Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights

Theofanis Karaletsos*, Thang D. Bui*
theofanis@uber.com, thang.bui@uber.com [1]

[1]Uber AI, San Francisco, California, USA
*authors contributed equally

Abstract

Probabilistic neural networks are typically modeled with independent weight priors, which do not capture weight correlations in the prior and do not provide a parsimonious interface to express properties in function space. A desirable class of priors would represent weights compactly, capture correlations between weights, facilitate calibrated reasoning about uncertainty, and allow inclusion of prior knowledge about the function space such as periodicity or dependence on contexts such as inputs. To this end, this paper introduces two innovations: (i) a Gaussian process-based hierarchical model for network weights based on unit embeddings that can flexibly encode correlated weight structures, and (ii) input-dependent versions of these weight priors that can provide convenient ways to regularize the function space through the use of kernels defined on contextual inputs. We show these models provide desirable test-time uncertainty estimates on out-of-distribution data, demonstrate cases of modeling inductive biases for neural networks with kernels which help both interpolation and extrapolation from training data, and demonstrate competitive predictive performance on an active learning benchmark.

1 Introduction

Bayesian neural networks (BNNs) [see e.g. MacKay, 1992, Neal, 1993, Ghahramani, 2016] are one of the research frontiers on combining Bayesian inference and deep learning, potentially offering flexible modelling power with calibrated predictive performance. In essence, applying probabilistic inference to neural networks allows all plausible network parameters, not just the most likely, to be used for predictions. Despite the strong interest in the community for the exploration of BNNs, there remain unanswered questions: (i) how can we model neural network functions to encourage behaviors such as interpolation between signals and extrapolation from data in meaningful ways, for instance by encoding prior knowledge, or how to specify priors which facilitate uncertainty quantification, and (ii) many scalable approximate inference methods are not rich enough to capture complicated posterior correlations in large networks, resulting in undesirable predictive performance at test time.

This paper attempts to tackle some of the aforementioned limitations. We propose inherently correlated weight priors by utilizing unit-level latent variables to obtain a compact parameterization of neural network weights and combine them with ideas from the Gaussian process (GP) literature to induce a hierarchical GP prior over weights. This prior flexibly models the correlations between weights in a layer and across layers. We explore the use of product kernels to implement input-dependence as a variation of the proposed prior, yielding models that have per-datapoint priors which facilitate inclusion of prior knowledge through kernels while maintaining their weight structure. A structured variational inference approach is employed that sidesteps the need to do inference in the weight space whilst retaining weight correlations in the approximate posterior. The proposed priors and approximate inference scheme are demonstrated to exhibit beneficial properties for tasks such as generalization, uncertainty quantification, and active learning.

The paper is organized as follows: in Section 2 we review hierarchical modeling for BNNs based on unit-variables. In Section 3 we introduce the global and local weight models and their applications to neural

networks. Efficient inference algorithms for both models are presented in Section 4, followed by a suite of experiment to validate their performance in Section 5. We review related work in Section 6.

2 Meta-representing weights and networks

Our work builds on and expands the class of hierarchical neural network models based on the concept of latent variables associated with units in a network as proposed in [Karaletsos et al., 2018]. In that model, each unit (visible or hidden) of the l-th layer of the network has a corresponding latent hierarchical variable $\mathbf{z}_{l,i}$, of dimensions D_z, where i denotes the index of the unit in a layer. Note that these latent variables do not describe the activation of units, but rather constitute latent features associated with a unit.

The design of these latent variables is judiciously chosen to construct the weights in the network as follows: a weight in the l-th layer, $w_{l,i,j}$ is generated by using the concatenation of latent variable z's of the i-th input unit and the j-th output unit as inputs of a mapping function $f(\left[\mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}\right])$.

We can summarize this relationship by introducing a set of *weight encodings* $\mathbf{C}_w(\mathbf{z})$, one for each individual weight in the network $\mathbf{c}_{w_{l,i,j}} = \left[\mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}\right]$, which can be deterministically constructed from the collection of unit latent variable samples $\mathbf{z}$ by concatenating them correctly according to network architecture. The probabilistic description of the relationship between the weight codes (summarizing the structured latent variables) and the weights $\mathbf{w}$ is:

$$p(\mathbf{w}|\mathbf{z}) = p(\mathbf{w}|\mathbf{C}_w(\mathbf{z})) = \prod_{l=1}^{L-1} \prod_{i=1}^{H_l} \prod_{j=1}^{H_{l+1}} p(w_{l,i,j}|\mathbf{c}_{w_{l,i,j}}),$$

where l denotes a visible or hidden layer and H_l is the number of units in that layer, L is the total number of layers in the network, and $\mathbf{w}$ denotes all the weights in this network.

In Karaletsos et al. [2018], a small parametric neural network regression model (conceptually a *structured hyper-network*) is chosen to map the latent variables to the weights, using either a Gaussian noise model $p(\mathbf{w}|\mathbf{C}_w(\mathbf{z}), \theta) = \mathcal{N}(\mathbf{w}|\mathrm{NN}_\theta(\mathbf{C}_w(\mathbf{z}))$ or an implicit noise model: $p(\mathbf{w}|\mathbf{C}_w(\mathbf{z}), \theta) \propto \mathrm{NN}_\theta(\mathbf{C}_w(\mathbf{z}), \epsilon)$, where ϵ is a random variate. We will call this network a *meta mapping*. Note that given the collection of sampled unit variables $\mathbf{z}$ and the derived codes $\mathbf{C}_w(\mathbf{z})$, the weights (or theirs mean and variance) can be obtained efficiently in parallel. A prior over latent variables $\mathbf{z}$ completes the model specification, $p(\mathbf{z}) = \prod_{l=0}^{L} \prod_{i=0}^{H_l} \mathcal{N}(\mathbf{z}_{l,i}; \mathbf{0}, \mathbf{I})$. The joint density of the resulting hierarchical BNN is then specified as follows,

$$p(\mathbf{y}, \mathbf{w}, \mathbf{z}|\mathbf{x}, \theta) = p(\mathbf{z})p(\mathbf{w}|\mathbf{C}_w(\mathbf{z}), \theta) \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{w}, \mathbf{x}_n),$$

with N denoting the number of observation tuples $\{\mathbf{x}_n, \mathbf{y}_n\}$.

Variational inference was employed in prior work to infer $\mathbf{z}$ (and $\mathbf{w}$ implicitly), and to obtain a point estimate of θ, as a by-product of optimising the variational lower bound. Critically, in this representation weights are only implicitly parametrized through the use of these latent variables, which transforms inference on weights into inference of the much smaller collection of latent unit variables.

The central motivations for our adoption of this parameterization are two-fold. *First*, the number of visible and hidden units in a neural network is typically much smaller than the number of weights. For example, for the l-th weight layer, there are $H_l \times H_{l+1}$ weights compared to $H_l + H_{l+1}$ associated latent variables. This encourages the development of models and inference schemes that can work in the lower-dimensional hierarchical latent space directly without the need to model individual weights privately. This structured representation per weight allows powerful hierarchical models to be used without requiring high dimensional parametrizations (i.e. hypernetworks for the entire weight tensor). Specifically, a GP-LVM prior Lawrence [2004] over all network weights appears infeasible without such an encoding of structure. *Second*, the compact latent space representation facilitates attempts at building fine-grained control into weight priors, such as structured prior knowledge as we will see in the following sections.

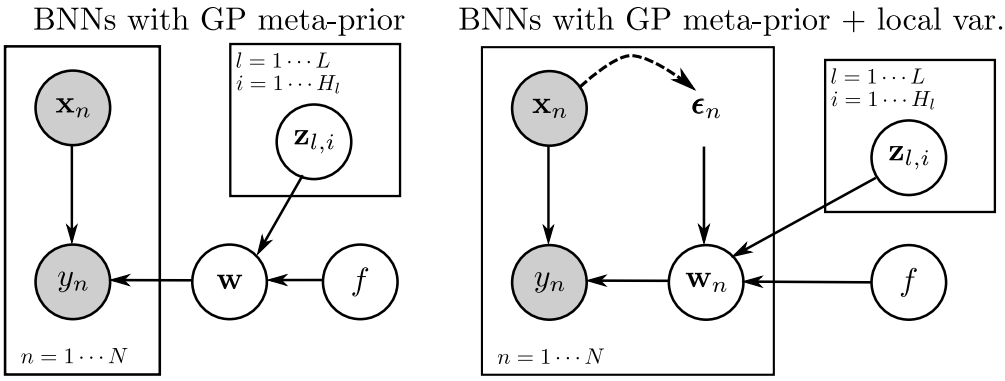

Figure 1: Graphical depiction of BNNs with hierarchical GP-MetaPriors and ones with input-dependent variables.

3 Hierarchical GP-Priors For BNN Weights

Notice that in Section 2, the meta mapping from the hierarchical latent variables to the weights is a parametric non-linear function, specified by a neural network. We replace the parametric neural network by a probabilistic functional mapping and place a nonparametric Gaussian process prior over this function. That is,

$$p(w_{l,i,j}|f, \mathbf{c}_{w_{l,i,j}}) = \mathcal{N}(w_{l,i,j}; f([\mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}]), \sigma_w^2),$$
$$p(f|\gamma) = \mathcal{GP}(f; \mathbf{0}, k_w(\cdot, \cdot|\gamma)),$$

where we have assumed a zero-mean GP, $k_\gamma(\cdot, \cdot)$ is a covariance function and γ is a small set of hyper-parameters, and a homoscedastic[1] Gaussian noise model with variance σ_w^2.

The effect is that the latent function introduces correlations for the individual weight predictions,

$$\begin{aligned} P(\mathbf{w}|\mathbf{z}) &= P(\mathbf{w}|\mathbf{C}_w(\mathbf{z})) \\ &= \int_f p(f) \Big[\prod_{l=1}^{L-1} \prod_{i=1}^{H_1} \prod_{j=1}^{H_{l+1}} p(w_{l,i,j}|f, \mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}) \Big]. \end{aligned}$$

Notably, while the number of latent variables and weights can be large, the input dimension to the GP mapping is only $2D_z$, where D_z is the dimensionality of each latent variable $\mathbf{z}$. The GP mapping effectively performs one-dimensional regression from latent variables to individual weights while capturing their correlations. We will refer to this mapping as a **GP-MetaPrior** (*MetaGP*). We define the following kernel at the example of two weights in the network,

$$k_w(c_{w_1}, c_{w_2}) = k_w([\mathbf{z}_{l^1,i^1}, \mathbf{z}_{l^1+1,j^1}], [\mathbf{z}_{l^2,i^2}, \mathbf{z}_{l^2+1,j^2}])$$

In this section and what follows, we will use the popular exponentiated quadratic (RBF) kernel with ARD lengthscales, $k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_k^2 \exp\left(\sum_{d=1}^{2D_z} \frac{-(x_{1,d}-x_{2,d})^2}{2l_d^2}\right)$, where $\{l_d\}_{d=1}^{2D_z}$ are the lengthscales and σ_k^2 is the kernel variance.

BNNs with GP-MetaPriors are then specified by the following joint density over all variables:

$$\begin{aligned} p(\mathbf{y}, \mathbf{w}, \mathbf{z}, f|\mathbf{x}) &= p(\mathbf{z})p(f)p(\mathbf{w}|f, \mathbf{z})p(\mathbf{y}|\mathbf{w}, \mathbf{x}) \\ &= p(\mathbf{z})p(f)p(\mathbf{w}|f, \mathbf{C}_w(\mathbf{z})) \prod_{n=1}^{N} \left[p(\mathbf{y}_n|\mathbf{w}, \mathbf{x}_n)\right]. \end{aligned}$$

[1]Here, we present a homoscedastic noise model for the weights, but the model is readily adaptable to a heteroscedastic noise model which we omit for clarity.

We show prior samples from this model in Fig. 2 by the following procedure: for a sample $\mathbf{z}_{\text{viz}} \sim p(\mathbf{z})$ we instantiate the covariance matrix $\mathbf{K}_w$ by constructing weight codes and applying the kernel function k_w. We draw weights from the GP by sampling the Normal distribution $\mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{K}_w + \sigma_w^2 \mathbf{I})$, where $\mathbf{K}_w \in \mathcal{R}^{|\mathbf{w}| \times |\mathbf{w}|}$, with $|\mathbf{w}|$ denoting the number of all parameters in the network. We then generate BNN function samples given the sampled weights with homoscedastic noise on the outputs. We highlight two properties of this model: *First*, as a hierarchical model, given a sample of the latent variables $\mathbf{z}$, the model instantiates a prior over weights from which we can further sample functions and thus encodes two levels of uncertainty over functions. *Second*, we demonstrate that changing the length-scale parameter of the mapping-kernel k_w, again even *given fixed samples* $\mathbf{z}$, leads to vastly differentiated function samples, showing the compact degree of control the mapping parameters have over the function space being modeled. In this case, the length-scale appears to control the variance over the function space, which matches an intuitive interpretation over the kernel parameter.

