# Characterizing and Warping the Function space of Bayesian Neural Networks

**Daniel Flam-Shepherd**
University of Toronto

**James Requeima**
University of Cambridge

**David Duvenaud**
University of Toronto

## Abstract

In this work we develop a simple method to construct priors for Bayesian neural networks that incorporates meaningful prior information about functions. This method allows us to characterize the relationship between weight space and function space.

## 1 Introduction and Motivation

In Bayesian neural networks (BNNs), we place prior distributions over weights in order to maintain a measure of meaningful uncertainty. However, this view is limited as it is difficult to incorporate meaningful prior information about functions. In order to address this, recent work [1] attempts to map Gaussian process (GP) priors to BNN priors. This is done by minimizing the KL divergence in function space of the BNN prior to the GP prior.

There are a few fundamental limitations with this work,

1. The prior distributions over functions $p_{\text{BNN}}(\boldsymbol{f})$ can't be accurately estimated.
2. A diagonal Gaussian prior is optimized in weight space but there is no way of characterizing the true weight space prior corresponding to a function space prior.
3. Their method works only with distributions of functions with defined densities $p(\boldsymbol{f})$ and will not work with implicit distribution of functions.

In this work, we build on [1] to find more principled priors for BNNs in function space. We demonstrate a new method for constructing priors that address all issues with [1]. Our method avoids the use of approximate inference and requires no estimates of intractable distributions over functions. It also works on implicit distributions of functions and allows us to roughly characterize the exact distribution on weights that corresponds to some specific distribution of functions.

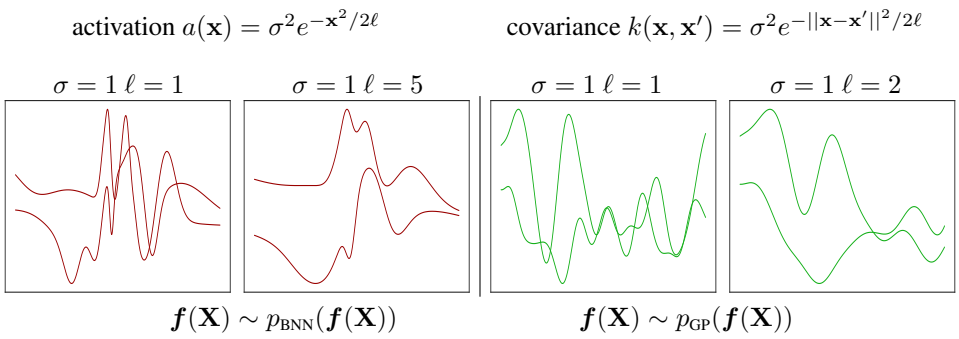



Figure 1: The function space of a BNN prior with an RBF activation function compared to a GP prior with an RBF covariance function.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

## 2 Background

### 2.1 Bayesian NNs and Mean field Variational Inference

For any dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, to specify a BNN we place priors on its parameters $p(\mathbf{w})$, then given a likelihood $p(\mathcal{D}|\mathbf{w})$ we can use variational inference to learn an approximation $q_{\boldsymbol{\phi}}(\mathbf{w})$ to the true posterior $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D})$ where $p(\mathcal{D}) = \int p(\mathcal{D}, \mathbf{w})d\mathbf{w}$ is the marginal likelihood. We determine $\boldsymbol{\phi}$ by maximizing a lower bound $\mathcal{L}(\boldsymbol{\phi})$ on the marginal log-likelihood:

$$\log p(\mathcal{D}) = \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}\left[\frac{p(\mathcal{D}, \mathbf{w})}{q_{\boldsymbol{\phi}}(\mathbf{w})}\right] \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}\left[\log \frac{p(\mathcal{D}, \mathbf{w})}{q_{\boldsymbol{\phi}}(\mathbf{w})}\right] \tag{1}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w})] - \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w})] \tag{2}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\log p(\mathcal{D}, \mathbf{w})] + \mathbb{H}[q_{\boldsymbol{\phi}}(\mathbf{w})] \tag{3}$$

