MLMI 4 Advanced Machine Learning

Matthew Ashman, James Branigan, Ionnis Tsetis and Rui Xia
Word Count: ????

1 Introduction

Gaussian processes (GPs) are a popular model for many regression and classification tasks. Utilising the Bayesian framework, they can elegantly handle the uncertainty caused by a lack of data and offer a principled method for hyperparameter tuning and model selection. Being non-parametric, they robust in settings in which the complexity of data is unknown, are able to incorporate additional data and provide good uncertainty estimates in regions in which data is sparse [Van der Wilk, 2018]. These benefits make GPs the model of choice in many problem domains, including continual learning, Bayesian optimisation and reinforcement learning [Van der Wilk, 2018, Snoek et al., 2012, Deisenroth and Rasmussen, 2011]. However, in their standard form GPs require computations which scale with $\mathcal{O}(N^3)$. This prohibits their application in the large data regime. Moreover, GPs are restricted to modelling Gaussian input-output functionality, often require intricate kernel composition Duvenaud [2014] and make no attempt to take advantage of the benefits of hierarchical feature representation [Bengio et al., 2013, Hinton, 2007].

Deep Gaussian processes (DGPs) [Damianou and Lawrence, 2013] are a recent development in which GPs are combined in a hierarchical composition. Theoretically, DGPs promise to overcome many of the limitations of standard GPs whilst retaining the advantages of being non-parametric and Bayesian. Moreover, they offer a potential solution the the failures of deep neural networks; namely, poorly-calibrated uncertainty estimates and sensitivity to directed adversarial attacks. Unfortunately, inference in DGPs is analytically intractable. In this report, we reproduce the work of Salimbeni and Deisenroth [2017]. In their paper, Doubly Stochastic Variational Inference for Deep Gaussian Processes, the authors present a novel method for performing approximate inference in DGPs. We compare the regression and classification performance of doubly stochastic inference (DSVI) for DGPs to standard GPs, and to using approximate expectation propagation (AEP) for inference in DGPs, on several benchmark experiments. Our experiments demonstrate that DGP models often exceed the performance of standard GPs, and rarely perform worse. We extend the results of Salimbeni and Deisenroth [2017], demonstrating the capability of DGPs on image completion tasks, and show that the original results can be improved upon through the use of automatic relevance determination (ARD). Further, we make novel contributions for performing approximate expectation propagation (AEP) in DGPs, extending the results of Bui et al. [2016] to multi-class classification and non-zero mean functions, and demonstrate the efficacy of our methods.

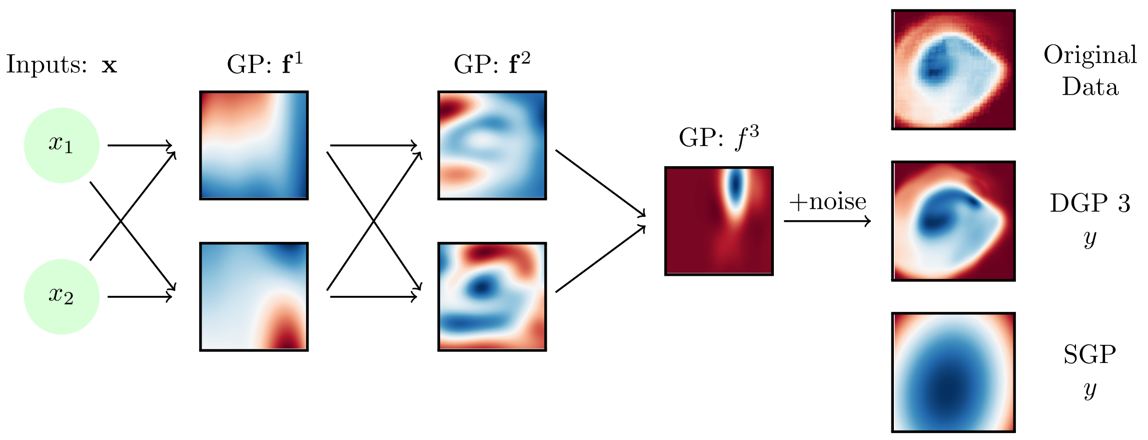


Figure 1: A 3 layer DGP is fitted to the value function of the mountain car problem. The input is two dimensional, describing the position and velocity of the car. For the first 2 layers, we use 2 independent GP functions. For the final layer, we use a single GP function to map to the one dimensional output. Whilst the functions in the first layer are simple, they have learnt to 'explain' different areas of the input space. We see that the DGP performs far better than the sparse GP on this task.

We believe that DGPs mark an important development in probabilistic machine learning. Moreover, the work of Salimbeni and Deisenroth draws upon several of the topics covered in the MLMI4 course; in particular, sparse Gaussian processes, variational inference and Bayesian neural networks (with which a comparison is made). It is for these reasons that we picked this paper. We reproduce all the experiments in the original paper, except for those involving datasets of a million of more[1].

2 Background

2.1 Gaussian Processes

A Gaussian process, $f(\mathbf{x})$, is a collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006]. It is fully specified by its mean function, $m(\mathbf{x})$, and covariance function, or kernel, $k(\mathbf{x}, \mathbf{x}')$:

$$\begin{aligned} f &\sim \mathcal{GP}(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')) \\ m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right]. \end{aligned} \tag{2.1}$$

Using this definition, the joint probability distribution of a finite collection of function values, $\mathbf{f}$, is given by

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}; m(\mathbf{X}),\ k(\mathbf{X}, \mathbf{X})) \tag{2.2}$$

where $m(\mathbf{X})_i = m(\mathbf{x}_i)$ and $k(\mathbf{X}, \mathbf{X})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Note that the mean and covariance of the outputs, $f(\mathbf{x})$, are functions of the inputs $\mathbf{x}$. Suppose we have a training set of N, D dimensional input vectors $\{\mathbf{x}_n\}_{n=1}^N$ and N scalar observations $\{y_n\}_{n=1}^N$. Typically in regression problems, we assumed that the scalar observations equal to some underlying function $f(\mathbf{x})$ with additive Gaussian noise:

$$p(y|\mathbf{x}, \sigma_n^2) = f(\mathbf{x}) + \sigma_n \epsilon, \qquad \epsilon \sim \mathcal{N}(\epsilon; 0, 1). \tag{2.3}$$

In GP regression, we specify a GP prior over the underlying function f. The complete probabilistic model of the training set is described by

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{GP}(m_\theta(\mathbf{x}), k_\theta(\mathbf{x}, \mathbf{x}')), \\ p(\mathbf{y}|\mathbf{X}, \sigma_n^2) &= \prod_{n=1}^N \mathcal{N}(\mathbf{y}; f(\mathbf{x}),\ \sigma_n^2). \end{aligned} \tag{2.4}$$

Here, θ are the mean function and kernel hyperparameters, $\mathbf{y} \in \mathbb{R}^N$ is a vector of training observations, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is a matrix of training input vectors. Since everything is Gaussian, it is straightforward to obtain analytical expressions for the predictive distribution and log marginal likelihood:

$$p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*; k(\mathbf{X}^*, \mathbf{X}) \left(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y},\ k(\mathbf{X}^*, \mathbf{X}^*) - k(\mathbf{X}^*, \mathbf{X}) \left(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} k(\mathbf{X}, \mathbf{X}^*)) \tag{2.5}$$

$$\log p(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^\top \left(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} - \frac{1}{2} \log \left|k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right| - \frac{n}{2} \log 2\pi. \tag{2.6}$$

Note that the dependence on $\mathbf{X}$, $\mathbf{X}^*$, θ and σ_n^2 has been suppressed for simplicity. We see that the first term in the log marginal likelihood is the only one that involves the observations. This term encourages the GP to fit the training data. On the other hand, the second term penalises the complexity of the model [Rasmussen and Williams, 2006]. Thus, hyperparameter selection through maximisation of the marginal likelihood is 'automatically' robust to overfitting. Whilst there are no guarantees that marginal likelihood is unimodal, in practice this is found to work well [Rasmussen and Williams, 2006].

The analytical expressions for both the predictive distribution and log marginal likelihood require the inversion of $\left(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)$. The computational complexity of this inversion scales as $\mathcal{O}(N^3)$, rendering standard GPs obsolete for large data tasks. Much of the research efforts into GPs have been concerned with reducing this computational complexity, and many excellent sparse approximations have been developed. In general, these sparse approximations can be interpreted as approximating the GP using $M < N$ 'pseudo points' $\mathbf{u}$ at pseudo-point locations $\mathbf{Z}$. One of the more popular approaches is variational inference, first introduced by Titsias [2009]. In the next section, we shall provide a brief outline of this approach before describing how it can be applied to DGPs.

