Training Deep Gaussian Processes using Stochastic Expectation Propagation and Probabilistic Backpropagation

Thang D. Bui
University of Cambridge
tdb40@cam.ac.uk

José Miguel Hernández-Lobato
Harvard University
jmhl@seas.harvard.edu

Yingzhen Li
University of Cambridge
yl494@cam.ac.uk

Daniel Hernández-Lobato
Universidad Autónoma de Madrid
daniel.hernandez@uam.es

Richard E. Turner
University of Cambridge
ret26@cam.ac.uk

Abstract

Deep Gaussian processes (DGPs) are multi-layer hierarchical generalisations of Gaussian processes (GPs) and are formally equivalent to neural networks with multiple, infinitely wide hidden layers. DGPs are probabilistic and non-parametric and as such are arguably more flexible, have a greater capacity to generalise, and provide better calibrated uncertainty estimates than alternative deep models. The focus of this paper is scalable approximate Bayesian learning of these networks. The paper develops a novel and efficient extension of probabilistic backpropagation, a state-of-the-art method for training Bayesian neural networks, that can be used to train DGPs. The new method leverages a recently proposed method for scaling Expectation Propagation, called stochastic Expectation Propagation. The method is able to automatically discover useful input warping, expansion or compression, and it is therefore is a flexible form of Bayesian kernel design. We demonstrate the success of the new method for supervised learning on several real-world datasets, showing that it typically outperforms GP regression and is never much worse.

1 Introduction

Gaussian Processes (GPs) are powerful nonparametric distributions over continuous functions which can be used for both supervised and unsupervised learning problems [1]. In this article, we study a multi-layer hierarchical generalisation of GPs or deep Gaussian Processes (DGPs) for supervised learning tasks. A GP is equivalent to an infinitely wide neural network with single hidden layer and similarly a DGP is a multi-layer neural network with multiple infinitely wide hidden layers [2]. The mapping between layers in this type of network is parameterised by a GP, and as a result DGPs are arguably more flexible, have a greater capacity to generalise, and are able to provide better calibrated predictive uncertainty estimates than standard multi-layer models [3].

More formally, suppose we have a training set comprising N D-dimensional input vectors $\{\mathbf{x}_n\}_{n=1}^N$ and corresponding real valued scalar observations $\{y_n\}_{n=1}^N$. The probabilistic representation of a DGP comprising of L layers can be written as follows,

$$p(f_l|\theta_l) = \mathcal{GP}(f_l; \mathbf{0}, \mathbf{K}_l), \quad l = 1, \cdots, L \tag{1}$$

$$p(\mathbf{h}_l|f_l, \mathbf{h}_{l-1}, \sigma_l^2) = \prod_n \mathcal{N}(h_{l,n}; f_l(h_{l-1,n}), \sigma_l^2), \quad h_{1,n} = \mathbf{x}_n \tag{2}$$

$$p(\mathbf{y}|f_L, \mathbf{X}_{L-1}, \sigma_L^2) = \prod_n \mathcal{N}(y_{L,n}; f_L(h_{L-1,n}), \sigma_L^2) \tag{3}$$

Here hidden layers are denoted $h_{l,n}$ and the mapping function between the layers, f_l, is drawn from a GP. A Gaussian (regression) likelihood is used in this work, a different likelihood can easily be accommodated.[1]

The DGP collapses back to a standard GP when $L = 1$ (when there are no hidden layers) or when only one of the functions is non-linear. The addition of non-linear hidden layers can potentially overcome practical limitations of *shallow* GPs. First, modelling real-world complex datasets often requires rich, hand-designed covariance functions. DGPs can perform input warping or dimensionality compression or expansion, and hence automatically learn to construct a kernel that works well for the data at hand. Second, the functional mapping from inputs to outputs specified by a DGP is non-Gaussian which is a more general and flexible modelling choice. Third, DGPs can repair damage done by sparse approximations to the representational power of each GP layer. For example, inducing point based approximation methods for GPs trade model complexity for a lower computational complexity of $\mathcal{O}(LNM^2)$ where L is the number of layers, N is the number of datapoints and M is the the number of inducing points. This complexity scales quadratically in M whereas the dependence on the number of layers L is only linear. Therefore, it can be cheaper to increase the representation power of the model by adding extra layers rather than adding more inducing points.

