Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions

Tim Pearce[1*], **Russell Tsuchida**[2], **Mohamed Zaki**[1], **Alexandra Brintrup**[1], **Andy Neely**[1]
[1] Department of Engineering, University of Cambridge
[2] School of ITEE, University of Queensland
[*] tp424@cam.ac.uk

Abstract

A simple, flexible approach to creating expressive priors in Gaussian process (GP) models makes new kernels from a combination of basic kernels, e.g. summing a periodic and linear kernel can capture seasonal variation with a long term trend. Despite a well-studied link between GPs and Bayesian neural networks (BNNs), the BNN analogue of this has not yet been explored. This paper derives BNN architectures mirroring such kernel combinations. Furthermore, it shows how BNNs can produce periodic kernels, which are often useful in this context. These ideas provide a principled approach to designing BNNs that incorporate prior knowledge about a function. We showcase the practical value of these ideas with illustrative experiments in supervised and reinforcement learning settings. [1]

1 INTRODUCTION

One of deep learning's major achievements was mastering Atari games to human level, with each of the 49 games learnt using an identical algorithm, neural network (NN) architecture, and hyperparameters (Mnih et al., 2015).

Conversely, Gaussian process (GP) modelling places great emphasis on tailoring structure and hyperparameters to individual tasks - four pages of the seminal GP text are dedicated to the careful design of a kernel for a dataset of just 545 datapoints (Rasmussen and Williams, 2006) [p118-122]. Indeed this incorporation of relevant prior knowledge is central to all Bayesian methods.

[1]Code for plots and experiments is available at: https://github.com/TeaPearce

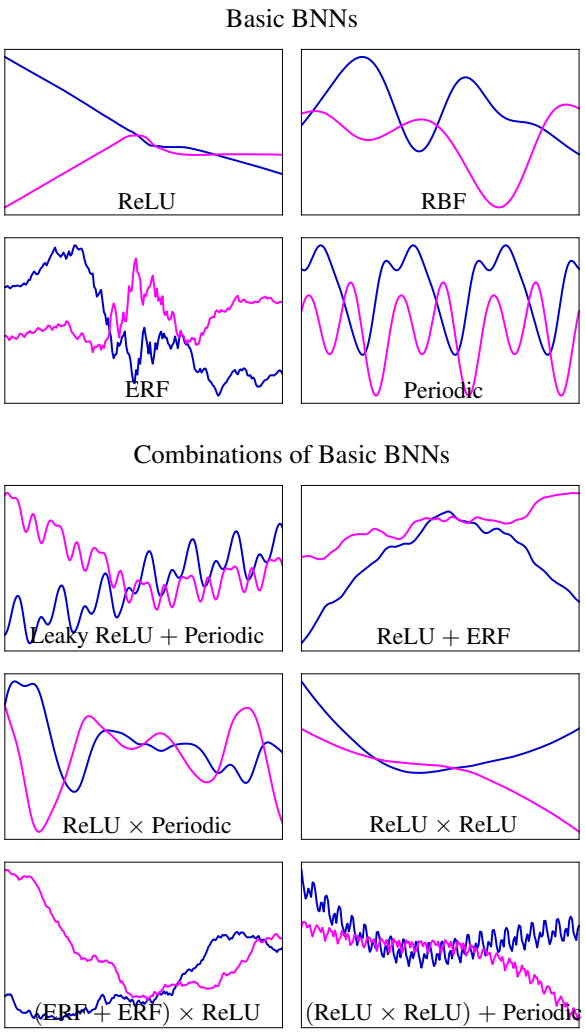

Figure 1: BNN architecture determines our prior belief about a function's properties. In general BNNs provide little flexibility in this regard - modifying only the activation function and length scale ('Basic BNNs'). This paper explores how to design BNNs to produce more expressive prior functions ('Combinations of Basic BNNs'). Two prior draws are shown for each BNN architecture.

Bayesian neural networks (BNNs) lie at the curious intersection between these two modelling philosophies. They have strong theoretical links to GPs (Neal, 1997), yet ultimately share architectures with deep learning models.

The majority of previous research on BNNs has focused on developing methods for efficient inference (Neal, 1997; Hernández-Lobato and Adams, 2015; Blundell et al., 2015), and, more recently, how they can benefit learning frameworks (Gal et al., 2017; Nguyen et al., 2018).

Relatively little work has explored prior design in BNNs - current wisdom takes an architecture expected to work well in non-Bayesian NNs, and places distributions over the weights. This can ignore significant prior information humans bring to tasks (Dubey et al., 2018). The observation that **there seems little point in adopting a Bayesian framework if we don't, and can't, specify effective priors**, forms the core motivation for this paper.

It is well known that BNNs converge to GPs (Neal, 1997). Whilst correspondence is only exact for BNNs of infinite width, this provides a useful lens through which to study the relationship between BNN architecture and prior.

The paper begins with an overview of this connection, discussing priors over functions produced by basic BNNs (which we define as fully-connected feed-forward NNs with iid Gaussian priors over parameters), and the effect of their hyperparameters. Our major contribution then follows in section 3: we consider porting an idea for prior design in GPs to BNNs. A simple approach to building expressive priors in GPs is to combine basic kernels to form a new kernel. We derive BNN architectures mirroring these effects. Figure 1 shows examples of priors that can be expressed by basic BNNs, followed by the richer class of priors that can be expressed using these combined BNN architectures.

One situation where kernel combinations are useful is for functions with imperfect periodicity. This property is easily captured by combining a periodic kernel with some other kernel. We explore periodicity in BNNs in section 4, showing it is not enough to simply use cosine activations, as might be expected. We develop an alternative approach that precisely recovers a popular GP periodic kernel.

Illustrative experiments in section 5 showcase the practical value of our theoretical results both in supervised time series prediction and in reinforcement learning (RL) on a classic control task.

This paper is important from three perspectives.

