All You Need is a Good Functional Prior for Bayesian Deep Learning

Ba-Hien Tran   BA-HIEN.TRAN@EURECOM.FR
Simone Rossi   SIMONE.ROSSI@EURECOM.FR
Dimitrios Milios   DIMITRIOS.MILIOS@EURECOM.FR
Maurizio Filippone   MAURIZIO.FILIPPONE@EURECOM.FR
Data Science Department
EURECOM
Sophia Antipolis, FR

Editor: Mohammad Emtiyaz Khan

Abstract

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 this work 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.

Keywords: neural networks, Bayesian inference, Gaussian processes, Wasserstein distance, prior distribution

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., 2019; 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 Figure 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 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
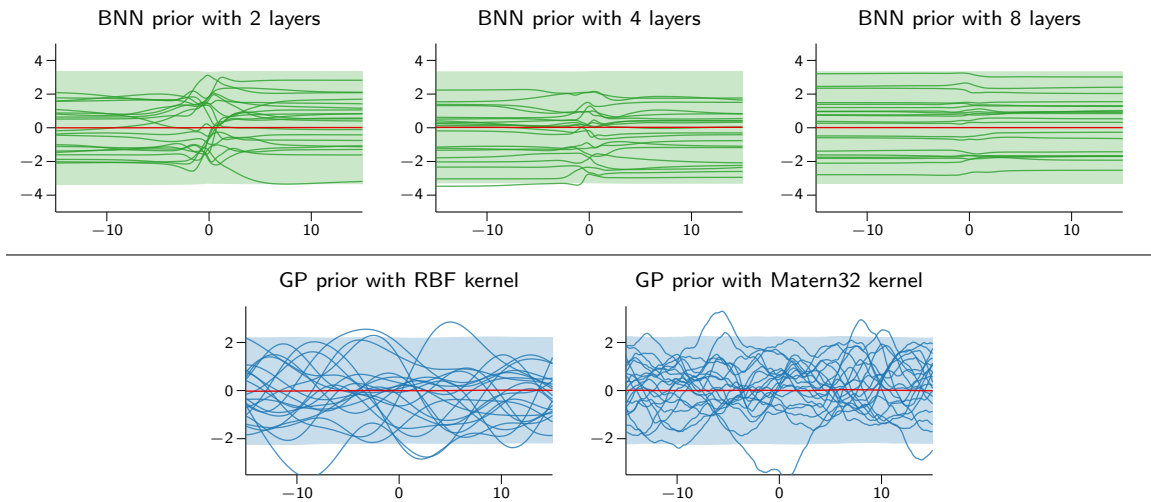
Figure 1: (**Top**) Sample functions of a fully-connected BNN with 2, 4 and 8 layers obtained by placing a Gaussian prior on the weights. (**Bottom**) Samples from a GP prior with two different kernels.

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 hyper-parameters. 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.

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 (2019, 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. (2021a) 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. (2021) 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.

(2021b) 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 paper, 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. Preliminaries

In this section, we establish some basic notation on BNNs that we follow throughout the paper, and we review stochastic gradient Hamiltonian Monte Carlo (SGHMC), which is the main sampling algorithm that we use in our experiments. Finally, 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.

3.1 Bayesian Neural Networks

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

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

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 $\mathbf{w}_l = \{W_l, b_l\}$, while the entirety of trainable network parameters will be denoted as $\mathbf{w} = \{\mathbf{w}_l\}_{l=1}^L$. In order to simplify the presentation, we focus on fully-connected DNNs; the weight and bias parameters of 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.

Inference   The Bayesian treatment of neural networks (MacKay, 1992; Neal, 1996) dictates that a prior distribution $p(\mathbf{w})$ is placed over the parameters. The learning problem is formulated as a transformation of a prior belief into a posterior distribution by means of Bayes' theorem. Given a dataset with N input-target pairs $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} \overset{\text{def}}{=} \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, the posterior over $\mathbf{w}$ is:

$$p(\mathbf{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}. \tag{2}$$

Although the posterior for most nonlinear models, such as neural networks, is analytically intractable, it can be approximated by MCMC methods, as they only require an unnormalized version of the target density. Regarding Equation (2), the unnormalized posterior density is given by the joint probability in the numerator, which can be readily evaluated if the prior and likelihood densities are known.

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987) considers the joint log-likelihood as a potential energy function $U(\mathbf{w}) = -\log p(\mathcal{D} \mid \mathbf{w}) - \log p(\mathbf{w})$, and introduces a set of auxiliary momentum variables $\mathbf{r}$. Samples are generated from the joint distribution $p(\mathbf{w}, \mathbf{r})$ based on the Hamiltonian dynamics:

$$d\mathbf{w} = \mathbf{M}^{-1}\mathbf{r}dt, \tag{3}$$

$$d\mathbf{r} = -\nabla U(\mathbf{w})dt, \tag{4}$$

where, $\mathbf{M}$ is an arbitrary mass matrix that plays the role of a preconditioner. In practice, this continuous system is approximated by means of a ε-discretized numerical integration and followed by Metropolis steps to accommodate numerical errors stemming from the integration.

However, HMC is not practical for large datasets due to the cost of computing the gradient $\nabla U(\mathbf{w}) = \nabla \log p(\mathcal{D}|\mathbf{w})$ on the entire dataset. To mitigate this issue, Chen et al. (2014) proposed SGHMC, which considers a noisy, unbiased estimate of the gradient $\nabla \tilde{U}(\mathbf{w})$

computed from a mini-batch of the data. The discretized Hamiltonian dynamics equations are then updated as follows

$$\Delta \mathbf{w} = \varepsilon \mathbf{M}^{-1} \mathbf{r}, \tag{5}$$

$$\Delta \mathbf{r} = -\varepsilon \nabla \tilde{U}(\mathbf{w}) - \varepsilon \mathbf{C} \mathbf{M}^{-1} \mathbf{r} + \mathcal{N}(0, 2\varepsilon(\mathbf{C} - \tilde{\mathbf{B}})), \tag{6}$$

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

In this work, we employ the SGHMC algorithm to generate posterior samples for all the models and datasets considered. The step size ε as well as the matrices $\mathbf{M}$, $\mathbf{C}$ and $\tilde{\mathbf{B}}$ constitute additional parameters that require careful tuning to guarantee the quality of samples produced by the algorithm. We adopt the tuning strategy of Springenberg et al. (2016), which involves a burn-in period during which the matrices $\mathbf{M}$ and $\tilde{\mathbf{B}}$ are adjusted by monitoring certain statistics of the dynamics. The only parameters that we manually define are the integration interval and the step size.

3.2 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 $\mu(\mathbf{x})$ and a covariance function $\kappa(\mathbf{x}, \mathbf{x}')$. The latter is also known as a kernel function, and it determines the covariance between the realization of the function at pairs of inputs $\mathbf{x}$ and $\mathbf{x}'$. For a finite set of inputs $\mathbf{X}$, a GP yields a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu} = \mu(\mathbf{X})$ and covariance matrix $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$.

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:

$$\kappa_{\alpha,l}(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}')}{l^2} \right), \tag{7}$$

which induces functions that are infinitely differentiable, as in Figure 1. The subscripts α, l denote the dependency on hyper-parameters: 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 hyper-parameters 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 $\kappa_{\alpha,l}(\mathbf{x}, \mathbf{x}')$ with hyper-parameters α and l, we have:

$$\alpha, l \sim \text{LogNormal}(m, s^2), \qquad f \sim \mathcal{N}(0, \kappa_{\alpha,l}(\mathbf{x}, \mathbf{x}')) \tag{8}$$

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 hyper-parameters 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 § 6, unless otherwise specified.

3.3 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* $\pi(\mathbf{x})$ and $\nu(\mathbf{y})$ 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:

$$W_p(\pi, \nu) = \left(\inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} D(\mathbf{x}, \mathbf{y})^p \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} \right)^{1/p}, \tag{9}$$

where $D(\mathbf{x}, \mathbf{y})$ is a proper distance metric between two points $\mathbf{x}$ and $\mathbf{y}$ 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 $\mathbf{x}$ and $\mathbf{y}$ coincide (i.e. $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subseteq \mathbb{R}^d$), with $D(\mathbf{x}, \mathbf{y})$ being the Euclidian norm distance, the Wasserstein-1 distance (also known in the literature as Earth-Mover distance) takes the following shape,

$$W_1(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|\mathbf{x} - \mathbf{y}\| \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y}. \tag{10}$$

With the exception of few cases where the solution is available analytically (e.g. π and ν being Gaussians), solving Equation (10) directly or via optimization is intractable. On the other hand, the Wasserstein distance defined in Equation (10) admits the following dual form (Kantorovich, 1942, 1948),

$$\begin{aligned} W_1(\pi, \nu) &= \sup_{\|\phi\|_L \leq 1} \left[\int \phi(\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x} - \int \phi(\mathbf{y}) \nu(\mathbf{y}) d\mathbf{y} \right] \\ &= \sup_{\|\phi\|_L \leq 1} \mathbb{E}_\pi \phi(\mathbf{x}) - \mathbb{E}_\nu \phi(\mathbf{x}), \end{aligned} \tag{11}$$

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.

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.

4.1 Wasserstein Distance Optimization

Assume a prior distribution $p(\mathbf{w}; \boldsymbol{\psi})$ on the weights of a BNN, where $\boldsymbol{\psi}$ is a set of parameters that determine the prior (e.g., $\boldsymbol{\psi} = \{\mu, \sigma\}$ 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:

$$p_{nn}(\mathbf{f}; \boldsymbol{\psi}) = \int p(\mathbf{f} \mid \mathbf{w}) p(\mathbf{w}; \boldsymbol{\psi}) \, \mathrm{d}\mathbf{w}, \tag{12}$$

where $p(\mathbf{f} \mid \mathbf{w})$ 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 Equation (8). Our target GP prior is $p_{gp}(\mathbf{f} \mid \mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is the covariance matrix obtained by computing the kernel function κ for each pair of $\{\mathbf{x}_i, \mathbf{x}_j\}$ in the training set. We aim at matching these two stochastic processes at a finite number of measurement points $\mathbf{X}_{\mathcal{M}} \stackrel{\text{def}}{=} [\mathbf{x}_1, ..., \mathbf{x}_M]^\top$ sampled from a distribution $q(\mathbf{x})$. To achieve this, we propose a sample-based approach using the 1-Wasserstein distance in Equation (11) as objective:

$$\min_{\boldsymbol{\psi}} \max_{\boldsymbol{\theta}} \mathbb{E}_q \Big[\underbrace{\mathbb{E}_{p_{gp}} [\phi_{\boldsymbol{\theta}}(\mathbf{f}_{\mathcal{M}})] - \mathbb{E}_{p_{nn}} [\phi_{\boldsymbol{\theta}}(\mathbf{f}_{\mathcal{M}})]}_{\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\theta})} \Big], \tag{13}$$

where $\mathbf{f}_{\mathcal{M}}$ denotes the set of random variables associated with the inputs at $\mathbf{X}_{\mathcal{M}}$, and $\phi_{\boldsymbol{\theta}}$ 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 $\boldsymbol{\theta}$.

Regarding the optimization of the $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ parameters we alternate between $n_{\text{Lipschitz}}$ steps of maximizing $\mathcal{L}$ with respect to the Lipschitz function's parameters $\boldsymbol{\theta}$ and one step of minimizing the Wasserstein distance with respect to the prior's parameters $\boldsymbol{\psi}$. We therefore use two independent optimizers (RMSprop–see, for example, Tieleman and Hinton, 2012) for $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$. Figure 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 Equation (13), 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 $p_{nn}(\mathbf{f}; \boldsymbol{\psi})$. 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

---

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$(x) = 1/(1 + \exp(-x))$.
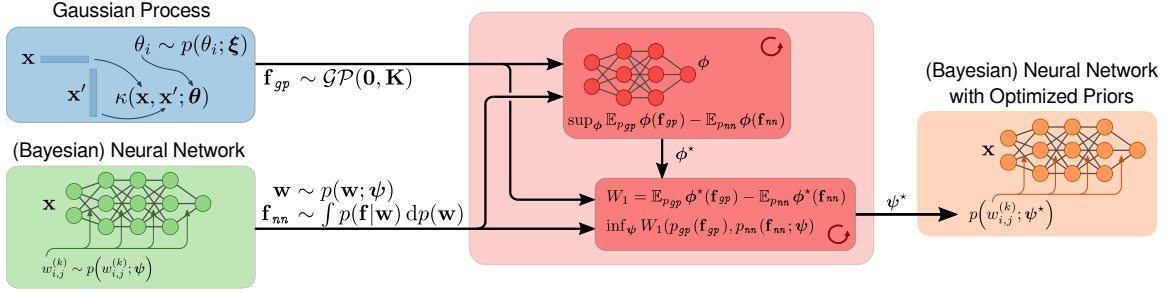
Figure 2: Schematic representation of the process of imposing GP priors on BNNs via Wasserstein distance minimization.

acknowledge that the two training steps could have been optimized jointly in a single loop, as Equation (11) defines a minimax problem. However, this choice allows $\phi_{\boldsymbol{\theta}}$ 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 B.6 we further discuss this choice and we show qualitatively the convergence improvements.

