} \pm 0.053$	0.240 ± 0.097
HTRU2	2	12898	5000	8	0.071 ± 0.007	0.072 ± 0.007	0.066 ± 0.008	0.068 ± 0.007	$\mathbf{0.066} \pm 0.008$	0.067 ± 0.008
MAGIC	2	14020	5000	10	0.316 ± 0.006	0.312 ± 0.005	0.286 ± 0.005	0.298 ± 0.004	$\mathbf{0.284} \pm 0.005$	0.294 ± 0.005
MINIBOO	2	120064	10000	50	0.218 ± 0.004	0.215 ± 0.004	0.179 ± 0.007	0.168 ± 0.004	$\mathbf{0.165} \pm 0.004$	0.207 ± 0.004
LETTER	26	15000	5000	16	0.445 ± 0.008	0.409 ± 0.008	0.166 ± 0.006	0.128 ± 0.005	$\mathbf{0.115} \pm 0.005$	0.147 ± 0.006
DRIVE	11	48509	10000	48	0.098 ± 0.002	0.088 ± 0.002	0.028 ± 0.001	0.023 ± 0.001	$\mathbf{0.022} \pm 0.001$	0.049 ± 0.002
MOCAP	5	68095	10000	37	0.060 ± 0.002	0.050 ± 0.002	0.032 ± 0.002	0.027 ± 0.001	$\mathbf{0.021} \pm 0.001$	0.040 ± 0.002

Table A.7: Average test NLL in nats on UCI classification datasets (errors are ± 1 standard error). Bold results indicate the best performance. Here, *Classes* is the number of classes; the N_{train}, N_{test} is the sizes of training set and test set, respectively; D is the number of input dimensions.

Data set	σ_ϵ^2	FG prior		GPi-G prior	
		HMC	SGHMC	HMC	SGHMC
BOSTON	0.1	3.065 ± 1.006	3.093 ± 1.001	2.821 ± 0.907	2.835 ± 0.922
CONCRETE	0.1	5.369 ± 0.294	5.488 ± 0.253	4.715 ± 0.431	4.801 ± 0.416
ENERGY	0.001	0.386 ± 0.064	0.389 ± 0.062	0.339 ± 0.075	0.343 ± 0.071
POWER	0.05	3.931 ± 0.165	4.008 ± 0.168	3.438 ± 0.201	3.723 ± 0.183
WINE	0.5	0.637 ± 0.043	0.641 ± 0.044	0.606 ± 0.046	0.609 ± 0.046

Table A.8: Average test RMSE results of full-batch HMC and SGHMC on UCI regression datasets (errors are ± 1 standard error). We use a MLP with two hidden layers of 100 neurons. σ_ϵ^2 is the noise variance.

allow more training of the discriminator for each step of the generator). Fig. A.3 illustrates the convergence behavior of these two algorithmic choices measured by the squared MMD between the target GP prior and the optimized BNN prior on the UCI regression datasets (see Appendix A.2.7 for the experimental protocol). Our optimization strategy demonstrates a much more stable convergence compared to the joint optimization approach.
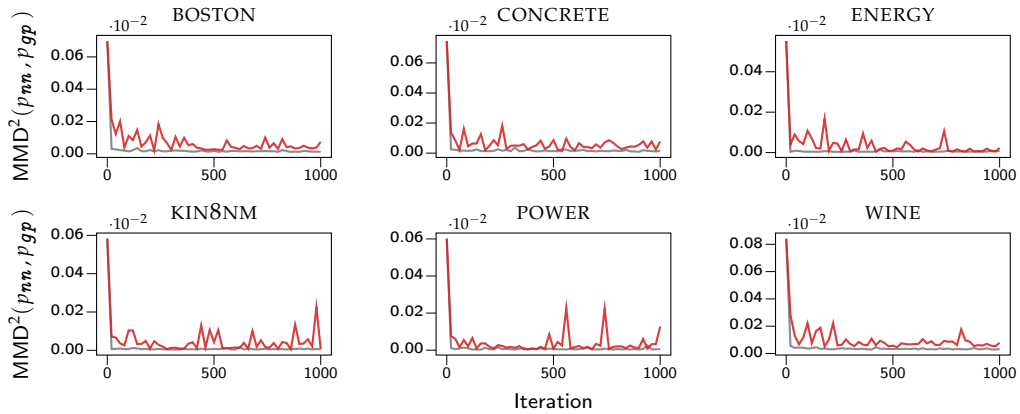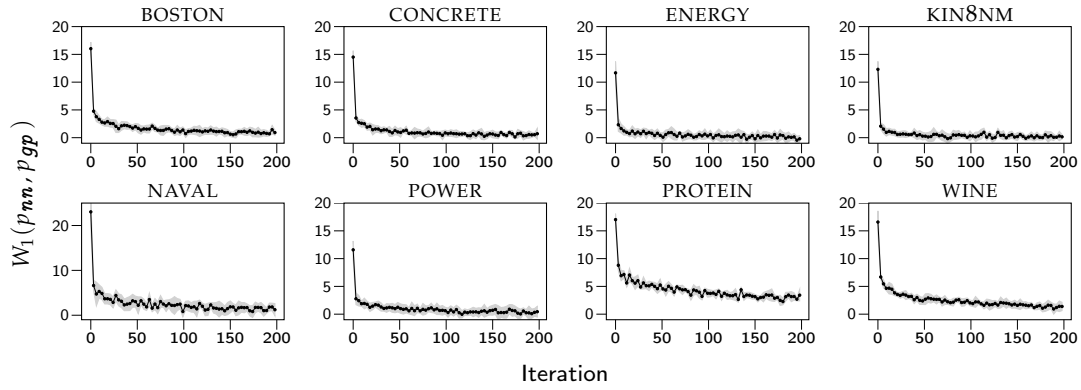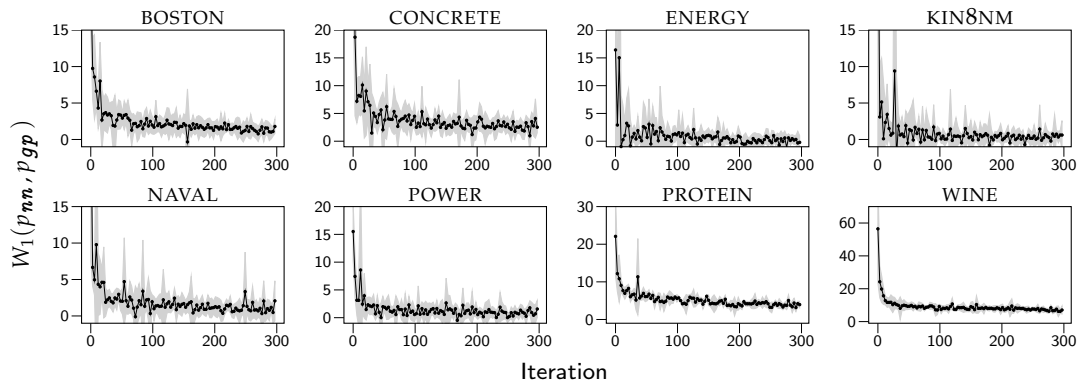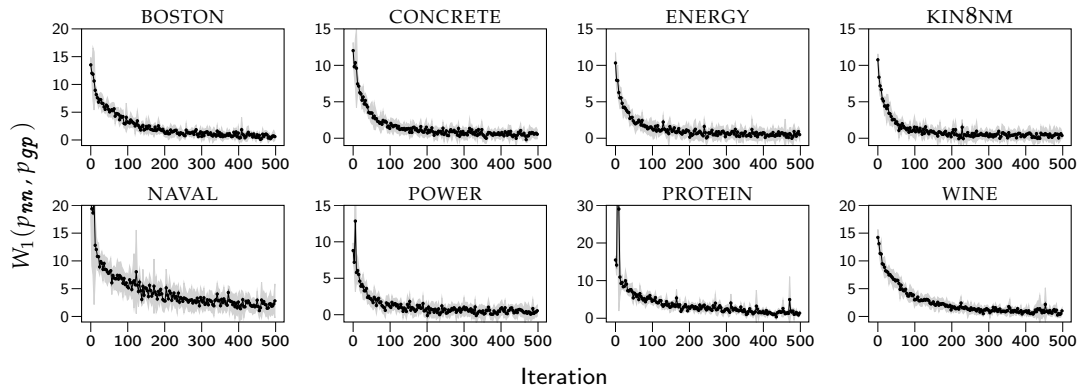

Figure A.3: Comparison between strategies to optimize the Lipschitz function and the Wasserstein distance: (▬) our strategy of separating these two operations; and (▬) the strategy of joint optimization. Here, the convergence is measured by the squared MMD between the target GP prior and the optimized BNN prior.
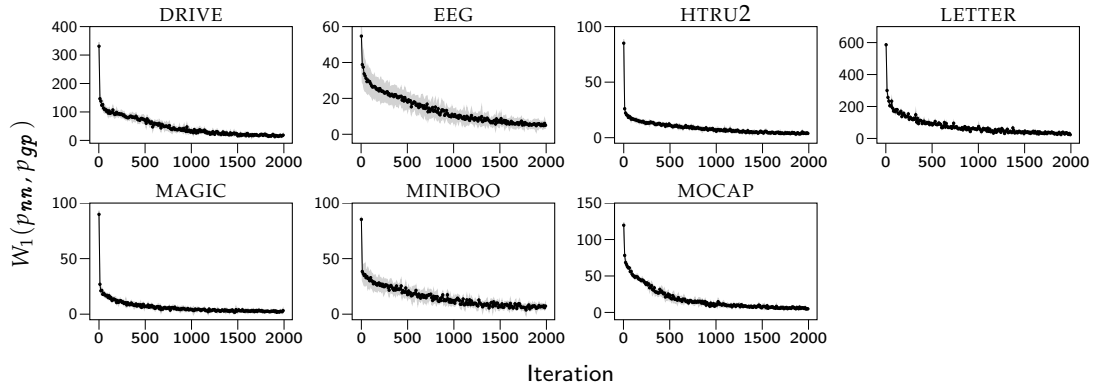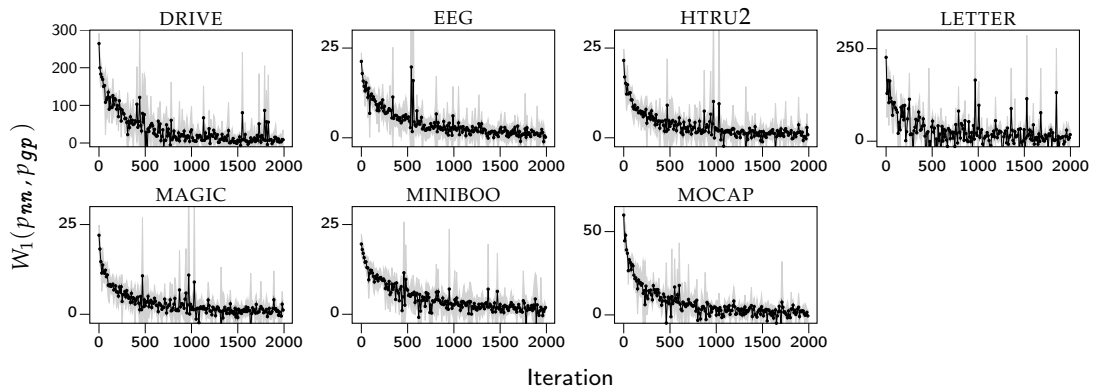
(a) GPi-G prior

(b) GPi-H prior

(c) GPi-NF prior

Figure A.4: Convergence of Wasserstein optimization for two-layer MLP on the UCI regression datasets.
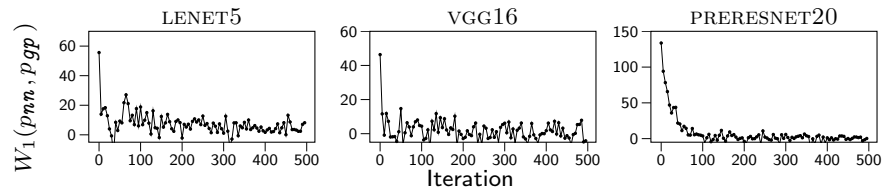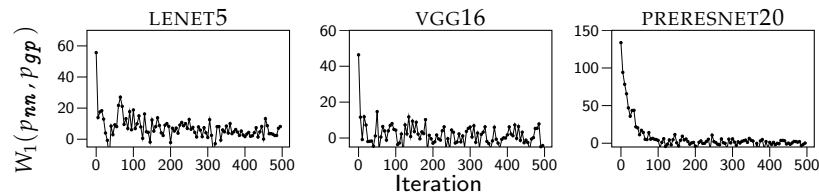
(a) GPi-G prior

(b) GPi-H prior

Figure A.5: Convergence of Wasserstein optimization for two-layer MLP on the UCI classification datasets.

(a) GPi-G prior

(b) GPi-H prior

Figure A.6: Convergence of Wasserstein optimization for CNN on the CIFAR10 dataset.

Appendix B

Appendix for Chapter 4

B.1 Derivation of Distributional Sliced-Wasserstein Distance

In this section, we review some key results on the Wasserstein distance. Given two probability measures π, ρ, both defined on $\mathbb{R}^D$ for simplicity, the p-Wasserstein distance between π and ρ is given by

$$W_p^p(\pi, \rho) = \inf_{\gamma \in \Gamma(\pi, \rho)} \int \|x - y\|^p \gamma(x, y) \mathrm{d}x \mathrm{d}y, \tag{B.1}$$

where $\Gamma(\pi, \rho)$ is the set of all possible distributions $\gamma(x, y)$ such that the marginals are $\pi(x)$ and $\rho(y)$ (Villani, 2008). While usually analytically unavailable, for $D = 1$ the distance has the following closed form solution,

$$W_p^p(\pi, \rho) = \int_0^1 |F_\pi^{-1}(z) - F_\rho^{-1}(z)|^p \mathrm{d}z, \tag{B.2}$$

where F_π and F_ρ are the cumulative density functions (CDFs) of π and ρ, respectively.

B.1.1 (Distributional) sliced-Wasserstein distance

The main idea underlying the distributional sliced-Wasserstein distance (DSWD) is to project the challenging estimation of distances for high-dimensional distributions into simpler estimation of multiple distances in one dimension, which all have closed-form solution (Eq. B.2). The projection is done using the Radon transform $\mathcal{R}$, an operator that maps a density function φ defined in $\mathbb{R}^D$ to the set of its integrals over hyperplanes in $\mathbb{R}^D$,

$$\mathcal{R}\varphi(t, \boldsymbol{\theta}) := \int \varphi(z) \delta(t - z^\top \boldsymbol{\theta}) \mathrm{d}z, \quad \forall t \in \mathbb{R}, \ \forall \boldsymbol{\theta} \in \mathbb{S}^{D-1}, \tag{B.3}$$

where $\mathbb{S}^{D-1}$ is the unit sphere in $\mathbb{R}^D$ and $\delta(\cdot)$ is the Dirac delta (Helgason, 2010). Using the Radon transform, for a given $\boldsymbol{\theta}$ we can project the two densities π and ρ

into one dimension,

$$W_p^p(\pi,\rho) = \int_{\mathbb{S}^{D-1}} W_p^p\left(\mathcal{R}\pi(t,\boldsymbol{\theta}), \mathcal{R}\rho(t,\boldsymbol{\theta})\right) \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{K}\sum_{i=1}^{K} W_p^p\left(\mathcal{R}\pi(t,\boldsymbol{\theta}_i), \mathcal{R}\rho(t,\boldsymbol{\theta}_i)\right),$$

(B.4)

where the approximation comes from using Monte-Carlo integration by sampling $\boldsymbol{\theta}_i$ uniformly in $\mathbb{S}^{D-1}$ (Bonneel et al., 2015). While having significant computational advantages, this approach might require to draw many unimportant projections that are computationally exhausting and that provide a minimal improvement on the overall distance approximation.

The *distributional sliced-Wasserstein distance* (DSW) (Nguyen et al., 2021) solves this issue by finding the optimal probability measure of slices $\sigma(\boldsymbol{\theta})$ on the unit sphere $\mathbb{S}^{D-1}$ and it's defined as follows,

$$DSW_p(\pi,\rho;C) := \sup_{\sigma\in\mathbb{M}_C} \left(\mathbb{E}_{\sigma(\boldsymbol{\theta})} W_p^p\left(\mathcal{R}\pi(t,\boldsymbol{\theta}), \mathcal{R}\rho(t,\boldsymbol{\theta})\right)\right)^{1/p},$$

(B.5)

where, for $C > 0$, $\mathbb{M}_C$ is the set of probability measures σ such that $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\theta}'\sim\sigma}\left[\boldsymbol{\theta}^\top\boldsymbol{\theta}'\right] \leq C$ (a constraint that aims to avoid directions to lie in only one small area). Critically, the definition of DSWD in Eq. B.5 does not suffer from the curse of dimensionality, indeed Nguyen et al. (2021) showed that the statistical error of this estimation scales down with $C_D \cdot n^{-\frac{1}{2}}$, where C_D is a constant depending on dimension D. Furthermore, while generally we have that $DSW_p(\pi,\rho) \leq W_p(\pi,\rho)$, it can be proved that under mild assumptions on C, the two distances are topological equivalent, i.e. converging in distribution on DSW_p implies the convergence on W_p see Theorem 2 in Nguyen et al., 2021.

The direct computation of DSW_p in Eq. B.5 is still challenging but it admits an equivalent dual form,

$$\sup_{h\in\mathcal{H}} \left\{\left(\mathbb{E}_{\bar{\sigma}(\boldsymbol{\theta})}\left[W_p^p\left(\mathcal{R}\pi(t,h(\boldsymbol{\theta})), \mathcal{R}\rho(t,h(\boldsymbol{\theta}))\right)\right]\right)^{1/p} - \lambda_C\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\theta}'\sim\bar{\sigma}}\left[\left|h(\boldsymbol{\theta})^\top h(\boldsymbol{\theta}')\right|\right]\right\} + \lambda_C C,$$

(B.6)

where $\bar{\sigma}$ is a uniform distribution in $\mathbb{S}^{D-1}$, $\mathcal{H}$ is a class of all Borel measurable functions $\mathbb{S}^{D-1} \to \mathbb{S}^{D-1}$ and λ_C is a regularization hyperparameter. The formulation in Eq. B.6 is obtained by employing the Lagrangian duality theorem and by reparameterizing $\sigma(\boldsymbol{\theta})$ as push-forward transformation of a uniform measure in $\mathbb{S}^{D-1}$ via h. Now, by parameterizing h using a deep neural network[1] with parameters $\boldsymbol{\phi}$, defined as $h_{\boldsymbol{\phi}}$, Eq. B.6 becomes an optimization problem with respect to the network parameters. The final step is to approximate the analytically intractable expectations with Monte

---

[1] We use a single multi layer perceptron (MLP) layer with normalized output as the h function.

Carlo integration,

$$DSW_p(\pi, \rho) \approx$$

$$\max_{\boldsymbol{\phi}} \left\{ \left[\frac{1}{K} \sum_{i=1}^{K} \left[W_p^p(\mathcal{R}\pi(t, h_{\boldsymbol{\phi}}(\boldsymbol{\theta}_i)), \mathcal{R}\rho(t, h_{\boldsymbol{\phi}}(\boldsymbol{\theta}_i))) \right] \right]^{1/p} \frac{\lambda_C}{K^2} \sum_{i,j=1}^{K} |h_{\boldsymbol{\phi}}(\boldsymbol{\theta}_i)^\top h_{\boldsymbol{\phi}}(\boldsymbol{\theta}_j)| + \lambda_C C \right\},$$

$$(B.7)$$

where $\boldsymbol{\theta}_i$ are uniform samples from the unit sphere $\mathbb{S}^{D-1}$ and $\forall t \in \mathbb{R}$. Finally, we can use stochastic gradient methods to update $\boldsymbol{\phi}$ and then use the resulting optima for the estimation of the original distance.

B.2 Numerical Implementation of Sliced-Wasserstein Distance

B.2.1 Wasserstein distance between two empirical 1D distributions

The Wasserstein distance between two one-dimensional distributions π and ρ is defined as in Eq. B.2. The integral in this equation can be numerically estimated by using the midpoint Riemann sum:

$$\int_0^1 |F_\pi^{-1}(z) - F_\rho^{-1}(z)|^p dz \approx \frac{1}{M} \sum_{m=1}^{M} |F_\pi^{-1}(z_m) - F_\rho^{-1}(z_m)|^p, \qquad (B.8)$$

where $z_m = \frac{2m-1}{M}$, M is the number of points used to approximate the integral. If we only have samples from the distributions, $x_m \sim \pi$ and $y_m \sim \rho$, we can obtain the empirical densities as follows

$$\pi(x) \approx \pi_M(x) = \frac{1}{M} \sum_{m=1}^{M} \delta(x - x_m), \qquad (B.9)$$

$$\rho(y) \approx \rho_M(y) = \frac{1}{M} \sum_{m=1}^{M} \delta(y - y_m), \qquad (B.10)$$

where δ is the Dirac delta function. The corresponding empirical cumulative density functions are

$$F_\pi(z) \approx F_{\pi,M}(z) = \frac{1}{M} \sum_{m=1}^{M} u(z - x_m), \qquad (B.11)$$

$$F_\rho(z) \approx F_{\rho,M}(z) = \frac{1}{M} \sum_{m=1}^{M} u(z - y_m), \qquad (B.12)$$

where M is the number of samples, $u(\cdot)$ is the step function.

Calculating the Wasserstein distance with the empirical distribution function is computationally attractive. To do that, we first sort x_ms in an ascending order, such that

$x_{i[m]} \leq x_{i[m+1]}$, where $i[m]$ is the index of the sorted x_ms. It is straightforward to show that $F_{\pi,M}^{-1}(z_m) = x_{i[m]}$. Thus, the Wasserstein distance can be approximated as follows

$$W_p^p(\pi, \rho) \approx \frac{1}{M} \sum_{m=1}^{M} |x_{i[m]} - y_{j[m]}|^p. \tag{B.13}$$

B.2.2 Slicing empirical distribution

According to the equation Eq. B.3, the marginal densities (i.e. slices) of the distribution π can be obtained as follows

$$\mathcal{R}\pi(t, \boldsymbol{\theta}) = \int \pi(\boldsymbol{x})\delta(t - \boldsymbol{x}^\top\boldsymbol{\theta})d\boldsymbol{x}, \quad \forall t \in \mathbb{R}. \tag{B.14}$$

Because, in practice, only samples from the distributions are available we aim to calculate a Radon slice of the empirical distribution of M samples $\pi_M = \frac{1}{M}\sum_{m=1}^{M}\delta(\boldsymbol{x} - \boldsymbol{x}_m)$:

$$\mathcal{R}\pi(t, \boldsymbol{\theta}) \approx \frac{1}{M}\sum_{m=1}^{M}\int \delta(\boldsymbol{x} - \boldsymbol{x}_m)\delta(t - \boldsymbol{x}^\top\boldsymbol{\theta})d\boldsymbol{x} \tag{B.15}$$

$$= \frac{1}{M}\sum_{m=1}^{M}\delta(t - \boldsymbol{x}_m^\top\boldsymbol{\theta}). \tag{B.16}$$

By using the approximation in Eq. B.16 and the empirical implementation of 1D Wasserstein distance (Eq. B.13), we are able to compute a proxy to the original distance in Eq. B.5.

