Variational Implicit Processes

Chao Ma[1] **Yingzhen Li**[2] **José Miguel Hernández-Lobato**[12]
[1]University of Cambridge [2]Microsoft Research Cambridge
{cm905,jmh233}@cam.ac.uk, Yingzhen.Li@microsoft.com

Abstract

We introduce the variational implicit processes (VIPs), a principled Bayesian modeling technique based on a class of highly flexible priors over functions. Similar to Gaussian processes (GPs), in implicit processes (IPs), an implicit multivariate prior (data simulators, Bayesian neural networks, non-linear transformations of stochastic processes, etc.) is placed over any finite collections of random variables. A novel and efficient approximate inference algorithm for IPs is derived using wake-sleep updates, which gives analytic solutions and allows scalable hyper-parameter learning with stochastic optimization. Experiments demonstrate that VIPs return better uncertainty estimates and superior performance over existing inference methods for GPs and Bayesian neural networks.

1 Introduction

Powerful models with *implicit distributions* as core components have recently attracted enormous interest in both deep learning as well as the approximate Bayesian inference community. In contrast to *prescribed probabilistic models* [24] that assign *explicit* densities to possible outcomes of the model, implicit models *implicitly assign* probability measures by the specification of the *data generating process*. One of the most well known implicit distributions is the generator of generative adversarial nets (GANs) [2, 9, 34, 73, 82, 107] that transforms isotropic noise into high dimensional data, say images, using neural networks. In approximate inference context, implicit distributions have also been deployed to postulate flexible approximate posterior distributions [61, 63, 65, 66, 85, 101, 104]. However, such generation process does not necessarily allow evaluation of densities point-wise, which becomes the main barrier for inference and learning.

This paper explores applications of implicit models to Bayesian modeling of *random functions*. Similar to the construction of Gaussian processes (GPs), we construct *implicit stochastic processes (IPs)* by assigning implicit distributions over any finite collections of random variables. Recall that a GP defines the distribution of a random function f by placing a multivariate Gaussian distribution $\mathcal{N}(\mathbf{f}; \mathbf{m}, \mathbf{K_{ff}})$ over any finite collection of function values $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))^\top$ evaluated at any given finite collection of input locations $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$.[1] An alternative parameterization of GPs defines the sampling process as $\mathbf{f} \sim \mathcal{N}(\mathbf{f}; \mathbf{m}, \mathbf{K_{ff}}) \Leftrightarrow \mathbf{z} \sim \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$, $\mathbf{f} = \mathbf{Bz} + \mathbf{m}$, with $\mathbf{K_{ff}} = \mathbf{BB}^\top$ the Cholesky decomposition of the covariance matrix. Observing this, we propose a generalization of the generative process by replacing the linear transform of the latent variable $\mathbf{z}$ with a nonlinear one. This gives the formal definition of implicit stochastic process as follows:

Definition 1 (noiseless implicit stochastic processes)**.** *An implicit stochastic process (IP) is a collection of random variables $f(\cdot)$, such that any finite collection $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))^\top$ has a joint distribution implicitly defined by the following generative process*

$$\mathbf{z} \sim p(\mathbf{z}), \quad f(\mathbf{x}_n) = g_\theta(\mathbf{x}_n, \mathbf{z}), \quad \forall\, \mathbf{x}_n \in \mathbf{X}. \tag{1}$$

A function distributed according to the above IP is denoted as $f(\cdot) \sim \mathcal{IP}(g_\theta(\cdot,\cdot), p_\mathbf{z})$.

[1]$\mathbf{x}_n$ can also be unobserved as in Gaussian process latent variable models [53].

Note that $\mathbf{z} \sim p(\mathbf{z})$ could be an infinite dimensional object such as a Gaussian Process (in this case, g becomes an operator). This definition of an IP is well-defined, in which we provide theoretical justifications in Appendix C.2 and C.3.