[1]This was due to time and storage constraints.

2.2 Sparse Gaussian Processes

We can interpret variational inference (VI) as a method for obtaining approximations to both objects of interest in Bayesian inference: the marginal likelihood and posterior distribution. Applying VI to Gaussian processes allows us to side-step the $\mathcal{O}(N^3)$ computational cost of inverting $\left(k(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)$. We can form a lower bound to the marginal likelihood by application of Jensen's inequality:

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y},\mathbf{f},\mathbf{u})d\mathbf{f}d\mathbf{u} \geq \int q(\mathbf{f},\mathbf{u}) \log \frac{p(\mathbf{y},\mathbf{f},\mathbf{u})}{q(\mathbf{f},\mathbf{u})} d\mathbf{f}d\mathbf{u} = \mathbb{E}_{q(\mathbf{f},\mathbf{u})}\left[\log \frac{p(\mathbf{y},\mathbf{f},\mathbf{u})}{q(\mathbf{f},\mathbf{u})}\right] = \mathcal{L}_{ELBO}. \tag{2.7}$$

$\mathcal{L}_{ELBO}$ is commonly referred to as the expected lower bound (ELBO). The 'gap' between $\log p(\mathbf{y})$ and $\mathcal{L}_{ELBO}$ is simply the KL-divergence:

$$\log p(\mathbf{y}) - \mathcal{L}_{ELBO} = \mathrm{KL}\left[p(\mathbf{f},\mathbf{u}|\mathbf{y}) \| q(\mathbf{f},\mathbf{u})\right]. \tag{2.8}$$

It is now clear that finding the variational distribution $q(\mathbf{f},\mathbf{u})$ which maximises $\mathcal{L}_{ELBO}$, is equivalent to finding the variational distribution which minimises the KL-divergence $\mathrm{KL}\left[p(\mathbf{f},\mathbf{u}|\mathbf{y}) \| q(\mathbf{f},\mathbf{u})\right]$. Thus, we jointly obtain approximations the the log marginal likelihood and posterior. Following Hensman et al. [2015], we choose our variational distribution to be of the form

$$q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}), \qquad q(\mathbf{u}) = \mathcal{N}(\mathbf{u};\mathbf{m},\mathbf{S}). \tag{2.9}$$

Note that we have used the GP prior $p(\mathbf{f}|\mathbf{u})$ in place of the exact GP posterior $p(\mathbf{f}|\mathbf{u},\mathbf{y})$. The variational parameters consists of the inducing locations $\mathbf{Z}$, mean $\mathbf{m}$ and covariance $\mathbf{S}$. $\mathcal{L}_{ELBO}$ can be simplified as follows:

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(\mathbf{f},\mathbf{u})}\left[\log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})}{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})}\right] = \mathbb{E}_{q(\mathbf{f},\mathbf{u})}\left[\log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{u})}{q(\mathbf{u})}\right] = \sum_{n=1}^{N} \mathbb{E}_{q(f_n)}\left[p(y_n|f_n)\right] - \mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right], \tag{2.10}$$

where we have used the notation $f_n = f(\mathbf{x}_n)$. Whilst an analytical solution to the maximum of this lower bound with respect to the variational parameters $\mathbf{m}$ and $\mathbf{S}$ can be found, the uncollapsed bound lends itself to stochastic optimisation via Monte Carlo sampling [Hensman et al., 2015]. An approximate posterior over training set function evaluations $\mathbf{f}$ can be obtained by marginalising out the psuedo points:

$$q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{u} = \mathcal{N}(\mathbf{f};\tilde{\boldsymbol{\mu}},\tilde{\mathbf{\Sigma}}) \tag{2.11}$$

$$[\tilde{\boldsymbol{\mu}}]_n = \mu_{\mathbf{m},\mathbf{S}}(\mathbf{x}_n) = m(\mathbf{x}_n) + \boldsymbol{\alpha}(\mathbf{x}_n)^T\left(\mathbf{m} - m(\mathbf{Z})\right) \tag{2.12}$$

$$\left[\tilde{\mathbf{\Sigma}}\right]_{n,m} = \Sigma_{\mathbf{S},\mathbf{Z}}(\mathbf{x}_n,\mathbf{x}_m) = k(\mathbf{x}_n,\mathbf{x}_m) - \boldsymbol{\alpha}(\mathbf{x}_n)^T\left(k(\mathbf{Z},\mathbf{Z}) - \mathbf{S}\right)\boldsymbol{\alpha}(\mathbf{x}_m) \tag{2.13}$$

where $\boldsymbol{\alpha}(\mathbf{x}) = k(\mathbf{Z},\mathbf{Z})^{-1}k(\mathbf{Z},\mathbf{X})$. Note that the computing the approximations $\mathcal{L}_{ELBO}$ and $q(\mathbf{u})$ has a computational complexity of $\mathcal{O}(NM^2)$. For $M \ll N$, this represents a very significant computational saving.

3 Doubly Stochastic Inference for Deep Gaussian Processes

As we have already discussed, GPs have several limitations. The first of which, a computational complexity of $\mathcal{O}(N^3)$, has been the focus of attention of many researchers and has led to the development of excellent sparse approximations. However, sparse approximations do not offer solutions to the others; namely, the difficultly of hand-engineering complex covariance functions and inability to model non-Gaussian functionality. DGPs [Damianou and Lawrence, 2013] offer a potential solution to both of these problems.

Formally, we define a GP prior over a finite set of functions $\{\{f_d^l(\cdot)\}_{d=1}^{D_l}\}_{l=1}^{L}$. L is the total number of GP layers and D_l represents the number of independent GPs per layer. To simplify notation, we shall assume the outputs y_n are scalar. The extension to multidimensional outputs is trivial. The complete probabilistic model of the training set is described by,

$$\begin{aligned} p(\mathbf{F}^l|\mathbf{H}^{l-1}) &= \prod_{d=1}^{D_l} \mathcal{N}\left(\mathbf{f}^{l,d}; m^{l,d}\left(\mathbf{H}^{l-1}\right), k^{l,d}\left(\mathbf{H}^{l-1},\mathbf{H}^{l-1}\right)\right), \qquad \mathbf{H}^0 = \mathbf{X} \\ p(\mathbf{H}^l|\mathbf{F}^l) &= \prod_{d=1}^{D_l}\prod_{n=1}^{N} \mathcal{N}(h_n^{l,d}; f_n^{l,d}, \sigma_{l,d}^2) \\ p(\mathbf{y}|\mathbf{F}^L) &= \prod_{n=1}^{N} \mathcal{N}(y_n; f_n^L, \sigma_L^2). \end{aligned} \tag{3.1}$$

The inputs to each layer of the DGP are the noisy outputs from the previous layer, $\mathbf{H}^{l-1}$. We shall refer to these as 'hidden variables'. Unlike in the standard GP case, the inputs to each layer are random variables themselves and thus the mapping from input to output is no longer Gaussian. Whilst this enables us to model non-Gaussian functionality, inference becomes analytically intractable.