B.3 Pseudocode of Prior Optimization Procedure

Algorithm 5 desribes the procedure of prior optimization for Bayesian autoencoders (BAEs).

B.4 PCA of the SGD Trajectory

Inspired by Izmailov et al. (2019), we use the subspace spanned by the SGD trajectory to visualize neural network's parameters in a low-dimensional space. This subspace is cheap to construct and can capture many of the sharp directions of the loss surface (Izmailov et al., 2019; Li et al., 2018; Maddox et al., 2019). More specifically, we perform SGD starting from a MAP solution with a constant learning rate. Here, the loss function is the negative log joint likelihood of the BAE:

$$\mathcal{L}(\boldsymbol{w}) = -\frac{N}{M}\sum_{i=1}^{M}\log p(\boldsymbol{y}_i \mid \boldsymbol{w}) - \log p(\boldsymbol{w}), \tag{B.17}$$

Algorithm 5: Prior Optimization

Input: Empirical distribution $\tilde{\pi}(\boldsymbol{y})$; prior over parameters $p_{\boldsymbol{\psi}}(\boldsymbol{w})$; number of prior samples N_S; mini-batch size N_B; number of random projections K; regularization coefficient λ_C.

Output: The optimized prior's parameters $\boldsymbol{\psi}$

1 **while** $\boldsymbol{\psi}$ *has not converged* **do**

2   Sample $\boldsymbol{x} \stackrel{\text{def}}{=} \boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1}^{N_B}$ from $\tilde{\pi}(\boldsymbol{y})$ // `Sample input data`

3   Sample $\mathcal{W} = \{\boldsymbol{w}_i\}_{i=1}^{N_S}$ from $p_{\boldsymbol{\psi}}(\boldsymbol{w})$ // `Sample parameters from the prior`

4   **foreach** $\boldsymbol{w}_i \in \mathcal{W}$ **do**

        `/* Following steps are performed in a batch manner          */`

5        $f_i = (f_{\text{dec}} \circ f_{\text{enc}})(\boldsymbol{x})$ // `Compute the functional outputs from Autoencoder`

6   Sample $\tilde{\boldsymbol{y}}_i$ from $p(\boldsymbol{y} \mid f_i)$ // `Sample from the likelihood`

7   Gather samples $\tilde{\boldsymbol{y}} = \cup\{\tilde{\boldsymbol{y}}_i\}_{i=1}^{N_s}$

8     $\mathcal{L} = DSW_2(\boldsymbol{y}, \tilde{\boldsymbol{y}}; K, \lambda_C)$ // `Compute the` DSW_2 `distance using` **Eq. B.7**

9     $\boldsymbol{\psi} \leftarrow \text{Optimizer}(\boldsymbol{\psi}, \nabla_{\boldsymbol{\psi}} \mathcal{L})$ // `Update prior's parameters`

10 **Return:** $\boldsymbol{\psi}$

---

where M is the mini-batch size and N is the size of training data. We store the deviations $\boldsymbol{a}_i = \overline{\boldsymbol{w}} - \boldsymbol{w}_i$ for the last M epochs, where $\overline{\boldsymbol{w}}$ is the running average of the first moment, M is determined by the amount of memory we can use. Then we perform PCA based on randomized SVD (Halko et al., 2011) on the matrix $\boldsymbol{A}$ comprised of vectors $\boldsymbol{a}_1, ..., \boldsymbol{a}_M$ to construct the subspace. The procedure is summarized in Algorithm 6.

Algorithm 6: Subspace construction with PCA

Input: Pretrained paremeters $\boldsymbol{w}_{\text{MAP}}$; learning rate η; number of steps τ; momentum update frequency c; maximum number of columns M in deviation matrix $\boldsymbol{A}$.

Output: Shift vector $\overline{\boldsymbol{w}}$; projection matrix $\boldsymbol{P}$ for subspace.

1 $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{w}_{\text{MAP}}$ // `Initialize mean`

2 **for** $i \leftarrow 1, 2, ..., T$ **do**

3     $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_{i-1} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}_{i-1})$ // `Perform SGD update`

4   **if** $\text{MOD}(i, c) = 0$ **then**

5        $n \leftarrow i/c$ // `Number of models`

6        $\overline{\boldsymbol{w}} \leftarrow \frac{n\overline{\boldsymbol{w}} + \boldsymbol{w}_i}{n+1}$ // `Update mean`

7   **if** $\text{NUM\_COLS}(\boldsymbol{A}) = M$ **then**

8           $\text{REMOVE\_COL}(\boldsymbol{A}[:, 1])$

9        $\text{APPEND\_COL}(\boldsymbol{A}, \boldsymbol{w}_{\text{i}} - \overline{\boldsymbol{w}})$ // `Store deviation`

10 $\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}^\top \leftarrow SVD(\boldsymbol{A})$ // `Perform truncated SVD`

11 **Return:** $\overline{\boldsymbol{w}}, \boldsymbol{P} = \boldsymbol{S}\boldsymbol{V}^\top$

B.5   Additional Details on Experimental Settings

B.5.1   Experimental environment

In our experiments, we use 4 workstations, which have the following specifications:

- **GPU**: NVIDIA Tesla P100 PCIe 16 GB.

- **CPU**: Intel(R) Xeon(R) (4 cores) @ 2.30GHz.

- **Memory**: 25.5 GiB (DDR3).

B.5.2   Preprocessing data

- MNIST (Lecun et al., 1998): The dataset is publicly available at `http://yann.lecun.com/exdb/mnist`. We keep the original resolution of $1 \times 28 \times 28$ of the MNIST dataset.

- FREY-YALE (Dai et al., 2015): The FREY and YALE datasets are publicly availaibe at `http://cs.nyu.edu/~roweis/data.html` and `http://vision.ucsd.edu/extyaleb/CroppedYaleBZip`, respectivey. All the images of FREY and YALE datasets are resized to the $1 \times 28 \times 28$ resolution.

- CELEBA (Liu et al., 2015): The dataset is publicly available at `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`. According to (Dinh et al., 2017b), we pre-process CELEBA images by first taking a 148×148 center crop and then resizing to the $3 \times 64 \times 64$ resolution.

B.5.3   Network architectures

In our experiments, we use convolutional networks for modeling both encoders and decoders. For a fair comparison, we employ the same network architecture for all models. The network's parameters are initialized by using the default scheme in PyTorch (Paszke et al., 2019).

Table B.1 shows details on the network architectures used in our experimental campaign.

	MNIST	FREY-YALE	CELEBA
ENCODER:	$x \in \mathbb{R}^{1 \times 28 \times 28}$	$x \in \mathbb{R}^{1 \times 28 \times 28}$	$x \in \mathbb{R}^{3 \times 64 \times 64}$
	$\rightarrow \text{CONV}_{32} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{64} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{64} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONV}_{64} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{128} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONV}_{64} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{256} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONV}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{256} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONV}_{512} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{FLATTEN} \rightarrow \text{FC}_{50 \times M}$	$\rightarrow \text{FLATTEN} \rightarrow \text{FC}_{50 \times M}$	$\rightarrow \text{FLATTEN} \rightarrow \text{FC}_{50 \times M}$
DECODER:	$z \in \mathbb{R}^{50} \rightarrow \text{FC}_{7 \times 7 \times 128}$	$z \in \mathbb{R}^{50} \rightarrow \text{FC}_{7 \times 7 \times 256}$	$z \in \mathbb{R}^{50} \rightarrow \text{FC}_{8 \times 8 \times 512}$
	$\rightarrow \text{LEAKYRELU}$	$\rightarrow \text{LEAKYRELU}$	$\rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONVT}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{256} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{512} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONVT}_{64} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{256} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONVT}_{64} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{128} \rightarrow \text{LEAKYRELU}$	$\rightarrow \text{CONVT}_{128} \rightarrow \text{LEAKYRELU}$
	$\rightarrow \text{CONVT}_{1} \rightarrow \text{SIGMOID}$	$\rightarrow \text{CONVT}_{1} \rightarrow \text{SIGMOID}$	$\rightarrow \text{CONVT}_{1} \rightarrow \text{SIGMOID}$

Table B.1: Convolutional Encoder-Decoder architectures. CONV_n denotes a convolutional layer with n filters, whereas FC_n represents a fully-connected layer with n units. All convolutions CONV_n and transposed convolutions CONVT_n have a filter size of 4×4 for MNIST and FREY-YALE and 5×5 for CELEBA. $M = 1$ for all models except for the variational autoencoders (VAEs) which have $M = 2$ as the encoder has to yield both mean and variance for each input.

B.5.4 Prior optimiziation

As done by Nguyen et al. (2021), we use a single-layer multilayer perceptron (MLP), h_ϕ, to represent the Borel measurable function in the dual form of DSWD (Eq. B.7). At each iteration of Algorithm 5, to find a local maxima, we optimize h_ϕ for 30 epochs by using an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0005. We use another Adam optimizer with a learning rate of 0.001 to update the prior's parameters. We use a mini-batch size of $N_B = 64$ and then generate $N_s = 32$ prior samples given each data point. By default, we use $K = 1000$ random projections with a regularization coefficient $\lambda_C = 100$ to estimate the 2-Wasserstein distance. The convergences of prior optimization on MNIST, FREY and CELEBA datasets are illustrated in Fig. B.8.

B.5.5 SGHMC hyperparameters

In Table B.2 we report the hyperparameters used in the experiments on MNIST, YALE and CELEBA datasets. As seen, we always use a fixed step size of 0.003, a momentum coefficient of 0.05, and a mini-batch size of 64. The number of collected samples after thinning is 32. The number of burn-in iterations and the thinning interval are increased according to the size of the training set.

	MNIST				YALE				CELEBA			
Training Size	200	500	1000	2000	50	100	200	500	500	1000	2000	4000
Mini-batch Size	64	64	64	64	64	64	64	64	64	64	64	64
Step Size (10^{-3})	3	3	3	3	3	3	3	3	3	3	3	3
Momentum (10^{-2})	5	5	5	5	5	5	5	5	5	5	5	5
Num. Burn-in Steps (10^3)	6	6	6	6	6	6	6	6	6	20	20	20
Num. Samples	32	32	32	32	32	32	32	32	32	32	32	32
Thinning Interval (10^3)	1	1	1	2	1	1	1	1	1	2	3	5

Table B.2: stochastic gradient Hamiltonian Monte Carlo (SGHMC) hyperparameters used in the experiments on MNIST, YALE and CELEBA datasets.

B.5.6 Competing approaches

- **VAE** (Kingma and Welling, 2014): The vanilla VAE model employed with a Gaussian encoder and a standard Gaussian prior on the latent space.

- β-**VAE** (Higgins et al., 2017): The Kullback-Leibler (KL) term in the VAE's objective is weighted by $\beta = 0.1$ to reduce the effect of the prior. This helps to avoid the over-regularization problem of VAEs and improve reconstruction quality.

- **VAE + Sylvester Flows** (Berg et al., 2018): One of the state-of-the-art normalizing flows for the encoder of VAEs, which has richer expressiveness than VAE's post-Gaussian encoder. As employed in Berg et al. (2018), we use Orthogonal Sylvester flows with 4 transformations and 32 orthogonal vectors.

- **VAE + VampPrior** (Tomczak and Welling, 2018): A flexible prior for VAEs, which is a mixture of variational posteriors conditioned on learnable pseudo-observations. This allows the variational posterior to learn more a potential latent representation. Due to using small training data, we use 100 trainable pseudo-observations in our experiments. We found that increasing more pseudo-observations may hurt the predictive performance because of overfitting.

- **2-Stage VAE** (Dai and Wipf, 2019): A simple and practical method to improve the quality of generated images from VAEs by performing a form of ex-post density estimation via a second VAE. As employed in Dai and Wipf (2019), for the second-stage VAE, we use a MLP having three 1024-dimensional hidden layers with ReLU activation function.

- **WAE** (Tolstikhin et al., 2018) Wasserstein Autoencoder: This model is an alternative of VAEs. By reformulating the objective function as an optimal transport (OT) problem, Wasserstein autoencoder (WAE) regularizes the averaged encoding distribution instead of each data point. This encourages the encoded training distribution to match the prior while still allowing to learn significant

information from the data. As suggested in Tolstikhin et al. (2018), we use WAE-MMD with the inverse multiquadratics kernel and a regularization coefficient $\lambda = 10$ due to its stability compared to WAE-GAN. We impose the standard Gaussian prior on the latent space.

- **NS-GAN** (Goodfellow et al., 2014): a standard generative adversarial network (GAN) with the non-saturating loss, which has been shown to be robust to the choice of hyperparameters on CELEBA (Lucic et al., 2018). For a fair comparison, we reuse the encoder and decoder architectures for the discriminator and generator, respectively.

- **DiffAugment-GAN** (Zhao et al., 2020): a more complex architecture (STYLEGAN2, see Karras et al., 2020) combined with a powerful differentiable augmentation scheme, specifically developed for low data regimes. We refer to the original work of Zhao et al. (2020) and the implementation in `https://github.com/mit-han-lab/data-efficient-gans` for additional details on the network architecture. We use the same latent size of 50, a maximum of 64 feature maps, and all available augmentations (color, cutout and translation). The remaining parameters are left at default value.

All autoencoder models are trained for 200 epochs with an Adam optimizer (Kingma and Ba, 2015) using the default hyperparameters in PyTorch, i.e. learning rate $= 0.001$, $\beta_1 = 0.9, \beta_2 = 0.999$. The NS-GAN is trained for 200 epochs with a learning rate of 0.0002. The DiffAugment-GAN is trained with learning rate of 0.001 for 1 million steps (expect for the case of 4 000 training samples, which was extended for 2 millions steps).

B.5.7   Performance evaluation

Test log-likelihood.   To evaluate the reconstruction quality, we use the mean predictive log-likelihood evaluated over the test set. This metric tells us how probable it is that the test targets were generated using the test inputs and our model. Notice that for the case of autoencoder models, the test targets are exactly the test inputs. The predictive likelihood is a proper scoring rule (Gneiting and Raftery, 2007) that depends on both the accuracy of predictions and their uncertainty.

For BAE, as done in the literature of Bayesian neural networks (BNNs) (Izmailov et al., 2021b; Osawa et al., 2019b), we can estimate the predictive likelihood for an unseen data point, $\boldsymbol{y}^*$, as follows

$$\mathbb{E}_{p(\boldsymbol{w}\,|\,\boldsymbol{y})}[p(\boldsymbol{y}^*\,|\,\boldsymbol{w})] \approx \frac{1}{M}\sum_{i=1}^{M} p(\boldsymbol{y}^*\,|\,\boldsymbol{w}_i), \quad \boldsymbol{w}_i \sim p(\boldsymbol{w}\,|\,\boldsymbol{y}),$$

where $\boldsymbol{w}_i$ is a sample from the posterior $p(\boldsymbol{w}\,|\,\boldsymbol{y})$ obtained from the SGHMC sampler.

For VAEs, because the randomness comes from the latent code not the network's parameters, we can use MC approximation to estimate the predictive likelihood as follows

$$\mathbb{E}_{q(z\,|\,x^*)}[p(y^*\,|\,z)] \approx \frac{1}{N}\sum_{i=1}^{N} p(y^*\,|\,z_i), \quad z_i \sim q(z\,|\,x^*),$$

where $x^* \overset{\text{def}}{=} y^*$, and $q(z\,|\,x^*)$ is the amortized approximate posterior. In our experiments, we use $N = 200$.

For completeness, we also report the test marginal log-likelihood $p(y)$ of VAEs, which is estimated by the importance weighted sampling (IWAE) method (Burda et al., 2016). More specifically,

$$\text{IWAE} = \log\left(\frac{1}{K}\sum_{i=1}^{K} \frac{p(y^*, z_i)}{q(z_i\,|\,x^*)}\right), \quad z_i \sim q(z\,|\,x^*).$$

It can be shown that IWAE lower bounds $\log p(y^*)$ and can be arbitrarily close to the target as the number of samples K grows. We use $K = 1000$ in the experiments. The full results of test marginal log-likelihood are reported in Tables **??**, **??** and **??**.

FID score. To assess the quality of the generated images, we employed the widely used Fréchet Inception Distance (Heusel et al., 2017). This metric is the Fréchet distance between two multivariate Gaussians, the generated samples and real data samples are compared through their distribution statistics:

$$\text{FID} = \|\mu_{\text{real}} - \mu_{\text{gen}}\|^2 + \text{Tr}(\Sigma_{\text{real}} + \Sigma_{\text{gen}} - 2\sqrt{\Sigma_{\text{real}}\Sigma_{\text{gen}}}). \tag{B.18}$$

Two distribution samples are calculated from the 2048-dimensional activations of pool3 layer of Inception-v3 network [2]. In our experiments, the statistics of generated and real data are computed over 10000 generated images and test data, respectively.

B.6 Additional Results of Comparison with Temperature Scaling

In Bayesian deep learning, *temperature scaling* is a practical technique to improve predictive performance (Zhang et al., 2018b; Izmailov et al., 2019; Wenzel et al., 2020). There are two main approaches to tempering the posterior, namely (1) *partial tempering* and (2) *full tempering* (Aitchison, 2021; Zeno et al., 2021). In this section, we investigate rigorously the posteriors induced by the $\mathcal{N}(0, 1)$ prior and optimized prior under different tempering settings. We use the same setup of MNIST as in the main paper,

---

[2]We use the original TensorFlow implementation of FID score which is available at `https://github.com/bioinf-jku/TTUR`.

with 200 examples for inference. For the optimized prior, we use 100 training samples for learning prior. For the $\mathcal{N}(0,1)$ prior, we use the union of 200 training samples and the data used to optimized prior for training.

B.6.1 Partial tempering

The *partially tempered* posterior is defined as follows Izmailov et al. (2019) and Wilson and Izmailov (2020)

$$p_{\tau_{\text{partial}}}(\boldsymbol{w}\mid\boldsymbol{y}) \propto \underbrace{p(\boldsymbol{y}\mid\boldsymbol{w})^{1/\tau}}_{\text{likelihood}}\,\underbrace{p(\boldsymbol{w})}_{\text{prior}},$$

where $\tau > 0$ is a *temperature* value. This parameter controls how the prior and likelihood interact in the posterior. When $\tau = 1$ the true posterior is recovered, and as τ becomes large, the tempered posterior approaches the prior. In the case of small training data and using a misspecified prior such as $\mathcal{N}(0,1)$, we would use a small temperature value (e.g. $\tau < 1$) to *reduce the effect of the prior*. This corresponds to artificially sharpening the posterior by overcounting the data by a factor of τ.

Fig. B.1a shows the test log-likelihood (LL) on MNIST for BAE with $\mathcal{N}(0,1)$ prior and different temperature values. As expected, the predictive performance of the posterior obtained via low temperatures $\tau < 1$ is much better than those at high temperatures $\tau > 1$. However, cooling the posterior only shows slight improvement compared to the true posterior induced from the optimized prior. In addition, in case $\tau > 1$, where the influence of the posterior becomes stronger, the tempered posterior w.r.t. the optimized prior is significantly better than using the $\mathcal{N}(0,1)$ prior. This again shows clearly that $\mathcal{N}(0,1)$ is a poor prior for a deep BAE.

Fig. B.2a illustrates samples from priors and posteriors in a low-dimensional space. We also consider the posterior obtained from the *entire* training data and the $\mathcal{N}(0,1)$ prior as "oracle" posterior. In this case, the choice of the prior does not strongly affect the posterior as this is dominated by the likelihood. It can be seen that, for high-temperature values $\tau > 1$, the *warm posteriors* w.r.t. $\mathcal{N}(0,1)$ prior are stretched out as the prior effect is too strong. These posteriors are mismatched with the "oracle" posterior as further confirmed by very low test log-likelihood. Meanwhile, due to the good inductive bias from the optimized prior, the corresponding tempered posterior is still located in regions nearby the "oracle" posterior. For low temperature values $\tau < 1$, the *cold posteriors* are more concentrated by overcounting evidence. However, if we use a very small temperature (e.g. $\tau = 10^{-5}$), the resulting posterior overly concentrates around the maximum likelihood estimation (MLE), becoming too constrained by the training data.
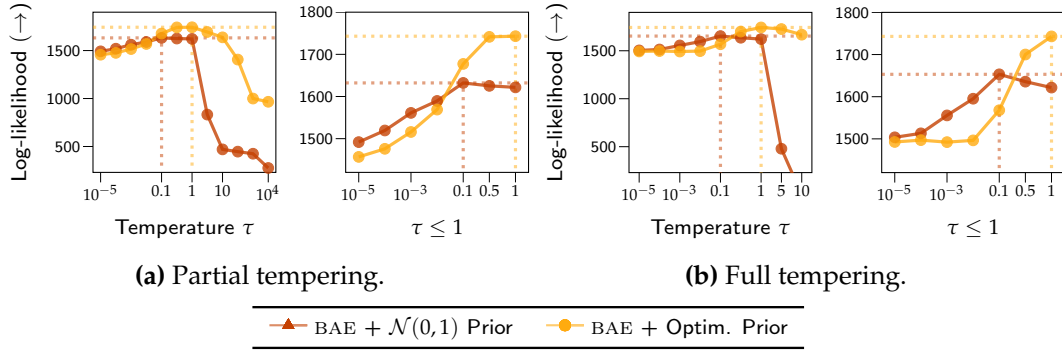
(a) Partial tempering.   **(b)** Full tempering.

