Deep Gaussian Processes for Machine Learning

Sarim Zubair, Hao Zhao and Upala Junaida Islam

December 2022

1 Introduction

With complementary advantages and drawbacks, Neural networks and Gaussian processes are two well-established frameworks for solving both regression and classification problems (Dutordoir et al., 2021). Gaussian Process Models perhaps are the less known machine learning algorithms in contrast to Neural Networks (i.e., tree based models). Based on the hype of 1987, Neural Networks were supposed to serve as intelligent AI models, which discovered new features and patterns in data. Whereas, Gaussian processes are non-parametric simply smoothing devices. In addition, Gaussian Process Models are one of the few machine learning models that can be solved analytically in complex systems. There are several other reasons a gaussian process can possibly replace neural networks:

Enormous datasets can be handled by computationally scalable neural networks (NNs), but uncertainty estimates along with robust solutions can be provided only by Gaussian processes (GPs) in lowe-data regimes. Damianou and Lawrence (Damianou & Lawrence, 2013) introduced the Deep Gaussian process (DGP) as a promising attempt to a single model that provides the best of both approaches: the ability to handle low and high dimensional inputs and to make robust uncertainty-aware predictions from the small to big data regimes (Dutordoir et al., 2021). By warping the input space through hidden Gaussian layers, and effectively moving some training samples closer and others farther apart, deep Gaussian processes (DGPs) achieve non-stationarity even under otherwise conventional kernel structures (Sauer, Gramacy, & Higdon, 2022).

The rest of the paper is organized as follows, Section 2 and 3 presents the background, structure, use and application of Gaussian process and deep Gaussian process respectively. Some experimentation and results are showcased in Section 4. The paper concludes in Section 5.

2 Gaussian Processes in Machine Learning

In simple terms, Machine learning is using data we have, also known as training data to learn a function that we can use to make predictions about data we don't have yet. In addition, Machine learning is an extension of linear regression in a few ways. Firstly that modern ML deals with much more complicated data, instead of learning a function to calculate a single number from another number like in linear regression, we might be dealing with different inputs and outputs such as Digit recognition (Classification Model) based on a database of handwritten digits (i.e., MNIST database).

In the Digit recognition system, the training set consists of small digitized images, together with a classification from 0, . . . , 9, normally provided by a human. The goal of the ML model is to learn a mapping from image to classification label, which can then be used on new, unseen images. Supervised learning Machine Learning is an effective way to attempt to tackle this problem since it is not easy to specify accurately the characteristics of, for instance, the handwritten digit 7. Further, modern ML uses much more significant methods for extracting patterns, such as deep learning techniques. Gaussian processes are another of these methods and their primary distinction from other complex models in machine learning is their relation to uncertainty.

The world around us is filled with uncertainty. In other words, we do not know exactly how long our commute will take or precisely what the weather will be at noon tomorrow. Some uncertainty is due to our dearth of knowledge and is intrinsic to the world no matter how much knowledge we currently hold. In the real world, we are unable to completely remove uncertainty from the universe and need to plan a good way of dealing with it. Therefore, Probability distributions are an effective measure to solve uncertainty problems and are the key to understanding Gaussian processes.

Today, in statistics and machine learning research, supervised learning approaches in the form of regression for continuous outputs or classification for discrete outputs plays a crucial role in analyzing data sets and solving complex data problems (Rasmussen, 2003). Generally, models (e.g., parametric models) that absorb data during training into parameters have been used for this purpose quite frequently. In the case of working with complex data sets, simple parametric models may lack adequate expressive power (Rasmussen, 2003). However, the Gaussian process offers model flexibility and practicality in terms of functionality that other parametric models fail to provide.