We shall introduce pseudo-points at each layer, $\{\mathbf{U}^l\}_{l=1}^L$, and approximate the exact posterior, $p(\{\mathbf{F}^l, \mathbf{U}^l, \mathbf{H}^l\}_{l=1}^L|\mathbf{y})$ with a variational distribution which replaces the exact posteriors $p(\mathbf{h}^{l,d}|\mathbf{f}^{l,d}, \mathbf{y})$ and $p(\mathbf{f}^{l,d}|\mathbf{u}^{l,d}, \mathbf{y})$ with the conditional $p(\mathbf{h}^{l,d}|\mathbf{f}^{l,d})$ and GP prior $p(\mathbf{f}^{l,d}|\mathbf{u}^{l,d})$, and is factorised across the pseudo-points at each layer and dimension:

$$q(\{\mathbf{F}^l, \mathbf{H}^l, \mathbf{U}^l\}_{l=1}^L) = \prod_{l=1}^{L}\prod_{d=1}^{D_l} p(\mathbf{h}^{l,d}|\mathbf{f}^{l,d})p(\mathbf{f}^{l,d}|\mathbf{u}^{l,d})q(\mathbf{u}^{l,d}), \tag{3.2}$$

where

$$q(\mathbf{u}^{l,d}) = \mathcal{N}(\mathbf{u}^{l,d}; \mathbf{m}^{l,d}, \mathbf{S}^{l,d}). \tag{3.3}$$

In the original presentation of DGP [Damianou and Lawrence, 2013], the variational distribution included a parameterised approximation to the posterior over hidden variables $q(\mathbf{h}^{l,d})$. This was memory intensive, scaling linearly with the number of datapoints and limiting the DGPs application to large datasets. Instead, although not explicitly stated by Salimbeni and Deisenroth, we choose to retain the conditional $p(\mathbf{h}^{l,d}|\mathbf{f}^{l,d})$ in place of the exact posterior $p(\mathbf{h}^{l,d}|\mathbf{f}^{l,d}, \mathbf{y})$. As noted by Bui [2018], we are concerned with obtaining good predictions on unseen data, not the accurate representations of the posterior distribution over hidden variables. The conditional also 'bakes in' posterior dependencies between hidden variables, and hidden variables and function values. With a slight abuse of notation, we shall absorb the conditional $p(\mathbf{H}^l|\mathbf{F}^l)$ into the Gaussian process kernel by adding diagonal noise, and do away with the hidden variables. This simplifies the variational approximation to

$$q(\{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L) = \prod_{l=1}^{L}\prod_{d=1}^{D_l} p(\mathbf{f}^{l,d}|\mathbf{u}^{l,d})q(\mathbf{u}^{l,d}), \tag{3.4}$$

where the new GP kernels are given by $k_{\text{new}}^{l,d}(\mathbf{x}_i, \mathbf{x}_j) = k^{l,d}(\mathbf{x}_i, \mathbf{x}_j) + \sigma_{l,d}^2\delta_{i,j}$.

Similar to the standard GP, the pseudo points can be marginalised out to obtain an approximate posterior over the function values:

$$q(\{\mathbf{F}^l\}_{l=1}^L) = \prod_{l=1}^{L}\prod_{d=1}^{D_l}\int p(\mathbf{f}^{l,d}|\mathbf{u}^{l,d})q(\mathbf{u}^{l,d})d\mathbf{u}^{l,d} = \prod_{l=1}^{L}\prod_{d=1}^{D_l}\mathcal{N}(\mathbf{f}^{l,d}; \tilde{\boldsymbol{\mu}}^{l,d}, \tilde{\boldsymbol{\Sigma}}^{l,d}) \tag{3.5}$$

$$\left[\tilde{\boldsymbol{\mu}}^{l,d}\right]_n = \mu_{\mathbf{m}^{l,d},\mathbf{S}^{l,d}}\left(\mathbf{f}_n^{l-1}\right) = m^{l,d}\left(\mathbf{f}_n^{l-1}\right) + \boldsymbol{\alpha}^{l,d}\left(\mathbf{f}_n^{l-1}\right)\left(\mathbf{m}^{l,d} - m^{l,d}\left(\mathbf{Z}^{l,d}\right)\right) \tag{3.6}$$

$$\begin{aligned}\left[\tilde{\boldsymbol{\Sigma}}^{l,d}\right]_{n,m} &= \Sigma_{\mathbf{S}^{l,d},\mathbf{Z}^{l,d}}\left(\mathbf{f}_n^{l-1}, \mathbf{f}_m^{l-1}\right) \\ &= k^{l,d}\left(\mathbf{f}_n^{l-1}, \mathbf{f}_m^{l-1}\right) - \boldsymbol{\alpha}^{l,d}\left(\mathbf{f}_n^{l-1}\right)^T\left(k^{l,d}\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right) - \mathbf{S}^{l,d}\right)\boldsymbol{\alpha}^{l,d}\left(\mathbf{f}_m^{l-1}\right)\end{aligned} \tag{3.7}$$

where $\boldsymbol{\alpha}^{l,d}\left(\mathbf{f}_n^{l-1}\right) = k^{l,d}\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}k^{l,d}\left(\mathbf{Z}^{l,d}, \mathbf{f}_n^{l-1}\right)$. We can obtain a lower bound to the log marginal likelihood in similar fashion to standard GPs. After some simplifying we obtain

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q\left(\{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L\right)}\left[\frac{p\left(\mathbf{y}, \left\{\mathbf{F}^l, \mathbf{U}^l\right\}_{l=1}^L\right)}{q\left(\left\{\mathbf{F}^l, \mathbf{U}^l\right\}_{l=1}^L\right)}\right] = \sum_{n=1}^{N}\mathbb{E}_{q(\mathbf{f}_n^L)}\left[\log p\left(y_n|\mathbf{f}_n^L\right)\right] - \sum_{l=1}^{L}\sum_{d=1}^{D_l}\mathrm{KL}\left[q\left(\mathbf{u}^{l,d}\right)\|p\left(\mathbf{u}^{l,d}\right)\right]. \tag{3.8}$$

The KL-divergence between two Gaussian can be evaluated analytically; however, unlike in standard GPs, the first term cannot. The reason for this is that computing the marginal $q(f_n^L)$ amounts to propagating Gaussian distributions through non-linear Gaussian distributions. However, although the notation is somewhat cluttered, it should be clear that the n^{th} marginal f_n^L depends only on the n^{th} marginal $\mathbf{f}_n^{L-1}$, which depends only on the n^{th} marginal $\mathbf{f}_n^{L-2}$ and so on. We can sample from $q(f_n^L)$ by simply drawing samples recursively according to

$$f_n^{l,d} \sim \mathcal{N}\left(f_n^{l,d}; \mu_{\mathbf{m}^{l,d},\mathbf{S}^{l,d}}\left(\mathbf{f}_n^{l-1}\right), \Sigma_{\mathbf{S}^{l,d},\mathbf{Z}^{l,d}}\left(\mathbf{f}_n^{l-1}, \mathbf{f}_n^{l-1}\right)\right). \tag{3.9}$$

When taking expectations of $\mathcal{L}_{ELBO}$ w.r.t the variational parameters, we can obtain a low variance gradient estimator through application of the reparameterisation trick [Kingma and Welling, 2013]. Specifically, we sample $\epsilon_n^{l,d} \sim \mathcal{N}(0,1)$ and obtain the sampled variable $f_n^{l,d}$ according to

$$f_n^{l,d} = \mu_{\mathbf{m}^{l,d},\mathbf{S}^{l,d}}\left(\mathbf{f}_n^{l-1}\right) + \Sigma_{\mathbf{S}^{l,d},\mathbf{Z}^{l,d}}\left(\mathbf{f}_n^{l-1}, \mathbf{f}_n^{l-1}\right)^{\frac{1}{2}} \times \epsilon_n^{l,d}. \tag{3.10}$$

To summarise: we have described a method to draw samples from $q(f_n^l)$, which in turn can be used to estimate $\mathcal{L}_{ELBO}$. We can perform stochastic gradient descent on our estimate to find the variational parameters, $\{\{\mathbf{m}^{l,d}, \mathbf{S}^{l,d}, \mathbf{Z}^{l,d}\}_{d=1}^{D_l}\}_{l=1}^{L}$, which maximise $\mathcal{L}_{ELBO}$ and in turn approximate the log marginal likelihood at posterior distribution. The sampling procedure described is the first source of stochasticity. The second source of stochasticity is introduced through minibatching the summation, allowing us to scale up to large datasets. We can obtain an approximate predictive distribution as a mixture of Gaussians by propagating S samples through the approximate posterior,

$$q\left(f_*^L\right) \approx \frac{1}{S}\sum_{s=1}^{S}\mathcal{N}\left(f_*^L; \mu_{\mathbf{m}^L,\mathbf{S}^L}\left(\mathbf{f}_*^{L-1^{(s)}}\right),\ \Sigma_{\mathbf{S}^L,\mathbf{Z}^L}\left(\mathbf{f}_*^{L-1^{(s)}}, \mathbf{f}_*^{L-1^{(s)}}\right)\right). \tag{3.11}$$

Samples $\mathbf{f}_*^{L-1^{(s)}}$ are drawn following equation 3.10.