Lipschitz constraint. In order to enforce the Lipschitz constraint on $\phi_{\boldsymbol{\theta}}$, Arjovsky et al. (2017) propose to clip the weights $\boldsymbol{\theta}$ to lie within a compact space $[-c, c]$ such that all functions $\phi_{\boldsymbol{\theta}}$ are K-Lipschitz. This approach usually biases the resulting $\phi_{\boldsymbol{\theta}}$ 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 $\phi_{\boldsymbol{\theta}}$ with respect to its input. More specifically, the loss of the Lipschitz function is augmented by a regularization term

$$\mathcal{L}_R(\boldsymbol{\psi}, \boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\theta}) + \underbrace{\lambda \mathbb{E}_{p_{\hat{\mathbf{f}}}}\left[\left(\left\|\boldsymbol{\nabla}_{\hat{\mathbf{f}}}\phi(\hat{\mathbf{f}})\right\|_2 - 1\right)^2\right]}_{\text{Gradient penalty}}. \tag{14}$$

Here $p_{\hat{\mathbf{f}}}$ is the distribution of $\hat{\mathbf{f}} = \varepsilon \mathbf{f}_{nn} + (1 - \varepsilon)\mathbf{f}_{gp}$ for $\varepsilon \sim \mathcal{U}[0, 1]$ and $\mathbf{f}_{nn} \sim p_{nn}$, $\mathbf{f}_{gp} \sim p_{gp}$ 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 $q(\mathbf{x})$ 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.

4.2 Prior Parameterization for Neural Networks

In the previous section, we have treated the parameters of a BNN prior $p_{nn}(\mathbf{f}; \boldsymbol{\psi})$ 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 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 $\boldsymbol{\psi} = \{\sigma_{l_\mathrm{w}}^2, \sigma_{l_\mathrm{b}}^2\}_{l=1}^L$, where $\sigma_{l_\mathrm{w}}^2$ is the prior variance shared across all weights in layer l, and $\sigma_{l_\mathrm{b}}^2$ is the respective variance for the bias parameters. For any weight and bias entries $w_l, b_l \in \mathbf{w}_l$ of the l-th layer, the prior is:

$$p(w_l) = \mathcal{N}\left(w_l;\ 0, \sigma_{l_\mathrm{w}}^2\right) \quad \text{and} \quad p(b_l) = \mathcal{N}\left(b_l;\ 0, \sigma_{l_\mathrm{b}}^2\right).$$

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 $w_l \sim \mathcal{N}\left(w_l;\ 0, \sigma_{l_\mathrm{w}}^2\right)$, we use the reparameterization trick (Rezende et al., 2014; Kingma and Welling, 2014), and sample from the noise distribution instead,

$$w_l := \sigma_{l_\mathrm{w}}\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1). \tag{15}$$

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:

$$\sigma_{l_\mathrm{w}}^2 \sim \Gamma^{-1}(\alpha_{l_\mathrm{w}}, \beta_{l_\mathrm{w}}) \quad \text{and} \quad \sigma_{l_\mathrm{b}}^2 \sim \Gamma^{-1}(\alpha_{l_\mathrm{b}}, \beta_{l_\mathrm{b}})$$

In this case, we have $\boldsymbol{\psi} = \{\alpha_{l_\mathrm{w}}, \beta_{l_\mathrm{w}}, \alpha_{l_\mathrm{b}}, \beta_{l_\mathrm{b}}\}_{l=1}^L$, where $\alpha_{l_\mathrm{w}}, \beta_{l_\mathrm{w}}, \alpha_{l_\mathrm{b}}, \beta_{l_\mathrm{b}}$ 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 $p_0(\mathbf{w}_l)$, by denoting $p(\mathcal{T}_K(\mathbf{w}_l))$ the final distribution, its log-density can be analytically computed by taking into account to Jacobian of the transformations as follows,

$$\log p(\mathcal{T}_K(\mathbf{w}_l)) = \log p_0(\mathbf{w}_l) - \sum_{k=1}^{K} \log \left| \det \frac{\partial t_k(\mathbf{w}_{l_{k-1}})}{\partial \mathbf{w}_{l_{k-1}}} \right|, \tag{16}$$

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

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 $p_0(\mathbf{w}_l)$ to a fully-factorized Gaussian $\mathcal{N}(\mathbf{w}_l \,|\, \mathbf{0}, \boldsymbol{\sigma}_l^2 \mathbf{I})$. We then employ a sequence of four *planar flows* (Rezende and Mohamed, 2015), each defined as

$$t_k(\mathbf{w}_{l_{k-1}}) = \mathbf{w}_{l_{k-1}} + \mathbf{u}_{l_k} h(\boldsymbol{\theta}_{l_k}^\top \mathbf{w}_{l_{k-1}} + b_{l_k}), \tag{17}$$

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

$$\log \left| \det \frac{\partial t_k(\mathbf{w}_{l_{k-1}})}{\partial \mathbf{w}_{l_{k-1}}} \right| = \log \left| 1 + \mathbf{u}_{l_k}^\top \boldsymbol{\theta}_{l_k} h'(\boldsymbol{\theta}_{l_k}^\top \mathbf{w}_{l_{k-1}} + b_{l_k}) \right|. \tag{18}$$

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

4.3 Algorithm and Complexity

Algorithm 1 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 $\mathbf{X}_\mathcal{M}$.

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

---

Algorithm 1: Wasserstein Distance Optimization

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

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

    draw $\mathbf{X}_{\mathcal{M}}$ from $q(\mathbf{x})$ // Sample measurement set ;

    for $t = 1, ..., n_{\text{Lipschitz}}$ **do**

        draw GP functions $\{\mathbf{f}_{gp}^{(i)}\}_{i=1}^{N_s} \sim p_{gp}(\mathbf{f}; \kappa)$ at $\mathbf{X}_{\mathcal{M}}$;

        draw NN functions $\{\mathbf{f}_{nn}^{(i)}\}_{i=1}^{N_s} \sim p_{nn}(\mathbf{f}; \boldsymbol{\psi})$ at $\mathbf{X}_{\mathcal{M}}$;

        $\mathcal{L}_R = N_s^{-1} \sum_{i=1}^{N_s} \mathcal{L}_R^{(i)}$ // Compute Lipschitz objective $\mathcal{L}_R$ using Equation (14) ;

        $\boldsymbol{\theta} \leftarrow \text{Optimizer}(\boldsymbol{\theta}, \boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathcal{L}_R)$ // Update Lipschitz function $\phi_{\boldsymbol{\theta}}$;

    end

    draw GP functions $\{\mathbf{f}_{gp}^{(i)}\}_{i=1}^{N_s} \sim p_{gp}(\mathbf{f}; \kappa)$ at $\mathbf{X}_{\mathcal{M}}$;

    draw NN functions $\{\mathbf{f}_{nn}^{(i)}\}_{i=1}^{N_s} \sim p_{nn}(\mathbf{f}; \boldsymbol{\psi})$ at $\mathbf{X}_{\mathcal{M}}$;

    $\widetilde{W}_1 = N_s^{-1} \sum_{i=1}^{N_s} \phi_{\boldsymbol{\theta}}\big(\mathbf{f}_{gp}^{(i)}\big) - \phi_{\boldsymbol{\theta}}\big(\mathbf{f}_{nn}^{(i)}\big)$ // Compute Wasserstein-1 distance using Equation (13) ;

    $\boldsymbol{\psi} \leftarrow \text{Optimizer}(\boldsymbol{\psi}, \boldsymbol{\nabla}_{\boldsymbol{\psi}} \widetilde{W}_1)$ // Update prior p_{nn} ;

end

---

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 1, 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 § 4.2 (i.e. GPi-G and GPi-H), the set $\boldsymbol{\psi}$ grows sub-linrealy 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 $\mathcal{O}(K N_{\text{BNN}})$. 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.

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 (§ 5.1) and how the choice of GP priors (in terms of kernel parameters) affects the BNN posterior for 2D classification examples (§ 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 (§ 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 HMC obtaining similar results (see Table 11 in the Appendix).

5.1 Visualization on a 1D regression synthetic dataset

The dataset used is built as follows: (1) we uniformly sample 64 input locations $\mathbf{x}$ 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 $\mathbf{f}$ from the GP prior ($l = 0.6, \alpha = 1$) computed at locations $\mathbf{x}$; (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. Figure 3 shows all the results. The 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.

### 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 hyper-parameters of the RBF kernel. Figure 4 (left) illustrates the predictive posterior of a two-layer BNN, whose prior has been adapted to different target GP priors, featuring different hyper-parameters. We observe that the decision boundaries are more complex for smaller lengthscales l and larger amplitudes α,
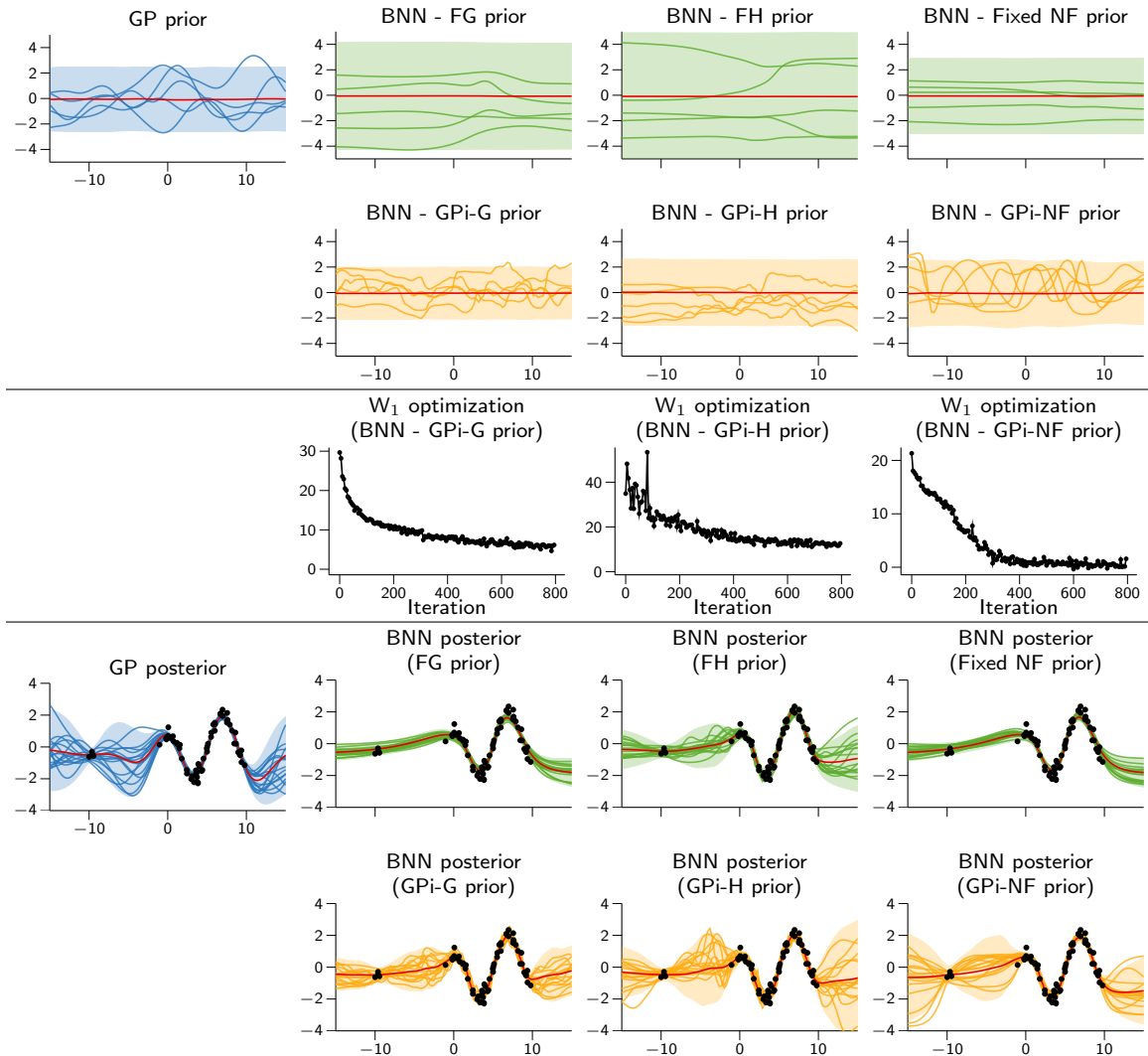
Figure 3: Visualization of one-dimensional regression example with a three-layer MLP. The first two rows illustrate the prior sample and distributions, whereas the last two rows show the corresponding posterior distributions. The means and the 95% credible intervals are represented by red lines and shaded areas, respectively. The middle row shows progressions of the prior optimization.

