Deep Gaussian Processes

Adam Buckley

February 2026

Deep Gaussian Processes

Deep

Hierarchical composition of multiple layers. Similar to deep neural networks, but with GPs as building blocks

Gaussian Processes

Non-parametric Bayesian models that provide predictions and uncertainty estimates. [14]

Presentation Outline

1. **Gaussian Processes (GPs)**
   - Deriving GPs
   - Defining Gaussian Processes
   - Advantages and limitations of GPs
2. **Deep GPs (DGPs)**
   - Why GP limitations motivate DGPs
   - Architecture & composition
   - Advantages and limitations of DGPs
   - Inference Intractability
3. **Training GPs & DGPs**
   - Variational inference
   - Inducing points
   - Doubly Stochastic Variational Inference
   - Alternative approaches to DGPs
4. **Credit Risk Modelling**
   - What is credit risk?
   - Why GPs/DGPs?
   - Results & comparisons

Towards Gaussian Processes

Suppose we have data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$ and want a function $f : \mathbb{R}^D \to \mathbb{R}$ such that $f(\mathbf{x}_i) \approx y_i$ for all i.

Normally, we might choose linear regression:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^D$ is our weight vector.

This is overly restrictive as assumes linearity.

Towards Gaussian Processes

Using basis functions $\phi_j(\mathbf{x})$, we get more flexibility [1]:

$$f(\mathbf{x}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}, \tag{2}$$

where $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x})]^\top$.

- We can introduce non-linearity through a choice of basis functions.
- Still have to choose fixed number and form of basis functions $\phi_j(\cdot)$.

From Basis Functions to Gaussian Processes

Suppose instead of treating $\mathbf{w}$ as point estimates, we place a Gaussian prior over weights [11]:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w) \tag{3}$$

This induces a prior over functions $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$.
For any finite collection of inputs $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, the function values are jointly Gaussian:

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^\top\right) \tag{4}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)]^\top$ is the $N \times M$ design matrix.

From Basis Functions to Gaussian Processes

The function distribution $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ depends on $\mathbf{\Phi}$ and $\mathbf{\Sigma}_w$ *only through* the kernel matrix:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Sigma}_w\mathbf{\Phi}^\top \quad \text{where} \quad K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \text{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] \tag{5}$$

This *kernel* or *covariance function* $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \mathbf{\Sigma}_w \phi(\mathbf{x}')$ [17] captures how similar two function values should be based on their inputs.

We never need $\mathbf{\Phi}$ or $\mathbf{w}$ individually — only their combined effect via $k(\cdot, \cdot)$.

Gaussian Processes

Since we only need $k(\cdot, \cdot)$, we can bypass basis functions entirely.

Instead of a distribution over *parameters* $\mathbf{w}$, we now have a distribution over *functions* $f(\cdot)$.

- Parametric view: uncertain on weights $\mathbf{w}$. Optimise weights to find a best set, which uniquely defines a single function.
- GP view: uncertain about functions $f(\cdot)$. Optimise the kernel $k(\cdot, \cdot)$ to find a best set of hyperparameters. Defines a distribution over functions rather than a single function.

Defining Gaussian Processes

Definition: A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution [14]. Formally, a GP is defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{6}$$

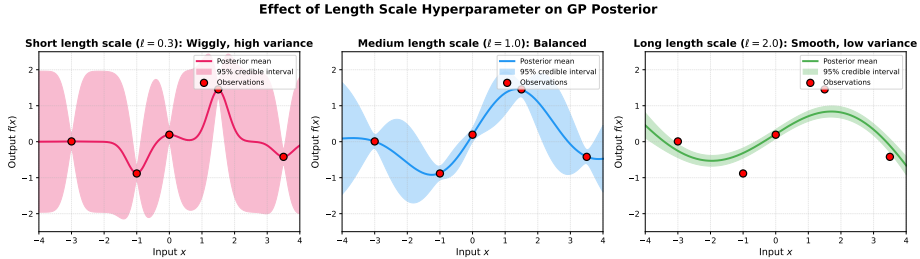
This means that for any finite set of inputs $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, the corresponding function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ are jointly Gaussian:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}), \tag{7}$$

where $\mathbf{K}$ is the covariance matrix defined by a kernel function $k(\cdot, \cdot)$ evaluated at the inputs.