The first term in (3) encourages $q_{\boldsymbol{\phi}}(\mathbf{w})$ to focus where the model puts high probability, $p(\mathcal{D}, \mathbf{w})$. While the entropy encourages $q_{\boldsymbol{\phi}}(\mathbf{w})$ to avoid concentrating mass to one area. In mean-field variational inference we specify a approximate posterior that factorizes ie $q_{\boldsymbol{\phi}}(\mathbf{w}) = \prod_i q_{\boldsymbol{\phi}}(\mathbf{w}_i)$ the diagonal gaussian $q_{\boldsymbol{\phi}}(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. We note that the prior on $\mathbf{w}$ induces an implicit prior on functions which we can sample from :

$$\mathbf{w} \sim p(\mathbf{w}), \;\; \mathbf{X} \sim p(\mathbf{X}), \;\; \boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) \sim p_{\text{BNN}}(\boldsymbol{f}(\mathbf{X})) \tag{4}$$

However, we cannot evaluate $p_{\text{BNN}}(\boldsymbol{f}(\mathbf{X}))$ point-wise.

### 2.2 Gaussian processes

A Gaussian process (GP) defines a distribution $p(\boldsymbol{f})$ over functions on some domain such that for any set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ the function values $\boldsymbol{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n))$ have a multivariate Gaussian distribution. Gaussian processes are parameterized by a mean function $\boldsymbol{\mu}(\cdot)$ and a covariance function $k(\cdot, \cdot)$. Any marginal distribution of function values is given by

$$p(\boldsymbol{f}(\mathbf{X})) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}), k(\mathbf{X}, \mathbf{X})) \text{ where } \boldsymbol{\mu}(\mathbf{X}) = \mathbb{E}[\boldsymbol{f}(\mathbf{X})], \;\; k(\mathbf{X}, \mathbf{X}) = \text{Cov}[\boldsymbol{f}(\mathbf{X}), \boldsymbol{f}(\mathbf{X})] \tag{5}$$

most kernels have hyperparameters which can be optimized by maximizing the log marginal likelihood under $\boldsymbol{f}(\mathbf{X})$. Sampling a function $\boldsymbol{f}(\mathbf{X})$ from a GP prior is straight forward .

$$\boldsymbol{f}(\mathbf{X}) = \boldsymbol{\mu}(\mathbf{X}) + k(\mathbf{X}, \mathbf{X})^{\frac{1}{2}}\mathbf{z}, \;\; \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}) \tag{6}$$

where $k(\mathbf{X}, \mathbf{X})^{\frac{1}{2}}$ denotes the Cholesky decomposition of the covariance matrix. BNNs with infinitely wide single hidden layers and certain prior distributions correspond to GPs [2]. As well, a one-layer BNN with non-linearity $\sigma()$ and mean-field Gaussian prior is approximately equivalent to a GP [3] with kernel function

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^\top \mathbf{x} + b)\sigma(\mathbf{w}^\top \mathbf{x}' + b)d\mathbf{w}db \tag{7}$$

There is also an approximate relationship between BNN activation functions and GP covariance functions. Certain activation and covariance functions will yield similar functions sampled from $p_{\text{BNN}}(\boldsymbol{f}(\mathbf{X}))$ and $p_{\text{GP}}(\boldsymbol{f}(\mathbf{X}))$, see the table below.