1. As a theoretical result further linking GPs with BNNs.

2. As a practical approach to creating more expressive priors in Bayesian deep learning models.

3. For non-Bayesian deep learning, enabling proper model specification for periodic and locally periodic functions.

2 BACKGROUND

2.1 GAUSSIAN PROCESSES

A GP is a stochastic process, fully described by its mean function, $\mathbb{E}[f(\mathbf{x})]$, and covariance function (or 'kernel'), $K(\mathbf{x}, \mathbf{x}')$. Any finite subset of a GP's realisations follows a multivariate Gaussian distribution, which makes many analytical computations possible. They are considered a Bayesian non-parametic model in machine learning - see Rasmussen and Williams (2006) for a full introduction. Duvenaud (2014) provides a reference for the below.

A GP's mean function is often assumed zero, as it will be throughout this work. The kernel then determines the generalisation properties of the model. Informally, a kernel is a function that describes how closely two arbitrary data points, $\mathbf{x}$ & $\mathbf{x}'$, are related. One might expect that if data points are similar, their outputs should also be similar. A common choice of kernel, squared exponential (SE), captures such behaviour,

$$K_{SE}(x, x') = \sigma^2 \exp\left(- \frac{||\mathbf{x} - \mathbf{x}'||_2^2}{l^2} \right), \qquad (1)$$

where, l, the length scale, provides some control over how quickly similarity fades, and σ^2 is a scaling parameter.

The behaviour implied by the SE kernel is not suitable for all datasets. Data generated by a periodic function, for instance, would not follow this simple similarity rule. Here a periodic kernel is appropriate, e.g. the exponential sine squared kernel (ESS), for the 1-D case is,

$$K_{ESS}(x, x') = \sigma^2 \exp\left(- \frac{2 \sin^2(\frac{\pi}{p}(x - x'))}{l^2} \right) \qquad (2)$$

where, p, determines the period over which the function repeats.

Many kernels are available, and selection of one that encodes properties of the function being modelled can be critical for good performance. By choosing kernels well suited to a problem, we are specifying appropriate priors.

What if a dataset has properties not well described by any of these kernels? A simple solution is to combine basic kernels together to make a new kernel. One can be surprisingly flexible in how this is done - directly multiplying or adding kernels, or applying warping to inputs (Steinwart and Christmann, 2008) [4.1]. This vastly increases the expressiveness of possible priors.

Using the kernels from above for illustration, in order to model the function, $f(x) = \sin(x)+x$, one might choose, $K = K_{ESS} + K_{SE}$. For the function, $f(x) = x\sin(x)$, a good choice might be, $K = K_{ESS} \times K_{SE}$. In section 5 we model two time series with similar properties.

2.2 BNNs CONVERGE TO GPs

Here we reproduce the derivation of infinitely wide single-layer BNNs as GPs (Williams, 1996; Neal, 1997).

Consider a single-layer NN, $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, with input, $\mathbf{x}$, weights, $\mathbf{w}_1$ & $\mathbf{w}_2$, biases $\mathbf{b}_1$, activation function, ψ, and hidden units H, with no final bias (to unclutter analysis),

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi(\mathbf{w}_{1i}\mathbf{x} + b_{1i}). \tag{3}$$

If priors centered at zero are placed over the parameters, we have a BNN with, $\mathbb{E}[f(\mathbf{x})] = 0$, hence the mean function is zero. Consider now the covariance of outputs corresponding to two arbitrary inputs, $\mathbf{x}$ & $\mathbf{x}'$. Denoting for convenience $\psi_i(\mathbf{x}) := \psi(\mathbf{w}_{1i}\mathbf{x} + b_{1i})$,

$$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] \tag{4}$$

$$= \mathbb{E}\Big[\big(\sum_{i=1}^{H} w_{2i}\psi_i(\mathbf{x})\big)\big(\sum_{j=1}^{H} w_{2j}\psi_j(\mathbf{x}')\big)\Big] \tag{5}$$

$$\begin{aligned}= \mathbb{E}\big[&w_{2,1}\psi_1(\mathbf{x})w_{2,1}\psi_1(\mathbf{x}') + w_{2,1}\psi_1(\mathbf{x})w_{2,2}\psi_2(\mathbf{x}') + \cdots \\ &w_{2,2}\psi_2(\mathbf{x})w_{2,1}\psi_1(\mathbf{x}') + w_{2,2}\psi_2(\mathbf{x})w_{2,1}\psi_1(\mathbf{x}') + \cdots \\ &\cdots + w_{2,H}\psi_H(\mathbf{x})w_{2,H}\psi_H(\mathbf{x}')\big]\end{aligned} \tag{6}$$

if parameter priors are independent, we find the terms between separate hidden units are zero, e.g. $\mathbb{E}[w_{2,1}\psi_1(\mathbf{x})w_{2,2}\psi_2(\mathbf{x}')] = \mathbb{E}[w_{2,1}]\mathbb{E}[\psi_1(\mathbf{x})]\mathbb{E}[w_{2,2}]\mathbb{E}[\psi_2(\mathbf{x}')] = 0$, so,

$$\begin{aligned}= \mathbb{E}\big[&w_{2,1}\psi_1(\mathbf{x})w_{2,1}\psi_1(\mathbf{x}') + w_{2,2}\psi_2(\mathbf{x})w_{2,2}\psi_2(\mathbf{x}') + \cdots \\ &\cdots + w_{2,H}\psi_H(\mathbf{x})w_{2,H}\psi_H(\mathbf{x}')\big]\end{aligned} \tag{7}$$

and if priors are identically distributed,

$$= H\mathbb{E}\big[w_2\psi(\mathbf{x})w_2\psi(\mathbf{x}')\big] \tag{8}$$

$$= \sigma_{w2}^2\mathbb{E}\big[\psi(\mathbf{x})\psi(\mathbf{x}')\big] \tag{9}$$

where w_2 prior variance is scaled by width, $1/H$.

Having derived expressions for mean and covariance, it remains to show that the distribution is Gaussian. Eq. 3 is a sum of iid random variables, hence, under mild conditions, the CLT states that the distribution over functions is normally distributed as $H \to \infty$.