Kernel Hyperparameters: Length Scale

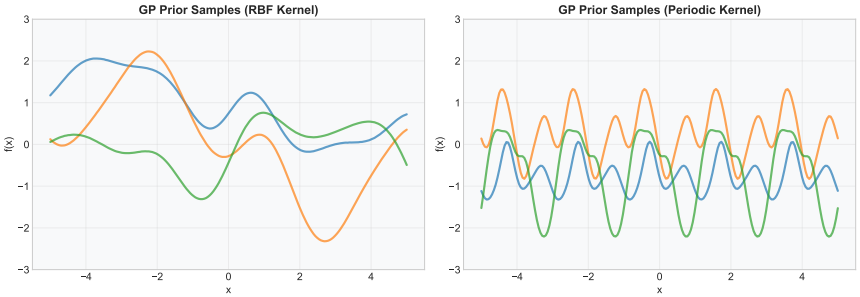
We mention before how Kernel choice encodes assumptions about function properties. Hyperparameters of kernels control these assumptions.



- In RBF kernel, the length scale ℓ controls how quickly correlation decays with distance.
- Short: wiggly, fits data closely; Long: smooth, broader generalizations
- Often determined by maximizing marginal likelihood or domain knowledge.

GP Prior: Distribution Over Functions

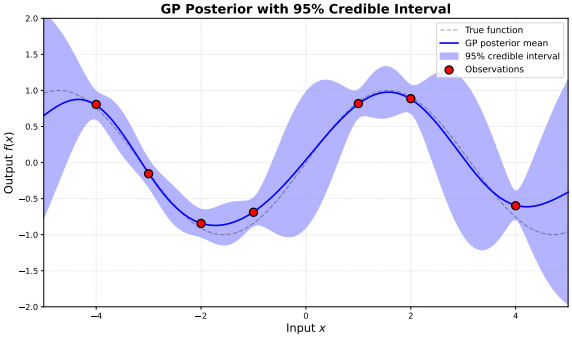
The GP prior $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$ captures assumptions about properties of functions, such as smoothness or periodicity, encoded by the kernel.



- Each line is a random function drawn from the GP prior
- All functions are plausible before seeing any data

GP Posterior: Learning From Data

Using Bayes rule, we condition the GP prior on observations to get the posterior: $p(f|\mathcal{D}) \propto p(\mathcal{D}|f)p(f)$.



- GP learns from noisy observations to predict the underlying function
- Uncertainty (shaded region) is small near data, large far from data
- Prior is updated by data to give a posterior distribution that captures what is learnt about the function.

Gaussian Processes: Summary

Advantages

- Non-parametric flexibility
- Automatic uncertainty quantification
- Exact inference (When have Gaussian likelihood)
- Kernel encodes assumptions [5]

Limitations

- $O(N^3)$ scalability [14]
- Limited expressiveness
- Struggles with non-stationarity [12]
- Manual kernel selection

$\Rightarrow$ **Motivates Deep GPs**

Deep Gaussian Processes: Motivation

- To solve the issues mentioned for GPs, we need a more flexible model.
- Aim to learn hierarchical representations of data
- Aim to avoid requirements for manual kernel design
- Aim to capture complex, non-stationary functions

Idea: Similar to deep neural networks [20], why not stack GPs on top of each other, where the output of one GP becomes the input to the next?

$$\mathbf{X} \xrightarrow{f^{(1)}} \mathbf{F}^{(1)} \xrightarrow{f^{(2)}} \mathbf{F}^{(2)} \xrightarrow{\cdots} \mathbf{F}^{(L)} \rightarrow \mathbf{y} \tag{8}$$

Even when each layer is stationary (e.g., RBF kernel), the composition becomes non-stationary, allowing for modelling of more complex functions [3].

