FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Variational Bayesian neural networks (BNN) perform variational inference over weights, but it is difficult to specify meaningful priors and approximate posteriors in a high-dimensional weight space. We introduce functional variational Bayesian neural networks (fBNNs), which maximize an Evidence Lower BOund (ELBO) defined directly on stochastic processes, i.e. distributions over functions. We prove that the KL divergence between stochastic processes is equal to the supremum of marginal KL divergences over all finite sets of inputs. Based on this, we introduce a practical training objective which approximates the functional ELBO using finite measurement sets and the spectral Stein gradient estimator. With fBNNs, we can specify priors which entail rich structures, including Gaussian processes and implicit stochastic processes. Empirically, we find that fBNNs extrapolate well using various structured priors, provide reliable uncertainty estimates, and can scale to large datasets.

1 INTRODUCTION

Bayesian neural networks (BNNs) (Hinton & Van Camp, 1993; Neal, 1995) have the potential to combine the scalability, flexibility, and predictive performance of neural networks with principled Bayesian uncertainty modelling. However, the practical effectiveness of BNNs is limited by our ability to specify meaningful prior distributions and by the intractability of posterior inference. Choosing a meaningful prior distribution over network weights is difficult because the weights have a complicated relationship to the function computed by the network. Stochastic variational inference is appealing because the update rules resemble ordinary backprop (Graves, 2011; Blundell et al., 2015), but fitting accurate posterior distributions is difficult due to strong and complicated posterior dependencies (Louizos & Welling, 2016; Sun et al., 2017; Zhang et al., 2018; Shi et al., 2018a).

In a classic result, Neal (1995) showed that under certain assumptions, as the width of a shallow BNN was increased, the limiting distribution is a Gaussian process (GP). Lee et al. (2018) recently extended this result to deep BNNs. And Deep Gaussian Processes (DGP) (Cutajar et al., 2017; Salimbeni & Deisenroth, 2017) have close connections to BNNs due to similar deep structures. However, the relationship of finite BNNs to GPs is unclear, and practical variational BNN approximations fail to match the predictions of the corresponding GP. Furthermore, because the aforementioned analyses related specific BNN architectures to specific GP kernels, it's not clear how one would design BNN architectures to match a given kernel structure. Given the rich variety of structural assumptions that GP kernels can represent (Rasmussen & Williams, 2006; Lloyd et al., 2014; Sun et al., 2018), there remains a significant gap in expressive power between BNNs and GPs (not to mention stochastic processes more broadly).

In this paper, we perform variational inference directly on the distribution of functions. Specifically, we introduce functional variational BNNs (fBNNs), where a BNN is trained to produce a distribution of functions with small KL divergence to the true posterior over functions. We prove that the KL divergence between stochastic processes[1] can be expressed as the supremum of marginal KL divergences at finite sets of points. Based on this, we present a novel objective functional ELBO (fELBO). Then we introduce a GAN-like minimax formulation and a sampling-based approximation for functional variational inference. To approximate the marginal KL divergence gradients, we adopt the recently proposed spectral Stein gradient estimator (SSGE) (Shi et al., 2018b).

[1] Matthews et al. (2016) studied the KL between stochastic processes without providing practical formulas.

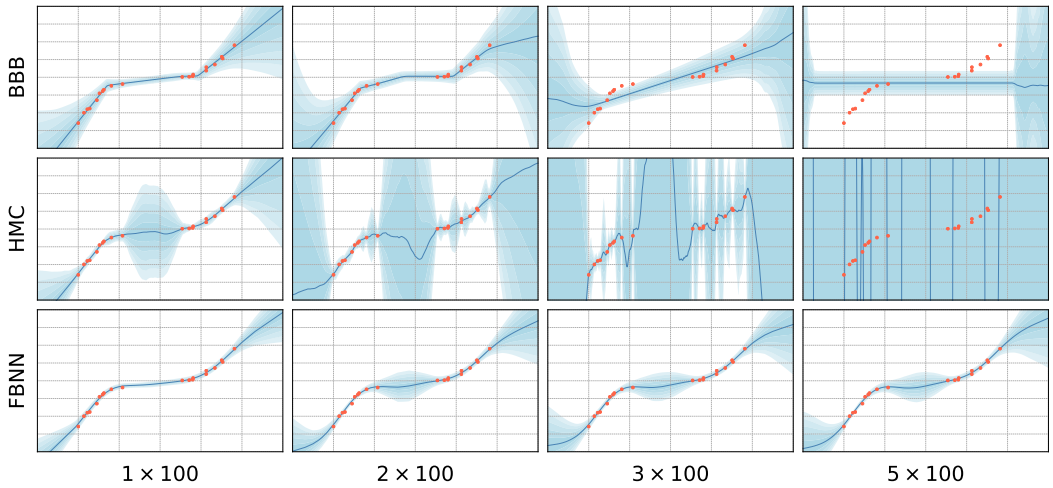


Figure 1: Predictions on x^3 problem. Here $a \times b$ represents a hidden layers of b units. Red dots are 20 training points. Blue line is the mean of final prediction and the shaded areas represent standard derivations. We compare fBNNs, BBB, and the golden standard Hamiltonian Monte Carlo (HMC) (Neal et al., 2011). For both BBB and HMC which performs weight-space inference, varying network sizes lead to drastically different predictions. For fBNNs which perform function-space inference, we observe consistent predictions for different networks. Note the 1×100 factorized Gaussian fBNNs network is not expressive enough to generate much diverse predictions.

Our fBNNs make it possible to specify stochastic process priors which encode richly structured dependencies between function values. This includes stochastic processes with explicit densities, such as GPs which can model various structures like smoothness and periodicity (Lloyd et al., 2014; Sun et al., 2018). We can also use stochastic processes with implicit densities, such as distributions over piecewise linear or piecewise constant functions. Furthermore, in contrast with GPs, fBNNs efficiently yield explicit samples of the function. This enables fBNNs to be used in settings that require explicit minimization of sampled functions, such as Thompson sampling (Thompson, 1933) or predictive entropy search (Hernández-Lobato et al., 2014; Wang & Jegelka, 2017).

One desideratum of Bayesian models is that they behave gracefully as their capacity is increased (Rasmussen & Ghahramani, 2001). Unfortunately, ordinary BNNs don't meet this basic requirement: unless the asymptotic regime is chosen very carefully (e.g. Neal (1995)), BNN priors may have undesirable behaviors as more units or layers are added. Furthermore, larger BNNs entail more difficult posterior inference and larger description length for the posterior, causing degeneracy for large networks, as shown in Figure 1. Our fBNNs do not have this problem, since the prior is defined directly over the space of functions, and therefore the BNN can be made arbitrarily large without changing the functional variational inference problem. Hence, their predictions behave well as the capacity is increased, as shown in Figure 1.

Empirically, we demonstrate fBNNs generate sensible extrapolations for both explicit periodic priors and implicit piecewise priors. We show fBNNs outperform competing approaches on both small scale and large scale regression datasets. And we manifest fBNNs output reliable uncertainty estimations by the state-of-art performance on the contextual bandits benchmark (Riquelme et al., 2018).

2 Background

2.1 Variational Inference for Bayesian Neural Networks

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a Bayesian neural network (BNN) is defined in terms of a prior $p(\mathbf{w})$ on the weights, as well as the likelihood $p(\mathcal{D}|\mathbf{w})$. Variational Bayesian methods (Hinton & Van Camp, 1993; Graves, 2011; Blundell et al., 2015) attempt to fit an approximate posterior $q(\mathbf{w})$ to maximize the evidence lower bound (ELBO):

$$\mathcal{L}_q = \mathbb{E}_q[\log p(\mathcal{D}|\mathbf{w})] - \mathrm{KL}[q(\mathbf{w})\|p(\mathbf{w})]. \tag{1}$$

The most commonly used variational BNN training method is Bayes By Backprop (BBB) (Blundell et al., 2015), which uses a fully factorized Gaussian approximation to the posterior, i.e. $q(\mathbf{w}) =$

$\mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$. Using the reparameterization trick (Kingma & Welling, 2013), the gradients of ELBO towards μ, σ can be computed by backpropagation, and then be used for updates.

Most commonly, the prior $p(\mathbf{w})$ is chosen for computational convenience; for instance, independent Gaussian or Gaussian mixture distributions. Other priors, including log-uniform priors (Kingma et al., 2015; Louizos et al., 2017) and horseshoe priors (Ghosh et al., 2018; Louizos et al., 2017), were proposed for specific purposes such as model compression and model selection. But the relationships of weight-space priors to the functions computed by networks are difficult to characterize.

2.2 Stochastic Processes

A stochastic process (Lamperti, 2012) F is typically defined as a collection of random variables, on a probability space $(\Omega, \mathcal{F}, P)$. The random variables, indexed by some set $\mathcal{X}$, all take values in the same mathematical space $\mathcal{Y}$. In other words, given a probability space (Ω, Σ, P), a stochastic can be simply written as $\{F(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$. For any point $\omega \in \Omega$, $F(\cdot, \omega)$ is a sample function mapping index space $\mathcal{X}$ to space $\mathcal{Y}$, which we denote as f for notational simplicity.

For any finite index set $\mathbf{x}_{1:n} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, we can define the finite-dimensional marginal joint distribution over function values $\{F(\mathbf{x}_1), \cdots, F(\mathbf{x}_n)\}$. For example, Gaussian Processes have marginal distributions as multivariate Gaussians, and Student-t Processes (Shah et al., 2014) have marginal distributions as multivariate Student-t distributions.

Kolmogorov Extension Theorem (Øksendal, 2003) shows that a stochastic process can be characterized by marginals over all finite index sets. Specifically, for a collection of joint distributions $\rho_{\mathbf{x}_{1:n}}$, we can define a stochastic process F such that $\forall \mathbf{x}_{1:n}$, $\rho_{\mathbf{x}_{1:n}}$ is the marginal joint distribution of F at $\mathbf{x}_{1:n}$, if ρ satisfies the following two conditions:

Exchangeability For any permutation π of $\{1, \cdots, n\}$, $\rho_{\pi(\mathbf{x}_{1:n})}(\pi(y_{1:n})) = \rho_{\mathbf{x}_{1:n}}(y_{1:n})$.

Consistency For any $1 \leq m \leq n$, $\rho_{\mathbf{x}_{1:m}}(y_{1:m}) = \int \rho_{\mathbf{x}_{1:n}}(y_{1:n}) dy_{m+1:n}$.

It is easy to show that BNNs satisfy both conditions, therefore define stochastic processes.

2.3 Spectral Stein Gradient Estimator (SSGE)

When applying Bayesian methods to modern probabilistic models, especially those with neural networks as components (e.g., BNNs and deep generative models), it is often the case that we have to deal with intractable densities. Examples include the marginal distribution of a non-conjugate model (e.g., the output distribution of a BNN), and neural samplers such as GANs (Goodfellow et al., 2014). A shared property of these distributions is that they are defined by a easy sampling process, despite the intractable density. Such distributions are called implicit distributions (Huszár, 2017).

The Spectral Stein Gradient Estimator (SSGE) (Shi et al., 2018b) is a recently proposed method for estimating the log density derivative function of an implicit distribution, only requiring samples from the distribution. Specifically, given a continuous differentiable density $q(\mathbf{x})$, and a positive definite kernel $k(\mathbf{x}, \mathbf{x}')$ in the Stein class (Liu et al., 2016) of q, they show

$$\nabla_{x_i} \log q(\mathbf{x}) = -\sum_{j=1}^{\infty} \Big[\mathbb{E}_q \nabla_{x_i} \psi_j(\mathbf{x})\Big] \psi_j(\mathbf{x}), \tag{2}$$

where $\{\psi_j\}_{j \geq 1}$ is a series of eigenfunctions of k described by the Mercer's theorem: $k(\mathbf{x}, \mathbf{x}') = \sum_j \mu_j \psi_j(\mathbf{x}) \psi_j(\mathbf{x}')$. Then the Nyström method (Baker, 1997; Williams & Seeger, 2001) is used to approximate the eigenfunctions $\psi_j(\mathbf{x})$ and their derivatives. The final estimator is given by truncating the sum in Equation (2) and replacing the expectation by Monte Carlo estimates.