| function property | activation function | kernel function |
|---|---|---|
| smoothness | $a_{\text{RBF}}(\mathbf{x}) = \sigma^2 \exp(-\mathbf{x}^2/\ell)$ | $k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\|\|\mathbf{x} - \mathbf{x}'\|\|^2/2\ell)$ |
| periodic | $a_{\text{PER}}(\mathbf{x}) = \sigma^2 \sin(\mathbf{x}/\ell)$ | $k_{\text{PER}}(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-2\sin^2(\pi k\|\|\mathbf{x}-\mathbf{x}'\|\|^2)/\ell}$ |
| scale variation | $a_{\text{RQ}}(\mathbf{x}) = \sigma^2 \tanh(\mathbf{x}/\ell)$ | $k_{\text{RQ}}(\mathbf{x}, \mathbf{x}') = \sigma^2(1 + \|\|\mathbf{x} - \mathbf{x}'\|\|^2/2\ell^2)$ |
| noise | $a_{\text{WN}}(\mathbf{x}) = \sigma^2\delta(\mathbf{x})$ | $k_{\text{WN}}(\mathbf{x}, \mathbf{x}') = \sigma^2\delta_{\mathbf{x},\mathbf{x}'}$ |
| linear | $a_{\text{LIN}}(\mathbf{x}) = \sigma^2\mathbf{x}$ | $k_{\text{LIN}}(\mathbf{x}, \mathbf{x}') = \sigma^2\mathbf{x}^\top\mathbf{x}$ |
| constant | $a_{\text{CON}}(\mathbf{x}) = \sigma^2$ | $k_{\text{CON}}(\mathbf{x}, \mathbf{x}') = \sigma^2$ |

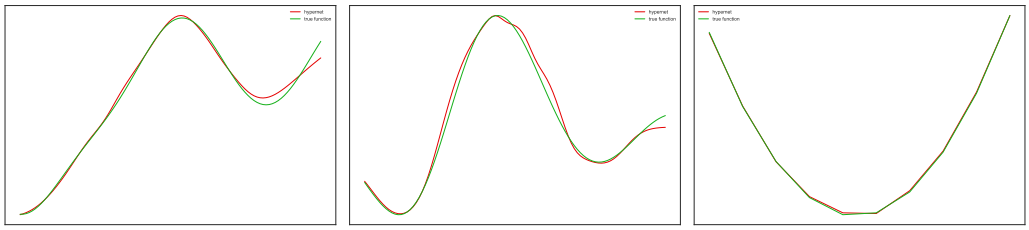


Figure 2: Functions fit by neural networks using weights output by a hypernetwork. Green is the true function. Red is a neural network with weights output by a hypernetwork.

## 3 Function Space priors for BNNs

### 3.1 HyperNetworks

HyperNetworks are recently introduced type of neural network that are used to generate the weights of another another primary network [4]. The hypernetwork and primary net together form a single model trained by backpropagation. A large primary network requires an impractically large hypernet since the number of weights in a neural network scales quadratically in the number of units per layer.

### 3.2 Fitting functions using a hypernetwork

To construct our prior we fit neural networks to a larger number of samples from a distribution over functions $p(\boldsymbol{f}(\mathbf{X}))$, which could be a GP prior or some implicit distribution. To speed training up we use a hypernetwork $\boldsymbol{h}_{\boldsymbol{\lambda}}$ which takes in a function $\boldsymbol{f}$ and its inputs $\mathbf{X}$ and outputs a set of weights of a neural network $\hat{\boldsymbol{f}}(\mathbf{X}, \mathbf{w})$ that will fit that function : that is

$$\mathbf{w} = \boldsymbol{h}_{\boldsymbol{\lambda}}(\boldsymbol{f}, \mathbf{X}) \text{ such that } \hat{\boldsymbol{f}} \approx \boldsymbol{f} \tag{8}$$

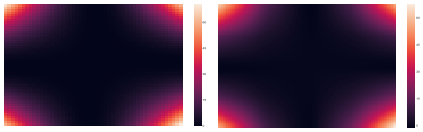
We optimize the following loss function to do so :

$$\mathcal{L}_{\mathbf{X},\boldsymbol{f}}(\boldsymbol{\lambda}) = \mathbb{E}_{\mathbf{w}\sim\boldsymbol{h}}\mathbb{E}_{\boldsymbol{f}\sim p(\boldsymbol{f}(\mathbf{X}))}[||\boldsymbol{f} - \hat{\boldsymbol{f}}(\mathbf{X}, \mathbf{w})||^2] \tag{9}$$

Using the optimized hypernet we can sample a large number of weights using samples from $p(\boldsymbol{f}(\mathbf{X}))$. Then we can take the sample mean and variance of the weights as the parameters a Gaussian distribution. This distribution can be used as our prior. However, using the hypernet limits the size of the primary network and the number of inputs $\mathbf{X}$ we can use when we construct our prior.