An important task is marginalization of the latent quantities given data to perform posterior inference, which we discuss in Section 4.1.

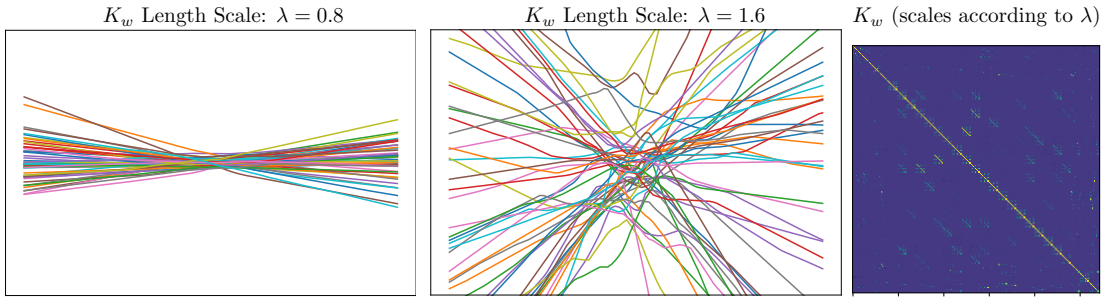

Figure 2: **MetaGP Prior Samples**: We show function samples generated from a [1,20,10,1] unit BNN with ReLUs with meta-GP prior. We draw one sample per unit $\mathbf{z}_{\text{viz}} \sim p(\mathbf{z})$ to instantiate the weight prior and subsequently draw 40 function samples (individually colored) from the BNN by drawing from the conditional prior $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{z}_{\text{viz}})$ and regressing $\mathbf{y} \sim p(\mathbf{y}|\mathbf{w}, \mathbf{x})$. (**left**) We show samples drawn when the global RBF kernel for the GP has a length scale set to a small value. (**middle**) we keep the same samples $\mathbf{z}$ and only change the length scale to be larger and visualize the functions induced by the BNN (**right**) We visualize the weight kernel given the latent variables $\mathbf{z}_{\text{viz}}$. Other samples of $\mathbf{z}$ would induce different weight covariance matrices. Overall this figure shows that even given $\mathbf{z}$, the proposed prior models a wide range of functions which have controllable properties based on the parameters of the kernel.

3.1 Input-kernels for modulating function priors

While the hierarchical latent variables and meta mappings introduce non-trivial coupling between the weights a priori, they are inherently global. That is, a function drawn from the model, represented by a set of weights, does not take into account the inputs at which the function will be evaluated. In this section, we will describe modifications to our weight prior which allow conditional weight models on inputs.

To this end, we introduce the input variable into the weight codes $\mathbf{c}_{w_{n,l,i,j}} = \left[\mathbf{c}_{w_{l,i,j}}, \mathbf{x}_n\right] = \left[\mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}, \mathbf{x}_n\right]$, which we utilize to yield input-conditional weight models $p(w_{n,l,i,j}|f, \mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}, \mathbf{x}_n)$ through the use of product kernels. Concretely, we introduce a new **input kernel** k_{aux} which multiplied with the global weight kernel k_w gives the kernel k_{local} for the meta mapping,

$$k_{\text{local}}(\mathbf{c}_{w_1,x_1}, \mathbf{c}_{w_2,x_2}) = k_w(\mathbf{c}_{w_1}, \mathbf{c}_{w_2}) \cdot k_{\text{aux}}(\mathbf{x}_1, \mathbf{x}_2)$$

where k_w is the kernel defined over latent-variable weight codes from Section 3, k_{aux} is an auxiliary kernel modeling input-dependence on $\mathbf{x}_n$, and $\mathbf{c}_{w_{l,i,j},x_n}$ is shorthand for $\mathbf{c}_{w_{n,l,i,j}}$. This factorization over kernels represents an assumption of separable influence on functions by latent variables $\mathbf{z}$ and inputs. The weight priors are now also local to each data point, in a similar vein to how functions are drawn from a GP, while still instantiating an explicit, weight-based model.

We demonstrate the effects of utilizing the auxiliary kernel in this factorized fashion by visualizing prior function samples from a BNN with this local prior when changing kernel parameters in Fig. 3, exemplifying the proposed model's ability to encode controlled periodic structure into BNN weight priors before seeing any data. As performed in Section 3, we sample from the GP to instantiate weights, but in the case of the local model we instantiate the covariance matrix $\mathbf{K}_{\text{local}} \in \mathcal{R}^{|\mathbf{w}|\times|\mathbf{w}|\times N\times N}$. We will discuss the handling of this conceptually large object in Section 4.2.

To scale this to large inputs, we learn transformations of inputs for the conditional weight model $\boldsymbol{\epsilon}_n = g(\mathbf{V}\mathbf{x}_n)$, for a learned mapping $\mathbf{V}$ and a nonlinearity g and generalize weight codes to $\mathbf{c}_{w_{n,l,i,j}} = \left[\mathbf{z}_{l,i}, \mathbf{z}_{l+1,j}, \epsilon_n\right]$, with $\mathbf{C}_{w,x}(\mathbf{z}, \mathbf{x})$ describing their collection. In detail, each auxiliary input is obtained via a (potentially nonlinear) transformation applied to an input: $\boldsymbol{\epsilon}_n = g(\mathbf{V}\mathbf{x}_n)$, where $\mathbf{V} \in \mathcal{R}^{D_{\text{aux}}\times D_x}$, and D_{aux} and D_x are the dimensionality of $\boldsymbol{\epsilon}_n$ and $\mathbf{x}_n$, respectively, and $g(\cdot)$ is an arbitrary transformation. We may also layer these transformations in general. We typically set $D_{\text{aux}} \ll D_x$ so this transformation could be thought of as a dimensionality reduction operation. For low dimensional inputs, we set $\boldsymbol{\epsilon}_n = \mathbf{x}_n$.

Including these transformations yields the weight model $p(w_{n,l,i,j}|f, \mathbf{z}, \mathbf{V}, \mathbf{x}_n) = \mathcal{N}(w_{n,l,i,j}; f(\mathbf{c}_{w_{n,l,i,j}}), \sigma_w^2)$, that is, the input dimension of the meta mapping is now $2D_z + D_{\text{aux}}$. Additionally, we also place a prior over the linear transformation: $p(\mathbf{V}) = \mathcal{N}(\mathbf{V}; \mathbf{0}, \mathbf{I})$. We will refer to this mapping as a **Local GP-MetaPrior** (*MetaGP-local*).

What effects should we expect from such a modulation? Consider the use of an exponentiated quadratic kernel: we would expect data which lies far away from training data to receive small kernel values from K_{aux}. This, in turn, would modulate the entire kernel K_{local} for that data point to small values, leading to a weight model that reverts increasingly to the prior. We would expect such a model to help with modeling uncertainty by resetting weights to uninformative distributions away from training data. One may also want to use this mechanism to express inductive biases about the function space, such as adding structure to the weight prior that can be captured with a kernel. This is an appealing avenue, as multiple useful kernels have been found in the GP literature that allow modelers to describe relationships between data, but have previously not been accessible to neural network modelers. We consider this a novel form of functional regularization through the weight prior, which can imbue the entire network with structure that will constrain its function space.

BNNs with Local GP-MetaPriors specify neural networks with individual weight priors per datapoint (also see Graphical Model in Fig. 1):

$$
\begin{aligned}
p(\cdot) &= p(\mathbf{z})p(f)\prod_{n=1}^{N} p(\mathbf{y}_n, \mathbf{w}_n|f, \mathbf{z}, \mathbf{x}_n) \\
&= p(\mathbf{z})p(f)\prod_{n=1}^{N} p(\mathbf{w}_n|f, \mathbf{C}_w(\mathbf{z}, \mathbf{x}_n))\left[p(\mathbf{y}_n|\mathbf{w}_n, \mathbf{x}_n)\right].
\end{aligned}
$$

Inference and learning are modified accordingly as explained in Section 4.2.

4 Inference and learning using stochastic structured variational inference

Performing inference is challenging due to the non-linearity of the neural network and the need to infer an entire latent function f. In Section 4.1 we address these problems for *MetaGP*, deriving a structured variational inference scheme that makes use of innovations from inducing point GP approximation literature [Titsias, 2009, Hensman et al., 2013, Quiñonero-Candela and Rasmussen, 2005, Matthews et al., 2016, Bui et al., 2017] and previous work on inferring meta-representations [Karaletsos et al., 2018]. In Section 4.2 we will highlight the modifications necessary to make this inference strategy work for *MetaGP-local*.

4.1 Inference for the global model

A common strategy for variational inference in GPs is the utilization of inducing points, which entails the construction of learned inputs to the function and corresponding function values which jointly take the

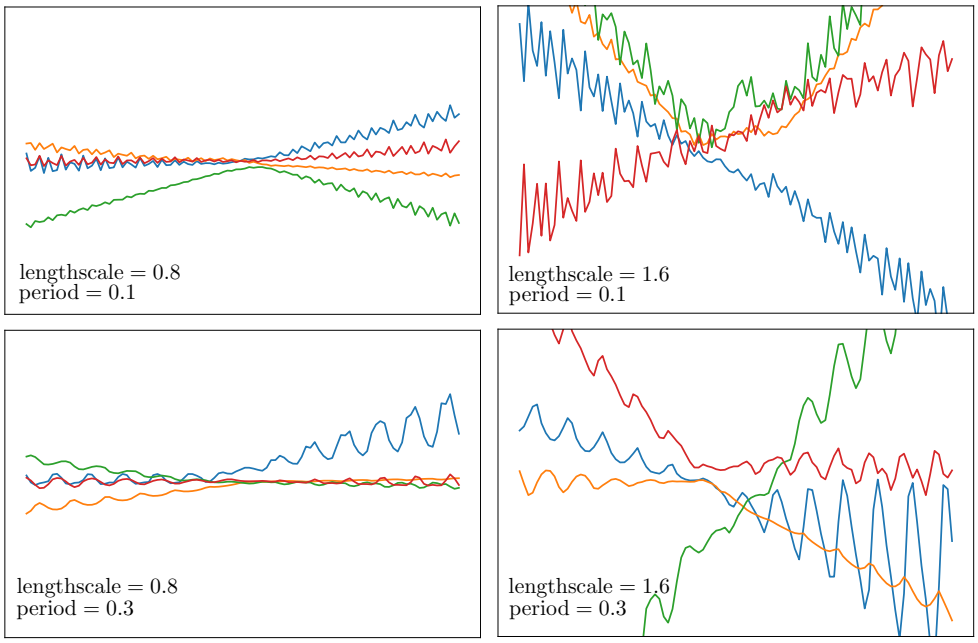

Figure 3: **Local MetaGP Samples**: We remind the reader that the input-dependent weight prior has a factorized kernel structure $k_{\text{local}} = k_w \cdot k_{\text{aux}}$, and we wish to demonstrate the effect of each kernel separately in terms of its effects on the induced function prior for the neural network. We are given the same samples $\mathbf{z}_{\text{viz}}$ as in Fig. 2 and also keep the two kernel parameter choices for k_w, while varying only the period parameter for an auxiliary periodic kernel. **Left**: We show function samples using a small period of 0.1 and a period of 0.3 in combination with the k_w kernel with length-scale 0.8. We can see, that while the functions are still relatively flat, the auxiliary kernel induces weight priors which lead to periodic function samples consistent with the auxiliary kernel setting. **Right**: Similarly, when performing the same protocol for the k_w with the larger lengthscale, we again observe periodic functions consistent with the set period (although we only apply the periodic kernel for the weight priors for the BNN), but see that the functions sampled have more variance, consistent with the larger length-scale of the weight-kernel k_w. Note that while the functions exhibit periodic structure, they have non-periodic global structure as well, as they also draw information from k_w and the priors are merely *modulated* by the auxiliary kernel. We thus see that our prior structure successfully induces function priors which naturally inherit properties we can express as kernel functions, but keep rich expressivity as weight based models.

place of representative data points. The inducing inputs and outputs, $\{\mathbf{C_u}, \mathbf{u}\}$, will be used to parameterize the approximation.