DGP: Definition

DGP definition [3, 15]: A DGP with L layers is a hierarchical composition where each layer ℓ is a GP conditioned on the previous layer's output:

$$f^{(\ell)} \mid \mathbf{F}^{(\ell-1)} \sim \mathcal{GP}\left(0, k^{(\ell)}(\cdot,\cdot)\right), \quad \ell = 1, \ldots, L. \tag{9}$$

This gives:

$$\mathbf{F}^{(1)} = f^{(1)}(\mathbf{X}), \qquad f^{(1)} \sim \mathcal{GP}(0, k^{(1)}), \tag{10}$$

$$\mathbf{F}^{(2)} = f^{(2)}(\mathbf{F}^{(1)}), \qquad f^{(2)} \sim \mathcal{GP}(0, k^{(2)}), \tag{11}$$

$$\vdots$$

$$\mathbf{F}^{(L)} = f^{(L)}(\mathbf{F}^{(L-1)}), \qquad f^{(L)} \sim \mathcal{GP}(0, k^{(L)}), \tag{12}$$

$$\mathbf{y} = \mathbf{F}^{(L)} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I}). \tag{13}$$

Each $f^{(\ell)}$ learns features from $\mathbf{F}^{(\ell-1)}$, with $\mathbf{F}^{(0)} = \mathbf{X}$.

Deep Gaussian Processes: Summary

Advantages

- Increased expressiveness [3]
- Models non-stationarity
- Hierarchical feature learning [16]
- Automatic kernel composition
- Uncertainty quantification

Challenges

- Intractable inference
- Non-Gaussian posteriors
- High computational cost
- Non-convex optimization [6]
- Careful initialization needed

Intractability of Exact Inference

Exact inference is intractable. The marginal likelihood requires integrating out all latent functions:

$$p(\mathbf{y} \mid \mathbf{X}) = \int p(\mathbf{y} \mid \mathbf{F}^{(L)}) \prod_{\ell=1}^{L} p(\mathbf{F}^{(\ell)} \mid \mathbf{F}^{(\ell-1)})\, d\mathbf{F}^{(1)} \cdots d\mathbf{F}^{(L)}, \tag{14}$$

which has no closed-form solution for $L \geq 2$.
The posterior over all latent functions is therefore intractable:

$$p(\mathbf{F}^{(1:L)} \mid \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{F}^{(L)}) \prod_{\ell=1}^{L} p(\mathbf{F}^{(\ell)} \mid \mathbf{F}^{(\ell-1)})}{p(\mathbf{y} \mid \mathbf{X})}. \tag{15}$$

Since both the numerator (a product of coupled Gaussians) and denominator (intractable marginal likelihood) often do not have closed forms.

Therefore how do we train DGPs?

Scalable Training via Variational Inference

Solution: Variational inference [9, 2] - approximate the intractable posterior $p(\mathbf{f}|\mathbf{y})$ with a simpler distribution $q(\mathbf{f})$

Evidence Lower Bound for GPs:

$$\log p(\mathbf{y}|\mathbf{X}) \geq \mathbb{E}_{q(\mathbf{f})}\left[\log p(\mathbf{y}|\mathbf{f})\right] - \mathrm{KL}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{X})) = \mathcal{L}$$

Maximize $\mathcal{L}$ to make $q(\mathbf{f})$ close to the true posterior $p(\mathbf{f}|\mathbf{y})$

Remaining problem: Optimizing over all N function values still gives $O(N^3)$ complexity

$\Rightarrow$ Need inducing points to reduce dimensionality from N to $M \ll N$

Sparse Variational Inference with Inducing Points

Inducing points [19, 8]: $M \ll N$ pseudo-inputs $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$ with function values $\mathbf{u} = f(\mathbf{Z})$ that summarize the dataset

Key idea: Parametrize $q(\mathbf{f})$ through inducing variables [13]:

$$q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u})\, d\mathbf{u} \quad \text{where} \quad q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$$

Optimize only M variational parameters $(\mathbf{m}, \mathbf{S})$ instead of N function values

Sparse GP ELBO:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{f})} [\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u}))$$