### 3.3 HyperNetwork accuracy

It is difficult to determine whether or not the functions the hypernetwork yield have the right properties. Also, mean square error can only tell us accuracy. We also plot the empirical covariance heatmap of the hypernet's functions to assess its quality (example heatmaps to the right).



$\text{Cov}_{\boldsymbol{f}\sim p(\boldsymbol{f})}[\boldsymbol{f}, \boldsymbol{f}']$ $\text{Cov}_{\boldsymbol{f}_{\mathbf{w}\sim\boldsymbol{h}}}[\boldsymbol{f}, \boldsymbol{f}']$

**Algorithm 1** warping $p_{\text{BNN}}(\boldsymbol{f})$

**Require** $p(\mathbf{X}), p(\boldsymbol{f})$ ▷ specifiy our prior data distribution and function distribution
$\{\mathbf{X}^s\}_{s=1}^S \sim p(\mathbf{X}), \{\boldsymbol{f}^s\}_{s=1}^S \sim p(\boldsymbol{f})$ ▷ sample data and functions
**Initialize** $\boldsymbol{\lambda}$ ▷ initialize our hypernet's parameters
**while** $\boldsymbol{\lambda}$ not converged **do**
  $\mathbf{w}^s = \boldsymbol{h}_{\boldsymbol{\lambda}}(\boldsymbol{f}^s, \mathbf{X}^s)$ ▷ output weights from hypernet
  $\hat{\boldsymbol{f}} = \boldsymbol{f}(\mathbf{X}, \mathbf{w}^s)$ ▷ make a function prediction
  $\mathcal{L}_{\mathbf{X},\boldsymbol{f}}(\boldsymbol{\lambda}) = \mathbb{E}_{p(\boldsymbol{f}(\mathbf{X}))}[||\boldsymbol{f} - \hat{\boldsymbol{f}}||^2]$ ▷ evalaute loss
  $\mathbf{g}_{\boldsymbol{\lambda}} \leftarrow \nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathbf{X},\boldsymbol{f}}(\boldsymbol{\lambda})$ ▷ get gradients of the loss
  $\boldsymbol{\phi} \leftarrow \text{adam}(\boldsymbol{\lambda}, \mathbf{g}_{\boldsymbol{\lambda}})$ ▷ update parameters
**Return** $\{\mathbf{w}^s\}_S \sim \boldsymbol{h}_{\boldsymbol{\lambda}^*}(\boldsymbol{f}^s, \mathbf{X}^s)$ ▷ Return sampled weights from the optimized hypernet
**Return** $p(\mathbf{w}|\boldsymbol{\phi}^*)$ where $\boldsymbol{\phi}^* = \{\boldsymbol{\mu}^* = \frac{1}{S}\sum_s \mathbf{w}_s,\ \boldsymbol{\sigma}^2 = \frac{1}{S}\sum_s(\mathbf{w}_s - \boldsymbol{\mu}_s)^2\}$ ▷ Return prior

