Deep Neural Networks as Point Estimates for Deep Gaussian Processes

Vincent Dutordoir[1,2], James Hensman[*3], Mark van der Wilk[4], Carl Henrik Ek[1], Zoubin Ghahramani[1,5], and Nicolas Durrande[2]

[1]University of Cambridge
[2]Secondmind.ai
[3]Amazon
[4]Imperial College London
[5]Google Research, Brain Team

Abstract

Deep Gaussian processes (DGPs) have struggled for relevance in applications due to the challenges and cost associated with Bayesian inference. In this paper we propose a sparse variational approximation for DGPs for which the approximate posterior mean has the same mathematical structure as a Deep Neural Network (DNN). We make the forward pass through a DGP equivalent to a ReLU DNN by finding an interdomain transformation that represents the GP posterior mean as a sum of ReLU basis functions. This unification enables the initialisation and training of the DGP as a neural network, leveraging the well established practice in the deep learning community, and so greatly aiding the inference task. The experiments demonstrate improved accuracy and faster training compared to current DGP methods, while retaining favourable predictive uncertainties.

1 Introduction

Bayesian inference has the potential to improve deep neural networks (DNNs) by providing 1) uncertainty estimates for robust prediction and downstream decision-making, and 2) an objective function (the marginal likelihood) for hyperparameter selection [MacKay, 1992a; 1992b; 2003]. The recent success of deep learning [Krizhevsky et al., 2012; Vaswani et al., 2017; Schrittwieser et al., 2020] has renewed interest in large-scale Bayesian Neural Networks (BNNs) as well, with effort mainly focused on obtaining useful uncertainty estimates [Blundell et al., 2015; Kingma et al., 2015; Gal and Ghahramani, 2016]. Despite already providing usable uncertainty estimates, there is significant evidence that current approximations to the uncertainty on neural network weights can still be significantly improved [Hron et al., 2018; Foong et al., 2020]. The accuracy of the uncertainty approximation is also linked to the quality of the marginal likelihood estimate [Blei et al., 2017]. Since hyperparameter learning using the marginal likelihood fails for most common approximations [e.g., Blundell et al., 2015], the accuracy of the uncertainty estimates is also questionable.

Damianou and Lawrence [2013] used Gaussian processes [Rasmussen and Williams, 2006] as layers to create a different Bayesian analogue to a DNN: the Deep Gaussian process (DGP). Gaussian processes (GPs) are a different representation of a single layer neural network, which is promising because it allows high-quality approximations to uncertainty [Titsias, 2009; Burt et al., 2019]. DGPs are promising, since they

*Work done while at Secondmind.ai. Correspondence to vd309@cam.ac.uk.

seem to inherit these high-quality approximations, as indicated by the successful use of marginal likelihood approximations for hyperparameter learning [Damianou and Lawrence, 2013; Dutordoir et al., 2020b].

Despite their advantages, DGPs have not been adopted as widely as DNNs, which can mainly be attributed to their larger computational cost and the fact that their are challenging to optimise. In the past, these challenges were also present in DNNs, although decades of work has led to standard practices (e.g. ReLU activations [Glorot et al., 2011], Xavier weight initialisations [Glorot and Bengio, 2010], and batch normalisation [Ioffe and Szegedy, 2015]) that have largely eliminated them. So far, it has not been possible to directly apply these methods to DGPs due to the mathematical differences between the models.

In this paper, we address the specific problems of computational cost and optimisation speed, as well as the broader problem of DGPs not being able to take advantage of improvements in DNNs. We do this by mathematically unifying the forward passes of the two methods, so a DGP configuration can be found with the same predictions as a “standard practice” DNN. This enables a two-stage training procedure, where a DNN is first trained as usual, after which it is then used as an initialisation for a more expensive DGP training step. The DNN training obtains a good fit quickly, while the DGP training improves uncertainty estimation. Our approach relies on establishing a correspondence between *inducing points* in GP approximations and neurons in a neural network layer. We make a forward pass through a DGP equivalent to a ReLU DNN by finding an *interdomain* transformation that represents the GP posterior mean as a sum of ReLU basis functions.

Empirically, we show that the modified training procedure mitigates some of the barriers which prevent the more widespread adoption of DGPs. More broadly, we hope that our work will lead to more research into getting the best properties of both DNNs and DGPs. With our equivalence, successful practices for training DNNs can be used to speed up DGPs, while the theoretically grounded objective functions of DGPs may inspire new training techniques in DNNs.

2 Related Work

MacKay [1992a; 1992b] noted very early that the Bayesian treatment of neural networks had large potential. To start, it can help to quantify estimates of the error bars on the network outputs, but it can also provide an objective that can be used for the comparison of alternative network architectures. The reality, however, was that Bayesian inference in neural networks was –and still is– challenging, and in practice one has to resort to either crude approximations (e.g., Laplace approximation [MacKay, 1998]) or lengthy computations (e.g., Markov Chain Monte Carlo [Neal, 1992]).

Building on the foundational work of MacKay, and driven by the amazing successes of large deterministic deep neural networks (DNNs), the literature has seen an upspring in the development of more scalable methods to perform approximate Bayesian inference in DNNs [Blundell et al., 2015; Kingma et al., 2015; Gal and Ghahramani, 2016]. However, it remains challenging to encode prior assumptions on functions through distributions on weights and the large number of parameters to be estimated makes it computationally prohibitive. Furthermore, the strong approximations used both during modelling and inference make it unclear to what extent these models approximate the true posterior distribution [Hron et al., 2018; Foong et al., 2020]. They also do not deliver on an important promise of Bayesian methods: an approximate marginal likelihood objective that can be used for automatic model selection and hyperparameter learning. A different approach may thus be necessary to unlock the Bayesian benefits in deep learning.

Neal [1995] showed that for infinitely wide single-layer BNNs the distribution over the non-linear functions are given by Gaussian processes. Williams [1998] and Cho and Saul [2009] extended this theory and derived the kernel corresponding to an infinite-width BNN with Sigmoidal and ReLU activation function, respectively. The beauty of this connection is that performing Bayesian inference in the corresponding GP model can be done exactly and analytically – all in a single elegant framework [Rasmussen and Williams, 2006]. Since, various approximations to the exact GP framework have been developed to allow for non-Gaussian likelihoods [Kuss and Rasmussen, 2005; Hensman et al., 2013], large datasets [Hensman et al., 2015; Wang et al., 2019], and even neural network like structures such as convolutions [van der Wilk et al., 2017]. Crucially, the approximations to the marginal likelihood still enable the main Bayesian benefits: model uncertainty and learning model hyperparameters (e.g., [van der Wilk et al., 2018; Dutordoir et al., 2020b]).

More recently, Matthews et al. [2018] discovered the equivalence between *deep* (i.e. multi-layer) BNNs and GPs. This has led to the development of deep (fully connected and convolutional) neural network kernels for GPs (NN-GPs). Interestingly, the performance of the non-Bayesian neural nets significantly outperforms the corresponding GPs [Garriga-Alonso et al., 2018; Novak et al., 2019]. The discrepancy hints at the fact that these single-layer GPs, even when configured with very expressive DNN equivalent kernels, are missing a crucial component: the ability to learn feature representations from the data.

Deep Gaussian processes [Damianou and Lawrence, 2013, DGPs] are a interesting avenue to tackle these challenges. They are built by hierarchically stacking GPs on top of each other, which enables the learning of more powerful representations through compositional features. Moreover, their Bayesian approximations in function-space seem to be of higher quality than those of weight-space BNNs, e.g. as supported by the successful use of marginal likelihood estimates for hyperparameter learning [Damianou and Lawrence, 2013].

While promising, DGPs have struggled for relevance in applications due to the challenges and cost associated with Bayesian inference. Training DGPs is computationally expensive and requires very careful setting of the parameters. Furthermore, the hierarchical prior induced by naively stacking stationary kernels gives rise to pathological, "collapsed" samples Duvenaud et al. [2014]. Considerable progress for scalable inference in DGPs was made by Hensman and Lawrence [2014] and Salimbeni and Deisenroth [2017], who derived stochastic variational lower bounds. The formulation of these bounds closely resembles the computations of training a feed-forward neural network with regularisation terms, and has greatly inspired this work.

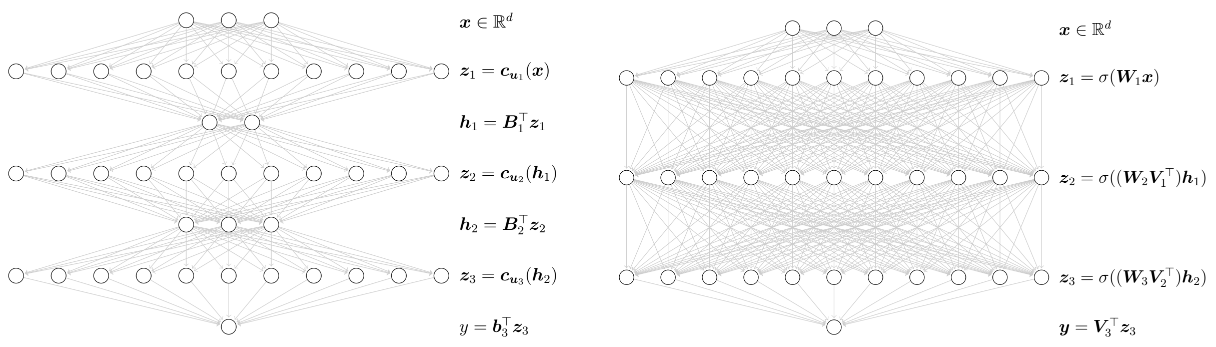
Building on Salimbeni and Deisenroth [2017] and to further improve the scalability of DGPs, Cutajar et al. [2017] used a Random Fourier Feature [Rahimi and Recht, 2008, RFF] approximation of the kernel. While successful, this approach introduces an approximation in both the prior and the posterior of the model. More recently, Rudner et al. [2020] proposed Fourier features of the Matérn RKHS, following Hensman et al. [2018, Variational Fourier Features (VFF)], to build inter-domain DGPs. Like for single layer VFF models, this approach can lead to faster training and improved performance, but is only computationally feasible for data of dimension one or two. In parallel with our work, Sun et al. [2021] explored the idea of using single-layer neural networks to parameterise inducing points in shallow GPs. Similar to this paper, their method makes use of the spherical harmonic decomposition of a kernel. However, their works differs in the fact that they focus on shallow GPs, on bounded inducing functions (e.g., erf), and directly use the Nyström approximation to approximate the model's uncertainty estimates rather than the ELBO to learn the variational parameters.

The analysis of (Bayesian) neural networks has led to several probabilistic models: NN-GPs, GPs and DGPs. In this work, however, rather than focusing on the *prior* induced by these equivalent models, the emphasis lies on the connection between the DGP *posterior* and the DNN. This has received much less attention in the literature, though we argue it is a more interesting regime to study. The connection between GP piors and neural nets is established only in the infinite limit of the number of hidden units. The sparse posterior DGP, on the contrary, is built out of a finite set of basis functions and can thus immediately be connected to finite-width neural networks — in this paper we connect both models in their modus operandi.

We formulate a DGP configuration for which the approximate posterior mean has the same mathematical structure as a deep neural network with fully connected layers and non-linear activation functions. We can use this unification to train the DGP like a neural network which allow us to leverage all the great research in this area for DGP inference. Furthermore, this connection between DGPs posteriors and DNNs highlights the great potential of DGPs as a model for learning powerful representations from data while being fully Bayesian.

3 Background

In this section, we review Gaussian processes (GPs) and Deep Gaussian processes (DGPs), with a particular focus on the structure of the sparse DGP approximation, and show how scalable variational inference is achieved in these models. We then discuss a framework for defining new inducing variables which leads to global inducing functions — a key ingredient for our proposed DGP approximation.