2.3 ANALYTICAL BNN KERNELS

To derive analytical kernels for specific activations, ψ, and priors, $p(\mathbf{w}_1)$ & $p(b_1)$, eq. 9 must be evaluated.

$$K(\mathbf{x}, \mathbf{x}') = \sigma_{w2}^2 \iint \psi(\mathbf{x})\psi(\mathbf{x}')p(\mathbf{w}_1)p(b_1)d\mathbf{w}_1 db_1 \tag{10}$$

The integral is generally not trivial, and several papers have focused on deriving analytical forms for popular activation functions, usually with normally distributed priors - ERF/probit (sigmoidal shape) and RBF (Williams, 1996), step function and ReLU (Cho and Saul, 2009), Leaky ReLU (Tsuchida et al., 2018). In section 4 we add to this list by considering cosine activations. Similar results have been shown for convolutional BNNs (Novak et al., 2019).

Naturally eq. 10 can be computed numerically where analytical forms do not exist. Recurrent computation is necessary for deeper BNNs, which also converge to GPs (Lee et al., 2018).

2.4 HYPERPARAMETER INTUITION

Having shown a correspondence between GPs and BNNs, we now provide, in intuitive terms, the effect of key BNN hyperparameters on GP priors, which is useful when modelling with BNNs - care should then be taken to select hyperparameters that suit properties of the function being modelled. We assume Gaussian priors on weights and biases, see Nalisnick (2018) for an investigation of other prior distributions.

- **Activation function** - Swapping activations effectively swaps the parametric form of kernel. Basic BNNs in figure 1 show example prior draws for single-layer BNNs with ReLU and ERF activations, as well as an RBF BNN.

- **Prior variances** - These have different effects depending on the layer. Roughly speaking, variance of first layer weights and biases controls how wiggly the priors are (similar effect to length scale in the SE kernel). Final layer weight variance simply scales the output range of priors (similar effect to σ^2 in the SE kernel).

- **Data noise variance** - A level of data noise variance (irreducible noise) must be specified to create a valid likelihood function when implementing BNNs. Normally distributed homoskedastic data noise is often assumed. Roughly speaking, data noise variance determines how perfectly the data should be fitted.

3 KERNEL COMBINATIONS IN BNNs

This section considers how to design BNN architectures such that, in the infinite width limit, they give rise to the equivalent GP kernel combinations.

The kernel combination operations we consider are;

- **Addition**: $K(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}') + K_B(\mathbf{x}, \mathbf{x}')$
- **Multiplication**: $K(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}')K_B(\mathbf{x}, \mathbf{x}')$
- **Polynomial**: e.g. $K(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}')^2$
- **Warping**: $K(\mathbf{x}, \mathbf{x}') = K_A(u(\mathbf{x}), u(\mathbf{x}'))$ for a function, $u : R^d \to R^m$

We begin by considering architectures that combine the output of two BNNs. This turns out to be a valid way to add kernels, but not to multiply kernels. We then consider architectures that combine BNNs, point wise, at the final hidden layer. This is valid for multiplicative kernels, but produces a small artefact for additive kernels.

Having derived architectures mirroring additive and multiplicative kernels, section 3.3 examines using these in more advanced ways.

3.1 COMBINING BNNs AT OUTPUT

A straightforward way to combine BNNs is to consider some operation combining their outputs.

3.1.1 Additive

Consider two independent GPs denoted $f_A(\mathbf{x})$ & $f_B(\mathbf{x})$, summed,

$$f_{add}(\mathbf{x}) = f_A(\mathbf{x}) + f_B(\mathbf{x}). \tag{11}$$

In general, it is known that $f_{add}(\mathbf{x})$ will also be a GP with kernel, $K_{add}(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}') + K_B(\mathbf{x}, \mathbf{x}')$, (Saul et al., 2016).

For two single-layer BNNs, this is recovered by a BNN of architecture,

$$= \sum_{i=1}^{H} w_{A2i}\psi_{Ai}(\mathbf{x}) + \sum_{j=1}^{H} w_{B2j}\psi_{Bj}(\mathbf{x}). \tag{12}$$

Since this converges to the sum of two independent GPs, regardless of depth (section 2.2), the general GP result applies, and suffices to show that independent BNNs (of infinite width) summed at outputs reproduce a GP with additive kernel.

3.1.2 Multiplicative

Two GPs multiplied together,

$$f_{mult}(\mathbf{x}) = f_A(\mathbf{x})f_B(\mathbf{x}), \tag{13}$$

do *not* generally produce a GP (Rasmussen and Williams, 2006) [4.2.4], even though there does exist a GP with kernel, $K_{mult}(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}')K_B(\mathbf{x}, \mathbf{x}')$. (Analogously, the product of two normally distributed random variables is not normally distributed.)

This means that independent BNNs multipled at outputs (shown for the single layer case),

$$= \sum_{i=1}^{H} w_{A2i}\psi_{Ai}(\mathbf{x}) \sum_{j=1}^{H} w_{B2j}\psi_{Bj}(\mathbf{x}) \tag{14}$$

do *not* produce a GP with multiplied kernel.

3.2 COMBINING BNNs AT HIDDEN LAYERS

Consider now combining BNNs by point wise operations at their hidden layers.

3.2.1 Additive

Taking two single-layer BNNs, the additive case is,

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\big(\psi_{Ai}(\mathbf{x}) + \psi_{Bi}(\mathbf{x})\big), \tag{15}$$

where ψ_A and ψ_B are hidden units for each sub-BNN. As in section 3.1, neither hyperparameters nor activation function need be shared, e.g. one could take a RBF and ReLU BNN, $\psi_A(\mathbf{x}) = \exp(-||\mathbf{x} - \mathbf{w}_{A1}^T||_2^2/\sigma_g^2)$, and, $\psi_B(\mathbf{x}) = \max(\mathbf{w}_{B1}\mathbf{x} + b_{B1}, 0)$. We now derive the equivalent GP for such an architecture.