2.1 Gaussian Process

In the absence of any information about the functional form of $f(\mathbf{x})$, we consider a Gaussian process regression as our underlying model. Gaussian Process Regression (GPR) is a non-parametric regression model that considers a distribution over the underlying function $f(\mathbf{x})$ and aims to specify the black-box function by its mean function $m(\mathbf{x})$ and covariance function $\Sigma(\mathbf{x}, \mathbf{x}')$. The observed data is likely to have noise inherited without the learner's knowledge, hence we consider the output y to be comprised of the function value $f(\mathbf{x})$ as

$$y = f(\mathbf{x}) + \varepsilon \qquad\qquad \varepsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\right) \tag{1}$$

Observing the noisy outputs, $\tilde{\mathbf{y}}$, GPR attempts to reconstruct the underlying function $f(\mathbf{x})$ by removing the contaminating noise ε (Williams & Rasmussen, 2006). We denote the training dataset as (X_o, Y_n). GPR imposes a zero-mean Gaussian process prior over the noisy outputs such that

$$Y_n \sim \mathcal{N}\left(\mathbf{0}, \Sigma(X_n)\right) \tag{2}$$

Here $\Sigma(X_n) \equiv \Sigma_n$ denotes the covariance matrix in between the training points. The covariance or kernel function can be constructed in various fashion depending on the desired smoothness and flexibility. One of the most frequently used kernel is the squared exponential kernel that has the form

$$\Sigma_n^{ij} = \tau^2 \left(\exp\left(-\frac{\|x_i - x_j\|^2}{\theta} \right) + g\mathbb{I}_{\{i=j\}} \right)$$

It represents the how two data values Y_i and Y_j are based on the corresponding correlation between X_i and X_j with the help of the hyperparameters τ, θ, and g. Here, τ represents how correlated pairs of points are, θ or the length scale denotes the spread of the similarity function, and finally g controls the noise level in prediction.

The joint prior distribution between the training output set $\mathbf{y}$ and test output set $\hat{\mathbf{f}}$ is as following

$$\begin{bmatrix} \mathbf{y} \\ \hat{\mathbf{f}} \end{bmatrix} \sim \mathcal{N}_m \left(\mathbf{0}, \begin{bmatrix} \Sigma(X_n, X_n) + \sigma_n^2 I & \Sigma(X_n, \mathcal{X}) \\ \Sigma(\mathcal{X}, X_n) & \Sigma(\mathcal{X}, \mathcal{X}) \end{bmatrix} \right) \tag{3}$$

where $\mathcal{X}$ is the set of unlabeled testing points. The posterior distribution at the test samples is given as $\{\hat{\mathbf{f}} | X_n, Y_n, \mathcal{X}\} \sim \mathcal{N}\left(\hat{\mathbf{f}}, \text{cov}(\hat{\mathbf{f}})\right)$, where

$$\hat{\mathbf{f}} = \Sigma(\mathcal{X}, X_n)[\Sigma(X_n, X_n) + \sigma_n^2 I]^{-1} Y_n \tag{4}$$

$$\text{cov}(\hat{\mathbf{f}}) = \Sigma(\mathcal{X}, \mathcal{X}) - \Sigma(\mathcal{X}, X_n)^T [\Sigma(X_n, X_n) + \sigma_n^2 I]^{-1} \Sigma(X_n, \mathcal{X}) \tag{5}$$

Spectral Mixture (SM) Kernel: The SM kernel allows to learn all the frequencies in the data at once by learning its spectral density. From Bochner's theorem, learning the spectral density itself is equivalent to learning a kernel (Wilson & Adams, 2013), which is

$$k_{SM}(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\left\{-2\pi^2\tau_p^2 v_q^{(p)}\right\} \cos\left(2\pi\tau_p\mu_q^{(p)}\right) \tag{6}$$

where Q is a mixture of Gaussians on $\mathbb{R}^P$, the q^{th} component has mean vector $\boldsymbol{\mu}_q = \left(\mu_q^{(1)}, \ldots, \mu_q^{(P)}\right)$ and covariance matrix $\mathbf{M}_q = \operatorname{diag}\left(v_q^{(1)}, \ldots, v_q^{(P)}\right)$, and τ_p is the p^{th} component of the P dimensional vector $\tau = x - x'$. The weights w_q specify the relative contribution of each mixture component.