$\boldsymbol{x} \in \mathbb{R}^d$
$\boldsymbol{z}_1 = \boldsymbol{c}_{\boldsymbol{u}_1}(\boldsymbol{x})$
$\boldsymbol{h}_1 = \boldsymbol{B}_1^\top \boldsymbol{z}_1$
$\boldsymbol{z}_2 = \boldsymbol{c}_{\boldsymbol{u}_2}(\boldsymbol{h}_1)$
$\boldsymbol{h}_2 = \boldsymbol{B}_2^\top \boldsymbol{z}_2$
$\boldsymbol{z}_3 = \boldsymbol{c}_{\boldsymbol{u}_3}(\boldsymbol{h}_2)$
$y = \boldsymbol{b}_3^\top \boldsymbol{z}_3$
$\boldsymbol{x} \in \mathbb{R}^d$
$\boldsymbol{z}_1 = \sigma(\boldsymbol{W}_1 \boldsymbol{x})$
$\boldsymbol{z}_2 = \sigma((\boldsymbol{W}_2 \boldsymbol{V}_1^\top)\boldsymbol{h}_1)$
$\boldsymbol{z}_3 = \sigma((\boldsymbol{W}_3 \boldsymbol{V}_2^\top)\boldsymbol{h}_2)$
$\boldsymbol{y} = \boldsymbol{V}_3^\top \boldsymbol{z}_3$

(a) Deep Gaussian process posterior

(b) Fully-connected deep neural net

Figure 1: Comparison between the mean of the approximate posterior of a DGP **(a)** and a DNN **(b)**. The goal is to design basis functions $\boldsymbol{c}_{\boldsymbol{u}}(\cdot)$ for the DGP that are similar to the by $\mathbf{W}$-parameterised activation functions $\sigma(\mathbf{W}\,\cdot)$ used in the DNN.

3.1 Gaussian Processes and Sparse Variational Gaussian Processes

Gaussian processes [Rasmussen and Williams, 2006, GPs] are a probabilistically grounded framework for modelling data allowing for principled uncertainty quantification and automatic selection of hyperparameters through a marginal likelihood objective. The distribution of the GPs is fully specified by a mean $\mu(\cdot)$ and kernel $k(\cdot,\cdot)$ function, the latter of which encodes the covariance between two GP evaluations $\mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')) = k(\boldsymbol{x},\boldsymbol{x}')$. Given a dataset with inputs $\mathbf{X} \in \mathbb{R}^{N\times d}$ and outputs $\boldsymbol{y} \in \mathbb{R}^N$, assumed corrupted by additive noise $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$, we are interested in reaching the posterior distribution $f(\cdot) \,|\, \boldsymbol{y}$.

Computing the exact posterior distribution scales as $\mathcal{O}(N^3)$, because it requires inverting a $N \times N$ matrix. To circumvent this, sparse approximations have been introduced [Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005]. The sparse approximation is constructed by introducing a set of M *inducing variables* $\{u_m\}_{m=1}^M$, which represent function values at some input locations $\mathbf{W} = \{\boldsymbol{w}_m\}_{m=1}^M$, such that $u_m = f(\boldsymbol{w}_m)$. Due to self-conjugacy and the marginal property of the GP the conditioned distribution is another GP,

$$p(f(\cdot)\,|\,\boldsymbol{u}) = \mathcal{GP}\big(\boldsymbol{a}^\top \boldsymbol{c}_{\boldsymbol{u}}(\cdot); \quad k(\cdot,\cdot') - \boldsymbol{c}_{\boldsymbol{u}}^\top(\cdot)\mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{c}_{\boldsymbol{u}}(\cdot')\big) \quad \text{with} \quad \boldsymbol{a} = \mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{u}, \tag{1}$$

and

$$[\boldsymbol{c}_{\boldsymbol{u}}(\cdot) \in \mathbb{R}^M]_m = \mathrm{Cov}(f(\cdot), u_m) = k(\cdot, \boldsymbol{w}_m), \text{ and } [\mathbf{C}_{\boldsymbol{u}\boldsymbol{u}} \in \mathbb{R}^{M\times M}]_{m,m'} = \mathrm{Cov}(u_m, u_{m'}) = k(\boldsymbol{w}_m, \boldsymbol{w}_{m'}).$$

In Sparse Variational GPs [Titsias, 2009, SVGPs] we interpret the inducing variables as *variational parameters* approximating the posterior $p(\boldsymbol{u}\,|\,\boldsymbol{y})$, as $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. By combining this variational distribution with the conditional, predictive distribution in Eq. (1) leads the approximate posterior distribution

$$q(f(\cdot)) = \mathcal{GP}\big(\boldsymbol{b}^\top \boldsymbol{c}_{\boldsymbol{u}}(\cdot); \quad k(\cdot,\cdot') + \boldsymbol{c}_{\boldsymbol{u}}^\top(\cdot)\mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}(\boldsymbol{\Sigma} - \mathbf{C}_{\boldsymbol{u}\boldsymbol{u}})\mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{c}_{\boldsymbol{u}}(\cdot')\big), \quad \text{where} \quad \boldsymbol{b} = \mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{\mu}. \tag{2}$$

When we are dealing with multi-output GPs $\{f_p(\cdot)\}_{p=1}^P$, then $\boldsymbol{b}_p = \mathbf{C}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{\mu}_p$ with $\boldsymbol{\mu}_p$ and $\boldsymbol{\Sigma}_p$ the mean and covariance of the inducing outputs corresponding to the p^{th} output, that is $u_{m,p} = f_p(\boldsymbol{z}_m)$ and $q(\boldsymbol{u}_p) = \mathcal{N}\big(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p\big)$. We collect $\{\boldsymbol{b}_p \in \mathbb{R}^M\}_{p=1}^P$ as the columns of matrix $\mathbf{B} \in \mathbb{R}^{M\times P}$.

The variational parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ (or $\{\boldsymbol{\mu}_p\}_{p=1}^P$, $\{\boldsymbol{\Sigma}_p\}_{p=1}^P$ in the multi-output case) and the hyperparameters can be optimised by minimising the Kullback-Leibler (KL) divergence between this approximate posterior and the true posterior through a lower bound. The overall complexity of the model is $\mathcal{O}(M^3 + PM^2N)$, so choosing $M \ll N$ enables significant speedups compared to vanilla GP models. It also enables mini-batching [Hensman et al., 2013] and non-Gaussian likelihoods [Hensman et al., 2015].

3.2 Deep Gaussian Processes and its Similarity with DNNs

A Deep Gaussian process [Damianou and Lawrence, 2013, DGP] is defined as the functional composition

$$\mathcal{F}(\cdot) = f_L(\dots f_2(f_1(\cdot))) \qquad \text{with} \qquad f_\ell(\cdot) \sim \mathcal{GP}\big(0, k_\ell(\cdot,\cdot)\big).$$

Each layer in a DGP is a GP, and the output of the previous layer is fed as the input to the next. Later, we will refer to the latent function-evaluation of a hidden GP as $\boldsymbol{h}_\ell = f_\ell(\boldsymbol{h}_{\ell-1})$ and define $\boldsymbol{h}_0 = \boldsymbol{x}$ for convenience. Note that the hidden GPs in DGPs are typically multi-output.

For inference, we define L independent approximate posterior GPs $q(f_\ell(\cdot))$ of the form in Eq. (2). Following this procedure, the mean of the deep approximate posterior admits a particular structure:

$$\mathbb{E}[\mathcal{F}_{\mathrm{DGP}}(\boldsymbol{x})] = \mathbf{B}_L^\top \boldsymbol{c}_{\boldsymbol{u}_L}(\dots \mathbf{B}_2^\top \boldsymbol{c}_{\boldsymbol{u}_2}(\mathbf{B}_1^\top \boldsymbol{c}_{\boldsymbol{u}_1}(\boldsymbol{x})). \tag{3}$$

A visual example of this compositional structure for a three-layer DGP is given in Fig. 1a. The relationship between the mean of the DGPs and DNNs becomes apparent when formulating a single fully-connected layer as $\mathbf{V}^\top \sigma(\mathbf{W}x)$ with non-linearity $\sigma(\cdot)$, and using this to compose a L-layer DNN. This yields:

$$\mathcal{F}_{\mathrm{DNN}}(\boldsymbol{x}) = \mathbf{V}_L^\top \sigma(\mathbf{W}_L \dots \mathbf{V}_2^\top \sigma(\mathbf{W}_2 \mathbf{V}_1^\top \sigma(\mathbf{W}_1 \boldsymbol{x})), \tag{4}$$

and is visualised in Fig. 1b for a three-layer case.

To unify DNNs and DGPs it becomes clear that we need to design a covariance function $\boldsymbol{c}_{\boldsymbol{u}}(\cdot)$ that is similar to the basis functions of a neural net $\sigma(\mathbf{W}\cdot)$. This will be the remaining challenge of the paper. Once this is achieved the mean of the posterior DGP becomes mathematically similar to the forward pass in a DNN. This connection allows us, firstly, to train the DGP posterior mean as if it is a neural network. Secondly, we can obtain DGPs with the same inductive bias as DNNs with ReLU activation functions, which has extensively been shown in practice to result in great accuracy [Glorot et al., 2011].

Once the mean of the DGP is trained using the objectives in deep learning (e.g., mean-squared-error for regression or cross-entropy for classification) we train the remaining variational parameters $\{\boldsymbol{\Sigma}_\ell\}_{\ell=1}^L$ and model hyperparameters by optimising the standard lower bound on the log marginal likelihood (the evidence lower bound, or ELBO):

$$\mathrm{ELBO} = \sum\nolimits_i \mathbb{E}_{q(h_{i,L})}[\log p(y_i \mid h_{i,L})] - \sum\nolimits_\ell \mathrm{KL}[q(\boldsymbol{u}_\ell) \,\|\, p(\boldsymbol{u}_\ell)] \leq \log p(\boldsymbol{y}). \tag{5}$$

An unbiased estimate of the ELBO can be achieved by subsampling the data in minibatches and iteratively sampling points through the layers of the DGP, following [Salimbeni and Deisenroth, 2017].

The remaining discrepancy between the DGP and DNN architectures is the low-rank structure induced on the weight matrices $\mathbf{W}_{\ell+1}\mathbf{V}_\ell^\top$ as the result of the explicit modelling of output heads in the GP layers. In vanilla DNNs this is not something that is done explicitly and weight matrices are typically set to be full-rank. However, it has been empirically shown that trained DNNs can have low rank matrices as evidenced by the ability to predict a part of the weights given the others [Denil et al., 2013]. Other research has also highlighted the surprisingly low intrinsic dimension of large DNNs [Li et al., 2018]. Dropout is another example used to control the complexity of overparameterised models [Srivastava et al., 2014].

3.3 Interdomain Gaussian Processes

The choice of inducing variable determines the computational complexity of the approximation as well as the basis functions $[\boldsymbol{c}_{\boldsymbol{u}}(\cdot)]_m = \mathrm{Cov}(f(\cdot), u_m)$, of which the approximate posterior mean in Eq. (2) is a linear combination. The most common approach is to condition the Sparse GP on pseudo observations at pseudo input locations, that is $u_m = f(w_m)$. This construction leads to basis functions $[\boldsymbol{c}_{\boldsymbol{u}}(\cdot)]_m = k(\boldsymbol{w}_m, \cdot)$ which are simple evaluations of the kernel. Using such inducing variables with stationary kernels (like the Squared Exponential or Matérn family kernels) result in basis functions that only have a local influence on the posterior, in the vicinity of the inducing points w_m. While this local sparse representation can be beneficial, it is fundamentally different to how DNNs are traditionally constructed where the activation functions are global. Furthermore, having only local influence can be problematic, especially in high dimensions, as many inducing points are necessary to sufficiently cover the space and obtain satisfactory performance.