Many powerful models, including neural samplers, warped Gaussian Processes [96] and Bayesian Neural Networks, can be represented by an IP (see Appendix C). With an IP as the prior, one can directly perform posterior inference over functions in a non-parametric fashion, which avoids typical issues of Bayesian inference in parameter space (e.g. symmetric modes in the posterior distribution of Bayesian neural network *weights*). Therefore IPs bring together the best of both worlds, that is, combining the elegance of inference, and the well-calibrated uncertainty of Bayesian nonparametric methods, with the strong representational power of implicit models.

However, standard approximate Bayesian inference techniques are non-applicable as the underlying distribution of an IP is intractable. As the main contribution of the paper, we develop a novel and fast variational framework that gives a closed-form approximation to the intractable IP posterior. We conduct experiments to compare IPs with the proposed inference method, and GPs/Bayesian neural networks/Bayesian RNNs with existing variational approaches.

2 Variational implicit processes

Consider the following regression model with an IP prior over the function:[2]

$$f(\cdot) \sim \mathcal{IP}(g_\theta(\cdot, \cdot), p_{\mathbf{z}}),\ y = f(\mathbf{x}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{2}$$

Equation (2) defines an implicit model as $p(\mathbf{y}, \mathbf{f}|\mathbf{x})$ is intractable in most cases. Given an observational dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and a set of test inputs $\mathbf{X}_*$, Bayesian predictive inference over $\mathbf{y}_*$ involves computing the predictive distribution $p(\mathbf{y}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}, \theta)$, which itself implies computing the posterior $p(f|\mathbf{X}, \mathbf{y}, \theta)$. Besides prediction, we may also want to learn the prior parameters θ and noise variance σ by maximizing the log marginal likelihood: $\log p(\mathbf{y}|\mathbf{X}, \theta) = \log \int_{\mathbf{f}} p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f}$, with $\mathbf{f}$ the evaluation of f on $\mathbf{X}$. Unfortunately, both the prior $p(\mathbf{f}|\mathbf{X}, \theta)$ and the posterior $p(f|\mathbf{X}, \mathbf{y}, \theta)$ are intractable as the implicit process does not allow point-wise density evaluation, let alone the integration tasks. Therefore, to address these tasks, we must resort to approximate inference methods.

We propose a generalization of the *wake-sleep* algorithm [43] to handle both intractabilities. This method returns (i) an approximate posterior distribution $q(f|\mathbf{X}, \mathbf{y})$ which is later used for predictive inference, and (ii) an approximation to the marginal likelihood $p(\mathbf{y}|\mathbf{X}, \theta)$ for hyper-parameter optimization. In this paper we choose $q(f|\mathbf{X}, \mathbf{y})$ to be the posterior distribution of a Gaussian process, $q_{\mathcal{GP}}(f|\mathbf{X}, \mathbf{y})$. Due to page limit we present the full algorithm in Appendix D, and a high-level summary of our algorithm is the following:

- *Sleep phase*: sample *dreamed* data (function values $\mathbf{f}$ and noisy outputs $\mathbf{y}$) as indicated in (2), and use them fit a GP regression model $q_{\mathcal{GP}}(\mathbf{y}, \mathbf{f}|\mathbf{X}) = p(\mathbf{y}|\mathbf{f}, \mathbf{X}) q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X})$ with maximum likelihood. This is equivalent to minimizing $\mathrm{D}_{\mathrm{KL}}[p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}, \mathbf{f}|\mathbf{X})]$ that is also a variational upper bound of $\mathrm{D}_{\mathrm{KL}}[p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y})]$. It has an analytic solution: the optimal GP $q_{\mathcal{GP}}(f)$ has the same mean and covariance functions as the IP prior.
- *Wake phase*: approximate the IP posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)$ with the GP posterior $q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y})$, and optimize hyper-parameters θ by maximizing the approximated marginal likelihood $\log p(\mathbf{y}|\mathbf{X}, \theta) \approx \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X})$. In Appendix D.2, this optimization task is further reduced to Bayesian linear regression (with hyper parameters being optimized simultaneously) using α-variational inference.