BAE + $\mathcal{N}(0,1)$ Prior   BAE + Optim. Prior

Figure B.1: Test LL as a function of temperature on MNIST using BAE with $\mathcal{N}(0,1)$ prior. The dotted lines indicate the best performance of LL.
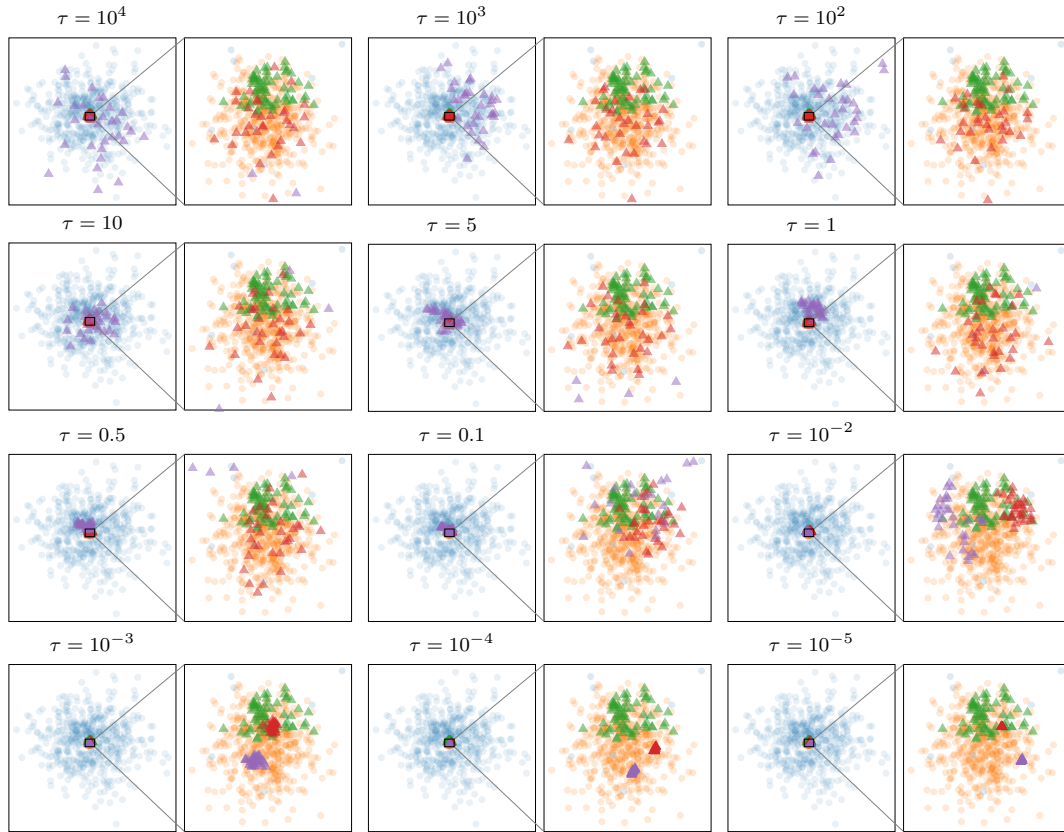
B.6.2 Full tempering

For the fully tempered posterior, instead of scaling the likelihood term only, we scale the whole posterior as follows

$$p_{\tau_{\text{full}}}(\boldsymbol{w} \mid \boldsymbol{y}) \propto \Big(\underbrace{p(\boldsymbol{y} \mid \boldsymbol{w})}_{\text{likelihood}} \underbrace{p(\boldsymbol{w})}_{\text{prior}} \Big)^{1/\tau}.$$

The only difference between partial and full tempering is whether we scale the prior. If we place Gaussian priors on the parameters, this scaling can be absorbed into the prior variance, $\sigma^2_{\text{full}} = \sigma^2_{\text{partial}} / \tau$.

Recently, Wenzel et al. (2020) argues that BNNs require a cold posterior, where a $\tau < 1$ is employed, to obtain a good performance. However, we hypothesize that *the cold posterior effect* may originate from using a poor prior. In this case, as shown in Fig. B.1b, the results of full tempering are similar to those of partial tempering. Cooling the posterior only helps to increase slightly predictive performance for $\mathcal{N}(0,1)$ prior. We also observe that the Markov chain Monte Carlo (MCMC) sampling is not converged if a very large τ is employed, thus we only consider small values of τ (e.g. $\tau \in \{5, 10\}$). In these cases, as depicted in Fig. B.2b, the samples from the posterior may be outside of the hypothesis space of the optimized prior.

In sum, the true posterior induced from our optimized prior is remarkably better than any types of tempered posteriors. These results suggest that, in the small-data regime, we should choose carefully a more sensible prior rather than simply using a vague prior and overcounting the data.

(a) Partial tempering.
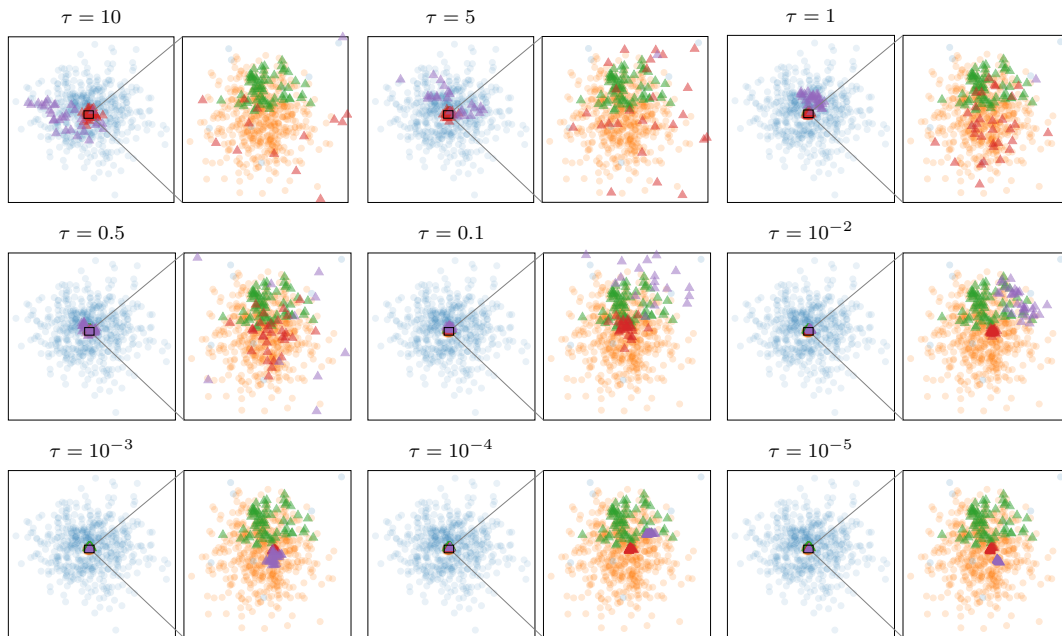
(b) Full tempering.

Optim. Prior   ▲ Posterior (Optim. Prior + ♦)
$\mathcal{N}(0,1)$ Prior   ▲ Posterior ($\mathcal{N}(0,1)$ Prior + ★)   ▲ Posterior ($\mathcal{N}(0,1)$ Prior + ■)

Figure B.2: Visualization of samples from priors and posteriors of BAE's parameters in the plane spanned by eigenvectors of the SGD trajectory. ♦ indicates using 200 samples for training; ★ indicates using the union of these samples and 100 samples used for learning the prior; ■ denotes using all 60000 training samples. Here, τ is the temperature value used for the ♦ and ★ cases. All plots are produced using convolutional BAE on MNIST.

B.7 Ablation Studies

B.7.1 Additional results of ablation study on the size of the dataset to optimize priors

In this experiment, we demonstrate that we can obtain a sensible result by using a small number of training instances to optimize the prior. Here, we use a set of 200 samples of 0-9 digits for inference, and another dataset also consisting of 0-9 digits for optimizing the prior. Fig. B.4 shows the predictive performance and samples from the posterior. We observe that the performance gain by using more data is not significant. We can achieve sensible results by using only about 10-50 samples for each class. In addition, as illustrated in the low-dimensional space (Fig. B.4), the hypothesis space of the prior is not collapsed as we increase the size of the dataset used to optimize the prior. As a result, the predictive posterior is also not concentrated to the MLE solutions as further demonstrated in Fig. B.3. This behavior is very different from overcounting the data by using temperature scaling, where the posterior becomes more concentrated as the temperature is decreased. This again demonstrates the practicality of our proposed method in the small-data regime.
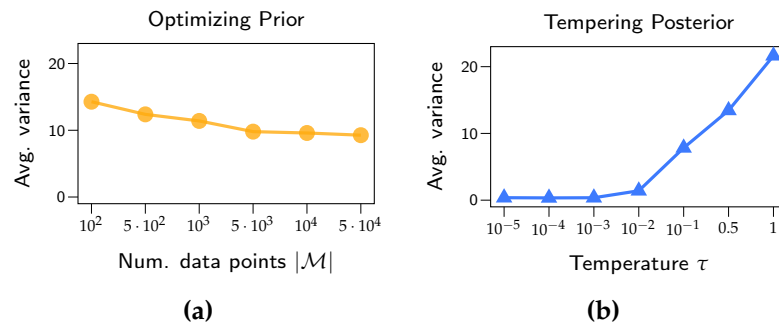
Figure B.3: The average predictive variance computed over test data points as a function of **(a)** the number of data points used to optimize prior, and **(b)** the temperature used for cooling the posterior. Here, we use 200 data points from MNIST dataset for inference. In figure **(a)**, we use the optimized prior and consider the true posterior without any tempering. In figure **(b)**, we use the standard Gaussian prior and employ partial tempering for the posterior.

B.7.2 Effect of the dimensionality of latent space

Fig. B.5 illustrates the predictive performance of VAEs and BAEs in terms test LL on MNIST for different size of the latent space and training size. It is clear that BAEs with optimized prior consistently outperforms other competitors across all dimensionalities of the latent space and training sizes.
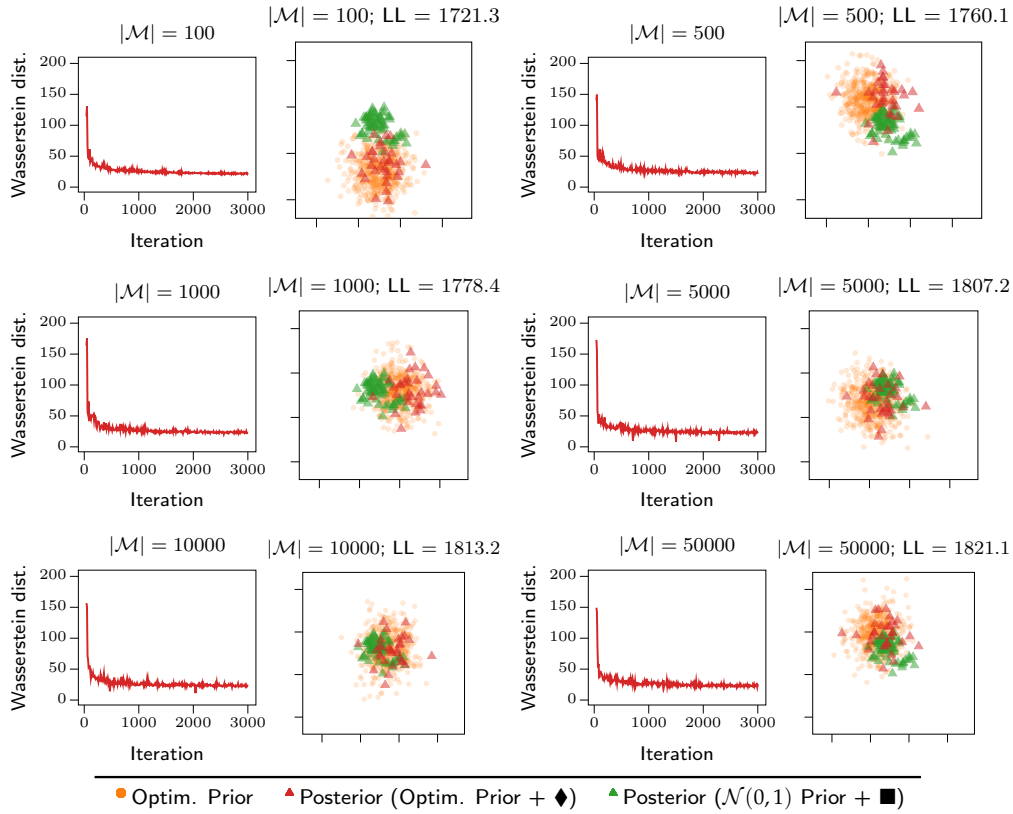
Figure B.4: Visualization of convergence Wasserstein optimization, and samples from priors and posteriors of BAE's parameters in the plane spanned by eigenvectors of the SGD trajectory corresponding to the first and second largest eigenvalues. Here, $|\mathcal{M}|$ is the size of dataset used for optimizing the prior; ◆ indicates using 200 training samples for inference; ∎ denotes using all 60000 training samples for inference; LL denotes the test log-likelihood performance of the posterior w.r.t. the optimized prior. All plots are produced using convolutional BAE on MNIST.
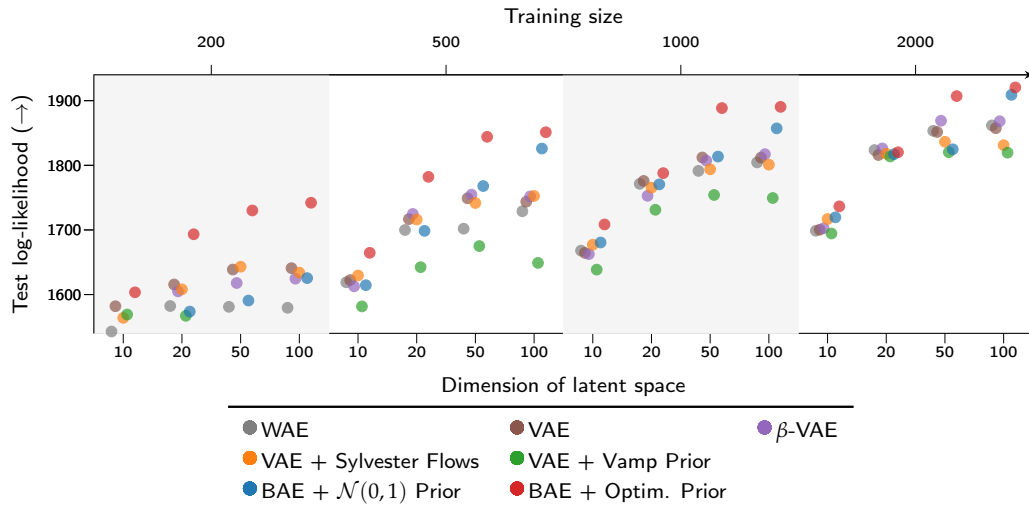
Figure B.5: Ablation study on the test log-likelihood on MNIST dataset for different sizes of the latent space and training sizes.

B.7.3 Visualizing 2-dimensional latent space

We run several experiments with a low latent space ($K = 2$) to test the efficacy of VAEs and BAEs as dimensionality reduction techniques. Fig. B.6 shows the results, where each color represents an MNIST digit. As seen, BAE with optimized prior produces a more well-defined class structure in comparision with other methods.

We also consider the 2D latent space to visualize that ex-post density estimation with Dirichlet process mixture model (DPMM) helps to reduce the mismatch between the aggregated posterior and the prior. As can be seen from Fig. B.7, there are large mismatches between aggregated posterior of VAEs and the $\mathcal{N}(0, 1)$ prior. We can reduce this problem by using a more expressive prior like VampPrior, or performing ex-post density estimation with a second VAE. For BAEs, it is clear that the flexible DPMM estimator effectively fixes the mismatch and this results in better sample quality as reported in the main paper.
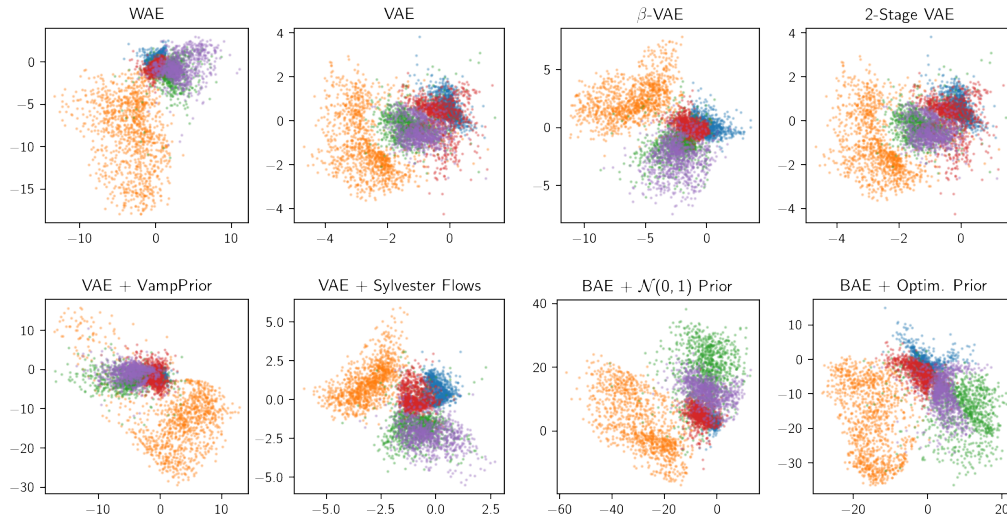
Figure B.6: Visualization of 2D latent spaces of variants of autoencoders on MNIST test set where each color represents a digit classs. We consider only 5 classes for easier visualization and comparison. All models are trained on 1000 training samples from MNIST dataset.
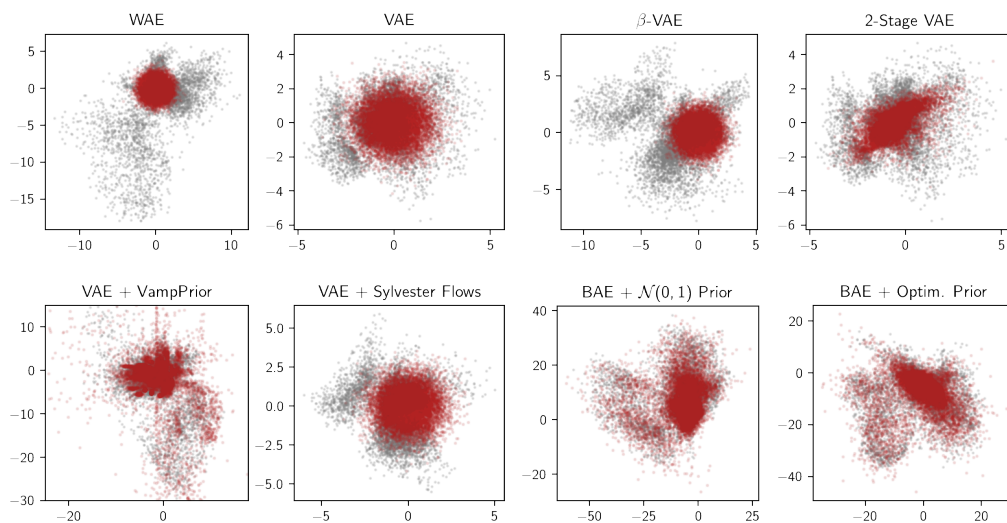
Figure B.7: Diffrent priors and density estimations on the 2-dimensional latent space of VAEs and BAEs. All models are trained on 1000 training samples from MNIST dataset. The gray points are test set samples while the red ones are samples from priors / density estimators. Here, we employ the isotropic Gaussian prior on the latent space of WAE, VAE, β-VAE and VAE with Sylveser Flows. The Vamp-Prior is learned to explicitly model the aggregated posterior while 2-Stage VAE uses another VAE to estimate the density of the learned latent space. Meanwhile, for BAEs, we use DPMMs for ex-post density estimation.

B.8 Additional Results

B.8.1 Convergence of Wasserstein optimization

Fig. B.8 depicts the progressions of Wasserstein optimization in the MNIST, FREY-YALE and CELEBA experiments.
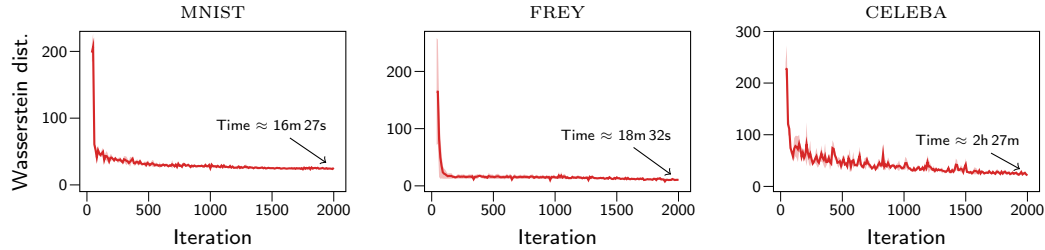
Figure B.8: Convergence of Wasserstein optimization. The shaded areas represent the standard deviation computed over 4 random data splits.

B.8.2 Tabulated results

Detailed results on MNIST, YALE and CELEBA datasets are reported from Table B.3 to Table B.9.

	LOG LIKELIHOOD ($\uparrow$)			
TRAINING SIZE	200	500	1000	2000
WAE	1590.0 (11.0)	1732.7 (19.2)	1809.5 (11.1)	1857.4 (4.8)
★ WAE	1675.2 (10.6)	1779.6 (10.2)	1839.3 (6.1)	1871.1 (3.1)
VAE	1635.1 (8.0)	1744.6 (4.5)	1805.5 (4.8)	1847.1 (3.7)
★ VAE	1697.0 (9.9)	1776.2 (6.8)	1829.5 (2.8)	1849.9 (4.4)
β-VAE	1626.2 (10.3)	1749.7 (9.2)	1812.8 (3.8)	1862.3 (4.9)
★ β-VAE	1698.2 (8.0)	1780.2 (9.3)	1841.2 (3.4)	1871.9 (4.4)
VAE + SYLVESER FLOWS	1635.4 (6.1)	1743.5 (1.5)	1799.1 (5.5)	1836.3 (7.2)
★ VAE + SYLVESER FLOWS	1711.4 (3.0)	1781.0 (2.9)	1816.7 (6.2)	1848.1 (6.5)
VAE + VAMPPRIOR	1543.0 (12.6)	1669.9 (22.0)	1756.8 (2.6)	1818.6 (3.6)
★ VAE + VAMPPRIOR	1609.6 (14.4)	1732.1 (14.2)	1798.1 (5.4)	1839.3 (4.0)
BAE + $\mathcal{N}(0,1)$ PRIOR	1609.0 (10.6)	1761.0 (9.1)	1837.6 (18.4)	1827.9 (5.7)
★ BAE + $\mathcal{N}(0,1)$ PRIOR	1681.2 (24.5)	1798.6 (22.8)	1827.0 (35.9)	1842.2 (37.4)
BAE + OPTIM. PRIOR (**OURS**)	**1743.5** (12.0)	**1845.1** (1.2)	**1879.1** (6.3)	**1906.8** (1.1)

Table B.3: Evaluation of all methods in terms of test log-likelihood (*the higher, the better*) on MNIST. The parentheses are the standard deviations. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model.