3 Functional Variational Bayesian Neural Networks

3.1 Functional Evidence Lower Bound (fELBO)

Given dataset $\mathcal{D} = (\mathbf{X}^D, \mathbf{y}^D)$, variational inference in weight space is based on the equivalence between maximizing the ELBO and minimizing the KL divergence from the true posterior. Now

we introduce functional variational inference, where the functional ELBO (fELBO) is defined with respect to a variational posterior over functions ($f : \mathcal{X} \to \mathcal{Y}$) rather than over weights.

$$\mathcal{L}(q) := \mathbb{E}_q[\log p(\mathbf{y}^D|f)] - \mathrm{KL}[q(f)||p(f)]. \tag{3}$$

Specifically, we assume a stochastic process prior $p(f)$ over functions. This could be a GP, but one could also use stochastic processes without closed-form marginal densities, such as distributions over piecewise linear functions. We consider a variational posterior $q_\phi(f) \in \mathcal{Q}$ defined in terms of a neural network with stochastic weights and/or stochastic inputs. Specifically, we sample a function from q by sampling a random noise ξ and reparameterize $f(\mathbf{x}) = g_\phi(\mathbf{x}, \xi)$. For example, the standard Bayesian neural networks with Gaussian weight distributions allow this reparameterization trick (Kingma & Welling, 2013). Note that because a *single* vector ξ is shared among all input locations, it corresponds to randomness in the *function*, rather than observation variance; hence, the sampling of ξ corresponds to epistemic, rather than aleatoric, uncertainty (Depeweg et al., 2017).

In Theorem 1, we prove that the functional KL divergence is equal to the supremum of marginal KL divergences over all finite sets of input locations $\mathbf{X} = \mathbf{x}_{1:n}$, which we term *measurement points*. Intuitively, this follows the Kolmogorov Extension Theorem in Section 2.2 which guarantees that we only need finite marginal distributions to define the stochastic process. Appendix A gives a full proof.

Theorem 1 (Functional KL Divergence)**.** *For two stochastic processes P, Q, the KL divergence is the supremum of marginal KL divergences over all finite subset of inputs $\mathbf{x}_{1:n}$:*

$$\mathrm{KL}[P\|Q] = \sup_{n,\, \mathbf{x}_{1:n}} \mathrm{KL}[P_{\mathbf{x}_{1:n}}\|Q_{\mathbf{x}_{1:n}}]. \tag{4}$$

fELBO Using this characterization of the functional KL divergence, we rewrite the fELBO:

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_q[\log p(\mathbf{y}^D|f)] - \sup_{\mathbf{X}} \mathrm{KL}[q(\mathbf{f}^{\mathbf{X}})||p(\mathbf{f}^{\mathbf{X}})] \\ &= \inf_{\mathbf{X}} \sum_{(\mathbf{x}^D, y^D)\in\mathcal{D}} \mathbb{E}_q[\log p(y^D|f(\mathbf{x}^D))] - \mathrm{KL}[q(\mathbf{f}^{\mathbf{X}})||p(\mathbf{f}^{\mathbf{X}})] \\ &:= \inf_{\mathbf{X}} \mathcal{L}_{\mathbf{X}}(q).\end{aligned} \tag{5}$$

Hence, maximizing $\mathcal{L}(q)$ can be viewed as a two-player game analogous to a GAN (Goodfellow et al., 2014): the generator is maximizing $\mathcal{L}_{\mathbf{X}}(q)$ with respect to q, and the discriminator is minimizing $\mathcal{L}_{\mathbf{X}}(q)$ with respect to $\mathbf{X}$.

Interestingly, we can show that if measurement points contain all training locations, $\mathcal{L}(q)$ is a lower bound of the log marginal likelihood $\log p(\mathcal{D})$. The proof is in Appendix B.4.

Theorem 2 (Lower Bound)**.** *If measurement points $\mathbf{X}$ contains all training input locations $\mathbf{X}^D$, then we have*

$$\mathcal{L}_{\mathbf{X}}(q) = \log p(\mathcal{D}) - \mathrm{KL}[q(\mathbf{f}^{\mathbf{X}})\|p(\mathbf{f}^{\mathbf{X}}|\mathcal{D})] \leq \log p(\mathcal{D}). \tag{6}$$

3.2 Functional Variational Inference

Now we present practical algorithms for functional variational inference.

Adversarial Functional Variational Inference Analogously to GANs[2], $\mathcal{L}(q)$ is likely to be infinite if the discriminator is completely unrestricted, because most stochastic process priors assign measure zero to the set of functions representable by a neural network (Arjovsky & Bottou, 2017). However, by limiting the capacity of the discriminator, we obtain a meaningful minimax objective which forces the generator to produce samples which resemble the true posterior. Most obviously, we can restrict the discriminator to choose measurement points of size N_m. Thus we have,

$$\max_{q\in\mathcal{Q}} \min_{|\mathbf{X}|=N_m} \mathcal{L}_{\mathbf{X}}(q). \tag{7}$$

To solve this minimax problem, we adopt simultaneous optimization analogously to GANs (Goodfellow et al., 2014). In the inner loop, we minimize $\mathcal{L}_{\mathbf{X}}(q)$ with respect to $\mathbf{X}$; in the outer loop, we maximize $\mathcal{L}_{\mathbf{X}}(q)$ with respect to the generator q.

[2]The ordinary GAN objective is typically infinite for an unrestricted discriminator because the generator typically generates from a submanifold of data space.

Sampling-Based Functional Variational Inference The minimax formulation given above can be difficult to train and slow to converge. Here we present sampling-based functional variational inference as an alternative which optimizes a single training objective. Specifically, we replace the minimization in Equation (7) with a sampling distribution c, and then maximize the expected $\mathcal{L}_{\mathbf{X}}(q)$.

However, without the minimization, the objective will be inclined to overfit on training locations because of the log likelihood term. We therefore let the measurement points $\mathbf{X}$ include a random subset $\mathbf{X}^{D_s}$ of training data and denote $\mathbf{X}^M = \mathbf{X} \backslash \mathbf{X}^{D_s}$. Then the sampling-based functional variational inference becomes,

$$\max_{q \in \mathcal{Q}} \mathbb{E}_{\mathbf{X}^M \sim c} \, \mathcal{L}_{\mathbf{X}^M, \mathbf{X}^{D_s}}(q). \tag{8}$$

where $\mathbf{X}^M$ are M points independently drawn from c.

To better understand the relationship between adversarial and sampling-based inference, we consider the idealized scenario where the measurement points in both approaches include all training locations, i.e., $\mathbf{X} = \{\mathbf{X}^D, \mathbf{X}^M\}$. Let $\mathbf{f}^M, \mathbf{f}^D$ be the function values at $\mathbf{X}^M, \mathbf{X}^D$, respectively. By Theorem 2,

$$\mathcal{L}_{\mathbf{X}^M, \mathbf{X}^D}(q) = \log p(\mathcal{D}) - \mathrm{KL}[q(\mathbf{f}^M, \mathbf{f}^D) \| p(\mathbf{f}^M, \mathbf{f}^D | \mathcal{D})]. \tag{9}$$

So maximizing $\mathcal{L}_{\mathbf{X}^M, \mathbf{X}^D}(q)$ is equivalent to minimizing the KL divergence from the true posterior on points $\mathbf{X}^M, \mathbf{X}^D$. Based on this, we have the following consistency theorem that helps justify the use of adversarial and sampling-based objectives with finite measurement points.

Theorem 3 (Consistency under finite measurement points)**.** *Assume that the true posterior $p(f|\mathcal{D})$ is a Gaussian process and the variational family $\mathcal{Q}$ is all Gaussian processes. We have the following results if $M > 1$ and $\mathrm{supp}(c) = \mathcal{X}$:*

$$\underbrace{\arg\max_{q \in \mathcal{Q}} \left\{ \min_{\mathbf{X}^M} \mathcal{L}_{\mathbf{X}^M, \mathbf{X}^D}(q) \right\}}_{\textit{Adversarial}} = \underbrace{\arg\max_{q \in \mathcal{Q}} \left\{ \mathbb{E}_{\mathbf{X}^M \sim c} \, \mathcal{L}_{\mathbf{X}^M, \mathbf{X}^D}(q) \right\}}_{\textit{Sampling-Based}} = p(f|\mathcal{D}) \tag{10}$$

The proof is given in Appendix B.5. Ideally, if all the training data instead of a subset is included in measurement points, we can show that both adversarial and sampling-based objectives lead to the same optimum, where the variational GP is identical to the true posterior GP.

3.3 KL Divergence Gradients

Computing the fELBO requires computing the likelihood term and the KL divergence. While the likelihood term is tractable, the KL divergence term remains intractable because we don't have an explicit formula for variational posterior density $q_\phi(\mathbf{f}^{\mathbf{X}})$. Note that $\nabla_\phi \mathrm{KL}[q_\phi(\mathbf{f}^{\mathbf{X}}) \| p(\mathbf{f}^{\mathbf{X}})]$ is

$$\mathbb{E}_q \left[\nabla_\phi \log q_\phi(\mathbf{f}^{\mathbf{X}}) \right] + \mathbb{E}_\xi \left[\nabla_\phi \mathbf{f}^{\mathbf{X}} (\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}}) - \nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})) \right]. \tag{11}$$

The first term (expected score function) in Equation (11) is zero, so we discard it. (But note that it may be useful as a control variate (Roeder et al., 2017).) Besides that, using parametric variational posterior, the gradients $\nabla_\phi \mathbf{f}^{\mathbf{X}}$ can be computed easily by backpropagation. It remains to estimate the log-density derivatives $\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}})$ and $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$. As introduced in Sec 2.3, the SSGE offers us a principled way to estimate the score function at both in-distribution samples and out-of-distribution samples. Therefore, we can apply SSGE to estimate the two derivatives above. Additionally, for priors with explicit densities like GPs (Rasmussen & Williams, 2006)[3], Student-t Processes (Shah et al., 2014) and Wishart Processes (Dawid, 1981), $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$ is analytic.

3.4 The Algorithm

Throughout this paper, we mainly investigate sampling-based functional variational inference. Now we present the whole algorithm for fBNNs in Algorithm 1. Because the log likelihood term has unbiased mini-batch estimations, we estimate $\mathcal{L}_{\mathbf{X}}(q)$ with mini-batch $\mathcal{D}_s$ for fast computation. Specifically, we are optimizing

$$\sum_{(\mathbf{x}, y) \in \mathcal{D}_s} \mathrm{E}_{q_\phi} \left[\log p(y | f(x)) \right] - \mathrm{KL}[q(\mathbf{f}^M, \mathbf{f}^{\mathcal{D}_s}) \| p(\mathbf{f}^{\mathcal{D}_s}, \mathbf{f}^M)], \tag{12}$$

which is also a proper lower bound of $\log p(\mathcal{D}_s)$, as shown in Theorem 2.

[3] Appendix D.1 introduces an additional fix to deal with the GP kernel matrix stability issue.