We name this inference framework as the *variational implicit processes* (VIPs). Our approach has two key advantages. First, the algorithm has no explicit sleep phase computation, and the sleep phase's analytic solution can be directly plugged into the wake-phase objective. Second, the proposed wake phase update is highly scalable: with stochastic optimization techniques, the Bayesian linear regression task has the same order of time complexity as training Bayesian neural networks. Crucially, our wake-sleep algorithm does not require the evaluation of the implicit prior distribution. Therefore the intractability of IP prior distributions is no longer an obstacle for approximate inference, and our inference framework can be applied to many implicit process models.

[2]Note that it is common to add Gaussian noise ϵ to an implicit model, e.g. see the noise smoothing trick used in training GANs [89, 97].

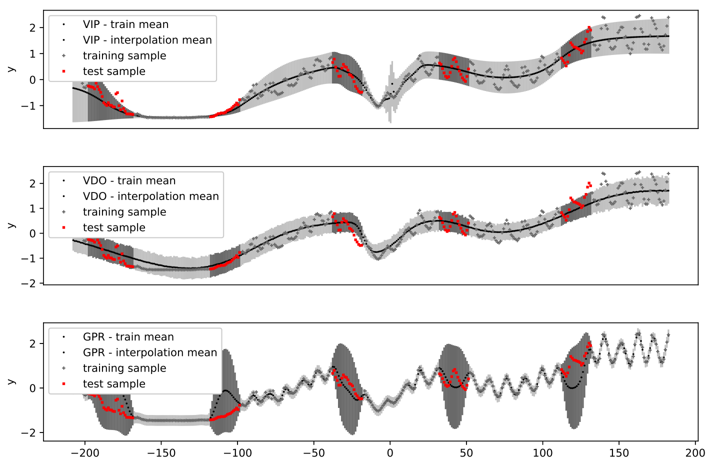

Figure 1: Interpolations returned by VIP (**top**), variational dropout (**middle**), and exact GP (**bottom**), respectively. SVGP is omitted as it looks nearly the same. **Grey dots**: training data, **red dots**: test data, **dark dots**: predictive means, **light grey and dark grey areas**: Confidence intervals with 2 standard deviations of the training and test set, respectively.

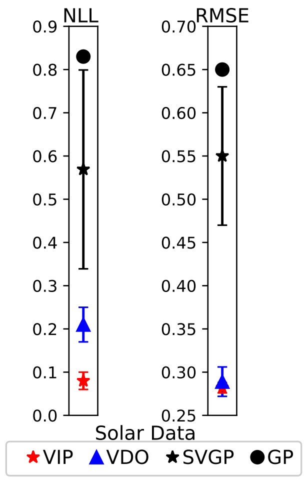

Figure 2: Test performance on solar irradiance interpolation. The lower the better.

3 Experiments

Solar irradiance prediction We compare the VIP with variational sparse GP (SVGP, 100 inducing points), exact GP and variational dropout Bayesian NN (VDO) on the solar irradiance dataset [57]. The dataset is constructed following [31], where 5 segments of length 20 are removed for interpolation. All the inputs are then centered, and the targets are standardized. We use the same general settings as specified in Appendix E.1, except that we run Adam with learning rate = 0.001 for 5000 iterations.

Predictive interpolations are shown in Figure 1. We see that VIP and VDO give similar interpolation behaviors. However, VDO overall under-estimates uncertainty when compared with VIP, especially in the interval $[-100, 200]$. VDO also incorrectly estimates the mean function around $x = -150$ where the ground truth there is a constant. On the contrary, VIP is able to recover the correct mean estimation around this interval with high confidence. GP methods recover the exact mean of training data with high confidence, but they return poor estimates of predictive means for interpolation. Quantitatively, the right two plots in Figure 2 show that VIP achieves the best NLL/RMSE performance, again indicating that its returns high-quality uncertainties and accurate mean predictions.