We first partition the space $\mathcal{C}$ of inputs (or *weight codes*) to the function f into a finite set of M variables called inducing inputs $\mathbf{C_u} = \{\mathbf{c}_{u,m}\}_{m=1}^{M}$ where $\mathbf{c}_{u,m} \in \mathcal{R}^{2D_z}$ and the remaining inputs, $\mathcal{C} = \{\mathbf{C_u}, \mathcal{C}_{\neq \mathbf{C_u}}\}$. The function f is partitioned identically, $f = \{\mathbf{u}, f_{\neq \mathbf{u}}\}$, where $\mathbf{u} = f(\mathbf{C_u})$. We can then rewrite the GP prior as follows, $p(f) = p(f_{\neq \mathbf{u}}|\mathbf{u})p(\mathbf{u})$.[2] In particular, a variational approximation is judiciously chosen to mirror the form of the joint density:

$$q(\mathbf{w}, \mathbf{z}, f) = q(\mathbf{z})p(f_{\neq \mathbf{u}}|\mathbf{u})q(\mathbf{u})p(\mathbf{w}|f, \mathbf{z}), \tag{1}$$

where the variational distribution over $\mathbf{w}$ is made to explicitly depend on remaining variables through the conditional prior, and $q(\mathbf{z})$ is chosen to be a diagonal (mean-field) Gaussian density, $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\mathbf{z}}, \text{diag}(\boldsymbol{\sigma}_{\mathbf{z}}^2))$, and $q(\mathbf{u})$ is chosen to be a correlated multivariate Gaussian, $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_{\mathbf{u}}, \Sigma_{\mathbf{u}})$. This approximation allows

[2]The conditioning on $\mathcal{C}_{\neq \mathbf{C_u}}$ and $\mathbf{C_u}$ in $p(\mathbf{u})$ and $p(f_{\neq \mathbf{u}}|\mathbf{u})$ is made implicit here and in the rest of this paper.

convenient cancellations yielding a tractable variational lower bound as follows,

$$\begin{aligned}\mathcal{F}(\cdot) &= \int_{q(\mathbf{w},\mathbf{z},f)} \log \frac{p(\mathbf{z})\cancel{p(f_{\neq\mathbf{u}}|\mathbf{u})}p(\mathbf{u})\cancel{p(\mathbf{w}|f,\mathbf{z})}p(\mathbf{y}|\mathbf{w},\mathbf{x})}{q(\mathbf{z})\cancel{p(f_{\neq\mathbf{u}}|\mathbf{u})}q(\mathbf{u})\cancel{p(\mathbf{w}|f,\mathbf{z})}} \\ &\approx -\mathrm{KL}[q(\mathbf{z})||p(\mathbf{z})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \\ &\quad + \frac{1}{S}\sum_{s=1}^{S}\int_{\mathbf{w},f} q(\mathbf{w},f|\mathbf{z}_s)\log p(\mathbf{y}|\mathbf{w},\mathbf{x}),\end{aligned}$$

where the last expectation has been approximated by simple Monte Carlo with the reparameterization trick, i.e. $\mathbf{z}_s \sim q(\mathbf{z})$ [Salimans and Knowles, 2013, Kingma and Welling, 2013, Titsias and Lázaro-Gredilla, 2014]. We will next discuss how to approximate the expectation $\mathcal{F}_s = \int_{\mathbf{w},f} q(\mathbf{w},f|\mathbf{z}_s)\log p(\mathbf{y}|\mathbf{w},\mathbf{x})$. Note that we split f into $f_{\neq\mathbf{u}}$ and $\mathbf{u}$, and that we can integrate $f_{\neq\mathbf{u}}$ out exactly to give, $q(\mathbf{w}|\mathbf{z}_s,\mathbf{u}) = \mathcal{N}(\mathbf{w};\mathbf{A}^{(s)}\mathbf{u},\mathbf{B}^{(s)})$,

$$\begin{aligned}\mathbf{A}^{(s)} &= \mathbf{K}_{\mathbf{wu}}^{(s)}\mathbf{K}_{\mathbf{uu}}^{-1}, \\ \mathbf{B}^{(s)} &= \mathbf{K}_{\mathbf{ww}}^{(s)} - \mathbf{K}_{\mathbf{wu}}^{(s)}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uw}}^{(s)} + \sigma_w^2\mathbf{I},\end{aligned}$$

where $\mathbf{K}_{\mathbf{wu}}^{(s)} = k_w(\mathbf{C}_{\mathbf{w}}^{(s)},\mathbf{C}_{\mathbf{u}})$, $\mathbf{K}_{\mathbf{uu}} = k_w(\mathbf{C}_{\mathbf{u}},\mathbf{C}_{\mathbf{u}})$, $\mathbf{K}_{\mathbf{ww}}^{(s)} = k_w(\mathbf{C}_{\mathbf{w}}^{(s)},\mathbf{C}_{\mathbf{w}}^{(s)})$. At this point, we can either (i) sample $\mathbf{u}$ from $q(\mathbf{u})$, or (ii) integrate $\mathbf{u}$ out analytically. Opting for the second approach gives $q(\mathbf{w}|\mathbf{z}_s) = \mathcal{N}(\mathbf{w};\mathbf{A}^{(s)}\boldsymbol{\mu}_{\mathbf{u}},\mathbf{B}^{(s)} + \mathbf{A}^{(s)}\Sigma_{\mathbf{u}}\mathbf{A}^{\intercal,(s)})$, the former just omits the second covariance term and uses a sample $\mathbf{u}^s$ for the predictive mean instead of $\boldsymbol{\mu}_{\mathbf{u}}$.

In contrast to GP regression and classification in which the likelihood term is factorized point-wise w.r.t. the parameters and thus their expectations only involve a low dimensional integral, we have to integrate out $\mathbf{w}$ which for GPs entails inversion of the $|\mathbf{w}| \times |\mathbf{w}|$ matrix $\mathbf{K}$ (which is $\mathbf{B}^{(s)}$ when we don't sample $\mathbf{u}$ or the full term above). This is feasible for small neural networks with up to a few thousand weights, but becomes intractable for more general architectures. In order to scale to larger networks, we introduce a diagonal approximation, which given a sample $\mathbf{u}^s$ looks as follows, $\hat{q}(\mathbf{w}|\mathbf{z}_s) = \mathcal{N}(\mathbf{w};\mathbf{A}^{(s)}\mathbf{u}^s,\mathrm{diag}(\mathbf{B}^{(s)}))$. Whilst the diagonal approximation above might look poor at first glance, it is conditioned on a sample of the latent variables $\mathbf{z}_s$ and thus the weights' correlations induced by the hierarchical unit-structure are retained after integrating out $\mathbf{z}$. Such correlations are illustrated in Fig. 5, showing the marginal and conditional covariance structures for the weights of a small neural network, separated into diagonal and full covariance models. We also provide a qualitative and quantitative analysis of performance of different approximations to $q(\mathbf{w}|\mathbf{z}_s)$ in the appendix, including the diagonal approximation presented here, and show that not only is this approximation fast but also that it performs competitively with full covariance models. Finally, the expected log-likelihood $\mathcal{F}_s$ is approximated by $\mathcal{F}_s \approx \frac{1}{J}\sum_{j=1}^{J}\log p(\mathbf{y}|\mathbf{w}_j,\mathbf{x})$ with samples $\mathbf{w}_j \sim \hat{q}(\mathbf{w}|\mathbf{z}_s)$ [3]. The final lower bound is then optimized to obtain the variational parameterers of $q(\mathbf{u})$, $q(\mathbf{z})$, and estimates for the noise in the meta-GP model, the kernel hyper-parameters and the inducing inputs.

4.2 Inference for the local model

The main difference in the local model is the dependence of weights on inputs. To handle inducing point kernels over both weight codes and inputs, we introduce inducing inputs $\tilde{\boldsymbol{\epsilon}} = \{\tilde{\epsilon}_m\}_{m=1}^M$ where $\tilde{\epsilon} \in \mathcal{R}^{D_{\mathrm{aux}}}$ for k_{aux}. We then concatenate the dimensions of $\mathbf{C}_{\mathbf{u}}$ in Section 4.1 with the new inducing inputs to form the new inputs $\tilde{\mathbf{C}}_{\mathbf{u}} = [\mathbf{C}_{\mathbf{u}};\tilde{\boldsymbol{\epsilon}}]$. The set of inputs $\tilde{\mathbf{C}}_{\mathbf{u}}$ now have dimensions $\tilde{\mathbf{c}}_u \in \mathcal{R}^{2D_z+D_{\mathrm{aux}}}$. The fully instantiated covariance matrix $\mathbf{K}_{\mathrm{local}} = K_w \otimes K_{\mathrm{aux}}$ would take the shape $|\mathbf{w}| \times |\mathbf{w}| \times N \times N$. As this kernel has Kronecker structure one could now consider using inference techniques such as in [Flaxman et al., 2015]. However, the tractability of the global kernel remains an issue even in this case. As such, we elect to inherit the diagonal approximation from Section 4.1 and apply it to the joint kernel, yielding an object of dimension $|\mathbf{w}| \times N$. The lower bound computation in Section 4.1 can thus be reused but with $\mathbf{A}^{(n,s)}$ and $\mathbf{B}^{(n,s)}$ being input-dependent[4]. We can handle large datasets by using inducing point kernels, which permit inference

[3] We can also use the *local reparameterization trick* [Kingma et al., 2015] to reduce variance.

[4] Specifically, $\mathbf{A}^{(n,s)} = \mathbf{K}_{\mathbf{wu}}^{(n,s)}\mathbf{K}_{\mathbf{uu}}^{-1}, \mathbf{B}_n^{(n,s)} = \mathbf{K}_{\mathbf{ww}}^{(n,s)} - \mathbf{K}_{\mathbf{wu}}^{(n,s)}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uw}}^{(n,s)} + \sigma_w^2\mathbf{I}$, where $\mathbf{K}_{\mathbf{wu}}^{(n,s)} = k_w(\mathbf{C}_{\mathbf{w}}^{(s)},\mathbf{C}_{\mathbf{u}}) \otimes k_{\mathrm{aux}}(\mathbf{x}_n,\tilde{\boldsymbol{\epsilon}})$, $\mathbf{K}_{\mathbf{uu}} = k_w(\mathbf{C}_{\mathbf{u}},\mathbf{C}_{\mathbf{u}}) \cdot k_{\mathrm{aux}}(\tilde{\boldsymbol{\epsilon}},\tilde{\boldsymbol{\epsilon}})$, $\mathbf{K}_{\mathbf{ww}}^{(n,s)} = k_w(\mathbf{C}_{\mathbf{w}}^{(s)},\mathbf{C}_{\mathbf{w}}^{(s)}) \otimes k_{\mathrm{aux}}(\mathbf{x}_n,\mathbf{x}_n)$.

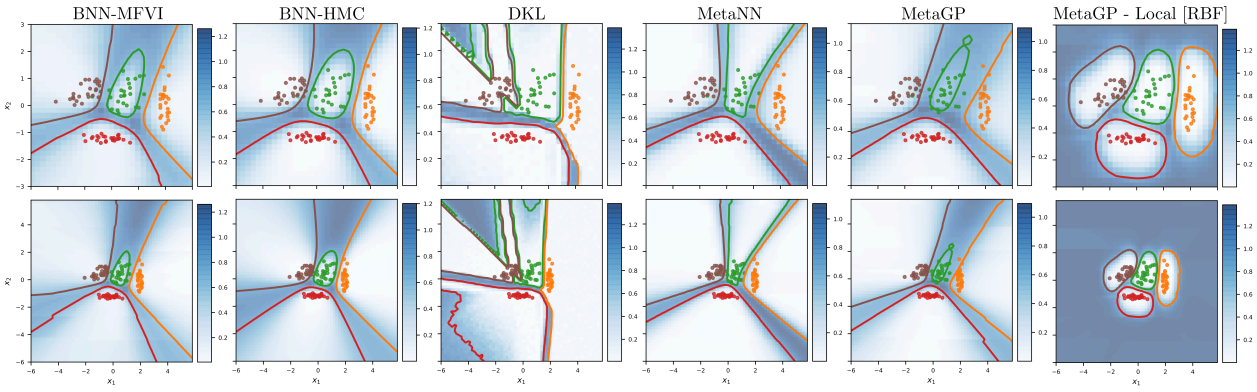

Figure 4: Predictive performance of various methods on a four-way classification problem. We compare the proposed approaches (MetaGP, MetaGP with an input-dependent RBF kernel and periodic kernel) to BNN with MFVI and HMC, DKL and MetaNN. Best viewed in colour. The background color shows the entropy of the predictive distribution. The contours show the 0.7 equiprobability contours. The bottom plots are the zoom-out version of the corresponding top plots, showing the predictive entropy further from the training points.

using minibatches. Another difference is the potential existence of the mapping $\mathbf{V}$ in the model, which we tackle by introducing a variational distribution $q(\mathbf{V}) = \mathcal{N}(\mathbf{V}; \boldsymbol{\mu}_{\mathbf{V}}, \text{diag}(\boldsymbol{\sigma}^2_{\mathbf{V}}))$. We can estimate the evidence lower bound by also drawing unbiased samples from this and jointly optimizing its parameters with the rest of the variational parameters. The overall computational complexity with data-subsampling in this section and Section 4.1 is $\mathcal{O}(M^3 + |\mathbf{w}|M^2)$.