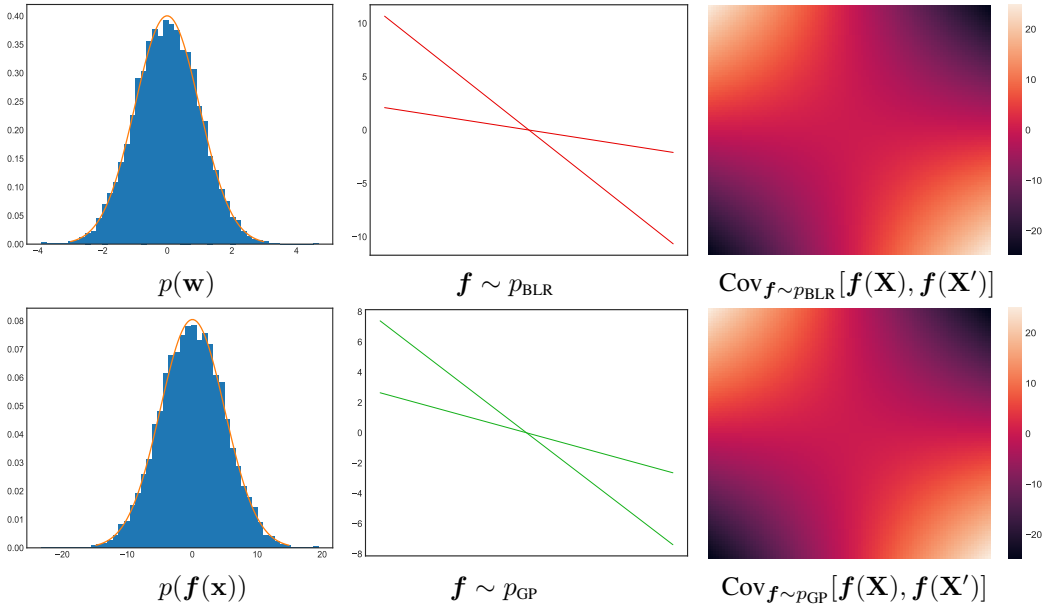



Figure 3: Sanity check: First column displays histograms of the marginal on the slope and one randomly selected function evaluation $\boldsymbol{f}(x)$. Second column displays draws of functions using our constructed prior and samples from the linear covariance GP used to construct the prior on weights. The third column shows the heatmap of covariance for both as well. Notice that the heatmaps are essentially identical. It seems that we have managed to warp $p_{\text{GP}} \sim p_{\text{BLR}}$ over some $p(\mathbf{X})$.

## 4 Experiments

We conduct a few exploratory experiments with our prior:

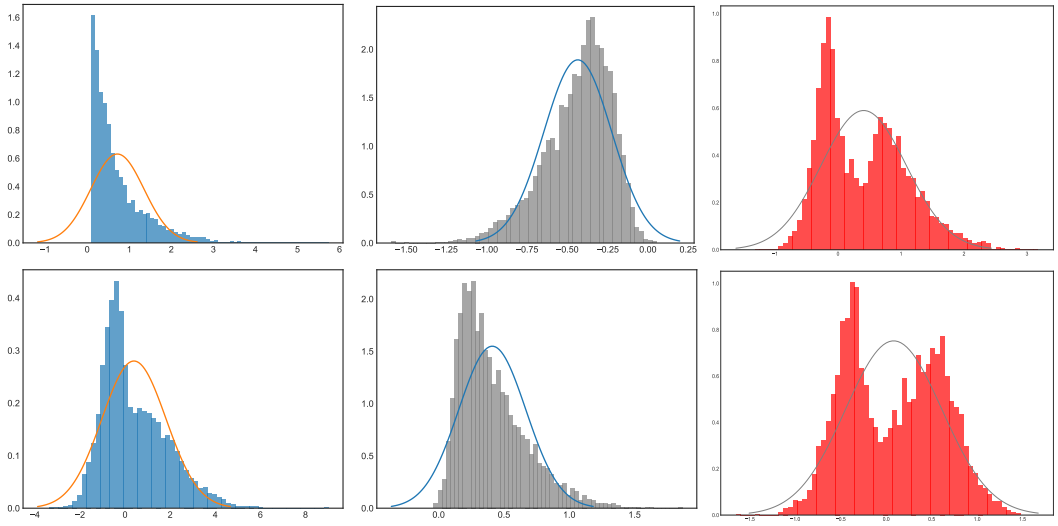
1. A sanity check with linear functions
2. An investigation into weight space - function space correspondence
3. Evaluating the impact on prior function space with our method

We conduct three different investigations into function space priors. The first is a sanity test, to check if our method works in the simplest scenario where we know exactly what the prior in weight-space is: the bayesian linear regression (BLR). The second is an investigation in the weight-space corresponding to certain function spaces. In the third, we seek to assess whether or not we can warp the prior function space of BNNs to incorporate certain properties of GPs or implicit distributions that we are interested in.