Interdomain Gaussian processes (see van der Wilk et al. [2020] and Leibfried et al. [2020] for a modern and thorough review) use an alternative construction for the inducing variables. They are obtained by applying a linear operator $\mathcal{L}_m$ on the entire GP $f(\cdot)$, which gives:

$$u_m = \mathcal{L}_m(f(\cdot)) \tag{6}$$

A common choice for $\mathcal{L}_m$ is to use the integral operator, parameterised by a function $g_m(\cdot)$ [Lázaro-Gredilla and Figueiras-Vidal, 2009], such that $u_m = \int f(\boldsymbol{x})\, g_m(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$. This generalisation of inducing variables has the potential to lead to global inducing variables that encode information about the entire GP — rather than just *local* information at inducing point locations. The resulting basis functions $[\boldsymbol{c}_{\boldsymbol{u}}(\cdot)]_m = \mathrm{Cov}(u_m, f(\cdot))$ (and $\mathbf{C}_{\boldsymbol{uu}} = \mathrm{Cov}(\boldsymbol{u}, \boldsymbol{u})$) depend on the choice of kernel and linear operator, but through this construction, we can give them global influence. Crucially, the SVGP framework and more specifically the mean and covariance in Eq. (2) remain unaltered by changing inducing variables.

Alternatively, Hensman et al. [2018] used the RKHS inner product to construct Fourier features, and Dutordoir et al. [2020a] to build spherical harmonic basis functions. Their methods induce sparse structure in $\mathbf{C}_{\boldsymbol{uu}}$ which leads to increased computational efficiency. This paper builds on top of these works. More specifically, access to the RKHS can be used to construct inducing variables u_m for which we have full control over the resulting basis functions $[\boldsymbol{c}_{\boldsymbol{u}}(\cdot)]_m$. We use this mechanism to design basis functions of the same form of neural networks.

4 Activated Gaussian Processes

We lay out the building blocks required to design SVGP layers for which the approximate posterior mean, given in Eq. (2), has the same mathematical expression as a fully connected neural net layer. By stacking these layers on top of each other we obtain a DGP whose approximate posterior mean is equivalent to a deep neural net with multiple fully connected layers. We refer to our model as *Activated* (D)GPs because of the activation function-like behaviour of the GP's basis functions.

We start by discussing the projection of our data onto the hypersphere, which is inspired by but not limited to the Arc Cosine kernel. Once the data lives on the hypersphere we recall zonal kernels and define their Reproducing Kernel Hilbert Space (RKHS) in Section 4.2. Finally, in Section 4.3 we construct our interdomain inducing variables that induce ReLU basis functions.

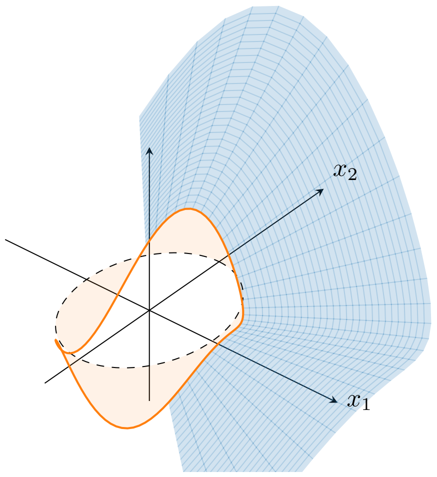
4.1 The Arc Cosine Kernel and Projection to $\mathbb{S}^{d-1}$

The Arc Cosine kernel mimics the computation of infinitely wide fully connected layers with ReLU activations. For $\sigma(t) = \max(0, t)$, Cho and Saul [2009] showed that the covariance between function values of $f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$ for $\boldsymbol{w} \sim \mathcal{N}\big(0, M^{-1/2}\mathbf{I}\big)$ and $\boldsymbol{w} \in \mathbb{R}^M$ is given by

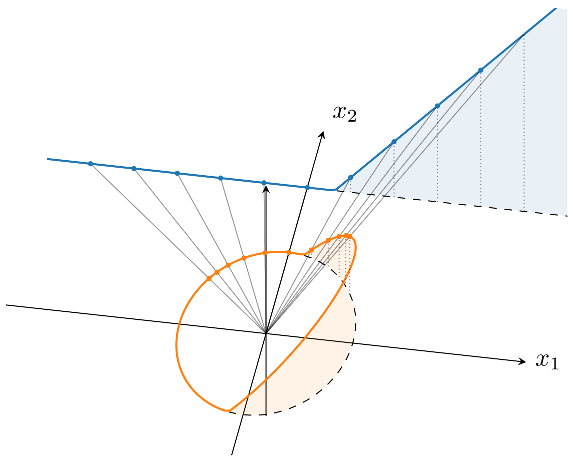
$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}}\big[\sigma(\boldsymbol{w}^\top \boldsymbol{x})\, \sigma(\boldsymbol{w}^\top \boldsymbol{x}')\big] = \underbrace{||\boldsymbol{x}||||\boldsymbol{x}'||}_{\text{radial}}\ \underbrace{\frac{1}{\pi}\big(\sqrt{1-t^2} + t\,(\pi - \arccos t)\big)}_{\text{angular (shape function) } s(t)}, \tag{7}$$

where $t = \cos \frac{\boldsymbol{x}^\top \boldsymbol{x}}{||\boldsymbol{x}||||\boldsymbol{x}'||}$. The factorisation of the radial and angular dependency in the kernel leads to an RKHS with only homogeneous functions of the form $f(\boldsymbol{x}) = ||\boldsymbol{x}||\, g(\frac{\boldsymbol{x}}{||\boldsymbol{x}||})$, where $g(\cdot)$ is defined on the unit hypersphere $\mathbb{S}^{d-1}$ but fully determines the function on $\mathbb{R}^d$. In Fig. 2, we give examples of such functions in $\mathbb{R}^2$ with $g(\cdot)$ defined on the unit circle $\mathbb{S}^1$.

In our method, we follow this setup and define the GP on the unit hypersphere $\mathbb{S}^{d-1}$. The kernels we employ for this GP on the hypersphere only depend on the dot-product between the inputs and are studied next. The overall function is defined in $\mathbb{R}^d$ by the homogeneous linear extension of the GP on the hypersphere.



(a) $g(\cdot)$ is a GP draw



(b) $g(\boldsymbol{x}) = \max(0, \boldsymbol{w}^\top \boldsymbol{x})$

Figure 2: Examples of homogeneous functions of the form $f(\boldsymbol{x}) = ||\boldsymbol{x}||\, g(\boldsymbol{x}/||\boldsymbol{x}||)$. In **(a)** we depict how the function is defined on $\mathbb{R}^2$ by linearly extrapolating the value on the unit circle. In **(b)** we show a slice of the homogeneous function $f(\boldsymbol{x})$ for a fixed value of x_2. In **(b)** we see that setting $g(\boldsymbol{x}) = \max(0, \boldsymbol{w}^\top \boldsymbol{x})$ we obtain ReLU behaviour for $f(\boldsymbol{x})$. The parameter $\boldsymbol{w} \in \mathbb{S}^1$ determines the ReLU's change point.

However, as is the case in Dutordoir et al. [2020a] we will only use a single slice of the overall function because we embed our $(d-1)$-dimensional dataset on a hyperplane in $\mathbb{R}^d$. In practice, this is simply done by concatenating the data with a constant bias $b \in \mathbb{R}$, $\boldsymbol{x} = [\boldsymbol{x}, b]$. Once the data is embedded in the plane, we linearly project the points to the unit hypersphere by normalisation $\frac{\boldsymbol{x}}{||\boldsymbol{x}||}$ and define the GP on there.

4.2 Reproducing Kernel Hilbert Space for Zonal Kernels on $\mathbb{S}^{d-1}$

A kernel $k(\cdot, \cdot')$ defined on the hypersphere $\mathbb{S}^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}||_2 = 1\}$ is said to be *zonal* if it depends on the dot-product between the inputs. In other words, we have for a zonal kernel $k : \mathbb{S}^{d-1} \times \mathbb{S}^{d-1} \to \mathbb{R}$ and $k(\boldsymbol{x}, \boldsymbol{x}') \mapsto s(\boldsymbol{x}^\top \boldsymbol{x}')$, where we refer to $s : [-1, 1] \to \mathbb{R}$ as the *shape* function. Similar to stationary kernels which are translationally invariant, zonal kernels are rotationally invariant. It has been shown (see, for example, Wendland [2005] or Dutordoir et al. [2020a]) that a zonal kernel shares its eigenfunctions with the Laplace-Beltrami operator, for which it is well-known that the eigenfunctions are the spherical harmonics $\phi_{n,j}(\cdot)$ [Dai and Y. Xu, 2013; Efthimiou and Frye, 2014]. We label this set using two indices: $n \in \mathbb{N}$ stands for the degree (also referred to as level or frequency) and $j \in [1, \dots, N_n^d]$ indexes the spherical harmonics within a level. See Fig. B.1 for a visual representation of the spherical harmonics on the unit sphere $\mathbb{S}^2$. Having access to the eigenvalues λ_n and eigenfunctions $\phi_{n,j}(\cdot)$ of a kernel allows us to write down the kernel's Mercer decomposition:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \lambda_n \phi_{n,j}(\boldsymbol{x})\, \phi_{n,j}(\boldsymbol{x}'). \tag{8}$$

The set of spherical harmonics $\{\phi_{n,j}(\cdot)\}$ forms an orthonormal basis on $\mathbb{S}^{d-1}$. There are $N_n^d = \frac{2n+d-2}{n}\binom{n+d-3}{d-1}$ number of harmonics in a level.

Crucially, the eigenvalues λ_n of a zonal kernel only depend on the degree of the spherical harmonic and *not* on j. They are given by

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} s(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t, \tag{9}$$

where $C_n^{(\alpha)}(\cdot)$ is the Gegenbauer polynomial of degree n with specificity $\alpha = \frac{d-2}{2}$, $\omega_d = \Omega_{d-2}/\Omega_{d-1}$ and Ω_{d-1} denotes the surface area of $\mathbb{S}^{d-1}$ (see Appendix B for analytical expressions).

As the kernel's eigenvalues only depend on the level of the spherical harmonics, we can make use of the so-called Addition theorem (see Appendix B) to simplify Eq. (8), thus obtaining

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{n=0}^{\infty} \lambda_n \frac{n+\alpha}{\alpha}\, C_n^{(\alpha)}(\boldsymbol{x}^\top \boldsymbol{x}'). \tag{10}$$

This is important as it circumvents the need to compute the spherical harmonics in order to evaluate the kernel.

A standard approach for constructing the RKHS of a p.d. kernel is through its spectral decomposition. For zonal kernels this yields

$$\mathcal{H} = \left\{ f(\cdot) = \sum_{\substack{n=0 \\ \lambda_n \neq 0}}^{\infty} \sum_{j=1}^{N_n^d} f_{n,j} \phi_{n,j}(\cdot) : ||f||_{\mathcal{H}} < \infty \right\}, \text{ with } \langle g, h \rangle_{\mathcal{H}} = \sum_{\substack{n=0 \\ \lambda_n \neq 0}}^{\infty} \sum_{j=1}^{N_n^d} \frac{g_{n,j} h_{n,j}}{\lambda_n} \tag{11}$$

the *reproducing* inner product between two functions in the RKHS $g(\cdot) = \sum_{n,j} g_{n,j}\phi_{n,j}(\cdot)$ and $h(\cdot) = \sum_{n,j} h_{n,j}\phi_{n,j}(\cdot)$, and $||f||_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$.

Decay rate The rate of decay for the eigenvalues of a kernel λ_n determines important characteristics of the RKHS [Kanagawa et al., 2018] and have been the subject of study in the recent literature. Bach [2017] investigated the decay rate of the first-order Arc Cosine kernel and noted that for large enough n, $\lambda_n = 0$ for n odd and decays as n^{-d-2} for n even (as shown Fig. 8). Bietti and Mairal [2019] showed that the Neural Tangent Kernel [Jacot et al., 2018, NTK] of a two-layer ReLU network has the same decay rate as the first-order Arc Cosine kernel, and pointed to the fact that the depth of a neural network does not change the corresponding RKHS. These results were generalised by Bietti and Bach [2020] for general zonal kernels. In parallel, Geifman et al. [2020] and Chen and S. Xu [2021] demonstrated that the RKHS on $\mathbb{S}^{d-1}$ of the Laplace kernel and NTK are closely related based on their similar decay rates.