Complexity reduction: $O(N^3) \to O(NM^2)$
Choose M based on computational budget (e.g., $M = 100$ to 1000)

Training DGPs: Multi-Layer Variational Inference

Extension to DGPs: Introduce inducing points $\mathbf{u}_l$ at each layer $l = 1, \ldots, L$

DGP ELBO:

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{f}^{(1:L)})}\left[\log p(\mathbf{y}|\mathbf{f}^{(L)})\right]}_{\text{Data fit}} - \underbrace{\sum_{l=1}^{L} \mathrm{KL}(q(\mathbf{u}_l)||p(\mathbf{u}_l))}_{\text{Regularization per layer}}$$

where $q(\mathbf{f}^{(l)}) = \int p(\mathbf{f}^{(l)}|\mathbf{u}_l)q(\mathbf{u}_l)\, d\mathbf{u}_l$ for each layer

Optimize jointly:

- Variational parameters $\{\mathbf{m}_l, \mathbf{S}_l\}_{l=1}^{L}$ for each layer
- Inducing locations $\{\mathbf{Z}_l\}_{l=1}^{L}$ and kernel hyperparameters $\{\theta_l\}_{l=1}^{L}$
- Use stochastic gradient descent with mini-batches for scalability

Key challenge: Propagating uncertainty through all L layers (handled via sampling)

Doubly Stochastic Variational Inference

Solution: Doubly stochastic optimization - use two levels of stochasticity:

1. **Mini-batch sampling:** Randomly sample $B \ll N$ data points

$$\mathbb{E}_{q(\mathbf{f}^{(1:L)})}\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{f}_i^{(L)})\right] \approx \frac{N}{B} \sum_{i \in \text{batch}} \mathbb{E}_{q(\mathbf{f}_i^{(1:L)})}\left[\log p(y_i|\mathbf{f}_i^{(L)})\right]$$

2. **Monte Carlo sampling:** Sample from $q(\mathbf{f}^{(l)})$ to estimate expectations using the reparameterization trick [10]

$$\mathbb{E}_{q(\mathbf{f}_i^{(1:L)})}\left[\log p(y_i|\mathbf{f}_i^{(L)})\right] \approx \frac{1}{S} \sum_{s=1}^{S} \log p(y_i|\mathbf{f}_i^{(L)(s)})$$

Result: Unbiased gradient estimates enabling scalable SGD for DGPs [16]

Alternative DGP Formulations

Beyond the standard composition-based DGPs, other formulations exist:

Thin and Deep GPs: [18]

- Propagate *weights* through GP layers instead of outputs
- Each layer: $\mathbf{w}^{(l)} \sim \mathrm{GP}(\cdot)$, then $\mathbf{f}^{(l)} = \mathbf{w}^{(l)}\mathbf{f}^{(l-1)}$
- More parameter-efficient but less flexible than standard DGPs
- Can help mitigate pathologies in deep compositions (e.g., degeneracy / manifold collapse) [4]

Deeply Non-Stationary GPs [12, 7]:

- Use GPs to modulate kernel hyperparameters (e.g., lengthscales) spatially
- Creates non-stationary covariance structure via deep composition
- Particularly useful when data characteristics vary across input space