RBF (Squared Exponential) Kernel: The most widely used covariance function is arguably the SE function, given by

$$k_{\text{RBF}}\left(\mathbf{x}_1, \mathbf{x}_2\right) = \exp\left(-\frac{1}{2}\left(\mathbf{x}_1 - \mathbf{x}_2\right)^\top \Theta^{-2}\left(\mathbf{x}_1 - \mathbf{x}_2\right)\right) \tag{7}$$

where Θ is a lengthscale parameter.

Periodic Kernel: The periodic kernel allows one to model functions which repeat themselves exactly,

$$k_{\text{Periodic}}\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-2\sum_i \frac{\sin^2\left(\frac{\pi}{p}\left(x_i - x_i'\right)\right)}{\lambda}\right) \tag{8}$$

where p is the period length parameter, λ is a lengthscale parameter.

Polynomial Kernel: Polynomial Kernel uses a polynomial function to map the data into a higher-dimensional space,

$$k_{\text{Poly}}\left(\mathbf{x}_1, \mathbf{x}_2\right) = \left(\mathbf{x}_1^\top \mathbf{x}_2 + c\right)^d \tag{9}$$

where c is an offset parameter, d is data dimensions.

3 Deep Gaussian Process

Due to the non-stationary flexibility as well as the ability to cope with abrupt regime changes in training data, deep Gaussian processes (DGPs) are becoming more popular predictive models in machine learning (ML) compared to ordinary stationary Gaussian processes (Sauer, Gramacy, & Higdon, 2022).

A DGP is a hierarchical layering of GPs each following conditional multivariate distribution. Therefore, an ordinary GP can be described as a "one-layer" GP, while in a multi-layer model, the input passes through the hidden layers of the intermediate GP before reaching the output response. DGP can be constructed in different ways depending on the feasibility and desired interpretability (Dunlop, Girolami, Stuart, & Teckentrup, 2018)

- The composition construction builds the hierarchy of layers using a stationary covariance function and composition (Damianou & Lawrence, 2013).
- The covariance function construction where the hierarchy is built builds the hierarchy using a stationary covariance function, and iteratively modifing its associated length scale (Paciorek, 2003).
- The covariance operator construction builds the hierarchy using a stochastic partial differential equations (SPDE) representation of stationary Matèrn fields, and again iteratively modifies their associated length scale (Lindgren, Rue, & Lindström, 2011).
- The convolution construction builds the hierarchy via iterative convolution of Gaussian random fields.

The form of the DGP likelihood makes direct inference impossible. Damianou and Lawrence took advantage of variational learning of inducing variables in the sparse GP to train the DGP (Damianou & Lawrence, 2013). These inducing variables are like a set of data that summarizes all the information of other data. Unlike this method which proposed under strong independence and Gaussianity assumptions (isotropic Gaussian kernel), Salimbeni presented a variational algorithm for inference in DGP models that do not force independence or Gaussianity between the layers (Salimbeni & Deisenroth, 2017). While Sauer and Gramacy were inspired by active learning (AL), they "actively" select the data based on maximum-variance acquisitions from the collected Markov chain Monte Carlo (MCMC) samples(Sauer, Gramacy, & Higdon, 2022). To expand the utility and reach of fully Bayesian posterior inference for DGP models, they deployed Vecchia approximation at each Gaussian layer with MCMC scheme unchanged, and this framework allows to choose nearest-neighbors conditioning sets from the randomizing over orderings data(Sauer, Cooper, & Gramacy, 2022). Another popular approach is do the random feature approximation for each layer of DGP, then it becomes the Deep Neural Networks (Cutajar, Bonilla, Michiardi, & Filippone, 2017).

Similar to Deep Neural Networks, DGP does not benefit from infinity layers, the distribution of the infinitely deep kernels suffer the saturated effect and its recursion converges to $\Sigma(\mathbf{x}, \mathbf{x}') = 1$ for all pairs of inputs (Duvenaud, Rippel, Adams, & Ghahramani, 2014).