Whilst it is typical to use a zero mean function [Rasmussen and Williams, 2006], this has been shown to result in pathological effects that significantly degrades performance of DGPs as the number of layers increases [Duvenaud, 2014]. Instead, we use a linear mean function $\mathbf{m}^l(\mathbf{F}^{l-1}) = \mathbf{F}^{l-1}\mathbf{W}$ for all inner GP layers, where $\mathbf{W}$ depends on the input and output dimensions, $D_{l,\text{in}}$ and $D_{l,\text{out}}$, according to,

$$\mathbf{W}^l = \begin{cases} \mathbf{I} & \text{if } D_{l,\text{in}} = D_{l,\text{out}} \\ [\mathbf{I}\ \mathbf{0}] & \text{if } D_{l,\text{in}} < D_{l,\text{out}} \\ \mathbf{V}[:D_{l,\text{out}},\ :] & \text{if } D_{l,\text{in}} > D_{l,\text{out}} \end{cases}, \tag{3.12}$$

where the columns of $\mathbf{V}$ are the right-singular vectors of $\mathbf{m}^{l-1}(\mathbf{m}^{l-2}(\ldots\mathbf{m}^1(\mathbf{X})\ldots))^2$. The mean function corresponding to each dimension is simply $m^{l,d}(\mathbf{F}^{l-1}) = \left[\mathbf{F}^{l-1}\mathbf{W}\right]_d$.

Although not explicitly stated, Salimbeni and Deisenroth [2017] use the same pseudo-point locations, $\mathbf{Z}^l$, and kernel, $k^l(\mathbf{x}_i, \mathbf{x}_j)$, for all dimensions. Whilst this simplifies the notation significantly, our derivation is more general. The use of independent pseudo-point locations and kernels across each dimension increases the computational complexity from $\mathcal{O}(LNM^2)$ to $\mathcal{O}(LDNM^2)$, where D is number number of dimensions per hidden layer. Preliminary experiments on UCI regression tasks suggested that this significant increase in computational complexity only resulted in slight improvements in performance. Thus, this modification was not explored further. Another modification proposed by Cheng and Boots [2017] is to decouple the inducing points used to parameterise the mean and covariance of each GP. This has been shown to improve performance and computational complexity in DGPs [Havasi et al., 2018a]; however, it was decided not to explore this modification in this report.

3.1 Extension: Approximate Expectation Propagation for Deep Gaussian Processes

An alternative method for approximate inference in DGPs is approximate expectation propagation (AEP) [Bui et al., 2016]. Whilst we shall omit a complete derivation of the method in this report, it is important to note that AEP for DGPs uses the Fully Independent Training Conditional (FITC) approximation in which the exact model is replaced with an approximation [Quiñonero-Candela and Rasmussen, 2005]. Although both the VI approach of Titsias [2009] and and FITC can be interpreted as performing approximate inference on the exact model [Bui et al., 2017a], there are important differences between the two approaches. VI obtains a true lower bound to the marginal likelihood and recovers the true posterior if possible; however, it is not trivial to optimise $\mathcal{L}_{ELBO}$ - it can be hindered by local optima, requiring careful initialisations of psuedo-point locations and hyperparameters. Moreover, $\mathcal{L}_{ELBO}$ may not be uniformly tight [Turner et al., 2011]. On the other hand, FITC is easier to optimise but may severely under-estimate the noise variance and ignore the addition of more pseudo-points [Van der Wilk, 2018].

3.1.1 Multi-class Classification

The authors of the original paper only consider the binary classification setting. In the multi-class setting, observations $\{y_n\}_{n=1}^N$ take discrete values $y_n \in \{1, \ldots, C\}$ with corresponding one-hot encoded labels $\bar{\mathbf{y}}_n \in$

[2] Although stated in their report that left-singular vectors are used, this was not the case in their implementation.

$\{0,1\}^C$. The likelihood function is given by $p(y_n|\mathbf{f}_n^L) = \prod_c \pi_c(f_{n,c}^L)^{\bar{y}_{n,c}}$, where

$$\pi_c(\mathbf{f}_n^L) = \begin{cases} 1-\epsilon & \text{if } c = \arg\max_c f_{n,c}^L \\ \epsilon/(C-1) & \text{otherwise} \end{cases}. \tag{3.13}$$

ϵ can be chosen to reflect the number of incorrect labels in the training dataset. At the final stage of the forward pass of probabilistic backpropagation, we are tasked with computing

$$\begin{aligned} Z_n &\approx \int p\left(y_n|\mathbf{f}_n^L\right)\mathcal{N}\left(\mathbf{f}_n^L;\mathbf{m}_n^L,\mathbf{S}_n^L\right)d\mathbf{f}_n^L \\ &= \int \left[p\left(\arg\max_c f_{n,c}^L = y_n\right)(1-\epsilon) + \left(1-p\left(\arg\max_c f_{n,c}^L = y_n\right)\right)\frac{\epsilon}{C-1}\right]\mathcal{N}\left(\mathbf{f}_n^L;\mathbf{m}_n^L,\mathbf{S}_n^L\right)d\mathbf{f}_n^L \\ &= \int \left[\prod_{c\neq y_n} p\left(f_{n,y_n}^L \geq f_{n,c}^L\right)\left(1-\frac{C\epsilon}{C-1}\right)+\frac{\epsilon}{C-1}\right]\mathcal{N}\left(\mathbf{f}_n^L;\mathbf{m}_n^L,\mathbf{S}_n^L\right)d\mathbf{f}_n^L. \end{aligned} \tag{3.14}$$

Using Gauss-Hermite quadrature, we can approximate this integral as

$$Z_n \approx \frac{1}{\sqrt{\pi}}\sum_{k=1}^K w_k \prod_{c\neq y_n}\left[\phi\left(\frac{\sqrt{2v_{n,y_n}^L x_k} + m_{n,y_n}^L - m_{n,c}^L}{v_{n,c}^L}\right)\left(1-\frac{C\epsilon}{C-1}\right)+\frac{\epsilon}{C-1}\right], \tag{3.15}$$

where K is the number of samples points and x_k are the roots of the physicists' version of the Hermite polynomial with corresponding weights w_k [Liu and Pierce, 1994]. Note that we have used the notation $v_{n,c}^L = \left[\mathbf{S}_n^L\right]_{c,c}$. Computation of the gradients of $\log Z_n$ are as follows:

$$\left[\frac{d\log Z_n}{dm_n^L}\right]_{c\neq y_n} = \frac{1}{Z_n}\frac{1}{\sqrt{2\pi}}\left(1-\frac{C\epsilon}{C-1}\right)\left(-\frac{1}{v_{n,c}^L}\right)\sum_{k=1}^K w_k \exp\left(-\frac{1}{2}\left(\frac{\tilde{x}_{kn}-m_{n,c}^L}{v_{n,c}^L}\right)^2\right)\prod_{c'\neq y_n,c}\phi\left(\frac{\tilde{x}_{kn}-m_{n,c'}^L}{v_{n,c'}^L}\right) \tag{3.16}$$

$$\left[\frac{d\log Z_n}{dm_n^L}\right]_{y_n} = \frac{1}{Z_n}\frac{1}{\sqrt{2\pi}}\left(1-\frac{C\epsilon}{C-1}\right)\sum_{k=1}^K w_k \sum_{c\neq y_n}\left(-\frac{1}{v_{n,c}^L}\right)\exp\left(-\frac{1}{2}\left(\frac{\tilde{x}_{kn}-m_{n,c}^L}{v_{n,c}^L}\right)^2\right)\prod_{c'\neq y_n,c}\phi\left(\frac{\tilde{x}_{kn}-m_{n,c'}^L}{v_{n,c'}^L}\right) \tag{3.17}$$

where we have used $\tilde{x}_{kn} = \sqrt{2v_{n,y_n}^L x_k} + m_{n,y_n}^L$.