### 4.1 Sanity check with linear functions

We first test our prior construction on a problem where we know the exact solution: Bayesian linear regression. Recall that using a linear covariance function in a Gaussian process corresponds to bayesian linear regression with a zero mean Gaussian prior on the parameters. We seek to reconstruct this prior with our method: by fitting neural networks to samples from the corresponding GP. Subsequently, we then take the empirical mean and standard deviation to use as the parameters of our prior.

We manage to recover the correct weight space prior. Evidence of this is displayed in Figure 3: the top left plot is the reconstructed prior on the slope, a zero mean Gaussian as we expected. The BLR model also samples reasonable lines similar to the GP (top middle) and has identical function space covariance (right). We also check a single marginal $p(\boldsymbol{f}(\mathbf{x}))$ which is Gaussian as expected.



$a_{\text{Tanh}}(\mathbf{x})$ and $k_{\text{LIN}}(\mathbf{x}, \mathbf{x}')$ $a_{\text{Relu}}(\mathbf{x})$ and $k_{\text{RBF}}(\mathbf{x}, \mathbf{x}')$ $a_{\text{Sigmoid}}(\mathbf{x})$ and $k_{\text{PER}}(\mathbf{x}, \mathbf{x}')$

Figure 4: Each column displays the histogram of weights randomly selected from neural networks with certain activation functions trained to fit samples of GP priors with specific covariance functions

### 4.2 Analyzing the weight-space corresponding to function space priors

In [1], to match the priors in function space, the authors optimize a diagonal Gaussian prior in weight space – that is $p(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. However, is it true that this family of prior can be optimized to warp some BNN function space to take on certain properties? Our method for prior construction allows us to determine roughly what the true prior distribution is. Using the hypernetwork we fit neural networks to samples of Gaussian process priors. We then analyze the distribution of those neural network parameters by randomly plotting histograms of individual marginal weights (Figure 4) as well as some joint scatter plots and hexplots of the samples of adjacent and randomly selected weights (Figure 5).

Figure 4 displays the histograms of some marginal weights (randomly selected from the network). All histograms demonstrate non Gaussian behavior: they display heavy tails and skewdness. In the last column they appear to be multimodal. In Figure 5, the joint plots show there is significant covariance between adjacent weights. This means the "True prior" on weights corresponding to some GP prior is likely some fairly complex distribution that we might be able to approximate with some mixture of Gaussian $p(\mathbf{w}) = \sum_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_K)$. This is not a scalable prior for large models. Instead we use a single diagonal Gaussian instead, which we show in section 4.3, can still incorporate meaningful prior information about functions.

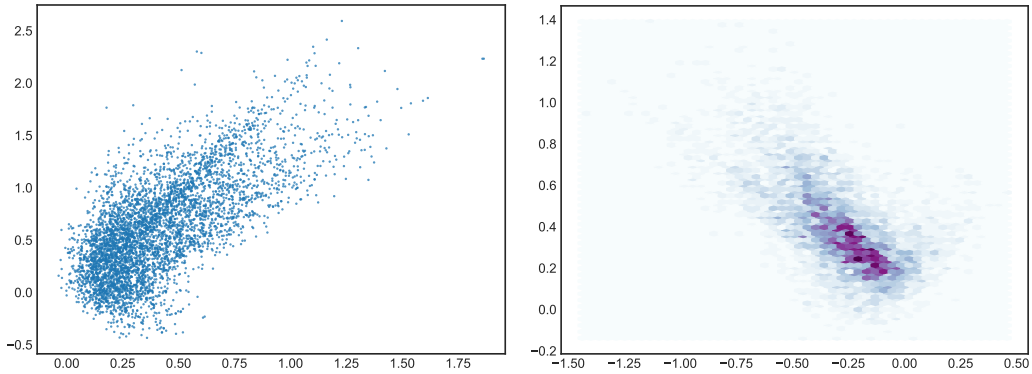


Figure 5: On the left, scatter plot of random selected adjacent weights. On the right, hex plot of random selected adjacent weights

### 4.3 Warping prior function space

We demonstrate our method for prior construction in function space. We plot samples of functions using a standard normal prior, our constructed prior and the prior on functions we are trying to match. To confirm our method, we also plot heatmaps of the covariance in function space for the target distribution, the hypernetwork's recreations, our learned prior and a $\mathcal{N}(\mathbf{0}, \mathbb{I})$ prior.