Viewing DGP as functional compositions, we consider a prior where the inputs X_n are mapped through one or more intermediate GPs before reaching the response Y_n. The inputs to the intermediate "layers" act as latent variables that warp the input space remaining unobserved.

In a two-layer DGP model, the inputs go to the new hidden GP that we label as W. The model structure is presented in Figure 1. The hierarchical model can be described as

$$Y_n|W \sim \mathcal{N}(0, \Sigma(W)) \qquad W \sim \mathcal{N}(0, \Sigma(X_n)) \tag{10}$$

Here, the marginal likelihood can be obtained by integrating as

$$\mathcal{L}(Y_n|X_n) \propto \int \mathcal{L}(Y_n|W)\mathcal{L}(W|X_n)dW \tag{11}$$

where $\log \mathcal{L}(Y_n|W)$ and $\log \mathcal{L}(W|X_n)$ are defined as

$$\log \mathcal{L}(Y_n|W) \propto -\frac{1}{2}\log|\Sigma(W)| - \frac{1}{2}Y_n^T\Sigma(W)^{-1}Y_n \tag{12}$$

$$\log \mathcal{L}(W|X_n) \propto -\frac{1}{2}\log|\Sigma| - \frac{1}{2}W^T\Sigma^{-1}W \tag{13}$$

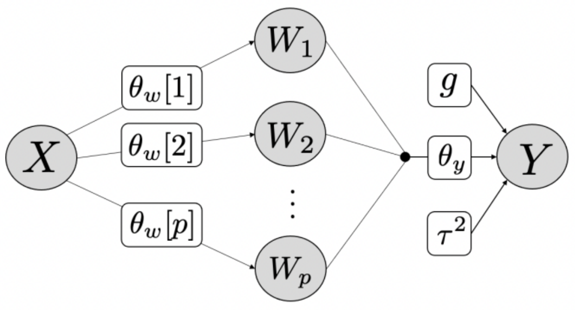


Figure 1: Model structure for a two-layer DGP with p latent nodes

In a three-layer DGP model, we have two hidden layers that we label as Z and W. The model structure is presented in Figure 2 whereas the hierarchical model is as following

$$Y_n|W \sim \mathcal{N}(0, \Sigma(W)) \qquad W \sim \mathcal{N}(0, \Sigma(Z)) \qquad Z \sim \mathcal{N}(0, \Sigma(X_n)) \tag{14}$$

The marginal likelihood involves three multivariate normal likelihoods with a double integral over Z and W

$$\mathcal{L}(Y_n|X_n) \propto \int \mathcal{L}(Y_n|W)\mathcal{L}(W|Z)\mathcal{L}(Z|X_n)dZdW \tag{15}$$

where the conditional log-likelihoods are again defined as following

$$\log \mathcal{L}(Y_n|W) \propto -\frac{1}{2}\log|\Sigma(W)| - \frac{1}{2}Y_n^T\Sigma(W)^{-1}Y_n \tag{16}$$

$$\log \mathcal{L}(W|Z) \propto -\frac{1}{2}\log|\Sigma(Z)| - \frac{1}{2}W^T\Sigma(Z)^{-1}W \tag{17}$$

$$\log \mathcal{L}(Z|X_n) \propto -\frac{1}{2}\log|\Sigma| - \frac{1}{2}Z^T\Sigma^{-1}Z \tag{18}$$