5 Experiments

In this section, we evaluate the proposed priors and inference scheme on several regression and classification datasets. These were implemented using PyTorch [Paszke et al., 2017] and the code will be available upon acceptance. Additional results are included in the appendices. We use $M = 50$ inducing points for all experiments in this section. All experiments were run on a Macbook pro.

5.1 Synthetic classification example

We first illustrate the performance of the proposed model on a classification example. We generate a dataset of 100 data points and four classes, and use a BNN with one hidden layer of 50 hidden units with ReLU non-linearities, and two dimensional latent variables $\mathbf{z}$. Figure 4 shows the predictive performance of the proposed priors and various alternatives, including BNN (with unit Normal priors on weights) with mean field Gaussian variational approximation (MFVI) [Blundell et al., 2015] and Hamiltonian Monte Carlo (HMC) [Neal, 1993], variational deep kernel learning (DKL) Wilson et al. [2016b] and MetaNN [Karaletsos et al., 2018]. We highlight that *MetaGP-local* with RBF kernel gives uncertainty estimates that are reminiscent to that of a GP model in that the predictions express *"I don't know"* away from the training data, despite being a neural network under the hood. Following Bradshaw et al. [2017], we also show the uncertainty for data further from the training instances. *MetaGP-local*(RBF) remains uncertain, as expected, for these points while MFVI and DKL produce arguably overconfident predictions.

5.2 Inductive Biases For Neural Networks With Input-Dependent Kernels

We explore the utility of the input-dependent prior towards modeling inductive biases for neural networks and evaluate predictive performance on a regression example. In particular, we generate 100 training points from a synthetic sinusoidal function and create two test sets that contain in-sample inputs and out-of-sample inputs, respectively. We test an array of models and inference methods, including BNN (with unit

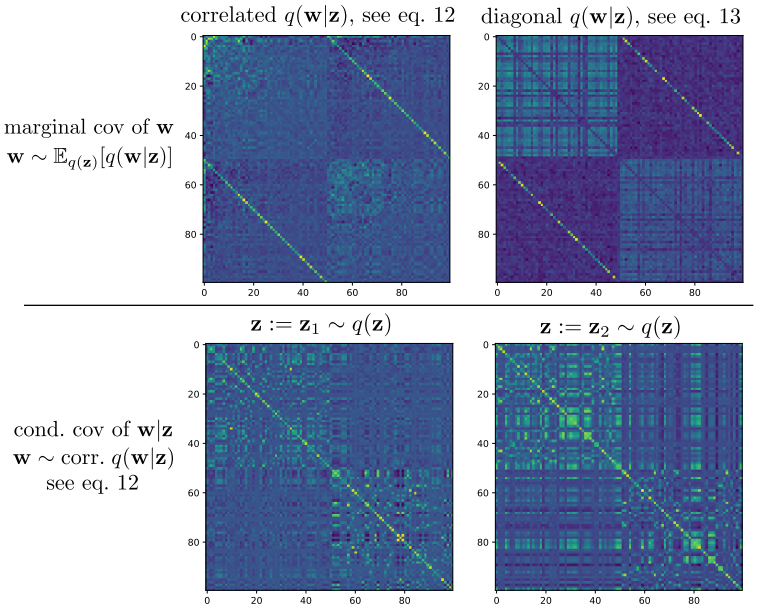

Figure 5: Marginal and conditional covariance structures over weights in a 1x50x1 BNN. Sampling from the posterior of the hierarchical model reveals that even a diagonal GP approximation can capture off-diagonal correlations induced through unit correlations. Also note the off-diagonal bands in the marginal plots above, which indicate the correlation structures induced by the latent variables of the hidden units connecting the layers.We remove the diagonal in the marginal plots for clarity.

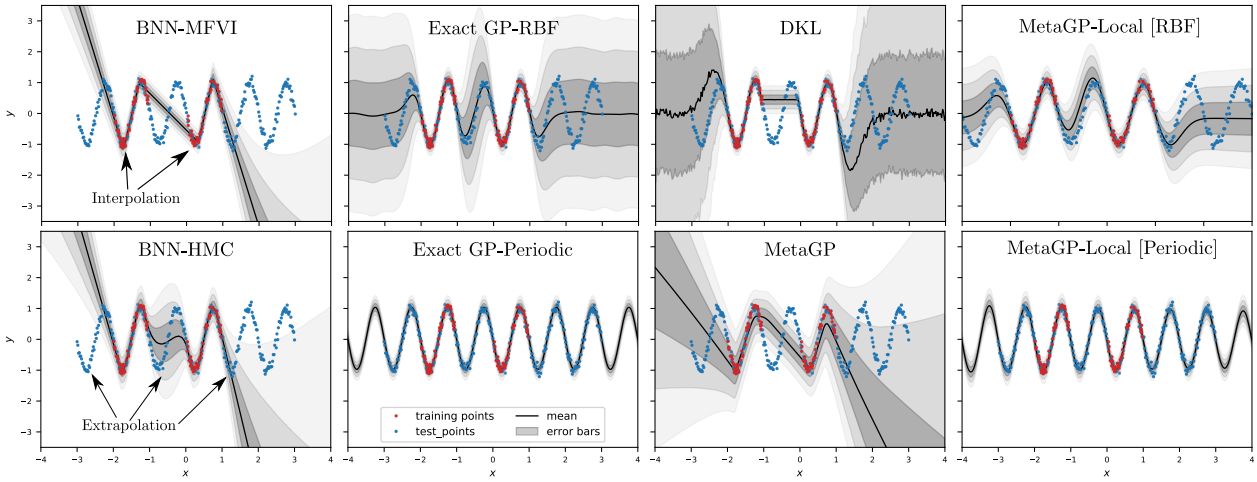

Figure 6: Predictive performance of various methods on a sinusoidal dataset. We also provide a quantitative comparison in Table 1.

Normal priors on weights) with MFVI and HMC, GPs with diverse kernel functions, DKL, MetaGP and local-MetaGP with input dependence given the same kernels as the GPs. We use RBF and periodic kernels [MacKay, 1998] for weight modulation and the pure GP in this example. Figure 6 summarizes the results. Note that the periodic kernel allows the BNN to discover and encode periodicity in its weights, leading to long-range confident predictions compared to that of the RBF kernel and significantly better extrapolation than BNNs with independent weight priors can obtain given the amount of training data, even when running HMC instead of VI.

We evaluate the quantitative utility of input-dependence and inductive biases on two test sets that contain in-sample inputs (between the training data) and out-of-sample inputs (outside the training range),

respectively. We report the performance of all methods in Table 1. The performance is measured by the root mean squared error (RMSE) and the negative log-likelihood (NLL) on the test set, and we explicitly evaluate separately for *extrapolation* and *interpolation*. In this example, the local MetaGP model is comparable to GP regression with a periodic kernel and superior to other methods, demonstrating good RMSE and NLL on both in-distribution and out-of-distribution examples.

Table 1: Average test error and negative log-likelihood for the sinusoid example in Fig. 6, averaged over five runs. Lower is better.

	Interpolation		Extrapolation	
Method	RMSE	NLL	RMSE	NLL
BNN-MFVI	0.17	-0.04	3.51	88.12
BNN-HMC	0.12	-0.69	4.34	10.98
Exact GP-RBF	**0.11**	**-0.81**	0.55	0.75
Exact GP-Periodic	**0.11**	-0.80	**0.11**	**-0.83**
DKL	0.12	-0.72	0.76	3.26
MetaGP	0.24	0.08	2.59	5.86
MetaGP-Local[RBF]	**0.11**	-0.80	0.74	1.50
MetaGP-Local[Periodic]	**0.11**	-0.76	0.12	-0.69

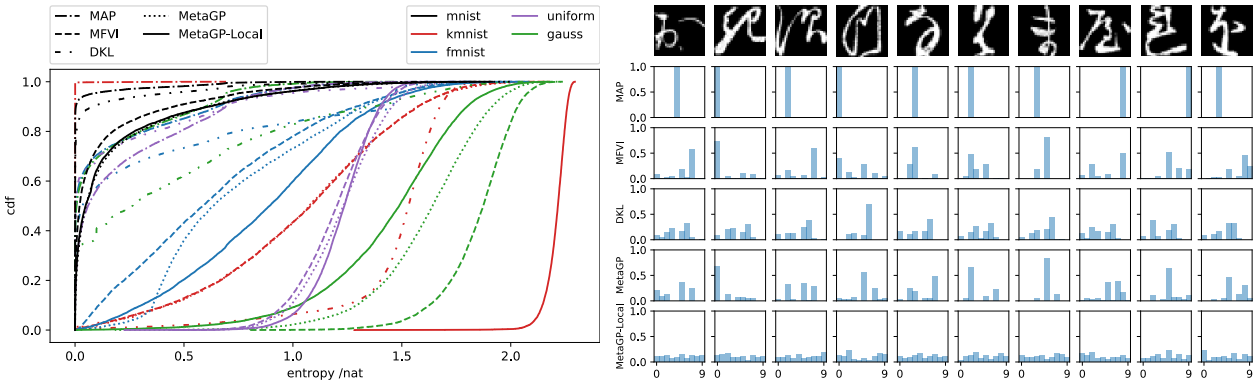

Figure 7: The CDFs of predictive entropies on in-distribution and out-of-distribution test sets for various methods [Left] and the predictive class probability for representative samples from out-of-distribution test sets [Right].

5.3 Input Dependent Neural Networks For Uncertainty Quantification

Motivated by the performance of the proposed *MetaGP-local* model in the synthetic examples in Figure 6, this section tests the ability of this model class to produce calibrated predictive uncertainty to out-of-distribution samples. That is, for test samples that do not come from the same training distribution, a robust and well-calibrated model should produce uncertain predictive distribution and thus high predictive entropy. Such a model could find applications in safety-critical tasks or in an area where detecting unfamiliar inputs is crucial such as active learning or reinforcement learning. In this experiment, we train a BNN classifier with one hidden layer of 100 rectified linear units on the MNIST dataset, with *MetaGP-local*-RBF only applied to the last layer of the network. The dimensions of the latent variables and the auxiliary inputs are both 2, with auxiliary inputs given by transforming MNIST images using a jointly learned linear projection $\mathbf{V}$. After training on MNIST, we compute the entropy of the predictions on various test sets, including notMNIST, fashionMNIST, Kuzushiji-MNIST, and uniform and Gaussian noise inputs. Following [Lakshminarayanan et al., 2017, Louizos and Welling, 2017], the CDFs of the predictive entropies for various methods are shown in Fig. 7. A calibrated classifier should give a CDF that bends towards the top-left corner of the plot for in-distribution examples and, vice versa, towards the bottom-right corner of the plot for out-of-distribution inputs. In most out-of-distribution sets considered, except Gaussian random noise, *MetaGP* and *MetaGP-local* demonstrate superior performance to all comparators, including DKL. Notably, MAP estimation, often

deployed in practice, tends to give wildly poor uncertainty estimates on out-of-distribution samples. We illustrate this behaviour and that of other methods on representative inputs of the Kuzushiji-MNIST dataset in Figure 7 and on MNIST digits in the appendix.