Analysis precisely as in section 2.2 can be followed up to eq. 9, leaving,

$$K_{add}(\mathbf{x}, \mathbf{x}') = \sigma_{w2}^2\mathbb{E}\big[\big(\psi_A(\mathbf{x}) + \psi_B(\mathbf{x})\big)\big(\psi_A(\mathbf{x}') + \psi_B(\mathbf{x}')\big)\big] \tag{16}$$

$$= \sigma_{w2}^2\mathbb{E}\big[\psi_A(\mathbf{x})\psi_A(\mathbf{x}') + \psi_A(\mathbf{x})\psi_B(\mathbf{x}') + \psi_B(\mathbf{x})\psi_A(\mathbf{x}') + \psi_B(\mathbf{x})\psi_B(\mathbf{x}')\big] \tag{17}$$

by linearity of expectation, and noting ψ_A and ψ_B are independent,

$$= \sigma_{w2}^2\mathbb{E}\big[\psi_A(\mathbf{x})\psi_A(\mathbf{x}')\big] + \sigma_{w2}^2\mathbb{E}\big[\psi_B(\mathbf{x})\psi_B(\mathbf{x}')\big] + \sigma_{w2}^2\mathbb{E}\big[\psi_A(\mathbf{x})\big]\mathbb{E}\big[\psi_B(\mathbf{x}')\big] + \sigma_{w2}^2\mathbb{E}\big[\psi_A(\mathbf{x}')\big]\mathbb{E}\big[\psi_B(\mathbf{x})\big] \tag{18}$$

$$= K_A(\mathbf{x}, \mathbf{x}') + K_B(\mathbf{x}, \mathbf{x}') + \sigma_{w2}^2 \mathbb{E}\big[\psi_A(\mathbf{x})\big]\mathbb{E}\big[\psi_B(\mathbf{x}')\big] + \sigma_{w2}^2 \mathbb{E}\big[\psi_A(\mathbf{x}')\big]\mathbb{E}\big[\psi_B(\mathbf{x})\big]. \tag{19}$$

This is the additive kernel plus two extra terms. The impact of these extra terms depends on the activation function, and could be compensated for. If either ψ is an odd function, $\mathbb{E}\big[\psi_{odd}(\cdot)\big] = 0$, the additive kernel is exactly recovered. Alternatively, if both ψ's are sigmoids, $\mathbb{E}\big[\psi_{sig}(\cdot)\big] = 0.5$, which results in the kernel, $K_A(\mathbf{x}, \mathbf{x}') + K_B(\mathbf{x}, \mathbf{x}') + c$, for some constant c. If ψ is a ReLU, $\mathbb{E}\big[\psi_{ReLU}(\cdot)\big]$ is input dependent, making compensation trickier (though still possible).

In general, summing point wise after hidden nodes is not a valid way to reproduce an additive GP kernel, although effects of the artefact terms could be compensated for.

3.2.2 Multiplicative

Following the same procedure for multiplication after hidden nodes,

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\big(\psi_{Ai}(\mathbf{x})\psi_{Bi}(\mathbf{x})\big) \tag{20}$$

$$K_{mult}(\mathbf{x}, \mathbf{x}') = \sigma_{w2}^2 \mathbb{E}\big[\big(\psi_A(\mathbf{x})\psi_B(\mathbf{x})\big)\big(\psi_A(\mathbf{x}')\psi_B(\mathbf{x}')\big)\big] \tag{21}$$

BNN independence allows the rearrangement,

$$= \sigma_{w2}^2 \mathbb{E}\big[\psi_A(\mathbf{x})\psi_A(\mathbf{x}')\big]\mathbb{E}\big[\psi_B(\mathbf{x})\psi_B(\mathbf{x}')\big] \tag{22}$$

$$= K_A(\mathbf{x}, \mathbf{x}')K_B(\mathbf{x}, \mathbf{x}') \tag{23}$$

and hence multiplying point wise after hidden nodes is a valid way to reproduce a multiplicative GP kernel.

3.3 EXTENSIONS

Whilst the previous results were explicitly shown for two single-layer BNNs, it is straightforward to extend them to a variety of situations. Following, we provide examples of useful constructions.

Additive and Multiplicative

Kernel:

$$K(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}') + K_B(\mathbf{x}, \mathbf{x}')K_C(\mathbf{x}, \mathbf{x}')K_D(\mathbf{x}, \mathbf{x}') \tag{24}$$

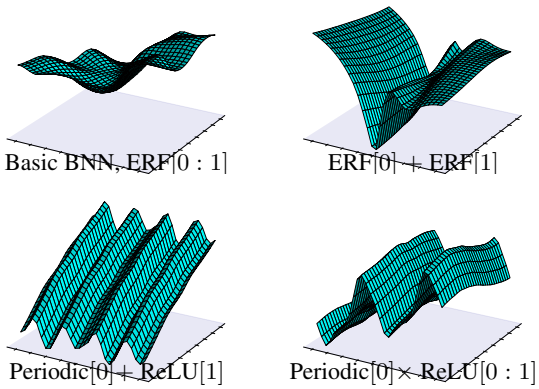

Figure 2: Prior draws for a 2-D input. Square brackets designate which dimension(s) each kernel is applied to.

BNN architecture:

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi_{Ai}(\mathbf{x}) + \sum_{j=1}^{H} w_{2j}\psi_{Bj}(\mathbf{x})\psi_{Cj}(\mathbf{x})\psi_{Dj}(\mathbf{x}) \tag{25}$$

Polynomials

Kernel:

$$K(\mathbf{x}, \mathbf{x}') = K_A(\mathbf{x}, \mathbf{x}')^2 \tag{26}$$

BNN architecture:

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi_{A1i}(\mathbf{x})\psi_{A2i}(\mathbf{x}) \tag{27}$$

Where ψ_{A1} and ψ_{A2} are separate nodes sharing common hyperparameters.

Warping

Kernel:

$$K(\mathbf{x}, \mathbf{x}') = K(u(\mathbf{x}), u(\mathbf{x}')) \tag{28}$$

BNN architecture:

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi_i(u(\mathbf{x})) \tag{29}$$

Separation of Inputs

It can be useful to consider multiple kernels taking subsets of inputs, combined through either addition or multiplication (Duvenaud, 2014) [2.3, 2.4], as visualised in figure 2.