The focus of this paper is Bayesian learning of DGPs, which involves inferring the posterior over the mappings between layers and hyperparameter tuning using the marginal likelihood. Unfortunately, exact Bayesian learning in this model is analytically intractable and as such approximate inference is needed. Recent work in this frontier largely focussed on variational free-energy approaches [4]. We introduce an alternative approximation scheme based on three approximations. First, in order to side step the cubic computational cost of GPs we leverage a well-known inducing point sparse approximation [5, 6]. Second, expectation propagation is used to approximate the analytically intractable posterior over the inducing points. Here we utilise stochastic expectation propagation (SEP) that prevents the memory overhead from increasing with the number of datapoints [7]. Third, the SEP moment computation is itself analytically intractable and requires one final approximation. For this we use the probabilistic backpropagation approximation [8]. The proposed enables the advantages of the DGP model to be realised through a computationally efficient, scalable and easy to implement algorithm.

2 The Fully Independent Training Conditional approximation

The computational complexity of full GP models scales cubically with the number of training instances, making it intractable in practice. Sparse approximation techniques are therefore often resorted to. They can be coarsely put into two classes: ones that explicitly sparsify and create a parametric representation that approximates the original model, and ones that retain the original nonparametric properties and perform sparse approximation to the exact posterior. The method we describe and use here, Fully Independent Training Conditional (FITC), falls into the first category. The FITC approximation is formed by considering a small set of function values $\mathbf{u}$ in the infinite dimensional vector f and assuming conditional independence between the remaining values given the set $\mathbf{u}$ [5,6]. This set is often called inducing points or pseudo datapoints and their input locations can be chosen by optimising the approximate marginal likelihood so that the approximate model is closer to the original model. The resulting model can be written as follows,

$$p(\mathbf{u}_l|\theta_l) = \mathcal{N}(\mathbf{u}_l; \mathbf{0}, \mathbf{K}_{\mathbf{z}_{l-1},\mathbf{z}_{l-1}}), \quad l = 1, \cdots, L \tag{4}$$

$$p(\mathbf{h}_l|\mathbf{u}_l, \mathbf{h}_{l-1}, \sigma_l^2) = \prod_n \mathcal{N}(h_{l,n}; \mathbf{C}_{n,l}\mathbf{u}_l, \mathbf{R}_{n,l}), \tag{5}$$

$$p(\mathbf{y}|\mathbf{u}_L, \mathbf{H}_{L-1}, \sigma_L^2) = \prod_n \mathcal{N}(y_n; \mathbf{C}_{n,L}\mathbf{u}_L, \mathbf{R}_{n,L}). \tag{6}$$

where $\mathbf{C}_{n,l} = \mathbf{K}_{\mathbf{h}_{l-1,n},\mathbf{z}_{l-1}}\mathbf{K}^{-1}_{\mathbf{z}_{l-1},\mathbf{z}_{l-1}}$ and $\mathbf{R}_{n,l} = \mathbf{K}_{\mathbf{h}_{l-1,n},\mathbf{h}_{l-1,n}} - \mathbf{K}_{\mathbf{h}l-1,n,\mathbf{z}_{l-1}}\mathbf{K}^{-1}_{\mathbf{z}_{l-1},\mathbf{z}_{l-1}}\mathbf{K}_{\mathbf{z}_{l-1},\mathbf{h}_{l-1,n}} + \sigma_l^2$. The FITC approximation creates a parametric

[1]Hidden variables in the intermediate layers can and will generally have multiple dimensions but we have omitted this here to lighten the notation.

model, but one which is cleverly structured so that the induced non-stationary noise captures the uncertainty introduced from the sparsification.

3 Stochastic expectation propagation for deep, sparse Gaussian processes

Having specified a probabilistic model for data using a deep sparse Gaussian processes we now consider inference for the inducing outputs $\mathbf{u}$ and learning of the inducing inputs $\{\mathbf{z}_l\}_{l=1}^L$ and hyperparameters $\{\theta\}_{l=1}^L$. The posterior distribution over the inducing points can be written as $p(\mathbf{u}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{u}) \prod_n p(y_n|\mathbf{u}, \mathbf{X}_n)$. This quantity can then be used for prediction of output given a test input, $p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = \int \mathrm{d}\mathbf{u} p(\mathbf{u}|\mathbf{X}, \mathbf{y}) p(y^*|\mathbf{u}, \mathbf{x}^*)$. However, the posterior of $\mathbf{u}$ is not analytically tractable when there is more than one GP layer in the model. As such, approximate inference is needed; here we use Stochastic Expectation Propagation (SEP), a recently proposed modification to Expectation Propagation (EP) [7].