5.4 Active learning

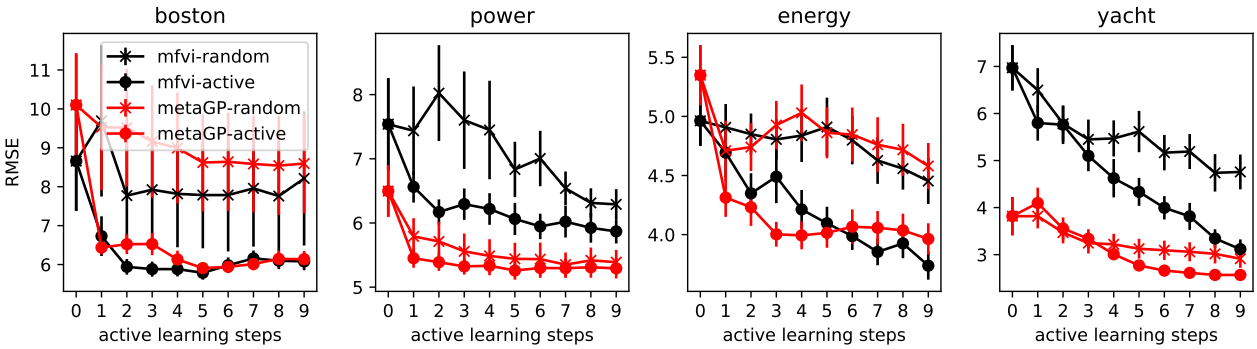

Figure 8: Active learning with BNNs using mean-field Gaussian variational inference [MFVI] and a meta-GP hierarchical prior [MetaGP] on several UCI regression datasets. Each trace shows the root mean squared error (RMSE) averaged across 40 runs.

We next stress-test the performance of the proposed model in a pool-based active learning setting for real-valued regression, where limited training data is provided initially and the target is to sequentially select points from a pool set to add to the training set. The criterion to select the next best point from the pool set is based on the entropy of the predictive distribution, i.e. we pick one with the highest entropy. Note that this selection procedure can be interpreted as selecting points that maximally reduce the posterior entropy of the network parameters Houlsby et al. [2011]. Four UCI regression datasets were considered, where each with 40 random train/test/pool splits. For each split, the initial train set has 20 data points, the test set has 100 data points, and the remaining points are used for the pool set, similar to the active learning set-up in Hernández-Lobato and Adams [2015]. We compare the performance of the proposed model and inference scheme to that of Gaussian mean-field variational inference and show the average results in Figure 8. Across all runs, we observe that active learning is superior to random selection and more crucially using the proposed model and inference scheme seems to yield comparable or better predictive errors with a similar number of queries.

6 Related work

There is a long history of research on developing (approximate) Bayesian inference methods for BNNs, i.e. in [Neal, 1993, 2012, Ghahramani, 2016]. Our work differs in that the model employs a hierarchical prior, and inference is done in a lower-dimensional latent space instead of the weight space. The variational approximation is chosen such that the marginal distribution over the weights is non-Gaussian and the correlations between weights are retained, in contrast to the popular mean-field Gaussian approximation. Imposing structure over the weights with a carefully chosen prior has been observed to improve predictive performance [Ghosh et al., 2018, Neal, 2012, Blundell et al., 2015], but it has remained elusive how to express prior knowledge or handle interpolation or extrapolation in such models. Modern deep Bayesian learning approaches often involve fusing neural networks and GPs, such as in deep kernel learning [Wilson et al., 2016b], which layers a GP on top of a neural network feature extractor. Another notable example is [Pearce et al., 2019], which blends kernels and activation functions to induce desired properties through architectural choices, but is not expressing these assumptions as a weight prior. The functional regularization approach introduced in [Sun et al., 2019] shares some of the motivations with our paper, but implements it very differently by explicitly instantiating a GP and performing a complex training scheme to learn neural networks that match that GP. Asymptotically, they match the GP, while in our model (i) the properties we

care about are already built into the weight prior allowing direct training on a dataset without the involved minimax approach, and (ii) our posterior can depart from that restrictive prior as it fundamentally only guides a weight based model, i.e. by learning posterior kernel parameters for k_{aux} to eliminate its influence on k_{local} (such as wide lengthscales).
Another related theme is hyper-networks, the core idea of which is to generate network parameters using another network [see e.g. Ha et al., 2016, Stanley et al., 2009]. Our model resembles a GP-LVM [Lawrence, 2004] hyper-GP, with a key structural assumption of node latent variables as introduced in [Karaletsos et al., 2018] to enable compact prediction per weight instead of per weight tensor.

7 Summary

We proposed a GP-based hierarchical prior over neural network weights, and a modification that permits input-dependent weight priors, along with an effective approximate inference strategy. We demonstrated utility of these models for interpolation, extrapolation, uncertainty quantification and active learning benchmarks, outperforming strong baselines. We plan to evaluate the performance of the model on more challenging decision making tasks.

References

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

J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

T. D. Bui, J. Yan, and R. E. Turner. A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *The Journal of Machine Learning Research*, 18(1):3649–3720, 2017.

S. Flaxman, A. Gelman, D. Neill, A. Smola, A. Vehtari, and A. G. Wilson. Fast hierarchical Gaussian processes. 2015.

Z. Ghahramani. A history of Bayesian neural networks. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

S. Ghosh, J. Yao, and F. Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. In *International Conference on Machine Learning*, pages 1739–1748, 2018.

D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

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

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

T. Karaletsos, P. Dayan, and Z. Ghahramani. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*, pages 329–336, 2004.

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

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

C. Louizos, X. Shi, K. Schutte, and M. Welling. Functional neural processes. *arXiv preprint arXiv:1906.08324*, 2019.

D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 1992.

D. J. MacKay. Introduction to Gaussian processes. *NATO ASI series. Series F: computer and system sciences*, pages 133–165, 1998.

A. G. D. G. Matthews, J. Hensman, R. E. Turner, and Z. Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *International Conference on Artificial Intelligence and Statistics*, 2016.

A. Miller, N. Foti, A. D'Amour, and R. P. Adams. Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems*, pages 3708–3718, 2017.

R. M. Neal. Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems*, pages 475–482, 1993.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

T. Pearce, M. Zaki, A. Brintrup, and A. Neely. Expressive priors in Bayesian neural networks: Kernel combinations and periodic functions. *Uncertainty In Artificial Intelligence*, 2019.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

T. Salimans and D. A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational Bayesian neural networks. *International Conference For Representation Learning*, 2019.

M. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, pages 1971–1979, 2014.

M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

B. Trippe and R. Turner. Overpruning in variational Bayesian neural networks. *arXiv preprint arXiv:1801.06230*, 2018.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016a.

A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016b.

A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt. Fixing variational Bayes: Deterministic variational inference for Bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

A Additional Background on Bayesian neural networks and variational inference

Consider a training set comprising of N input-output pairs, $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, and a neural network parameterized by weights and biases, $\mathbf{w}$, that describes the distribution over an output y_n given an input $\mathbf{x}_n$, $p(y_n|\mathbf{w}, \mathbf{x}_n)$. We follow a Bayesian approach by placing a prior distribution over the network parameters, $p(\mathbf{w})$, and obtaining the posterior distribution $p(\mathbf{w}|\mathcal{D})$, which involves calulation of the marginal likelihood $p(\mathcal{D}) = \int \mathrm{d}\mathbf{w} p(\mathbf{w})p(\mathcal{D}|\mathbf{w})$. However, obtaining $p(\mathbf{w}|\mathcal{D})$ and $p(\mathcal{D})$ exactly is intractable when N is large or when the network is large and as such, approximation methods are often required. In particular, mean-field Gaussian variational inference (MFVI) has recently become a method of choice for approximate inference for Bayesian neural networks due to its simplicity and the recently popularized *reparameterization trick* [Salimans and Knowles, 2013, Kingma and Welling, 2013, Titsias and Lázaro-Gredilla, 2014, Blundell et al., 2015]. MFVI sidesteps the intractability by positing a diagonal Gaussian approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ and optimising an approximate lower bound to the marginal likelihood $\mathcal{L}_{\mathrm{MFVI}}(q(\mathbf{w})) \approx -\mathrm{KL}[q(\mathbf{w})||p(\mathbf{w})] + \frac{1}{K}\sum_{k=1}^K \sum_{n=1}^N \log p(y_n|\mathbf{w}_k, \mathbf{x}_n)$, where $\mathbf{w}_k = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon_k$ and $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, i.e. $\mathbf{w}_k$ is a sample from $q(\mathbf{w})$. Note that the mean-field variational Gaussian approximation with a standard normal prior, presented in is often outperformed by point estimation in certain settings [Trippe and Turner, 2018]. Despite being practical and able to give reasonable uncertainty estimates, improving MFVI is still an active research area, and the main focuses of which are (i) improving the reparameterization gradient estimator to enable faster convergence [Miller et al., 2017, Wu et al., 2018], (ii) replacing the typical standard Normal prior, $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{I})$ by a structured prior that better models the structures present in the weight a-priori [Ghosh et al., 2018, Neal, 2012, Blundell et al., 2015], and (iii) using structured variational approximations that can potentially capture weight correlations in the posterior [Louizos and Welling, 2016, Zhang et al., 2017]. This paper builds on the two latter themes and proposes a hierarchical model for the prior and a structured variational scheme that explicitly model and infer weight structures.

B Extra experimental results

B.1 An empirical evaluation of various approximations for $q(\mathbf{w}|\mathbf{z}_k, \mathbf{V}_k, \mathbf{x})$

In this section, we analyze the impact of different approximations to the covariance matrix of $q(\mathbf{w}|\mathbf{z}_k, \mathbf{V}_k, \mathbf{x})$:

$$q(\mathbf{w}|\mathbf{z}_k, \mathbf{V}_k, \mathbf{x}) = \mathcal{N}(\mathbf{w}; \mathbf{A}^{(k)}\boldsymbol{\mu}_u, \mathbf{B}^{(k)} + \mathbf{A}^{(k)}\Sigma_{\mathbf{u}}\mathbf{A}^{\intercal,(k)}).$$

If we use the exact, fully correlated Gaussian distribution above, it is necessary to sample from this distribution to evaluate the lower bound. This step costs $\mathcal{O}(W^3)$ where W is the number of parameters in the network.

The complexity can be greatly improved by making a diagonal approximation to $\mathbf{B}^{(k)}$ as follows,

$$\hat{q}_{\mathrm{FITC}}(\mathbf{w}|\mathbf{z}_k, \mathbf{V}_k, \mathbf{x}) = \mathcal{N}(\mathbf{w}; \mathbf{A}^{(k)}\boldsymbol{\mu}_u, \mathrm{diag}(\mathbf{B}^{(k)}) + \mathbf{A}^{(k)}\Sigma_{\mathbf{u}}\mathbf{A}^{\intercal,(k)}).$$

Sampling from this distribution can be done in $\mathcal{O}(WM^2)$ where M is the number of pseudo-points.

This can be further approximated by assuming a diagonal covariance matrix,

$$\hat{q}_{\mathrm{diag}}(\mathbf{w}|\mathbf{z}_k, \mathbf{V}_k, \mathbf{x}) = \mathcal{N}(\mathbf{w}; \mathbf{A}^{(k)}\boldsymbol{\mu}_u, \mathrm{diag}(\mathbf{B}^{(k)} + \mathbf{A}^{(k)}\Sigma_{\mathbf{u}}\mathbf{A}^{\intercal,(k)})).$$

The variational bound can then be evaluated by drawing samples from $\hat{q}_{\text{diag}}$ as in the above approximation, or by drawing activity samples by employing the local reparameterization trick [Kingma et al., 2015].

We evaluate the performance of using the exact and approximate conditional distributions above in a range of toy regression and classification, and show representative results in Figs. 9 and 10. We note that the diagonal approximation is fast and gives qualitatively similar performance compared to more structured approximation or the exact case, in both cases where there is a single GP for all weights in the network and there is multiple GPs, one for each weight layer in the network.