3.2 Linear Mean Function

As discussed, the use of a zero mean function results in pathological effects as the number of GP layers is increased. To ensure a fair comparison between AEP and DSVI, we felt it necessary to implement the same strategy for avoiding such effects. In the forward pass of probabilistic backpropagation, the conditional mean for datapoint n, layer l and dimension d, $m_{l|\mathbf{f}_n^{l-1}}^d$, is given by

$$m_{l|\mathbf{f}_n^{l-1}}^d = \left(\mathbf{f}_n^{l-1}\right)^T \mathbf{W}_d^l + k\left(\mathbf{f}_n^{l-1},\mathbf{Z}^{l,d}\right)k\left(\mathbf{Z}^{l,d},\mathbf{Z}^{l,d}\right)^{-1}\left(\mathbf{m}_l^{\setminus 1} - \mathbf{Z}^{l,d}\mathbf{W}_d^l\right), \tag{3.18}$$

where $\mathbf{W}_d^l$ is the d^{th} column of $\mathbf{W}^l$ given in equation 3.12. The conditional variance $v_{l|\mathbf{f}_n^{l-1}}^d$ is unchanged. Following Bui et al. [2016], the mean and variance of the Gaussian approximation, $m_{n,d}^l$ and $v_{n,d}^l$, is obtained using the law of iterated conditional expectations:

$$\begin{aligned} m_{n,d}^l &= \mathbb{E}_{q(\mathbf{f}_n^{l-1})}[m_{l|\mathbf{f}_n^{l-1}}^d] \\ &= \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\left(\mathbf{f}_n^{l-1}\right)^T\mathbf{W}_d^l + k\left(\mathbf{f}_n^{l-1},\mathbf{Z}^{l,d}\right)\underbrace{k\left(\mathbf{Z}^{l,d},\mathbf{Z}^{l,d}\right)^{-1}\left(\mathbf{m}_l^{\setminus 1}-\mathbf{Z}^{l,d}\mathbf{W}_d^l\right)}_{\mathbf{a}}\right] \\ &= \left(\mathbf{m}_n^{l-1}\right)^T\mathbf{W}_d^l + \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k\left(\mathbf{f}_n^{l-1},\mathbf{Z}^{l,d}\right)\mathbf{a}\right] \\ &= \left(\mathbf{m}_n^{l-1}\right)^T\mathbf{W}_d^l + \psi_{n,0}^l\mathbf{a} \end{aligned} \tag{3.19}$$

$$
\begin{aligned}
v_{n,d}^l &= \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\left[v_{l|\mathbf{f}_n^{l-1}}^d\right] + \mathrm{Var}_{q(\mathbf{f}_n^{l-1})}\left[m_{l|\mathbf{f}_n^{l-1}}^d\right]\right. \\
&= \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\sigma_{l,d}^2 + k\left(\mathbf{f}_n^{l-1}, \mathbf{f}_n^{l-1}\right)\right. \\
&\qquad \left. - k\left(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d}\right)\left(k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1} - k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}\mathbf{S}_l^{\setminus 1}k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}\right)k\left(\mathbf{Z}^{l,d}, \mathbf{f}_n^{l-1}\right)\right] \\
&\quad + \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\left(\left(\mathbf{f}_n^{l-1}\right)^T\mathbf{W}_d^l + k\left(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d}\right)k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}\left(\mathbf{m}_l^{\setminus 1} - \mathbf{Z}^{l,d}\mathbf{W}_d^l\right)\right)^2\right] - \left(m_{n,d}^l\right)^2 \\
&= \sigma_{l,d}^2 + \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k\left(\mathbf{f}_n^{l-1}, \mathbf{f}_n^{l-1}\right)\right] + \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\left(\left(\mathbf{f}_n^{l-1}\right)^T\mathbf{W}_d^l\right)^2\right] \\
&\quad + 2\mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\left(\mathbf{f}_n^{l-1}\right)^T\mathbf{W}_d^l k\left(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d}\right)\mathbf{a}\right] \\
&\quad + \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k\left(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d}\right)\mathbf{B}k\left(\mathbf{Z}^{l,d}, \mathbf{f}_n^{l-1}\right)\right] - \left(m_{n,d}^l\right)^2 \\
&= \sigma_{l,d}^2 + \psi_{n,0}^l + \psi_{n,3}^l + \psi_{n,4}^l\mathbf{a} + \mathrm{tr}\left(\mathbf{B}\psi_{n,2}^l\right) - \left(m_{n,d}^l\right)^2 \qquad (3.20)
\end{aligned}
$$

where we have used

$$
\mathbf{B} = k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}\left(\mathbf{S}_l^{\setminus 1} + \left(\mathbf{m}_l^{\setminus 1} - \mathbf{Z}^{l,d}\mathbf{W}_d^l\right)\left(\mathbf{m}_l^{\setminus 1} - \mathbf{Z}^{l,d}\mathbf{W}_d^l\right)^T\right)k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1} - k\left(\mathbf{Z}^{l,d}, \mathbf{Z}^{l,d}\right)^{-1}. \qquad (3.21)
$$

Bui et al. [2016] provide derivations for $\psi_{n,0}^l = \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k(\mathbf{f}_n^{l-1}, \mathbf{f}_n^{l-1})\right]$, $\psi_{n,1}^l = \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d})\right]$ and $\psi_{n,2}^l = \mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[k(\mathbf{Z}^{l,d}, \mathbf{f}_n^{l-1})k(\mathbf{f}_n^{l-1}, \mathbf{Z}^{l,d})\right]$. $\psi_{n,3}^l$ is readily computed as

$$
\begin{aligned}
\psi_{n,3}^l &= \left(\mathbf{W}_d^l\right)^T\mathbb{E}_{q(\mathbf{f}_n^{l-1})}\left[\mathbf{f}_n^{l-1}\left(\mathbf{f}_n^{l-1}\right)^T\right]\mathbf{W}_d^l \\
&= \left(\mathbf{W}_d^l\right)^T\left(\mathbf{S}_n^{l-1} + \mathbf{m}_n^{l-1}\left(\mathbf{m}_n^{l-1}\right)^T\right)\mathbf{W}_d^l. \qquad (3.22)
\end{aligned}
$$

Computation of $\psi_{n,4}^l$ is more involved. Fortunately, if a squared exponential kernel with independent lengthscales, α_q, per dimension is used then this is analytically tractable. The m^{th} component of $\psi_{n,4}^l$ is given by

$$
\begin{aligned}
(\psi_{n,4}^l)_m &= \sigma_f^2\int \exp\left(-\frac{1}{2}\sum_{q=1}^{D_{l-1}}\frac{(f_{n,q}^{l-1} - z_{m,q}^{l,d})^2}{\alpha_q}\right) \times \sum_{q=1}^{D_{l-1}} f_{n,q}^{l-1}w_{dq}^l \times \mathcal{N}\left(\mathbf{f}_n^{l-1}; \mathbf{m}_n^{l-1}, \mathbf{S}_n^{l-1}\right)d\mathbf{f}_n^{l-1} \\
&= \sum_{q=1}^{D_{l-1}} w_{dq}^l\left(\psi_{n,1}^l\right)_m\frac{z_{m,q}^{l,d}v_{n,q}^{l-1} + m_{n,q}^{l-1}\alpha_q}{v_{n,q}^{l-1} + \alpha_q}. \qquad (3.23)
\end{aligned}
$$

4 Experimental Setup

The original implementation is freely available online at `https://github.com/ICL-SML/Doubly-Stochastic-DGP`; however, it was decided to re-implement the code in GPflow 2.0 De G. Matthews et al. [2017] to consolidate our own understanding and take advantage of the benefits of Tensorflow 2.0. Our implementation, and the code necessary to run all experiments, can be found at `https://github.com/MattAshman/MLMI4`. In all experiments, we use $\min(30, D^0)$ dimensions for all inner layers of the DGP models, where D^0 is the inputs dimension, and the squared exponential kernel with a lengthscale for each dimension for all layers. Such a kernel implements automatic relevance determination (ARD) [Neal, 1994].

In all cases, we evaluate the predictive log likelihood and root mean squared error on the held out test data, propagating 100 samples to estimate the predictive distribution as described in equation 3.11. For training, we use Adam [Kingma and Ba, 2014] with default parameters and a learning rate of 0.01. In all experiments, pseudo-point locations were initialised using the k-means clustering algorithm. All lengthscales and variances were initialised to 1. We compare the performance of doubly stochastic variational inference for DGPs (DSVI) to sparse variational Gaussian processes (SGPs) with 100 and 500 pseudo points and squared exponential kernels and Bayesian neural networks (BNNs) with ReLu activations, 50 hidden units (100 for protein and year), with inference by probabilistic backpropagation [Hernández-Lobato and Adams, 2015][3].

To compare the performance of DSVI with AEP for inference in DGPs, we modified the original code `https://github.com/thangbui/geepee` to include the results in Section 3.1. Whilst Salimbeni and Deisenroth [2017] evaluate the performance of AEP for DGPs using the same model architecture as DSVI for DGPs, we limit the model to 2 dimensions for all inner layers and 100 pseduo-points, taking advantage of the initialisation scheme proposed by Bui et al.. Unlike Salimbeni and Deisenroth, Bui et al. use independent pseudo-points and kernels across dimensions. As the number of dimensions is kept small, this is not too heavy a computational burden. Many of the training details are omitted from Bui et al. [2016]; however, an earlier paper [Bui et al., 2015] states that 4000 iterations and a minibatch size of 50 is used for training on UCI regression tasks. Whilst, the authors use Bayesian optimisation for selecting the learning rate of Adam, we use the same fixed learning rate of 0.01. We believe that this allows for a fairer comparison between the approximate inference methods. For further implementation details, see [Bui et al., 2016, 2015].