In SEP, the posterior $p(\mathbf{u}|\mathbf{X}, \mathbf{y})$ is approximated by $q(\mathbf{u}) \propto p(\mathbf{u}) g(\mathbf{u})^N$, where the factor $g(\mathbf{u})$ could be thought of as an *average* data factor that captures the average effect of a likelihood term on the posterior. The form chosen, though seems limited as first, in practice performs almost as well as EP in which there is a factor $g_n(\mathbf{u})$ per datapoint, while significant reducing EP's memory footprint [7]. Specifically for our model, the memory complexity of EP is $\mathcal{O}(NM^2)$ as we need to store the mean and covariance matrix for each data factor; in contrast, such requirement for SEP is only $\mathcal{O}(M^2)$ regardless of the number of training points.

The SEP procedure involves looping through the dataset multiple times and performing the following steps: 1. remove $g(\mathbf{u})$ from the approximate posterior to form the cavity distribution, 2. incorporate a likelihood term into the cavity to form the tilted distribution, 3. moment match the approximate posterior to this distribution, and in addition to EP, 4. perform a small update to the *average* factor $g(\mathbf{u})$. We choose a Gaussian form for both $q(\mathbf{u})$ and $g(\mathbf{u})$, and as a result steps 1 and 4 are analytically tractable. We will discuss how to deal with the intermediate steps given a training datapoint $(\mathbf{x}, y)$ in the next section.

4 Probabilistic backpropagation for deep, sparse Gaussian processes

The moment matching step in SEP is analytically intractable as it involves propagating a Gaussian distribution through a DGP and computing the moments of the resulting complex distribution. However, for certain choices of covariance functions $\{\mathbf{K}_l\}_{l=1}^L$, it is possible to use an efficient and accurate approximation which propagates a Gaussian through the first layer of the network and projects this non-Gaussian distribution back to a moment matched Gaussian before propagating through the next layer and repeating the same steps. This scheme is a central part of the probabilistic backpropagation algorithm that has been applied to standard neural networks [8].

In more detail, let $q^{\backslash 1}(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}^{\backslash 1}, \mathbf{V}^{\backslash 1})$ be the cavity distribution, the difficult steps above are equivalent to the following updates to the mean and covariance of the approximate posterior:

$$\mathbf{m} = \mathbf{m}^{\backslash 1} + \mathbf{V}^{\backslash 1} \frac{\mathrm{d} \log \mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}}, \qquad \mathbf{V} = \mathbf{V}^{\backslash 1} - \mathbf{V}^{\backslash 1} \left[\frac{\mathrm{d} \log \mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}} \left(\frac{\mathrm{d} \log \mathcal{Z}}{\mathrm{d}\mathbf{m}^{\backslash 1}} \right)^{\intercal} - 2 \frac{\mathrm{d} \log \mathcal{Z}}{\mathrm{d}\mathbf{V}^{\backslash 1}} \right] \mathbf{V}^{\backslash 1}, \tag{7}$$

where $\mathcal{Z} = \int \mathrm{d}\mathbf{u} p(y|\mathbf{x}, \mathbf{u}) q^{\backslash 1}(\mathbf{u})$ [9]. The inference scheme therefore reduces to evaluating the normalising constant $\mathcal{Z}$ and its gradient. By reintroducing the hidden variables in the middle layers, we perform Gaussian approximation to $\mathcal{Z}$ in a sequential fashion, taking a two GP layers case as an example:

$$\mathcal{Z} = \int \mathrm{d}\mathbf{u} p(y|\mathbf{x}, \mathbf{u}) q^{\backslash 1}(\mathbf{u}) = \int \mathrm{d}h_1 \mathrm{d}\mathbf{u}_2 p(y|h_1, \mathbf{u}_2) q^{\backslash 1}(\mathbf{u}_2) \int \mathrm{d}\mathbf{u}_1 p(h_1|\mathbf{x}, \mathbf{u}_1) q^{\backslash 1}(\mathbf{u}_1) \tag{8}$$