	Log Likelihood (↑)			
Training Size	50	100	200	500
WAE	689.7 (10.4)	724.8 (4.4)	754.5 (3.9)	787.0 (0.7)
★ WAE	718.4 (0.9)	740.6 (4.6)	765.7 (2.2)	794.3 (1.6)
VAE	692.3 (8.4)	723.5 (2.8)	738.4 (3.2)	774.1 (1.3)
★ VAE	701.2 (5.9)	728.2 (3.5)	749.4 (2.0)	774.8 (2.1)
β-VAE	707.1 (5.7)	733.8 (8.5)	761.1 (3.4)	791.8 (0.7)
★ β-VAE	712.1 (7.6)	737.8 (4.7)	763.4 (1.3)	790.8 (1.5)
VAE + Sylvester Flows	705.4 (4.8)	729.3 (4.4)	738.2 (1.6)	766.8 (0.9)
★ VAE + Sylvester Flows	682.1 (11.7)	716.3 (4.3)	739.6 (2.1)	765.3 (1.2)
VAE + VampPrior	690.0 (6.9)	722.8 (1.9)	740.6 (1.8)	766.8 (2.7)
★ VAE + VampPrior	691.7 (6.1)	716.9 (4.7)	737.8 (5.3)	764.2 (2.2)
BAE + $\mathcal{N}(0,1)$ Prior	426.1 (27.6)	668.8 (12.8)	724.9 (21.2)	775.5 (4.6)
★ BAE + $\mathcal{N}(0,1)$ Prior	388.0 (13.6)	570.4 (9.1)	688.2 (5.1)	752.5 (1.0)
BAE + Optim. Prior (**Ours**)	**730.3** (3.0)	**754.3** (3.1)	**771.6** (3.0)	**793.5** (2.0)

Table B.4: Evaluation of all methods in terms of test log-likelihood (*the higher, the better*) on YALE. The same interpretation as Table B.3.

	Log Likelihood (↑)			
Training Size	500	1000	2000	4000
WAE	5732.6 (35.3)	6266.4 (73.4)	6703.6 (24.9)	6928.3 (32.5)
★ WAE	6509.7 (49.2)	6659.8 (30.4)	6864.0 (23.7)	7021.6 (24.3)
VAE	5914.2 (78.3)	6406.4 (39.6)	6683.6 (87.5)	6976.4 (11.9)
★ VAE	6460.1 (33.7)	6694.1 (63.1)	6831.8 (97.2)	7039.5 (36.5)
β-VAE	5710.2 (49.0)	6192.5 (91.9)	6640.6 (139.4)	7000.9 (7.9)
★ β-VAE	6445.3 (94.0)	6654.6 (44.5)	6859.0 (39.8)	7007.7 (86.3)
VAE + Sylvester Flows	5481.6 (108.4)	5984.2 (37.4)	6415.5 (33.5)	6699.9 (46.9)
★ VAE + Sylvester Flows	6241.3 (149.2)	6437.2 (58.2)	6519.9 (88.5)	6831.5 (121.2)
VAE + VampPrior	5776.6 (95.9)	6242.2 (92.2)	6691.5 (24.4)	6999.7 (15.9)
★ VAE + VampPrior	6531.7 (61.5)	6591.6 (97.4)	6868.3 (27.8)	6990.7 (37.3)
2-Stage VAE	5914.2 (78.3)	6406.4 (39.6)	6683.6 (87.5)	6976.4 (11.9)
★ 2-Stage VAE	6460.1 (33.7)	6694.1 (63.1)	6831.8 (97.2)	7039.5 (36.5)
BAE + $\mathcal{N}(0,1)$ Prior	5581.9 (70.8)	6273.3 (54.2)	6848.3 (15.1)	7154.5 (15.6)
★ BAE + $\mathcal{N}(0,1)$ Prior	6574.1 (46.6)	6826.5 (31.0)	7038.3 (17.8)	7223.1 (13.2)
BAE + Optim. Prior (**Ours**)	**6781.3** (32.4)	**7065.8** (15.0)	**7244.7** (8.7)	**7370.0** (13.2)

Table B.5: Evaluation of all methods in terms of test log-likelihood (*the higher, the better*) on CELEBA. The same interpretation as Table B.3.

TRAINING SIZE	FID (↓)			
	500	1000	2000	4000
WAE	342.14 (19.02)	309.79 (12.58)	275.10 (8.71)	253.06 (5.52)
★ WAE	294.26 (8.41)	276.24 (10.49)	261.64 (6.08)	246.92 (3.28)
VAE	271.70 (5.12)	240.69 (3.44)	230.61 (7.05)	209.08 (6.28)
★ VAE	248.18 (12.20)	237.29 (12.48)	231.50 (14.17)	206.92 (9.91)
β-VAE	323.00 (10.88)	295.54 (12.45)	276.71 (15.61)	250.61 (5.30)
★ β-VAE	285.81 (5.58)	277.44 (12.97)	271.82 (6.69)	262.72 (17.92)
VAE + SYLVESTER FLOWS	221.71 (10.50)	214.94 (12.01)	207.86 (9.93)	198.94 (10.10)
★ VAE + SYLVESTER FLOWS	210.24 (3.48)	215.00 (5.79)	204.42 (11.86)	179.26 (49.53)
VAE + VAMPPRIOR	144.41 (16.61)	131.02 (2.22)	112.82 (4.05)	96.20 (2.79)
★ VAE + VAMPPRIOR	120.02 (8.62)	120.23 (7.16)	102.67 (7.61)	95.95 (4.86)
2-STAGE VAE	78.23 (2.56)	69.37 (2.39)	67.69 (1.55)	74.47 (4.52)
★ 2-STAGE VAE	72.21 (3.05)	69.25 (3.32)	72.64 (4.62)	84.95 (3.91)
NS-GAN	252.33 (27.03)	171.18 (15.51)	205.05 (97.46)	128.29 (3.81)
★ NS-GAN	151.28 (2.27)	150.74 (4.39)	137.64 (4.14)	139.43 (8.77)
★ DIFFAUGMENT-GAN	**66.09** (0.27)	**58.76** (0.17)	**50.22** (2.62)	**45.14** (0.13)
BAE + $\mathcal{N}(0,1)$ PRIOR	89.36 (4.56)	81.31 (2.50)	72.50 (1.37)	71.85 (0.17)
★ BAE + $\mathcal{N}(0,1)$ PRIOR	86.03 (3.53)	75.86 (0.45)	71.21 (1.41)	70.72 (0.39)
BAE + OPTIM. PRIOR (**OURS**)	68.59 (3.08)	66.11 (0.96)	68.34 (0.86)	67.18 (0.80)

Table B.6: Evaluation of all methods in terms of FID (*the lower, the better*) on CELEBA. The same interpretation as Table B.3.

TRAINING SIZE	LOG MARGINAL LIKELIHOOD (↑)			
	200	500	1000	2000
VAE	1648.2 (10.1)	1744.0 (5.6)	1795.1 (2.6)	1829.7 (2.5)
★ VAE	1702.4 (8.9)	1771.0 (6.7)	1816.2 (4.6)	1832.2 (4.8)
β-VAE	1497.1 (12.5)	1625.9 (7.7)	1687.3 (3.8)	1734.4 (4.2)
★ β-VAE	1570.1 (7.8)	1655.7 (7.9)	1715.2 (2.9)	1747.1 (5.6)
VAE + SYLVESTER FLOWS	1627.0 (6.9)	1709.8 (1.8)	1755.4 (4.6)	1783.3 (5.5)
★ VAE + SYLVESTER FLOWS	1688.0 (3.2)	1741.8 (2.5)	1771.4 (3.2)	1794.8 (5.0)
VAE + VAMPPRIOR	1545.6 (10.5)	1681.7 (20.2)	1758.4 (4.1)	1810.7 (2.2)
★ VAE + VAMPPRIOR	1616.3 (15.3)	1737.5 (11.4)	1795.6 (4.8)	1829.1 (2.6)

Table B.7: Evaluation of all methods in terms of test log marginal likelihood of VAE models (*the higher, the better*) on MNIST. The same interpretation as Table B.3.

	Log Marginal Likelihood (↑)			
Training Size	50	100	200	500
VAE	693.8 (7.8)	720.8 (3.4)	734.5 (2.8)	767.2 (0.6)
★ VAE	704.2 (5.6)	723.7 (3.1)	742.4 (1.9)	765.1 (1.2)
β-VAE	628.1 (2.7)	655.2 (9.9)	683.0 (3.9)	712.5 (1.6)
★ β-VAE	658.5 (13.4)	683.9 (5.4)	707.2 (2.7)	731.5 (2.3)
VAE + Sylvester Flows	668.6 (5.2)	686.5 (3.4)	695.1 (1.5)	718.0 (0.8)
★ VAE + Sylvester Flows	655.6 (4.9)	677.2 (3.8)	695.7 (0.7)	717.2 (0.7)
VAE + VampPrior	672.7 (7.9)	697.4 (6.8)	733.4 (3.2)	759.0 (1.2)
★ VAE + VampPrior	703.9 (4.2)	721.5 (4.0)	736.8 (4.1)	760.0 (2.2)

Table B.8: Evaluation of all methods in terms of test log marginal likelihood (*the higher, the better*) of VAE models on YALE. The same interpretation as Table B.3.

	Log Marginal Likelihood (↑)			
Training Size	500	1000	2000	4000
VAE	5973.4 (66.7)	6416.6 (36.6)	6673.7 (82.6)	6943.4 (8.7)
★ VAE	6470.0 (30.7)	6676.7 (57.4)	6807.6 (89.3)	7001.2 (37.5)
β-VAE	5496.0 (52.9)	6007.8 (89.8)	6457.8 (147.2)	6820.1 (7.4)
★ β-VAE	6294.4 (98.1)	6472.2 (46.1)	6680.5 (42.8)	6844.6 (93.9)
VAE + Sylvester Flows	5545.8 (97.8)	5988.2 (40.9)	6387.9 (37.8)	6649.9 (47.1)
★ VAE + Sylvester Flows	6226.9 (140.3)	6406.2 (53.7)	6485.3 (85.4)	6787.9 (126.9)
VAE + VampPrior	5842.2 (82.8)	6273.8 (86.3)	6682.6 (16.8)	6984.6 (7.4)
★ VAE + VampPrior	6538.3 (62.4)	6595.9 (93.8)	6852.3 (18.6)	6966.1 (28.0)

Table B.9: Evaluation of all methods in terms of test log marginal likelihood (*the higher, the better*) of VAE models on CELEBA. The same interpretation as Table B.3.

		Log Likelihood ($\uparrow$)		
Training Size	500	1000	2000	4000
WAE	7418.8 (123.3)	8342.0 (73.8)	8840.3 (26.7)	9230.2 (0.0)
★ WAE	8644.3 (72.7)	8889.5 (58.5)	9033.6 (97.5)	9257.9 (68.2)
VAE	7575.3 (60.1)	8343.9 (42.3)	8817.6 (124.7)	9251.8 (23.9)
★ VAE	8608.2 (15.8)	8855.1 (65.7)	9079.7 (50.0)	9276.7 (12.2)
β-VAE	7632.0 (115.3)	8220.7 (161.7)	8910.7 (33.9)	9305.2 (14.2)
★ β-VAE	8647.3 (30.1)	8768.6 (60.4)	9132.0 (22.1)	9290.8 (87.5)
VAE + Sylvester Flows	6976.7 (162.5)	7898.8 (106.1)	8430.5 (58.6)	8939.1 (53.5)
★ VAE + Sylvester Flows	8240.8 (45.2)	8446.7 (40.6)	8700.4 (65.6)	9045.1 (48.8)
VAE + VampPrior	7447.5 (77.9)	8251.2 (39.5)	8775.4 (40.9)	9261.8 (14.0)
★ VAE + VampPrior	8466.2 (50.7)	8814.3 (61.2)	9069.2 (41.1)	9355.3 (9.5)
2-Stage VAE	7575.3 (60.1)	8343.9 (42.3)	8817.6 (124.7)	9251.8 (23.9)
★ 2-Stage VAE	8608.2 (15.8)	8855.1 (65.7)	9079.7 (50.0)	9276.7 (12.2)
BAE + $\mathcal{N}(0,1)$ Prior	7097.4 (75.0)	8299.5 (7.6)	9009.8 (11.4)	9326.9 (8.4)
★ BAE + $\mathcal{N}(0,1)$ Prior	8562.8 (43.8)	8770.5 (158.6)	9219.2 (9.2)	9380.2 (97.2)
BAE + Optim. Prior (**Ours**)	**8975.1 (32.4)**	**9244.3 (15.2)**	**9424.8 (7.2)**	**9629.5 (3.9)**

Table B.10: Evaluation of all methods in terms of test log-likelihood (*the higher, the better*) on CELEBA. Here, all models are employ with the *truncated Gaussian likelihood*. The same interpretation as Table B.5.

		FID ($\downarrow$)		
Training Size	500	1000	2000	4000
WAE	328.85 (17.16)	296.43 (7.93)	291.25 (12.45)	247.82 (0.00)
★ WAE	287.16 (22.87)	273.54 (6.44)	279.81 (17.56)	255.96 (13.60)
VAE	299.73 (5.21)	271.97 (5.50)	248.95 (13.13)	235.11 (6.48)
★ VAE	255.93 (12.86)	256.20 (4.28)	242.79 (9.26)	231.78 (7.74)
β-VAE	334.00 (9.85)	322.93 (14.36)	295.70 (7.74)	283.72 (4.58)
★ β-VAE	307.30 (9.06)	301.14 (8.67)	286.14 (7.51)	276.94 (9.99)
VAE + Sylvester Flows	238.95 (16.95)	239.26 (19.51)	229.78 (8.82)	217.97 (9.79)
★ VAE + Sylvester Flows	231.82 (9.54)	243.18 (2.56)	221.53 (5.51)	206.25 (6.18)
VAE + VampPrior	127.05 (6.18)	126.32 (4.19)	105.52 (3.60)	97.56 (1.08)
★ VAE + VampPrior	110.61 (1.29)	113.03 (1.67)	101.26 (3.58)	88.87 (1.50)
2-Stage VAE	97.77 (1.01)	92.52 (2.81)	95.63 (3.19)	101.73 (5.24)
★ 2-Stage VAE	90.01 (11.92)	95.29 (6.39)	100.32 (2.41)	105.47 (3.99)
BAE + $\mathcal{N}(0,1)$ Prior	84.11 (4.09)	72.54 (2.21)	67.87 (0.61)	67.00 (0.44)
★ BAE + $\mathcal{N}(0,1)$ Prior	78.06 (1.42)	78.13 (6.90)	66.55 (0.87)	70.47 (6.95)
BAE + Optim. Prior (**Ours**)	**62.75 (3.61)**	**62.42 (1.20)**	**62.17 (0.89)**	**58.84 (1.26)**

Table B.11: Evaluation of all methods in terms of FID (*the lower, the better*) on CELEBA. Here, all models are employed with the *truncated Gaussian likelihood*. The same interpretation as Table B.6.

B.8.3 More qualitative results

Figure B.9: Qualitative evaluation for sample quality for autoencoders and GANs on CELEBA. Here, we use 500 samples for training/inference.

(a) WAE

(b) VAE

(c) β-VAE

(d) VAE + Sylvester Flows

(e) VAE + VampPrior

(f) 2Stage-VAE

(g) NS-GAN

(h) StyleGAN2

(i) BAE + $\mathcal{N}(0,1)$ Prior

(j) BAE + Optim. Prior (**Ours**)

Figure B.10: Qualitative evaluation for sample quality for autoencoders with the *truncated Gaussian likelihood* on CELEBA. Here, we use 500 samples for training/inference.

(a) WAE

(b) VAE

(c) β-VAE

(d) VAE + Sylvester Flows

(e) VAE + VampPrior

(f) 2Stage-VAE

(g) BAE + $\mathcal{N}(0, 1)$. Prior

(h) BAE + Optim. Prior

CELEBA - RECONSTRUCTIONS

GROUND TRUTH	
★ WAE	
★ VAE	
★ β-VAE	
★ VAE + SYLVESTER FLOWS	
★ VAE + VAMPPRIOR	
★ 2-STAGE VAE	
★ BAE + $\mathcal{N}(0,1)$ PRIOR	
BAE + OPTIM. PRIOR (**OURS**)	

Table B.12: Qualitative evaluation for reconstructed samples on CELEBA. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 1000.

CELEBA - RECONSTRUCTIONS

GROUND TRUTH	
★ WAE	
★ VAE	
★ β-VAE	
★ VAE + SYLVESTER FLOWS	
★ VAE + VAMPPRIOR	
★ 2-STAGE VAE	
★ BAE + $\mathcal{N}(0,1)$ PRIOR	
BAE + OPTIM. PRIOR (**OURS**)	

Table B.13: Qualitative evaluation for reconstructed samples on CELEBA with the *truncated Gaussian likelihood*. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 1000.

MNIST - RECONSTRUCTIONS

GROUND TRUTH	
WAE	
★ WAE	
VAE	
★ VAE	
β-VAE	
★ β-VAE	
VAE + SYLVESTER FLOWS	
★ VAE + SYLVESTER FLOWS	
VAE + VAMPPRIOR	
★ VAE + VAMPPRIOR	
2-STAGE VAE	
★ 2-STAGE VAE	
BAE + $\mathcal{N}(0,1)$ PRIOR	
★ BAE + $\mathcal{N}(0,1)$ PRIOR	
BAE + OPTIM. PRIOR (**OURS**)	

Table B.14: Qualitative evaluation for reconstructed samples on MNIST. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 200.
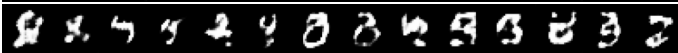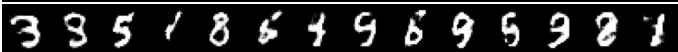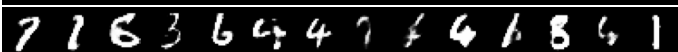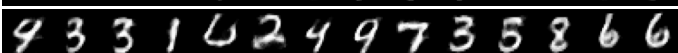
MNIST - GENERATED SAMPLES
WAE
★ WAE
VAE
★ VAE
β-VAE
★ β-VAE
VAE + SYLVESTER FLOWS
★ VAE + SYLVESTER FLOWS
VAE + VAMPPRIOR
★ VAE + VAMPPRIOR
2-STAGE VAE
★ 2-STAGE VAE
BAE + $\mathcal{N}(0,1)$ PRIOR
★ BAE + $\mathcal{N}(0,1)$ PRIOR
BAE + OPTIM. PRIOR (**OURS**)

Table B.15: Qualitative evaluation for generated samples on MNIST. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 200.
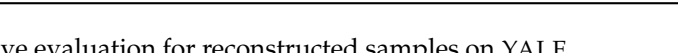
YALE - RECONSTRUCTIONS
GROUND TRUTH
WAE
★ WAE
VAE
★ VAE
β-VAE
★ β-VAE
VAE + SYLVESTER FLOWS
★ VAE + SYLVESTER FLOWS
VAE + VAMPPRIOR
★ VAE + VAMPPRIOR
2-STAGE VAE
★ 2-STAGE VAE
BAE + $\mathcal{N}(0,1)$ PRIOR
★ BAE + $\mathcal{N}(0,1)$ PRIOR
BAE + OPTIM. PRIOR (**OURS**)

Table B.16: Qualitative evaluation for reconstructed samples on YALE. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 500.

YALE - GENERATED SAMPLES
WAE
★ WAE
VAE
★ VAE
β-VAE
★ β-VAE
VAE + SYLVESTER FLOWS
★ VAE + SYLVESTER FLOWS
VAE + VAMPPRIOR
★ VAE + VAMPPRIOR
2-STAGE VAE
★ 2-STAGE VAE
BAE + $\mathcal{N}(0,1)$ PRIOR
★ BAE + $\mathcal{N}(0,1)$ PRIOR
BAE + OPTIM. PRIOR (**OURS**)

Table B.17: Qualitative evaluation for generated samples on YALE. ★ indicates that we use the union of the training data and the data used to optimize prior to train the model. Here, the training size is 500.

Appendix C

Appendix for Chapter 5

C.1   A Taxonomy of Latent Variable Models

By considering the characteristics of the prior distribution on latent variables, the likelihood function, input dependencies, and Bayesian treatments, we can draw connections between our proposed models and other latent variable models. Fig. C.1 summarizes these relationships. Here, we assume an isotropic Gaussian likelihood with precision β for the high-dimensional observed data y_n as used in our experiments.

Probabilistic principal component analysis (PCA) (Bishop, 2006) is a simple latent variable model that imposes an isotropic Gaussian prior over the latent space and linear mapping from the latent variables to the observed data. variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) build upon this by introducing a nonlinear parametric mapping to the observed data, while Gaussian process latent variable models (GPLVMs) utilize a nonparametric Gaussian process (GP) mapping. Recently, Lalchand et al. (2022a) extended GPLVMs in a fully Bayesian manner using stochastic variational inference (VI). Conditional Variational Autoencoder (CVAE) (Sohn et al., 2015) is an extension of VAEs that utilizes an input-dependent prior in the latent space for conditional generation tasks. However, this model does not account for correlations between latent representations. Gaussian Process VAEs (Casale et al., 2018; Pearce, 2019; Jazbec et al., 2021) overcome this problem by imposing GP priors on the latent space. Our model, Sparse Gaussian Process Bayesian Autoencoder (SGP-BAE), further enhances the modeling capabilities of these models by using a fully Bayesian approach for the latent variables, decoder, and GP hyperparameters in a fully Bayesian manner, and decoupling the model from the variational inference formulation.