Further experiments We have further conducted comprehensive experiments in Appendix E, including a synthetic example (Sec E.1), multivariate regression on UCI datasets (Sec E.2), approximate Bayesian computation (ABC) on the Lotka–Volterra implicit model (Sec E.3), and Bayesian LSTM for predicting power conversion efficiency of organic photovoltaics molecules (Sec E.4). These results demonstrate that VIPs return better uncertainty estimates and superior performance over existing inference methods for GPs and Bayesian NNs.

4 Conclusions

We presented a variational approach for learning and Bayesian inference over function space based on implicit process priors. It provides a powerful framework that combines the rich representational power of implicit models with the well-calibrated uncertainty estimates from (parametric/nonparametric) Bayesian models. As an example, with Bayesian neural networks as the implicit process prior, our approach outperformed many existing Gaussian process/Bayesian neural network methods and achieved significantly improved results on the a number of experiments. Many directions remain to be explored. Classification models with implicit process priors will be developed. Implicit process latent variable models will also be derived in a similar fashion as Gaussian process latent variable models [53]. Future work will investigate novel inference methods to models equipped with other implicit process priors, e.g. data simulators in astrophysics, ecology and climate science.

References

[1] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1):2850–2886, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv:1701.07875*, 2017.

[3] Matej Balog, Balaji Lakshminarayanan, Zoubin Ghahramani, Daniel M Roy, and Yee Whye Teh. The mondrian kernel. *arXiv preprint arXiv:1606.05241*, 2016.

[4] David Barber and Christopher M Bishop. Ensemble learning in Bayesian neural networks. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 168:215–238, 1998.

[5] Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate bayesian computation. *Biometrika*, 96(4):983–990, 2009.

[6] Daniel Beck and Trevor Cohn. Learning kernels over strings using gaussian processes. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 67–73, 2017.

[7] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[8] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of dimensionality for local kernel machines. *Techn. Rep*, 1258, 2005.

[9] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv:1703.10717*, 2017.

[10] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv:1505.05424*, 2015.

[11] Fernando V Bonassi, Mike West, et al. Sequential monte carlo with adaptive weights for approximate bayesian computation. *Bayesian Analysis*, 10(1):171–187, 2015.

[12] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv:1707.02476*, 2017.

[13] 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.

[14] Thang D Bui and Richard E Turner. Tree-structured Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 2213–2221, 2014.

[15] Thang D Bui, Josiah Yan, and Richard E Turner. A unifying framework for sparse Gaussian process approximation using power expectation propagation. *arXiv:1605.07066*, 2016.

[16] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350, 2009.

[17] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM, 2008.

[18] John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. Fast gaussian process methods for point process intensity estimation. In *Proceedings of the 25th International Conference on Machine Learning*, pages 192–199. ACM, 2008.

[19] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep Gaussian processes. *arXiv:1610.04386*, 2016.

[20] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[21] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems*, pages 2253–2261, 2016.

[22] John Denker and Yann Lecun. Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems 3*. Citeseer, 1991.

[23] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *arXiv:1605.07127*, 2016.

[24] Peter J Diggle and Richard J Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 193–227, 1984.

[25] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[26] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *arXiv:1302.4922*, 2013.

[27] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[28] Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping gaussian process priors to bayesian neural networks. *NIPS Bayesian deep learning workshop*, 2017.

[29] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[30] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027, 2016.

[31] Yarin Gal and Richard Turner. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning*, pages 655–664, 2015.

[32] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[33] Amir Globerson and Roi Livni. Learning infinite-layer networks: beyond the kernel trick. arxiv preprint. *arXiv preprint arXiv:1606.05316*, 2016.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[35] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[36] William H Guss. Deep function machines: Generalized neural networks for topological layer expression. *arXiv preprint arXiv:1612.04799*, 2016.