We can exactly marginalise out the inducing points for each GP layer leading to $\mathcal{Z} = \int \mathrm{d}h_1 \mathrm{d}\mathbf{u}_2 q(y|h_1) q(h_1)$ where $q(h_1) = \mathcal{N}(h_1; m_1, v_1)$, $q(y|h_1) = \mathcal{N}(y|h_1; m_{2|h_1}, v_{2|h_1})$ and

$$m_1 = \mathbf{K}_{\mathbf{x},\mathbf{z}_1} \mathbf{K}_{\mathbf{z}_1,\mathbf{z}_1}^{-1} \mathbf{m}_1^{\backslash 1}, \quad v_1 = \sigma_1^2 + K_{\mathbf{x},\mathbf{x}} - \mathbf{K}_{\mathbf{x},\mathbf{z}_1} \mathbf{K}_{\mathbf{z}_1,\mathbf{z}_1}^{-1} \mathbf{K}_{\mathbf{z}_1,\mathbf{x}} + \mathbf{K}_{\mathbf{x},\mathbf{z}_1} \mathbf{K}_{\mathbf{z}_1,\mathbf{z}_1}^{-1} \mathbf{V}_1^{\backslash 1} \mathbf{K}_{\mathbf{z}_1,\mathbf{z}_1}^{-1} \mathbf{K}_{\mathbf{z}_1,\mathbf{x}}$$

$$m_{2|h_1} = \mathbf{K}_{h_1,\mathbf{z}_2} \mathbf{K}_{\mathbf{z}_2,\mathbf{z}_2}^{-1} \mathbf{m}_2^{\backslash 1}, \quad v_{2|h_1} = \sigma_2^2 + K_{h_1,h_1} - \mathbf{K}_{h_1,\mathbf{z}_2} \mathbf{K}_{\mathbf{z}_2,\mathbf{z}_2}^{-1} \mathbf{K}_{\mathbf{z}_2,h_1} + \mathbf{K}_{h_1,\mathbf{z}_2} \mathbf{K}_{\mathbf{z}_2,\mathbf{z}_2}^{-1} \mathbf{V}_1^{\backslash 1} \mathbf{K}_{\mathbf{z}_2,\mathbf{z}_2}^{-1} \mathbf{K}_{\mathbf{z}_2,h_1}$$

Following [10], we can approximate the difficult integral in the equation above by a Gaussian $\mathcal{Z} \approx \mathcal{N}(y|m_2, v_2)$ where the mean and variance take the following form,

$$m_2 = \mathrm{E}_{q(h_1)}[m_{2|h_1}] = \mathrm{E}_{q(h_1)}[\mathbf{K}_{h_1,\mathbf{z}_2}] \mathbf{K}_{\mathbf{z}_2,\mathbf{z}_2}^{-1} \mathbf{m}_2^{\backslash 1} \tag{9}$$

$$
\begin{aligned}
v_2 &= \mathrm{E}_{q(h_1)}[v_{2|h_1}] + \mathrm{var}_{q(h_1)}[m_{2|h_1}] && (10)\\
&= \sigma_2^2 + \mathrm{E}_{q(h_1)}[K_{h_1,h_1}] + \mathrm{tr}\left(\mathbf{B}\mathrm{E}_{q(h_1)}[\mathbf{K}_{\mathbf{z}_2,h_1}\mathbf{K}_{h_1,\mathbf{z}_2}]\right) - m_2^2 && (11)
\end{aligned}
$$

where $\mathbf{B} = \mathbf{K}^{-1}_{\mathbf{z}_2,\mathbf{z}_2}(\mathbf{V}_2^{\backslash 1} + \mathbf{m}_2^{\backslash 1}\mathbf{m}_2^{\backslash 1,\mathrm{T}})\mathbf{K}^{-1}_{\mathbf{z}_2,\mathbf{z}_2} - \mathbf{K}^{-1}_{\mathbf{z}_2,\mathbf{z}_2}$. The equations above require the expectations of the kernel matrix under a Gaussian distribution over the inputs, which are analytically tractable for widely used kernels such as exponentiated quadratic, linear or a more general class of spectral mixture kernels [11]. Importantly, the computation graph of the approximation to $\log \mathcal{Z}$ and its gradient can be easily programmed using symbolic packages such as Theano [12].