Latent neural processes (Garnelo et al., 2018) can be seen as an extension of CVAE. However, this model follows a meta-learning approach and splits the data into a context set, $\left\{ x_n^{[C]}, y_n^{[C]} \right\}_{n=1}^{N_C}$, and a target set, $\left\{ x_n^{[T]}, y_n^{[T]} \right\}_{n=1}^{N_T}$. This model uses a global latent variable z to capture the global properties of the context data. The likelihood is conditioned on both new target input $x_n^{[T]}$ and the global latent variable z.
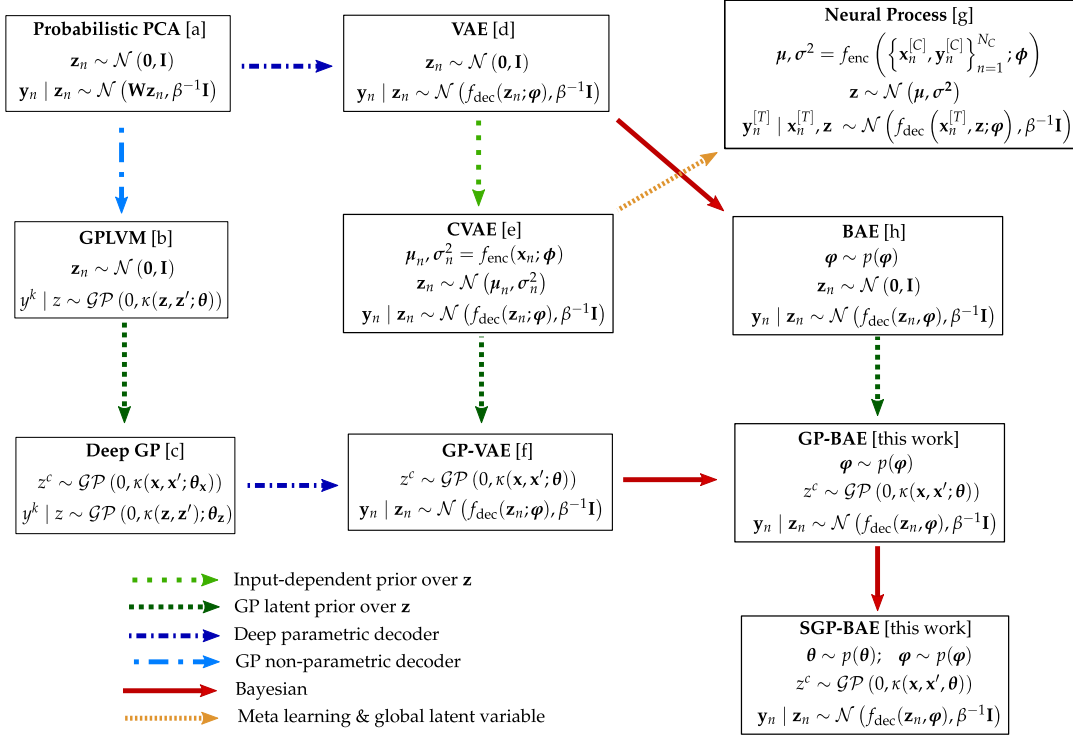
Figure C.1: Connections between our proposed models and other latent variables models. References are [a] for Bishop (2006), [b] for Lawrence (2005), [c] for Damianou and Lawrence (2013), [d] for Kingma and Welling (2014) and Rezende et al. (2014), [e] for Sohn et al. (2015), [f] for Casale et al. (2018), Pearce (2019), and Jazbec et al. (2021), [g] for Garnelo et al. (2018), and [h] for this work and Kingma and Welling (2014) and Daxberger and Hernández-Lobato (2019).

C.2 Details of the Scalable Sampling Objective for Sparse Gaussian Processes

GPs (Rasmussen and Williams, 2006) are one of the main workhorses of Bayesian non-parametric statistics and machine learning, as they provide a flexible and powerful tool for imposing a prior distribution over functions:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})), \tag{C.1}$$

where $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ maps D-dimensional inputs into one-dimensional outputs. GPs are fully characterized by their mean and their covariance:

$$\mathbb{E}[f(\boldsymbol{x})] = m(\boldsymbol{x}), \qquad \text{cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}), \tag{C.2}$$

where $m : \mathbb{R}^D \to \mathbb{R}$ is the mean and $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel function with hyperparameters $\boldsymbol{\theta}$. GPs indeed can be viewed as an extension of a multivariate Gaussian distribution to infinitely many dimensions. For any fixed set of inputs $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, the realizations of functions with a GP prior at these inputs, denoted by

$f \in \mathbb{R}^N$, follow the Gaussian distribution:

$$p(f) = \mathcal{N}(\mathbf{0}, K_{xx \mid \theta}). \tag{C.3}$$

Here, we assume a zero-mean GP prior and $K_{XX \mid \theta}$ is the covariance matrix obtained by evaluating $\kappa(x, x'; \theta)$ over all input pairs x_i, x_j (we will drop the explicit parameterization on θ for the sake of notation). From now on, in order to keep the notation uncluttered, we focus on a single channel of latent space of autoencoders (AEs). We assume that the latent codes Z are stochastic realizations based on f and additive Gaussian noise i.e. $Z \mid f \sim \mathcal{N}(f, \sigma^2 I)$. We further assume a full factorization $p(Z \mid f; \sigma^2 I) = \prod_{n=1}^{N} p(z_n \mid f_n; \sigma^2)$.

Though GPs provide an elegant mechanism to handle uncertainties, their computational complexity grows cubically with the number of inputs, i.e. $\mathcal{O}(N^3)$. This problem is commonly tackled by sparse GPs, which are a family of approximate models that address the scalability issue by introducing a set of M inducing variables $u = (u_1, ..., u_M)^\top \in \mathbb{R}^{M \times 1}$ at the corresponding inducing inputs $S = [s_1^\top, ..., s_M^\top]^\top \in \mathbb{R}^{M \times D}$ with $u_m \equiv f(s_m)$. The inducing points S can be interpreted as a compressed version of the training data where the number of inducing points M acts as a trade-off parameter between the goodness of the approximation and scalability. The inducing variables are assumed to be drawn from the same GP as the original process, i.e.:

$$p(f, u) = p(u)p(f \mid u), \text{with} \tag{C.4}$$

$$p(u) = \mathcal{N}(\mathbf{0}, K_{ss}), \tag{C.5}$$

$$p(f \mid u) = \mathcal{N}(K_{xs}K_{ss}^{-1}u, K_{xx} - K_{xs}K_{ss}^{-1}K_{sx}), \tag{C.6}$$

where the covariance matrices K_{ss}, K_{xs} are computed between the elements in S and $\{X, S\}$, respectively, and $K_{sx} = K_{xs}^\top$.

There is a line of works using variational techniques for sparse GPs priors for VAEs (Jazbec et al., 2021; Ashman et al., 2020). More specifically, they rely on the variational framework proposed by Titsias (2009) and Hensman et al. (2013), enabling the use of GP priors on very large datasets. However, the posterior of the inducing variables u under this framework involves constraining to have a parametric form (usually a Gaussian).

In this work, we take a different route as we treat the inducing variables u, inducing inputs S as well as the kernel hyperparameters θ in a fully Bayesian way. As discussed in the main paper, we wish to infer these variables together with the decoder parameters and the latent codes by using a powerful and scalable stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) sampler. To do so, the sampling objective Eq. 5.8 should be decomposed over all data points. The main obstacle is the joint distribution $p(Z, u \mid \theta) = \mathbb{E}_{p(f \mid u)}[p(Z \mid f; \sigma^2 I)]$, which has a complicated form due to the expectation under the conditional $p(f \mid u)$. As discussed by Rossi et al.

(2021), this problem can be solved effectively by the fully independent training conditionals (FITC; see Quiñonero-Candela and Rasmussen, 2005), i.e., parameterizing $p(f \mid u)$ as follows:

$$p(f \mid u) \approx \mathcal{N}(f; K_{xs}K_{ss}^{-1}u, \mathrm{diag}[K_{xx} - K_{xs}K_{ss}^{-1}K_{sx}]) \tag{C.7}$$

$$= \prod_{n=1}^{N} p(f_n \mid u) = \prod_{n=1}^{N} \mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2), \tag{C.8}$$

where

$$\tilde{\mu}_n = \kappa(x_n, S)K_{ss}^{-1}u, \tag{C.9}$$

$$\tilde{\sigma}_n^2 = \kappa(x_n, x_n) - \kappa(x_n, S)K_{ss}^{-1}\kappa(S, x_n). \tag{C.10}$$

We then can decompose the log-joint distribution over all data points as follows:

$$\log p(Z, u \mid \theta) = \log \mathbb{E}_{p(f \mid u, \theta)}[p(z \mid f; \sigma^2 I)] \tag{C.11}$$

$$= \log \int p(f \mid u, \theta)p(Z \mid f, \sigma^2 I)df \tag{C.12}$$

$$= \log \int \prod_{n=1}^{N} p(z_n \mid f_n; \sigma^2)p(f_n \mid u, \theta)df_1...df_n \tag{C.13}$$

$$= \log \prod_{n=1}^{N} \int \mathcal{N}(z_n; f_n, \sigma^2)\mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2)df_n \tag{C.14}$$

$$= \log \prod_{n=1}^{N} \mathcal{N}(z_n; \tilde{\mu}_n, \tilde{\sigma}_n^2 + \sigma^2) \tag{C.15}$$

$$= \sum_{n=1}^{N} \log \mathcal{N}(z_n; \tilde{\mu}_n, \tilde{\sigma}_n^2 + \sigma^2), \tag{C.16}$$

where $\tilde{\mu}_n, \tilde{\sigma}_n^2$ are given by Eq. C.9 and Eq. C.10, respectively.

In this work, we adopt the approach proposed by Hensman et al. (2015a), which involves sampling the hyperparameters θ and u jointly. To achieve sampling efficiency, a whitening representation is utilized, where the inducing variables are reparameterized as $u = L_{ss}v$, with $K_{ss} = L_{ss}L_{ss}^{\top}$. Subsequently, the sampling process involves obtaining samples from the joint posterior distribution over v and θ.

C.3   Details of the Extension to deep Gaussian Processes

We assume a deep Gaussian process prior $f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$ (Damianou and Lawrence, 2013), where each $f^{(l)}$ is a GP. For the sake of notation, we use $\mathbf{\Psi}^{(l)}$ to indicate the set of kernel hyperparameters and inducing inputs of the l-th layer and $f^{(0)}$ as the input X. We additionally denote $\mathbf{\Psi} = \{\boldsymbol{\varphi}\} \cup \left\{u^{(l)}, \mathbf{\Psi}^{(l)}\right\}_{l=1}^{L}$, where $\boldsymbol{\varphi}$ is the decoder's parameters.

The full joint distribution is as follows:

$$p\left(\mathbf{\Psi}, \{f^{(l)}\}_{l=1}^L, Z, Y \,\middle|\, X\right) = p(\mathbf{\Psi}) \underbrace{\prod_{l=1}^L p\left(f^{(l)} \mid f^{(l-1)}, \mathbf{\Psi}\right) p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right)}_{\text{Deep GP prior}} p(Y \mid Z, \boldsymbol{\varphi}).$$

(C.17)

Here we omit the dependency on X for notational brevity.

We wish to infer the posterior over $\mathbf{\Psi}$ and the latent variables Z. To do this, the hidden layers $f^{(l)}$ have to be marginalized and propagated up to the final layer L (Salimbeni and Deisenroth, 2017). In particular, the log posterior is as follows:

$$\log p(\mathbf{\Psi}, Z \mid Y, X) = \log p(\mathbf{\Psi}) + \log \mathbb{E}_{p\left(\{f^{(l)}\}_{l=1}^L \mid \{u^{(l)}, \mathbf{\Psi}^{(l)}\}_{l=1}^L\right)} p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right) +$$
$$+ \log p(Y \mid Z, \boldsymbol{\varphi}) - \log C,$$

(C.18)

where C is a normalizing constant.

The above posterior is intractable, but we have obtained the form of its (un-normalized) log-joint, from which we can sample using the Hamiltonian Monte Carlo (HMC) method. Unfortunately, the expectation term is intractable, but we can estimate it using Monte Carlo sampling as follows:

$$\log \mathbb{E}_{p\left(\{f^{(l)}\}_{l=1}^L \mid \{u^{(l)}, \mathbf{\Psi}^{(l)}\}_{l=1}^L\right)} p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right) \approx$$
$$\approx \log \mathbb{E}_{p\left(\{f^{(l)}\}_{l=2}^L \mid \widetilde{f}^{(1)}, \{u^{(l)}, \mathbf{\Psi}^{(l)}\}_{l=2}^L\right)} p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right), \qquad \widetilde{f}^{(1)} \sim p\left(f^{(1)} \,\middle|\, u^{(1)}, \mathbf{\Psi}^{(1)}, f^{(0)}\right),$$
$$\approx \log \mathbb{E}_{p\left(\{f^{(l)}\}_{l=3}^L \mid \widetilde{f}^{(2)}, \{u^{(l)}, \mathbf{\Psi}^{(l)}\}_{l=3}^L\right)} p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right), \quad \widetilde{f}^{(2)} \sim p\left(f^{(2)} \,\middle|\, u^{(2)}, \mathbf{\Psi}^{(2)}, \widetilde{f}^{(1)}\right),$$
$$\approx \ldots$$
$$\approx \log \mathbb{E}_{p\left(f^{(L)} \mid \widetilde{f}^{(L-1)}, u^{(L)}, \mathbf{\Psi}^{(L)}\right)} p\left(Z \,\middle|\, f^{(L)}; \sigma^2 I\right), \quad \widetilde{f}^{(L-1)} \sim p\left(f^{L-1} \,\middle|\, u^{(L-1)}, \mathbf{\Psi}^{(L-1)}, \widetilde{f}^{(L-2)}\right),$$
$$= \sum_{n=1}^N \log \mathbb{E}_{p\left(f_n^L \mid \widetilde{f}_n^{L-1}, u^{(L)}, \mathbf{\Psi}^{(L)}\right)} p\left(z_n \,\middle|\, f_n^{(L)}; \sigma^2\right)$$

(C.19)

Notice that each step of the approximation is unbiased due to the layer-wise factorization of the joint distribution (Eq. C.17). We can obtain a closed form of the last-layer expectation as $z_n \mid f_n^{(L)}$ is a Gaussian. In the case of using a different distribution, this expectation can be approximated using quadrature (Hensman et al., 2015b).

C.4 Experimental Details

In this section, we present details on implementation and hyperparameters used in our experimental campaign. All experiments were conducted on a server equipped

with a Tesla T4 GPU having 16 GB RAM. Throughout all experiments, unless otherwise stated, we impose an isotropic Gaussian prior over the decoder parameters. We use the radial basis function (RBF) kernel with automatic relevance determination (ARD) with marginal variance and independent lengthscales λ_i per feature (MacKay, 1996a). We place a lognormal prior with unit variance and means equal to 1 and 0.05 for the lengthscales and variance, respectively. Since the auxiliary data of most of the considered datasets are timestamps, we impose a non-informative uniform prior on the inducing inputs. We observe that this prior works well in our experiments. The lengthscales are initialized to 1, whereas the inducing points are initialized by a k-means algorithm as commonly used in practice (Hensman et al., 2015b). For inference, we use an adaptive version of SGHMC (Springenberg et al., 2016) in which the hyperparameters are automatically tuned during a burn-in phase. The hyperparameters are tuned according to the performance on the validation set.

The random seed for the stochastic inference network is drawn from an isotropic Gaussian distribution, i.e. $q(\varepsilon) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In case the encoder is a multilayer perceptron (MLP), we concatenate the random seeds and the input into a long vector. The dimension of the random seeds is the same as that of the input. If the encoder is a convolutional neural network, we spatially stack the random seeds and the input. We use an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 for optimizing the encoder network. Unless otherwise specified, we set the default hyperparameters of the number of SGHMC and encoder optimization steps $J = 30$, and $K = 50$, respectively.

C.4.1 Moving ball experiment

We use the same network architectures as in Jazbec et al. (2021) and Pearce (2019). We follow the data generation procedure of Jazbec et al. (2021), in which a squared-exponential GP kernel with a lengthscale $l = 2$ was used. Notice that, unlike Jazbec et al. (2021), we generate a fixed number of 35 videos for training and another 35 videos for testing rather than generating new training videos at each iteration, regardless of the fact that tens of thousands of iterations are performed. This new setting is more realistic and is designed to show the data efficiency of the considered methods. The number of frames in each video is 30. The dimension of each frame is 32×32. Table C.1 presents the default hyperparameters used for our SGP-BAE and Gaussian Process Bayesian Autoencoder (GP-BAE) models. For the competing methods, we follow the setups specified in Jazbec et al. (2021).

C.4.2 Rotated MNIST experiment

For the rotated MNIST experiment, we follow the same setup as in Jazbec et al. (2021) and Casale et al. (2018). In particular, we use the GP kernel proposed by Casale et al.

(2018) and a neural network consisting of three convolutional layers followed by a fully connected layer in the encoder and vice-versa in the decoder. The details of hyperparameters used for our models are presented in Table C.2. For the competing methods, we refer to Jazbec et al. (2021).

C.4.3 Missing imputation experiment

We follow the experimental setting in Ashman et al. (2020), in which squared exponential GP kernels are used. Notice that, to ensure a fair comparison, we handle partially missing data by treating it as zero and feeding it into the inference network (encoder) for SGP-VAE (Ashman et al., 2020) and our SGP-BAE model. This is because it is not trivial to adapt partial inference networks (Ashman et al., 2020) to our stochastic inference network, and we leave this for future work. Table C.3 and Table C.4 show the default hyperparameters used for our models. For multi-output GP baselines, we refer to Requeima et al. (2019).

C.5 Additional Results

C.5.1 Ablation study on Bayesian treatments of autoencoders

There are several approaches to treating autoencoder models in a fully Bayesian manner. In fact, in Appendix E of the seminal paper on VAE, Kingma and Welling (2014) had already considered a fully Bayesian treatment of VAEs by introducing a prior on the decoder. VI is employed to infer the decoder and the latent variables. Daxberger and Hernández-Lobato (2019) suggested employing SGHMC for sampling decoder parameters, but they use the evidence lower bound (ELBO) of the marginal likelihood of VAE as the objective for sampling. This may result in suboptimal approximations. In contrast, our proposed approach entails direct sampling from the posterior of the decoder and latent variables. In order to differentiate our model from these approaches, we name them as Bayesian variational autoencoders (BVAEs), following the terminology used by Glazunov and Zarras (2022). In particular, we consider the VI (Kingma and Welling, 2014) and SGHMC approaches (Daxberger and Hernández-Lobato, 2019) to treat the decoder of VAEs in a Bayesian manner. These approaches are hereafter referred to as BVAE-VI and BVAE-SGHMC, respectively.

In this section, with the aim of thoroughly disentangling the factors contributing to the superior performance of our proposed models, we present a comprehensive ablation study on these Bayesian treatments of AEs and AE-style models with GP priors. Based on this set of experiments, we identified three key factors that contribute to the improved performance of our proposed models. These factors are: (i) the new amortized SGHMC scheme for inference of the latent variables Z; (ii) treating

Table C.1: Parameter settings for the moving ball experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	2
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN FEEDFORWARD LAYER	500
DIMENSIONALITY OF LATENT SPACE	2
ACTIVATION FUNCTION	TANH
NUM. OF INDUCING POINTS	10
MINI-BATCH SIZE	FULL
STEP SIZE	0.005
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	100
THINNING INTERVAL	400

Table C.2: Parameter settings for the rotated MNIST experiment.

PARAMETER	VALUE
NUM. OF CONV. LAYERS IN ENCODER	3
NUM. OF CONV. LAYERS IN DECODER	3
NUM. OF FILTERS PER CONV. CHANNEL	8
FILTER SIZE	3×3
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	1
ACTIVATION FUNCTION	ELU
DIMENSIONALITY OF LATENT SPACE	16
NUM. OF INDUCING POINTS	32
MINI-BATCH SIZE	512
STEP SIZE	0.01
MOMENTUM	0.01
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	200
THINNING INTERVAL	800

the decoder in a Bayesian manner and using SGHMC for inference properly; (iii) employing a full Bayesian sparse GP prior on the latent variables and using SGHMC for inference. To evaluate the impact of each factor, we evaluate different modeling and inference choices of the decoder, latent variables, and GP prior. As shown in Fig. C.2, the results further demonstrate that our proposal in fact offers the best performance.

Regarding (i), we consider baselines in which the decoder in our models, BAE, GP-BAE, and SGP-BAE, are treated in a non-Bayesian way. In the figure, we name these baselines as BAE-NonBayesDec, GP-BAE-NonBayesDec, SGP-BAE-NonBayesDec, respectively. The results of these new models are slightly worse than the original ones. However, they are still significantly better than variational-based models. Moreover,

Table C.3: Parameter settings for the JURA experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN ENCODER LAYER	20
WIDTH OF A HIDDEN DECODER LAYER	5
DIMENSIONALITY OF LATENT SPACE	2
ACTIVATION FUNCTION	RELU
NUM. OF INDUCING POINTS	128
MINI-BATCH SIZE	100
STEP SIZE	0.002
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	50
THINNING INTERVAL	180

Table C.4: Parameter settings for the EEG experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN ENCODER LAYER	20
WIDTH OF A HIDDEN DECODER LAYER	5
DIMENSIONALITY OF LATENT SPACE	3
ACTIVATION FUNCTION	RELU
NUM. OF INDUCING POINTS	128
MINI-BATCH SIZE	100
STEP SIZE	0.003
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	50
THINNING INTERVAL	180

this also implies the benefit to treat the decoder Bayesian properly (ii).

For (ii), we additionally consider Bayesian VAEs, such as BVAE-VI and BVAE-SGHMC. We find that the Bayesian treatment of the decoder of VAEs is not clearly helpful, even when using SGHMC for the decoder. This is because BVAE-SGHMC still relies on the ELBO as the sampling objective. In contrast, in our models, we sample the decoder directly from the posterior. This is aligned with the recent success of SGHMC on modern Bayesian neural networks (Tran et al., 2022; Izmailov et al., 2021b). Moreover, our models jointly sample all parameters at once, which avoids the inefficiency of iterative switching between optimizing the inference network using VI and sampling the decoder.