Algorithm 1 Functional Variational Bayesian Neural Networks (fBNNs)

Require: Dataset $\mathcal{D}$, sampling distribution c, variational posterior $g(\cdot)$, prior p (explicit or implicit).

while ϕ not converged **do**
  $\mathbf{X}^M \sim c$; $D_S \subset \mathcal{D}$ ▷ sample measurement points
  $\mathbf{f}_i = g([\mathbf{X}^M, \mathbf{X}^{D_S}], \xi_i; \phi),\ i = 1 \cdots k.$ ▷ sample k function values
  $\mathbf{g}_1 = \frac{1}{k}\sum_i \sum_{(x,y)} \nabla_\phi \log p(y|\mathbf{f}_i(x))$ ▷ compute log likelihood gradients
  $\mathbf{g}_2 = \mathrm{SSGE}(p, \mathbf{f}_{1:k})$ ▷ estimate KL gradients
  $\phi \leftarrow \text{Optimizer}(\phi, \mathbf{g}_1 - \mathbf{g}_2)$ ▷ update the parameters
end while

4 Related work

Bayesian neural networks Variational inference was first applied to neural networks by Peterson (1987) and Hinton & Van Camp (1993). More recently, Graves (2011) proposed a practical method for variational inference with fully factorized Gaussian posteriors which used a simple (but biased) gradient estimator. Improving on that work, Blundell et al. (2015) proposed an unbiased gradient estimator using the reparameterization trick of Kingma & Welling (2013). There has also been much work (Louizos & Welling, 2016; Sun et al., 2017; Zhang et al., 2018) on modelling the correlations between weights using more complex Gaussian variational posteriors. Some non-Gaussian variational posteriors have been proposed, such as Multiplicative Normalizing Flows (Louizos & Welling, 2017) and implicit distributions (Shi et al., 2018a). Neural networks with dropout were also interpreted as BNNs (Gal & Ghahramani, 2016; Gal et al., 2017). However, all these methods place priors over the network parameters. Often, spherical Gaussian priors are placed over the weights for convenience and the prior assumptions are not well-understood.

Structured Priors There have been other recent attempts to train BNNs with structured priors. Flam-Shepherd et al. (2017) trained a BNN prior to mimic a GP prior, but their method still required variational inference in weight space. Noise Contrastive Priors (Hafner et al., 2018) are somewhat similar in spirit to our work in that they use a random noise prior in the function space. However, instead of doing proper Bayesian inference, the prior is incorporated by adding a regularization term to the weight-space ELBO, and is not rich enough to encourage extrapolation and pattern discovery. Neural Processes (Garnelo et al., 2018) try to model any conditional distribution given arbitrary data points, which struggle to scale to even tens of dimensions. Variational Implicit Processes (Ma et al., 2018) specify BNN priors and approximately fit a GP for the posterior, thus the mis-specified prior problem also applies to their work.

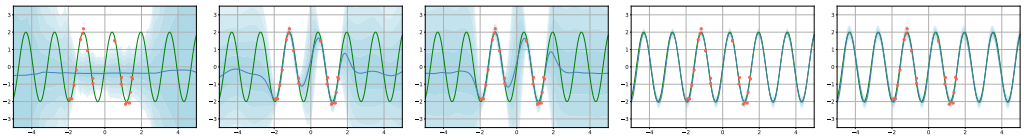
Scalable Gaussian Process Gaussian processes are difficult to apply exactly to large datasets since the computational requirements scale as $O(N^3)$ time, and as $O(N^2)$ memory, where N is the number of training cases. Multiple approaches have been proposed to reduce the computational complexity. However, sparse GP methods (Lázaro-Gredilla et al., 2010; Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013; 2015; Krauth et al., 2016) still suffer for very big dataset, while random feature method (Rahimi & Recht, 2008; Le et al., 2013) and KISS-GP (Wilson & Nickisch, 2015) are only applicable to specific kernels. In comparison to BNNs, GPs fail to provide explicit function samples.

5 Experiments

In this section, we present some experiments testing fBNNs' ability to extrapolate using a variety of structural motifs. We also evaluate its predictive performance on regression benchmarks and its ability to guide exploration in the contextual bandit setting.

In all of our experiments, the variational posterior is represented as a stochastic neural network with independent Gaussian distributions over the weights, i.e. $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$.[4] We always

[4]We note that other choices are possible, since functional variational inference imposes no requirement that epistemic uncertainty be represented in terms of a distribution over the weights of a network. (E.g., in principle,



(a) BBB (b) FBNN-RBF (c) GP-RBF (d) FBNN-PER (e) GP-PER

Figure 2: Predictions on periodic structures. Red dots are 20 training points. The green and blue lines represent ground truth and mean prediction, respectively. Shaded areas correspond to standard deviations. We used two kinds of kernels for fBNNs and GP: RBF and PER + RBF. Given PER + RBF kernel, fBNNs can recover periodic patterns; when RBF kernel is misused, fBNNs generate reasonable uncertainties beyond training data.

used the ReLU activation function unless otherwise specified. Denote $\mathrm{xmin}, \mathrm{xmax}$ as dimension-wise minimal and maximal input locations, $\mathrm{xd} = \mathrm{xmax} - \mathrm{xmin}$. We set the sampling distribution $c = \mathbb{U}[\mathrm{xmin} - \mathrm{xd}/2, \mathrm{xmax} + \mathrm{xd}/2]$ unless otherwise specified. For experiments where we used GP priors, we first fit the GP hyperparameters to maximize the marginal likelihood on subsets of the training examples, and then fixed those hyperparameters to obtain the prior for the fBNNs.

5.1 Extrapolation Using Structured Priors

Making sensible predictions outside the range of the observed data requires exploiting the underlying structure. In this section, we consider some toy datasets where fBNNs are able to use structured priors to make sensible extrapolations.

5.1.1 Learning Periodic Structures

Gaussian processes can model periodic structure using a periodic kernel plus a RBF kernel:

$$k(x, x') = \sigma_1^2 \exp\left\{-\frac{2\sin^2(\pi|x - x'|/p)}{l_1^2}\right\} + \sigma_1^2 \exp(-\frac{(x - x')^2}{2l_2^2}) \tag{13}$$

where p determines the period. In this experiment, we consider 20 inputs randomly sampled from the interval $[-2, -0.5] \cup [0.5, 2]$, and targets y which are noisy observations of a periodic function: $y = 2 * \sin(4x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.04)$. We used a BNN with five hidden layers, each with 500 units. The inputs and targets were normalized to have zero mean and unit variance. For all methods, the observation noise variance was set to the true value. We compared our method with Bayes By Backprop (BBB) (Blundell et al., 2015) (with a spherical Gaussian prior on $\mathbf{w}$) and Gaussian Processes. We used the trained GP as the prior of our fBNNs. We experimented with PER + RBF in eq. (13) and the single RBF kernel. In each iteration, measurement points included all training examples, plus 40 points randomly sampled from $[-5, 5]$. We used a training budget of 20,000 iterations, and annealed the weighting factor of the KL term from 0 to 1 linearly.

As shown in Fig. 2, BBB fails to fit the training data, let alone recover the periodic pattern (since its prior does not encode any periodic structure). GPs (as usual) fitted the data well and extrapolated the period structure beyond the range of the data; we can view GP predictions as a sort of gold standard. Compared to BBB, fBNNs were able to extrapolate similarly to GP, though they predicted slightly smaller uncertainty. Interestingly, we notice that fBNNs made smooth predictions despite the ReLU activation functions.

5.1.2 Implicit Priors

Given that fBNN is based on implicit variational inference (as introduced in Section 2.3), we note that the prior $p(f)$ doesn't need to have a tractable explicit density, which means we can also use implicit priors. In particular, we consider two implicit priors: distributions over piecewise constant functions, and distributions over piecewise linear functions. Appendix D.2 provides detailed explanations for the sampling process. Some samples from these priors are shown in Figure 3.

In each run of the experiment, we first sampled a random function from the prior, and then sampled 20 points from $[0, 0.2]$ and another 20 points from $[0.8, 1]$, giving a training set of 40 data points. The standard deviation of observation noise was chosen to be 0.02. In each iteration, we took all

a distribution over functions could be represented with a deterministic network with stochastic inputs.) However, we stick with Gaussian distributions on the weights because it worked as well as anything else we've tried.

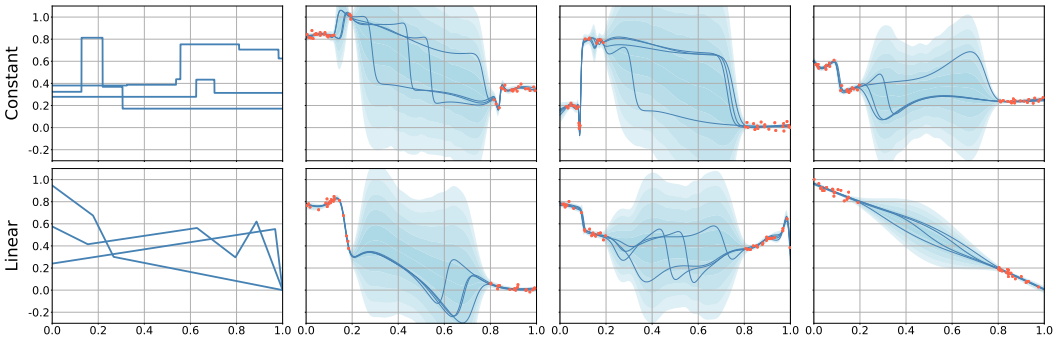


Figure 3: Implicit function priors and variational posteriors. The leftmost column shows 3 prior samples. Other columns are three different runs. Red dots are 40 training samples. We plot 4 posterior samples and plot the standard derivation as shaded areas.

training examples, together with 40 points randomly sampled from $[0, 1]$ as measurement points. We used a fully connected network with 2 hidden layers of 100 units, and Tanh activations. For this experiment, the reason why we used Tanh rather than ReLU is that the latter always generates piecewise linear functions. The network was trained for 20,000 iterations. We show the predictive samples and variances of three different runs in Figure 3.

As shown in Figure 3, fBNNs captured the basic structures for both piecewise constant priors and piecewise linear priors, though the posterior samples did not seem to capture the full diversity of possible explanations of the data. Surprisingly, even though the Tanh activation function typically encourages smoothness, the network learned to generate functions with sharp transitions.

5.2 Predictive Performance

5.2.1 Small Scale Datasets

Following previous work (Zhang et al., 2018), we then experimented with standard regression benchmark datasets from the UCI collection (Asuncion & Newman, 2007). In particular, we only used the datasets with less than 2000 data points so that we can fit GP hyperparameters by maximizing marginal likelihood exactly. Each dataset was randomly split into training and validation sets, comprising 90% and 10% of the data respectively. This splitting process was repeated 10 times to reduce randomness.

Table 1: Averaged validation RMSE and log-likelihood for the regression benchmarks.

	Validation RMSE			Validation log-likelihood		
Dataset	BBB	NNG	Ours	BBB	NNG	Ours
Boston	3.171±0.149	2.742±0.125	**2.378±0.104**	-2.602±0.031	-2.446±0.029	**-2.301±0.038**
Concrete	5.678±0.087	5.019±0.127	**4.935±0.180**	-3.149±0.018	**-3.039±0.025**	-3.096±0.016
Energy	0.565±0.018	0.485±0.023	**0.412±0.017**	-1.500±0.006	-1.421±0.005	**-0.684±0.020**
Wine	0.643±0.012	**0.637±0.011**	0.673±0.014	-0.977±0.017	**-0.969±0.014**	-1.040±0.013
Yacht	1.174±0.086	0.979±0.077	**0.607±0.068**	-2.408±0.007	-2.316±0.006	**-1.033±0.033**

For all datasets, we used networks with one hidden layer of 50 hidden units. For all experiments, we first fit GP hyper-parameters using marginal likelihood with a budget of 10,000 iterations. In each experiment, fBNNs were trained for 2,000 epochs. And in each iteration, measurement points included 20 training examples, plus 5 points randomly sampled from the data range.