In row one we demonstrate that we can force a BNN prior on functions defined by a Relu activation function to take on properties similar to functions from a GP prior with a RBF covariance function. We also show that we can force a sigmoid BNN prior on functions to take on periodic properties (row 2) and smoothness properties similar to RBF (row 3). In the last row we force a BNN prior with tanh activations to take on properties similar to the absolute value function.

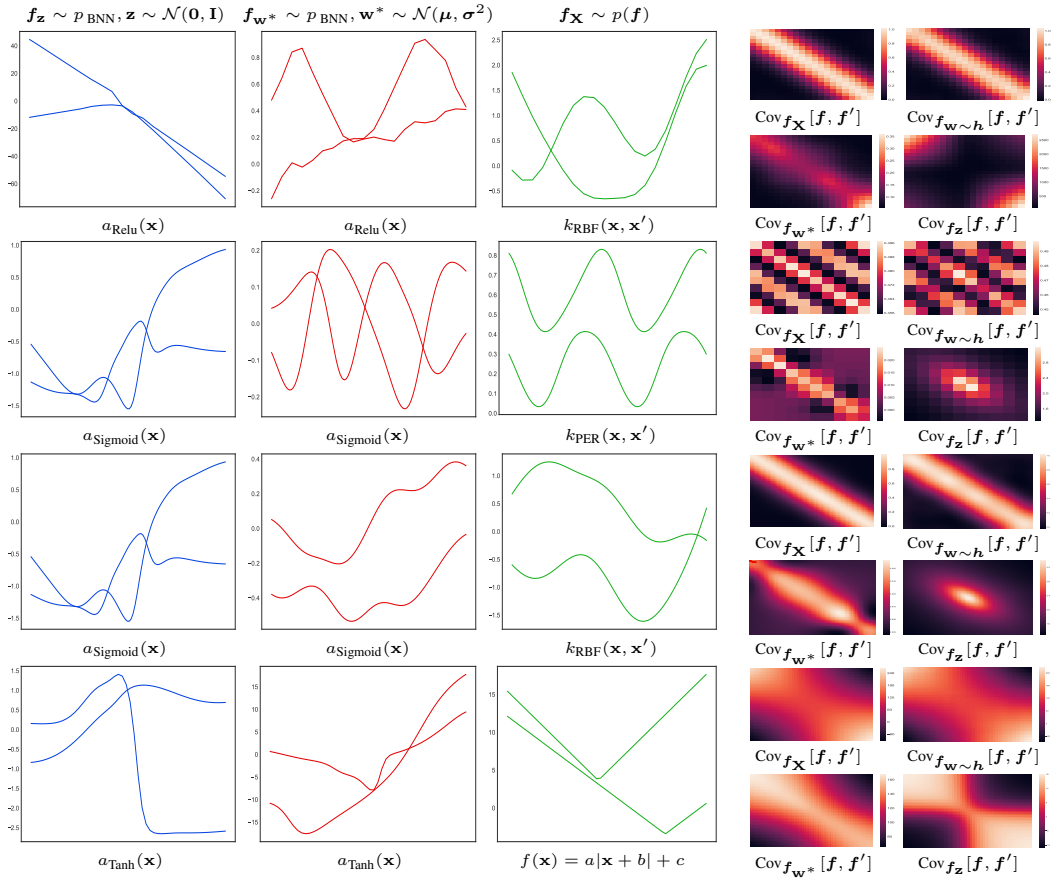
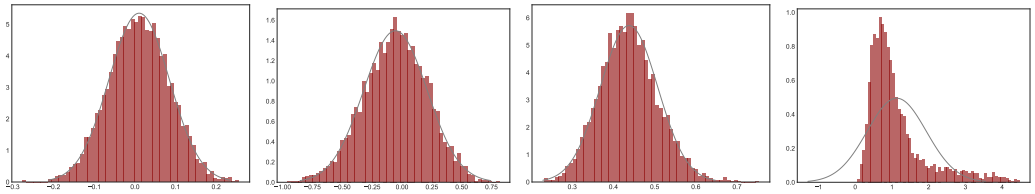