For the first-order Arc Cosine kernel, the zero Fourier coefficients for odd levels above a given threshold impose a loose parity constraint on functions in the RKHS on $\mathbb{S}^{d-1}$. However, as discussed in Basri et al. [2019] and akin to our approach, this constraint is removed by adding a bias term to the data.

Many well-known kernels are zonal. Stationary kernels, such as the Squared-Exponential, Matérn family and Laplace, restricted to the hypersphere, are also zonal kernels. This can most easily be seen by noting that the distance between two vectors $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{S}^{d-1}$ can be written as a function of the dot-product: $||\boldsymbol{x} - \boldsymbol{x}'||_2^2 = 2(1 - \boldsymbol{x}^\top \boldsymbol{x}')$. The shallow and deep arc-cosine kernels from Cho and Saul [2009] and the more recent Neural Tangent Kernel [Jacot et al., 2018] are also zonal kernels when restricted to the hypersphere. These kernels conveniently enable access to both the analytic (left-hand-side) and eigenvalues and eigenfunctions (right-hand-side) of the Mercer decomposition from Eq. (8). In contrast, Solin and Särkkä [2020] and Borovitskiy et al. [2020] introduce zonal kernels on Riemannian manifolds, such as the hypersphere, but these kernels can only be expressed as an infinite sum.

4.3 Activated Inducing Variables

Following the interdomain approach (see Section 3.3) we define the inducing functions $g_m(\cdot) : \mathbb{S}^{d-1} \to \mathbb{R}$ as $\boldsymbol{x} \mapsto \sigma(\boldsymbol{w}_m^\top \boldsymbol{x})$, where we recall that $\sigma(t) = \max(0, t)$ and $\boldsymbol{w}_m \in \mathbb{S}^{d-1}$ is a parameter. As $g_m(\cdot)$ is a function

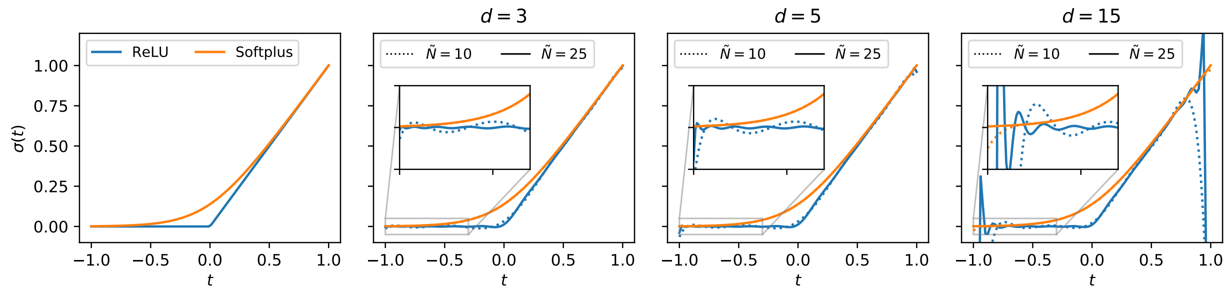


Figure 3: ReLU and Softplus[1] activation function **(left)**. Approximation of the activation function for different truncation levels $\tilde{N} = 10$ (dotted) and $\tilde{N} = 25$ (solid) for dimension $3, 5$ and 15 **(right)**.

of the dot-product of its arguments $\boldsymbol{w}_m$ and $\boldsymbol{x}$, its spectral decomposition can be computed similar to that of zonal kernels. This gives

$$g_m(\boldsymbol{x}) = \sigma(\boldsymbol{w}_m^\top \boldsymbol{x}) = \sum_{n=0}^{\infty} \sigma_n \frac{n+\alpha}{\alpha} C_n^{(\alpha)}(\boldsymbol{w}_m^\top \boldsymbol{x}), \tag{12}$$

where σ_n are the Fourier coefficients of the ReLU function that can be computed using Eq. (9) when $s(t)$ is replaced by $\sigma(t)$. As shown in Fig. 2b, this particular choice of inducing function gives rise to ReLU when linearly projecting the function values on $\mathbb{S}^{d-1}$ to the data plane.

To define the inducing variables we make use of the RKHS' inner product (Eq. (11)) and the inducing function defined above, yielding

$$u_m = \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}, \tag{13}$$

which is the projection in the RKHS of $f(\cdot) \sim \mathcal{GP}$ on the inducing functions $g_m(\cdot)$. The parameters of these inducing variables (similar to the inducing point input locations $\boldsymbol{w}_m$ in the traditional SVGP framework) are the $\boldsymbol{w}_m \in \mathbb{S}^{d-1}$. They determine the direction in which the one-dimensional shape function is oriented on the hypersphere.

By definition, the RKHS $\mathcal{H}$ only consists of functions that have a finite norm (see Eq. (11)). Many more functions, however, can be represented as a linear combination of the basis functions $\phi_{n,j}(\cdot)$ without being an element of the RKHS. A typical example of such a function is a sample from the GP itself. In principle, the definition of the inner product is limited to functions that are part of RKHS, which makes our inducing variables definition as per Eq. (13) questionable. This definition can however be given a rigorous meaning if the spectrum of $g_m(\cdot)$ decays quick enough so that the double sum in the inner product definition converges. In other words, the application $(f(\cdot), g_m(\cdot)) \mapsto \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}$ can be extended to less regular functions $f(\cdot)$, such as GP samples, by ensuring that $g_m(\cdot)$ is regular enough. For most common kernels, this additional regularity condition will not be satisfied if $g_m(\cdot)$ is defined as a ReLU inducing variable, but a trivial strategy to ensure the convergence is to make all the Fourier coefficients of the inducing function $g_m(\cdot)$ equal to zero after a certain truncation level $\tilde{N}$. Henceforth, we define a *truncated* inducing function $\tilde{g}_m(\cdot)$ for which the Fourier coefficients for $n \le \tilde{N}$ are equal to $g_m(\cdot)$ and 0 for $n > \tilde{N}$. Figure 3 shows truncated activation functions for the ReLU and Softplus for different truncation levels and input dimensions. We notice that for Softplus the truncated version is very similar to the true, even for small $\tilde{N}$.

The inducing variable defined in Eq. (13) can be used in the SVGP framework if we can (1) compute the pairwise covariance between the inducing variables for $\boldsymbol{c_u}(\cdot)$ and (2) the covariance between the GP and the inducing variables. We start with the latter, which yields

$$\mathrm{Cov}(u_m, f(\cdot)) = \langle k(\boldsymbol{x}, \cdot), \tilde{g}_m(\cdot) \rangle_{\mathcal{H}} = \tilde{g}_m(\boldsymbol{x}). \tag{14}$$

[1]We use a rescaled version of the softplus activation function $t \mapsto \frac{1}{\beta} \log(1 + \exp(\beta\, t))$ with $\beta = 5.0$ so that its values at -1.0 and 1.0 are 0.0 and 1.0, respectively. This is important because with this, the homogeneous extension of the function is the activation function itself.

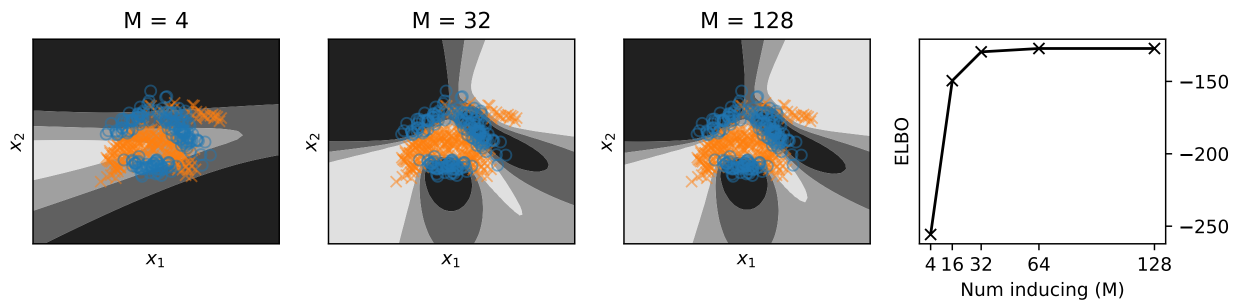


Figure 4: Fits on the Banana binary classification dataset with growing number of inducing variables M. **(left three panels)** In blue and orange we display the data. The model's $p(y = 1 \mid \boldsymbol{x})$ is given as shades of grey. **(right panel)** Evidence Lower BOund (ELBO) as a function of the number of inducing variables.

as a result of the reproducing property of the RKHS. This procedure gives rise to basis functions that match the ReLU activation function once projected onto the data plane, as shown in Fig. 2b. As a result, using these inducing variables leads to an approximate posterior GP (Eq. (2)) which has a mean that is equivalent to a fully connected neural net layer with a ReLU activations.

Finally, the covariance between inducing variables, required to populate $\mathbf{C}_{\boldsymbol{uu}}$, is given by

$$\mathrm{Cov}(u_m, u_{m'}) = \langle \tilde{g}_m(\cdot), \tilde{g}_{m'}(\cdot) \rangle_{\mathcal{H}} = \sum_{\substack{n=0 \\ \lambda_n \neq 0}}^{\tilde{N}} \frac{\sigma_n^2}{\lambda_n} \frac{n+\alpha}{\alpha} C_n^{(\alpha)}(\boldsymbol{w}_m^\top \boldsymbol{w}_{m'}), \tag{15}$$

where we used the definition of the RKHS inner product and the addition theorem for spherical harmonics.

5 Experiments

The experiments are organised as follows. In Section 5.1 we start with the Banana dataset, a toy classification problem which we use to verify the validity of our method and to investigate the uncertainty estimates produced by the models. In Section 5.2 we test our approach on a series of UCI regression benchmarks, where we compare models in terms of uncertainty estimation, accuracy and training time. Finally, in Section 5.3 we investigate different flavours of the model and identify deficiencies in certain configurations.

In the following sections we will refer to our models as *Activated* GPs and/or *Activated* DGPs because of the activation function-like basis functions.

5.1 Banana Dataset: 2D Toy Classification Problem

Tightness of the bound. An attractive property of SVGPs (and DGPs) is that the objective is a lower bound on the log marginal likelihood, where the gap is the KL divergence between the true and approximate posterior. By increasing the number of inducing variables we make our approximate posterior richer, which can only lower the error, or equivalently, tighten the bound. In Fig. 4 we verify this property for our ReLU inducing variables in a shallow (single layer) SVGP model. In the right panel we see that the ELBO increases with M and then starts to plateau. In the left panels we show a few of the fitted models. In this experiment we used the Arc Cosine kernel (Eq. (7)) with the Softplus activation function and set $\tilde{N} = 10$. Given the limited size of the dataset, we optimised the model's parameter using BFGS. This experiment validates the fact that increasing the number of inducing variables improves the ELBO, and that a good model can be obtained with a limited number of inducing variables.