We compared our method with Bayes By Backprop (BBB) (Blundell et al., 2015) and Noisy Natural Gradient (NNG) (Zhang et al., 2018). In accordance with (Zhang et al., 2018), we report the standard metrics including root mean square error (RMSE) and validation log-likelihood. The results are summarized in Table 1. On most datasets, our fBNNs outperformed both BBB and NNG, sometimes by a significant margin.

5.2.2 Large Scale Datasets

Observe that fBNNs are naturally scalable to large datasets because they access the data only through the expected log-likelihood term, which can be estimated stochastically. In this section, we verify

Table 2: Averaged validation RMSE and log-likelihood for the regression benchmarks.

		Test RMSE		Test log-likelihood	
Dataset	N	BBB	FBNN	BBB	FBNN
Naval	11934	1.6E-4±0.000	**1.2E-4±0.000**	6.950±0.052	**7.130±0.024**
Protein	45730	4.331±0.033	**4.326±0.019**	-2.892±0.007	**-2.892±0.004**
Video Memory	68784	1.879±0.265	**1.858±0.036**	**-1.999±0.054**	-2.038±0.021
Video Time	68784	3.632±1.974	**3.007±0.127**	**-2.390±0.040**	-2.471±0.018
GPU	241600	21.886±0.673	**19.50±0.171**	-4.505±0.031	**-4.400±0.009**

this experimentally. We compared fBNNs and BBB with large scale UCI datasets, including *Naval*, *Protein Structures*, *Video Transcoding (Memory, Time)* and *GPU kernel performance*. We randomly split the datasets into 80% training, 10% validation, and 10% test. We used the validating set to select the hyperparameters and performed early stopping.

Both methods were trained for 80,000 iterations.[5] We used 1 hidden layer with 100 hidden units for all datasets. For the prior of fBNNs, we used a GP with Neural Kernel Network (NKN) kernels as used in Sun et al. (2018). We note that GP hyperparameters were fitted using mini-batches of size 1000 with 10000 iterations. In each iteration, measurement sets consist of 500 training samples and 5 or 50 points from the sampling distribution c, tuned by validation performance. We ran each experiment 5 times, and report the mean and standard deviation in Table 2.

5.3 Contextual Bandits

Table 3: Contextual bandits regret. Results are relative to the cumulative regret of the Uniform algorithm. Numbers after the algorithm are the network sizes. We report the mean and standard derivation over 10 trials.

	M. Rank	M. Value	MushRoom	Statlog	Covertype	Financial	Jester	Adult	Census	Wheel
FBNN 1×50	5.875	46.0	21.38 ± 7.00	8.85 ± 4.55	47.16 ± 2.39	9.90 ± 2.40	75.55 ± 5.51	**88.43 ± 1.95**	51.43 ± 2.34	65.05 ± 20.10
FBNN 2×50	7.125	47.0	24.57 ± 10.81	10.08 ± 5.66	49.04 ± 3.75	11.83 ± 2.95	73.85 ± 6.82	88.81 ± 3.29	50.09 ± 2.74	67.76 ± 25.74
FBNN 3×50	8.125	48.9	34.03 ± 13.95	7.73 ± 4.37	50.14 ± 3.13	14.14 ± 1.99	74.27 ± 6.54	89.68 ± 1.66	52.37 ± 3.03	68.60 ± 22.24
FBNN 1×500	**4.875**	**45.3**	21.90 ± 9.95	6.50 ± 2.97	47.45 ± 1.86	**7.83 ± 0.77**	74.81 ± 5.57	89.03 ± 1.78	50.73 ± 1.53	63.77 ± 25.80
FBNN 2×500	**5.0**	**44.2**	23.93 ± 11.59	7.98 ± 3.08	46.00 ± 2.01	10.67 ± 3.52	**68.88 ± 7.09**	89.70 ± 2.01	51.87 ± 2.38	54.57 ± 32.92
FBNN 3×500	**4.75**	**44.6**	19.07 ± 4.97	10.04 ± 5.09	**45.24 ± 2.11**	11.48 ± 2.20	69.42 ± 7.56	90.01 ± 1.70	**49.73 ± 1.35**	61.57 ± 21.73
MultitaskGP	5.875	46.5	20.75 ± 2.08	7.25 ± 1.80	48.37 ± 3.50	8.07 ± 1.13	76.99 ± 6.01	88.64 ± 3.20	57.86 ± 8.19	64.15 ± 27.08
BBB 1×50	11.5	56.6	24.41 ± 6.70	25.67 ± 3.46	58.25 ± 5.00	37.69 ± 15.34	75.39 ± 6.32	95.07 ± 1.57	63.96 ± 3.95	72.37 ± 16.87
BBB 1×500	13.375	68.1	26.41 ± 8.71	51.29 ± 11.27	83.91 ± 4.62	57.20 ± 7.19	78.94 ± 4.98	99.21 ± 0.79	92.73 ± 9.13	55.09 ± 13.82
BBAlphaDiv	16.0	87.4	61.00 ± 6.47	70.91 ± 10.22	97.63 ± 3.21	85.94 ± 4.88	87.80 ± 5.08	99.60 ± 1.06	100.41 ± 1.54	95.75 ± 12.31
ParamNoise	10.125	53.0	20.33 ± 13.12	13.27 ± 2.85	65.07 ± 3.47	17.63 ± 4.27	74.94 ± 7.24	95.90 ± 2.20	82.67 ± 3.86	54.38 ± 16.20
NeuralLinear	10.375	52.3	16.56 ± 11.60	13.96 ± 1.51	64.96 ± 2.54	18.57 ± 2.02	82.14 ± 3.64	96.87 ± 0.92	78.94 ± 1.87	46.26 ± 8.40
LinFullPost	9.25	nan	14.71 ± 0.67	19.24 ± 0.77	58.69 ± 1.17	10.69 ± 0.92	77.76 ± 5.67	95.00 ± 1.26	Nan ± Nan	**33.88 ± 15.15**
Dropout	7.625	48.3	**12.53 ± 1.82**	12.01 ± 6.11	48.95 ± 2.19	14.64 ± 3.95	71.38 ± 7.11	90.62 ± 2.21	58.53 ± 2.35	77.46 ± 27.58
RMS	8.875	53.0	15.29 ± 3.06	11.38 ± 5.63	58.96 ± 4.97	10.46 ± 1.61	72.09 ± 6.98	95.29 ± 1.50	85.29 ± 5.85	75.62 ± 30.43
BootRMS	7.5	51.9	18.05 ± 11.20	**6.13 ± 1.03**	53.63 ± 2.15	8.69 ± 1.30	74.71 ± 6.00	94.18 ± 1.94	82.27 ± 1.84	77.80 ± 29.55
Uniform	16.75	100	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

One of the most important applications of uncertainty modelling is to guide exploration in settings such as bandits, Bayesian optimization (BO), and reinforcement learning. In this section, we evaluate fBNNs on a recently introduced contextual bandits benchmark (Riquelme et al., 2018). In contextual bandits problems, the agent tries to select the action with highest reward given some input context. Because the agent learns about the model gradually, it should balance between exploration and exploitation to maximize the cumulative reward. Thompson sampling (Thompson, 1933) has been shown to be promising by keeping a posterior of the model and selecting the action in proportion to the posterior distribution. Therefore, the key problem of Thompson sampling is to learn a reliable Bayesian uncertainty model.

We compared our fBNNs with the algorithms benchmarked in (Riquelme et al., 2018). We used the multi-task GP of Riquelme et al. (2018) as the prior for fBNNs. We run the experiments for all algorithms and tasks using the default settings open sourced by Riquelme et al. (2018). For fBNNs, we also kept the same setting, including batchsize (512), training epochs (100) and training frequency (50). Measurement sets consisted of training batches, combined with 10 points sampled from c. We ran each experiment 10 times and report the mean and standard derivation in Table 3. Similarly to Riquelme et al. (2018), we also report the mean rank and mean regret.

As shown in Table 3, fBNNs outperformed other methods by a wide margin. Additionally, fBNNs maintained consistent performance with various network sizes. By comparison, BBB suffered significant performance degeneration when the hidden size was increased from 50 to 500. This is

[5] We tune the learning rate from $[0.001, 0.01]$. We tuned between not annealing the learning rate or annealing it by 0.1 at 40000 iterations. We evaluated the validating set in each epoch, and selected the epoch for testing based on the validation performance.

consistent with our hypothesis that functional variational inference can gracefully handle networks with high capacity. We also show a BO experiment in App C.2, which also needs reliable uncertainty.

6 CONCLUSIONS

In this paper we investigated variational inference between stochastic processes. We proved that the KL divergence between stochastic processes equals the supremum of KL divergence for marginal distributions over all finite measurement sets. Then we presented two practical functional variational inference approaches: adversarial and sampling-based. Adopting BNNs as the variational posterior yields our functional variational Bayesian neural networks (fBNNs). Empirically, we demonstrated that fBNNs extrapolate well over various structures, estimate reliable uncertainties, and scale to large datasets.

REFERENCES

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

Christopher T. Baker. *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1997.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep gaussian processes. In *International Conference on Machine Learning*, pp. 884–893, 2017.

A Philip Dawid. Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, 68(1):265–274, 1981.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Uncertainty decomposition in Bayesian neural networks with latent variables. *arXiv preprint arXiv:1706.08495*, 2017.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping Gaussian process priors to Bayesian neural networks. In *NIPS Bayesian deep learning workshop*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017.

M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. In *International Conference on Machine Learning*, pp. 1744–1753, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Robert M. Gray. *Entropy and Infomation Theory*. Springer, 2011.

Danijar Hafner, Dustin Tran, Alex Irpan, Timothy Lillicrap, and James Davidson. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv preprint arXiv:1807.09289*, 2018.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, pp. 282, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360, 2015.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pp. 918–926, 2014.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Conference on Computational Learning Theory*, pp. 5–13, 1993.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Karl Krauth, Edwin V Bonilla, Kurt Cutajar, and Maurizio Filippone. AutoGP: Exploring the capabilities and limitations of Gaussian process models. *arXiv preprint arXiv:1610.05392*, 2016.

John Lamperti. *Stochastic processes: a survey of the mathematical theory*, volume 23. Springer Science & Business Media, 2012.

Miguel Lázaro-Gredilla, Joaquin Quiñonero Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881, 2010.

Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood — approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, volume 85, 2013.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pp. 276–284, 2016.

James Robert Lloyd, David K Duvenaud, Roger B Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *AAAI*, pp. 1242–1250, 2014.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning*, pp. 1708–1716, 2016.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*, pp. 2218–2227, 2017.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

Chao Ma, Yingzhen Li, and José Miguel Hernández-Lobato. Variational implicit processes. *arXiv preprint arXiv:1806.02390*, 2018.

Anton Mallasto and Aasa Feragen. Learning from uncertain curves: The 2-wasserstein metric for Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 5660–5670, 2017.

Alexander G de G Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Artificial Intelligence and Statistics*, pp. 231–239, 2016.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1: 995–1019, 1987.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2008.

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems*, pp. 294–300, 2001.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pp. 6925–6934, 2017.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 4588–4599, 2017.

Amar Shah, Andrew Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 877–885, 2014.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel implicit variational inference. In *International Conference on Learning Representations*, 2018a.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. *International Conference on Machine Learning*, 2018b.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2006.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in Bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li, and Roger Grosse. Differentiable compositional kernel learning for Gaussian processes. In *International Conference on Machine Learning*, pp. 4828–4837, 2018.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning*, pp. 3627–3635, 2017.

Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pp. 682–688, 2001.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pp. 1775–1784, 2015.