[37] Johannes Hachmann, Roberto Olivares-Amaya, Adrian Jinich, Anthony L Appleton, Martin A Blood-Forsythe, Laszlo R Seress, Carolina Roman-Salgado, Kai Trepte, Sule Atahan-Evrenk, Süleyman Er, et al. Lead candidates for high-performance organic photovoltaics from high-throughput quantum chemistry–the harvard clean energy project. *Energy & Environmental Science*, 7(2):698–704, 2014.

[38] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv:1508.05133*, 2015.

[39] Uri Heinemann, Roi Livni, Elad Eban, Gal Elidan, and Amir Globerson. Improper deep kernels. In *Artificial Intelligence and Statistics*, pages 1159–1167, 2016.

[40] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv:1309.6835*, 2013.

[41] José Miguel Hernández-Lobato and Ryan P Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. *arXiv:1502.05336*, 2015.

[42] José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Daniel Hernández-Lobato, Thang Bui, and Richard Eric Turner. Black-box α-divergence minimization. 2016.

[43] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

[44] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[45] Geoffrey E Hinton and Ruslan R Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1249–1256, 2008.

[46] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13. ACM, 1993.

[47] Tomoharu Iwata and Zoubin Ghahramani. Improving output uncertainty estimation and generalization in deep learning via neural network Gaussian processes. *arXiv:1707.05922*, 2017.

[48] 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.

[49] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413, 2013.

[50] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[51] Karl Krauth, Edwin V Bonilla, Kurt Cutajar, and Maurizio Filippone. Autogp: Exploring the capabilities and limitations of Gaussian process models. *arXiv:1610.05392*, 2016.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[53] Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*, pages 329–336, 2004.

[54] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum gaussian process regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881, 2010.

[55] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8595–8598. IEEE, 2013.

[56] Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pages 404–411, 2007.

[57] Judith Lean, Juerg Beer, and Raymond Bradley. Reconstruction of solar irradiance since 1610: Implications for climate change. *Geophysical Research Letters*, 22(23):3195–3198, 1995.

[58] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. *arXiv:1711.00165*, 2017.

[59] Yingzhen Li and Yarin Gal. Dropout inference in Bayesian neural networks with alpha-divergences. *arXiv:1703.02914*, 2017.

[60] Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Stochastic expectation propagation. In *Advances in Neural Information Processing Systems*, pages 2323–2331, 2015.

[61] Yingzhen Li and Qiang Liu. Wild variational approximations.

[62] Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081, 2016.

[63] Yingzhen Li, Richard E Turner, and Qiang Liu. Approximate inference with amortised MCMC. *arXiv:1702.08343*, 2017.

[64] Moshe Lichman et al. Uci machine learning repository, 2013.

[65] Qiang Liu and Yihao Feng. Two methods for wild variational inference. *arXiv:1612.00081*, 2016.

[66] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, pages 2370–2378, 2016.

[67] Michel Loeve. In *Probability Theory I-II*. Springer, 1977.

[68] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[69] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv:1804.11271*, 2018.

[70] 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.

[71] Thomas Minka. Power EP. Technical report, Technical report, Microsoft Research, Cambridge, 2004.

[72] Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[73] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

[74] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[75] Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[76] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[77] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 1028–1036, 2016.

[78] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems*, pages 3360–3368, 2016.

[79] Edward O Pyzer-Knapp, Kewei Li, and Alan Aspuru-Guzik. Learning from the harvard clean energy project: The use of neural networks to accelerate materials discovery. *Advanced Functional Materials*, 25(41):6495–6502, 2015.

[80] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[81] 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.

[82] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[83] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[84] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[85] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv:1505.05770*, 2015.

[86] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[87] Yunus Saatçi. *Scalable inference for structured Gaussian process models*. PhD thesis, University of Cambridge, 2012.

[88] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *AISTATS*, volume 1, page 3, 2009.