Kernel:

$$K(\mathbf{x}, \mathbf{x}') = K_A(x_1, x_1') + K_B(x_2, x_2') \tag{30}$$

BNN architecture:

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi_{Ai}(x_1) + \sum_{j=1}^{H} w_{2j}\psi_{Bj}(x_2) \tag{31}$$

Kernel:

$$K(\mathbf{x}, \mathbf{x}') = K_A(x_1, x_1')K_B(x_2, x_2'). \tag{32}$$

BNN architecture:

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\psi_{Ai}(x_1)\psi_{Bi}(x_2). \tag{33}$$

Deeper BNNs

Out of convenience, constructions have been shown for single-layer BNNs. These could be replaced by deep BNNs, which equally correspond to GP kernels (section 2.3).

4 PERIODIC BNN KERNELS

This section considers how BNNs can be designed to model periodic functions. To our knowledge this analysis is entirely novel. We define a periodic function as, $f(x) = f(x + p)$, for some scalar period, $p \in \mathbb{R}_+$.

We find that cosine activations do not produce a periodic kernel, but applying warping to inputs followed by standard activations functions, does.

4.1 COSINE ACTIVATIONS

Consider a single-layer BNN with cosine activation functions; intuition might suggest this leads to a periodic kernel. (Note such activations have been explored in other contexts (Parascandolo et al., 2017; Ramachandran et al., 2017).)

$$f(\mathbf{x}) = \sum_{i=1}^{H} w_{2i}\cos(\mathbf{w}_{1i}\mathbf{x} + b_{1i}) \tag{34}$$

Following the usual GP kernel derivation in section 2.2,

$$K_{cos}(\mathbf{x}, \mathbf{x}') = \sigma_{w2}^2 \iint \cos(\mathbf{w}_1\mathbf{x} + b_1)\cos(\mathbf{w}_1\mathbf{x}' + b_1)p(\mathbf{w}_1)p(b_1)d\mathbf{w}_1 db_1 \tag{35}$$

Assuming priors, $p(\mathbf{w}_1) \sim \mathcal{N}(0, \sigma_{w1}^2 I)$, and, $p(b_1) \sim \mathcal{N}(0, \sigma_{b1}^2)$, we find,[2]

$$= \frac{\sigma_{w2}^2}{2}\left(\exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||_2^2}{2/\sigma_{w1}^2}\right) + \exp\left(-\frac{||\mathbf{x} + \mathbf{x}'||_2^2}{2/\sigma_{w1}^2} + 2\sigma_{b1}^2\right)\right). \tag{36}$$

[2]Rewrite $\cos(A)\cos(B) = \frac{1}{2}[\cos(A - B) + \cos(A + B)]$, then use, $\mathbb{E}[\cos(\mathbf{xw})] = \exp(-\frac{1}{2}\mathbf{x}^T\Sigma\mathbf{x})$, if $\mathbf{w} \sim \mathcal{N}(0, \Sigma)$.

Slightly counter-intuitively, the kernel is not periodic. Rather it is the sum of the SE kernel (eq. 1), and another term.

We further considered using Laplace and uniform distributions for priors, which did result in kernels containing trigonometric functions, but the forms were untidy and not apparently useful.

Note that our analysis is from the perspective of equivalent GP kernels. It is possible to consider narrow BNNs with cosine activations that *would* produce periodic predictive distributions. If initialised suitably, these may be of some use.

4.2 INPUT WARPING

Whilst modifying the activation function failed to produce periodic kernels, applying a warping to inputs was more successful.

The most common periodic kernel used in GP modelling is the ESS kernel (Duvenaud, 2014) [p. 25], as shown in eq. 2. Having established its value in periodic modelling, we wanted to reproduce this as closely as possible with a BNN. Surprisingly, an exact recovery is possible as follows.

Apply a warping to a 1-D input, $x \rightarrow (\cos(\frac{2\pi x}{p}), \sin(\frac{2\pi x}{p}))$, followed by a single-layer RBF BNN taking this 2-D warping as input.

In general, an infinitely wide single-layer RBF BNN produces the following GP kernel (Williams, 1996),

$$K_{RBF_{BNN}}(\mathbf{x}, \mathbf{x}') = \left(\frac{\sigma_e}{\sigma_u}\right)^d \exp\left(-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma_m^2}\right) \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma_s^2}\right) \exp\left(-\frac{\mathbf{x}'^T\mathbf{x}'}{2\sigma_m^2}\right) \tag{37}$$

where, $1/\sigma_e^2 = 2/\sigma_g^2 + 1/\sigma_u^2$, $\sigma_s^2 = 2\sigma_g^2 + \sigma_g^4/\sigma_u^2$, and $\sigma_m^2 = 2\sigma_u^2 + \sigma_g^2$. If the discussed warping is first applied, for the 1-D case this becomes,

$$K_{RBFPer_{BNN}}(\mathbf{x}, \mathbf{x}') = \left(\frac{\sigma_e}{\sigma_u}\right)^2 \exp\left(-\frac{\cos^2(\frac{2\pi x}{p}) + \sin^2(\frac{2\pi x}{p})}{2\sigma_m^2}\right) \exp\left(-\frac{\left(\cos(\frac{2\pi x}{p}) - \cos(\frac{2\pi x'}{p})\right)^2}{2\sigma_s^2} + \frac{\left(\sin(\frac{2\pi x}{p}) - \sin(\frac{2\pi x'}{p})\right)^2}{2\sigma_s^2}\right) \exp\left(-\frac{\cos^2(\frac{2\pi x'}{p}) + \sin^2(\frac{2\pi x'}{p})}{2\sigma_m^2}\right). \tag{38}$$