Andrew G Wilson, Elad Gilboa, Arye Nehorai, and John P Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, pp. 3626–3634, 2014.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861, 2018.

A STOCHASTIC PROCESSES

A.1 BASIC MEASURE THEORY NOTATIONS

π-system A π-system is a collection of subsets that is closed under finitely many intersections.

σ-algebra generated by cylinder sets Suppose

$$X \subset \mathbb{R}^{\mathbb{T}} = \{f : f(t) \in \mathbb{R}, t \in \mathbb{T}\}$$

is a set of real-valued functions. Let $\mathcal{B}(\mathbb{R})$ denote the Borel subsets of $\mathbb{R}$. A cylinder subset of X is a finitely restricted set defined as

$$C_{t_1,\cdots,t_n}(B_1, \cdots, B_n) = \{f \in X : f(t_i) \in B_i, 1 \leq i \leq n\}$$

Each

$$\mathcal{G}_{t_1,\cdots,t_n} = \{C_{t_1,\cdots,t_n}(B_1, \cdots, B_n) : B_i \in \mathcal{B}(\mathbb{R}), 1 \leq i \leq n\}$$

is a π-system that generates a σ-algebra $\Sigma_{t_1,\cdots,t_n}$. Then the family of subsets

$$\mathcal{F}_X = \bigcup_{n=1}^{\infty} \bigcup_{t_i \in \mathbb{T}, i \leq n} \Sigma_{t_1,\cdots,t_n}$$

is an algebra that generates the cylinder σ-algebra for X.

Definition 1 (KL divergence (Gray, 2011))**.** *Given two probability measures q and p on measurable space $(\Omega, \mathscr{B})$, the KL divergence of q with respect to p is defined as*

$$\mathrm{KL}[q\|p] = \sup_{Q} \mathrm{KL}_Q[q\|p]. \tag{14}$$

where the supremum is over all finite measurable partitions Q of Ω.

A.2 PUSHFORWARD AND KOLOMOGROV EXTENSION THEOREM

Definition 2 (Pushforward measure)**.** *Given probability spaces (X, Σ_X, μ) and (Y, Σ_Y, ν), we say that measure ν is a pushforward of μ if $\nu(A) = \mu(T^{-1}(A))$ for a measurable $T : X \to Y$ and any $A \in \Sigma_Y$, denoted by $\nu = \mu \circ T^{-1}$.*

Definition 3 (Canonical projection map)**.** *Let T be an arbitrary index set, and $\{(\Omega_t, \mathcal{F}_t)\}_{t \in T}$ be some collection of measurable spaces. For each subset $J \subset I \subset T$, define $\Omega^J = \prod_{t \in J} \Omega_t$. We call $\pi_{I \to J}$ the canonical projection map from I to J if*

$$\pi_{I \to J}(w) = w|_J \in \Omega^J, \forall w \in \Omega^I. \tag{15}$$

Theorem 4 (Kolmogorov extension theorem)**.** *Let T be an arbitrary index set. $(\Omega, \mathcal{F})$ is a standard measurable space, whose product space on T is $(\Omega^T, \mathcal{F}^T)$. For each finite subset $I \subset T$, suppose we have a probability measure μ_I on Ω^I that satisfies the following compatibility relation: For each finite subset $J \subset I \subset T$, we have*

$$\mu_J = \mu_I \circ \pi^{-1}_{I \to J}. \tag{16}$$

Then there exists a unique probability measure μ on Ω^T such that $\forall$ finite subset $I \subset T$,

$$\mu_I = \mu \circ \pi^{-1}_{T \to I}. \tag{17}$$

In Gaussian processes, μ is a Borel measure on a separable Hilbert space $(\mathcal{H}, \mathscr{B}(\mathcal{H}))$. μ_I is a marginal Gaussian measure at any finite set (i.e., I) of input positions (Mallasto & Feragen, 2017).

A.3 PROOF

We first introduce several lemmas for proving the theorem.

Lemma 5.

$$\mathcal{G}_X = \bigcup_{n=1}^{\infty} \bigcup_{t_i \in \mathbb{T}, i \leq n} \mathcal{G}_{t_1,\cdots,t_n}$$

is a π-system. And $\sigma(\mathcal{G}_X)$ generates the same cylinder σ-algebra of X as $\mathcal{F}_X$.

Definition 4. *Let H be a set of functions, define its $\mathbb{T}$-set $\tau(H)$ as*

$$\tau(H) = \mathbb{T}\backslash\{t|t \in \mathbb{T}, \exists S_{\mathbb{T}\backslash\{t\}}, s.t.\ S_{\mathbb{T}\backslash\{t\}} \times \mathbb{R}_t = H\}$$

Here $\times$ represents product-algebra. Intuitively, $\tau(H)$ represents all timesteps that restricts the functions in H. Or we can say if some t doesn't belong to $\tau(H)$, then function values of t can be any number in real line $\mathbb{R}$.

Lemma 6. *For any set $G \in \sigma(\mathcal{G}_X)$, its $\mathbb{T}$-set $\tau(G)$ is countable.*

Proof. We build the sigma-algebra $\sigma(\mathcal{G}_X)$ through an iterative process. We begin by $\mathcal{H}_0 = \mathcal{G}_X$, in each iteration, let

$$\begin{aligned} \mathcal{H}_{i+1} &= \mathcal{H}_i \cup \{H^c | H \in \mathcal{H}_i\}, \\ \mathcal{H}_{i+1} &= \mathcal{H}_{i+1} \cup \{\bigcup_j H_j | H_1, H_2, \cdots \in \mathcal{H}_i\}. \end{aligned} \tag{18}$$

Recursively applying this extension, we can generate $\sigma(\mathcal{G}_X) = \mathcal{H}_\infty$.

Because $\tau(H^c) = \tau(H)$, and countable union of countable sets is still countable. We know the two updates above keep the countability of its $\mathbb{T}$-set. Therefore, $\forall G \in \sigma(\mathcal{G}_X), \tau(G)$ is countable.

□

Theorem 7. *For two stochastic processes P, M on functional measurable space $(\Omega^{\mathbb{T}}, \mathcal{F}^{\mathbb{T}})$, the KL divergence between P and M,*

$$\mathrm{KL}[P\|M] = \sup_{\mathbb{T}_f} \mathrm{KL}[P_{\mathbb{T}_f}\|M_{\mathbb{T}_f}],$$

Where the supremum is over all finite subsets, and $P_{\mathbb{T}_f}, M_{\mathbb{T}_f}$ represent the marginal distribution of P, M on $\mathbb{T}_f$, respectively.

Proof. **Sketches**

1. The cylinder σ-algebra defines a σ-algebra $\sigma(\mathcal{G}_X)$ on the function space. By Lemma 5.
2. Each set in the $\sigma(\mathcal{G}_X)$ is only related to countable number of timesteps. By Lemma 6.
3. Any finite measurable partitions are related to countable number of timesteps, thus correspond to the marginal distribution over a countable subset by an inclusion map. Therefore the KL between stochastic processes reduces to the KL over marginal distributions of countable subsets.
4. KL over marginal distributions of countable subsets can be represented as supremum of KL over marginal distributions over its finite subsets.

By Definition 1,

$$\mathrm{KL}[P\|M] = \sup_{Q_{\Omega^{\mathbb{T}}}} \mathrm{KL}_{Q_{\Omega^{\mathbb{T}}}}[P\|M],$$

where the sup is over all finite measurable partitions of the function space, denoted by $Q_{\Omega^{\mathbb{T}}}$:

$$Q_{\Omega^{\mathbb{T}}} = \{Q^{(1)}_{\Omega^{\mathbb{T}}}, \ldots, Q^{(k)}_{\Omega^{\mathbb{T}}} | \bigcup_{i=1}^{k} Q^{(i)}_{\Omega^{\mathbb{T}}} = \Omega^{\mathbb{T}}, Q^{(i)}_{\Omega^{\mathbb{T}}} \in \mathcal{F}^{\mathbb{T}} \text{ are disjoint sets}, k \in \mathbb{N}^+\}$$

By Lemma 6, $\tau(Q^{(i)}_{\Omega^{\mathbb{T}}})$ is countable. Therefore, $\mathbb{T}_c := \bigcup_{i=1}^{k} \tau(Q^{(i)}_{\Omega^{\mathbb{T}}})$ is countable.

Consider the canonical projection mapping $\pi_{\mathbb{T}\to\mathbb{T}_c}$, which induces a partition on $\Omega^{\mathbb{T}_c}$, denoted by $Q_{\Omega^{\mathbb{T}_c}}$:

$$Q^{(i)}_{\Omega^{\mathbb{T}_c}} = \pi_{\mathbb{T}\to\mathbb{T}_c}(Q^{(i)}_{\Omega^{\mathbb{T}}}).$$

The pushforward measure defined by this mapping is

$$P_{\mathbb{T}_c} = P \circ \pi^{-1}_{\mathbb{T}\to\mathbb{T}_c}, \quad M_{\mathbb{T}_c} = M \circ \pi^{-1}_{\mathbb{T}\to\mathbb{T}_c}.$$

Then we have

$$\mathrm{KL}[P\|M] = \sup_{Q_{\Omega^{\mathbb{T}}}} \mathrm{KL}_{Q_{\Omega^{\mathbb{T}}}}[P\|M] \tag{19}$$

$$= \sup_{Q_{\Omega^{\mathbb{T}}}} \sum_i P(Q^{(i)}_{\Omega^{\mathbb{T}}}) \log \frac{P(Q^{(i)}_{\Omega^{\mathbb{T}}})}{M(Q^{(i)}_{\Omega^{\mathbb{T}}})} \tag{20}$$

$$= \sup_{\mathbb{T}_c} \sup_{Q_{\Omega^{\mathbb{T}_c}}} \sum_i P_{\mathbb{T}_c}(Q^{(i)}_{\Omega^{\mathbb{T}_c}}) \log \frac{P_{\mathbb{T}_c}(Q^{(i)}_{\Omega^{\mathbb{T}_c}})}{M_{\mathbb{T}_c}(Q^{(i)}_{\Omega^{\mathbb{T}_c}})} \tag{21}$$

$$= \sup_{\mathbb{T}_c} \mathrm{KL}[P_{\mathbb{T}_c}\|M_{\mathbb{T}_c}] \tag{22}$$

Denote $\mathcal{F}(\mathbb{T}_c)$ as the collection of all finite subsets of $\mathbb{T}_c$. For any finite set $\mathbb{T}_j \in \mathcal{F}(\mathbb{T}_c)$, we denote $P_{\mathbb{T}_j}$ as the pushforward of P on finite set $\mathbb{T}_j$. According to Kolmogorov Extension Theorem (Theorem 4), $P_{\mathbb{T}_j}$ is the marginal distribution of P at $\mathbb{T}_j$. Because $\mathbb{T}_c$ is countable, we have

$$\mathrm{KL}[P\|M] = \sup_{\mathbb{T}_c} \mathrm{KL}[P_{\mathbb{T}_c}\|M_{\mathbb{T}_c}] \tag{23}$$

$$= \sup_{\mathbb{T}_c} \sup_{\mathbb{T}_j \in \mathcal{F}(\mathbb{T}_c)} \mathrm{KL}[P_{\mathbb{T}_j}\|M_{\mathbb{T}_j}] \tag{24}$$

Note for any finite subset $\mathbb{T}_j$, we can build a finite measurable partition $Q_{\Omega^{\mathbb{T}}}$ such that its $\mathbb{T}_c$ equals to $\mathbb{T}_j$. Let Ω_1, Ω_2 be any two-element partition of Ω. In fact, we can let $Q_{\Omega^{\mathbb{T}}}$ be a partition with $2^{|\mathbb{T}_j|}$ subsets, that each subset is a cylinder subset with timesteps in $\mathbb{T}_j$ and for each timestep, the function values belong to Ω_1 or Ω_2. Therefore, we have shown the set $\{\mathbb{T}_j | \mathbb{T}_j \in \mathcal{F}(\mathbb{T}_c), \mathbb{T}_c\}$ contains all finite subsets.