[89] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[90] Yves-Laurent Kom Samo and Stephen Roberts. String gaussian process kernels. *arXiv preprint arXiv:1506.02239*, 2015.

[91] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[92] Amar Shah, Andrew Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to Gaussian processes. In *Artificial Intelligence and Statistics*, pages 877–885, 2014.

[93] Jiaxin Shi, Jianfei Chen, Jun Zhu, Shengyang Sun, Yucen Luo, Yihong Gu, and Yuhao Zhou. ZhuSuan: A library for Bayesian deep learning. *arXiv:1709.05870*, 2017.

[94] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[95] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

[96] Edward Snelson, Zoubin Ghahramani, and Carl E Rasmussen. Warped gaussian processes. In *Advances in neural information processing systems*, pages 337–344, 2004.

[97] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[98] Maxwell B Stinchcombe. Neural network approximation of continuous functionals and continuous functions on compactifications. *Neural Networks*, 12(3):467–477, 1999.

[99] Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

[100] Felipe Tobar, Thang D Bui, and Richard E Turner. Learning stationary time series using Gaussian processes with nonparametric kernels. In *Advances in Neural Information Processing Systems*, pages 3501–3509, 2015.

[101] Dustin Tran, Rajesh Ranganath, and David M Blei. Deep and hierarchical implicit models. *arXiv:1702.08896*, 2017.

[102] Richard E Turner and Maneesh Sahani. Statistical inference for single-and multi-band probabilistic amplitude demodulation. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5466–5469. IEEE, 2010.

[103] Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 2845–2854, 2017.

[104] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv:1611.01722*, 2016.

[105] Christopher KI Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, pages 295–301, 1997.

[106] Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.

[107] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[108] Huaiyu Zhu and Richard Rohwer. Information geometric measurements of generalisation. 1995.

Appendix

A Brief review of Gaussian Processes

Gaussian Processes [84], as a popular example of Bayesian nonparametrics, provides a principled probabilistic framework for non-parametric Bayesian inference over functions, by imposing rich and flexible nonparametric priors over functions of interest. As flexible and interpretable function approximators, their Bayesian nature also enables Gaussian Processes to provide valuable information of uncertainties regarding predictions for intelligence systems, all wrapped up in a single, *exact* closed form solution of posterior inference.

Despite the success and popularity of Gaussian Processes (as well as other Bayesian nonparametrics) in the past decades, their $\mathcal{O}(N^3)$ computation and $\mathcal{O}(N^2)$ storage complexities make it impractical to apply Gaussian Processes to large and complicated datasets. Therefore, people often resort to complicated approximate methods [14, 15, 18, 40, 81, 87, 91, 95, 99, 102].

An inevitable issue that must also be addressed is the representational power of GP kernels. It has been argued that [8] local kernels commonly used for nonlinear regressions are not able to obtain hierarchical representations for high dimensional data, which limits the usefulness of Bayesian nonparametric methods for complicated tasks. A number of novel schemes were proposed, including deep GPs [13, 19, 20], the design of expressive kernels [26, 100, 103], and the hybrid models using deep neural nets as input features of GPs [45, 106]. However, the first two approaches still struggle to model complex high dimensional data such as texts and images easily; and in the third approach, the merits of fully Bayesian approach has been discarded.

We breifly introduce Gaussian Processes for regression. Assume that we have a set of observational data $\{(\mathbf{x}_n, y_n\}_{n=1}^N)$, where $\mathbf{x}_n$ is the D dimensional input of n th data point, and y_n is the corresponding scalar target of the regression problem. A Gaussian Process model assumes that y_n is generated according the following procedure: firstly a function $f(\cdot)$ is drawn from a Gaussian Process $\mathcal{GP}(m, k)$ (to be defined later). Then for each input data $\mathbf{x}_n$, the corresponding y_n is then drawn according to:

$$y_n = f(\mathbf{x}_n) + \epsilon_n, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2), \;\; n = 1, \cdots, N$$