5 Stochastic optimisation of hyperparameters and inducing point locations

We complete the main text by discussing the optimisation of model hyperparameters and inducing point locations. Given the approximation to the posterior obtained by using SEP as described above, one can also obtain the approximate marginal likelihood and its gradients. As a result, parameter training now involves iterating between running SEP and updating the hypeparameters based on these gradients. However, this procedure can only made efficient and scalable by following two observations discussed in [13], which include 1. we do not need to wait for (S)EP to converge before making an update to the parameters, and 2. the gradients involve a sum across the whole training set, enabling fast optimisation using stochastic gradients computed on minibatches of datapoints.

6 Experimental results

We test our approximation method on several DGP architectures for a regression task on several real-world datasets. We obtain 20 random splits of each dataset, 90% for training and 10% for testing and report the average results and their standard deviations in table 1. The prediction errors are evaluated using two metrics: root mean squared error (RMSE) and mean log loss (MLL). We use an exponentiated quadratic kernel with ARD lengthscales. The lengthscales and inducing points of the first GP layer are sensibly initialised based on the median distance between datapoints in the input space and the k-means cluster centers respectively. We use long lengthscales and initial inducing points between $[-1, 1]$ for the higher layers to force them to start up with an identity mapping. For all results reported here, we use Adam [14] with minibatch size of 50 datapoints and run the optimiser for 4000 iterations. The learning rate is selected by optimising the predictive errors on a small subset of training points using Bayesian optimisation [15]. We experiment with two one-hidden-layer DGP networks with hidden variables of one and two dimensions, 50 inducing points per layer and denote them as [DGP, 1, 50] [2] and [DGP, 2, 50] respectively. We compare them against sparse GP regression with the same number of inducing points [GP, 50]. The results in the table below show that overall DGP with two dimensional hidden variables perform as well or better than sparse GP, and almost always better than the architecture with one dimensional hidden variables. Taking the Boston Housing dataset as an example, the mean test log likelihood using [DGP, 2, 50] is -2.12, which is, to the best of our knowledge, better than state-of-the-art results which were obtained using Bayesian neural networks with probabilistic backpropagation: -2.57 [8], dropout: -2.46 [16] or SGLD: -2.31 [17].

			RMSE			MLL		
Dataset	N	D	GP, 50	DGP, 1, 50	DGP, 2, 50	GP, 50	DGP, 1, 50	DGP, 2, 50
boston	506	13	3.09 ± 0.63	2.85 ± 0.65	$\mathbf{2.47 \pm 0.49}$	-2.26 ± 0.31	-2.30 ± 0.53	$\mathbf{-2.12 \pm 0.37}$
concrete	1030	8	5.24 ± 0.55	5.91 ± 1.65	$\mathbf{5.21 \pm 0.90}$	-2.97 ± 0.10	-3.07 ± 0.14	$\mathbf{-2.70 \pm 0.35}$
energy 1	768	8	0.50 ± 0.10	0.77 ± 0.59	$\mathbf{0.48 \pm 0.05}$	-0.26 ± 0.13	-0.39 ± 0.37	$\mathbf{-0.20 \pm 0.14}$
energy 2	768	8	1.60 ± 0.15	1.78 ± 0.43	$\mathbf{1.37 \pm 0.23}$	-1.05 ± 0.28	-1.14 ± 0.32	$\mathbf{-0.76 \pm 0.15}$
kin8nm	8192	8	0.04 ± 0.00	0.07 ± 0.04	$\mathbf{0.02 \pm 0.00}$	2.02 ± 0.06	1.71 ± 0.35	$\mathbf{2.48 \pm 0.03}$
naval 1	11934	16	0.02 ± 0.01	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	2.64 ± 1.14	$\mathbf{5.14 \pm 0.37}$	5.02 ± 0.59
naval 2	11934	16	0.01 ± 0.00	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$	3.52 ± 0.02	4.67 ± 0.68	$\mathbf{5.24 \pm 0.48}$
power	9568	4	3.19 ± 0.18	3.35 ± 0.20	$\mathbf{2.95 \pm 0.30}$	-2.53 ± 0.03	-2.61 ± 0.05	$\mathbf{-2.38 \pm 0.13}$
red wine	1588	11	$\mathbf{0.48 \pm 0.06}$	0.62 ± 0.05	0.54 ± 0.11	-0.06 ± 0.15	-0.10 ± 0.64	$\mathbf{0.29 \pm 0.65}$
white wine	4898	11	0.37 ± 0.04	0.49 ± 0.09	$\mathbf{0.34 \pm 0.07}$	0.01 ± 0.11	-0.17 ± 0.36	$\mathbf{0.66 \pm 0.31}$
creep	2066	31	95.87 ± 18.03	74.86 ± 13.66	$\mathbf{70.58 \pm 15.55}$	-5.85 ± 0.35	-5.45 ± 0.20	$\mathbf{-5.28 \pm 0.29}$