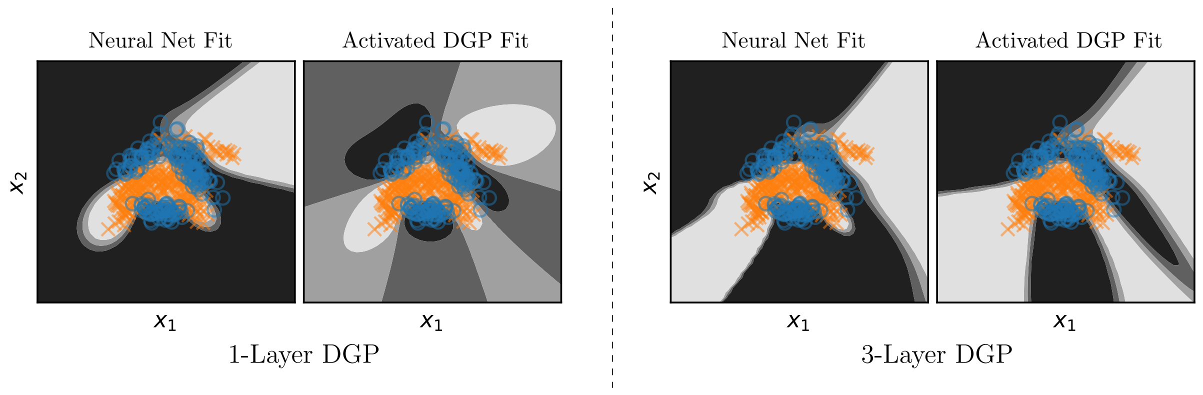


Figure 5: We evaluate both a single-layer DGP (left) and a three-layer DGP (right) and compare the fit of the equivalent neural network to that of the DGP. The experiment start by training the equivalent neural network using a binary cross-entropy objective. We notice how the neural networks make very confident decisions, even far away from the data. In a second step, we use the neural network to initialise the equivalent DGP posterior mean, and train the remaining parameters using the DGP ELBO. This leads, in both situations, to a more calibrated model.

From Neural Network to Activated DGP. In Fig. 5 we showcase our ReLU inducing variables for both a single-layer DGP (equivalent to a SVGP) and a three-layer DGP. In both models we use the Arc Cosine kernel, Softplus activation function and 100 inducing variables.

The mean of the approximate posterior for these DGP models corresponds to, respectively, a one- and three-layered fully-connected neural network with Softplus activations and 100 hidden units. For both models, we show the fit after minimising the network parameters using a binary cross-entropy objective on the left. More specifically, we plot $p(y \,|\, \boldsymbol{x})$ using different shadings of grey. From the plot, we notice how the neural network model exhibits very confident and strong extrapolations away from the data with predictive probabilities $p(y \,|\, \boldsymbol{x})$ that are very close to 0 or 1. The decision boundary ($p(y \,|\, \boldsymbol{x}) = 0.5$) is also very sharp.

If we now take the neural network model to initialise the equivalent DGP and continue training using the ELBO (given in Eq. (5)), we obtain the prediction on the right. For the single layer model, we notice how uncertainty away from the data is expressed by letting $p(y \,|\, \boldsymbol{x}) \approx 0.5$. In the three layer model we also see how the black regions are shrunk to lie closer to the data for which we are certain about class labels. The model's behaviour can be attributed to the fact that we continue training the model using the ELBO rather than the binary cross-entropy. The ELBO balances both data fit and model complexity and simultaneously trains the uncertainty in the hidden layers of the DGP.

5.2 Regression on UCI benchmarks

In Table 1 we show the evaluation of our Activated DGPs on a series of UCI datasets, including datasets with 308 to 45730 data points and ranging from 4 to 14 dimensions. We benchmark single and three layered DGP models and for each we have a model with standard pseudo-point inducing variables (DGP) as well as one with our activated inducing variables (ADGP). All models are implemented in Dutordoir et al. [2021] and use the Arc Cosine kernels of the first order, 512 inducing variables per layer and a constant mean function. For the deep models we configure the hidden layers with 5 output dimensions. The Activated models use a Softplus activation with $\tilde{N} = 10$. All models are optimised using Adam [Kingma and Ba, 2014] using a minibatch size of 1024 and a learning rate that starts at 0.01 which is configured to reduce by a factor of 0.9 every time the objective plateaus.

We measure accuracy of the models using Mean Squared Error (MSE) and uncertainty quantification using Test Log Likelihood (TLL) given that this is a proper scoring rule [Gneiting and Raftery, 2007]. For each dataset, we randomly select 90% of the data for training and 10% for testing and repeat the experiment

Table 1: UCI Benchmarks: Mean Squared Errors (MSEs) and Test Log Likelihood (TLL) with one standard deviation based on 5 splits. We report the performance of standard inducing point variable DGPs and DGP using activated inducing variables, respectively referred to as DGP and ADGP. We report both single layer (A)DGP-1 and three layered models (A)DGP-3.

			MSE				TLL			
Dataset	N↓	D	ADGP-1	ADGP-3	DGP-1	DGP-3	ADGP-1	ADGP-3	DGP-1	DGP-3
Yacht	308	6	0.035 (0.015)	**0.004 (0.004)**	0.282 (0.092)	0.008 (0.006)	0.289 (0.174)	0.925 (0.118)	−4.164 (1.680)	**1.017 (0.134)**
Boston	506	13	0.101 (0.022)	**0.080 (0.014)**	0.148 (0.024)	0.105 (0.012)	−0.748 (0.027)	−6.963 (2.555)	**−0.475 (0.113)**	−2.292 (0.459)
Energy	786	8	0.033 (0.011)	**0.006 (0.002)**	0.064 (0.012)	0.043 (0.009)	−0.049 (0.033)	**0.802 (0.083)**	−0.184 (0.154)	−0.449 (0.373)
Concrete	1030	8	0.195 (0.009)	**0.135 (0.018)**	0.248 (0.012)	0.157 (0.028)	**−0.623 (0.019)**	−4.416 (0.752)	−2.524 (0.295)	−4.421 (1.047)
Kin8mn	8192	8	0.086 (0.007)	**0.066 (0.003)**	0.159 (0.011)	0.080 (0.009)	−0.775 (0.016)	−1.001 (0.173)	−0.509 (0.041)	**−0.503 (0.111)**
Power	9568	4	0.058 (0.003)	**0.056 (0.003)**	0.058 (0.004)	**0.056 (0.004)**	−0.260 (0.070)	−0.104 (0.043)	**−0.069 (0.057)**	−0.099 (0.059)
Protein	45730	9	0.630 (0.011)	**0.470 (0.012)**	0.573 (0.011)	0.533 (0.033)	−1.239 (0.013)	−1.658 (0.038)	−1.141 (0.010)	**−1.104 (0.031)**

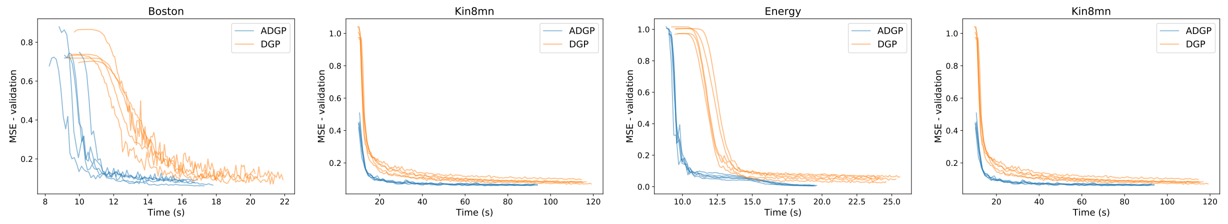


Figure 6: Traces of the MSE on the test set during the course of training for a three layer Activated DGP (ADGP) and vanilla DGP on several UCI datasets. We repeated the experiment with 5 different splits.

5 times to obtain confidence intervals. We apply an affine transformation to the input and output variables to ensure they are zero-mean and unit variance. The MSE and TLL are computed on the normalised data.

In term of accuracy, the three layered Activated DGP performs best, outperforming the shallow models and the standard DGP. We attribute this success to the ability to optimise the ADGP models using the equivalent neural network. Not only does this approach speed up training (see Fig. 6), but it enables us to initialise the $\{\boldsymbol{\mu}_\ell\}_\ell^L$ and $\{\mathbf{W}_\ell\}_{\ell=1}^L$ of the ADGP to a solution that fits the data well. The analysis of the TLL does not result in any method performing much better than others (each method there is a dataset for which it is the best performing). However, it can be seen that the deep models may leads to very poor Test Log Likelihoods (TLLs) in some instances. This is particularly true for ADGP-3.

5.3 Ablation Study: Kernel and Activated Inducing Variable Mismatch

In Section 4.2 we discussed that the stationary kernels on $\mathbb{R}^d$ are zonal kernels when restricted to $\mathbb{S}^{d-1}$. In the experiments, however, we solely made use of the Arc Cosine, for the following reasons.

Fig. 7 plots the fit (mean ± 2 standard deviations) of two models on a synthetic dataset. The data consists of 10 points that are drawn from a GP with Arc Cosine kernel of order 1, with unit length scale and variance. We corrupted the data with white Gaussian noise of variance 0.01. The models we compare are both single layer SVGPs with our activated inducing variables. We configured them to have 32 inducing variables, a Softplus activation function and set $\tilde{N} = 10$. We fixed the hyperparameters to the true values and set the variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to their optimal value using Titsias [2009]. The difference between the two models is that the left one uses the zonal Matérn-5/2 kernel, given by:

$$k(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x}||||\boldsymbol{x}'||s_{\text{mat-5/2}}\Big(\frac{\boldsymbol{x}^\top \boldsymbol{x}'}{||\boldsymbol{x}||||\boldsymbol{x}'||}\Big), \text{ with } s_{\text{mat-5/2}}(t) = \sigma^2\left(1 + \frac{\sqrt{5}t}{\rho} + \frac{5t^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}t}{\rho}\right),$$

where σ^2 is the kernel's variance and ρ is the length scale, while the model on the right uses the Arc Cosine as specified in Eq. (7).

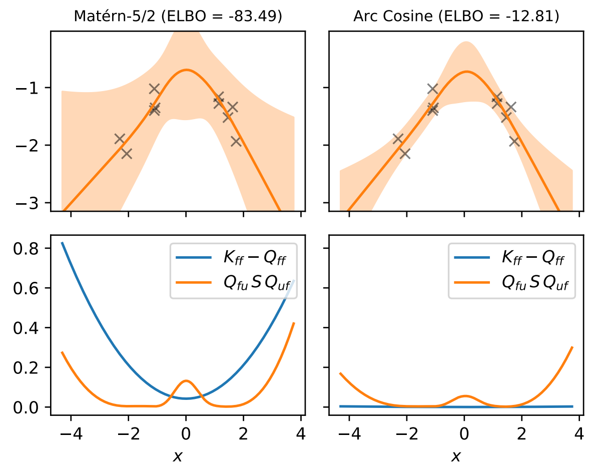


Figure 7: **(Top)** Mean $\pm$ 2 standard deviations of the Activated GP's fit on a synthetic dataset (black crosses). **(Bottom)** The two terms that constitute the standard deviation above.

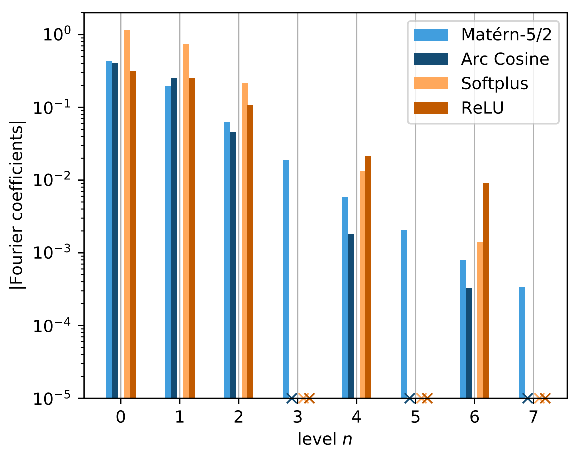


Figure 8: Spectra of the Matérn-5/2 and Arc Cosine kernels (λ_n), and Softplus and ReLU activation functions (σ_n) as a function of level. Crosses correspond to coefficients being zero.