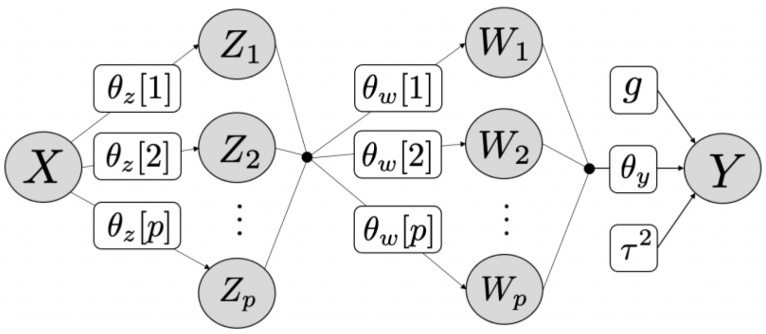
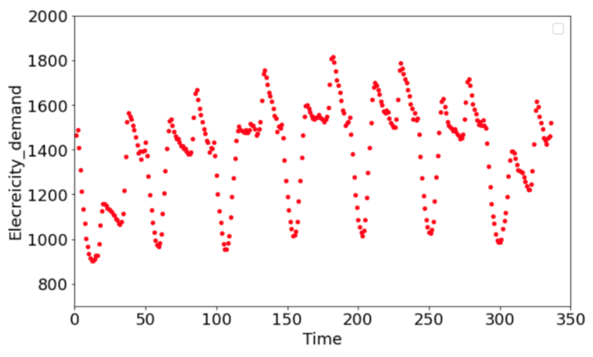


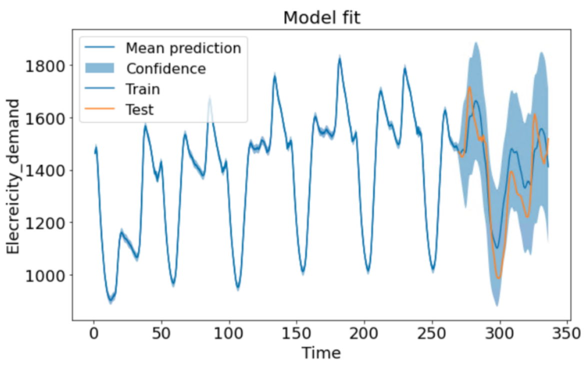
Figure 2: Model structure for a three-layer DGP with p latent nodes in both layers

4 Experiments

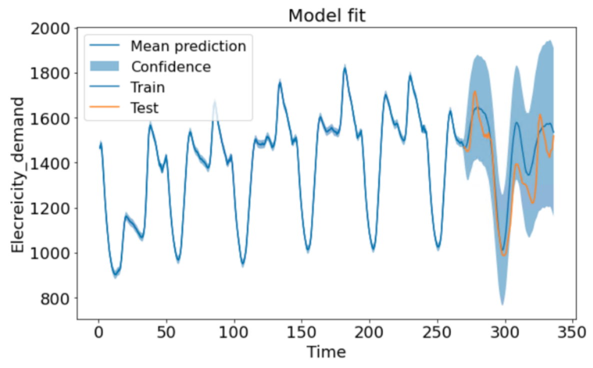
The dataset used for the experimentation of both GP and DGP contain the record of electricity demanded in seven consecutive days at a city in Australia. Here, each entry represents the demand of a thirty-minute duration; therefore, total 48 entries for each day. We split the data into train/test at 80%/20%. Figure 3a shows the several notable features in these data: short-term daily variations, long-term concave trends, and white noise artifacts. As seen in Figure 3b, an addictive kernel $k_1 = k_{\text{SM}} + k_{\text{Poly}} + k_{\text{RBF}}$ is used to fit the data, the SM kernel focus on learning the periodic contents in the data, the polynomial kernel is to model the concave trend and an RBF kernel to model local changes, but it kind of overestimates the amount of electricity demand. We also tried different kernels to model GP, a product kernel $k_2 = k_{\text{SM}} * k_{\text{Poly}} * k_{\text{RBF}}$ shown in Figure 3c. And $k_3 = k_{\text{Periodic}} * k_{\text{RBF}}$ shown in Figure 3d, the prediction can not capture the periodic changes and gradually return to the constant mean. In addition, we compare the Root Mean Square Error (RMSE) of these three GP models, the results are indicated in Table 1. We also tried other kernel compositions, however their performance are not as goog as we expected.