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 hyper-parameters 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 § 3.2, we approach this issue by means of hierarchical GPs. In the rightmost plot of Figure 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
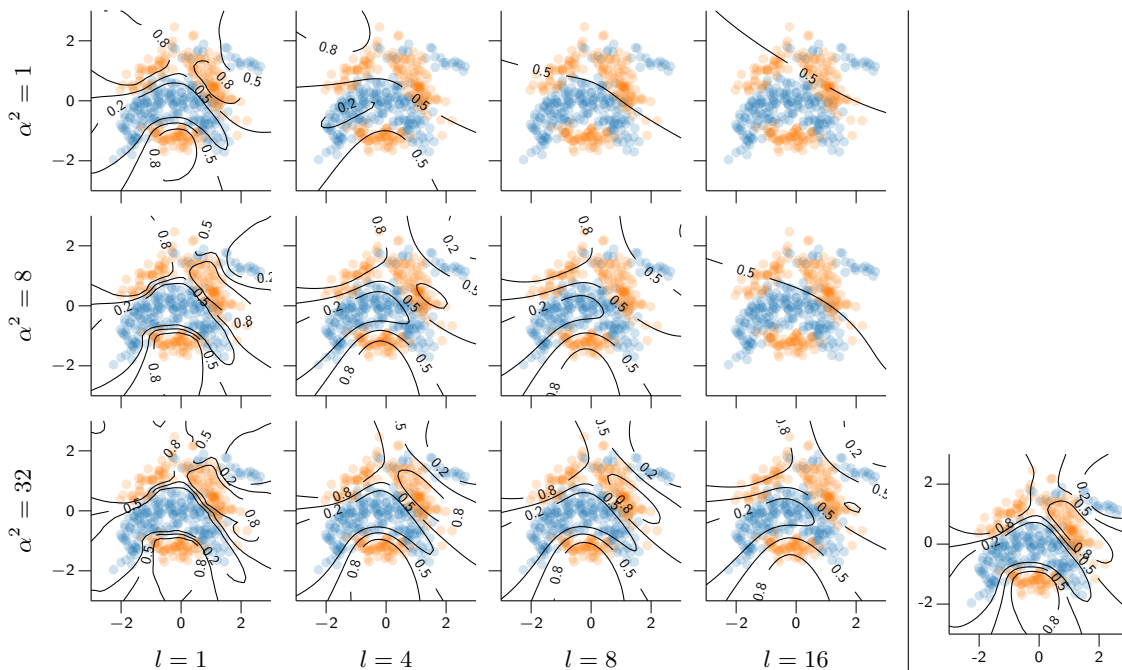
Figure 4: (Left) The effect of using different hyper-parameters of the RBF kernel of the target GP prior to the predictive posterior. Rows depict increasing the amplitude α, whist columns show increasing the lengthscale l. In each panel the orange and blue dots represent the training points from the 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.

can positively impact the expressiveness of the BNN posterior without explicitly worrying which GP prior works best.

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:

$$\text{KL}\left[p_{nn} \parallel p_{gp}\right] = -\int p_{nn}(\mathbf{f}; \boldsymbol{\psi}) \log p_{gp}(\mathbf{f}) \, d\mathbf{f} + \underbrace{\int p_{nn}(\mathbf{f}; \boldsymbol{\psi}) \log p_{nn}(\mathbf{f}; \boldsymbol{\psi}) \, d\mathbf{f}}_{\text{Entropy (intractable)}}, \qquad (19)$$

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., $-\int p_{nn}(\mathbf{f}; \boldsymbol{\psi}) \log p_{gp}(\mathbf{f}) \, d\mathbf{f}$). 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.
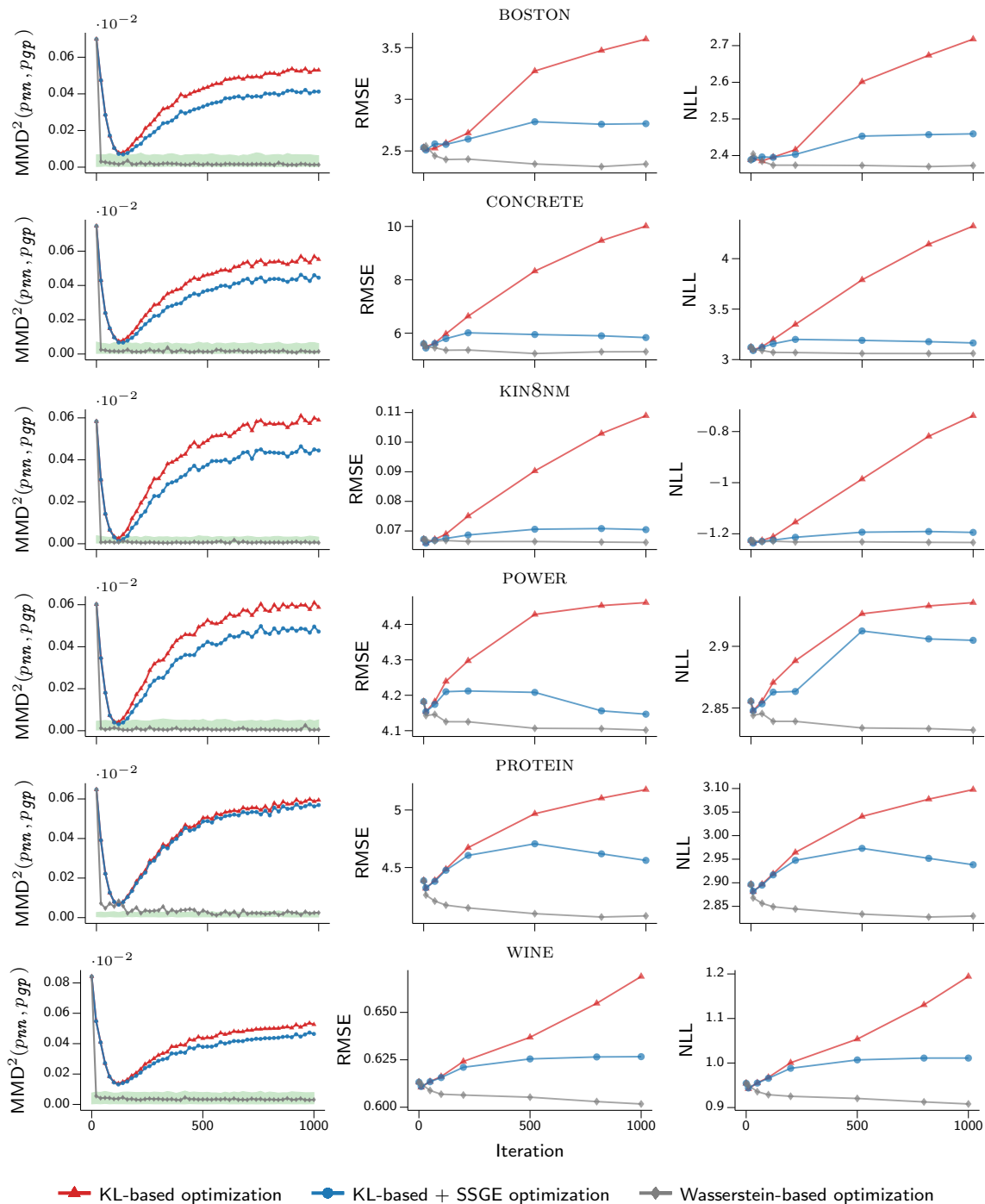
Figure 5: Comparison between KL-based and Wasserstein-based optimization. The green shaded area is for calibration and denotes the difference between the squared maximum mean discrepancy (MMD) of the target GP to itself and to another GP with a doubled lengthscale.

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 Figure 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 of the similarity between BNNs and GPs. A detailed description of estimating this discrepancy and experimental settings are available in Appendix A.8. 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 loglikelihood (NLL) and root mean square 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 B.3 we include a complete account on the convergence of Wasserstein distance for all experiments that follow in the next section.

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 § 6.1. We then investigate the effect of functional priors on classic UCI benchmark datasets for regression in § 6.2 and classification in § 6.3. Bayesian CNNs are explored in § 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 § 6.5. We then perform experiments on active learning (§ 6.6), where having good and calibrated estimates of uncertainty is critical to achieve fast convergence. Finally, we conclude in § 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.

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.

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

Table 1: Glossary of methods used in the experimental campaign. Here, $p(\mathbf{f}) = \int p(\mathbf{f} \mid \mathbf{w})\,\mathrm{d}p(\mathbf{w})$ 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. (2021a).

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

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., 2021a), 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 for implementation details and more detailed hyper-parameter settings. Table 1 presents an overview of the methods considered in the experiments.

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.
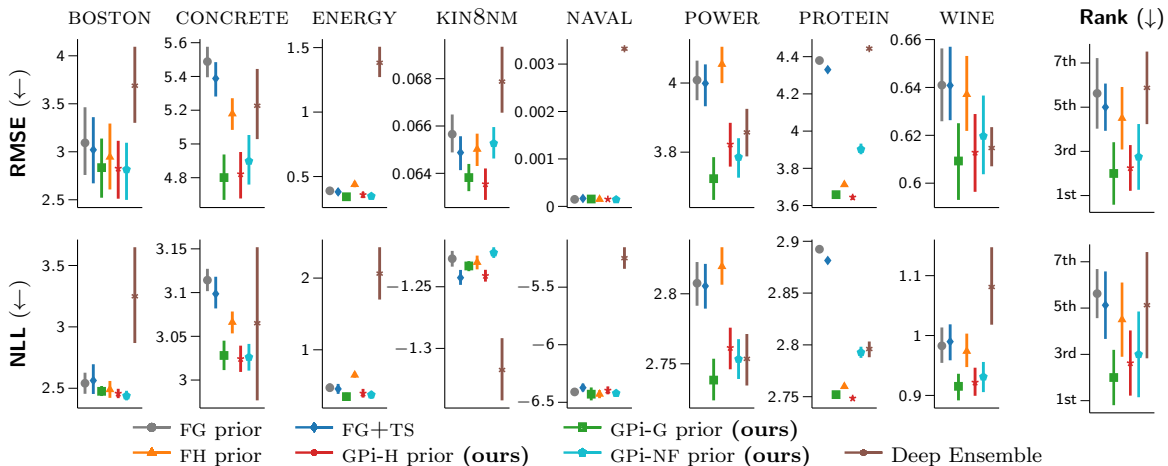
Figure 6: UCI regression benchmark results. The dots and error bars represent the means and standard errors over the test splits, respectively. Average ranks are computed across datasets.

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, 1996). 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.

Figure 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.

Ablation study on the model capacity. We further investigate the relation of the model capacity to the prior effect. Figure 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 § 6.4 with much more complex models such as CNNs.

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
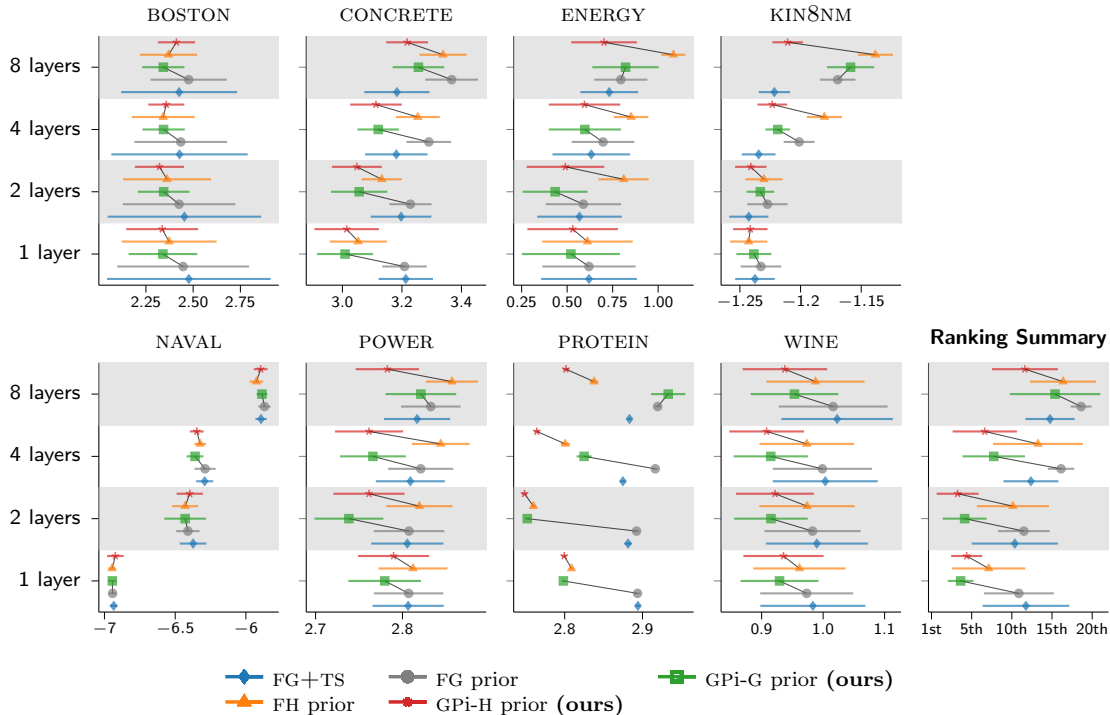
Figure 7: Ablation study on the test NLL based on the UCI regression benchmark for different number of hidden layers of MLP. Error bars represent one standard deviation. We connect the fixed and GP-induced priors with a thin black line as an aid for easier comparison. Further to the left is better.