5 Results

5.1 Regression

We evaluated the performance of 2, 3, 4 and 5 layer DGPs, each with 100 pseudo-points per layer, on 9 benchmark UCI regression datasets with size ranging from 506 to 515345 and input dimension ranging from 4 to 90. For all but the year dataset, we perform 20-fold cross validation using a 90/10 train/test split. At each stage of cross-validation, the DGP is trained for 20000 iterations using a minibatch size of $\min(10000, |\mathcal{D}_{train}|)$. For the year dataset, we evaluate the performance on a single split with $|\mathcal{D}_{train}| = 463,715$ and $|\mathcal{D}_{test}| = 51,630$, as to prevent songs from a given artist ending up in both the train and test set, and train the DGP for 100000 iterations using a minibatch size of 20000.

Figures 2 and 3 plot the mean $\pm$ standard deviation predictive log likelihood and RMSE, respectively, of all models on the 8 small to medium sized UCI regression datasets. On 'wine_red', the DGP recovers the SGP with 100 pseudo points. As noted by the original authors, this dataset is hardly a challenge for a GP as it is near-linear. DGPs exceed the performance of the SGP with 100 pseudo points on all other datasets, and exceed the performance of the SGP with 500 pseudo points on all other datasets exept for 'naval'[4]. This demonstrates the ability of DGPs to handle more complex data than standard GPs, owing to its increased flexibility. The results indicate that by increasing the number of layers, the performance of DGPs improves - that is, overfitting does not occur in the DGP model, even on small datasets. However, as the computational complexity scales with $\mathcal{O}(LNM^2)$, these performance improvements come at a significant computational cost. We found that for all GP models, our results exceed those reported by the original authors. If a single lengthscale was used across all dimensions (i.e. no ARD), the original results are recovered. Indeed, inspection of the original code suggests that the authors forgot to include ARD lengthscales in their experiments. This was confirmed with Hugh through email [Salimbeni, 2020].

Interestingly, all experimental results had significantly greater variance than reported by Salimbeni and Deisenroth. A potential source of this additional stochasticity in the DSVI implementation is the number of samples used to estimate $\mathcal{L}_{ELBO}$ for each training point, and the number of samples used to estimating the predictive distribution in equation 3.11. Whilst we used a single training sample for $\mathbb{E}_{q(\mathbf{f}_n^L)}\left[\log p\left(y_n|\mathbf{f}_n^L\right)\right]$ and 100 test samples for the predictive distribution, Salimbeni and Deisenroth did not explicitly state the number of samples they used. It is likely that they used more.

Comparing the performance of DSVI with AEP for inference in DGPs, we see that using AEP for inference in a DGP with 2 layers performs worse than DSVI on all but 'boston'. Bui et al. [2016] obtain significant improvements in performance through increasing the number of hidden layers. However, we found these results too computationally prohibitive to obtain. Finally, it is clear that the BNN performs worse than all DGP models on each of the datasets, and only performs better than the SGP models on 'energy'.

Table 1 compares the performance of all models on the 'year' dataset. Similar to the other UCI regression tasks, the DGPs outperform all SGP models; however, we see that increasing the depth of the DGP has no effect on its performance. Again, the results obtained here exceed those obtained by Salimbeni and Deisenroth.

[3]Note that these results are taken directly from the original paper.

[4]On the 'naval' dataset, all GP models achieved a remarkably low predictive RMSE - less than 0.001.

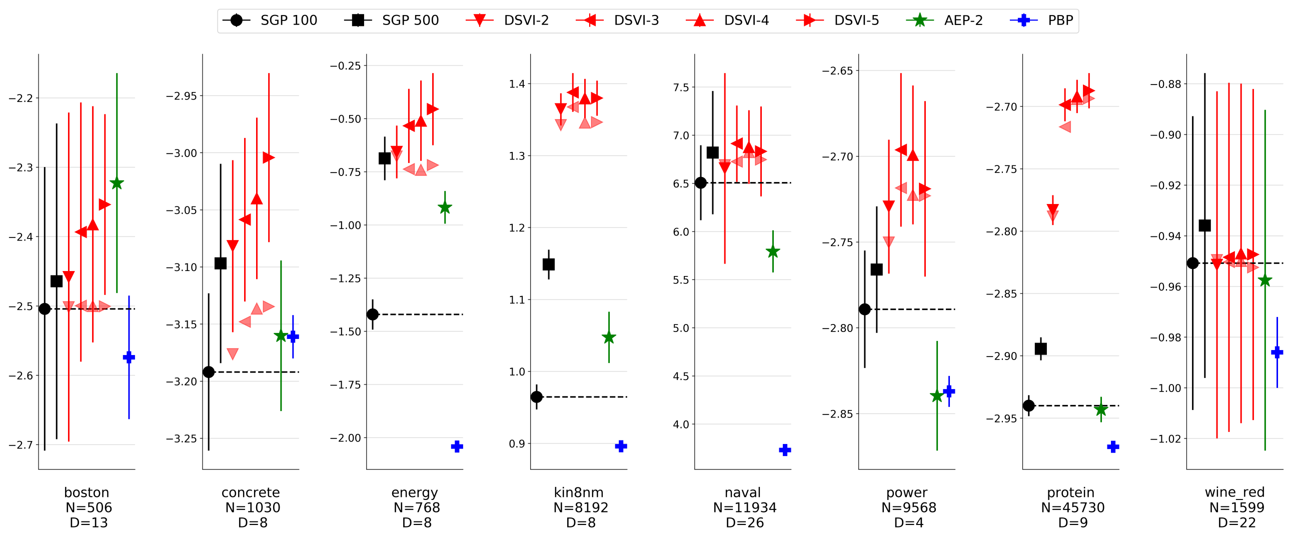


Figure 2: Average predictive log likelihood for the SGP, BNN and two approaches for inference in DGPs across 8 UCI regression tasks. The higher the better. The partially coloured red triangles show the mean predictive log likelihood without ARD. The complete results are provided in the Appendix.

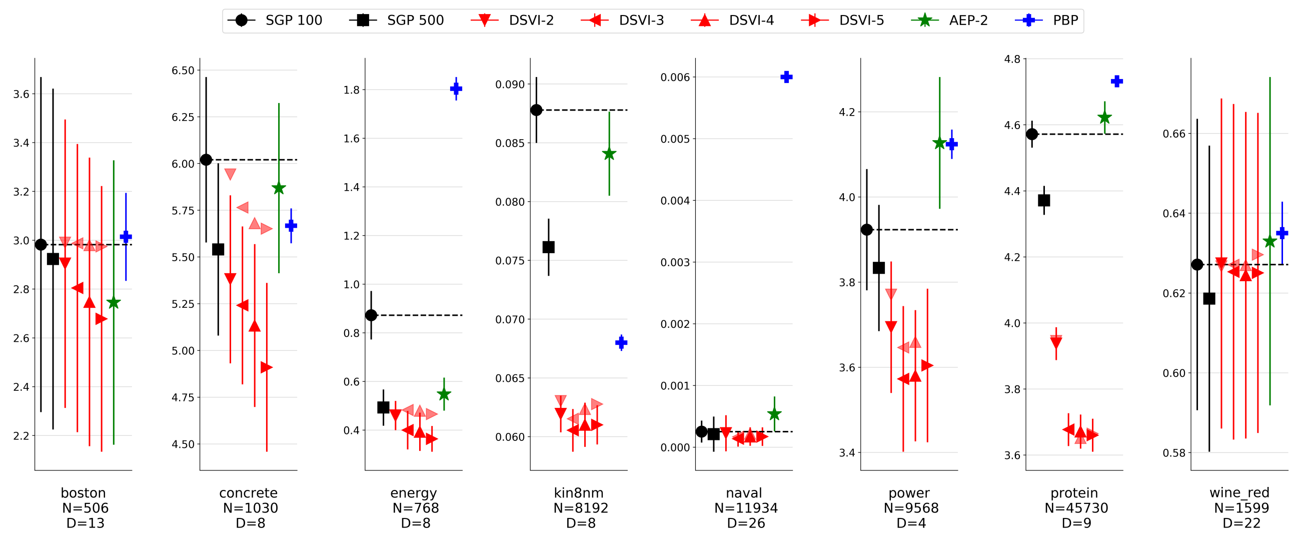


Figure 3: Average predictive RMSE for the SGP, BNN and two approaches for inference in DGPs across 8 UCI regression tasks. The lower the better. The partially coloured red triangles show the mean predictive log likelihood without ARD. The complete results are provided in the Appendix.

year	SGP 100	SGP 500	DSVI-2	DSVI-3	DSVI-4	DSVI-5	AEP-2	PBP
Log-likelihood	-3.647	-3.644	-3.581	-3.573	-3.568	-3.571	**-3.563**	-3.603
RMSE	9.268	9.237	8.752	8.752	**8.743**	8.746	9.083	8.879

Table 1: Predictive log likelihood and RMSE for the large scale 'year' dataset (N=515345, D=90).