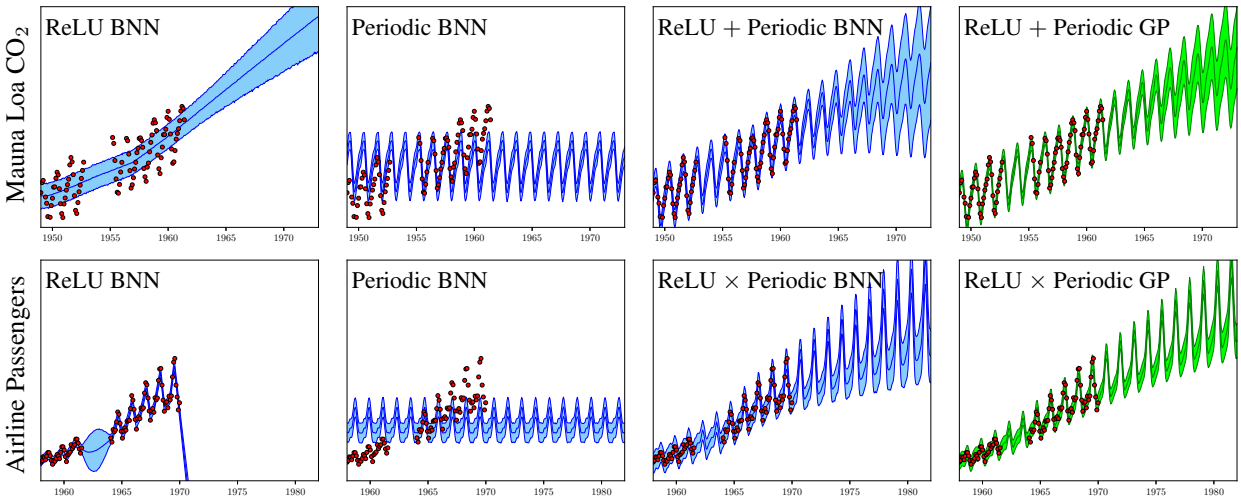

Figure 3: Two time series with seasonal fluctuations and long term trends. ReLU and periodic BNNs are separately unable to capture these patterns (first and second from left). However, they succeed when combined in BNN architectures as proposed in this paper (third from left), closely approximating the predictive distributions of exact GP inference with the equivalent kernel combinations (right).

Noting, $\left(\cos(\frac{2\pi x}{p}) - \cos(\frac{2\pi x'}{p})\right)^2 + \left(\sin(\frac{2\pi x}{p}) - \sin(\frac{2\pi x'}{p})\right)^2 = 4\sin^2(\frac{\pi}{p}(x - x'))$, and also, $\cos^2(\cdot) + \sin^2(\cdot) = 1$, this reduces to,

$$= \left(\frac{\sigma_e}{\sigma_u}\right)^2 \exp\left(-\frac{1}{\sigma_m^2}\right) \exp\left(-\frac{2\sin^2(\frac{\pi}{p}(x - x'))}{\sigma_s^2}\right) \tag{39}$$

which is of the same form as the periodic ESS kernel. Indeed there is a connection to the derivation of the ESS kernel, which used the same warping followed by the SE kernel (MacKay, 1998).

It is equally plausible to apply the same warping followed by BNNs of other architectures. For example, the single-layer ReLU case results in,

$$K_{ReLUPer}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_{w_2}^2}{\pi}(\sin\omega + (\pi - \omega)\cos\omega) \tag{40}$$

where,

$$\omega = \cos^{-1}\left(\frac{\sigma_{b_1}^2 + \sigma_{w_1}^2\cos(\frac{2\pi}{p}(x - x'))}{\sigma_{b_1}^2 + \sigma_{w_1}^2}\right). \tag{41}$$

This is equally suited to periodic modelling, and perhaps more convenient in BNNs given the prevalence of ReLUs.

5 ILLUSTRATIVE EXPERIMENTS

This section provides examples of where, all things being equal, BNNs designed to incorporate suitable prior knowledge can deliver a performance boost over basic BNNs. These gains should be independent of learning algorithm or inference method, but are necessarily task specific. Hence, experiments are framed as illustrative rather than exhaustive.

We showcase situations benefiting simultaneously from both of the ideas introduced in this paper - combinations of BNNs *and* periodic function modelling, although either can also be used separately.

All experiments used BNN widths of 50 hidden nodes. Their success supports our claim that, despite the theory presented in this paper being exact only for infinite-width BNNs, it provides sound principles for building expressive BNN models of finite width.

5.1 SUPERVISED LEARNING: TIME SERIES

Time series data often have seasonal fluctuations combined with longer term trends. These experiments show that where basic BNNs struggle to capture such patterns, simple combinations of these basic BNNs produce appropriate priors.

We considered two prediction tasks: CO_2 levels recorded at a Hawaiian volcano (Mauna), and numbers of airline passengers flying internationally (Airline). For both datasets we used ten years of monthly recordings, then deleted data between years 3-5 to create a gap in the series. Below, we qualitatively assess the predictive distribution in both the interpolation region (3-5 years) and an extrapolation region (10-20 years).

In Mauna, seasonal variations appear to be of constant amplitude, suggesting an additive relationship between trend

and period, whilst Airline shows increasing amplitudes, suggesting a multiplicative relationship.

Figure 3 shows the two datasets and the predictive distributions produced by four types of model (shading gives ± 3 standard deviations). Inference was performed with HMC for BNNs (Neal, 1997), and analytically for GP.

1. **ReLU BNN** - single-layer BNN with ReLU activations. There are two possibilities with this model - a long length scale, as shown for Mauna, which captures the long term trend but does not fit the seasonal variations. Alternatively, a short length scale allows better fitting of the training data, but at the expense of extrapolations - in Airline this produces a nonsensical 10-20 year forecast.

2. **Periodic BNN** - single-layer BNN with cos/sin warping applied, followed by RBF activations. This is the structure derived earlier, with equivalent kernel in eq. 39. Since these BNNs output pure periodic functions they are unable to fit the data well.