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

Figure 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.

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
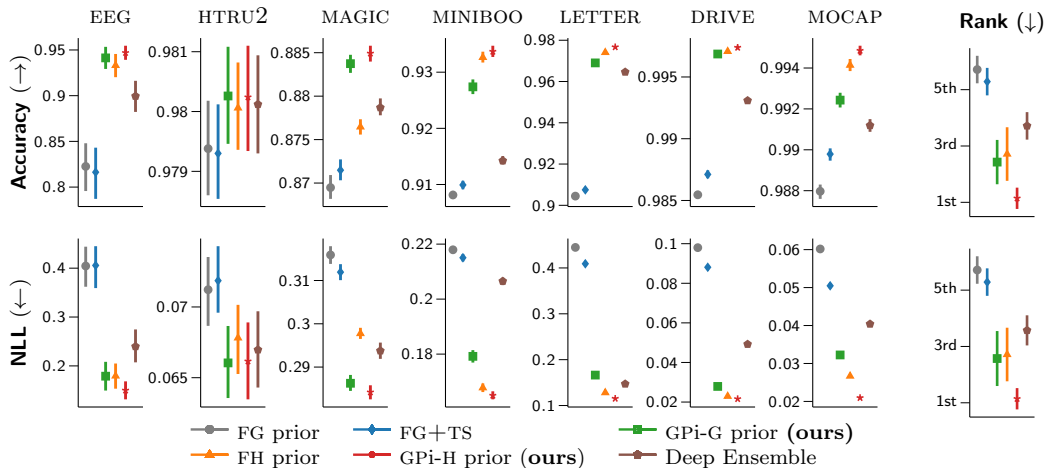
Figure 8: UCI classification benchmark results. The dots and error bars represent the means and standard errors over the test splits, respectively. Average ranks are computed across datasets.

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 Figure 9b). We also visualize the average prior distribution obtained from 200 samples of parameters (see the leftmost column of Figure 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 Figure 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 (Figure 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. Figure 9c and Figure 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, 2015) 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 $N_{\mathcal{M}} = 32$

(a) Target GP prior.

(b) BNN - FG prior.

(c) BNN - GPi-G prior.

(d) BNN - FH prior.
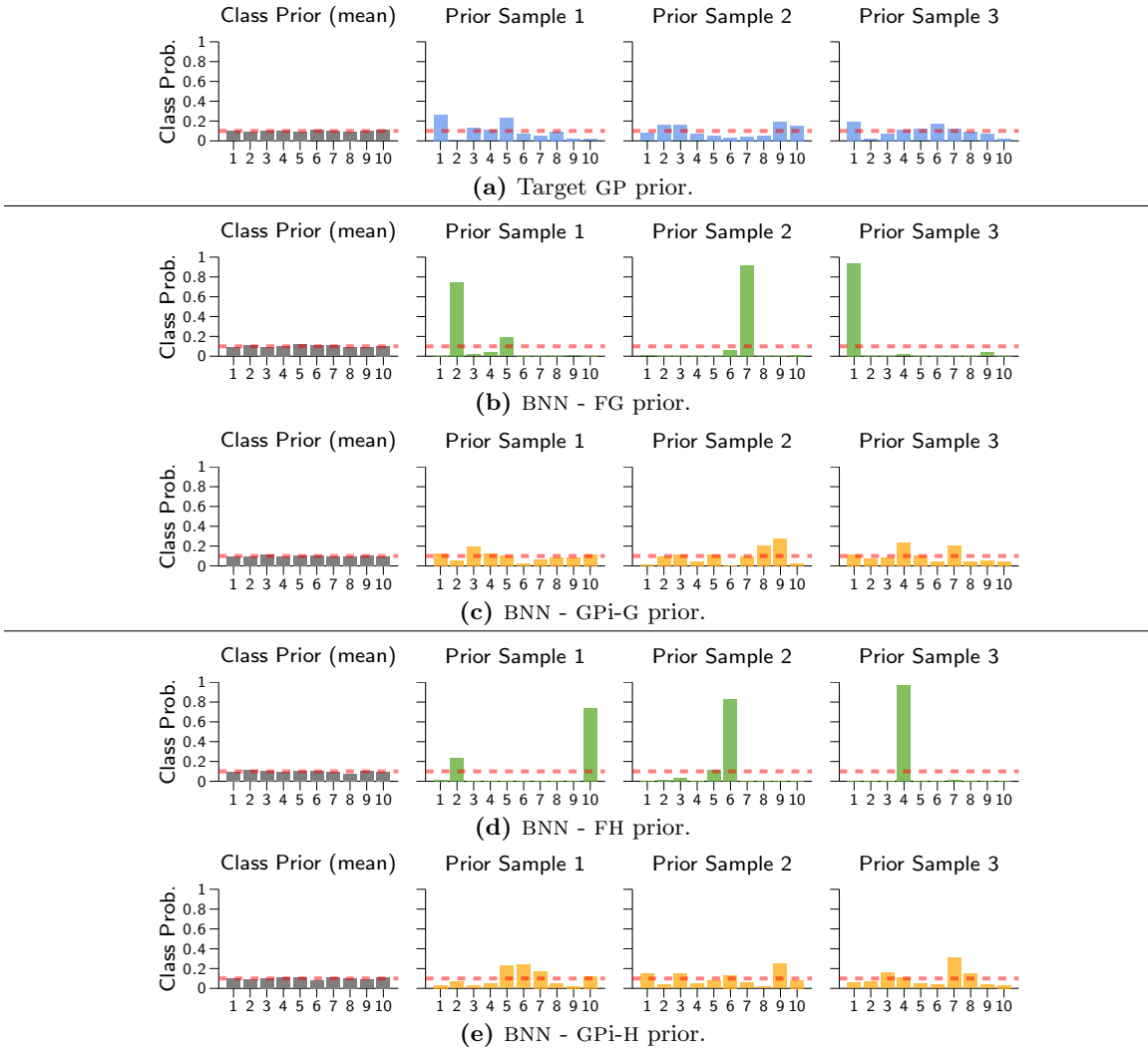
(e) BNN - GPi-H prior.

Figure 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.

measurement points sampled from the empirical distribution of the training data regarding prior optimization.

Table 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

Table 2: Results for different convolutional neural networks on the CIFAR10 dataset (errors are ± 1 standard error computed over 4 running times).

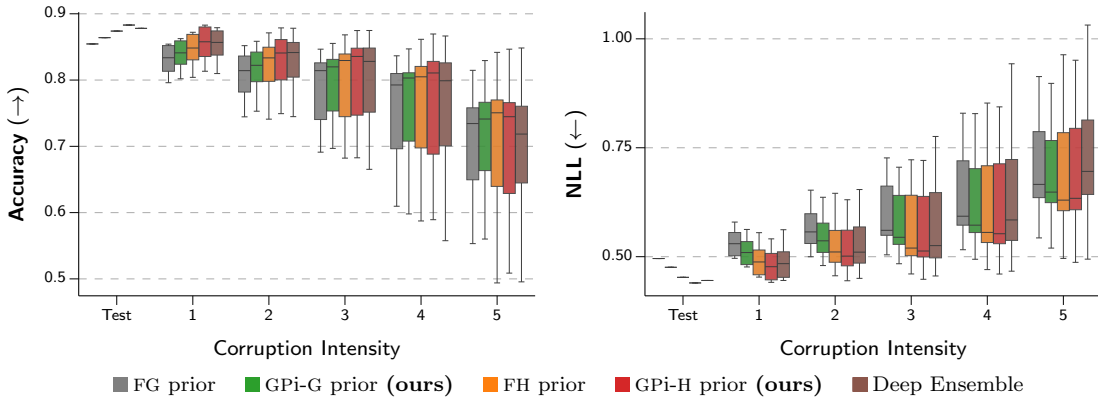| Architecture	Method	Accuracy - % ($\uparrow$)	NLL ($\downarrow$)
LENET5	Deep Ensemble	71.13 ± 0.10	0.8548 ± 0.0010
	FG prior	74.65 ± 0.25	0.7482 ± 0.0025
	FG+TS	74.08 ± 0.24	0.7558 ± 0.0024
	GPi-G prior (**ours**)	75.15 ± 0.24	0.7360 ± 0.0024
	FH prior	75.22 ± 0.40	0.7209 ± 0.0040
	GPi-H prior (**ours**)	$\mathbf{76.51} \pm \mathbf{0.21}$	$\mathbf{0.6952} \pm \mathbf{0.0021}$
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**)	$\mathbf{87.03} \pm \mathbf{0.07}$	$\mathbf{0.4127} \pm \mathbf{0.0007}$

LENET5, the predictive performance dramatically degraded when using temperature scaling. In addition to the results in § 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.

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 Figure 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

Figure 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.
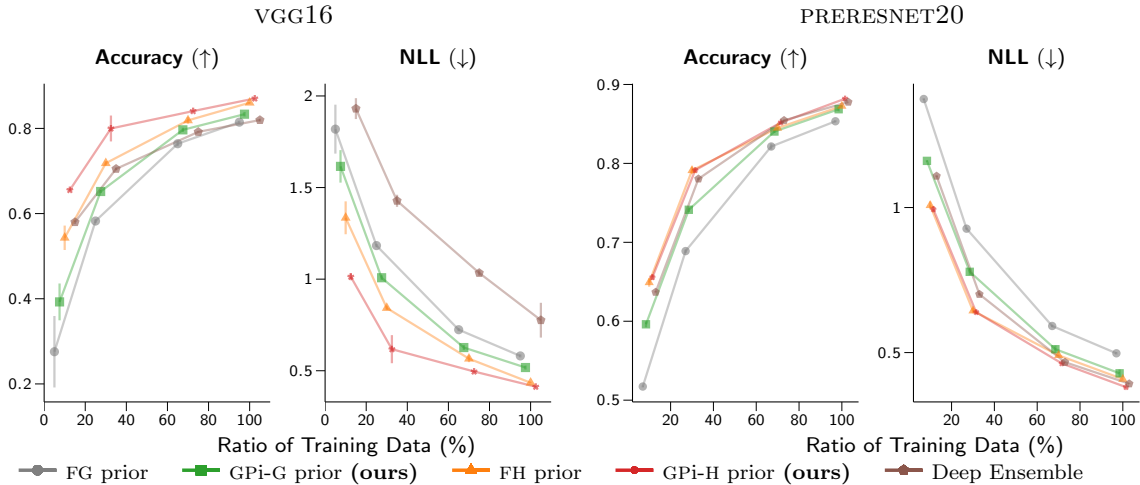

Figure 11: Accuracy and NLL on CIFAR10 at varying the training set's size. The bars indicate one standard error.

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. Figure 11 shows the accuracy and NLL on the test 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
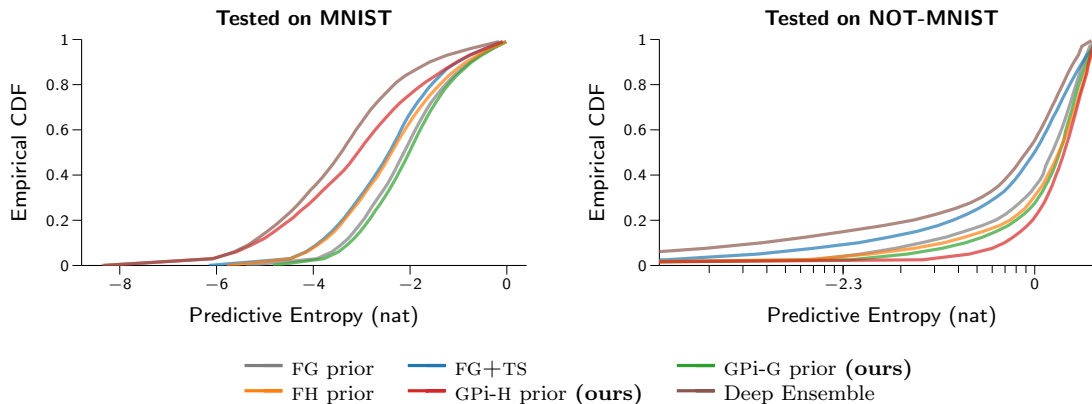
Figure 12: Cumulative distribution function plot of predictive entropies when the models trained on MNIST are tested on MNIST (left, the higher the better) and NOT-MNIST (right, the lower the better).

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.

Figure 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.