Using GPs and DGPs for Credit Risk Modeling

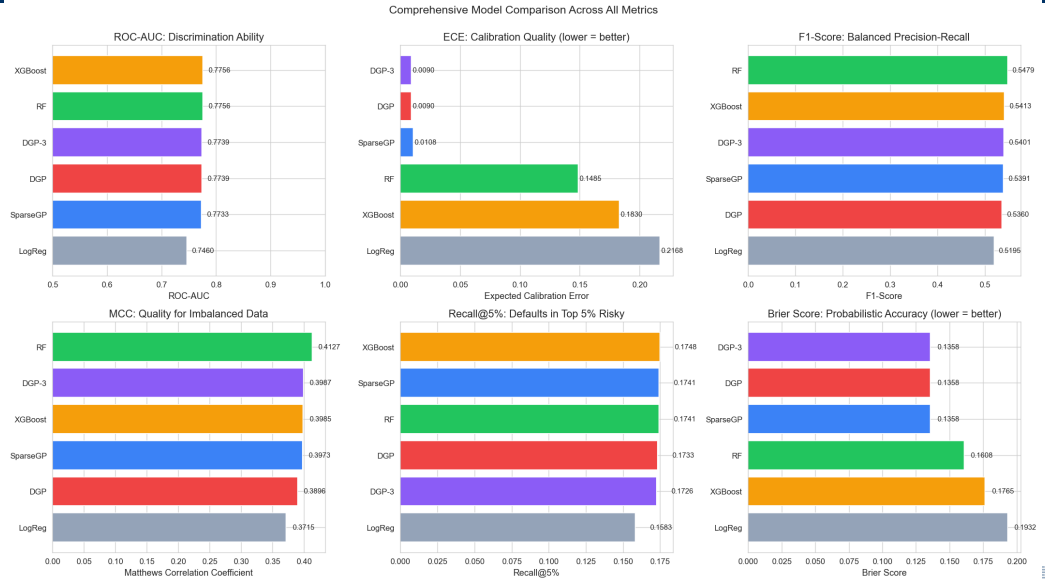
The goal in this application is to predict the probability of default for a customer, which can be used to inform intervention strategies (e.g., offering payment plans, credit line adjustments) to reduce losses from defaults.

Early Results: Credit Risk Modelling

- UCI Credit Card Dataset: 30,000 transactions with 23 features, binary labels for fraud detection. (80% train, 20% test split)
- We trained a single-layer GP and two DGP architectures (2-layer and 3-layer) using doubly stochastic variational inference. Comparison standard ML models included as baselines.

Full results, code and analysis will be included in final dissertation. Here we present a summary of key findings from very early results.

Early Results: Credit Risk Modelling (Cont.)



Hypothetical Scenario: Simple Demonstration of Potential Impact

FP Cost = \$500 (investigation cost), TP Revenue = \$25,000 (\$30k balance × 85% LGD), 6000 customers/month

Net Benefit vs Random Sampling (20% Intervention Rate):

Model	Net Benefit/month	vs Random	Lift
Random	\$6,157,500	—	—
LogReg	\$15,796,500	+\$9,639,000	+156.5%
RF	\$16,561,500	+\$10,404,000	**+169.0%**
XGBoost	\$16,357,500	+\$10,200,000	+165.7%
SparseGP	\$16,332,000	+\$10,174,500	+165.2%
DGP (2-layer)	\$16,357,500	+\$10,200,000	+165.7%
DGP (3-layer)	\$16,332,000	+\$10,174,500	+165.2%

Key Insight: All ML models deliver ˜165% lift over random sampling
GP models competitive with tree-based methods (\$10M+ incremental value)

Optimal Intervention Strategy

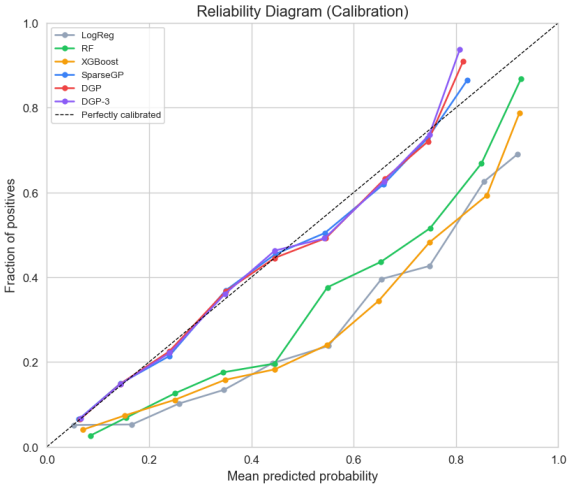
Net benefit varies with intervention volume

Intervention	Best Model	Net Benefit/Month	Lift vs Random
Top 5%	XGBoost	\$5,766,000	+276.1%
Top 10%	SparseGP	\$10,410,000	+239.5%
Top 15%	XGBoost	\$13,881,000	+200.2%
Top 20%	RF	\$16,561,500	+169.0%
Top 30%	RF	\$20,239,500	+118.8%
Top 40%	**DGP-3**	\$22,795,500	+85.1%
Top 50%	XGBoost	\$25,147,500	+63.2%

Limitations: Assumptions on FP cost, TP revenue, and customer volume are simplified. We assume predicting default always saves a fixed cost. We do not account for the customer relationship effects from interventions. These results are illustrative of potential value in a hypothetical and simplified scenario and are not close to realistic estimates of actual financial impact. This is purely for model-to-model comparison.