5.2 Classification

We evaluated the performance on two classification datasets - Rectangle-Images[5] and MNIST. The Rectangle-Images dataset consists of a collection of rectangles displayed against a background. The task is to detect whether the height or width of the rectangle is greater. The dataset consists of 12000 training images and 50000 test images, each of size 28×28. The MNIST dataset consists of a collection of hand-written digits. The

[5]http://www.iro.umontreal.ca/ lisa/icml2007data/rectangles_images.zip

task is to correctly classify the images into the 10 categories 0-9. The dataset consists of 60000 training images and 10000 test images, each of size 28×28. We compare the performance of 2 and 3 layer DGPs, each with 100 inducing points using the probit likelihood and robust maximum likelihood[6] for the binary and multiclass settings, respectively. We train each DGP model for 5000 iterations using a batch size of 1000 and use $K = 20$.. Table 2 and 3 compare the classification results of all models.

Rectangle	**SGP 100**	**SGP 500**	**DSVI-2**	**DSVI-3**	**AEP-2**
Log-likelihood	−0.6575	−0.6541	−0.4646	**−0.4620**	−0.4815
Accuracy (%)	72.12	72.87	**77.51**	77.48	75.19

Table 2: Predictive log likelihood and accuracy for the DGP and two approaches for inference in DGPs for the Rectangle-Images binary classification dataset.

MNIST	**SGP 100**	**SGP 500**	**DSVI-2**	**DSVI-3**	**AEP-2**
Log-likelihood	−0.2807	−0.2623	−0.0782	**−0.0729**	−0.1294
Accuracy (%)	92.26	92.88	97.73	**97.99**	96.46

Table 3: Predictive log likelihood and accuracy for the DGP and two approaches for inference in DGPs for the MNIST multiclass classification dataset.

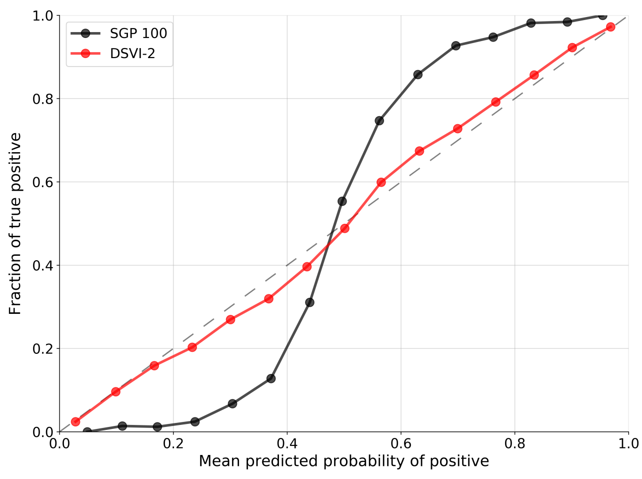
The results show that the use of DGPs, regardless of the form of inference, significantly outperform standard GP models on image classification tasks. This reflects the increased flexibility of the DGPs enabling it to handle much complex data, such as images, and supports the hypothesis that DGPs are able to 'automatically' construct complex kernels that are well suited for the data at hand.. Similar to the regression results, we found that the use of AEP for DGPs did not improve classification results.

Interestingly, we found that for the Rectangle-Images task when using DSVI for DGPs, all models achieved very similar performance when the number of hidden dimensions is as low as 2 and as high as 30. This was not true for the MNIST task, in which the DGP performed significantly better as the number of hidden dimensions is increased from 2 to 30. We note that in the Rectangle-Images task, the important features consist of 2 straight edges that are perpendicular to each other. In contrast, for the MNIST task there are many important visual features which must be used to correctly classify images. This suggests that each dimension of each DGP layer can be interpreted as extracting individual features of the input data. In fact, the Rectangle-Images dataset was specifically constructed to distinguish between models which perform hierarchical feature extraction and those that do not [Larochelle et al., 2007]. The fact that DGPs significantly outperforms standard GPs indicates its capcity to perform hierarchical feature extraction.

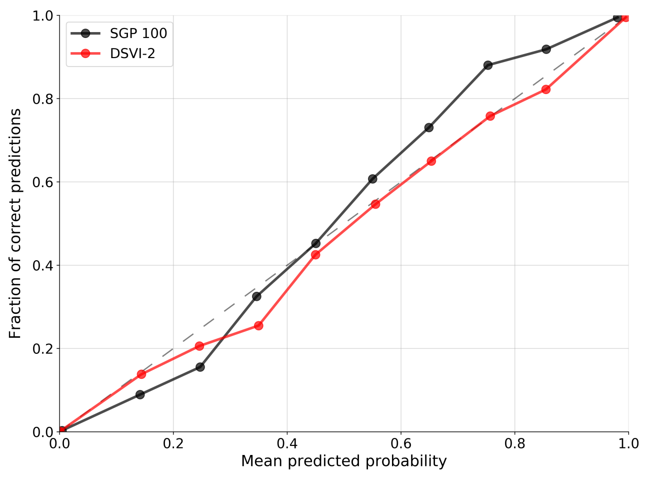
Figure 4 compares the calibration curves for the SGP and DGP on both classification tasks. Whilst the SGP is under-confident in it's uncertainty estimates (especially on the Rectangle-Images task), the DGP's uncertainty estimates are very well calibrated - that is, it's uncertainty predictions agree with empirical results.

5.3 Image Completion

[6]Using $K = 20$ samples for estimating the Z_n using Gauss-Hermite quadrature for AEP.



(a)



(b)

Figure 4: Calibration curves for a) Rectangle-Images binary classification task and b) MNIST multiclass classification task.

6 Conclusion

In this report, we re-implement doubly stochastic variational inference for deep Gaussian processes [Salimbeni and Deisenroth, 2017]. We outperform the results presented in the original paper through the use of automatic relevance determination, and demonstrate the superior capability of DGPs relative to both standard Gaussian processes and Bayesian neural networks 7 out of 9 small to large scale regression. Whilst our method uses very few hyperparameters and simple kernel structures, we show that DGPs far exceed the performance of sparse GPs on complex image classification, achieving an accuracy of 97.99% and 77.51% compared to 92.88% and 72.87% on the MNIST multiclass classification and the Rectangle-Images binary classification datasets, respectively. Moreover, we find that DGPs have much better calibrated uncertainty estimates than sparse GPs. The results support the hypothesis that DGPs are able to ‘automatically’ learn to construct complex kernels that are well suited for the data being modelled.

Our results suggest that by increasing the number of layers, DGPs offer significant performance improvements upon standard GPs even when far fewer pseudo points are used. Moreover, the computational complexity for our method scales with $\mathcal{O}(LNM^2)$, implying that increasing depth as opposed to the number of pseudo point not only improves performance, but also computational efficiency. Importantly, we found that DGPs do not suffer from overfitting unlike their DNN counterparts, even on datasets with less than 500 training examples.

7 Critical Evaluation

Need to write something here...

7.1 Limitations

Despite these impressive results, there are several limitations of doubly stochastic variational inference for DGPs:

Handling multi-modal posteriors : unlike in standard GPs, the true posterior distribution is not Gaussian, nor is it factorised between layers. In fact, Havasi et al. [2018b] demonstrates that the true posterior distribution is multi-modal. The Gaussian variational approximation is not capable of modelling such multi-modality and will inevitably provide a poor approximation. We should look to using alternative variational distributions or alternative approximate inference methods to overcome this limitation. Indeed, Havasi et al. demonstrate efficacy of Hamiltonian Monte Carlo for inference in DGPs.