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 the marginal likelihood $p(\mathcal{D}; \psi) = \int p(\mathcal{D} \mid \mathbf{w}) p(\mathbf{w}; \psi) \, \mathrm{d}\mathbf{w}$ w.r.t. the prior parameters. However, these methods

---

2. NOT-MNIST dataset is available at `http://yaroslavvb.blogspot.fr/2011/09/notmnist-dataset.html`.
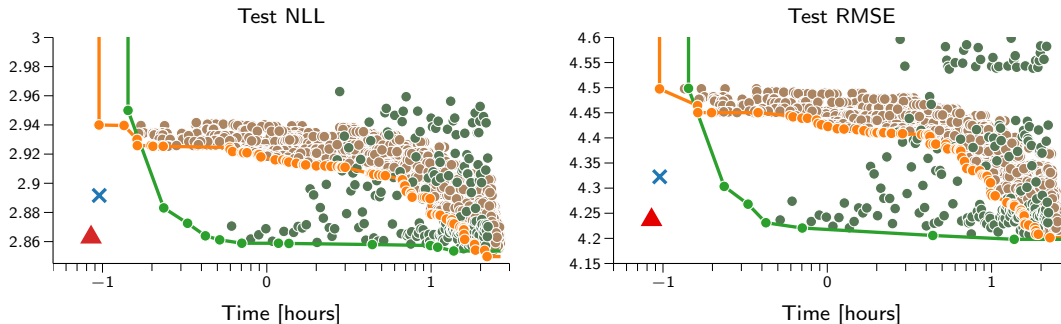
Figure 13: A timing comparison between imposing functional prior and cross-validation with either grid-search (▬) or Bayesian optimization (▬). In the plots, each ● corresponds to a run of a single configuration, while ━●━ highlights the Pareto front of the cross validation procedure. The figure also reports the $\mathcal{N}(0,1)$ prior as ✕, while ▲ is our proposal of using functional prior (GPi-G).

present significant challenges: (i) for CV, the number of hyper-parameter 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 § 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 Figure 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.
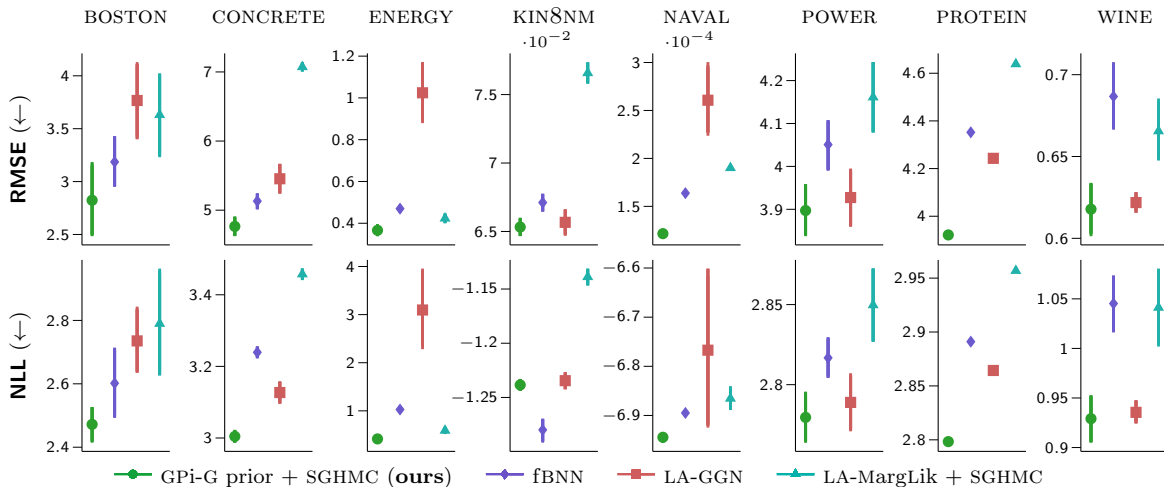
Figure 14: Comparison with empirical Bayes and functional inference approaches on the UCI regression datasets. The dots and error bars represent the means and standard errors over the test splits, respectively.

The hyper-parameters 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. (2021a). 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.9. In Figure 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 Figure 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 B.4.

From a more philosophical point of view, it is worth noting that cross-validating prior 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.
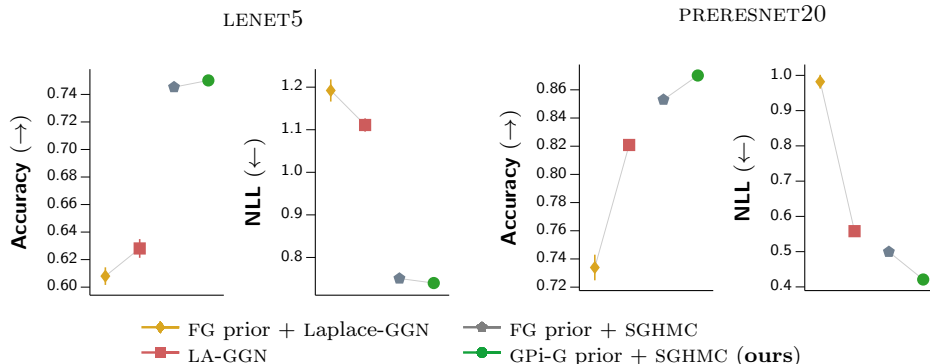
Figure 15: Comparison with empirical Bayes and functional inference approaches on the CIFAR10 dataset. A thin black line is used as an aid to see the performance improvement by using the optimized prior instead of the fixed prior (the standard Gaussian prior). The error bars indicate one standard deviation which is estimated by running 4 different random initializations.

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: Results for the active learning scenario. Average test RMSE evaluated at the last step of the iterative data gathering procedure.

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 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.

Figure 16 shows the progressions of average test RMSE during the data collection process. We observe that, on most datasets (CONCRETE, KIN8NM, POWER, PROTEIN, and WINE),
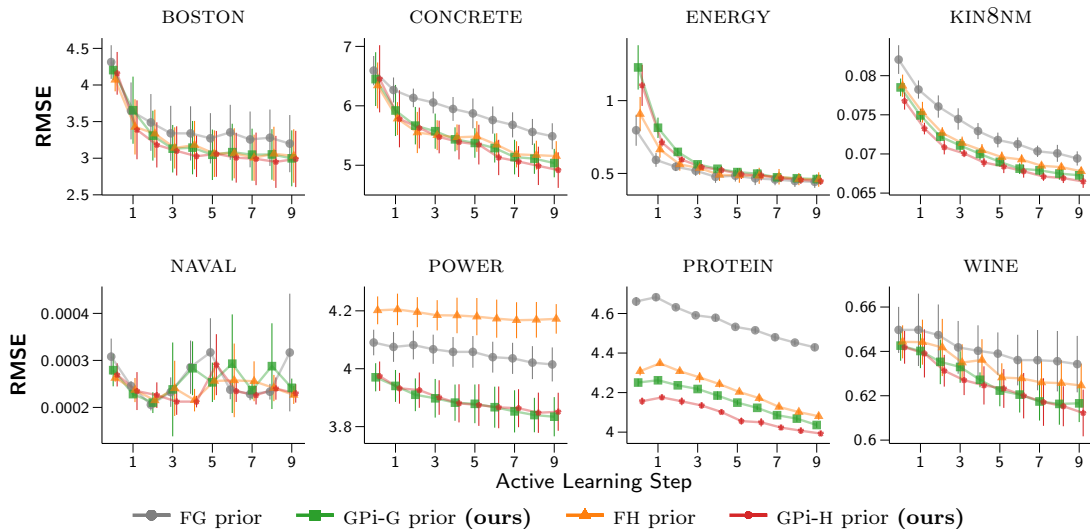
Figure 16: The progressions of average test RMSE and standard errors in the active learning experiment.

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 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. These results show that the GPi-H prior performs best, while the GPi-G prior outperforms the FG prior in most cases.

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 (Figure 17). For the CNNs, as shown in Figure 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.
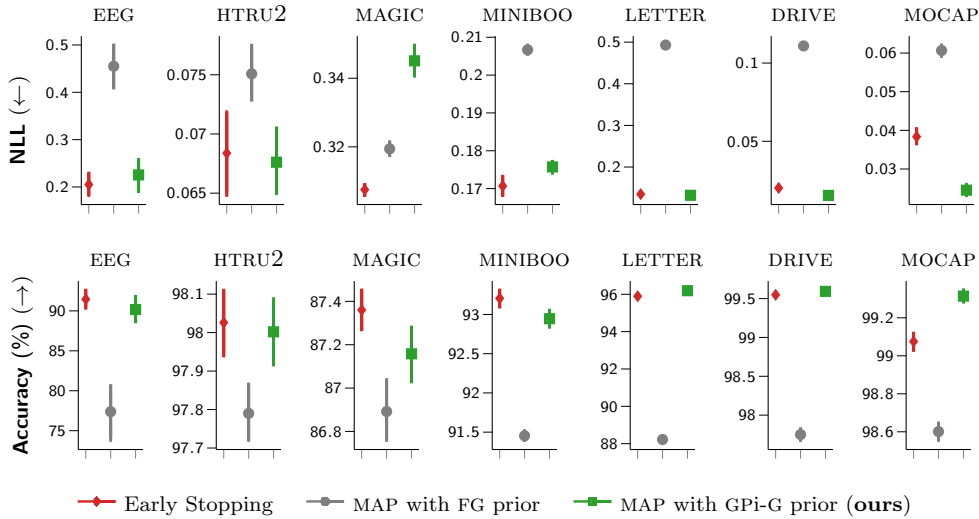
Figure 17: Comparison between early stopping and MAP optimization with the FG and GPi-G priors on the UCI classification datasets.
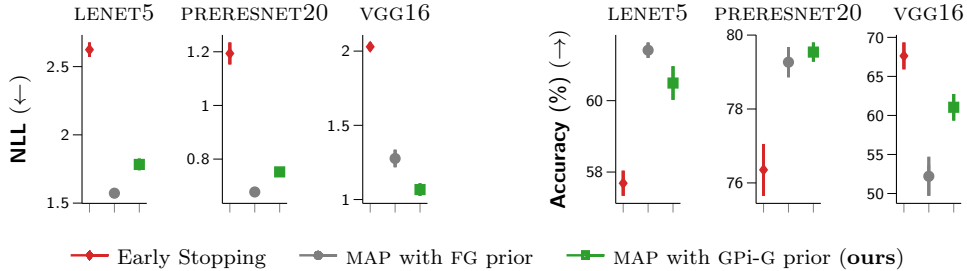
Figure 18: Comparison between early stopping and MAP optimization with the FG and GPi-G priors for three different CNN architectures on the CIFAR10 dataset.

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., 2019; 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, 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.

Acknowledgments

Appendix A. 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 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 1, 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 1 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 B.3 for the progressions of prior optimization.

A.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

$$\mathcal{L}(\mathbf{w}) := -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i \,|\, \mathbf{x}_i, \mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \tag{20}$$

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

$$\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w}) = -\frac{1}{2} \log \sigma_{\mathbf{w}}^2(\mathbf{x}_i) - \frac{(y_i - \mu_{\mathbf{w}}(\mathbf{x}_i))^2}{2\sigma_{\mathbf{w}}^2(\mathbf{x}_i)} + \text{const.} \tag{21}$$

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 MLPs, we use a fixed learning rate 0.01 and total epochs of 50. Whereas 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 $\lambda \in \left\{ 10^k \,|\, k \text{ from -8 to -1} \right\}$.

A.2 Likelihoods for BNNs

Similarly to the prior, the likelihood for BNNs is a modeling choice. It is a function of the model predictions $\hat{\mathbf{y}}$ and the correct targets $\mathbf{y}$. For multi-class C-way classification, the 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, $p(\mathcal{D} \,|\, \mathbf{w}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \hat{y}_{n,c}^{y_{n,c}}$, 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 $p(\mathcal{D} \,|\, \mathbf{w}) = \mathcal{N}(\mathbf{y} \,|\, \hat{\mathbf{y}}, \sigma_\epsilon^2)$. Notice that the noise variance σ_ϵ^2 is treated as a hyperparameter. We do choose this hyperparameter over the grid $\sigma_\epsilon^2 \in \left\{ 5^k, 10^k \,|\, k \text{ from -3 to -1} \right\}$. The optimal values are selected according to the NLL result of the predictive posterior. Table 7 and Table 9 present σ_ϵ^2 used in the UCI regression experiments.

A.3 Sampling from the posterior using scale-adapted SGHMC

As mentioned in § 3.1, we make use of the SGHMC (Chen et al., 2014) to generate posterior samples for BNNs. One caveat of SGHMC and MCMC algorithms, in general, is the difficulty of choosing hyperparameters. To mitigate this problem, in our experiments, we use a scale-apdated version of SGHMC (Springenberg et al., 2016), where the hyperparameters are adjusted automatically during a burn-in phase. After this period, all hyper-parameters stay fixed.

Estimating M. We set the mass matrix $\mathbf{M}^{-1} = \text{diag}\left(\hat{V}_{\mathbf{w}}^{-1/2} \right)$, where $\hat{V}_{\mathbf{w}}$ is an estimate of the uncentered variance of the gradient, $\hat{V}_{\mathbf{w}} \approx \mathbb{E}[(\nabla \tilde{U}(\mathbf{w}))^2]$, which can be estimated by using exponential moving average as follows

$$\Delta \hat{V}_{\mathbf{w}} = -\tau^{-1} \hat{V}_{\mathbf{w}} + \tau^{-1} \nabla (\tilde{U}(\mathbf{w}))^2, \tag{22}$$

where τ is a parameter vector that specifies the moving average windows. This parameter can be automatically chosen by using an adaptive estimate (Springenberg et al., 2016) as follows

$$\Delta \tau = -g_{\mathbf{w}}^2 \hat{V}_{\mathbf{w}}^{-1} \tau + 1, \quad \text{and,} \quad \Delta g_{\mathbf{w}} = -\tau^{-1} g_{\mathbf{w}} + \tau^{-1} \nabla \tilde{U}(\mathbf{w}), \tag{23}$$

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

Estimating $\tilde{\mathbf{B}}$. For the estimate for the noise of the gradient evaluation, $\tilde{\mathbf{B}}$, it should be ideally the estimate of the empirical Fisher information matrix of $U(\mathbf{w})$, which is prohibitively expensive to compute. We therefore use a diagonal approximation, $\tilde{\mathbf{B}} = \frac{1}{2} \varepsilon \hat{V}_{\mathbf{w}}$, which is already available from the step of estimating $\mathbf{M}$.