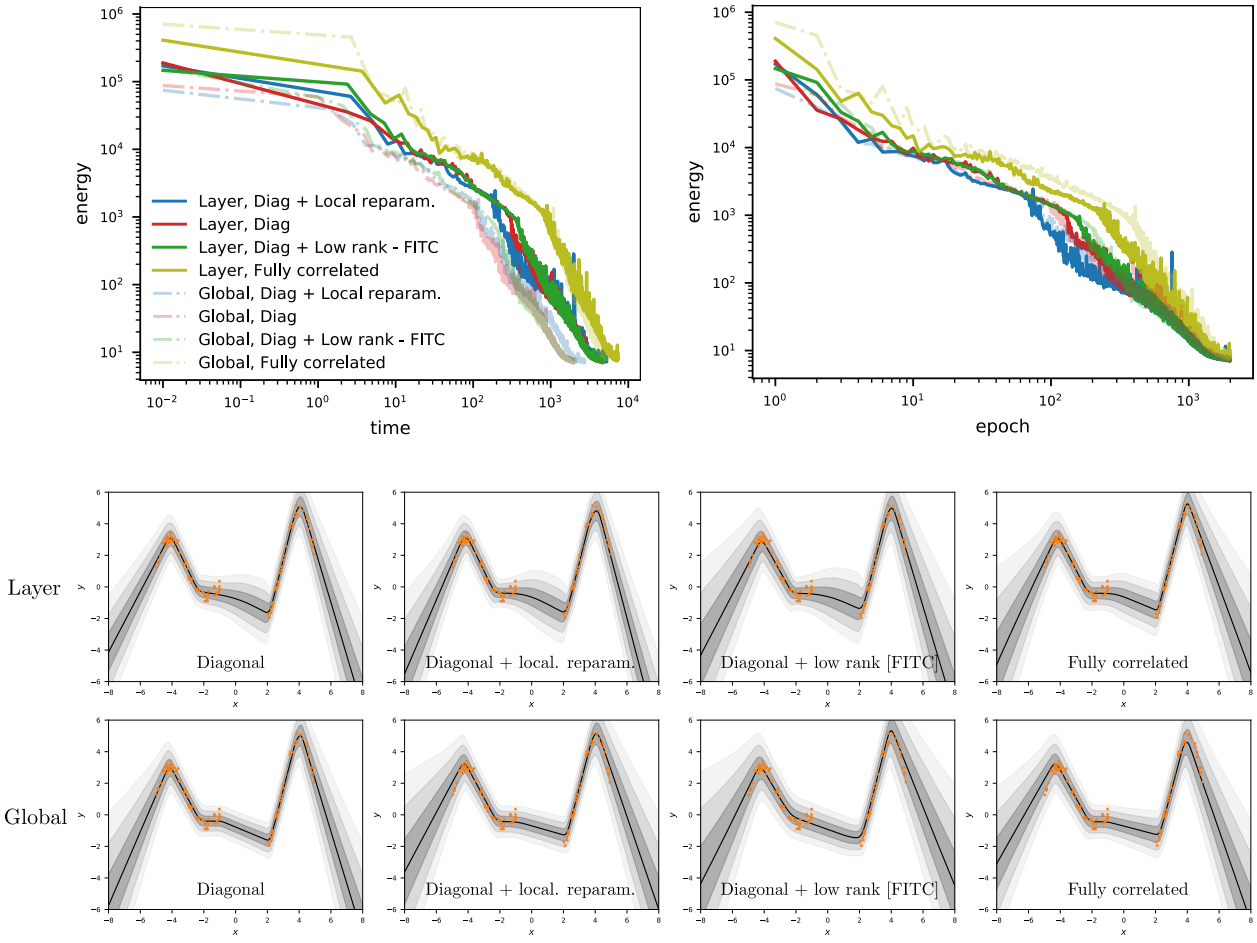

Figure 9: An evaluation of the covariance matrix approximations in a toy regression example. Top: objective function during training vs epoch/time. Bottom: Predictions after training using one of the approximations discussed in the text. Global: there is one GP for all weights in the network. Layer: there are multiple GPs, one for each weight layer in the network. Note that we are not using the auxiliary kernel here. Best viewed in colour.

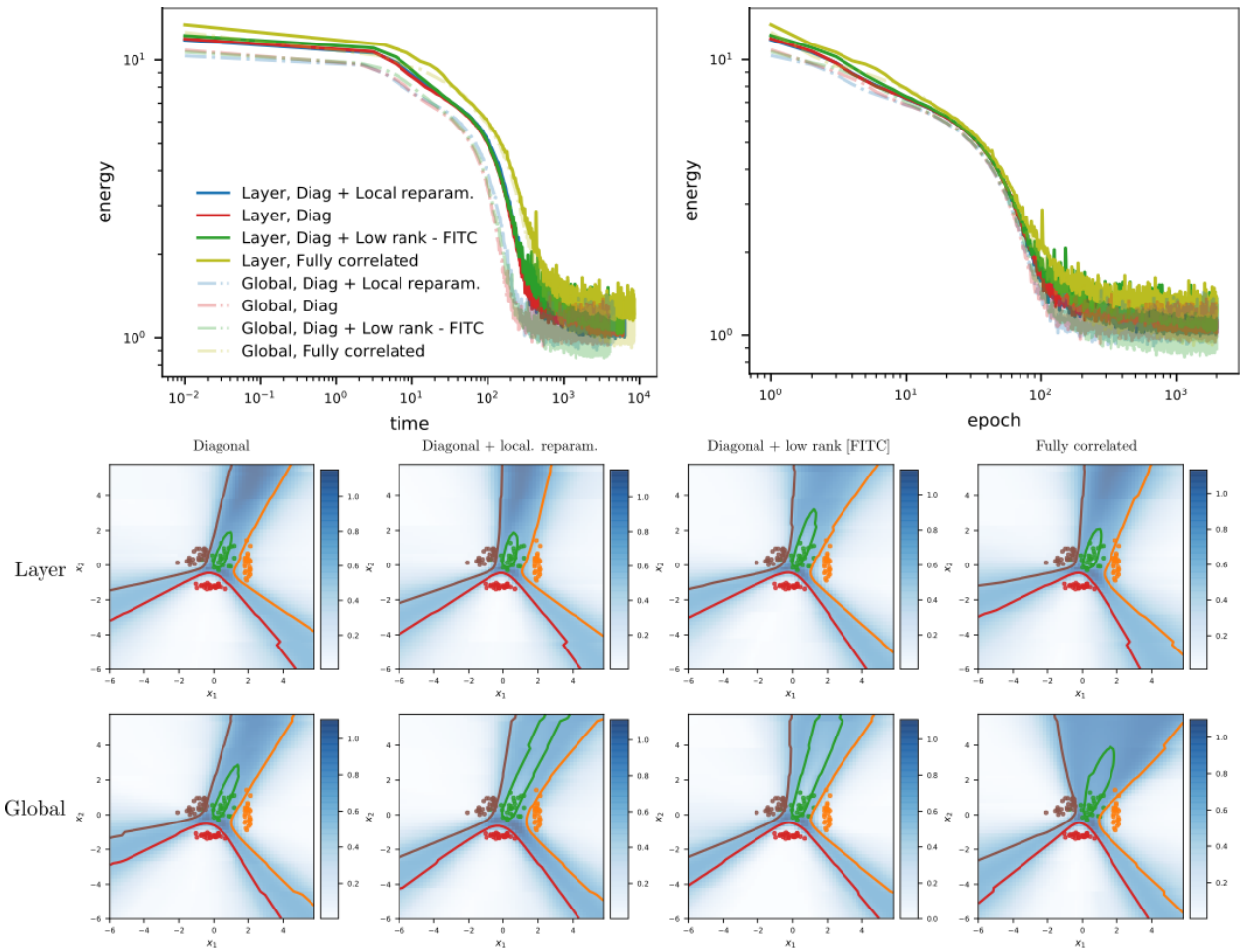

Figure 10: An evaluation of the covariance matrix approximations in a toy classification example. Top: objective function during training vs epoch/time. Bottom: Predictions after training using one of the approximations discussed in the text. Global: there is one GP for all weights in the network. Layer: there are multiple GPs, one for each weight layer in the network. Note that we are not using the auxiliary kernel here. Best viewed in colour.

B.2 Results on a synthetic regression example

In this section, we demonstrated the performance of the proposed priors on a 1D test function, as used in [Louizos et al., 2019]. We compare to BNN with independent Gaussian priors and a mean-field Gaussian variational approximation, and MetaNN [Karaletsos et al., 2018]. The training points, and predictive mean and error bars are shown in Fig. 11.

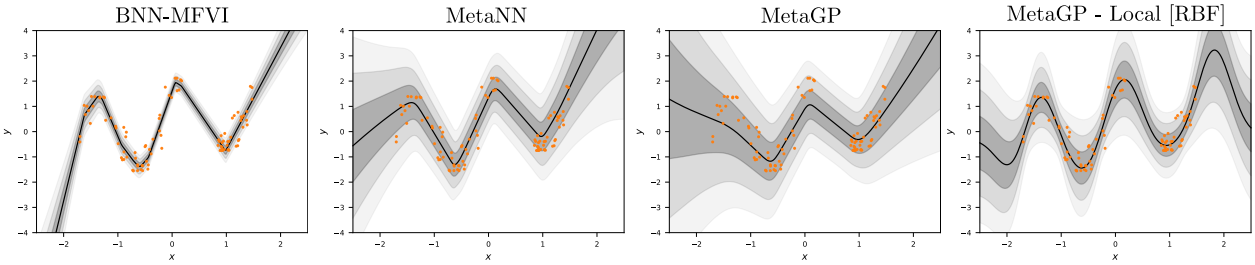

Figure 11: Predictive performance of various methods on a 1D test function. We compare the proposed approaches (MetaGP and MetaGP with an input-dependent kernel) to BNN-MFVI and MetaNN. Best viewed in colour.

B.3 Robustness in various data regimes for a toy regression problem

In this experiment, we evaluate the qualitative performance of various methods, including MFVI, MetaNN, MetaGP and MetaGP with local, input-dependent kernel, on a toy regression problem, in different data regimes. In particular, we considers 10, 20, and 50 training points respectively, and plot the predictions in Fig. 12. MetaGP demonstrates consistent performance across all data regimes, comparable to that of MetaNN. The input-dependent kernel helps the performance further in the out-of-distribution area.

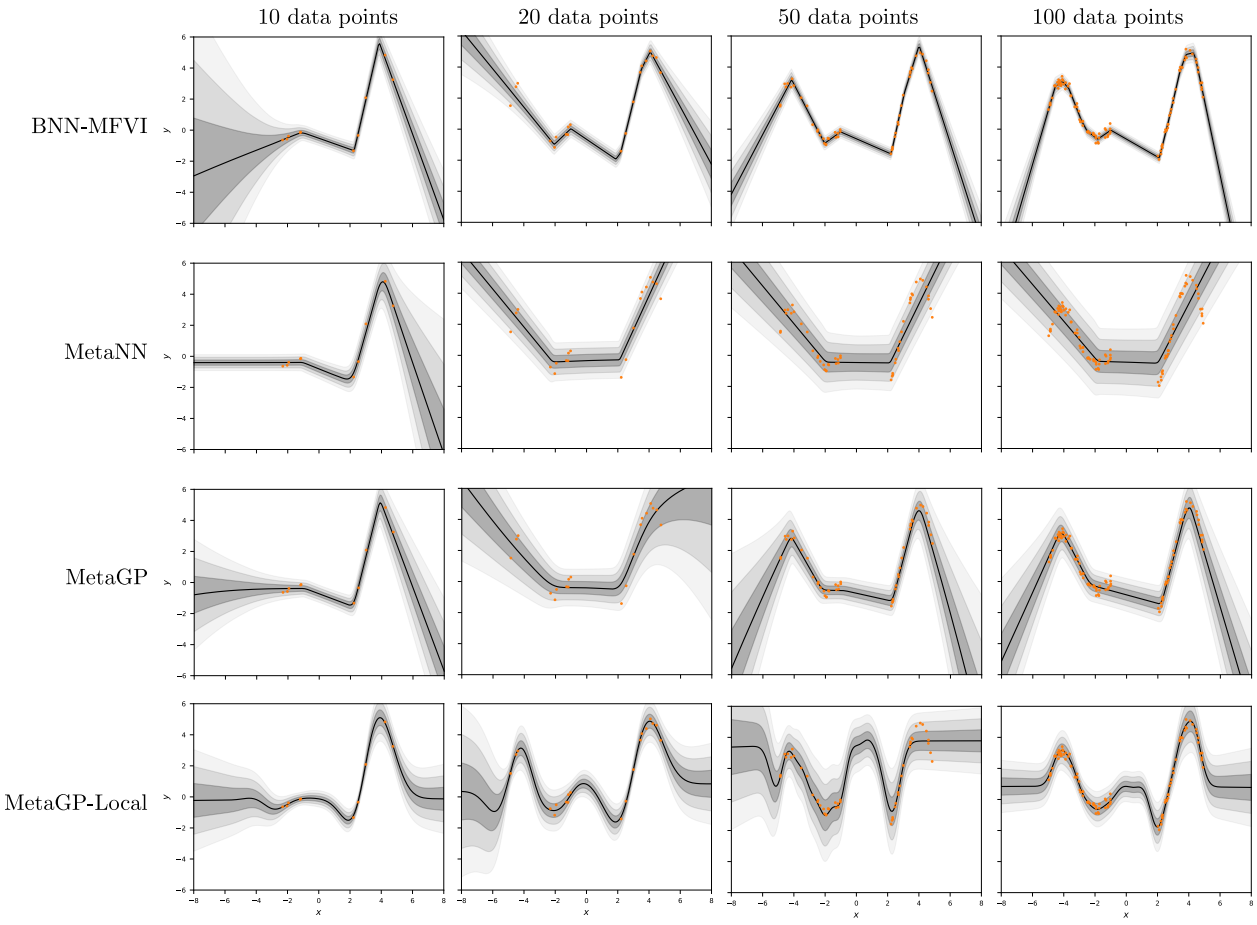

Figure 12: Performance of mean-field variational inference, MetaNN with variational inference and MetaGP with variational inference on a toy regression problem with various number of training points. Best viewed in colour.