On the other hand, every element in $\{\mathbb{T}_j | \mathbb{T}_j \in \mathcal{F}(\mathbb{T}_c), \mathbb{T}_c\}$ is a finite subset, we know $\{\mathbb{T}_j | \mathbb{T}_j \in \mathcal{F}(\mathbb{T}_c), \mathbb{T}_c\}$ equals to the collection of all finite subsets.

Therefore we have proven the theorem. □

B Additional Theoretical Results

B.1 KL Divergence between Conditional Stochastic Processes

In this section, we give an example of computing the KL divergence between two conditional stochastic processes. Consider two datasets $\mathcal{D}_1, \mathcal{D}_2$, the KL divergence between two conditional stochastic processes is

$$\begin{aligned}
\mathrm{KL}[p(f|\mathcal{D}_1)\|p(f|\mathcal{D}_2)] &= \sup_{n, \mathbf{x}_{1:n}} \mathrm{KL}[p(\mathbf{f}^{\mathbf{x}}|\mathcal{D}_1)\|p(\mathbf{f}^{\mathbf{x}}|\mathcal{D}_2)] \\
&= \sup_{n, \mathbf{x}_{1:n}} \mathbb{E}_{p(\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1)} \log \frac{p(\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1)}{p(\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_2)} \\
&= \sup_{n, \mathbf{x}_{1:n}} \mathbb{E}_{p(\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1)} \log \frac{p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1) p(\mathbf{f}^{\mathbf{x}}|\mathbf{f}^{D_1 \cup D_2}, \mathcal{D}_1)}{p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_2) p(\mathbf{f}^{\mathbf{x}}|\mathbf{f}^{D_1 \cup D_2}, \mathcal{D}_2)} \\
&= \sup_{n, \mathbf{x}_{1:n}} \mathbb{E}_{p(\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1)} \log \frac{p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1) p(\mathbf{f}^{\mathbf{x}}|\mathbf{f}^{D_1 \cup D_2})}{p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_2) p(\mathbf{f}^{\mathbf{x}}|\mathbf{f}^{D_1 \cup D_2})} \\
&= \mathrm{KL}[p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_1)\|p(\mathbf{f}^{D_1 \cup D_2}|\mathcal{D}_2)]
\end{aligned} \tag{25}$$

Therefore, the KL divergence between these two stochastic processes equals to the marginal KL divergence on the observed locations. When $\mathcal{D}_2 = \emptyset$, $p(f|\mathcal{D}_2) = p(f)$, this shows the KL divergence between posterior process and prior process are the marginal KL divergence on observed locations.

This also justifies our usage of M measurement points in the adversarial functional VI and sampling-based functional VI of Section 3.

B.2 FUNCTION-SPACE INFERENCE VERSUS WEIGHT-SPACE INFERENCE

In this section, we compare function-space inference with weight-space inference. We study the setting that we have a weight-space prior which matches the functional prior in function space. Despite that prior is not a problem anymore under this setting, we show that performing inference in function space is still superior to performing inference in weight space.

Consider a deep linear network with weights $\{\mathbf{W}_l\}_{l=1}^{L-1}$, the network defines a function:

$$f(\mathbf{x}) = \mathbf{W}_{L-1} \cdots \mathbf{W}_1 \mathbf{W}_0 \mathbf{x} = \prod_{l=0}^{L-1} \mathbf{W}_l \mathbf{x} =: \mathcal{W}\mathbf{x} \tag{26}$$

Given dataset $\mathcal{D} = (\mathbf{X}^{\mathcal{D}}, \mathbf{y}^{\mathcal{D}})$ and weight-space prior $p_w(\mathbf{W}_0, \cdots, \mathbf{W}_{L-1})$, variational inference is adopted to approximate the intractable posterior.

Weight-Space Variational Inference. Assuming a variational posterior $q_w(\mathbf{W}_0, \cdots, \mathbf{W}_{L-1})$,

$$\mathcal{L}_w = \mathbb{E}_{q_w} \log p(\mathbf{y}^{\mathcal{D}} | \mathbf{X}^{\mathcal{D}}, \prod_{l=0}^{L-1} \mathbf{W}_l) - \mathrm{KL}[q([\mathbf{W}_0, \cdots, \mathbf{W}_{L-1}]) \| p([\mathbf{W}_0, \cdots, \mathbf{W}_{L-1}])] \tag{27}$$

Function-Space Variational Inference. Because $f(\mathbf{x}) = \prod_{l=0}^{L-1} \mathbf{W}_l \mathbf{x} =: \mathcal{W}\mathbf{x}$, the weight-space prior p_w corresponds to a prior $p_{\mathcal{W}}(\mathcal{W})$, thus corresponds to a functional prior $p_f(f)$. Similarly, the weight-space variational posterior q_w corresponds to a prior $q_{\mathcal{W}}(\mathcal{W})$, thus corresponds to a functional prior $q_f(f)$.

Consider the functional variational inference,

$$\mathcal{L}_f = \mathbb{E}_{q_f} \log p(\mathbf{y}^{\mathcal{D}} | \mathbf{X}^{\mathcal{D}}, f) - \mathrm{KL}[q_f(f) \| p_f(f)] \tag{28}$$

$$= \mathbb{E}_{q_w} \log p(\mathbf{y}^{\mathcal{D}} | \mathbf{X}^{\mathcal{D}}, \prod_{l=0}^{L-1} \mathbf{W}_l) - \mathrm{KL}[q_f(f) \| p_f(f)] \tag{29}$$

Theorem 8 (ELBO Ordering)**.** *The functional ELBO is no smaller than the weight-space ELBO.*

$$\mathcal{L}_f \geq \mathcal{L}_w \tag{30}$$

Lemma 9 (Linear Processes KL Divergence)**.** *Consider functions $f(\mathbf{x}) = \mathcal{W}\mathbf{x} : \mathbb{R}^d \to \mathbb{R}^o$ and two distributions $p_{\mathcal{W}}(\mathcal{W}), q_{\mathcal{W}}(\mathcal{W})$. Equivalently, the two distributions correspond to function distributions $p_f(f), q_f(f)$, respectively. Then the functional KL divergence*

$$\mathrm{KL}[q_f(f) \| p_f(f)] = \mathrm{KL}[q_{\mathcal{W}}(\mathcal{W}) \| p_{\mathcal{W}}(\mathcal{W})] \tag{31}$$

Proof of Lemma 9. By Theorem 1, the functional KL divergence is given by the supremum over all finite sets of input locations,

$$\begin{aligned}
\mathrm{KL}[q_f(f) \| p_f(f)] &= \sup_{n, \mathbf{x}_{1:n}} \mathrm{KL}[q(\mathbf{f}_{\mathbf{x}_{1:n}}) \| p(\mathbf{f}_{\mathbf{x}_{1:n}})] \\
&= \sup_{n, \mathbf{x}_{1:n}} \mathrm{KL}[q([\mathcal{W}\mathbf{x}_1, \cdots \mathcal{W}\mathbf{x}_n]) \| p([\mathcal{W}\mathbf{x}_1, \cdots \mathcal{W}\mathbf{x}_n])] \\
&= \sup_{n, \mathbf{x}_{1:n}} \mathrm{KL}[q_{\mathcal{W}}(\mathcal{W}) \| p_{\mathcal{W}}(\mathcal{W})] \\
&= \mathrm{KL}[q_{\mathcal{W}}(\mathcal{W}) \| p_{\mathcal{W}}(\mathcal{W})]
\end{aligned} \tag{32}$$

Where the second last equation comes from the bijection between $\mathcal{W}$ and $[\mathcal{W}\mathbf{x}_1, \cdots \mathcal{W}\mathbf{x}_n]$, where KL divergence keeps unchanged with bijection mapping (In rare cases $\mathbf{x}_{1:n}$ don't have sufficient rank to support a bijection, the KL is smaller than $\mathrm{KL}[q_{\mathcal{W}}(\mathcal{W}) \| p_{\mathcal{W}}(\mathcal{W})]$ thus diminishes with sup). □

Proof of Theorem 8. Based on Lemma 9, the functional ELBO

$$\begin{aligned}\mathcal{L}_f &= \mathbb{E}_{q_w} \log p(\mathbf{y}^{\mathcal{D}}|\mathbf{X}^{\mathcal{D}}, \prod_{l=0}^{L-1} \mathbf{W}_l) - \mathrm{KL}[q_f(f)\|p_f(f)] \\ &= \mathbb{E}_{q_w} \log p(\mathbf{y}^{\mathcal{D}}|\mathbf{X}^{\mathcal{D}}, \prod_{l=0}^{L-1} \mathbf{W}_l) - \mathrm{KL}[q_{\mathcal{W}}(\mathcal{W})\|p_{\mathcal{W}}(\mathcal{W})] \\ &\geq \mathbb{E}_{q_w} \log p(\mathbf{y}^{\mathcal{D}}|\mathbf{X}^{\mathcal{D}}, \prod_{l=0}^{L-1} \mathbf{W}_l) - \mathrm{KL}[q([\mathbf{W}_0, \cdots, \mathbf{W}_{L-1}])\|p([\mathbf{W}_0, \cdots, \mathbf{W}_{L-1}])] \\ &= \mathcal{L}_w\end{aligned}$$

Where the inequality stems from the fact that $\mathcal{W} = \prod_{l=0}^{L-1} \mathbf{W}_l$. □

Example 1 (Deep Addition Networks)**.** *Similar to deep linear networks, deep addition networks performs addition in each layer, thus it defines the function $f : \mathbb{R} \to \mathbb{R}$ that*

$$f(x) = x + w_0 + \cdots + w_{L-1} = x + \sum_{l=0}^{L-1} w_l =: x + w \tag{33}$$

Consider Gaussian weights prior $p_w^l(w_l) = \mathcal{N}(0, \frac{\eta^2}{L}), l = 0 : L-1$, and Gaussian weights variational posterior $q_w^l(w_l) = \mathcal{N}(\mu_l, \sigma_l^2), l = 0 : L-1$. They are equivalent to the prior $p_w(w) = \mathcal{N}(0, \eta^2)$ and the variational posterior $q_w(w) = \mathcal{N}(\sum_l \mu_l, \sum_l \sigma_l^2)$. The likelihood term is

$$y = \mathcal{N}(f(x), \nu^2) \tag{34}$$

Assume we have a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$,

Function-Space Inference *The exact posterior over w is a Gaussian distribution, the same as $q_w(w)$. Therefore the optimal variational posterior using functional inference is the true posterior*

$$p(w|\mathcal{D}) = \mathcal{N}(\frac{\eta^2 \sum_i (y_i - x_i)}{n\eta^2 + \nu^2}, \frac{\nu^2\eta^2}{n\eta^2 + \nu^2}) \tag{35}$$

Weight-Space Inference *Under factorized Gaussian variational posterior, we can get the optimal q_w^l as*

$$q_w^{l\star}(w_l) = \mathcal{N}(\frac{1}{L}\frac{\eta^2 \sum_i (y_i - x_i)}{n\eta^2 + \nu^2}, \frac{\nu^2\eta^2}{n\eta^2 + L\nu^2}) \tag{36}$$

Therefore, the aggregated w distribution using weight-space inference becomes

$$q_w^{\star}(w) = \mathcal{N}(\frac{\eta^2 \sum_i (y_i - x_i)}{n\eta^2 + \nu^2}, \frac{\nu^2\eta^2}{n\eta^2/L + \nu^2}) \tag{37}$$

Equation (37) and Equation (35) have the same mean, but different covariances. Considering using deeper networks for weight-space inference, ν^2 will finally dominate $n\eta^2/L$. In the limit that $L \to \infty$, the variance will become prior variance.