Choosing C. For the friction matrix, in practice, one can simply choose $\mathbf{C} = C\mathbf{I}$, i.e. the same independent noise for each element of $\mathbf{w}$.

The discretized Hamiltonian dynamics. By substituting $\mathbf{v} := \varepsilon \hat{V}_{\mathbf{w}}^{-1/2} \mathbf{r}$, Equation (5) and Equation (6) become

$$\Delta \mathbf{w} = \mathbf{v}, \tag{24}$$

$$\Delta \mathbf{v} = -\varepsilon^2 \hat{V}_{\mathbf{w}}^{-1/2} \nabla \tilde{U}(\mathbf{w}) - \varepsilon C \hat{V}_{\mathbf{w}}^{-1/2} \mathbf{v} + \mathcal{N}(0, 2\varepsilon^3 C \hat{V}_{\mathbf{w}}^{-1} - \varepsilon^4 \mathbf{I}). \tag{25}$$

Following (Springenberg et al., 2016), we choose C such that $\varepsilon C \hat{V}_{\mathbf{w}}^{-1/2} = \alpha \mathbf{I}$. This is equivalent to using a constant momentum coefficient of α. The final discretized dynamics are then

$$\Delta \mathbf{w} = \mathbf{v}, \tag{26}$$

$$\Delta \mathbf{v} = -\varepsilon^2 \hat{V}_{\mathbf{w}}^{-1/2} \nabla \tilde{U}(\mathbf{w}) - \alpha \mathbf{v} + \mathcal{N}(0, 2\varepsilon^2 \alpha \hat{V}_{\mathbf{w}}^{-1/2} - \varepsilon^4 \mathbf{I}). \tag{27}$$

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.4 Tempered posterior

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

$$p(\mathbf{w} \,|\, \mathcal{D}) \propto \exp(-U(\mathbf{w})/T), \tag{28}$$

where $U(\mathbf{w}) = -\log p(\mathcal{D} \,|\, \mathbf{w}) - \log p(\mathbf{w})$ 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 $p(\mathbf{w})^{\frac{1}{T}}$. 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.5 Details on the sampling scheme for BNN hierarchical priors

As mentioned in § 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

$$w_l^{(1)}, ..., w_l^{(N_l)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{l_w}^2), \tag{29}$$

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:

$$\sigma_{l_w}^2 \sim \Gamma^{-1}(\alpha_{l_w}, \beta_{l_w}). \tag{30}$$

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

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

(ii) Sample the variance from $p\left(\sigma_{l_w}^2 \,|\, \{w_l^{(i)}\}_{i=1}^{N_l}\right)$ 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

$$
\begin{aligned}
p\left(\sigma_{l_w}^2 \,|\, \{w_l^{(i)}\}_{i=1}^{N_l}\right) &\propto \left(\prod_{i=1}^{N_l} p\left(w_l^{(i)} \,|\, \sigma_{l_w}^2\right)\right) p\left(\sigma_{l_w}^2 \,|\, \alpha_{l_w}, \beta_{l_w}\right) \\
&\propto \left(\prod_{i=1}^{N_l} \left(\sigma_{l_w}^2\right)^{-1/2} \exp\left\{-\frac{1}{2\sigma_{l_w}^2}\left(w_l^{(i)}\right)^2\right\}\right) \left(\sigma_{l_w}^2\right)^{-\alpha_{l_w}-1} \exp\left\{-\frac{1}{\sigma_{l_w}^2}\beta_{l_w}\right\} \\
&= \left(\sigma_{l_w}^2\right)^{-(\alpha_{l_w}+N_l/2)-1} \exp\left\{-\frac{1}{\sigma_{l_w}^2}\left(\beta_{l_w} + \frac{1}{2}\sum_{i=1}^{N_l}\left(w_l^{(i)}\right)^2\right)\right\} \\
&\propto \Gamma^{-1}\left(\alpha_{l_w} + \frac{N_l}{2}, \beta_{l_w} + \frac{1}{2}\sum_{i=1}^{N_l}\left(w_l^{(i)}\right)^2\right).
\end{aligned}
\tag{31}
$$

As a default, in our experiments, we set the resampling interval $K = 100$ except for the experiment on the 1D synthetic dataset (§ 5.1), in which we use $K = 20$.

A.6 MAP estimation with Gaussian prior

For completeness, we describe the MAP estimation for the case of Gaussian prior used in § 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:

$$
\begin{aligned}
\mathbf{w}_{\mathrm{MAP}} &= \arg\max_{\mathbf{w}} p(\mathbf{w} \,|\, \mathcal{D}) \\
&= \arg\max_{\mathbf{w}} p(\mathcal{D} \,|\, \mathbf{w})p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \{\log p(\mathcal{D} \,|\, \mathbf{w}) + \log p(\mathbf{w})\}.
\end{aligned}
\tag{32}
$$

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

$$
\mathbf{w}_{\mathrm{MAP}} = \arg\max_{\mathbf{w}} \left\{\log p(\mathcal{D} \,|\, \mathbf{w}) - \frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu})\right\}.
\tag{33}
$$

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

$$
\mathbf{w}_{\mathrm{MAP}} = \arg\max_{\mathbf{w}} \left\{\log p(\mathcal{D} \,|\, \mathbf{w}) - \frac{1}{2\sigma_{\mathrm{prior}}^2}\|\mathbf{w}\|_2^2\right\}
\tag{34}
$$

Here, we use the same likelihoods $p(\mathcal{D}|\mathbf{w})$ as in Appendix A.2.

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

$$
\mathbf{w}_{\mathrm{MAP}} = \arg\max_{\mathbf{w}} \left\{-\frac{1}{2\sigma_\epsilon^2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 - \frac{1}{2\sigma_{\mathrm{prior}}^2}\|\mathbf{w}\|_2^2\right\}.
\tag{35}
$$

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

$$\mathbf{w}_{\text{MAP}} = \arg\min_{\mathbf{w}} \left\{ \sum_{n=1}^{N} (\hat{y}_n - y_n)^2 + \frac{\sigma_\epsilon^2}{\sigma_{\text{prior}}^2} \|\mathbf{w}\|_2^2 \right\}. \tag{36}$$

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

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

$$\mathbf{w}_{\text{MAP}} = \arg\min_{\mathbf{w}} \left\{ -\sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c}) + \frac{1}{2\sigma_{\text{prior}}^2} \|\mathbf{w}\|_2^2 \right\}, \tag{37}$$

where $\frac{1}{\sigma_{\text{prior}}^2}$ 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 4: LENET5

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 5: PRERESNET20

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 6: VGG16

A.7 Network architectures

As previously mentioned in § 3.1, 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 4 to 6 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.8 Measuring similarity between GPs and BNNs using maximum mean discrepancy

In § 5.3, we adopted the approach of Matthews et al. (2018) to measure the similarity between GPs and BNNs using a kernel two-sample test based on MMD Gretton et al. (2012). The MMD between two distributions p_{gp} and p_{nn} is defined as follows

$$\mathrm{MMD}(p_{gp}, p_{nn}) = \sup_{\|h\|_{\mathcal{H}} \leq 1} \left[\mathbb{E}_{p_{gp}}[h] - \mathbb{E}_{p_{nn}}[h] \right], \tag{38}$$

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 $\mathbf{X}_{\mathcal{M}}$, we can obtain the closed form of MMD as follows

$$\mathrm{MMD}^2(p_{gp}, p_{nn}) = \mathbb{E}_{\mathbf{f}_{\mathcal{M}}, \mathbf{f}'_{\mathcal{M}} \sim p_{gp}}[K(\mathbf{f}_{\mathcal{M}}, \mathbf{f}'_{\mathcal{M}})] + \mathbb{E}_{\mathbf{f}_{\mathcal{M}}, \mathbf{f}'_{\mathcal{M}} \sim p_{nn}}[K(\mathbf{f}_{\mathcal{M}}, \mathbf{f}'_{\mathcal{M}})] \tag{39}$$
$$- 2\mathbb{E}_{\mathbf{f}_{\mathcal{M}} \sim p_{gp}, \mathbf{f}'_{\mathcal{M}} \sim p_{nn}}[K(\mathbf{f}_{\mathcal{M}}, \mathbf{f}'_{\mathcal{M}})],$$

which can be estimated by using samples from p_{nn} and p_{gp} evaluated at $\mathbf{X}_{\mathcal{M}}$ (Gretton et al., 2012). For the MMD estimate, we use an 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.9 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)[3]. 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., 2021a), following the Authors' repository[4], 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., 2021a) including the optimizer, the early stopping scheme and the frequency of updating the prior.

Appendix B. Additional results

B.1 Additional results on MAP estimation with GP-induced priors

Figure 19 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 § 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.

---

3. `https://github.com/ssydasheng/FBNN`
4. `https://github.com/AlexImmer/marglik`
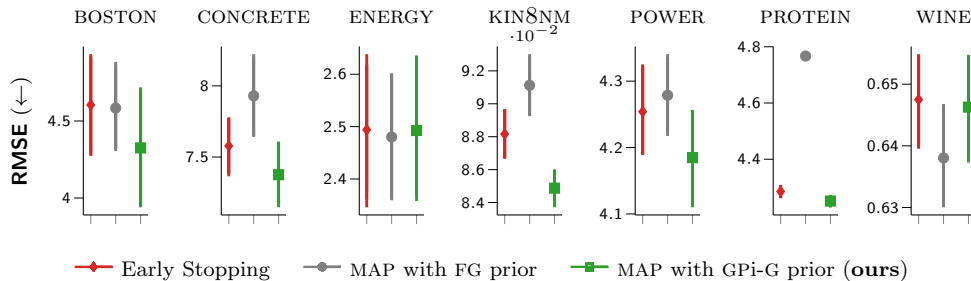
Figure 19: Comparison between early stopping and MAP estimations with respect to the FG and GPi-G priors on the UCI regression datasets.
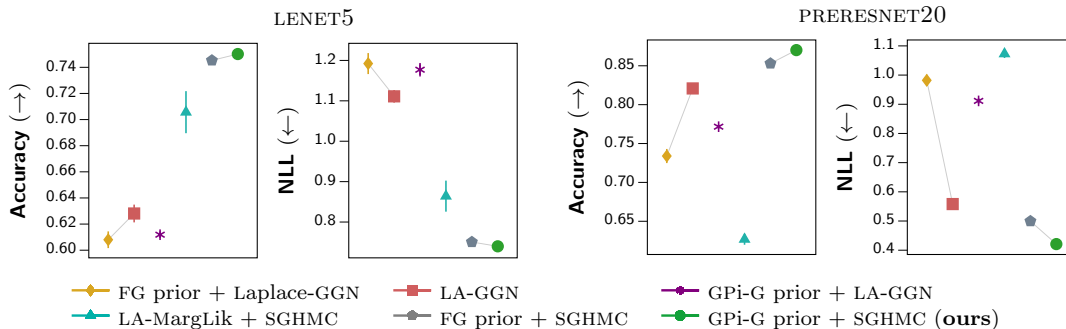
Figure 20: Comparison with empirical Bayes and functional inference methods on CIFAR10 dataset.

B.2 Tabular results on the UCI benchmarks

Detailed results on the UCI regression and classification datasets are reported in Tables 7 to 10.

B.3 Convergence of Wasserstein optimization

Figures 22 to 24 depict the progressions of Wasserstein optimization in the UCI regression, UCI classification, and CIFAR10 experiments, respectively.

B.4 Additional comparisons with the empirical Bayes approach

We complement the results presented in § 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., 2021b) for inference, refered as "GPi-G prior + LA-GGN". As shown in Figure 20, 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., 2021a) 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 hyper-parameters 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 7: Average test 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.

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 8: 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	$\mathbf{0.476} \pm 0.166$	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	-5.926 ± 0.051	$\mathbf{-5.895} \pm 0.051$	-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 9: 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.

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 10: 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.

B.5 Additional results with full-batch Hamiltonian Monte Carlo

Table 11 shows a comparison between full-batch HMC and SGHMC using the FG and our GPi-G priors on small UCI regression datasets. We use the no-u-turn (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

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 11: 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.

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.