To verify (iii), we evaluate our model SGP-BAE in the case where the sparse GP is not treated in a fully Bayesian way. The inducing points and kernel parameters are
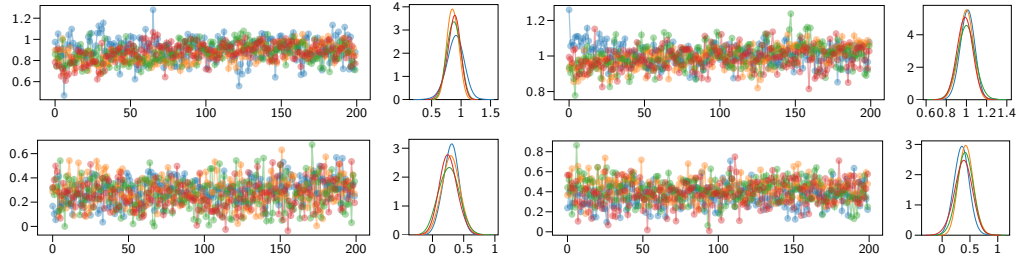
Figure C.3: Trace plots for four test points on the rotated MNIST dataset. Here, we simulate 4 chains with 200 samples for each.

optimized using the objective of Titsias (2009). In Fig. C.2, we term this baseline as SGP-BAE-NonBayesGP. We observe that this model performs much worse than our SGP-BAE model.
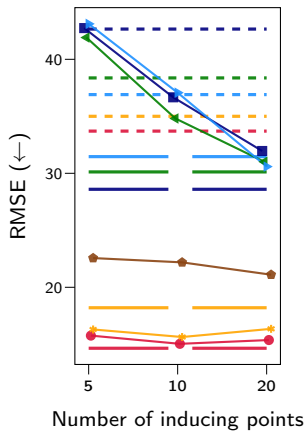
Model	Decoder	Latent variables	GP prior
VAE	Non-Bayesian	VI	None
BVAE-VI	VI	VI	None
BAE-SGHMC	SGHMC	VI	None
BAE	SGHMC	SGHMC	None
BAE-NonBayesDec	None	SGHMC	None
GP-VAE	None	VI	Full GP (Fixed)
GP-BVAE-VI	SGHMC	VI	Full GP (Fixed)
GP-BVAE-SGHMC	SGHMC	VI	Full GP (Fixed)
GP-BAE	SGHMC	SGHMC	Full GP (Fixed)
GP-BAE-NonBayesDec	Non-Bayesian	SGHMC	Full GP (Fixed)
SVGP-VAE	Non-Bayesian	VI	Sparse GP (VI)
SVGP-BVAE-VI	VI	VI	Sparse GP (VI)
SVGP-BVAE-SGHMC	SGHMC	VI	Sparse GP (VI)
SGP-BAE	SGHMC	SGHMC	Bayes. sparse GP (SGHMC)
SGP-BAE-NonBayesDec	Non-Bayesian	SGHMC	Bayes. sparse GP (SGHMC)
SGP-BAE-NonBayesGP	SGHMC	SGHMC	Sparse GP (VI)

Figure C.2: An ablation study on different Bayesian treatments of AE models and AE-style models with GP priors on the moving ball dataset.

C.5.2   Convergence of SGHMC

We follow the common practice of using the $\hat{R}$ potential scale-reduction diagnostic (Gelman and Rubin, 1992) to assess the convergence of Markov chain Monte Carlo (MCMC) on Bayesian neural networks (BNNs) Izmailov et al. (2021b) and Tran et al. (2022). Given two or more chains, $\hat{R}$ estimates the ratio between the chain variance and the average within-chain variance. For the large-scale MNIST experiment, we compute the $\hat{R}$ statistics on the predictive posterior over four independent chains and obtain a median value of less than 1.1, which suggests good convergence Izmailov et al. (2021b). In addition, we present trace plots of the predictive posterior in Fig. C.3, which also support the conclusion of good mixing.

Appendix D

Appendix for Chapter 6

D.1 A Primer on Normalizing Flows and VAEs

Given a dataset $\mathcal{D}$ consisting of N i.i.d samples $\mathcal{D} \stackrel{\Delta}{=} \{x_i\}_{i=1}^N$ with $x_i \in \mathbb{R}^D$, we aim at estimating the unknown continuous generating distribution $p_{\text{data}}(x)$. In order to do so, we introduce a model $p_\theta(x)$ with parameters θ and attempt to estimate θ based on the dataset $\mathcal{D}$. A common approach to estimate θ is to maximize the likelihood of the data, which is equivalent to minimizing the following objective:

$$\mathcal{L}(\theta) \stackrel{\Delta}{=} -\mathbb{E}_{p_{\text{data}}(x)} \left[\log p_\theta(x) \right]. \tag{D.1}$$

Optimization for this objective can be done through a stochastic gradient descent algorithm using minibatches of samples from $p_{\text{data}}(x)$.

D.1.1 Normalizing flows

In flow-based generative models (Papamakarios et al., 2021; Kobyzev et al., 2021), the generative process is defined as:

$$z \sim p_\phi(z); \quad x = f_\psi(z), \tag{D.2}$$

where $z \in \mathbb{R}^D$ is a latent variable, and $p_\phi(z)$ is a tractable base distribution with parameters ϕ, such as an isotropic multivariate Gaussian. The function $f_\psi : \mathbb{R}^D \to \mathbb{R}^D$ is invertible, such that given any input vector x we have $z = f_\psi^{-1}(x)$. A normalizing flow (NF) (Rezende and Mohamed, 2015) defines a sequence of invertible transformations $f = f_1 \circ f_2 \circ \cdots f_K$, such that the relationship between x and z can be written as:

$$x \xleftrightarrow{f_1} h_1 \xleftrightarrow{f_2} h_2 \cdots \xleftrightarrow{f_K} z, \tag{D.3}$$

where $h_k = f_k^{-1}(h_{k-1}; \psi_k)$ and ψ_k are the parameters of the transformation f_k. For the sake of simplicity, we define $h_0 \stackrel{\Delta}{=} x$ and $h_K \stackrel{\Delta}{=} z$. The likelihood of the model given a

data point can be computed analytically using the change of variables as follows:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log p_{\boldsymbol{\phi}}(\boldsymbol{z}) + \log |\det(\partial \boldsymbol{z}/\partial \boldsymbol{x})| \tag{D.4}$$

$$= \log p_{\boldsymbol{\phi}}(\boldsymbol{z}) + \sum_{k=1}^{K} \log |\det(\partial \boldsymbol{h}_k/\partial \boldsymbol{h}_{k-1})|, \tag{D.5}$$

where $\log |\det(\partial \boldsymbol{h}_k/\partial \boldsymbol{h}_{k-1})|$ is the logarithm of absolute value of the determinant of the Jacobian matrix $\partial \boldsymbol{h}_k/\partial \boldsymbol{h}_{k-1}$. This term accounts for the change of measure when going from $\boldsymbol{h}_{k-1}$ to $\boldsymbol{h}_k$ using the transformation f_k. The resulting NF model is then characterized by the set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\phi}\} \cup \{\boldsymbol{\psi}_k\}_{k=1}^K$, which can be estimated using the maximum likelihood estimation (MLE) objective Eq. D.1.

Though NFs allow for exact likelihood computation, they require f_k to be invertible and to have a tractable inverse and Jacobian determinant. This restricts the flexibility to certain transformations that can be used within NFs see e.g., Papamakarios et al., 2021; Kobyzev et al., 2021, and references therein, such as affine coupling (Dinh et al., 2015; Dinh et al., 2017a), invertible convolution (Kingma and Dhariwal, 2018), spline (Durkan et al., 2019; Dolatabadi et al., 2020), or inverse autoregressive transformations (Kingma et al., 2016).

D.1.2 Variational autoencoders

variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende and Mohamed, 2015) introduce a low-dimensional latent variable $\boldsymbol{z} \in \mathbb{R}^P$, with $P \ll D$, to the generative process as follows:

$$\boldsymbol{z} \sim p(\boldsymbol{z}); \quad \boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}). \tag{D.6}$$

Here, $p(\boldsymbol{z})$ is a tractable prior distribution over the latent variables $\boldsymbol{z}$, and $p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z})$, which is also known as a decoder, is usually implemented by a flexible neural network parameterized by $\boldsymbol{\theta}$. Different from NFs, VAEs employ a stochastic transformation $p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z})$ to map $\boldsymbol{z}$ to $\boldsymbol{x}$. Indeed, NFs can be viewed as VAEs where the decoder and encoder are modelled by Dirac deltas $p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}) = \delta(f_{\boldsymbol{\theta}}(\boldsymbol{x}))$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \delta(f_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}))$ respectively, using a restricted family of transformations $f_{\boldsymbol{\theta}}$.

The marginal likelihood of VAEs is intractable and given by:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}. \tag{D.7}$$

A variational lower bound on ther marginal likelihood can be obtained by introducing

a variational distribution $q_\phi(z \mid x)$, with parameters ϕ, which acts as an approximation to the the unknown posterior $p(z \mid x)$:

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{q_\phi(z \mid x)}[\log p_\theta(x \mid z)] - \mathrm{KL}\left[q_\phi(z \mid x) \,\|\, p(z)\right]}_{\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi)}, \qquad \text{(D.8)}$$

where, $\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi)$ is known as the evidence lower bound (ELBO), and the expectation can be approximated by using Monte Carlo samples from $q_\phi(z \mid x)$. This objective can be optimized with stochastic optimization w.r.t. parameters θ and ϕ in place of Eq. D.1.

To tighten the gap between the ELBO and the true marginal likelihood, VAEs can be employed with an expressive form of the approximate posterior $q_\phi(z \mid x)$ such as importance weighted sampling (Burda et al., 2016) or normalizing flows (Kingma et al., 2016; Berg et al., 2018). In addition, to avoid the over regularization offect induced by the prior $p(z)$, one can utilize a flexible prior such as multi-modal distributions (Dilokthanakul et al., 2016; Tomczak and Welling, 2018), hierarchical forms (Sø nderby et al., 2016; Klushyn et al., 2019), or simply reweighing the KL divergence term in the ELBO (Higgins et al., 2017).

D.2 Details on Blurring Mollification

Recently, Rissanen et al. (2023), Hoogeboom and Salimans (2023), and Daras et al. (2023) have proposed appproaches to destroy information of images using blurring operations for diffusion-type generative models. Their approach involves stochastically reversing the heat equation, which is a partial differential equation (PDE) that can be used to erase fine-scale information when applied locally to the 2D plane of an image. In particular, the Laplace PDE for heat diffusions is as follows:

$$\frac{\partial}{\partial t}\tilde{x}(i,j,t) = \Delta \tilde{x}(i,j,t), \qquad \text{(D.9)}$$

where we consider the initial state of the system to be x, the true image data. This PDE can be effectively solved by employing a diagonal matrix within the frequency domain of the cosine transform, provided that the signal is discretized onto a grid. The solution to this equation at time t can be effectively computed by:

$$\tilde{x}_t = A_t x = V D_t V^\top x, \qquad \text{(D.10)}$$

Here, $V^\top$ and V denote the discrete cosine transformation (DCT) and inverse DCT, respectively; the diagonal matrix D_t is the exponent of a weighting matrix for frequencies Λ so that $D_t = \exp(\Lambda t)$. We refer the reader to Appendix A of Rissanen et al. (2023) for the specific definition of Λ. We can evaluate Eq. D.10 in the Fourier domain, which is fast to compute, as the DCT and inverse DCT require $\mathcal{O}(N \log N)$

operations. The equivalent form of Eq. D.10 in the Fourier domain is as follows:

$$\tilde{u}_t = \exp(\boldsymbol{\Lambda} t)\boldsymbol{u}, \tag{D.11}$$

where $\boldsymbol{u} = \boldsymbol{V}^\top \boldsymbol{x} = \mathrm{DCT}(\boldsymbol{x})$. As $\boldsymbol{\Lambda}$ is a diagonal matrix, the above Fourier-space model is fast to evaluate. A Python implementation of this blurring mollification is presented in Algorithm 7.

We follow Rissanen et al. (2023) to set the schedule for the blurring mollification. In particular, we use a logarithmic spacing for the time steps t_k, where $t_0 = \sigma_{B,\max}^2/2$ and $t_T = \sigma_{B,\min}^2/2 = {}^{0.5^2}/2$, corresponding to sub-pixel-size blurring. Here, $\sigma_{B,\max}^2$ is the effective lengthscale-scale of blurring at the beginning of the mollification process. Following Rissanen et al. (2023), we set this to half the width of the image.

Algorithm 7: Python code for blurring mollification

```python
import numpy as np
from scipy.fftpack import dct, idct

def blurring_mollify(x, t):
    # Assuming the image u is an (KxK) numpy array
    K = x.shape[-1]
    freqs = np.pi*np.linspace(0,K-1,K)/K
    frequencies_squared = freqs[:,None]**2 + freqs[None,:]**2
    x_proj = dct(u, axis=0, norm='ortho')
    x_proj = dct(x_proj, axis=1, norm='ortho')
    x_proj = np.exp(-frequencies_squared * t) * x_proj
    x_mollified = idct(x_proj, axis=0, norm='ortho')
    x_mollified = idct(x_mollified, axis=1, norm='ortho')
    return x_mollified
```

D.3 Implementation of Noise Schedules

Algorithm 8 shows the Python code for the noise schedules used in this work. For the sigmoid schedule, following Chen (2023), we set the default values of `start` and `end` to 0 and 3, respectively.

D.4 Experimental Details

D.4.1 Datasets

Synthetic datasets.

- *Mixture of Gaussians*: We consider a mixture of two Gaussians with means $\mu_k = (2\sin(\pi k), 2\cos(\pi k))$ and covariance matrices $\boldsymbol{\Sigma}_k = \sigma^2 \boldsymbol{I}$, where $\sigma = \frac{2}{3}\sin(\pi/2)$. We generate 10K samples for training and 10K samples for testing from this distribution.

Algorithm 8: Python code for noise schedules

```python
import numpy as np

def sigmoid(x):
    # Sigmoid function.
    return 1 / (1 + np.exp(-x))

def sigmoid_schedule(t, T, tau=0.7, start=0, end=3, clip_min=1e-9):
    # A scheduling function based on sigmoid function with a temperature tau.
    v_start = sigmoid(start / tau)
    v_end = sigmoid(end / tau)
    return (v_end - sigmoid((t/T * (end - start) + start) / tau)) / (v_end -
    v_start)

def linear_schedule(t, T):
    # A scheduling function based on linear function.
    return 1 - t/T

def cosine_schedule(t, T, ns=0.0002, ds=0.00025):
    # A scheduling function based on cosine function.
    return np.cos(((t/T + ns) / (1 + ds)) * np.pi / 2)**2
```

- *Von Mises distribution:* We use a von Mises distribution with parameters $\kappa = 1$, and then transform to Cartesian coordinates to obtain a distribution on the unit circle in $\mathbb{R}^2$. We generate 10K training samples and 10K testing from this distribution.

Image datasets. We consider two image datasets including CIFAR10 (Krizhevsky and Hinton, 2009) and CELEBA (Liu et al., 2015). These datasets are publicly available and widely used in the literature of generative models. We use the official train/val/test splits for both datasets. The resolution of CIFAR10 is $3 \times 32 \times 32$. For CELEBA, we pre-process images by first taking a 148×148 center crop and then resizing to the $3 \times 64 \times 64$ resolution.

D.4.2   Software and computational resources

We use NVIDIA P100 and A100 GPUs for the experiments, with 16GB and 80GB of memory respectively. All models are trained on a single GPU except for the experiments with NVAE model (Vahdat and Kautz, 2020), where we employ two A100 GPUs. We use PyTorch (**paszke2019pytorch**) for the implementation of the models and the experiments. Our experiments with VAEs and NFs are relied on the pythae (Chadebec et al., 2022) and normflows (Stimper et al., 2023) libraries, respectively.

D.4.3 Training details

Toy examples.

In the experiments on synthetic datasets, we use a REAL-NVP flow (Dinh et al., 2017a) with 5 affine coupling layers consisting of 2 hidden layers of 64 units each. We train the model for 20000 itereations using an Adam optimizer (Kingma and Ba, 2015) with a learning rate of $5 \cdot 10^{-4}$ and a mini-batch size of 256.

Imaging experiments.

REAL-NVP. We use the multi-scale architecture with deep convolutional residual networks in the coupling layers as described in Dinh et al., 2017a. For the CIFAR10 dataset, we use 4 residual blocks with 32 hidden feature maps for the first coupling layers with checkerboard masking. For the CELEBA dataset, 2 resdiual blocks are employed. We use an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3} and a mini-batch size of 64. We train the model for 100 and 80 epochs on the CIFAR10 and CELEBA datasets, respectively. For the mollification training, we perturb the data for 50 and 40 epochs for CIFAR10 and CELEBA, respectively.

GLOW. We use a multi-scale architecture as described in Kingma and Dhariwal, 2018. The architecture has a depth level of $K = 20$, and a number of levels $L = 3$. We use the AdaMax (Kingma and Ba, 2015) optimizer with a learning rate of $3 \cdot 10^{-4}$ and a mini-batch size of 64. We train the model for 80 and 40 epochs on the CIFAR10 and CELEBA datasets, respectively. For the mollification training, we perturb the data for 50 and 20 epochs for CIFAR10 and CELEBA, respectively.

Table D.1: Neural network architectures used for VAEs in our experiments. Here, CONV$_{(n,s,p)}$ and CONVT$_{(n,s,p)}$ respectively denotes convolutional layer and transposed convolutional layers with n filters, a stride of s and a padding of p, whereas FC$_n$ represents a fully-connected layer with n units, and BN denotes a batch-normalization layer.

	CIFAR10	CELEBA
ENCODER:	$x \in \mathbb{R}^{3 \times 32 \times 32}$	$x \in \mathbb{R}^{3 \times 64 \times 64}$
	$\to$ CONV$_{(128,4,2)} \to$ BN $\to$ RELU	$\to$ CONV$_{(128,4,2)} \to$ BN $\to$ RELU
	$\to$ CONV$_{(256,4,2)} \to$ BN $\to$ RELU	$\to$ CONV$_{(256,4,2)} \to$ BN $\to$ RELU
	$\to$ CONV$_{(512,4,2)} \to$ BN $\to$ RELU	$\to$ CONV$_{(512,4,2)} \to$ BN $\to$ RELU
	$\to$ CONV$_{(1024,4,2)} \to$ BN $\to$ RELU	$\to$ CONV$_{(1024,4,2)} \to$ BN $\to$ RELU
	$\to$ FLATTEN $\to$ FC$_{256 \times 2}$	$\to$ FLATTEN $\to$ FC$_{256 \times 2}$
DECODER:	$z \in \mathbb{R}^{256} \to$ FC$_{8 \times 8 \times 1024}$	$z \in \mathbb{R}^{256} \to$ FC$_{8 \times 8 \times 1024}$
	$\to$ CONVT$_{(512,4,2)} \to$ BN $\to$ RELU	$\to$ CONVT$_{(512,5,2)} \to$ BN $\to$ RELU
	$\to$ CONVT$_{(256,4,2)} \to$ BN $\to$ RELU	$\to$ CONVT$_{(256,5,2)} \to$ BN $\to$ RELU
	$\to$ CONVT$_{(3,4,1)}$	$\to$ CONVT$_{(128,5,2)} \to$ BN $\to$ RELU
		$\to$ CONVT$_{(3,4,1)}$

VAEs. We use convolutional networks for both the encoder and decoder of VAEs (Kingma and Welling, 2014; Rezende et al., 2014). Table D.1 shows the details of the network architectures. We use an Adam optimizer (Kingma and Ba, 2015) with a learning rate of $3 \cdot 10^{-4}$ and a mini-batch size of 128. We train the model for 200 and 100 epochs on the CIFAR10 and CELEBA datasets, respectively. For the mollification training, we perturb the data for 100 and 50 epochs for CIFAR10 and CELEBA, respectively. The addtional details of the variants of VAEs are as follows:

- VAE-IAF (Kingma et al., 2016): We use a 3-layer MADE (Germain et al., 2015) with 128 hidden units and RELU activation for each layer and stack 2 blocks of Masked Autoregressive Flow to create the flow for approximating the posterior.

- β-VAE (Higgins et al., 2017): We use a coefficient of $\beta = 0.1$ for the Kullback-Leibler (KL) term in the ELBO objective.

- IWAE (Burda et al., 2016): We use a number of importance samples of $K = 5$.

- HVAE (Caterini et al., 2018): We set the number of leapfrog steps to used in the integrator to 1. The leapfrog step size is adaptive with an initial value of 0.001

NVAE. We use the default network architecture as described in Vahdat and Kautz, 2020. We train the model on the CIFAR10 for 300 epochs with an AdaMax optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3} and a mini-batch size of 200. For the mollification training, we perturb the data for first 150 epochs.

D.5 Addtional Results

Table D.2 and Table D.3 illustrate uncurated samples from the trained models. Fig. D.1 and Fig. D.2 show the progression of FID scores during training on the CIFAR10 and CELEBA datasets, respectively.

Table D.2: Uncurated samples from the models trained on the CI-FAR10 dataset.

	VANILLA	GAUSS. MOLLIFICATION	BLUR. MOLLIFICATION
REAL-NVP			
GLOW			
VAE			
VAE-IAF			
IWAE			
β-VAE			
HVAE			

Table D.3: Uncurated samples from the models trained on the CELEBA dataset.

	VANILLA	GAUSS. MOLLIFICATION	BLUR. MOLLIFICATION
REAL-NVP			
GLOW			
VAE			
VAE-IAF			
IWAE			
β-VAE			
HVAE			
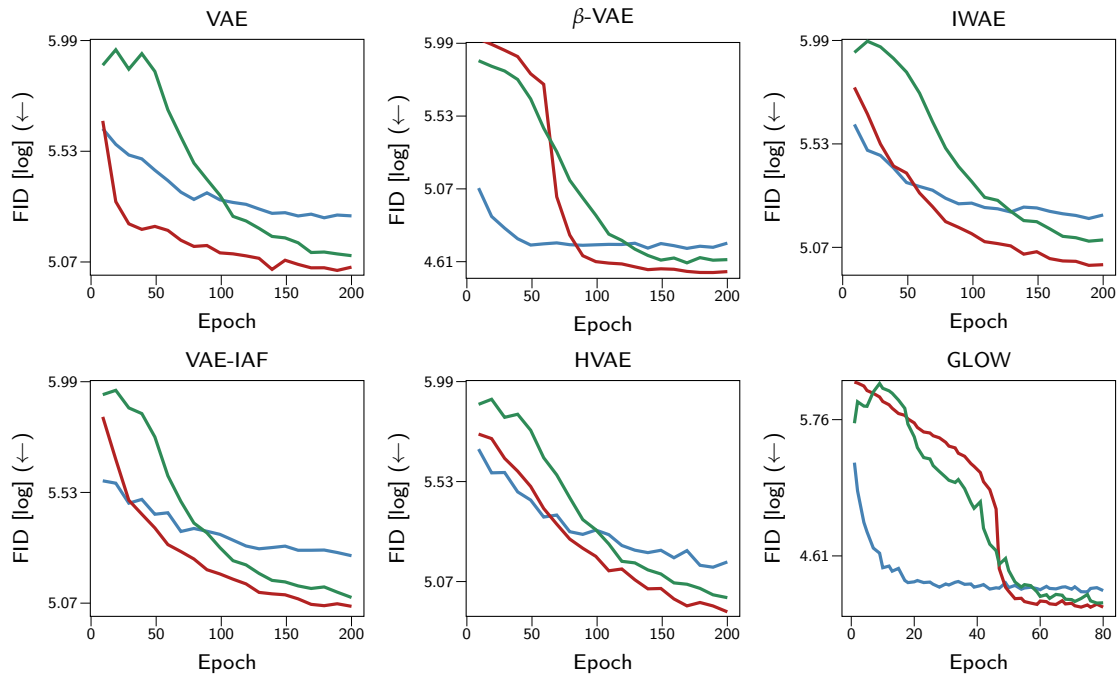
Figure D.1: The progression of FID scores during training on the
CIFAR10 dataset.
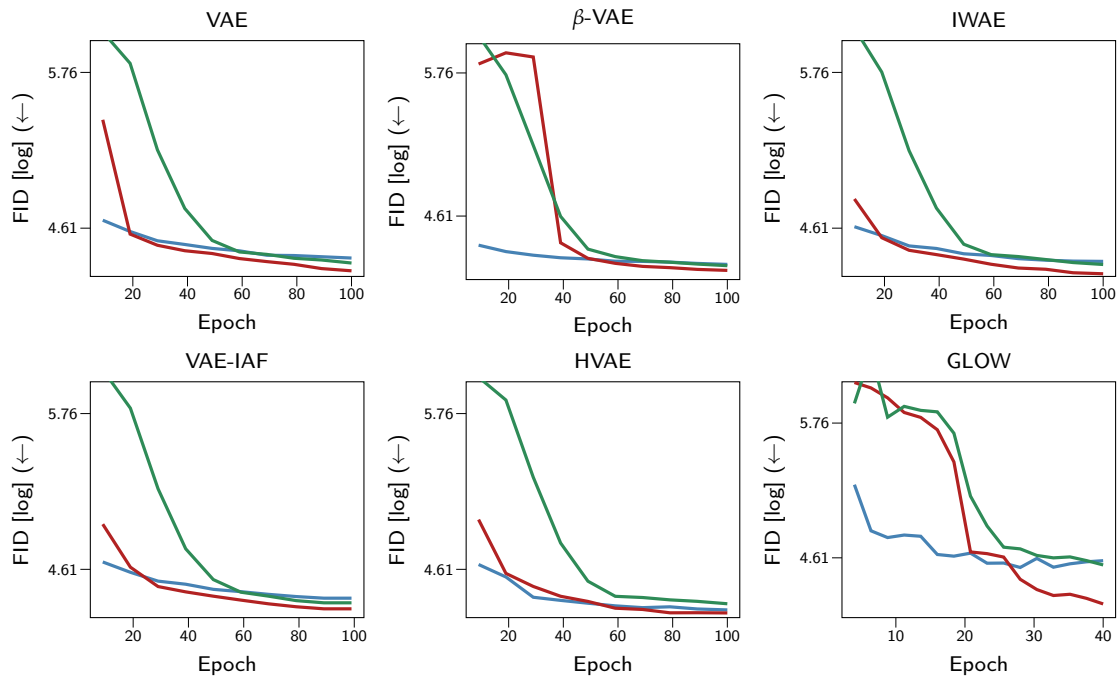
Figure D.2: The progression of FID scores during training on the
CELEBA dataset.