B.3 Optimize GP Hyperparameters using Mini-Batch

Consider multivariate Gaussian distribution,

$$\log p(\mathbf{y}|\mathbf{0}, \mathbf{K}_\theta) = -\frac{1}{2}(\mathbf{y}^\top \mathbf{K}_\theta^{-1}\mathbf{y} + \log|\mathbf{K}_\theta|) + \text{const}$$

Its derivative towards $\mathbf{K}_\theta$ is $-\frac{1}{2}\mathbf{K}_\theta^{-1}(\mathbf{K}_\theta - \mathbf{y}\mathbf{y}^\top)\mathbf{K}_\theta^{-1}$, which attains zero only at $\mathbf{K}^\star = \mathbf{y}\mathbf{y}^\top$. Therefore we have

$$\log p(\mathbf{y}|\mathbf{0}, \mathbf{K}_\theta) \leq \log p(\mathbf{y}|\mathbf{0}, \mathbf{K}^\star) \tag{38}$$

Gaussian process optimization maximizes data likelihood $\log p(\mathbf{y}|\mathbf{0}, \mathbf{K}_\theta)$. According to Equation (38), its maximum is obtained at $\mathbf{K}^\star = \mathbf{y}\mathbf{y}^\top$. In contrast, mini-batch GP optimization maximizes expected subset likelihood $\mathbb{E}_S\left[\log p(\mathbf{y}_S|\mathbf{0}, \mathbf{K}_\theta^S)\right]$, with S being a subset. According to Equation (38), $\log p(\mathbf{y}_S|\mathbf{0}, \mathbf{K}_\theta^S) \leq \log p(\mathbf{y}_S|\mathbf{0}, \mathbf{y}_S\mathbf{y}_S^\top)$. Therefore, its maximum is also obtained at $\mathbf{K}^\star = \mathbf{y}\mathbf{y}^\top$.

Based on the results above, for expressive enough kernels, mini-batch optimized GP reaches the same maximum as full-batch optimized GP.

B.4 Proof for Evidence Lower Bound

This section provides proof for Theorem 2.

Proof of Theorem 2. Let $\mathbf{X}^M = \mathbf{X} \backslash \mathbf{X}^D$ be measurement points which aren't in the training data.

$$\begin{aligned}
\mathcal{L}_{\mathbf{X}}(q) &= \mathbb{E}_q[\log p(\mathbf{y}^D|\mathbf{f}^D) + \log p(\mathbf{f}^{\mathbf{X}}) - \log q_\phi(\mathbf{f}^{\mathbf{X}})] \\
&= \mathbb{E}_q[\log p(\mathbf{y}^D|\mathbf{f}^D) + \log p(\mathbf{f}^D, \mathbf{f}^M) - \log q_\phi(\mathbf{f}^D, \mathbf{f}^M)] \\
&= \log p(\mathcal{D}) - \mathbb{E}_q\left[\log \frac{q_\phi(\mathbf{f}^D, \mathbf{f}^M)p(\mathcal{D})}{p(\mathbf{y}^D|\mathbf{f}^D)p(\mathbf{f}^D, \mathbf{f}^M)}\right] \\
&= \log p(\mathcal{D}) - \mathbb{E}_q\left[\log \frac{q_\phi(\mathbf{f}^D, \mathbf{f}^M)}{p(\mathbf{f}^D, \mathbf{f}^M|\mathcal{D})}\right] \\
&= \log p(\mathcal{D}) - \mathrm{KL}[q_\phi(\mathbf{f}^D, \mathbf{f}^M)\|p(\mathbf{f}^D, \mathbf{f}^M|\mathcal{D})] \qquad (39)
\end{aligned}$$

□

B.5 Consistency for Gaussian Processes

This section provides proof for consistency in Theorem 3.

Proof of Theorem 3. By the assumption that both $q(\mathcal{D})$ and $p(f|\mathcal{D})$ are Gaussian processes:

$$p(f(\cdot)|\mathcal{D}): \quad \mathcal{GP}(m_p(\cdot), k_p(\cdot,\cdot)), \qquad (40)$$

$$q(f(\cdot)): \quad \mathcal{GP}(m_q(\cdot), k_q(\cdot,\cdot)), \qquad (41)$$

where m and k denote the mean and covariance functions, respectively.

In this theorem, we also assume the measurement points cover all training locations as in Equation (9), where we have (based on Theorem 2):

$$\mathcal{L}_{\mathbf{X}}(q) = \log p(\mathcal{D}) - \mathrm{KL}[q(\mathbf{f}^D, \mathbf{f}^M)\|p(\mathbf{f}^D, \mathbf{f}^M|\mathcal{D})] \leq \log p(\mathcal{D}). \qquad (42)$$

Therefore, when the variational posterior process is sufficiently expressive and reaches its optimum, we must have $\mathrm{KL}[q(\mathbf{f}^D, \mathbf{f}^M)\|p(\mathbf{f}^D, \mathbf{f}^M|\mathcal{D})] = 0$ and thus $\mathrm{KL}[q(\mathbf{f}^M)\|p(\mathbf{f}^M|\mathcal{D})] = 0$ at $\mathbf{X}^M = [\mathbf{x}_1, \ldots, \mathbf{x}_M]^\top \in \mathcal{X}^M$, which implies

$$\mathcal{N}(m_p(\mathbf{X}^M), k_p(\mathbf{X}^M, \mathbf{X}^M)) = \mathcal{N}(m_q(\mathbf{X}^M), k_q(\mathbf{X}^M, \mathbf{X}^M)). \qquad (43)$$

Here $m(\mathbf{X}) = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_M)]^\top$, and $\left[k(\mathbf{X}^M, \mathbf{X}^M)\right]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Remember that in sampling-based functional variational inference, $\mathbf{X}^M$ are randomly sampled from $c(\mathbf{x})$, and $\mathrm{supp}(c) = \mathcal{X}$. Thus when it reaches optimum, we have $\mathbb{E}_{\mathbf{X}^M \sim c}\mathrm{KL}[q(\mathbf{f}^M)\|p(\mathbf{f}^M|\mathcal{D})] = 0$. Because the KL divergence is always non-negative, we have that $\mathrm{KL}[q(\mathbf{f}^M)\|p(\mathbf{f}^M|\mathcal{D})] = 0$ for any $\mathbf{X}^M \in \mathcal{X}^M$. For adversarial functional variational inference, this is also obvious due to $\sup_{\mathbf{X}^M} \mathrm{KL}[q(\mathbf{f}^M)\|p(\mathbf{f}^M|\mathcal{D})] = 0$.

So we have that Equation (43) holds for any $\mathbf{X}^M \in \mathcal{X}^M$. Given $M > 1$, then for $\forall 1 \leq i < j \leq M$, we have $m_p(\mathbf{x}_i) = m_q(\mathbf{x}_i)$, and $k_p(\mathbf{x}_i, \mathbf{x}_j) = k_q(\mathbf{x}_i, \mathbf{x}_j)$, which implies

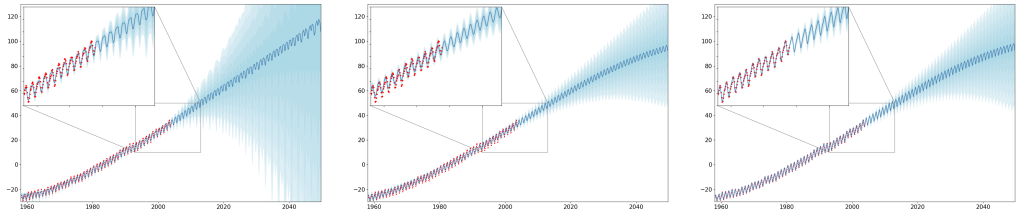
$$m_p(\cdot) = m_q(\cdot), \quad k_p(\cdot,\cdot) = k_q(\cdot,\cdot). \qquad (44)$$

Because GPs are uniquely determined by their mean and covariance functions, we arrive at the conclusion. □

C Additional Experiments

C.1 Time-series Extrapolation

Besides the toy experiments, we would like to examine the extrapolation behavior of our method on real-world datasets. Here we consider a classic time-series prediction problem concerning the



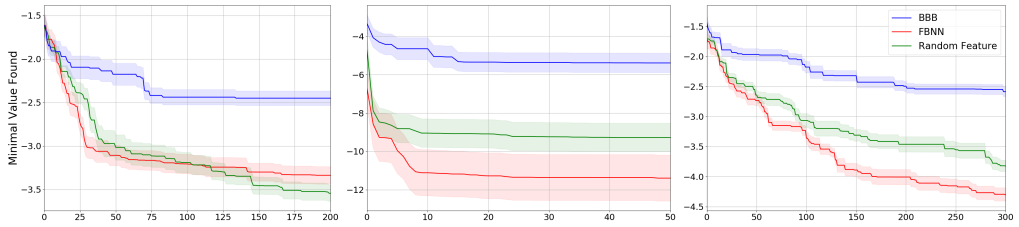
(a) BBB (b) FBNN (c) GP

Figure 4: Predictions on Mauna datasets. Red dots are training points. The blue line is the mean prediction and shaded areas correspond to standard deviations.

concentration of CO_2 in the atmosphere at the Mauna Loa Observatory, Hawaii (Rasmussen & Williams, 2006). The training data is given from 1958 to 2003 (with some missing values). Our goal is to model the prediction for an equally long period after 2003 (2004-2048). In Figure 4 we draw the prediction results given by BBB, fBNN, and GP. We used the same BNN architecture for BBB and fBNN: a ReLU network with 2 hidden layers, each with 100 units, and the input is a normalized year number augmented by its $\sin$ transformation, whose period is set to be one year. This special design allows both BBB and fBNN to fit the periodic structure more easily. Both models are trained for 30k iterations by the Adam optimizer, with learning rate 0.01 and batch size 20. For fBNN the prior is the same as the GP experiment, whose kernel is a combination of RBF, RBF$\times$PER (period set to one year), and RQ kernels, as suggested in Rasmussen & Williams (2006). Measurement points include 20 training samples and 10 points sampled from $\mathbb{U}[1958, 2048]$, and we jointly train the prior GP hyperparameters with fBNN.

In Figure 4 we could see that the performance of fBNN closely matches the exact prediction by GP. Both of them give visually good extrapolation results that successfully model the long-term trend, local variations, and periodic structures. In contrast, weight-space prior and inference (BBB) neither captures the right periodic structure, nor does it give meaningful uncertainty estimates.

C.2 Bayesian Optimization



(a) RBF (b) ArcCosine (c) Matern12

Figure 5: Bayesian Optimization. We plot the minimal value found along iterations. We compare fBNN, BBB and Random Feature methods for three kinds of functions corresponding to RBF, Order-1 ArcCosine and Matern12 GP kernels. We plot mean and 0.2 standard derivation over 10 independent runs.

In this section, we adopt Bayesian Optimization to explore the advantage of coherent posteriors. Specifically, we use Max Value Entropy Search (MES) (Wang & Jegelka, 2017), which tries to maximize the information gain about the minimum value $y^\star$,

$$\alpha_t(\mathbf{x}) = \mathbb{H}(p(y|D_t, \mathbf{x})) - \mathbb{H}(p(y|D_t, \mathbf{x}, y^\star)) \approx \frac{1}{K}\sum_{y^\star}[\frac{\gamma_{y^\star}(\mathbf{x})\phi(\gamma_{y^\star}(\mathbf{x}))}{\Psi(\gamma_{y^\star}(\mathbf{x}))} - \log(\Psi(\gamma_{y^\star}(\mathbf{x})))]$$

Where ϕ and Ψ are probability density function and cumulative density function of a standard normal distribution, respectively. The $y^\star$ is the minimum of a random function from the posterior, and $\gamma_{y^\star}(\mathbf{x}) = \frac{\mu_t(\mathbf{x}) - y^\star}{\sigma_t(\mathbf{x})}$.