In this experiment, we notice that the model with the Matérn-5/2 kernel overestimates its marginal predictive variance (orange shaded area) compared to the true GP regression model (see Appendix D). From the bottom panel in Fig. 7, where we plot the two terms that form this predictive variance, we see that this is caused by the large contribution of the $\mathbf{K}_{\boldsymbol{ff}} - \mathbf{Q}_{\boldsymbol{ff}}$ term. This error indicates that the Nyström approximation $\mathbf{Q}_{\boldsymbol{ff}} = \mathbf{C}_{\boldsymbol{fu}} \mathbf{C}_{\boldsymbol{uu}}^{-1} \mathbf{C}_{\boldsymbol{fu}}^\top$ is unable to approximate the prior covariance $\mathbf{K}_{\boldsymbol{ff}}$ well. In comparison, for the Arc Cosine model, we notice the fitness of the Nyström approximation to explain the prior covariance $\mathbf{K}_{\boldsymbol{ff}}$ and observe how the marginal predictive variance is now mainly due to $\mathbf{Q}_{\boldsymbol{fu}}^\top \mathbf{\Sigma} \mathbf{Q}_{\boldsymbol{fu}}$ with $\mathbf{Q}_{\boldsymbol{fu}} = \mathbf{C}_{\boldsymbol{fu}} \mathbf{C}_{\boldsymbol{uu}}^{-1}$, which stems from the uncertainty on the inducing variables, rather than from the incapability of the approximation to explain the prior variance.

On closer inspection, in Fig. 8, we see how the Matérn-5/2's spectrum contains non-zero coefficients for all spherical harmonic levels. The Softplus and ReLU activation functions, however, only have non-zero coefficients for harmonics of degree 0 and 1, followed by all even degrees [Bach, 2017]. The error in the Nyström approximation for the Matérn-5/2 model arises from the fact that we are trying to approximate a function containing all frequencies by a set of functions that is missing many of them. For the Arc Cosine kernel we notice the alignment between the kernel's and the activation functions' non-zero Fourier coefficient pattern.

6 Discussion and Conclusion

We studied the RKHS of zonal kernels on the hypersphere and the set of homogeneous functions they produce. The RKHS, defined through the spectral decomposition of the kernel, enabled us to design a so-called *activated* inducing variable. This is an inducing variable for which the resulting basis functions in the sparse variational GP are equivalent to the basis functions of a fully-connected neural network layer with a non-linear activation (e.g. ReLU and Softplus). Moreover, by hierarchically stacking these GP layers, we obtain a DGP with a posterior mean that is equivalent to a deep neural network.

In the experiments, we showed that this setup leads to DGPs that can be trained quickly and accurately, as a result of leveraging the progress made by the deep learning community in training and initialising neural network models. We also studied the properties of certain activation functions for approximating zonal kernels, and highlighted the importance of the match between the kernel's and feature's eigenvalue decay rate.

References

Francis Bach (2017). “Breaking the Curse of Dimensionality with Convex Neural Networks”. In: *Journal of Machine Learning Research.*

Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman (2019). “The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies”. In: *Advances in Neural Information Processing Systems 32 (NeurIPS).*

Alberto Bietti and Francis Bach (2020). “Deep Equals Shallow for ReLU Networks in Kernel Regimes”. In: *arXiv preprint arXiv:2009.14397.*

Alberto Bietti and Julien Mairal (2019). “On the Inductive Bias of Neural Tangent Kernels”. In: *Advances in Neural Information Processing Systems 32 (NeurIPS).*

David M Blei, Alp Kucukelbir, and Jon D McAuliffe (2017). “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association.*

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). “Weight Uncertainty in Neural Network”. In: *Proceedings of The 32nd International Conference on Machine Learning (ICML).*

Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc P. Deisenroth (2020). “Matern Gaussian processes on Riemannian manifolds”. In: *Advances in Neural Information Processing Systems 33 (NeurIPS).*

David R. Burt, Carl E. Rasmussen, and Mark van der Wilk (2019). “Rates of Convergence for Sparse Variational Gaussian Process Regression”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML).*

Lin Chen and Sheng Xu (2021). “Deep Neural Tangent Kernel and Laplace Kernel Have the Same RKHS”. In: *Proceedings of the 9th International Conference on Learning Representations (ICLR).*

Youngmin Cho and Lawrence K. Saul (2009). “Kernel Methods for Deep Learning”. In: *Advances in Neural Information Processing Systems 22 (NIPS).*

Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone (2017). “Random feature expansions for deep Gaussian processes”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML).*

Feng Dai and Yuan Xu (2013). *Approximation Theory and Harmonic Analysis on Spheres and Balls.* Springer.

Andreas Damianou and Neil D. Lawrence (2013). “Deep Gaussian Processes”. In: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Misha Denil, Babak Shakibi, Laurent Dinh, Marc\textquotesingle Aurelio Ranzato, and Nando de Freitas (2013). “Predicting Parameters in Deep Learning”. In: *Advances in Neural Information Processing Systems 26 (NIPS).*

Vincent Dutordoir, Nicolas Durrande, and James Hensman (2020a). “Sparse Gaussian Processes with Spherical Harmonic Features”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML).*

Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P Deisenroth, and ST John (2021). “GPflux: A Library for Deep Gaussian Processes”. In: *arXiv preprint arXiv:2003.01115.*

Vincent Dutordoir, Mark van der Wilk, Artem Artemev, and James Hensman (2020b). “Bayesian Image Classification with Deep Convolutional Gaussian Processes”. In: *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics (AISTATS).*

David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani (2014). "Avoiding pathologies in very deep networks". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Costas Efthimiou and Christopher Frye (2014). *Spherical Harmonics in p Dimensions.* World Scientific Publishing.

Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner (2020). "On the Expressiveness of Approximate Inference in Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems 33 (NeurIPS).*

Yarin Gal and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning Zoubin Ghahramani". In: *Proceedings of The 33rd International Conference on Machine Learning (ICML).*

Adrià Garriga-Alonso, Carl E. Rasmussen, and Laurence Aitchison (2018). "Deep Convolutional Networks as shallow Gaussian Processes". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR).*

Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, and Ronen Basri (2020). "On the Similarity between the Laplace and Neural Tangent Kernels". In: *Advances in Neural Information Processing Systems 33 (NeurIPS).*

Xavier Glorot and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Xavier Glorot, Antoine Bordes, and Yoshua Bengio (2011). "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Tilmann Gneiting and Adrian E. Raftery (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association.*

James Hensman, Nicolas Durrande, and Arno Solin (2018). "Variational Fourier Features for Gaussian Processes". In: *Journal of Machine Learning Research.*

James Hensman, Nicolo Fusi, and Neil D. Lawrence (2013). "Gaussian Processes for Big Data". In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI).*

James Hensman and Neil D. Lawrence (2014). "Nested Variational Compression in Deep Gaussian Processes". In: *arXiv preprint arXiv:1412.1370.*

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani (2015). "Scalable Variational Gaussian Process Classification". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani (2018). "Variational Bayesian dropout: pitfalls and fixes". In: *Proceedings of the 35th International Conference on Machine Learning (ICML).*

Sergey Ioffe and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of The 32nd International Conference on Machine Learning (ICML).*

Arthur Jacot, Franck Gabriel, and Clément Hongler (2018). "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *Advances in Neural Information Processing Systems 31 (NeurIPS).*

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur (2018). "Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences". In: *arXiv preprint arXiv:1807.02582.*

Durk Kingma and Jimmy Ba (2014). “Adam: A Method for Stochastic Optimization”. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Durk Kingma, Tim Salimans, and Max Welling (2015). “Variational Dropout and the Local Reparameterization Trick”. In: *Advances in Neural Information Processing Systems 28 (NIPS)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25 (NIPS)*.

Malte Kuss and Carl E. Rasmussen (2005). “Assessing Approximate Inference for Binary Gaussian Process Classification”. In: *Journal of Machine Learning Research*.

Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal (2009). “Inter-domain Gaussian Processes for Sparse Inference using Inducing Features”. In: *Advances in Neural Information Processing Systems 22 (NIPS)*.

Felix Leibfried, Vincent Dutordoir, S. T. John, and Nicolas Durrande (2020). “A Tutorial on Sparse Gaussian Processes and Variational Inference”. In: *arXiv preprint arXiv:2012.13962*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski (2018). “Measuring the Intrinsic Dimension of Objective Landscapes”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

David J. C. MacKay (1992a). “A Practical Bayesian Framework for Backpropagation Network”. In: *Neural Computation*.

David J. C. MacKay (1992b). “Bayesian Model Comparison and Backprop Nets”. In: *Advances in Neural Information Processing Systems 4 (NIPS 1991)*.

David J. C. MacKay (1998). “Choice of Basis for Laplace Approximation”. In: *Machine Learning*.

David J. C. MacKay (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani (2018). “Gaussian process behaviour in wide deep neural networks”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Radford M. Neal (1992). “Bayesian Mixture Modeling”. In: *Maximum Entropy and Bayesian Methods*.

Radford M. Neal (1995). *Bayesian Learning for Neural Networks*. Springer.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein (2019). “Bayesian deep convolutional networks with many channels are Gaussian processes”. In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Joaquin Quiñonero-Candela and Carl E. Rasmussen (2005). “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research*.

Ali Rahimi and Benjamin Recht (2008). “Random Features for Large-Scale Kernel Machines”. In: *Advances in Neural Information Processing Systems 20 (NIPS 2007)*.

Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Tim G. J. Rudner, Dino Sejdinovic, and Yarin Gal (2020). “Inter-domain Deep Gaussian Processes”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Hugh Salimbeni and Marc P. Deisenroth (2017). “Doubly Stochastic Variational Inference for Deep Gaussian Processes”. In: *Advances in Neural Information Processing Systems 30 (NIPS)*.

Julian Schrittwieser et al. (2020). “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model”. In: *Nature.*

Edward Snelson and Zoubin Ghahramani (2005). “Sparse Gaussian Processes using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems 4 (NIPS 2005).*

Arno Solin and Simo Särkkä (2020). “Hilbert space methods for reduced-rank Gaussian process regression”. In: *Statistics and Computing.*

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research.*

Shengyang Sun, Jiaxin Shi, and Roger B. Grosse (2021). “Neural Networks as Inter-Domain Inducing Points”. In: *3rd Symposium on Advances in Approximate Bayesian Inference.*

Michalis K. Titsias (2009). “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Mark van der Wilk, Matthias Bauer, S. T. John, and James Hensman (2018). “Learning Invariances using the Marginal Likelihood”. In: *Advances in Neural Information Processing Systems 31 (NeurIPS).*

Mark van der Wilk, Vincent Dutordoir, S. T. John, Artem Artemev, Vincent Adam, and James Hensman (2020). “A Framework for Interdomain and Multioutput Gaussian Processes”. In: *arXiv preprint arXiv:2003.01115.*

Mark van der Wilk, Carl E. Rasmussen, and James Hensman (2017). “Convolutional Gaussian Processes”. In: *Advances in Neural Information Processing Systems 30 (NIPS).*

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30 (NIPS).*

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q. Weinberger, and Andrew Gordon Wilson (2019). “Exact Gaussian Processes on a Million Data Points”. In: *Advances in Neural Information Processing Systems 32 (NeurIPS).*

Holger Wendland (2005). *Scattered Data Approximation.* Cambridge University Press.

Christopher K. I. Williams (1998). “Computing with Infinite Networks”. In: *Advances in Neural Information Processing Systems 11 (NIPS 1997).*

Appendix: Deep Neural Networks as Point Estimates for Deep Gaussian Processes

A Nomenclature

Table A.1: Nomenclature

indices	
$n \in \mathbb{N}$	Spherical harmonic degree, level or frequency
$j \in \{1, \dots, N_n^d\}$	Spherical harmonic orientation (indexes harmonics in a level)
$m \in \{1 \dots M\}$	inducing variables
$i \in \{1 \dots N\}$	datapoints
$\ell \in \{1 \dots L\}$	layers of a DGP
constants	
N	number of datapoints
M	number of inducing variables
P	number of outputs of the GPs
L	number of layers in a DGP
$d-1$	data input dimension, data + bias is d dimensional
$\alpha = \frac{d-2}{2}$	specifies Gegenbauer polynomial
N_n^d	number of spherical harmonics of degree n on $\mathbb{S}^{d-1}$ (see Eq. (B.3))
λ_n	eigenvalue (Fourier coefficient) of degree n for a zonal kernel
σ_n	eigenvalue (Fourier coefficient) of degree n for the activation function
$\tilde{N}$	truncation level (maximum frequency Gegenbauer polynomial in approximation)
Ω_{d-1}	surface area of $\mathbb{S}^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : \|\|x\|\|_2 = 1\}$ (see Eq. (B.2))
functions	
$\phi_{n,j}(\cdot)$	Spherical harmonic of degree n and orientation j
$C_n^{(\alpha)}(\cdot)$	Gegenbauer polynomial of degree n and specificity α
$k(\cdot,\cdot)$	kernel function
$s(\cdot)$	shape function such that for zonal kernels $k(\boldsymbol{x}, \boldsymbol{x}') = s(\boldsymbol{x}^\top \boldsymbol{x}')$
$g_m(\cdot)$	m-th inducing function
$\sigma(\cdot)$	activation function (e.g., $\max(0,t)$, softplus, swish, etc.)