Bibliography

Aitchison, Laurence (2021). "A Statistical Theory of Cold Posteriors in Deep Neural Networks". In: *International Conference on Learning Representations*.

Akaike, H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle". In: *2nd International Symposium on Information Theory, 1973*. Publishing House of the Hungarian Academy of Sciences, pp. 268–281.

Ambrogioni, Luca, Umut Güçlü, Yağmur Güçlütürk, Max Hinne, Marcel A. J. van Gerven, and Eric Maris (2018). "Wasserstein Variational Inference". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Amit, Ron and Ron Meir (2018). "Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 205–214.

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané (2016). "Concrete Poblems in AI Safety". In: *arXiv preprint arXiv:1606.06565*.

Antoran, Javier, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato (2021). "Getting a CLUE: A Method for Explaining Uncertainty Estimates". In: *International Conference on Learning Representations*.

Arjovsky, Martín, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. Vol. 70. Proceedings of Machine Learning Research. Sydney, NSW, Australia: PMLR, pp. 214–223.

Ashman, Matthew, Jonathan So, Will Tebbutt, Vincent Fortuin, Michael Pearce, and Richard E Turner (2020). "Sparse Gaussian Process Variational Autoencoders". In: *arXiv preprint arXiv:2010.10177*.

Ashukha, Arsenii, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov (2020). "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning". In: *International Conference on Learning Representations*.

Atanov, Andrei, Arsenii Ashukha, Kirill Struminsky, Dmitriy Vetrov, and Max Welling (2019). "The Deep Weight Prior". In: *International Conference on Learning Representations*.

Baldi, Pierre and Kurt Hornik (1989). "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima". In: *Neural networks* 2.1, pp. 53–58.

Bauer, Matthias and Andriy Mnih (2019). "Resampled Priors for Variational Autoencoders". In: *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 66–75.

Bengio, Yoshua, Li Yao, Guillaume Alain, and Pascal Vincent (2013a). "Generalized Denoising Auto-Encoders as Generative Models". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 899–907.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013b). "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828.

Bengio, Yoshua, Eric Laufer, Guillaume Alain, and Jason Yosinski (2014). "Deep Generative Stochastic Networks Trainable by Backprop". In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, pp. 226–234.

Berg, Rianne van den, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling (2018). "Sylvester Normalizing Flows for Variational Inference". In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*. AUAI Press, pp. 393–402.

Bishop, Christopher M. (2006). *Pattern recognition and machine learning*. 1st ed. 2006. Corr. 2nd printing 2011. Springer.

Blei, David M. and Michael I. Jordan (2006). "Variational Inference for Dirichlet Process Mixtures". In: *Bayesian Analysis* 1.1, pp. 121 –143. DOI: 10.1214/06-BA104.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational Inference: A Review for Statisticians". In: *Journal of the American statistical Association* 112.518, pp. 859–877.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Network". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1613–1622.

Böhm, Vanessa and Uroš Seljak (2020). "Probabilistic Auto-Encoder". In: *arXiv preprint arXiv:2006.05479*.

Bommasani, Rishi, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill,

et al. (2021). "On the Opportunities and Risks of Foundation Models". In: *arXiv preprint arXiv:2108.07258*.

Bonilla, Edwin V., Karl Krauth, and Amir Dezfouli (2019). "Generic Inference in Latent Gaussian Process Models". In: *Journal of Machine Learning Research* 20.117, pp. 1–63.

Bonneel, Nicolas, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister (2015). "Sliced and Radon Wasserstein Barycenters of Measures". In: *Journal of Mathematical Imaging and Vision* 51.1, pp. 22–45.

Briol, François-Xavier, Chris J. Oates, Mark Girolami, Michael A. Osborne, and Dino Sejdinovic (2019). "Probabilistic Integration: A Role in Statistical Computation?" In: *Statistical Science* 34.1, pp. 1–22.

Brown, Bradley CA, Anthony L. Caterini, Brendan Leigh Ross, Jesse C Cresswell, and Gabriel Loaiza-Ganem (2023). "Verifying the Union of Manifolds Hypothesis for Image Data". In: *International Conference on Learning Representations*.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901.

Burda, Yuri, Roger B. Grosse, and Ruslan Salakhutdinov (2016). "Importance Weighted Autoencoders". In: *International Conference on Learning Representations*.

Burkardt, John (2014). "The Truncated Normal Distribution". In: *Department of Scientific Computing Website, Florida State University*, pp. 1–35.

Carbone, Ginevra, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti (2020). "Robustness of Bayesian Neural Networks to Gradient-Based Attacks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 15602–15613.

Casale, Francesco Paolo, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi (2018). "Gaussian Process Prior Variational Autoencoders". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Caterini, Anthony L, Arnaud Doucet, and Dino Sejdinovic (2018). "Hamiltonian Variational Auto-Encoder". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Chadebec, Clément, Louis Vincent, and Stephanie Allassonniere (2022). "Pythae: Unifying Generative Autoencoders in Python - A Benchmarking Use Case". In:

Advances in Neural Information Processing Systems. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., pp. 21575–21589.

Chen, Tianqi, Emily Fox, and Carlos Guestrin (2014). "Stochastic Gradient Hamiltonian Monte Carlo". In: *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*. Proceedings of Machine Learning Research. Bejing, China: PMLR, pp. 1683–1691.

Chen, Tianrong, Guan-Horng Liu, and Evangelos Theodorou (2022). "Likelihood Training of Schrödinger Bridge using Forward-Backward SDEs Theory". In: *International Conference on Learning Representations*.

Chen, Ting (2023). "On the Importance of Noise Scheduling for Diffusion Models". In: *arXiv preprint arXiv:2301.10972*.

Chen, Xi, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel (2017). "Variational Lossy Autoencoder". In: *International Conference on Learning Representations*.

Cockayne, Jon, Chris J. Oates, Ilse C.F. Ipsen, and Mark Girolami (2019). "A Bayesian Conjugate Gradient Method (with Discussion)". In: *Bayesian Analysis* 14.3, pp. 937–1012.

Cornish, Rob, Anthony Caterini, George Deligiannidis, and Arnaud Doucet (2020). "Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 2133–2143.

Cottrell, Garrison W., Paul Munro, and David Zipser (1989). "Image Compression by Back Propagation: A Demonstration of Extensional Programming". In: *Models of Cognition*, pp. 208–240.

Cremer, Chris, Xuechen Li, and David Duvenaud (2018). "Inference Suboptimality in Variational Autoencoders". In: *International Conference on Learning Representations*.

Dai, Bin and David Wipf (2019). "Diagnosing and Enhancing VAE Models". In: *International Conference on Learning Representations*.

Dai, Zhenwen, Andreas C. Damianou, Javier González, and Neil D. Lawrence (2015). "Variational Auto-encoded Deep Gaussian Processes". In: *International Conference on Learning Representations*.

Damianou, Andreas and Neil D. Lawrence (2013). "Deep Gaussian Processes". In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. Vol. 31. Proceedings of Machine Learning Research. Scottsdale, Arizona, USA: PMLR, pp. 207–215.

Dangel, Felix, Frederik Kunstner, and Philipp Hennig (2020). "BackPACK: Packing more into Backprop". In: *International Conference on Learning Representations*.

Daras, Giannis, Mauricio Delbracio, Hossein Talebi, Alex Dimakis, and Peyman Milanfar (2023). "Soft Diffusion: Score Matching with General Corruptions". In: *Transactions on Machine Learning Research*. ISSN: 2835-8856.

Daxberger, Erik A. and José Miguel Hernández-Lobato (2019). "Bayesian Variational Autoencoders for Unsupervised Out-of-Distribution Detection". In: *arXiv preprint arXiv:1912.05651*.

Daxberger, Erik A., Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, M. Bauer, and Philipp Hennig (2021). "Laplace Redux – Effortless Bayesian Deep Learning". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 20089–20103.

Delattre, Sylvain and Nicolas Fournier (2017). "On the Kozachenko–Leonenko entropy estimator". In: *Journal of Statistical Planning and Inference* 185, pp. 69–93.

Der Kiureghian, Armen and Ove Ditlevsen (2009). "Aleatory or Epistemic? Does It Matter?" In: *Structural Safety* 31.2, pp. 105–112.

Dhariwal, Prafulla and Alexander Nichol (2021). "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 8780–8794.

Dieng, Adji Bousso, Dustin Tran, Rajesh Ranganath, John Paisley, and David Blei (2017). "Variational Inference via χ Upper Bound Minimization". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Dilokthanakul, Nat, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan (2016). "Deep unsupervised Clustering with Gaussian Mixture Variational Autoencoders". In: *arXiv preprint arXiv:1611.02648*.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). "NICE: Non-linear Independent Components Estimation". In: *International Conference on Learning Representations*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017a). "Density Estimation Using Real NVP". In: *International Conference on Learning Representations*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017b). "Density Estimation Using Real NVP". In: *International Conference on Learning Representations*.

Dolatabadi, Hadi Mohaghegh, Sarah Erfani, and Christopher Leckie (2020). "Invertible Generative Modeling using Linear Rational Splines". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 4236–4246.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*.

Du, Yilun and Igor Mordatch (2019). "Implicit Generation and Modeling with Energy Based Models". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*.

Duane, Simon, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth (1987). "Hybrid Monte Carlo". In: *Physics Letters B* 195.2, pp. 216 –222.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61, pp. 2121–2159.

Durkan, Conor, Artur Bekasov, Iain Murray, and George Papamakarios (2019). "Neural Spline Flows". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Duvenaud, David (2014). "Automatic Model Construction with Gaussian Processes". PhD thesis. University of Cambridge.

Duvenaud, David, Oren Rippel, Ryan Adams, and Zoubin Ghahramani (2014). "Avoiding pathologies in very deep networks". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, pp. 202–210.

Elinas, Pantelis, Edwin V Bonilla, and Louis Tiao (2020). "Variational Inference for Graph Convolutional Networks in the Absence of Graph Data and Adversarial Settings". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 18648–18660.

Feng, Yihao, Dilin Wang, and Qiang Liu (2017). "Learning to Draw Samples with Amortized Stein Variational Gradient Descent". In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press.

Flam-Shepherd, Daniel, James Requeima, and David Duvenaud (2017). "Mapping Gaussian Process Priors to Bayesian Neural Networks". In: *NeurIPS workshop on Bayesian Deep Learning*.

– (2018). "Characterizing and Warping the Function space of Bayesian Neural Networks". In: *NeurIPS workshop on Bayesian Deep Learning*.

Fortuin, Vincent (2022). "Priors in Bayesian Deep Learning: A Review". In: *International Statistical Review* 90.3, pp. 563–591.

Fortuin, Vincent, Dmitry Baranchuk, Gunnar Raetsch, and Stephan Mandt (2020). "GP-VAE: Deep Probabilistic Time Series Imputation". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. Proceedings of Machine Learning Research. PMLR, pp. 1651–1661.

Gal, Yarin (2016). "Uncertainty in Deep Learning". In: *PhD Thesis, University of Cambridge*.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*. Vol. 48. Proceedings of Machine Learning Research. New York City, NY, USA: JMLR, pp. 1050–1059.

Gargiani, Matilde, Andrea Zanelli, Quoc Tran-Dinh, Moritz Diehl, and Frank Hutter (2020). "Convergence Analysis of Homotopy-SGD for Non-convex Optimization". In: *arXiv preprint arXiv:2011.10298*.

Garnelo, Marta, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh (2018). "Neural Processes". In: *arXiv preprint arXiv:1807.01622*.

Gelman, Andrew and Donald B. Rubin (1992). "Inference from Iterative Simulation using Multiple Sequences". In: *Statistical Science* 7.4, pp. 457–472.

Genevay, Aude, Gabriel Peyré, and Marco Cuturi (2017). "GAN and VAE from an Optimal Transport Point of View". In: *arXiv preprint arXiv:1706.01807*.

Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle (2015). "MADE: Masked Autoencoder for Distribution Estimation". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 881–889.

Ghahramani, Zoubin (2015). "Probabilistic Machine Learning and Artificial Intelligence". In: *Nature* 521.7553, pp. 452–459.

Ghosh, Partha, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf (2020). "From Variational to Deterministic Autoencoders". In: *International Conference on Learning Representations*.

Glazunov, Misha and Apostolis Zarras (2022). "Do Bayesian Variational Autoencoders Know What They Don't Know?" In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. Vol. 180. Proceedings of Machine Learning Research. PMLR, pp. 718–727.

Gneiting, Tilmann and Adrian E Raftery (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American statistical Association* 102.477, pp. 359–378.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., pp. 2672–2680.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.

Graves, Alex (2011). "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., pp. 2348–2356.

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola (2012). "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* 13, pp. 723–773.

Grigorescu, Sorin, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu (2020). "A Survey of Deep Learning Techniques for Autonomous Driving". In: *Journal of Field Robotics* 37.3, pp. 362–386.

Grover, Aditya, Manik Dhar, and Stefano Ermon (2018). "Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models". In: *Proceedings of the 32nd Conference on Artificial Intelligence, AAAI 2018*. New Orleans, Louisiana, USA: AAAI Press, pp. 3069–3076.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., pp. 5767–5777.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (2017). "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 1321–1330.

Ha, David, Andrew M. Dai, and Quoc V. Le (2017). "HyperNetworks". In: *International Conference on Learning Representations*.

Hafner, Danijar, Dustin Tran, Timothy P. Lillicrap, Alex Irpan, and James Davidson (2019). "Noise Contrastive Priors for Functional Uncertainty". In: *Proceedings of the 35h Conference on Uncertainty in Artificial Intelligence, UAI 2019*. Tel Aviv, Israel: AUAI Press, p. 332.

Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp (2011). "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Rev.* 53.2, pp. 217–288. DOI: 10.1137/090771806.

Hasanzadeh, Arman, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian (2020). "Bayesian Graph Neural

Networks with Adaptive Connection Sampling". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 4094–4104.

Hazan, Elad, Kfir Yehuda Levy, and Shai Shalev-Shwartz (2016). "On Graduated Optimization for Stochastic Non-Convex Problems". In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1833–1841.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Identity Mappings in Deep Residual Networks". In: *Proceeding of the 14th European Conference on Computer Vision*. Vol. 9908 (Part IV). Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, pp. 630–645.

Heek, Jonathan and Nal Kalchbrenner (2019). "Bayesian Inference for Large Scale Image Classification". In: *arXiv preprint arXiv:1908.03491*.

Hein, Matthias, Maksym Andriushchenko, and Julian Bitterwolf (2019). "Why ReLU Networks Yield High-confidence Predictions Far Away from the Training Data and How to Mitigate the Problem". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 41–50.

Helgason, Sigurdur (2010). *Integral Geometry and Radon Transforms*. Springer Science & Business Media.

Hendrycks, Dan and Thomas Dietterich (2019). "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *International Conference on Learning Representations*.

Hensman, James, Nicolò Fusi, and Neil D. Lawrence (2013). "Gaussian Processes for Big Data". In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI'13. Bellevue, WA: AUAI Press, 282–290.

Hensman, James, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani (2015a). "MCMC for Variationally Sparse Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc.

Hensman, James, Alexander G. de G. Matthews, and Zoubin Ghahramani (2015b). "Scalable Variational Gaussian Process Classification". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*. Vol. 38. JMLR Workshop and Conference Proceedings. JMLR.org.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems 30:*

Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 6626–6637.

Higgins, Irina, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 6840–6851.

Hoffman, Matthew D. and Andrew Gelman (2014). "The No-U-turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *Journal of Machine Learning Research* 15.1, pp. 1593–1623.

Hoogeboom, Emiel and Tim Salimans (2023). "Blurring Diffusion Models". In: *International Conference on Learning Representations*.

Hoogeboom, Emiel, Jonathan Heek, and Tim Salimans (2023). "simple diffusion: End-to-end Diffusion for High Resolution Images". In: *arXiv preprint arXiv:2301.11093*.

Houlsby, Neil, Ferenc Huszar, Zoubin Ghahramani, and Jose Hernández-lobato (2012). "Collaborative Gaussian Processes for Preference Learning". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., pp. 2096–2104.

Hsu, Hans Hao-Hsun, Yuesong Shen, Christian Tomani, and Daniel Cremers (2022). "What Makes Graph Neural Networks Miscalibrated?" In: *Advances in Neural Information Processing Systems*. Vol. 35.

Huszár, Ferenc (2017). "Variational Inference using Implicit Distributions". In: *arXiv preprint arXiv:1702.08235*.

Hyvärinen, Aapo (2005). "Estimation of Non-Normalized Statistical Models by Score Matching". In: *Journal of Machine Learning Research* 6.24, pp. 695–709.

Immer, Alexander, Maciej Korzepa, and Matthias Bauer (2021a). "Improving Predictions of Bayesian Neural Nets via Local Linearization". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 703–711.

Immer, Alexander, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan (2021b). "Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 4563–4573.

Immer, Alexander, Tycho van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk (2022). "Invariance Learning in Deep Neural Networks with Differentiable Laplace Approximations". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 12449–12463.

Iwakiri, Hidenori, Yuhang Wang, Shinji Ito, and Akiko Takeda (2022). "Single Loop Gaussian Homotopy Method for Non-convex Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 7065–7076.

Izmailov, Pavel, Wesley Maddox, Polina Kirichenko, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson (2019). "Subspace Inference for Bayesian Deep Learning". In: *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*. Vol. 115. Proceedings of Machine Learning Research. AUAI Press, pp. 1169–1179.

Izmailov, Pavel, Patrick Nicholson, Sanae Lotfi, and Andrew G Wilson (2021a). "Dangers of Bayesian Model Averaging under Covariate Shift". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 3309–3322.

Izmailov, Pavel, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson (2021b). "What Are Bayesian Neural Network Posteriors Really Like?" In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 4629–4640.

Jabri, Allan, David Fleet, and Ting Chen (2022). "Scalable Adaptive Computation for Iterative Generation". In: *arXiv preprint arXiv:2212.11972*.

Jacot, Arthur, Franck Gabriel, and Clement Hongler (2018). "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., pp. 8571–8580.

Jankowiak, Martin and Fritz Obermeyer (2018). "Pathwise Derivatives Beyond the Reparameterization Trick". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 2240–2249.

Jazbec, Metod, Matt Ashman, Vincent Fortuin, Michael Pearce, Stephan Mandt, and Gunnar Rätsch (2021). " Scalable Gaussian Process Variational Autoencoders ". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 3511–3519.

Jeffreys, Harold (1946). "An Invariant Form for the Prior Probability in Estimation Problems". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 186.1007, pp. 453–461.

Jordan, Michael I, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine learning* 37.2, pp. 183–233.

Kantorovich, Leonid Vitaliyevich (1942). "On the transfer of masses". In: *Doklady Akademii Nauk SSSR* 37, pp. 227–229.

– (1948). "On a problem of Monge". In: *Uspekhi Matematicheskikh Nauk* 3, pp. 225–226.

Karaletsos, Theofanis and Thang D. Bui (2020). "Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 17141–17152.

Karras, Tero, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila (2020). "Analyzing and Improving the Image Quality of StyleGAN". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, pp. 8107–8116. DOI: 10.1109/CVPR42600.2020.00813.

Kass, Robert E and Adrian E Raftery (1995). "Bayes Factors". In: *Journal of the american statistical association* 90.430, pp. 773–795.

Kendall, Alex and Yarin Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., pp. 5574–5584.

Khan, Mohammad Emtiyaz E, Alexander Immer, Ehsan Abedi, and Maciej Korzepa (2019). "Approximate Inference Turns Deep Networks into Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Kim, Samuel, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljacic (2022). "Deep Learning for Bayesian Optimization of Scientific Problems with High-Dimensional Structure". In: *Transactions on Machine Learning Research*.

Kingma, Diederik, Tim Salimans, Ben Poole, and Jonathan Ho (2021). "Variational Diffusion Models". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 21696–21707.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*.

– (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4, pp. 307–392.

Kingma, Durk P and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Kingma, Durk P, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improved Variational Inference with Inverse Autoregressive Flow".

In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., pp. 4743–4751.

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*.

Kirillov, Alexander, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. (2023). "Segment Anything". In: *arXiv preprint arXiv:2304.02643*.