Table 1: Predictive errors using DGPs and GPs for regression on several UCI datasets

[2] This is the same as Bayesian warped GPs, that is the one dimensional output of the first layer is warped through a GP to form the prediction/output.

In addition, we also vary the number of inducing points per layer for the above networks and trace out the speed-accuracy frontier. Preliminary results indicates that DGPs is very efficient using our inference technique and with a small number of inducing points, can obtain a predictive performance that would require many more inducing points in a shallower architecture.

7 Conclusion

We have proposed a novel approximation scheme for deep Gaussian processes for supervised learning. Our method extends probabilistic backpropagation for Bayesian neural networks, combines it with an inducing point based sparse GP approximation and a recently proposed method for scalable approximate Bayesian inference, Stochastic Expectation Propagation. We systematically evaluate our approach on several regression datasets and the initial experimental results demonstrate the validity of our method and the effectiveness of DGPs compared to GPs. Our method is fast, easy to implement and promisingly, gives state-of-the-art performance in various regression tasks.

Current work includes performing experiments on large scale datasets, comparing our method to the variational approach presented in [4], extension to classification and unsupervised learning, and understanding the effect of the network architectures on prediction quality.

Acknowledgements

TB thanks Google for funding his European Doctoral Fellowship. JMHL acknowledges support from the Rafael del Pino Foundation. DHL and JMHL acknowledge support from Plan Nacional I+D+i, Grant TIN2013-42351-P, and from CAM, Grant S2013/ICE-2845 CASI-CAM-CM. YL thanks the Schlumberger Foundation for her Faculty for the Future PhD fellowship. RET thanks EPSRC grants EP/G050821/1 and EP/L000776/1.

References

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[2] A. C. Damianou and N. D. Lawrence, “Deep Gaussian processes,” in *16th International Conference on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[3] A. Damianou, *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, University of Sheffield, 2015.

[4] J. Hensman and N. D. Lawrence, “Nested variational compression in deep Gaussian processes,” *arXiv preprint arXiv:1412.1370*, 2014.

[5] E. Snelson and Z. Ghahramani, “Sparse Gaussian processes using pseudo-inputs,” in *Advances in Neural Information Processing Systems 19*, pp. 1257–1264, 2006.

[6] J. Quiñonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate Gaussian process regression,” *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[7] Y. Li, J. M. Hernandez-Lobato, and R. E. Turner, “Stochastic expectation propagation,” in *Advances in Neural Information Processing Systems 29*, 2015.

[8] J. M. Hernández-Lobato and R. P. Adams, “Probabilistic backpropagation for scalable learning of Bayesian neural networks,” in *32nd International Conference on Machine Learning*, 2015.

[9] T. P. Minka, *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[10] A. Girard, C. E. Rasmussen, J. Quiñonero-Candela, and R. Murray-Smith, “Gaussian process priors with uncertain inputs — application to multiple-step ahead time series forecasting,” in *Advances in Neural Information Processing Systems 15*, pp. 529–536, 2003.

[11] A. Wilson and R. Adams, “Gaussian process kernels for pattern discovery and extrapolation,” in *30th International Conference on Machine Learning*, pp. 1067–1075, 2013.

[12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, “Theano: new features and speed improvements.” Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[13] D. Hernández-Lobato and J. M. Hernández-Lobato, “Scalable Gaussian process classification via expectation propagation,” *arXiv preprint arXiv:1507.04513*, 2015.

[14] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” in *3rd International Conference on Learning Representations*, 2015.

[15] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems 25*, pp. 2951–2959, 2012.

[16] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” *arXiv preprint arXiv:1506.02142*, 2015.

[17] A. Korattikara, V. Rathod, K. Murphy, and M. Welling, “Bayesian dark knowledge,” in *Advances in Neural Information Processing Systems 29*, 2015.