B A primer on Spherical Harmonics

This section gives a brief overview of some of the useful properties of spherical harmonics. We refer the interested reader to Dai and Y. Xu [2013] and Efthimiou and Frye [2014] for an in-depth overview.

Spherical harmonics are special functions defined on a hypersphere and originate from solving Laplace's equation. They form a complete set of orthogonal functions, and any sufficiently regular function defined on the sphere can be written as a sum of these spherical harmonics, similar to the Fourier series with sines and cosines. Spherical harmonics have a natural ordering by increasing angular frequency. In Fig. B.1 we plot the first 4 levels of spherical harmonics on $\mathbb{S}^2$. In the next paragraphs we introduce these concepts more formally.

We adopt the usual L_2 inner product for functions $f : \mathbb{S}^{d-1} \to \mathbb{R}$ and $g : \mathbb{S}^{d-1} \to \mathbb{R}$ restricted to the

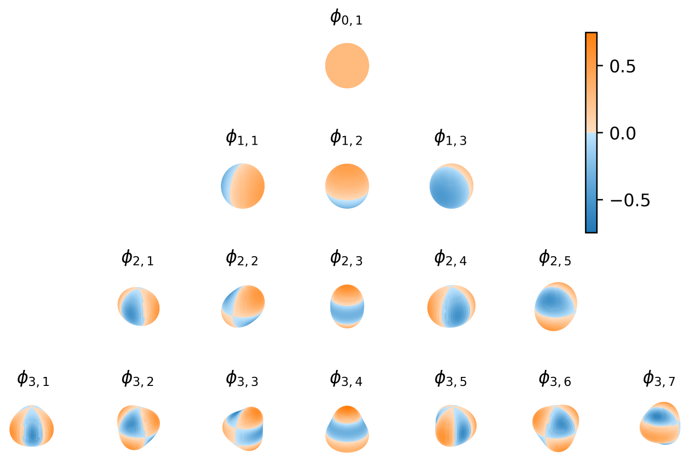


Figure B.1: Spherical Harmonics on $\mathbb{S}^2$

sphere

$$\langle f, g\rangle_{L_2(\mathbb{S}^{d-1})} = \frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(x)\, g(x)\, \mathrm{d}\omega(\boldsymbol{x}), \tag{B.1}$$

where $\mathrm{d}\omega(x)$ is the surface area measure such that Ω_{d-1} denotes the surface area of $\mathbb{S}^{d-1}$

$$\Omega_{d-1} = \int_{\mathbb{S}^{d-1}} \mathrm{d}\omega(\boldsymbol{x}) = \frac{2\pi^{d/2}}{\Gamma(d/2)}. \tag{B.2}$$

Definition 1. *Spherical harmonics of degree (or level) n, denoted as ϕ_n, are defined as the restriction to the unit hypersphere $\mathbb{S}^{d-1}$ of the harmonic homogeneous polynomials (with d variables) of degree n. It is the map $\phi_n : \mathbb{S}^{d-1} \to \mathbb{R}$ with ϕ_n a homogeneous polynomial and $\Delta\phi_n = 0$.*

For a specific dimension d and degree n there exist

$$N_n^d = \frac{2n + d - 2}{n} \binom{n + d - 3}{d - 1} \tag{B.3}$$

different linearly independent spherical harmonics on $\mathbb{S}^{d-1}$. This grows $\mathcal{O}(n^d)$ for large n. We refer to the complete set as $\{\phi_{n,j}^d\}_{j=1}^{N_n^d}$. Note that in the subsequent we will drop the dependence on the dimension d. The set is ortho-normal, which yields

$$\langle \phi_{n,j}, \phi_{n',j'}\rangle_{L_2(\mathbb{S}^{d-1})} = \delta_{nn'}\delta_{jj'}. \tag{B.4}$$

Theorem 1. *Since the spherical harmonics form an ortho-normal basis, every function $f : \mathbb{S}^{d-1} \to \mathbb{R}$ can be decomposed as*

$$f = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \widehat{f}_{n,j} \phi_{n,j}, \;\; \textit{with}\; \widehat{f}_{n,j} = \langle f, \phi_{n,j}\rangle_{L_2(\mathbb{S}^{d-1})}. \tag{B.5}$$

Which can be seen as the spherical analogue of the Fourier decomposition of a periodic function in $\mathbb{R}$ onto a basis of sines and cosines.

B.1 Gegenbauer polynomials

Gegenbauer polynomials $C_n^{(\alpha)} : [-1, 1] \to \mathbb{R}$ are orthogonal polynomials with respect to the weight function $(1 - z^2)^{\alpha-1/2}$. A variety of characterizations of the Gegenbauer polynomials are available. We use, both, the polynomial characterisation for its numerical stability

$$C_n^{(\alpha)}(z) = \sum_{j=0}^{\lfloor n/2 \rfloor} \frac{(-1)^j \, \Gamma(n-j+\alpha)}{\Gamma(\alpha)\Gamma(j+1)\Gamma(n-2j+1)} (2z)^{n-2j}, \tag{B.6}$$

and Rodrigues' formulation:

$$C_n^{(\alpha)}(z) = \frac{(-1)^n}{2^n n!} \frac{\Gamma(\alpha+\frac{1}{2})\Gamma(n+2\alpha)}{\Gamma(2\alpha)\Gamma(\alpha+n+\frac{1}{2})} (1-z^2)^{-\alpha+1/2} \frac{d^n}{dz^n}\Big[(1-z^2)^{n+\alpha-1/2}\Big]. \tag{B.7}$$

The polynomials normalise by

$$\int_{-1}^{1} \Big[C_n^{(\alpha)}(z)\Big]^2 (1-z^2)^{\alpha-\frac{1}{2}} \mathrm{d}z = \frac{\Omega_{d-1}}{\Omega_{d-2}} \frac{\alpha}{n+\alpha} C_n^{(\alpha)}(1) = \frac{\pi 2^{1-2\alpha}\Gamma(n+2\alpha)}{n!(n+\alpha)\Gamma(\alpha)^2}, \tag{B.8}$$

with $C_n^{(\alpha)}(1) = \frac{\Gamma(2\alpha+n)}{\Gamma(2\alpha)\, n!}$. Also note the following relationship $\frac{n+\alpha}{\alpha} C_n^{(\alpha)}(1) = N_n^d$.

There exists a close relationship between Gegenbauer polynomials (also known as *generalized Legendre polynomials*) and spherical harmonics, as we will show in the next theorems.

Theorem 2 (Addition)**.** *Between the spherical harmonics of degree n in dimension d and the Gegenbauer polynomials of degree n there exists the relation*

$$\sum_{j=1}^{N_n^d} \phi_{n,j}(\boldsymbol{x})\phi_{n,j}(\boldsymbol{x}') = \frac{n+\alpha}{\alpha}\, C_n^{(\alpha)}(\boldsymbol{x}^\top \boldsymbol{x}'), \tag{B.9}$$

with $\alpha = \frac{d-2}{2}$.

As a illustrative example, this property is analogues to the trigonometric addition formula: $\sin(x)\sin(x') + \cos(x)\cos(x') = \cos(x - x')$.

Theorem 3 (Funk-Hecke)**.** *Let $s(\cdot)$ be an integrable function such that $\int_{-1}^{1} \|s(t)\|(1-t^2)^{(d-3)/2}\mathrm{d}t$ is finite and $d \geq 2$. Then for every $\phi_{n,j}$*

$$\frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} s(\boldsymbol{x}^\top \boldsymbol{x}')\, \phi_{n,j}(\boldsymbol{x}')\, \mathrm{d}\omega(\boldsymbol{x}') = \lambda_n\, \phi_{n,j}(\boldsymbol{x}), \tag{B.10}$$

where $\widehat{a}_n$ is a constant defined by

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} s(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t, \tag{B.11}$$

with $\alpha = \frac{d-2}{2}$, $\omega_d = \frac{\Omega_{d-2}}{\Omega_{d-1}} = \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}$.

Funk-Hecke simplifies a $(d-1)$-variate surface integral on $\mathbb{S}^{d-1}$ to a one-dimensional integral over $[-1, 1]$. This theorem gives us a practical way of computing the Fourier coefficients for any zonal kernel.

C Analytic computation of eigenvalues for zonal functions

The eigenvalues of a zonal function are given by the one-dimensional integral:

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} s(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t, \tag{C.1}$$

where $C_n^{(\alpha)}(\cdot)$ is the Gegenbauer polynomial of degree n with $\alpha = \frac{d-2}{2}$ and $\omega_d = \Omega_{d-2}/\Omega_{d-1}$ denotes the surface area of $\mathbb{S}^{d-1}$ (see Appendix B for analytical expressions of these quantities). The shape function $s(t)$ determines whether this integral can be computed in closed-form. In the next sections we derive analytical expressions for the eigenvalues of the Arc Cosine kernel and ReLU activation function in the case the d is odd. For d even, other kernels (e.g., Matérn) or activation functions (e.g., Softplus, Swish, etc.) we rely on numerical integration (e.g., Gaussian quadrature) to obtain these coefficients. We will show that both approaches lead to highly similar results.

C.1 Arc Cosine kernel

The shape function of the first-order Arc Cosine kernel [Cho and Saul, 2009] is given by:

$$s : [0, \pi] \to \mathbb{R}, \quad s : x \mapsto \sin x + (\pi - x)\cos x, \tag{C.2}$$

where we expressed the shape function as a function of the angle between the two inputs, rather than the cosine of the angle. For notational simplicity, we also omitted the factor $1/\pi$.

Using a change of variables we rewrite Eq. (C.1)

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_0^{\pi} s(x)\, C_n^{(\alpha)}(\cos x) \sin^{d-2} x \,\mathrm{d}x, \tag{C.3}$$

Substituting $C_n^{(\alpha)}(\cos x)$ by its polynomial expansion (Eq. (B.6)), it becomes evident that we need a general solution of the integral for n, $m \in \mathbb{N}$

$$\int_0^{\pi} [\sin(x) + (\pi - x)\cos(x)] \cos^n(x) \sin^m(x) \mathrm{d}x. \tag{C.4}$$

The first term can be computed with this well-known result:

$$\int_0^{\pi} \sin^n(x)\cos^m(x)\mathrm{d}x = \begin{cases} 0 & \text{if } m \text{ odd} \\ \frac{(n-1)!!\ (m-1)!!}{(n+m)!!}\pi & \text{if } m \text{ even and } n \text{ odd,} \\ \frac{(n-1)!!\ (m-1)!!}{(n+m)!!} 2 & \text{if } n, m \text{ even.} \end{cases} \tag{C.5}$$

The second term is more cumberstone and is given by:

$$I := \int_0^{\pi} (\pi - x) \sin^n(x) \cos^m(x) \mathrm{d}x \tag{C.6}$$

which we solve using integration by parts with $u = \pi - x$ and $\mathrm{d}v = \sin^n(x)\cos^m(x)\mathrm{d}x$, yielding

$$I = u(0)v(0) - u(\pi)v(\pi) + \int_0^{\pi} v(x')\mathrm{d}x', \tag{C.7}$$

where $v(x') = \int_0^{x'} \sin^n(x)\cos^m(x)\mathrm{d}x$. This gives $v(0) = 0$ and $u(0) = 0$, simplifying $I = \int_0^{\pi} v(x')\mathrm{d}x'$.