B.4 Robustness of MetaGP with network architectures

In this experiment, we compare the performance of MetaGP for various numbers of hidden units (20, 50 and 100) and two activation functions (Tanh and ReLU) on a toy regression problem. The observation noise is fixed in this experiment. We observe that the performance of the models is in general consistent across different activation functions and numbers of hidden units. We show the results in Fig. 13.

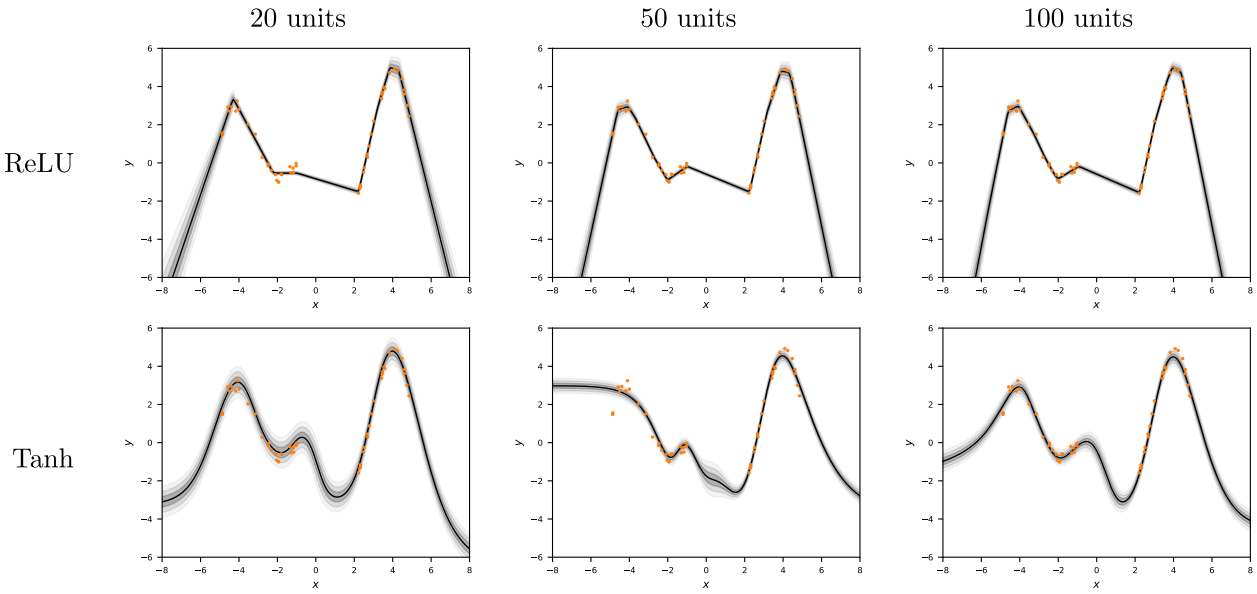

Figure 13: Performance of MetaGP on a toy regression problem, with various numbers of hidden units and different activation functions. Best viewed in colour.

B.5 Effect of input-dependent kernels

To understand the impact of the auxiliary kernel to the prediction, we use a model trained on the sinusoid dataset, as shown in the main text, and vary the period hyper-parameter in the kernel whilst keeping other hyper-parameters and variational parameters fixed. The predictions for a few hyperparameters are shown in Fig. 14. We note the variation/period in the data is captured by weight modulation, governed by the input-dependent kernel. Changing the period hyperparameter affects how fast or slow the weights are changing wrt the input.

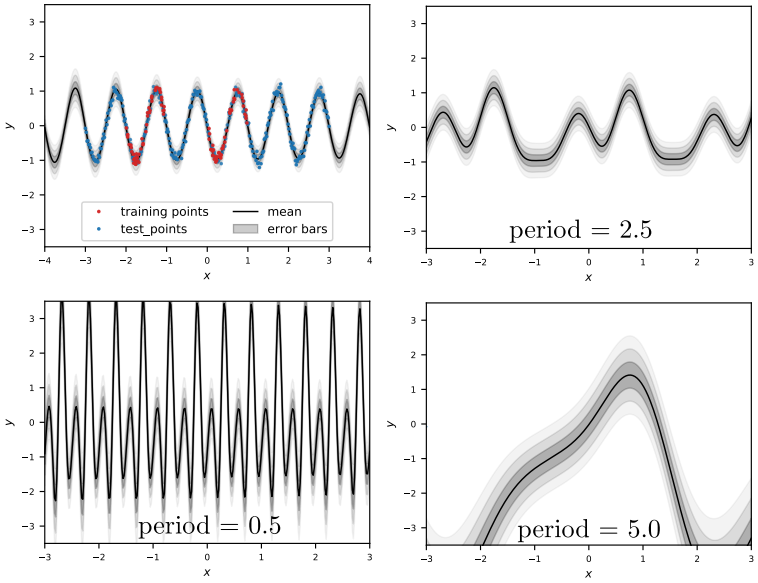

Figure 14: We first train a model with an input-dependent kernel on a sinusoid data set (top left) and then vary the period hyperparameter of the input-dependent kernel whilst keeping other hyperparameters and variational parametes fixed (others). Best viewed in colour.

B.6 MNIST experiment: full figures

In this section, we include the full figures of the MNIST out-of-distribution uncertainty experiment, as well as additional results using deep kernel learning [Wilson et al., 2016a]. In particular, we employ the same network architecture with the last layer being replaced by multiple independent GPs, one for each class (output dimension). As exact inference is intractable, variational inference based on inducing points is employed – we used 50 inducing points for each output. The full results of all models/methods considered are shown in Fig. 15. For clarify, the results of deep kernel learning and MetaGP are shown in **??**. MetaGP with the input-dependent kernel shows good performance, outperforming deep kernel learning in all cases. In addition, we include the full figures for the predictive distributions on representative test examples in Figs. 16 and 17.

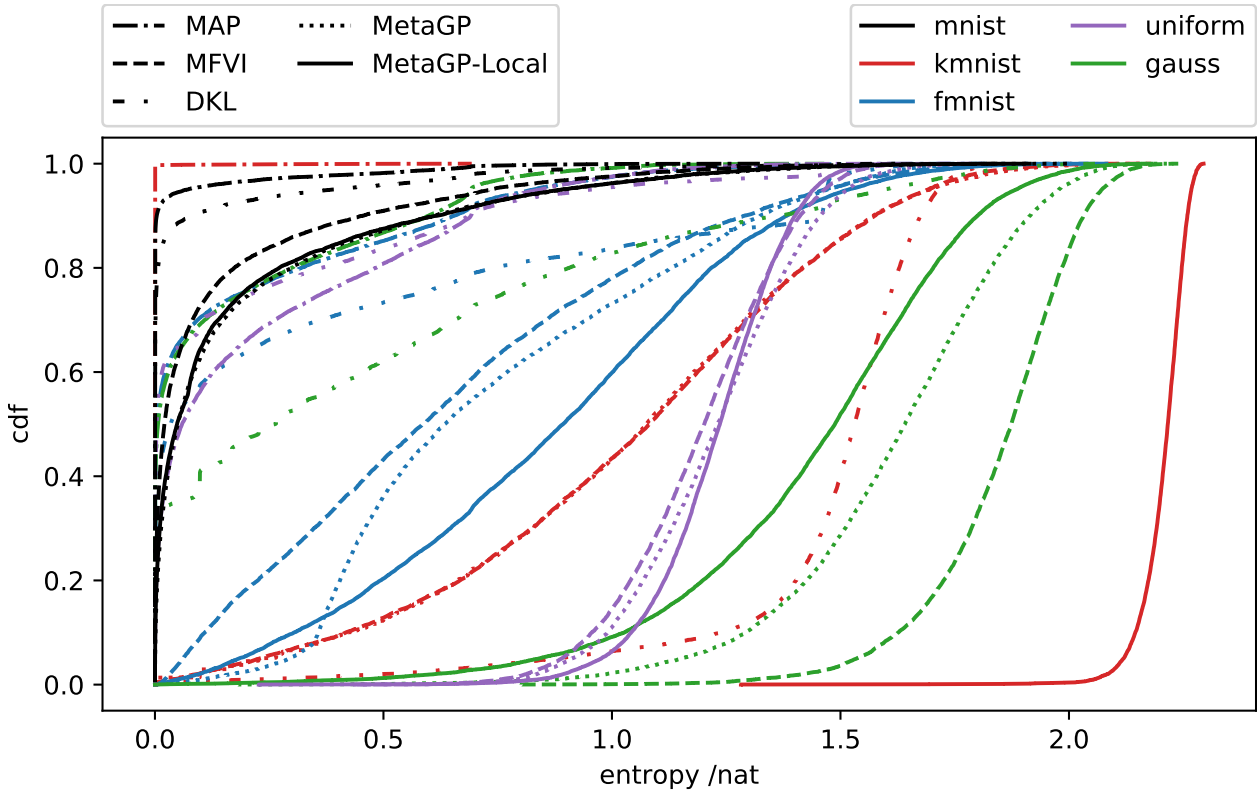

Figure 15: Full results of the MNIST out-of-distribution uncertainty experiment. Best viewed in colour.

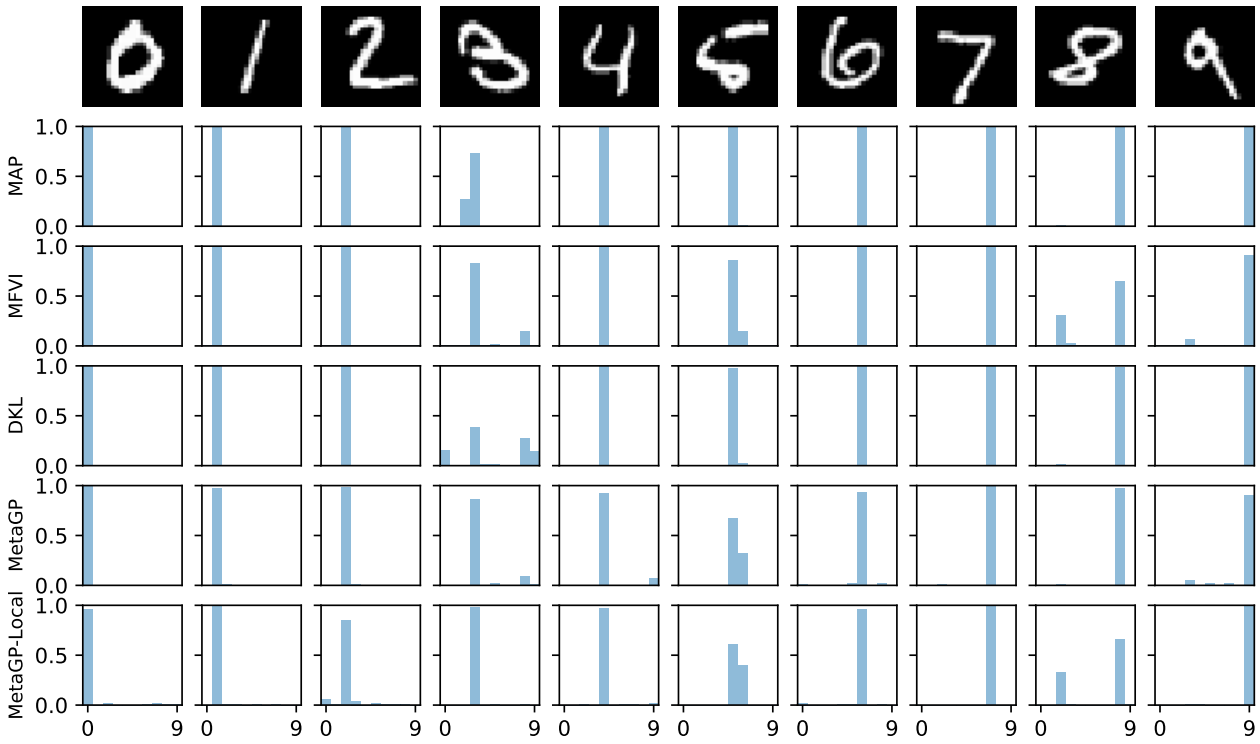

Figure 16: Predictive distribution for representative MNIST test examples by various methods. Best viewed in colour.