### B.6 Additional discussion on the optimization of Wasserstein distance

In the Algorithm 1, we have opted to separate the two optimization procedures for the Lipschitz function $\phi_{\boldsymbol{\theta}}$ and the Wasserstein distance. We acknowledge that the two could have been optimized jointly in a single loop, as Equation (13) defines a minimax problem. However, our choice allows $\phi_{\boldsymbol{\theta}}$ 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 allow more training of the discriminator for each step of the generator). Figure 21 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.8 for the experimental protocol). Our optimization strategy demonstrates a much more stable convergence compared to the joint optimization approach.
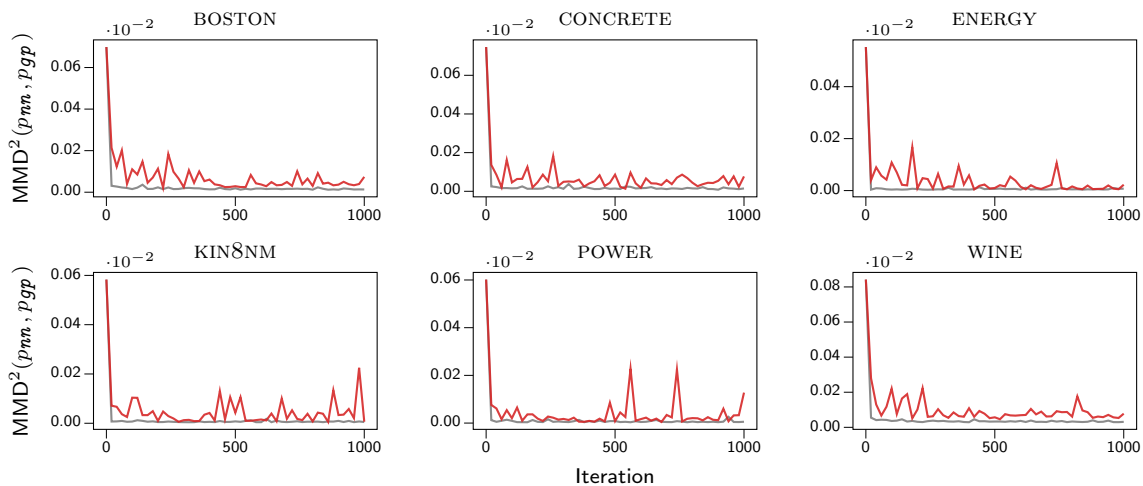
Figure 21: Comparison between strategies to optimize the Lipschitz function and the Wasserstein distance: (▬) our strategy of separating these two operations; and (▬) the strategy of joint optimization. Here, the convergence is measured by the squared MMD between the target GP prior and the optimized BNN prior.

(a) GPi-G prior

(b) GPi-H prior

(c) GPi-NF prior
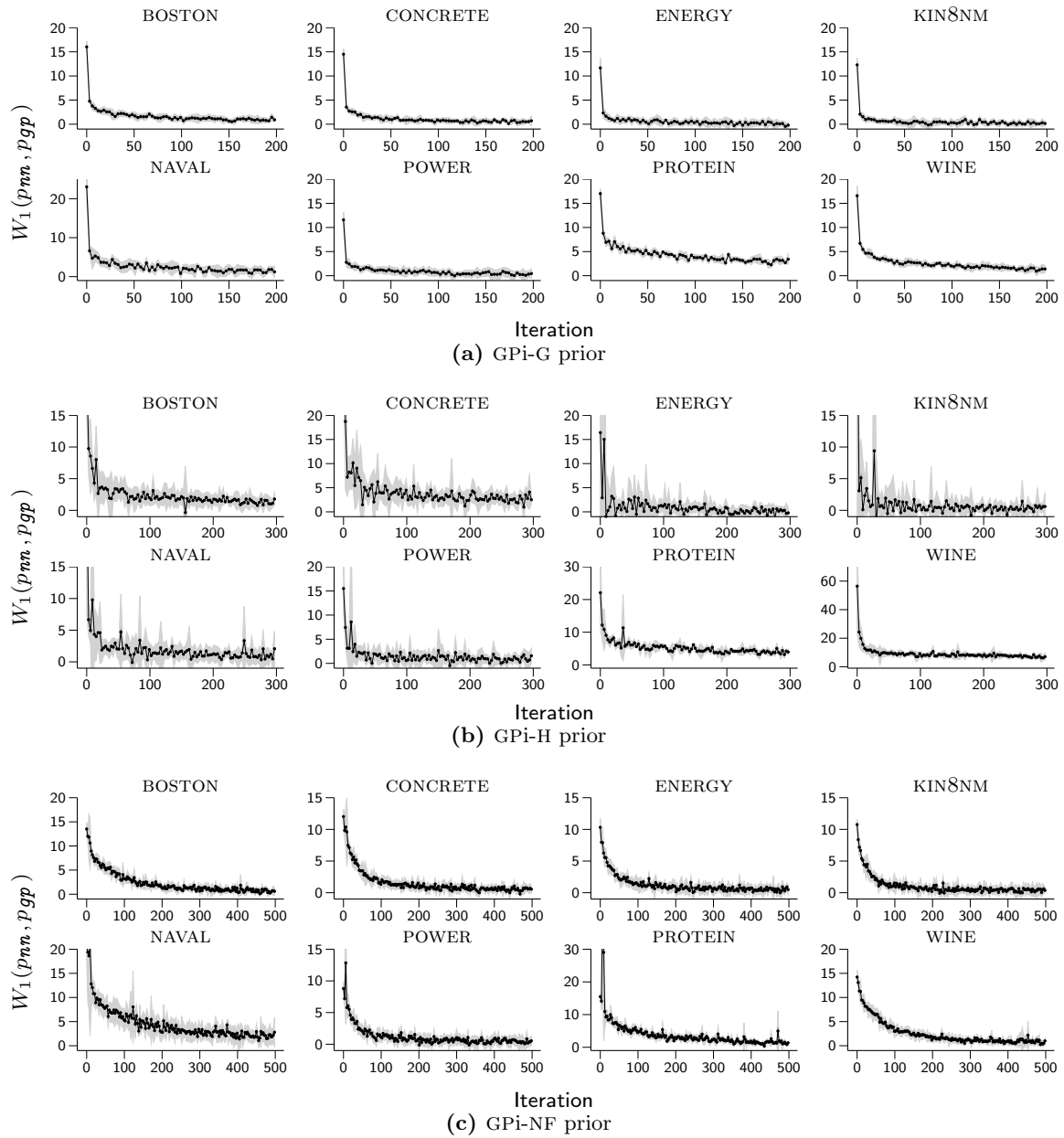
Figure 22: Convergence of Wasserstein optimization for two-layer MLPs on the UCI regression datasets.
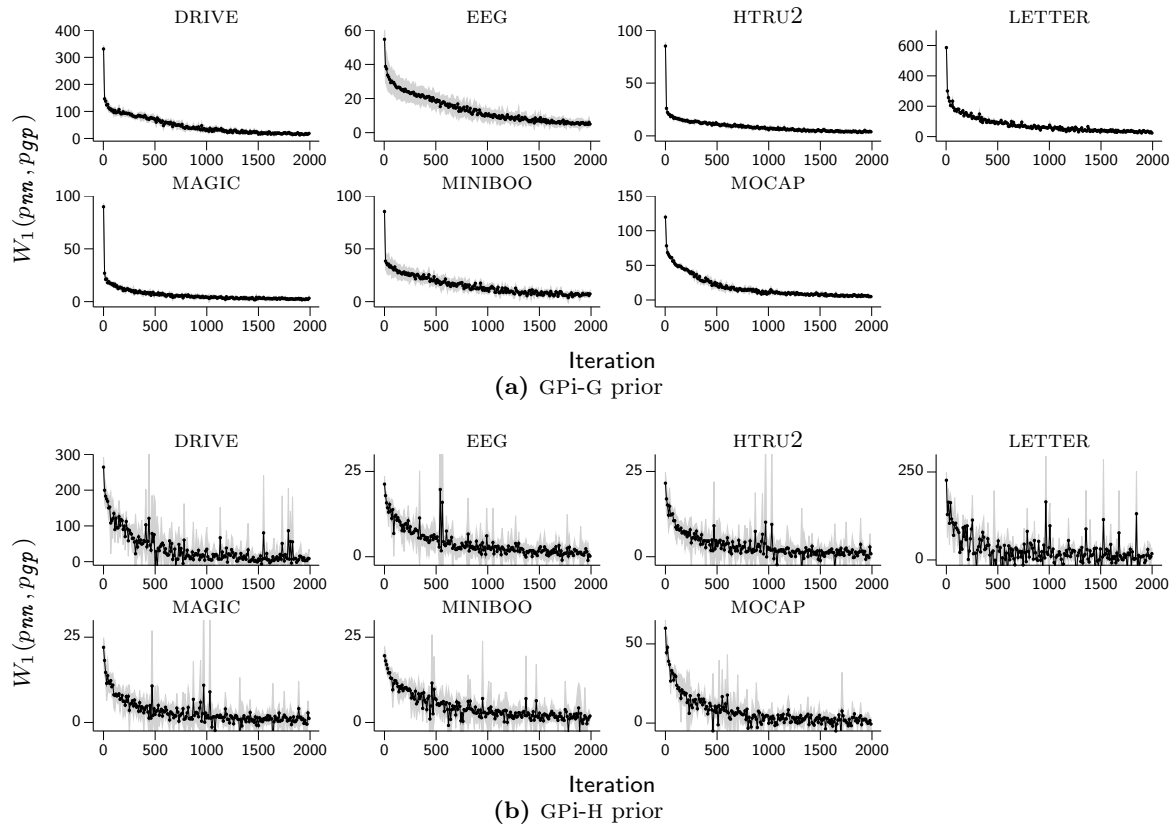
Figure 23: Convergence of Wasserstein optimization for two-layer MLPs on the UCI classification datasets.
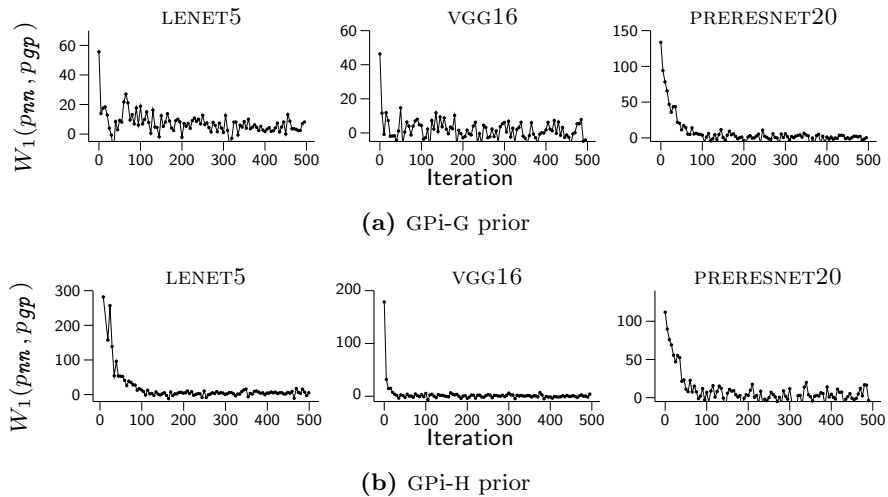
Figure 24: Convergence of Wasserstein optimization for CNNs on the CIFAR10 dataset.

Appendix C. A primer on Wasserstein Distance

Given two *Borel's probability measures* $\pi(\mathbf{x})$ and $\nu(\mathbf{y})$ 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

$$W_p(\pi, \nu) = \left(\inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} D(\mathbf{x}, \mathbf{y})^p \gamma(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{y} \right)^{1/p}, \tag{40}$$

where $D(\mathbf{x}, \mathbf{y})$ is a proper distance metric between two points $\mathbf{x}$ and $\mathbf{y}$ 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 $\mathbf{x}$ and $\mathbf{y}$ coincides (i.e. $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subseteq \mathbb{R}^d$), the most used formulation is the 1-Wasserstein distance with Euclidian norm as distance,

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|\mathbf{x} - \mathbf{y}\| \gamma(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{y}, \tag{41}$$

This is also known in the literature as the Earth-Mover distance. Intuitively, here γ measures how much mass must be transported from $\mathbf{x}$ to $\mathbf{y}$ 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 Equation (41). 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 1 *On the same setup as before, the Wasserstein distance defined as*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|\mathbf{x} - \mathbf{y}\| \gamma(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{y}, \tag{42}$$

admits the following dual form

$$W(\pi, \nu) = \sup_{\|f\|_L \leq 1} \int_{\mathcal{X}} f(\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{X}} f(\mathbf{y}) \nu(\mathbf{y}) d\mathbf{y} \tag{43}$$

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

Step 1: Kantorovich duality

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

Theorem 2 *Given a nonnegative measurable function $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the 1-Wasserstein is computed as follows,*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int D(\mathbf{x}, \mathbf{y}) \gamma(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{y}, \tag{44}$$

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

$$W(\pi, \nu) = \sup_{\substack{f,g \\ f(\mathbf{x})+g(\mathbf{y}) \leq D(\mathbf{x},\mathbf{y})}} \int f(\mathbf{x})\pi(\mathbf{x}) \, d\mathbf{x} + \int g(\mathbf{y})\nu(\mathbf{y}) \, d\mathbf{y} \,. \tag{45}$$