Figure 6: Blue are samples of functions using a standard normal prior on weights $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. Red are samples from the constructed prior. Green are samples from the target distribution on functions. Below each plot is the corresponding activation or covariance function. Except for the implicit distribution of functions in row four, where the family of functions is placed.

Beside each row of samples we also plot 4 covariance heat maps of

- $\text{Cov}_{f_{\mathbf{X}}}[f, f']$ from samples of functions from the target distribution (the top left),
- $\text{Cov}_{f_{\mathbf{w}\sim h}}[f, f']$ from the functions reconstructed using the hypernet : (top right)
- $\text{Cov}_{f_{\mathbf{w}^*}}[f, f']$ from functions sampled using the learned prior (bottom left)
- $\text{Cov}_{f_{\mathbf{z}}}[f, f']$ from functions sampled with the $\mathcal{N}(\mathbf{0}, \mathbb{I})$ prior (bottom right).

We notice that the red optimized samples resemble the target samples much more than the blue samples. Also we confirm, the covariance heat map of the optimized samples, in each row, resembles the target covariance heat map much more than the $N(\mathbf{0}, \mathbb{I})$ heat map.



row 1 row 2 row 3 row 4

Figure 7: Histograms of a single randomly selected marginal function evaluations $p_{\text{BNN}}(\boldsymbol{f}(\mathbf{x}))$ from our learned prior from each row in Figure 6.

#### 4.3.1 Assessing our learned marginals in function space

To check if our constructed prior actually has the desired properties, we plot our priors learned marginal distributions in function space. Choosing a single random function evaluation, we plot histograms of these evaluations from each row in Figure 6. We expect for the first three rows that the histograms are roughly normal as our target $p(\boldsymbol{f})$ is a GP prior. However we do not expect this for row 4, since our target $p(\boldsymbol{f})$ is a distribution of random absolute value functions, which is unlikely to be normal. We observe the exact expected marginal distributions in the histograms in Figure 7.

## 5 Conclusions

In this work, we have developed a simple way to construct priors for BNNs in function space. Our method addresses limitations of prior work on the topic [1]: it avoids approximate inference, can use implicit distributions of functions and allows us to characterize the weight space prior that corresponds to a particular function space prior. This last benefit can be done offline after training, where we can then specifically select our prior distribution. In practice, it makes sense to parameterize our prior with the empirical mean and standard deviation of the weights sampled from the trained hypernetwork.

**Limitations**: our method relies on the use of a hypernetwork and so we cannot specify priors for very large networks. We also cannot use too many inputs into the functions $\boldsymbol{f}(\mathbf{X})$ we feed into the hypernetwork, as this can drastically slow training as well.

**Future work** will evaluate the effectiveness of transmitting properties of our prior to the posterior. Currently using stochastic variational inference, the desired properties are poorly transmitted.

We have demonstrated a new method for constructing function space priors for BNNs that addresses issues in [1] and allows us to investigate the relationship between weight space and function space.

## References


[1] Daniel Flam-Shepherd, James Requiema, and David Duvenaud. Mapping gaussian process priors. *NIPS workshops*, 2017.

[2] Radford Neal. Priors for infinite networks. *In Bayesian Learning for Neural Networks*, 1996.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *In International Conference on Machine Learning*, 2018.

[4] Quoc V. Le David Ha, Andrew Dai. Hypernetworks. *ICLR*, 2017.