We first focus on $v(x')$: for <u>n odd</u>, there exists a $n' \in \mathbb{N}$ so that $n = 2n' + 1$, resulting

$$v(x') = \int_0^{x'} \sin^{2n'}(x)\cos^m(x)\sin(x)\mathrm{d}x = -\int_0^{\cos(x')} (1-u^2)^{n'} u^m \mathrm{d}u \tag{C.8}$$

Where we used $\sin^2(x)+\cos^2(x)=1$ and the substitution $u=\cos(x) \implies \mathrm{d}u=-\sin(x)\mathrm{d}x$. Using the binomial expansion, we get

$$v(x')=-\int_0^{\cos(x')}\sum_{i=0}^{n'}\binom{k}{i}(-u^2)^i u^m \mathrm{d}u=\sum_{i=0}^{n'}(-1)^{i+1}\binom{k}{i}\frac{\cos(x')^{2i+m+1}-1}{2i+m+1}. \tag{C.9}$$

Similarly, for <u>m odd</u>, we have $m=2m'+1$ and use the substitution $u=\sin(x)$, to obtain

$$v(x')=\sum_{i=0}^{m'}(-1)^i\binom{k}{i}\frac{\sin(x')^{2i+n+1}}{2i+n+1}. \tag{C.10}$$

For <u>n and m even</u>, we set $n'=n/2$ and $m'=m/2$ and use the double-angle identity, yielding

$$v(x')=\int_0^{x'}\left(\frac{1-\cos(2x)}{2}\right)^{n'}\left(\frac{1+\cos(2x)}{2}\right)^{m'}\mathrm{d}x. \tag{C.11}$$

Making use of the binomial expansion twice, we retrieve

$$v(x')=2^{-(n'+m')}\sum_{i,j=0}^{n',m'}(-1)^i\binom{n'}{i}\binom{m'}{j}\int_0^{x'}\cos(2x)^{i+j}\mathrm{d}x. \tag{C.12}$$

Returning back to the original problem $I=\int_0^\pi v(x')\mathrm{d}x'$. Depending on the parity of n and m we need to evaluate:

$$\int_0^\pi \cos(x')^p \mathrm{d}x'=\begin{cases}\frac{(p-1)!!}{p!!}\pi & \text{if } p \text{ even}\\ 0 & \text{if } p \text{ odd,}\end{cases} \quad \text{or} \quad \int_0^\pi \sin(x')^p\mathrm{d}x'=\begin{cases}\frac{(p-1)!!}{p!!}\pi & \text{if } p \text{ even}\\ \frac{(p-1)!!}{p!!}2 & \text{if } p \text{ odd.}\end{cases} \tag{C.13}$$

For m and n even we require the solution to the double integral

$$\int_0^\pi\int_0^{x'}\cos(2x)^p\mathrm{d}x\mathrm{d}x'=\begin{cases}\frac{(p-1)!!}{p!!}\frac{\pi^2}{2} & \text{if } p \text{ even}\\ 0 & \text{if } p \text{ odd.}\end{cases} \tag{C.14}$$

Combining the above intermediate results gives the solution to Eq. (C.1) for the Arc Cosine kernel. In Table C.1 we list the first few eigenvalues for different dimensions and compare the analytical to the numerical computation.

C.2 ReLU activation function

Thanks to the simple form of the ReLU's activation shape function $\sigma(t)=\max(0,t)$, its Fourier coefficients can also be computed analytically. The integral to be solved is given by

$$\sigma_n=\frac{\omega_d}{C_n^{(\alpha)}(1)}\int_0^1 t\, C_n^{(\alpha)}(t)\,(1-t^2)^{\alpha-1/2}\mathrm{d}t. \tag{C.15}$$

Using Rodrigues' formula for $C_n^{(\alpha)}(t)$ in Eq. (B.7) and the identities in Eq. (B.8), we can conveniently cancel the factor $(1-t^2)^{\alpha-1/2}$. Yielding

$$\sigma_n=\omega_d\frac{(-1)^n}{2^n}\frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha+n+\frac{1}{2})}\int_0^1 t\frac{d^n}{dt^n}\left[(1-t^2)^{n+\alpha-1/2}\right]\mathrm{d}t \tag{C.16}$$

Table C.1: Eigenvalues for the first-order Arc Cosine kernel Eq. (7) computed analytically and numerically for different degrees n and dimensions d. In the experiments we set values smaller than 10^{-9} to zero.

	$d=3$		$d=5$		$d=7$	
n	numerical	analytical	numerical	analytical	numerical	analytical
0	0.375	0.375	0.352	0.352	0.342	0.342
1	0.167	0.167	0.1	0.1	0.0714	0.0714
2	0.0234	0.0234	0.00977	0.00977	0.00534	0.00534
3	−2.44e−09	−3.53e−17	1.59e−09	4.24e−17	7.79e−10	5.3e−17
4	0.000651	0.000651	0.000153	0.000153	5.34e−05	5.34e−05
5	−2.01e−09	−7.07e−17	1.86e−10	−1.01e−16	−2.11e−10	−2.52e−17
6	9.16e−05	9.16e−05	1.37e−05	1.37e−05	3.34e−06	3.34e−06
7	−1.23e−09	2.83e−16	1.53e−10	2.36e−17	−1.44e−10	−4.5e−17
8	2.29e−5	2.29e−05	2.38e−06	2.38e−06	4.26e−07	4.26e−07
9	−1.78e−10	1.7e−15	−2.19e−10	3.7e−16	3.72e−11	1.9e−16

Using integration by parts for $n \geq 2$ we can solve the integral [Bach, 2017, Appendix D]

$$\int_0^1 t \frac{d^n}{dt^n}\Big[(1-t^2)^{n+\alpha-1/2}\Big]\mathrm{d}t = \binom{n+\alpha-1/2}{k}(-1)^k(2k)! \text{ for } 2k = n-2 \tag{C.17}$$

$$= \frac{\Gamma(n+\alpha+\frac{1}{2})(-1)^{n/2-1}\Gamma(n-1)}{\Gamma(\frac{n}{2})\Gamma(\frac{n}{2}+\alpha+\frac{3}{2})} \tag{C.18}$$

Thus, substituting $\alpha = \frac{d-2}{2}$, yields

$$\sigma_n = \frac{\Gamma(\frac{d}{2})(-1)^{n/2-1}}{\sqrt{\pi}\, 2^n} \frac{\Gamma(n-1)}{\Gamma(\frac{n}{2})\Gamma(\frac{n}{2}+\frac{d+1}{2})}, \text{ for } n = 2,4,6,\dots, \tag{C.19}$$

and $\sigma_n = 0$ for $n = 3,5,7,\dots$. Finally, for $n=0$ and $n=1$, we obtain

$$\sigma_0 = \frac{1}{2\sqrt{\pi}} \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d+1}{2})}, \qquad \sigma_1 = \frac{1}{2\,(d-1)} \frac{\Gamma(\frac{d}{2})\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d-1}{2})\Gamma(\frac{d}{2}+1)}. \tag{C.20}$$

In Table C.2 we compare the analytic expression to numerical integration using quadrature. There is a close match for eigenvalues of significance and a larger discrepancy for very small eigenvalues. In practice we set values smaller than 10^{-9} to zero.

D GP Regression fit on Synthetic dataset

Table C.2: Eigenvalues for the ReLU activation Eq. (7) computed analytically and numerically for different degrees n and dimensions d. In the experiments we set values smaller than 10^{-9} to zero.

	$d=3$		$d=5$		$d=7$	
n	numerical	analytical	numerical	analytical	numerical	analytical
0	0.25	0.25	0.188	0.188	0.156	0.156
1	0.167	0.167	0.1	0.1	0.0714	0.0714
2	0.0625	0.0625	0.0313	0.0312	0.0195	0.0195
3	9.08e−10	0	5.86e−10	3.37e−17	−2.05e−10	2.69e−17
4	−0.0104	−0.0104	−0.00391	−0.00391	−0.00195	−0.00195
5	−1.54e−09	0	−2.77e−10	6.75e−17	1.27e−10	5.37e−17
6	0.00391	0.00391	0.00117	0.00117	0.000488	0.000488
7	−1.44e−09	2.83e−16	−2.38e−10	1.35e−16	−9.22e−11	0
8	−0.00195	−0.00195	−0.000488	−0.000488	−0.000174	−0.000174
9	6.6e−10	1.7e−15	1.38e−10	−8.1e−16	−1.49e−11	2.15e−15

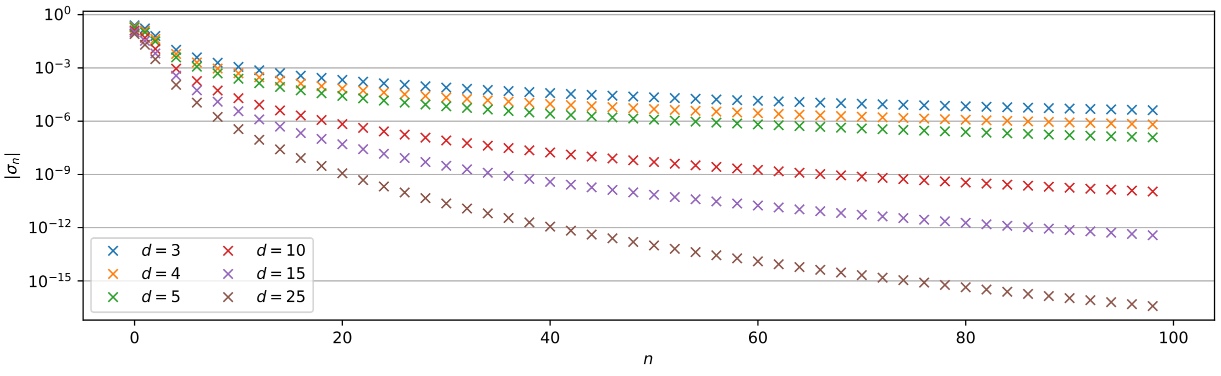


Figure C.1: ReLU coefficients σ_n as a function of degree n for different dimensions d

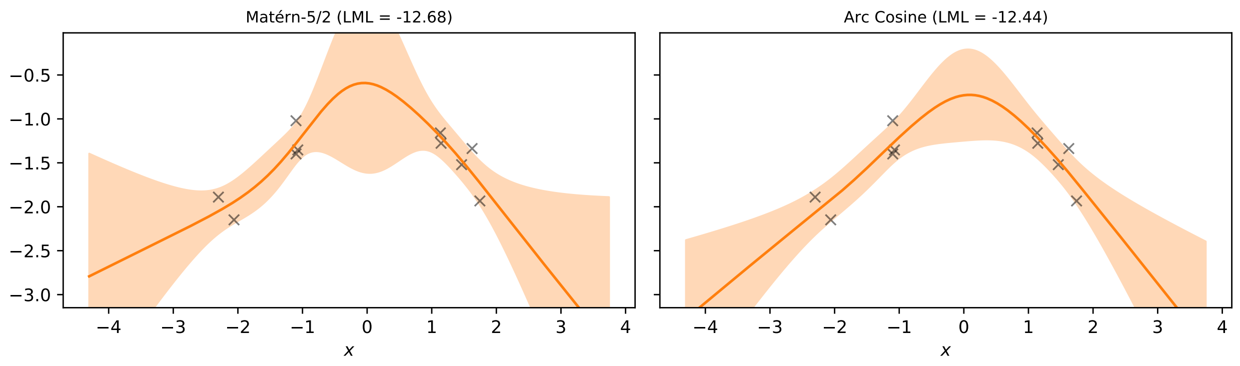


Figure D.1: Gaussian process Regression fit ($\mu \pm 2\sigma$) on synthetic dataset with corresponding Log Marginal Likelihood (LML) for a Zonal Matérn-5/2 **(left)** and Arc Cosine **(right)** kernel.