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

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

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 $\pi(\mathbf{x}) = \int \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{y}$ and $\nu(\mathbf{y}) = \int \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x}$. By substituiting these quantities, it follows that

$$\int f(\mathbf{x})\pi(\mathbf{x}) \, d\mathbf{x} + \int g(\mathbf{y})\nu(\mathbf{y}) \, d\mathbf{y} = \int f(\mathbf{x}) \int \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{y} \, d\mathbf{x} + \int g(\mathbf{y}) \int \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y}$$

$$= \iint [f(\mathbf{x}) + g(\mathbf{y})] \, \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} \,.$$

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 Equation (42),

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

$$= \inf_\gamma \left[\int D(\mathbf{x}, \mathbf{y})\gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} + \sup_{f,g} \left[\int f(\mathbf{x})\pi(\mathbf{x}) \, d\mathbf{x} + \int g(\mathbf{y})\nu(\mathbf{y}) \, d\mathbf{y} - \right. \right. \tag{46}$$

$$\left. \left. \iint [f(\mathbf{x}) + g(\mathbf{y})] \, \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} \right] \right] \,,$$

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

$$W(\pi, \nu) = \inf_\gamma \sup_{f,g} \Upsilon(\gamma, (f, g)) \tag{47}$$

$$\Upsilon(\gamma, (f, g)) \overset{\text{def}}{=} \int D(\mathbf{x}, \mathbf{y})\gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} + \int f(\mathbf{x})\pi(\mathbf{x}) \, d\mathbf{x} + \int g(\mathbf{y})\nu(\mathbf{y}) \, d\mathbf{y} -$$

$$\iint [f(\mathbf{x}) + g(\mathbf{y})] \, \gamma(\mathbf{x}, \mathbf{y}) \, d\mathbf{x} \, d\mathbf{y}$$

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,

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

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

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

If the function $\zeta(\mathbf{x}, \mathbf{y}) = D(\mathbf{x}, \mathbf{y}) - (f(\mathbf{x}) + g(\mathbf{y}))$ takes a negative value at some point $(\mathbf{x}_0, \mathbf{y}_0)$, then by choosing $\gamma = \lambda \delta(\mathbf{x}_0, \mathbf{y}_0)$ with $\lambda \to +\infty$ (i.e. a Dirac delta in $(\mathbf{x}_0, \mathbf{y}_0)$), we see that the infimum is infinite. On the other hand, is $\zeta(\mathbf{x}, \mathbf{y})$ 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 Equation (43).

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 $f(\mathbf{x})$. This means that the objective is to find a good $g(\mathbf{y})$ that for all $\mathbf{x}, \mathbf{y}$ satisfy the condition

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

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

$$g(\mathbf{y}) \leq \inf_{\mathbf{x}} [D(\mathbf{x}, \mathbf{y}) - f(\mathbf{x})].$$

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,

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

We therefore have the following formulation of the Wasserstein distance,

$$W(\pi, \nu) = \sup_{f} \left[\int f(\mathbf{x}) \pi(\mathbf{x}) \, \mathrm{d}\mathbf{x} + \int \bar{f}(\mathbf{y}) \nu(\mathbf{y}) \, \mathrm{d}\mathbf{y} \right]$$

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

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

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

$$W(\pi, \nu) = \sup_{f} \left[\int \bar{\bar{f}}(\mathbf{x}) \pi(\mathbf{x}) \, \mathrm{d}\mathbf{x} + \int \bar{f}(\mathbf{y}) \nu(\mathbf{y}) \, \mathrm{d}\mathbf{y} \right]$$

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 3 *When $D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ 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 $\mathbf{x}$

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

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

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

Since this is true for all $\mathbf{y}$,

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

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

Step 3. Putting everything together

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

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

We then proved that

$$\sup_{\substack{f,g \\ f(\mathbf{x})+g(\mathbf{y}) \leq D(\mathbf{x},\mathbf{y})}} \left[\int f(\mathbf{x})\pi(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int g(\mathbf{y})\nu(\mathbf{y})\,\mathrm{d}\mathbf{y} \right] =$$
$$= \sup_{\substack{f \\ \bar{f}=\inf_{\mathbf{x}} D-f}} \left[\int f(\mathbf{x})\pi(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int \bar{f}(\mathbf{y})\nu(\mathbf{y})\,\mathrm{d}\mathbf{y} \right],$$

Finally, given $D(\mathbf{x}, \mathbf{y})$ 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 Equation (43),

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

References

R. Amit and R. Meir. Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 205–214. PMLR, 2018.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. In *International Conference on Learning Representations*, 2020.

A. Atanov, A. Ashukha, K. Struminsky, D. Vetrov, and M. Welling. The Deep Weight Prior. In *International Conference on Learning Representations*, 2019.

C. M. Bishop. *Pattern recognition and machine learning.* Springer, 1st ed. 2006. corr. 2nd printing 2011 edition, Aug. 2006.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic Integration: A Role in Statistical Computation? *Statistical Science*, 34(1):1–22, 02 2019.

T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1683–1691. PMLR, 2014.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2017.

J. Cockayne, C. J. Oates, I. C. Ipsen, and M. Girolami. A Bayesian Conjugate Gradient Method (with Discussion). *Bayesian Analysis*, 14(3):937–1012, 09 2019.

E. A. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace Redux – Effortless Bayesian Deep Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 20089–20103, 2021.

S. Delattre and N. Fournier. On the Kozachenko–Leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.

D. Dua and C. Graff. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2017. URL http://archive.ics.uci.edu/ml.

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

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding Pathologies in Very Deep Networks. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 202–210. PMLR, 2014.

D. Flam-Shepherd, J. Requeima, and D. Duvenaud. Mapping Gaussian Process Priors to Bayesian Neural Networks. In *NeurIPS workshop on Bayesian Deep Learning*, 2017.

D. Flam-Shepherd, J. Requeima, and D. Duvenaud. Characterizing and Warping the Function space of Bayesian Neural Networks. In *NeurIPS workshop on Bayesian Deep Learning*, 2018.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33nd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. JMLR, June 19-24 2016.

A. Gelman and D. B. Rubin. Inference from Iterative Simulation using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

A. Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356. Curran Associates, Inc., 2011.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13:723–773, 2012.

A. Grover, M. Dhar, and S. Ermon. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In *Proceedings of the 32nd Conference on Artificial Intelligence*, pages 3069–3076. AAAI Press, 2018.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777. Curran Associates, Inc., 2017.

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

D. Hafner, D. Tran, T. P. Lillicrap, A. Irpan, and J. Davidson. Noise Contrastive Priors for Functional Uncertainty. In *Proceedings of the 35h Conference on Uncertainty in Artificial Intelligence*, page 332. AUAI Press, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In *Proceeding of the 14th European Conference on Computer Vision*, volume 9908 (Part IV) of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016.

J. Heek and N. Kalchbrenner. Bayesian Inference for Large Scale Image Classification. arXiv:1908.03491, 2019.

D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019.

M. D. Hoffman and A. Gelman. The No-U-turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

N. Houlsby, F. Huszar, Z. Ghahramani, and J. Hernández-lobato. Collaborative Gaussian Processes for Preference Learning. In *Advances in Neural Information Processing Systems*, volume 25, pages 2096–2104. Curran Associates, Inc., 2012.

A. Immer, M. Bauer, V. Fortuin, G. Rätsch, and M. E. Khan. Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4563–4573. PMLR, 2021a.

A. Immer, M. Korzepa, and M. Bauer. Improving Predictions of Bayesian Neural Nets via Local Linearization. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 703–711. PMLR, 2021b.

A. Jacot, F. Gabriel, and C. Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 8571–8580. Curran Associates, Inc., 2018.

M. Jankowiak and F. Obermeyer. Pathwise Derivatives Beyond the Reparameterization Trick. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2240–2249. PMLR, 2018.

L. V. Kantorovich. On the transfer of masses. *Doklady Akademii Nauk SSSR*, 37:227–229, 1942.

L. V. Kantorovich. On a problem of Monge. *Uspekhi Matematicheskikh Nauk*, 3:225–226, 1948.

T. Karaletsos and T. D. Bui. Gaussian Process Meta-Representations For Hierarchical Neural Network Weight Priors. In *2nd Symposium on Advances in Approximate Bayesian Inference*, 2019.

T. Karaletsos and T. D. Bui. Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, volume 30, pages 5574–5584. Curran Associates, Inc., 2017.

M. E. Khan, A. Immer, E. Abedi, and M. Korzepa. Approximate Inference Turns Deep Networks into Gaussian Processes. In *Advances in Neural Information Processing Systems*, pages 3088–3098, 2019.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems*, volume 29, pages 4743–4751. Curran Associates, Inc., 2016.

A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413. Curran Associates, Inc., 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

J. Lee, S. S. Schoenholz, J. Pennington, B. Adlam, L. Xiao, R. Novak, and J. Sohl-Dickstein. Finite Versus Infinite Neural Networks: an Empirical Study. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

H. Liu, Y. S. Ong, X. Shen, and J. Cai. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11): 4405–4423, 2020.

Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances in Neural Information Processing Systems*, volume 29, pages 2378–2386. Curran Associates, Inc., 2016.

C. Louizos and M. Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2218–2227. PMLR, 2017.

C. Ma, Y. Li, and J. M. Hernández-Lobato. Variational Implicit Processes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4222–4233. PMLR, 2019.

D. J. MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

D. J. MacKay. Probable Networks and Plausible Predictions - a Review of Practical Bayesian Methods for Supervised Neural Networks. *Network: Computation in Neural Systems*, 6 (3):469–505, 1995.

D. J. MacKay. Bayesian non-linear modeling for the prediction competition. In *Maximum Entropy and Bayesian Methods*, pages 221–234. Springer, 1996.

D. J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st edition, 2003.

T. Matsubara, C. J. Oates, and F. Briol. The Ridgelet Prior: A Covariance Function Approach to Prior Specification for Bayesian Neural Networks. *Journal of Machine Learning Research*, 22:1–57, 2021.

A. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, 2018.

J. Močkus. On Bayesian Methods for Seeking the Extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk*, pages 400–404. Springer Berlin Heidelberg, 1975.

A. Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

E. T. Nalisnick, J. Gordon, and J. M. Hernández-Lobato. Predictive Complexity Priors. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 694–702. PMLR, 2021.

R. M. Neal. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1st edition, Aug. 1996.

F. Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL `https://github.com/fmfn/BayesianOptimization`.

A. O'Hagan. Bayes–Hermite quadrature. *Journal of Statistical Planning and Inference*, 29 (3):245 – 260, 1991.

K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota. Practical Deep Learning with Bayesian Principles. In *Advances in Neural Information Processing Systems*, volume 32, pages 4287–4299. Curran Associates, Inc., 2019.

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems*, volume 32, pages 13991–14002. Curran Associates, Inc., 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019.

T. Pearce, R. Tsuchida, M. Zaki, A. Brintrup, and A. Neely. Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*, page 25. AUAI Press, 2019.

D. Phan, N. Pradhan, and M. Jankowiak. Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro. arXiv:1912.11554, 2019.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems*, volume 15, pages 489–496. MIT Press, 2002.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32 of *Proceeding of Machine Learning Research*, pages 1278–1286, Beijing, China, 21-26 June 2014. PMLR.

S. Rossi, P. Michiardi, and M. Filippone. Good Initializations of Variational Bayes for Deep Models. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5487–5497, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

S. Rossi, S. Marmin, and M. Filippone. Walsh-Hadamard Variational Inference for Bayesian Deep Learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

B. Settles. Active Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

J. Shi, S. Sun, and J. Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4651–4660. PMLR, 2018.

J. Shi, M. E. Khan, and J. Zhu. Scalable Training of Inference Networks for Gaussian-Process Models. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5758–5768. PMLR, 2019.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

N. Skafte, M. Jorgensen, and S. Hauberg. Reliable Training and Estimation of Variance Networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 6326–6336. Curran Associates, Inc., 2019.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 4134–4142. Curran Associates, Inc., 2016.

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022. Omnipress, 2010.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional Variational Bayesian Neural Networks. In *International Conference on Learning Representations*, 2019.

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the Gradient by a Running Average of Its Recent Magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Tishby, Levin, and Solla. Consistent inference of probabilities in layered networks: predictions and generalizations. In *International 1989 Joint Conference on Neural Networks*, pages 403–409 vol.2, 1989.

B.-H. Tran, S. Rossi, D. Milios, P. Michiardi, E. V. Bonilla, and M. Filippone. Model Selection for Bayesian Autoencoders. In *Advances in Neural Information Processing Systems*, volume 34, pages 19730–19742. Curran Associates, Inc., 2021.

C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003.

F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How Good is the Bayes Posterior in Deep Neural Networks Really? In *Proceeding of the 37th International Conference on Machine Learning*, 2020.

W. Yang, L. Lorch, M. A. Graule, S. Srinivasan, A. Suresh, J. Yao, M. F. Pradier, and F. Doshi-velez. Output-Constrained Bayesian Neural Networks. In *ICML workshop on Uncertainty & Robustness in Deep Learning*, 2019.

Y. Yao, L. Rosasco, and A. Caponnetto. On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, 26(2):289–315, 2007.

R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. In *International Conference on Learning Representations*, 2020.