3. **Combined BNN** - these models combined the ReLU & Periodic BNNs from 1. and 2. above. For Mauna, an addition operation at outputs was applied, whilst for Airline, hidden nodes were multiplied point wise. Note that the main characteristics of the datasets are captured. **This creates sensible interpolation and extrapolation predictions.** Importantly, uncertainty increases with the time horizon.

4. **Combined GP** - the GPs corresponding to the combined BNNs in 3. were implemented. This enables verification that the BNN architectures produce a predictive distribution corresponding to the GP's (which could be thought of as the 'ground truth'). The slight differences could be put down to the finite width of the BNNs, and imperfect inference procedure.

5.2 REINFORCEMENT LEARNING: PENDULUM SWING UP

We considered the pendulum swing up task; an agent applies torque to a bar on a pivot, maximising rewards by controlling the bar to be vertically upright. Observations consist of angle, θ, and angular velocity, $\dot{\theta}$.

We used a slightly modified version of the task. Actions were discretised so that torque $\in \{-1, 0, +1\}$. Dynamics were also modified - usually the update rule for θ is,

$$\theta_t = \theta_{t-1} + \dot{\theta}_t dt, \tag{42}$$

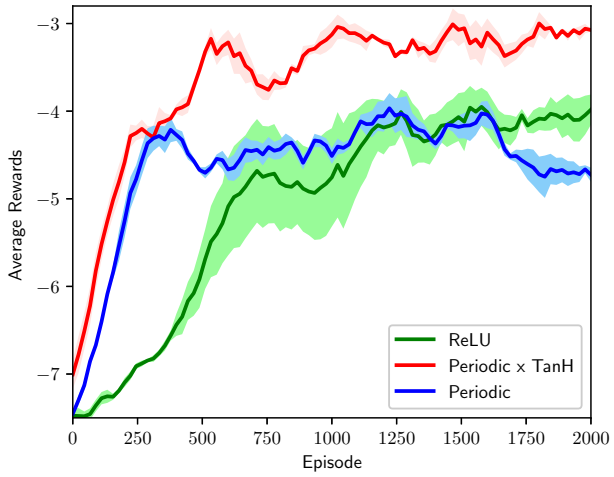

Figure 4: Learning for three different BNN architectures on the pendulum task. The BNN incorporating a suitable prior for task, 'Periodic×TanH', outperforms basic BNNs. Mean $\pm$ 1 standard error, three runs.

where t is time, and $\dot{\theta}$ is a function of the applied torque and gravity. We modified this to,

$$\theta_t = \theta_{t-1} + \frac{2}{1 - e^{-\theta_{t-1}/3}} \dot{\theta}_t dt. \tag{43}$$

This effectively introduces a frictional force that varies according to the absolute value of θ. Crucially this means that as the bar spins, slightly different dynamics are experienced - this could arise from the bar spinning along a thread.

A priori, we therefore know that the function is locally periodic. This makes the task challenging for basic BNN architectures - enforcing exact periodicity is just as inappropriate as ignoring it entirely.

We tested three BNN architectures on the task.

1. **ReLU**: a two-layer ReLU BNN with raw angle, θ, and angular velocity, $\dot{\theta}$, as input. Priors: $\sigma_{w1}^2 = \sigma_{b1}^2 = 1$, $\sigma_{w2}^2 = \sigma_{b2}^2 = 1/50$, $\sigma_{w2}^2 = \sigma_{b2}^2 = 10.0$.

2. **Periodic**: cos/sin input warping applied to θ, raw angular velocity, $\dot{\theta}$, followed by a two-layer ReLU BNN. Prior variances as for 1.

3. **Periodic×TanH**: takes the Periodic BNN as in 2., multiplied by a single-layer TanH BNN (taking only θ as input) with long length scale, $\sigma_{w1}^2 = \sigma_{b1}^2 = 0.2$. This combines multiplication, warping and separation of inputs from section 3.

Note that the benefits of BNN architecture should be independent of the learning algorithm and inference method.

Here we used Bayesian Q-learning (Dearden et al., 1998), similar to regular Q-learning, but with Q-values modelled as *distributions* rather than point estimates, with BNNs as the function approximators.

It was important that a scalable technique be used for inference. Q-learning is sample inefficient, and the experience buffer accumulates hundreds of thousands of data points ($2,000$ episodes $\times$ 200 time steps). Both GPs and HMC struggle with data of this size. We used Bayesian ensembles (Pearce et al., 2018, 2019) for inference - a recently proposed scalable, easily implementable technique.

Figure 4 shows cumulative rewards for the three different architectures over 2,000 episodes. Periodic$\times$TanH clearly outperforms other models, both in terms of learning speed and quality of final policy. This is an example of the *blessing of abstraction* at work - the more structure we account for, the less data we need (Duvenaud, 2014) [p13]. The Periodic BNN has similar learning speed early on, but plateaus since it does not have the flexibility to fully capture system dynamics. ReLU, meanwhile, learns slowly, but has enough flexibility to capture closer to the true dynamics, and eventually surpasses the Periodic BNN.

Figure 5 provides evidence for these comments. It shows the dynamics learnt for three revolutions of the pendulum for each BNN. The Periodic and ReLU BNNs are only able to approximate the optimum dynamics found by Periodic$\times$TanH.

6 RELATED WORK

Two recent works proposed methods to overcome the limited expressivity of BNN priors. Flam-Shepherd et al. (2017) trained a BNN to output GP priors before running inference on a task. Sun et al. (2019) had a similar approach that did not require pretraining.

Both methods operate roughly in a supervised learning fashion, training BNNs to match the output of some GP, on training data augmented with sampled data points. In contrast, our approach directly incorporates priors into the model structure.

Several other works are of relevance. Ma et al. (2019) propose variational implicit processes for BNNs. Gaier and Ha (2019) could be interpreted as fixing a posterior over parameters, and using evolutionary search to find a BNN architecture producing suitable posterior functions.

An orthogonal line of work to ours considers how to improve the scalability of GPs over the default $\mathcal{O}(N^3)$, e.g. (Snelson and Ghahramani, 2006). There also exist other techniques for creating expressive priors in GPs, e.g. (Wilson and Adams, 2013).