Klushyn, Alexej, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt (2019). "Learning Hierarchical Priors in VAEs". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Kobyzev, Ivan, Simon J. D. Prince, and Marcus A. Brubaker (2021). "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11, pp. 3964–3979.

Koh, Pang Wei and Percy Liang (2017). "Understanding Black-box Predictions via Influence Functions". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 1885–1894.

Kolouri, Soheil, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K. Rohde (2019). "Generalized Sliced Wasserstein Distances". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 261–272.

Korattikara Balan, Anoop, Vivek Rathod, Kevin P Murphy, and Max Welling (2015). "Bayesian Dark Knowledge". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc.

Krishnan, Rahul, Dawen Liang, and Matthew Hoffman (2018). "On the Challenges of Learning with Inference Networks on Sparse, High-dimensional Data". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 143–151.

Krizhevsky, A. and G. Hinton (2009). "Learning multiple layers of features from tiny images". In: *Master's thesis, Department of Computer Science, University of Toronto*.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., pp. 6402–6413.

Lalchand, Vidhi, Aditya Ravuri, and Neil D. Lawrence (2022a). " Generalised GPLVM with Stochastic Variational Inference ". In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Vol. 151. Proceedings of Machine Learning Research. PMLR, pp. 7841–7864.

Lalchand, Vidhi, Wessel Bruinsma, David Burt, and Carl Edward Rasmussen (2022b). "Sparse Gaussian Process Hyperparameters: Optimize or Integrate?" In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 16612–16623.

Lawrence, Neil D. (2005). "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models". In: *Journal of Machine Learning Research* 6, pp. 1783–1816.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

Lee, Jaehoon, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein (2020). "Finite Versus Infinite Neural Networks: an Empirical Study". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 15156–15172.

Lee, Kimin, Honglak Lee, Kibok Lee, and Jinwoo Shin (2018). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *International Conference on Learning Representations*.

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein (2018). "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6391–6401.

Li, Yingzhen and Richard E Turner (2016). "Rényi Divergence Variational Inference". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc.

Li, Yingzhen, Richard E Turner, and Qiang Liu (2017). "Approximate Inference with Amortised MCMC". In: *arXiv preprint arXiv:1702.08343*.

Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez (2017). "A Survey on Deep Learning in Medical Image Analysis". In: *Medical image analysis* 42, pp. 60–88.

Liu, H., Y. S. Ong, X. Shen, and J. Cai (2020). "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11, pp. 4405–4423.

Liu, Qiang and Dilin Wang (2016). "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., pp. 2378–2386.

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep Learning Face Attributes in the Wild". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, pp. 3730–3738. DOI: 10.1109/ICCV.2015.425.

Liutkus, Antoine, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter (2019). "Sliced-Wasserstein Flows: Nonparametric Generative Modeling via Optimal Transport and Diffusions". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 4104–4113.

Loaiza-Ganem, Gabriel and John P. Cunningham (2019). "The Continuous Bernoulli: Fixing a Pervasive Error in Variational Autoencoders". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13266–13276.

Loaiza-Ganem, Gabriel, Brendan Leigh Ross, Luhuan Wu, John P Cunningham, Jesse C Cresswell, and Anthony L Caterini (2022a). "Denoising Deep Generative Models". In: *I Can't Believe It's Not Better Workshop at NeurIPS 2022*.

Loaiza-Ganem, Gabriel, Brendan Leigh Ross, Jesse C Cresswell, and Anthony L. Caterini (2022b). "Diagnosing and Fixing Manifold Overfitting in Deep Generative Models". In: *Transactions on Machine Learning Research*. ISSN: 2835-8856.

Lotfi, Sanae, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson (2022). "Bayesian Model Selection, the Marginal Likelihood, and Generalization". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 14223–14247.

Louizos, Christos and Max Welling (2017). "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 2218–2227.

Lucic, Mario, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet (2018). "Are GANs Created Equal? A Large-Scale Study". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 698–707.

Ma, Chao, Yingzhen Li, and Jose Miguel Hernandez-Lobato (2019). "Variational Implicit Processes". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 4222–4233.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605.

Mackay, David J. C. (2003). *Information Theory, Inference and Learning Algorithms*. First Edition. Cambridge University Press.

MacKay, David JC (1992). "Information-based objective functions for active data selection". In: *Neural computation* 4.4, pp. 590–604.

– (1995). "Probable Networks and Plausible Predictions - a Review of Practical Bayesian Methods for Supervised Neural Networks". In: *Network: Computation in Neural Systems* 6.3, pp. 469–505.

– (1996a). "Bayesian Methods for Backpropagation Networks". In: *Models of neural networks III*. Springer, pp. 211–254.

– (1996b). "Bayesian non-linear modeling for the prediction competition". In: *Maximum Entropy and Bayesian Methods*. Springer, pp. 221–234.

MacKay, David JC and Mark N Gibbs (1999). "Density Networks". In: *Statistics and Neural Networks: Advances at the Interface*, pp. 129–144.

Mackay, David John Cameron (1992). "Bayesian Methods for Adaptive Models". UMI Order No. GAX92-32200. PhD thesis. USA: California Institute of Technology.

Maddox, Wesley J., Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson (2019). "A Simple Baseline for Bayesian Uncertainty in Deep Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13132–13143.

Marcus, Gary (2018). "Deep Learning: A Critical Appraisal". In: *arXiv preprint arXiv:1801.00631*.

Marino, Joe, Yisong Yue, and Stephan Mandt (2018). "Iterative Amortized Inference". In: *International Conference on Machine Learning*. PMLR, pp. 3403–3412.

Matsubara, Takuo, Chris J. Oates, and François-Xavier Briol (2021). "The Ridgelet Prior: A Covariance Function Approach to Prior Specification for Bayesian Neural Networks". In: *Journal of Machine Learning Research* 22.157, pp. 1–57.

Matthews, Alexander, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani (2018). "Gaussian Process Behaviour in Wide Deep Neural Networks". In: *International Conference on Learning Representations*.

Meng, Chenlin, Jiaming Song, Yang Song, Shengjia Zhao, and Stefano Ermon (2021). "Improved Autoregressive Modeling with Distribution Smoothing". In: *International Conference on Learning Representations*.

Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger (2017). "Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 2391–2400.

Miani, Marco, Frederik Warburg, Pablo Moreno-Muñoz, Nicki Skafte, and Sø ren Hauberg (2022). "Laplacian Autoencoders for Learning Stochastic Representations". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 21059–21072.

Mikkola, Petrus, Osvaldo A. Martin, Suyog Chandramouli, Marcelo Hartmann, Oriol Abril Pla, Owen Thomas, Henri Pesonen, Jukka Corander, Aki Vehtari, Samuel Kaski, Paul-Christian Bürkner, and Arto Klami (2023). "Prior Knowledge Elicitation: The Past, Present, and Future". In: *Bayesian Analysis*, pp. 1 –33.

Minderer, Matthias, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic (2021). "Revisiting the Calibration of Modern Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 15682–15694.

Močkus, J. (1975). "On Bayesian Methods for Seeking the Extremum". In: *Optimization Techniques IFIP Technical Conference Novosibirsk*. Springer Berlin Heidelberg, pp. 400–404.

Montavon, Grégoire, Wojciech Samek, and Klaus-Robert Müller (2018). "Methods for Interpreting and Understanding Deep Neural Networks". In: *Digital signal processing* 73, pp. 1–15.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (2016). "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 2574–2582.

Müller, Alfred (1997). "Integral Probability Metrics and Their Generating Classes of Functions". In: *Advances in Applied Probability* 29.2, pp. 429–443.

Murray, Iain and Zoubin Ghahramani (2005). *A Note on the Evidence and Bayesian Occam's Razor*. Tech. rep. GCNU-TR 2005-003. Gatsby Computational Neuroscience Unit, University College London.

Nalisnick, Eric T. (2018). "On Priors for Bayesian Neural Networks". PhD thesis. University of California, Irvine, USA.

Nalisnick, Eric T. and Padhraic Smyth (2017). "Stick-Breaking Variational Autoencoders". In: *International Conference on Learning Representations*.

Nalisnick, Eric T., Jonathan Gordon, and José Miguel Hernández-Lobato (2021a). "Predictive Complexity Priors". In: vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 694–702.

– (2021b). "Predictive Complexity Priors". In: *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 694–702.

Narayanan, Hariharan and Sanjoy Mitter (2010). "Sample Complexity of Testing the Manifold Hypothesis". In: *Advances in Neural Information Processing Systems*. Vol. 23. Curran Associates, Inc.

Neal, Radford M. (1996). *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. 1st ed. Springer.

– (2011). "MCMC Using Hamiltonian Dynamics". In: *Handbook of Markov Chain Monte Carlo*. CRC Press. Chap. 5.

Nguyen, Khai, Nhat Ho, Tung Pham, and Hung Bui (2021). "Distributional Sliced-Wasserstein and Applications to Generative Modeling". In: *International Conference on Learning Representations*.

Nguyen, XuanLong, Martin J. Wainwright, and Michael I. Jordan (2010). "Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization". In: *IEEE Transactions on Information Theory* 56.11, pp. 5847–5861.

Nichol, Alexander Quinn and Prafulla Dhariwal (2021). "Improved Denoising Diffusion Probabilistic Models". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8162–8171.

Nogueira, Fernando (2014). *Bayesian Optimization: Open source constrained global optimization tool for Python*. URL: https://github.com/fmfn/BayesianOptimization.

O'Hagan, A. (1991). "Bayes–Hermite quadrature". In: *Journal of Statistical Planning and Inference* 29.3, pp. 245 –260.

Onken, Derek, Samy Wu Fung, Xingjian Li, and Lars Ruthotto (2021). "OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10, pp. 9223–9232.

Osawa, Kazuki, Siddharth Swaroop, Mohammad Emtiyaz Khan, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, and Rio Yokota (2019b). "Practical Deep Learning with Bayesian Principles". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4289–4301.

Osawa, Kazuki, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota (2019a). "Practical Deep Learning with Bayesian Principles". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 4287–4299.

Ovadia, Yaniv, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek (2019). "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 13991–14002.

Ozakin, Arkadas and Alexander Gray (2009). "Submanifold Density Estimation". In: *Advances in Neural Information Processing Systems*. Vol. 22. Curran Associates, Inc.

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2021). "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57, pp. 1–64.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 8026–8037.

Pavon, Michele, Giulio Trigila, and Esteban G Tabak (2021). "The Data-Driven Schrödinger Bridge". In: *Communications on Pure and Applied Mathematics* 74.7, pp. 1545–1573.

Pearce, Michael (2019). "The Gaussian Process Prior VAE for Interpretable Latent Dynamics from Pixels". In: *Symposium on Advances in Approximate Bayesian Inference, AABI 2019, Vancouver, BC, Canada, December 8, 2019*. Vol. 118. Proceedings of Machine Learning Research. PMLR, pp. 1–12.

Pearce, Tim, Russell Tsuchida, Mohamed Zaki, Alexandra Brintrup, and Andy Neely (2019). "Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions". In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*. Tel Aviv, Israel: AUAI Press, p. 25.

Pennec, Xavier (2006). "Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements". In: *Journal of Mathematical Imaging and Vision* 25, pp. 127–154.

Phan, Du, Neeraj Pradhan, and Martin Jankowiak (2019). "Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro". In: arXiv:1912.11554.

Pope, Phil, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein (2021). "The Intrinsic Dimension of Images and Its Impact on Learning". In: *International Conference on Learning Representations*.

Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research* 6.65, pp. 1939–1959.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference*

on Machine Learning. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8748–8763.

Ramchandran, Siddharth, Gleb Tikhonov, Kalle Kujanpää, Miika Koskinen, and Harri Lähdesmäki (2021). " Longitudinal Variational Autoencoder ". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 3898–3906.

Rasmussen, Carl E. and Christopher Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rasmussen, Carl Edward and Zoubin Ghahramani (2002). "Bayesian Monte Carlo". In: *Advances in Neural Information Processing Systems*. Vol. 15. MIT Press, pp. 489–496.

Requeima, James, William Tebbutt, Wessel Bruinsma, and Richard E. Turner (2019). "The Gaussian Process Autoregressive Regression Model (GPAR)". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 1860–1869.

Reshetova, Daria, Yikun Bai, Xiugang Wu, and Ayfer Özgür (2021). "Understanding Entropic Regularization in GANs". In: *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 825–830.

Rezende, Danilo and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1530–1538.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*. Vol. 32. Proceeding of Machine Learning Research. Beijing, China: PMLR, pp. 1278–1286.

Rissanen, Severi, Markus Heinonen, and Arno Solin (2023). "Generative Modelling with Inverse Heat Dissipation". In: *International Conference on Learning Representations*.

Robbins, Herbert (1956). "An Empirical Bayes Approach to Statistics". In: *Proceedings of the 3rd Berkeley Symposium on Mathematical Statistics and Probability, 1956*. Vol. 1, pp. 157–163.

Rossi, Simone, Pietro Michiardi, and Maurizio Filippone (2019). "Good Initializations of Variational Bayes for Deep Models". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 5487–5497.

Rossi, Simone, Sebastien Marmin, and Maurizio Filippone (2020). "Walsh-Hadamard Variational Inference for Bayesian Deep Learning". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 9674–9686.

Rossi, Simone, Markus Heinonen, Edwin Bonilla, Zheyang Shen, and Maurizio Filippone (2021). " Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations ". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 1837–1845.

Roweis, Sam T and Lawrence K Saul (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500, pp. 2323–2326.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536.

Salimbeni, Hugh and Marc Deisenroth (2017). "Doubly Stochastic Variational Inference for Deep Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Salmona, Antoine, Valentin De Bortoli, Julie Delon, and Agnes Desolneux (2022). "Can Push-forward Generative Models Fit Multimodal Distributions?" In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 10766–10779.

Sanjabi, Maziar, Jimmy Ba, Meisam Razaviyayn, and Jason D Lee (2018). "On the Convergence and Robustness of Training GANs with Regularized Optimal Transport". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Sebastian, W. Ober, Carl E. Rasmussen, and Mark van der Wilk (2021). "The Promises and Pitfalls of Deep Kernel Learning". In: *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence, UAI 2021, July 27-30, 2021, Virtual Event*.

Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra (2017). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 618–626.

Settles, Burr (2009). *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences.

Shi, Jiaxin, Shengyang Sun, and Jun Zhu (2018). "A Spectral Approach to Gradient Estimation for Implicit Distributions". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4644–4653.

Shi, Jiaxin, Mohammad Emtiyaz Khan, and Jun Zhu (2019). "Scalable Training of Inference Networks for Gaussian-Process Models". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 5758–5768.

Shwartz-Ziv, Ravid, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew G Wilson (2022). "Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 27706–27715.

Shwartz-Ziv, Ravid, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew Gordon Wilson (2022). "Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors". In: *Advances in Neural Information Processing Systems*. Vol. 35.

Simonyan, Karen and Andrew Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv preprint arXiv:1409.1556*.

Skafte, Nicki, Martin Jorgensen, and Soren Hauberg (2019). "Reliable training and estimation of variance networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 6326–6336.

Snelson, Edward and Zoubin Ghahramani (2005). "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc.

Sø nderby, Casper Kaae, Tapani Raiko, Lars Maalø e, Søren Kaae Sø nderby, and Ole Winther (2016). "Ladder Variational Autoencoders". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc.

Sohl-Dickstein, Jascha, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli (2015). "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 2256–2265.

Sohn, Kihyuk, Honglak Lee, and Xinchen Yan (2015). "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc.

Song, Yang and Stefano Ermon (2019). "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Song, Yang, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Er- mon, and Ben Poole (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*.

Springenberg, Jost Tobias, Aaron Klein, Stefan Falkner, and Frank Hutter (2016). "Bayesian Optimization with Robust Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., pp. 4134– 4142.

Srinivas, Niranjan, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger (2010). "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*. Haifa, Israel: Omnipress, pp. 1015–1022.

Stimper, Vincent, David Liu, Andrew Campbell, Vincent Berenz, Lukas Ryll, Bernhard Schölkopf, and José Miguel Hernández-Lobato (2023). "normflows: A PyTorch Package for Normalizing Flows". In: *arXiv preprint arXiv:2302.12014*.

Sun, Shengyang, Guodong Zhang, Jiaxin Shi, and Roger Grosse (2019). "Functional Variational Bayesian Neural Networks". In: *International Conference on Learning Representations*.

Teixeira, Leonardo, Brian Jalaian, and Bruno Ribeiro (2019). "Are Graph Neural Networks Miscalibrated?" In: *ICML Workshop on Learning and Reasoning with Graph- Structured*.

Tempczyk, Piotr, Rafał Michaluk, Lukasz Garncarek, Przemysław Spurek, Jacek Tabor, and Adam Golinski (2022). "LIDL: Local Intrinsic Dimension Estimation Using Approximate Likelihood". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 21205–21231.

Tenenbaum, Joshua B, Vin de Silva, and John C Langford (2000). "A Global Geo- metric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500, pp. 2319–2323.

Tieleman, T. and G. Hinton (2012). *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning.

Tishby, Levin, and Solla (1989). "Consistent Inference of Probabilities in Layered Networks: Predictions and Generalizations". In: *International Joint Conference on Neural Networks*, 403–409 vol.2.

Titsias, Michalis (2009). "Variational Learning of Inducing Variables in Sparse Gaus- sian Processes". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, pp. 567–574.

Titsias, Michalis K. and Francisco Ruiz (2019). "Unbiased Implicit Variational Inference". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 167–176.

Tolstikhin, Ilya, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf (2018). "Wasserstein Auto-Encoders". In: *International Conference on Learning Representations*.

Tomczak, Jakub M (2022). "Deep Generative Modeling". In: *Deep Generative Modeling*. Springer, pp. 1–12.

Tomczak, Jakub M. and Max Welling (2018). "VAE with a VampPrior". In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 1214–1223.

Tran, Ba-Hien, Simone Rossi, Dimitrios Milios, Pietro Michiardi, Edwin V Bonilla, and Maurizio Filippone (2021). "Model Selection for Bayesian Autoencoders". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 19730–19742.

Tran, Ba-Hien, Simone Rossi, Dimitrios Milios, and Maurizio Filippone (2022). "All You Need is a Good Functional Prior for Bayesian Deep Learning". In: *Journal of Machine Learning Research* 23.74, pp. 1–56.

Trinh, Trung Q, Markus Heinonen, Luigi Acerbi, and Samuel Kaski (2022). "Tackling Covariate Shift with Node-based Bayesian Neural Networks". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 21751–21775.

Uria, Benigno, Iain Murray, and Hugo Larochelle (2013). "RNADE: The Real-valued Neural autoregressive Density-estimator". In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc.

Vahdat, Arash and Jan Kautz (2020). "NVAE: A Deep Hierarchical Variational Autoencoder". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 19667–19679.

van der Wilk, Mark, Carl Edward Rasmussen, and James Hensman (2017). "Convolutional Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

van der Wilk, Mark, Matthias Bauer, ST John, and James Hensman (2018). "Learning Invariances using the Marginal Likelihood". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Villani, C. (2003). *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society.

Villani, Cédric (2008). *Optimal Transport: Old and New*. Vol. 338. Springer Science & Business Media.

Vincent, Pascal (2011). "A Connection Between Score Matching and Denoising Autoencoders". In: *Neural Computation* 23, pp. 1661–1674.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol (2008). "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. Helsinki, Finland, pp. 1096–1103.

Wan, Neng, Dapeng Li, and Naira Hovakimyan (2020). "f-Divergence Variational Inference". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 17370–17379.

Wang, Dilin and Qiang Liu (2016). "Learning to Draw Samples: With Application to Amortized MLE for Generative Adversarial Learning". In: *arXiv preprint arXiv:1611.01722*.

Wang, Kuan-Chieh, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel (2018). "Adversarial Distillation of Bayesian Neural Network Posteriors". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5190–5199.

Wang, Xiao, Hongrui Liu, Chuan Shi, and Cheng Yang (2021). "Be Confident! Towards Trustworthy Graph Neural Networks via Confidence Calibration". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 23768–23779.

Wang, Zhendong, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou (2023). "Diffusion-GAN: Training GANs with Diffusion". In: *International Conference on Learning Representations*.

Wang, Zi, George E. Dahl, Kevin Swersky, Chansoo Lee, Zelda Mariet, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani (2022). "Pre-training Helps Bayesian Optimization Too". In: *ICML Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

Wenzel, Florian, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin (2020). "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *Proceeding of the 37th International Conference on Machine Learning, ICML 2020*. Virtual.

Wilson, Andrew G and Pavel Izmailov (2020). "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 4697–4708.

Wolpert, David H (1996). "The Lack of a Priori Distinctions Between Learning Algorithms". In: *Neural computation* 8.7, pp. 1341–1390.

Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat (2022). "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs". In: *International Conference on Learning Representations*.

Yang, Wanqian, Lars Lorch, Moritz A. Graule, Srivatsan Srinivasan, Anirudh Suresh, Jiayu Yao, Melanie F. Pradier, and Finale Doshi-velez (2019). "Output-Constrained Bayesian Neural Networks". In: *ICML workshop on Uncertainty & Robustness in Deep Learning*.

Yang, Wanqian, Lars Lorch, Moritz A. Graule, Himabindu Lakkaraju, and Finale Doshi-Velez (2020). "Incorporating Interpretable Output Constraints in Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yang, Yibo, Stephan Mandt, and Lucas Theis (2022). "An Introduction to Neural Data Compression". In: *arXiv preprint arXiv:2202.06533*.

Yao, Yuan, Lorenzo Rosasco, and Andrea Caponnetto (2007). "On Early Stopping in Gradient Descent Learning". In: *Constructive Approximation* 26.2, pp. 289–315.

Zeno, Chen, Itay Golan, Ari Pakman, and Daniel Soudry (2021). "Why Cold Posteriors? On the Suboptimal Generalization of Optimal Bayes Estimates". In: *Third Symposium on Advances in Approximate Bayesian Inference*.

Zhang, Cheng, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt (2018a). "Advances in Variational Inference". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8, pp. 2008–2026.

Zhang, Guodong, Shengyang Sun, David Duvenaud, and Roger B. Grosse (2018b). "Noisy Natural Gradient as Variational Inference". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5847–5856.

Zhang, Hao, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum (2023). "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection". In: *International Conference on Learning Representations*.

Zhang, Ruqi, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson (2020). "Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning". In: *International Conference on Learning Representations*.

Zhao, Shengyu, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han (2020). "Differentiable Augmentation for Data-Efficient GAN Training". In: *Advances in Neural Information*

Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Zhu, Harrison, Carles Balsells Rodas, and Yingzhen Li (2022). "Markovian Gaussian Process Variational Autoencoders". In: *arXiv preprint arXiv:2207.05543.*