Calibration Quality: A Critical Advantage

- The GPs give near-perfect calibration (ECE < 0.011) while tree-based models are poorly calibrated (ECE 0.15–0.22). This means GP probabilities are trustworthy, critical for interpretability and decision-making in regulated environments.



Key Takeaways

1. GPs/DGPs deliver competitive business value, but all models have similar classification performance.

2. Calibration massively better in GPs and DGPs

- ECE $< 1\%$ vs 15-22% for tree models. Huge difference in probability quality.
- Probabilities are *trustworthy* without recalibration
- Critical for regulated environments (banking, healthcare)

Questions?

References I

[1] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. ISSN 0162-1459. doi: 10.1080/01621459.2017.1285773.

[3] Andreas C. Damianou and Neil D. Lawrence. Deep Gaussian Processes, March 2013. URL `http://arxiv.org/abs/1211.0358`. arXiv:1211.0358 [stat].

[4] Matthew M Dunlop, Mark A Girolami, Andrew M Stuart, and Aretha L Teckentrup. How Deep Are Deep Gaussian Processes? *Journal of Machine Learning Research*, 19(54):1–46, 2018.

[5] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1166–1174. PMLR, 2013.

References II

[6] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks, February 2014. URL http://arxiv.org/abs/1402.5836. arXiv:1402.5836 [stat].

[7] Markus Heinonen, Henrik Mannerstrom, Juho Rousu, Samuel Kaski, and Harri Lahdesmaki. Non-Stationary Gaussian Process Regression with Hamiltonian Monte Carlo. In *Artificial Intelligence and Statistics (AISTATS)*, pages 732–740. PMLR, 2016.

[8] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290. AUAI Press, 2013.

[9] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2): 183–233, 1999. ISSN 0885-6125. doi: $10.1023/\mathrm{A}{:}1007665907178$.

[10] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [stat].

References III

[11] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992. doi: 10.1162/neco.1992.4.3.448. URL https://authors.library.caltech.edu/records/8n76z-g9k64.

[12] Christopher J. Paciorek and Mark J. Schervish. Nonstationary Covariance Functions for Gaussian Process Regression. In *Advances in Neural Information Processing Systems*, volume 16, pages 273–280. MIT Press, 2004.

[13] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. In *Journal of Machine Learning Research*, volume 6, pages 1939–1959, 2005.

[14] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., 3. print edition, 2008. ISBN 978-0-262-18253-9.

[15] Hugh Salimbeni. *Deep Gaussian Processes: Advances in Models and Inference*. PhD thesis, Imperial College London, 2019.

References IV

[16] Hugh Salimbeni and Marc Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes, November 2017. URL `http://arxiv.org/abs/1705.08933`. arXiv:1705.08933 [stat].

[17] Bernhard Schölkopf. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2002. ISBN 978-0-262-19475-4 978-0-262-25693-3 978-0-585-47759-6.

[18] Daniel Augusto de Souza, Alexander Nikitin, S. T. John, Magnus Ross, Mauricio A. Álvarez, Marc Peter Deisenroth, João P. P. Gomes, Diego Mesquita, and César Lincoln C. Mattos. Thin and Deep Gaussian Processes, October 2023. URL `http://arxiv.org/abs/2310.11527`. arXiv:2310.11527 [stat].

[19] Michalis Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 567–574. PMLR, 2009.

References V

[20] Christopher Williams. Computing with Infinite Networks. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper_files/paper/1996/file/ae5e3ce40e0404a45ecacaaf05e5f735-Paper.pdf.