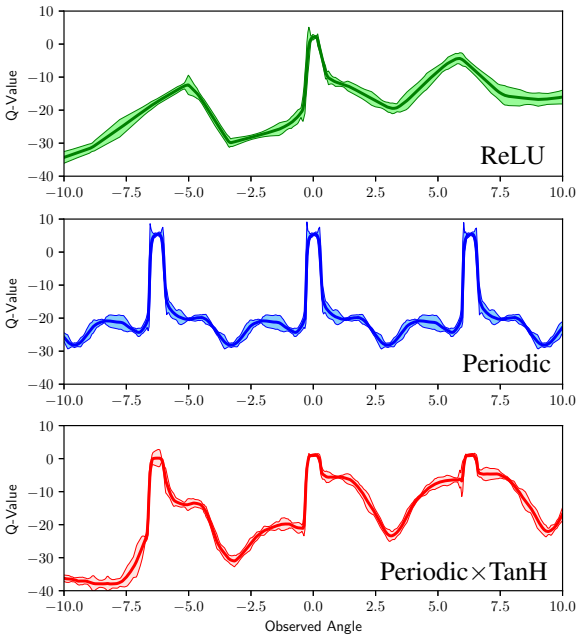

Figure 5: Q-values learnt for the action, torque $= 0$, conditioned on observations of $\dot{\theta} = 0$, and input angle, θ, varied on x-axis. Periodic$\times$TanH captures the local periodicity of the function.

7 CONCLUSION

Expressive priors can be created in GPs by combining basic kernels into a new kernel. Noting the equivalence between GPs and infinitely-wide BNNs, this paper ported the idea to BNNs, deriving architectures that mirror such kernel combinations. Furthermore, we advanced the modelling of periodic functions with BNNs, which are often useful in this context.

These ideas are of practical benefit when some property is known about a function a priori, and basic BNNs do not model this well. We showcased two scenarios for which this was the case; time series modelling, and a RL task involving a locally periodic function.

In many learning tasks, a function's properties may be unknown or difficult to interpret, e.g. how does one specify priors in an Atari game learning from pixels? Impact of our ideas could be amplified by research into automation of BNN design (Duvenaud et al., 2013; Steinruecken et al., 2018), and into how priors could be specified at a more abstract level.

Acknowledgments

Thanks to the anonymous reviewers for their helpful comments and suggestions. The lead author was funded through EPSRC (EP/N509620/1).

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstr, D. (2015). Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*.

Cho, Y. and Saul, L. K. (2009). Kernel Methods for Deep Learning. In *Advances in Neural Information Processing Systems 22*.

Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian Q-learning. In *American Association for Artificial Intelligence (AAAI)*.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., and Efros, A. A. (2018). Investigating Human Priors for Playing Video Games. In *Proceedings of the 35th International Conference on Machine Learning*.

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2013). Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In *Proceedings of the 30th International Conference on Machine Learning*.

Duvenaud, D. K. (2014). *Automatic Model Construction with Gaussian Processes*. PhD thesis.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. (2017). Mapping Gaussian Process Priors to Bayesian Neural Networks. *Bayesian Deep Learning Workshop, Neural Information Processing Systems (NeurIPS)*.

Gaier, A. and Ha, D. (2019). Weight Agnostic Neural Networks. *arXiv preprint: 1906.04358*.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian Active Learning with Image Data. In *Proceedings of the 34th International Conference on MachineLearning*.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-dickstein, J. (2018). Deep neural networks as gaussian processes. In *ICLR 2018*.

Ma, C., Li, Y., and Hernández-Lobato, J. M. (2019). Variational Implicit Processes. In *Proceedings of the 36th International Conference on Machine Learning*.

MacKay, D. J. C. (1998). *Introduction to Gaussian Processes*. Springer-Verlag.

Mnih, V., Antonoglou, I., Fidjeland, A. K., Wierstra, D., King, H., Bellemare, M. G., Legg, S., Petersen, S., Riedmiller, M., Beattie, C., Graves, A., Sadik, A., Kavukcuoglu, K., Ostrovski, G., Veness, J., Rusu, A. A., Silver, D., Hassabis, D., and Kumaran, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nalisnick, E. T. (2018). *On Priors for Bayesian Neural Networks*. PhD thesis.

Neal, R. M. (1997). *Bayesian Learning for Neural Networks*. PhD thesis.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational Continual Learning. In *ICLR 2018*.

Novak, R., Xiao, L., Lee, J., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. (2019). Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. In *ICLR 2019*.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2017). Taming the Waves: Sine As Activation Function in Deep Neural Networks. *openreview preprint*.

Pearce, T., Anastassacos, N., Zaki, M., and Neely, A. (2018). Bayesian Inference with Anchored Ensembles of Neural Networks, and Application to Exploration in Reinforcement Learning. In *Exploration in Reinforcement Learning Workshop, ICML*.

Pearce, T., Brintrup, A., and Neely, A. (2019). Uncertainty in Neural Networks: Bayesian Ensembling. *arXiv preprint 1810.05546*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for Activation Functions. *arXiv preprint 1710.05941*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*.

Saul, A. D., Hensman, J., and Lawrence, N. D. (2016). Chained Gaussian Processes. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016*.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*.

Steinruecken, C., Smith, E., Janz, D., Lloyd, J., and Ghahramani, Z. (2018). The Automatic Statistician. *https://www.ml4aad.org*.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines*. Springer.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional Variational Bayesian Neural Networks. In *ICLR 2019*.

Tsuchida, R., Roosta-Khorasani, F., and Gallagher, M. (2018). Invariance of Weight Distributions in Rectified MLPs. In *Proceedings of the 35th International Conference on Machine Learning*.

Williams, C. K. I. (1996). Computing with infinite networks. In *Advances in Neural Information Processing Systems 9*.

Wilson, A. G. and Adams, R. P. (2013). Gaussian Process Kernels for Pattern Discovery and Extrapolation. In *Proceedings of the 30th International Conference on Machine Learning*.