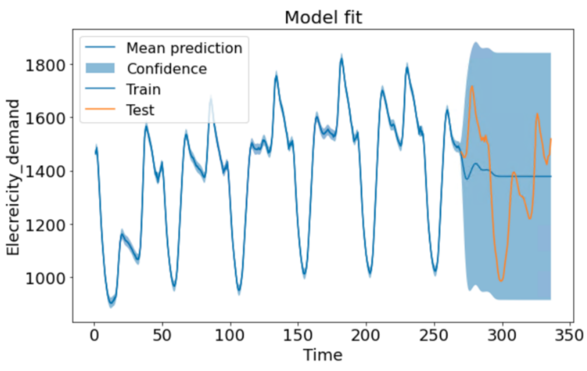
(a) 7 days electricity demand data



(b) Addictive kernel $k_1 = k_{\text{SM}} + k_{\text{Poly}} + k_{\text{RBF}}$



(c) Product kernel $k_2 = k_{\text{SM}} * k_{\text{Poly}} * k_{\text{RBF}}$



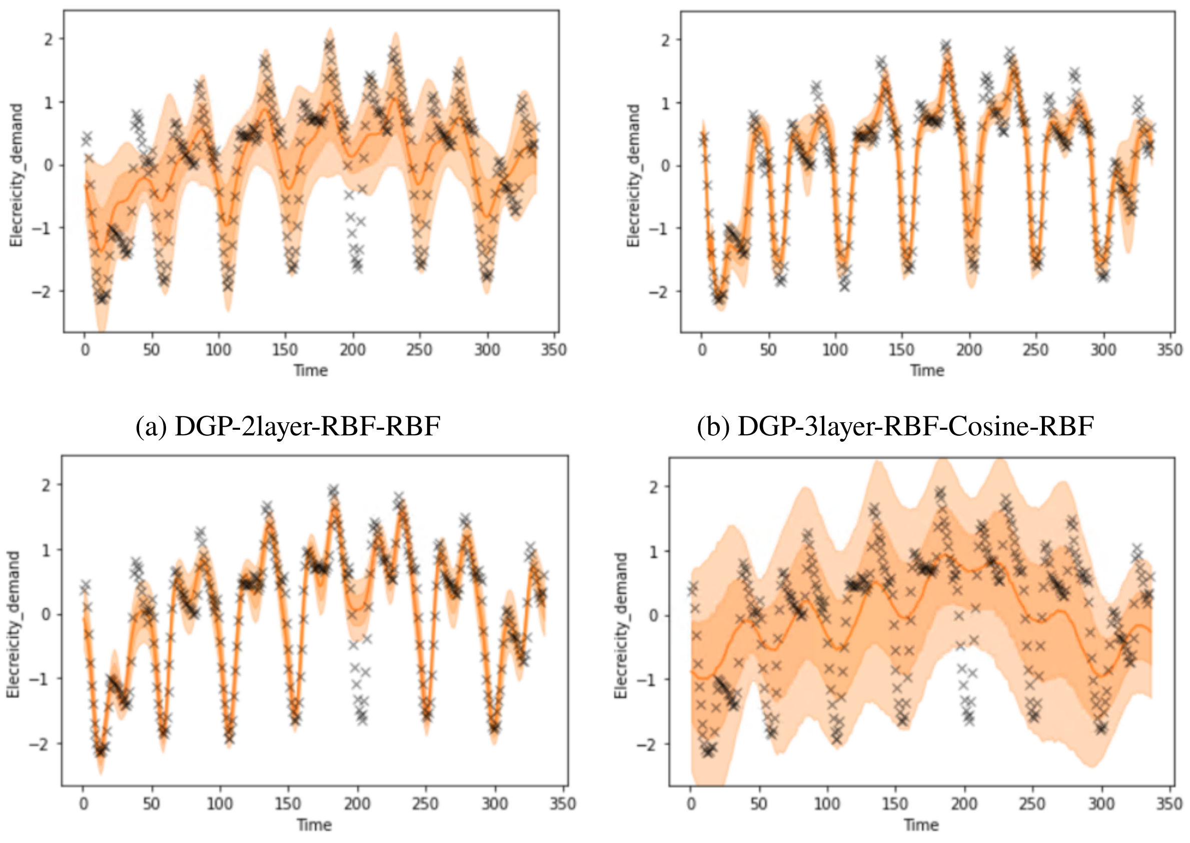
(d) Product kernel $k_3 = k_{\text{Periodic}} * k_{\text{RBF}}$

Figure 3: GP fit. (a) Raw data. (b), (c), (d) GP fit with some different kernel combinations.

	k_1	k_2	k_3
RMSE	100.041	108.772	183.702

Table 1: RMSE of GP Models.