Handling high dimensional data : in this report, we evaluated the performance of DGPs on datasets of input dimension up to 728. Scaling GP models, including DGPs, to high dimensional datasets requires the use of kernels that reflect invariances within datasets. For example, the use of convolutional kernels significantly improves the performance of GPs on image classification tasks [Van der Wilk et al., 2017]. Whilst DGPs have demonstrated the ability to ‘automatically’ construct kernels suitable for the data at hand, they do not provide a principled framework for learning invariances within the data. This limits their applicability to high dimensional datasets.

7.2 Future Work

In addition to implementing the regression and classification experiments on datasets with more than a million data points, had we more time we would have enjoyed exploring some of the following avenues:

Modelling correlated outputs : in this approach, GPs within each layer were modelled as independent of each other. This prevents DGPs from exploiting dependencies between outputs, limiting their predictive performance on highly correlated data. There have been several attempts to develop GPs which model such dependencies [Requeima et al., 2019], and they make for a natural extension to the current DGP framework.

Applications : for applications in which handling uncertainty is crucial, we believe that DGPs have the ability to achieve state-of-the-art performance. This includes active learning, Bayesian optimisation, continual learning and model based reinforcement learning. For continual learning, the DGP must be able to handle sequentially arriving data in a principled manner. This will require model adaptation, either through the addition of more pseudo-points or increasing the depth of the DGP. Extending the results of Bui et al. [2017b] to DGPs would be worthwhile.

A UCI Regression Results

Dataset	**SGP 100**	**SGP 500**	**DSVI-2**	**DSVI-3**	**DSVI-4**	**DSVI-5**	**AEP-2**	**PBP**
boston	-2.504 ± 0.204	-2.464 ± 0.228	-2.458 ± 0.237	-2.393 ± 0.187	-2.382 ± 0.170	-2.354 ± 0.130	$\mathbf{-2.323 \pm 0.159}$	-2.574 ± 0.089
concrete	-3.192 ± 0.069	3.097 ± 0.087	-3.082 ± 0.075	-3.059 ± 0.072	-3.040 ± 0.071	$\mathbf{-3.004 \pm 0.074}$	-3.160 ± 0.066	-3.161 ± 0.019
energy	-1.421 ± 0.071	-0.687 ± 0.102	-0.657 ± 0.124	-0.534 ± 0.174	-0.509 ± 0.189	$\mathbf{-0.455 \pm 0.170}$	-0.917 ± 0.077	-2.042 ± 0.019
kin8nm	0.965 ± 0.017	1.149 ± 0.021	1.364 ± 0.022	$\mathbf{1.388 \pm 0.027}$	1.380 ± 0.027	1.380 ± 0.024	1.047 ± 0.036	0.896 ± 0.006
naval	6.507 ± 0.389	6.820 ± 0.640	6.656 ± 0.990	$\mathbf{6.913 \pm 0.396}$	6.878 ± 0.383	6.832 ± 0.467	5.795 ± 0.219	3.731 ± 0.006
power	-2.789 ± 0.039	-2.766 ± 0.037	02.727 ± 0.039	$\mathbf{-2.696 \pm 0.045}$	-2.699 ± 0.040	-2.712 ± 0.051	-2.840 ± 0.032	-2.837 ± 0.009
protein	-2.940 ± 0.008	-2.894 ± 0.009	-2.783 ± 0.012	-2.699 ± 0.013	-2.692 ± 0.013	$\mathbf{-2.687 \pm 0.014}$	-2.943 ± 0.010	-2.973 ± 0.003
wine_red	-0.951 ± 0.058	$\mathbf{-0.936 \pm 0.060}$	-0.951 ± 0.069	-0.948 ± 0.0689	-0.947 ± 0.067	-0.947 ± 0.065	-0.958 ± 0.067	-0.968 ± 0.014

Table 4: UCI regression predictive log likelihoods. The higher the better.

Dataset	**SGP 100**	**SGP 500**	**DSVI-2**	**DSVI-3**	**DSVI-4**	**DSVI-5**	**AEP-2**	**PBP**
boston	2.982 ± 0.686	2.923 ± 0.698	2.904 ± 0.591	2.804 ± 0.590	2.748 ± 0.591	$\mathbf{2.678 \pm 0.544}$	2.745 ± 0.582	3.228 ± 0.195
concrete	6.020 ± 0.443	5.531 ± 0.461	5.381 ± 0.449	5.242 ± 0.422	5.134 ± 0.436	$\mathbf{4.910 \pm 0.452}$	5.869 ± 0.455	5.977 ± 0.221
energy	0.872 ± 0.100	0.473 ± 0.074	0.460 ± 0.060	0.400 ± 0.079	0.393 ± 0.079	$\mathbf{0.364 \pm 0.053}$	0.548 ± 0.068	1.098 ± 0.074
kin8nm	0.088 ± 0.003	0.076 ± 0.002	0.062 ± 0.002	$\mathbf{0.061 \pm 0.002}$	$\mathbf{0.061 \pm 0.002}$	$\mathbf{0.061 \pm 0.002}$	0.084 ± 0.004	0.091 ± 0.002
naval	0.0003 ± 0.0002	0.0002 ± 0.0003	0.0002 ± 0.0003	$\mathbf{0.0001 \pm 0.0001}$	0.0002 ± 0.0001	0.0002 ± 0.0002	0.0005 ± 0.0003	0.001 ± 0.0001
power	3.923 ± 0.142	3.833 ± 0.148	3.694 ± 0.154	3.573 ± 0.171	$\mathbf{3.580 \pm 0.154}$	3.604 ± 0.180	4.127 ± 0.155	4.182 ± 0.040
protein	4.572 ± 0.041	4.371 ± 0.044	3.938 ± 0.050	3.677 ± 0.050	3.671 ± 0.051	$\mathbf{3.660 \pm 0.050}$	4.622 ± 0.049	4.539 ± 0.029
wine_red	0.627 ± 0.037	$\mathbf{0.619 \pm 0.038}$	0.627 ± 0.041	0.625 ± 0.042	0.624 ± 0.041	0.625 ± 0.040	0.633 ± 0.041	0.635 ± 0.010

Table 5: UCI regression predictive RMSE. The lower the better.

References

Mark Van der Wilk. *Sparse Gaussian process approximations and applications.* PhD thesis, University of Cambridge, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

Marc Peter Deisenroth and Carl E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

David Duvenaud. *Automatic Model Construction with Gaussian Processes.* PhD thesis, University of Cambridge, 2014.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.50.

Geoffrey E Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

Andreas C. Damianou and Neil D. Lawrence. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pages 207–215, 2013. URL `http://proceedings.mlr.press/v31/damianou13a.html`.

Hugh Salimbeni and Marc Peter Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4591–4602, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep Gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pages 1472–1481, 2016.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X. URL `http://www.worldcat.org/oclc/61285753`.

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 567–574, 2009. URL `http://proceedings.mlr.press/v5/titsias09a.html`.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, 2015. URL `http://proceedings.mlr.press/v38/hensman15.html`.

Thang Bui. *Efficient Deterministic Approximate Bayesian Inference for Gaussian Process models*. PhD thesis, University of Cambridge, 2018.

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

Ching-An Cheng and Byron Boots. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, pages 5184–5194, 2017.

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Deep Gaussian processes with decoupled inducing inputs. *arXiv preprint arXiv:1801.02939*, 2018a.

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

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

Richard Turner, M. Sahani, D. Barber, Ali Cemgil, and S. Chiappa. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, pages 109–130, 01 2011.

Qing Liu and Donald A Pierce. A note on Gauss—Hermite quadrature. *Biometrika*, 81(3):624–629, 1994.

Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Thang D Bui, José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato, and Richard E Turner. Training deep Gaussian processes using stochastic expectation propagation and probabilistic backpropagation. *stat*, 1050:11, 2015.

Hugh Salimbeni. Private Communication, 2020.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 7506–7516, 2018b.

Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 2849–2858, 2017.

James Requeima, William Tebbutt, Wessel Bruinsma, and Richard E. Turner. The Gaussian process autoregressive regression model (GPAR). In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1860–1869. PMLR, 16–18 Apr 2019. URL `http://proceedings.mlr.press/v89/requeima19a.html`.

Thang D Bui, Cuong Nguyen, and Richard E Turner. Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 3299–3307, 2017b.