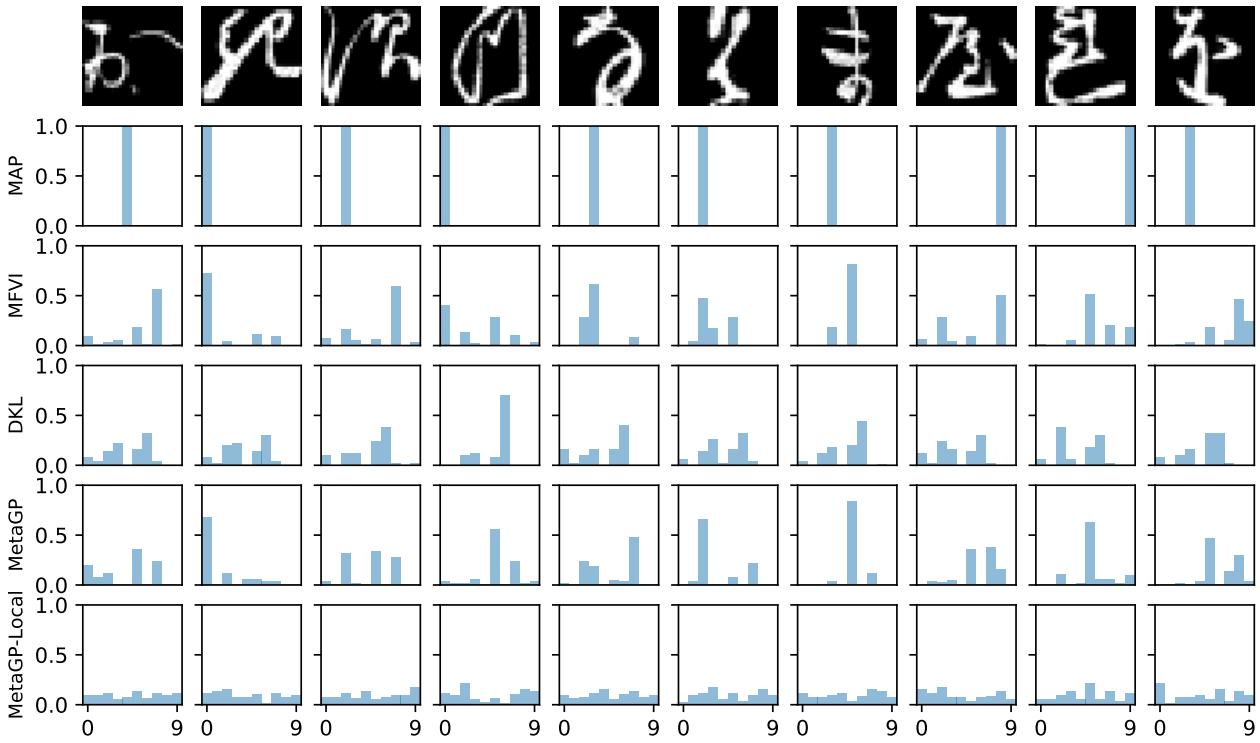

Figure 17: Predictive distribution for representative KMNIST test examples by various methods. Best viewed in colour.

B.7 A toy active learning problem

In this section, we provide a visualisation of the predictive performance of different methods in an active learning setting. Please see Fig. 18 and the associated caption.

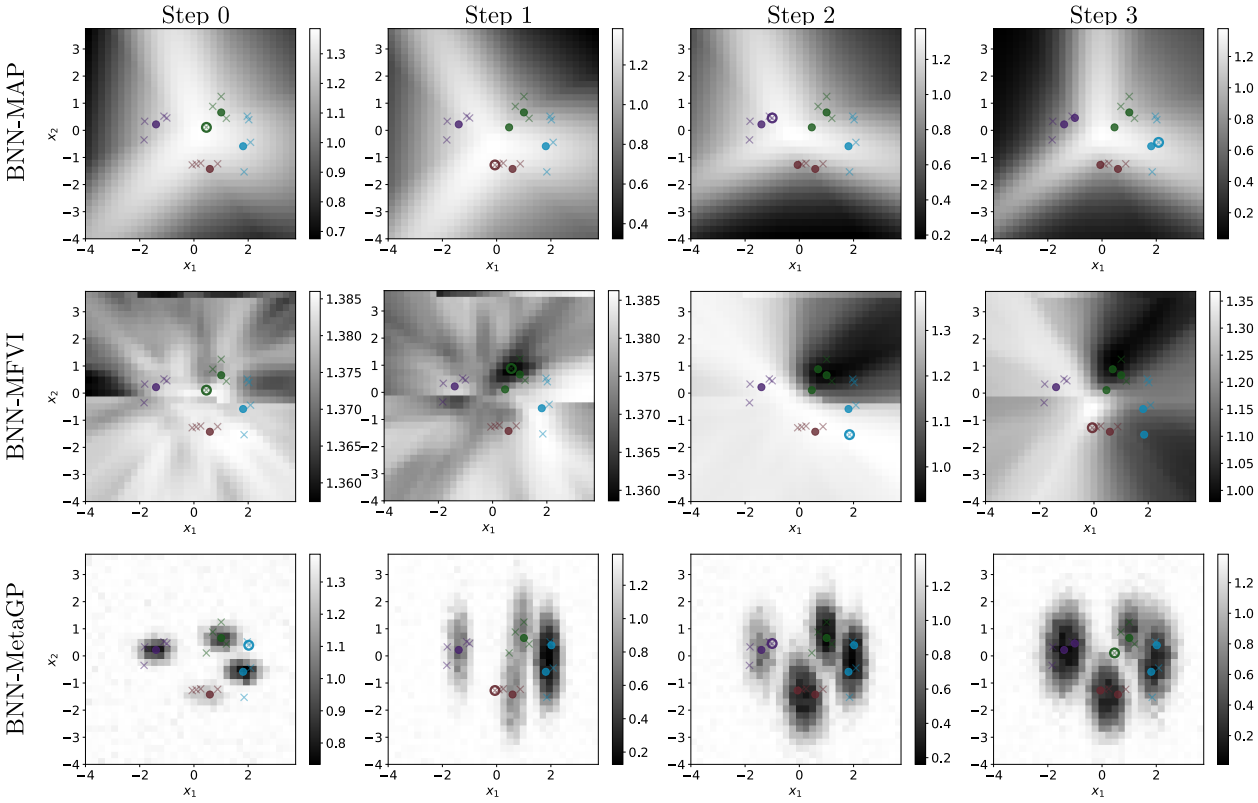

Figure 18: Active learning with BNNs using maximum a posteriori estimation [BNN-MAP], mean-field Gaussian variational inference [BNN-MFVI] and a meta-GP hierarchical prior [BNN-MetaGP] on a toy multi-class classification task. For each plot, the filled circle markers are the current training points, with different colours illustrating different classes. The shaded crosses are the examples in the pool set, one of which we wish to pick and evaluate to be included in the training set. The unfilled circle markers are the examples from the pool set selected at a step. The objective function for selecting points from the pool set is the entropy of the predictive probability. Best viewed in colour.

B.8 Applications to multi-task learning

We further investigate using the proposed model for multi-task learning. In particular, the latent variable $\mathbf{z}$ and corresponding hyper-parameters and variational parameters can be shared across different tasks whilst the meta mapping and the input-dependent kernel are private to each individual task. We first train the model on four regression tasks, each corresponds to a sinusoid of a particular frequency. At test time, a novel test set is shown to the model. The hyper-parameters of the input kernel and variational parameters corresponding to this new test set are optimised while other hyper-parameters and the latent variables are kept fixed. We evaluate the performance of the model on the novel test sets to see how the latent variables can be reused and shared across tasks to facilitate fast adaptation to new settings. The performance of the model on the tasks used for training and new tasks at test time is shown in Fig. 19. This result demonstrates the ability of the model trained with multiple similarly related tasks to faithfully and quickly adapt to new settings.

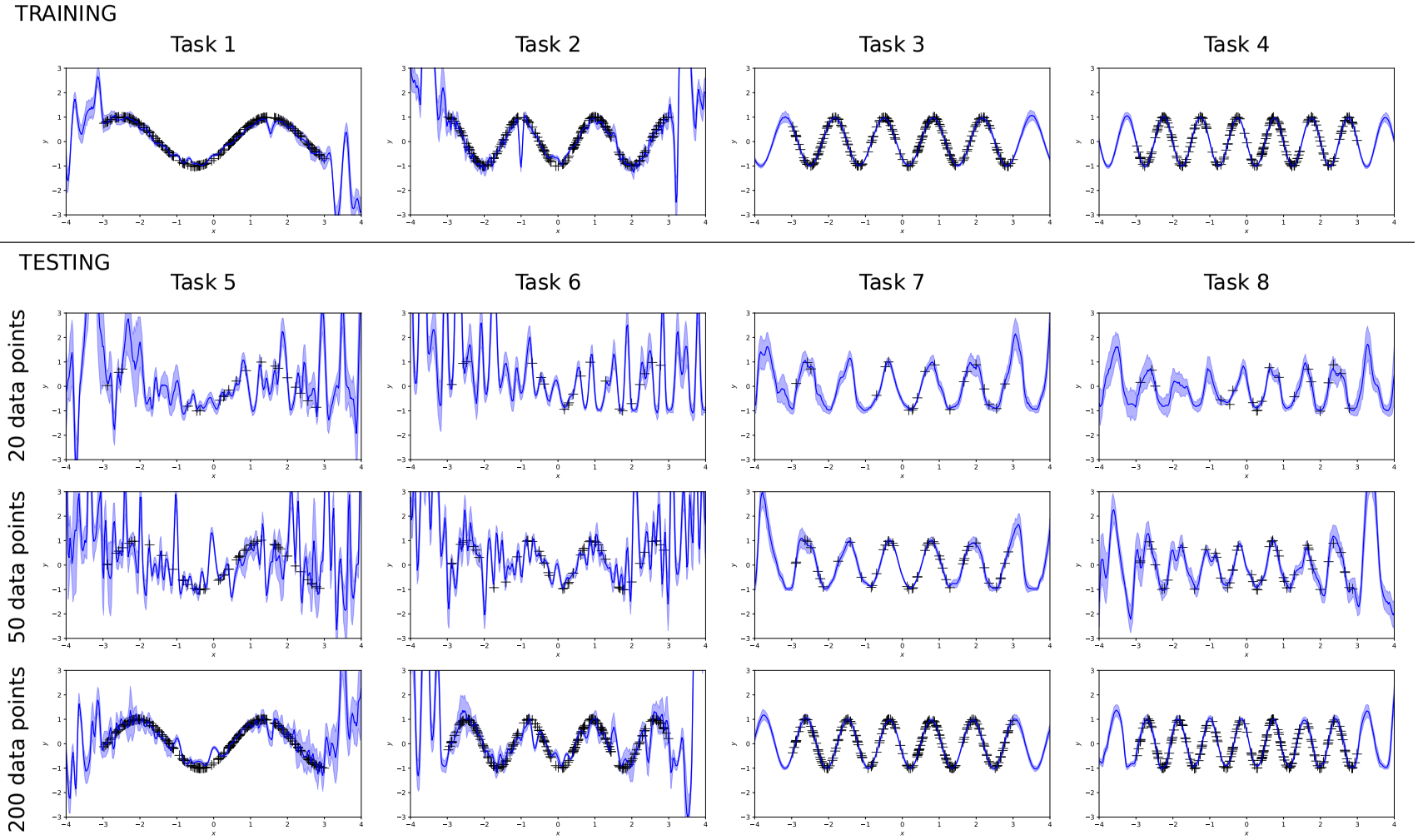

Figure 19: Training on multiple related tasks and adaptation to novel tasks at test time. In this case, the latent variables (as well as weight code hyperparameters) are shared across tasks while each individual has its own input-dependent kernel. At test time, only the private parameters for the new task are re-initialised and optimised. Best viewed in colour.