For DGP method, we train the model with inducing variable approximation (inducing variables = 20) (Damianou & Lawrence, 2013), Figure 4a shows DGP model with two latent layers and RBF kernel for both layers after data normalization. The model kind of under-fitting but it does convey the concave trend. Whereas Figure 4b, where is a 3 layer DGP model with RBF kernel for the first and the last layer and a Cosine kernel for the second layer, which fits better than 2 layers model but still loses the structure of periodic trend. As seen in Figure 4c, the kernel applied to the middle layer is changed to an addictive kernel Cosine + Poly(2), and the model not only learned the period structure but also shows the curve trend. Adding an additional layer does not improve the overall performance but fails to learn the existing pattern, the plot in Figure 4d is far from Gaussian as four layers combined together.



(a) DGP-2layer-RBF-RBF

(b) DGP-3layer-RBF-Cosine-RBF

(c) DGP-3layer-RBF-Cosine+Poly-RBF

(d) DGP-4layer-RBF-Cosine-Poly-RBF

Figure 4: DGP fit. (a) (b), (c), (d) DGP fit with some different layers and kernel combinations.

5 Conclusion

We have introduced the basic framework of GP and DGP. The approaches we used performs efficient Bayesian training on hierarchical Gaussian Process maps, allowing automatic discovery of structure in hierarchies. The method is able to successfully learn feature hierarchies describing city 7-day electricity demand. Despite the relative scarcity of data in our experiments (336 data points), our variational lower bounds select deep hierarchical representations for the electricity demand. Our experiment results provide convincing evidence that DGP models are sufficiently powerful to encode abstract information for smaller datasets.

References

Cutajar, K., Bonilla, E. V., Michiardi, P., & Filippone, M. (2017). Random feature expansions for deep gaussian processes. In *International conference on machine learning* (pp. 884–893).

Damianou, A., & Lawrence, N. D. (2013). Deep gaussian processes. In *Artificial intelligence and statistics* (pp. 207–215).

Dunlop, M. M., Girolami, M. A., Stuart, A. M., & Teckentrup, A. L. (2018). How deep are deep gaussian processes? *Journal of Machine Learning Research*, *19*, 1–46.

Dutordoir, V., Hensman, J., van der Wilk, M., Ek, C. H., Ghahramani, Z., & Durrande, N. (2021). Deep neural networks as point estimates for deep gaussian processes. *Advances in Neural Information Processing Systems*, *34*, 9443–9455.

Duvenaud, D., Rippel, O., Adams, R., & Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *Artificial intelligence and statistics* (pp. 202–210).

Lindgren, F., Rue, H., & Lindström, J. (2011). An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*(4), 423-498.

Paciorek, C. J. (2003). Nonstationary gaussian processes for regression and spatial modelling. *ProQuest Dissertations and Theses*, 231.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning* (pp. 63–71).

Salimbeni, H., & Deisenroth, M. (2017). Doubly stochastic variational inference for deep gaussian processes. *Advances in neural information processing systems*, *30*.

Sauer, A., Cooper, A., & Gramacy, R. B. (2022). Vecchia-approximated deep gaussian processes for computer experiments. *arXiv preprint arXiv:2204.02904*.

Sauer, A., Gramacy, R. B., & Higdon, D. (2022). Active learning for deep gaussian process surrogates. *Technometrics*, 1–15.

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Wilson, A., & Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning* (pp. 1067–1075).