With a probabilistic model, we can compute or estimate the mean $\mu_t(\mathbf{x})$ and the standard deviation $\sigma_t(\mathbf{x})$. However, to compute the MES acquisition function, samples $y^\star$ of function minima are required as well, which leads to difficulties. Typically when we model the data with a GP, we can get the posterior on a specific set of points but we don't have access to the extremes of the underlying function. In comparison, if the function posterior is represented in a parametric form, we can perform gradient decent easily and search for the minima.

We use 3-dim functions sampled from some Gaussian process prior for Bayesian optimization. Concretely, we experiment with samples from RBF, Order-1 ArcCosine and Matern12 kernels. We compare three parametric approaches: fBNN, BBB and Random Feature (Rahimi & Recht, 2008). For fBNN, we use the true kernel as functional priors. In contrast, ArcCosine and Matern12 kernels do not have simple explicit random feature expressions, therefore we use RBF random features for all three kernels. When looking for minima, we sample 10 $y^\star$. For each $y^\star$, we perform gradient descent along the sampled parametric function posterior with 30 different starting points. We use 500 dimensions for random feature. We use network with 5×100 for fBNN. For BBB, we select the network within $1 \times 100, 3 \times 100$. Because of the similar issue in Figure 1, using larger networks won't help for BBB. We use batch size 30 for both fBNN and BBB. The measurement points contain 30 training points and 30 points uniformly sampled from the known input domain of functions. For training, we rescale the inputs to $[0, 1]$, and we normalize outputs to have zero mean and unit variance. We train fBNN and BBB for 20000 iterations and anneal the coefficient of log likelihood term linearly from 0 to 1 for the first 10000 iterations. The results with 10 runs are shown in Figure 5.

As seen from Figure 5, fBNN and Random feature outperform BBB by a large margin on all three functions. We also observe fBNN performs slightly worse than random feature in terms of RBF priors. Because random feature method is exactly a GP with RBF kernel asymptotically, it sets a high standard for the parametric approaches. In contrast, fBNN outperforms random feature for both ArcCosine and Matern12 functions. This is because of the big discrepancy between such kernels and RBF random features. Because fBNN use true kernels, it models the function structures better. This experiment highlights a key advantage of fBNN, that fBNN can learn parametric function posteriors for various priors.

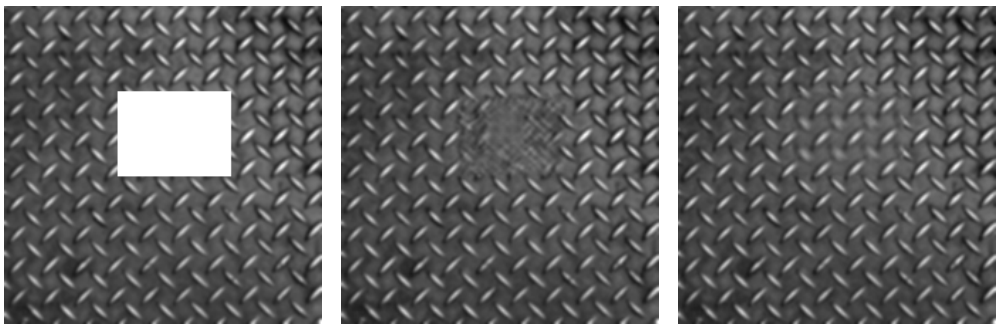
C.3 Texture Extrapolation

In this section we perform a texture extrapolation experiment.

With a texture image, texture extrapolation tries to recover the missing region based on the observed region in some image. We use a image of 224×224, and the central 60×80 region is removed from the image. Treating each pixel as a data point, this forms 45376 training examples and 4800 test examples. Because the training size is too big to be afforded by a standard Gaussian process. Previous GP methods (Wilson et al., 2014; Sun et al., 2018) investigate the additive structure of such patterns and take advantage of the Kronecker structure of the kernel matrix. However, such Kronecker structure is only applicable to axis-aligned structures. This experiment adopts fBNN with a GAN-like generator to perform the recovery.

We use a GP prior with the kernel used in (Sun et al., 2018). We firstly train the prior hyperparameters with GP regression with batch size 300. For the variational network, we use a GAN-like generator to generate the whole image starting from a 40 dimensional $\mathbb{U}[0, 1]$ noise. The network arranges as 40-(64*7*7)-(32*28*28)-(16*112*112)-(8*224*224)-(1*224*224). We use filter size 5 and leaky ReLU activations. Measurement points include 800 training points and 200 points sampled from the missing region (FBNN w. missing). We use learning rate 0.003 to train fBNN for 40000 iterations and anneal the kl coefficient from 0 to 1 linearly for the first 20000 iterations. We also evaluate a setting where measurement points only contain training regions (FBNN w.o. missing). The results are shown in Figure 6.

As shown in Figure 6, fBNN correctly recovers the underlying structure and extrapolates to the missing region. However, if measurement points do not contain missing regions, fBNN fits training set perfectly but fails to extrapolate. This highlights the extrapolation effect of measurement points.



(a) Training (b) FBNN w.o. missing (c) FBNN w. missing

Figure 6: Texture Extrapolation for FBNN. For FBNN w.o. missing, the measurement points does not contain the missing region; for FBNN w. missing, the measurement points contain the missing region.

Table 4: Averaged validation RMSE and log-likelihood for the regression benchmarks. We compared BBB, fBNNs and VFE. The numbers $a \times b$ represent networks with a hidden layers of b units.

		Test RMSE			Test log-likelihood		
Dataset	N	BBB	FBNN	VFE	BBB	FBNN	VFE
Kin8nm (1×100)	8192	0.082±0.001	0.079±0.001	0.071±0.001	1.082±0.008	1.112±0.007	1.241±0.005
Kin8nm (2×100)	8192	0.074±0.001	0.075±0.001	0.071±0.001	1.191±0.006	1.151±0.007	1.241±0.005
Kin8nm (5×500)	8192	0.266±0.003	0.076±0.001	0.071±0.001	-0.279±0.007	1.144±0.008	1.241±0.005
Power Plant (1×100)	9568	4.127±0.057	4.099±0.051	3.092±0.052	-2.837±0.013	-2.833±0.012	-2.531±0.018
Power Plant (2×100)	9568	4.081±0.054	3.830±0.055	3.092±0.052	-2.826±0.013	-2.763±0.013	-2.531±0.018
Power Plant (5×500)	9568	17.166±0.099	3.542±0.054	3.092±0.052	-4.286±0.007	-2.691±0.016	-2.531±0.018

C.4 Varying depth

To compare with Variational Free Energy (VFE) (Titsias, 2009), we experimented with two medium-size datasets so that we can afford to use VFE with full batch. For VFE, we used 1000 inducing points initialized by k-means of training point. For BBB and FBNNs, we used batch size 500 with a budget of 2000 epochs. As shown in Table 4, FBNNs performed slightly worse than VFE, but the gap became smaller as we used larger networks. By contrast, BBB totally failed with large networks (5 hidden layers with 500 hidden units each layer). Finally, we note that the gap between FBNNs and VFE diminishes if we use fewer inducing points (e.g., 300 inducing points).

C.5 Large scale regression with deeper networks

Table 5: Large scale regression. BBB and FBNN used networks with 5 hidden layers of 100 units.

		Test RMSE			Test log-likelihood		
Dataset	N	BBB	FBNN	SVGP	BBB	FBNN	SVGP
Naval	11934	1.3E-4±0.000	0.7E-4±0.000	0.3E-4±0.000	6.968±0.014	7.237±0.009	8.523±0.06
Protein	45730	3.684±0.041	3.659±0.026	3.740±0.015	-2.715±0.012	-2.721±0.010	-2.736±0.003
Video Memory	68784	0.984±0.074	0.967±0.040	1.417±0.234	-1.231±0.078	-1.337±0.058	-1.723±0.179
Video Time	68784	1.056±.0.178	1.083±0.288	3.216±1.154	-1.180±0.070	-1.468±0.279	-2.475±0.409
GPU	241600	5.136±0.087	4.806±0.116	21.287±0.571	-2.992±0.013	-2.973±0.019	-4.557±0.021

In this section we experimented on large scale regression datasets with deeper networks. For BBB and fBNNs, we used a network with 5 hidden layers of 100 units, and kept all other settings the same as Section 5.2.2. We also compared with the stochastic variational Gaussian processes (SVGP) (Hensman et al., 2013), which provides a principled mini-batch training for sparse GP methods, thus enabling GP to scale up to large scale datasets. For SVGP, we used 1000 inducing points initialized by k-means of training points (Note we cannot afford larger size of inducing points because of the cubic computational cost). We used batch size 2000 and iterations 60000 to match the training time with fBNNs. Likewise for BNNs, we used validation set to tune the learning rate from $\{0.01, 0.001\}$. We also tuned between not annealing the learning rate or annealing it by 0.1 at 30000 iterations. We evaluated the validating set in each epoch, and selected the epoch for testing based on the validation performance. The averaged results over 5 runs are shown in Table 5.

As shown in Table 5, SVGP performs better than BBB and fBNNs in terms of the smallest *naval* dataset. However, with dataset size increasing, SVGP performs worse than BBB and fBNNs by a

large margin. This stems from the limited capacity of 1000 inducing points, which fails to act as sufficient statistics for large datasets. In contrast, BNNs including BBB and fBNNs can use larger networks freely without the intractable computational cost.

D IMPLEMENTATION DETAILS

D.1 INJECTED NOISES FOR GAUSSIAN PROCESS PRIORS

For Gaussian process priors, $p(\mathbf{f}^{\mathbf{X}})$ is a multivariate Gaussian distribution, which has an explicit density. Therefore, we can compute the gradients $\nabla_{\mathbf{f}} \log p_{\phi}(\mathbf{f}^{\mathbf{X}})$ analytically.

In practice, we found that the GP kernel matrix suffers from stability issues. To stabilize the gradient computation, we propose to inject a small amount of Gaussian noise on the function values, i.e., to instead estimate the gradients of $\nabla_{\phi}\mathrm{KL}[q_{\phi} * p_{\gamma} \| p * p_{\gamma}]$, where $p_{\gamma} = \mathcal{N}(0, \gamma^2)$ is the noise distribution. This is like the instance-noise trick that is commonly used for stabilizing GAN training (Sønderby et al., 2016). Note that injecting the noise on the GP prior is equivalent to have a kernel matrix $\mathbf{K} + \gamma^2\mathbf{I}$. Beyond that, injecting the noise on the parametric variational posterior does not affect the reparameterization trick either. Therefore all the previous estimation formulas still apply.

D.2 IMPLICIT PRIORS

Our method is applicable to implicit priors. We experiment with piecewise constant prior and piecewise linear prior. Concretely, we randomly generate a function $f : [0, 1] \to R$ with the specific structure. To sample piecewise functions, we first sample $n \sim \mathrm{Poisson}(3.)$, then we have $n+1$ pieces within $[0, 1]$. We uniformly sample n locations from $[0, 1]$ as the changing points. For piecewise constant functions, we uniformly sample $n + 1$ values from $[0, 1]$ as the function values in each piece; For piecewise linear functions, we uniformly sample $n + 1$ values for the values at first $n + 1$ locations, we force $f(1) = 0.$. Then we connect together each piece by a straight line.