A Gaussian Process is a nonparametric distribution defined over the space of functions, such that:

Definition 2 (Gaussian Processes). *A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distributions. A Gaussian Process is fully specified by its mean function $m(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}$ and covariance function $\mathcal{K}(\cdot, \cdot) : (\mathbb{R}^D, \mathbb{R}^D) \mapsto \mathbb{R}$, such that any finite collection of function values $\mathbf{f}$ are distributed as Gaussian distribution $\mathcal{N}(\mathbf{f}; \mathbf{m}, \mathbf{K_{ff}})$, where $(\mathbf{m})_n = m(\mathbf{x}_n)$, $(\mathbf{K_{ff}})_{n,n'} = \mathcal{K}(\mathbf{x}_n, \mathbf{x}_{n'})$.*

Now, given a set of observational data $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, we are able to perform probabilistic inference and assign posterior probabilities over all plausible functions that might have generated the data. Under the setting of regression, given a new test point input data $\mathbf{x}_*$, we are interested in posterior distributions over f_*. Fortunately, this posterior distribution of interest admits a closed form solution $f_* \sim \mathcal{N}(\mu_*, \Sigma_*)$:

$$\mu_* = \mathbf{m} + K_{\mathbf{x}_*\mathbf{f}}(\mathbf{K_{ff}} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}) \tag{A.1}$$

$$\Sigma_* = K_{\mathbf{x}_*\mathbf{x}_*} - K_{\mathbf{x}_*\mathbf{f}}(\mathbf{K_{ff}} + \sigma^2\mathbf{I})^{-1}K_{\mathbf{f}\mathbf{x}_*} \tag{A.2}$$

In our notation, $(\mathbf{y})_n = y_n$, $(K_{\mathbf{x}_*\mathbf{f}})_n = \mathcal{K}(\mathbf{x}_*, \mathbf{x}_n)$, and $K_{\mathbf{x}_*\mathbf{x}_*} = \mathcal{K}(\mathbf{x}_*, \mathbf{x}_*)$. Although the Gaussian Process regression framework is theoretically very elegant, in practice its computational burden is prohibitive for large datasets since the matrix inversion $(\mathbf{K_{ff}} + \sigma^2\mathbf{I})^{-1}$ takes $\mathcal{O}(N^3)$ time due to Cholesky decomposition. Once matrix inversion is done, predictions in test time can be made in $\mathcal{O}(N)$ for posterior mean μ_* and $\mathcal{O}(N^2)$ for posterior uncertainty Σ_*, respectively.

B Brief Review of Variational inference, and black-box αnergy

We give a brief review of modern variational techniques, including standard variational inference and black-box α-Divergence minimization (BB-α), on which our methodology is heavily based. Considers the problem of finding the posterior distribution, $p(\theta|\mathcal{D},\tau)$, $\mathcal{D}=\{\mathbf{x}_n\}_{n=1}^N)$ under the model likelihood $p(\mathbf{x}|\theta,\tau)$ and a prior distribution $p_0(\theta)$:

$$p(\theta|\mathcal{D},\tau)\propto\frac{1}{Z}p_0(\theta)\prod_n p(\mathbf{x}_n|\theta,\tau)$$

Variational inference [48] transfers the above inference problem to an optimization problem, by first proposing a class of approximate posterior $q(\theta)$, and then minimize the KL-divergence from the approximate posterior to the true posterior $\mathcal{D}_{\text{KL}}(q||p)$. Equivalently, VI optimizes the following variational free energy,

$$\mathcal{F}_{\text{VFE}}=\log p(\mathcal{D}|\tau)-\mathcal{D}_{\text{KL}}[q||p(\theta|\mathcal{D})]=\left\langle\log\frac{p(\mathcal{D},\theta|\tau)}{q(\theta)}\right\rangle_{q(\theta)}.$$

Built upon the idea of VI, BB-α is a modern black-box variational inference framework that unifies and interpolates between Variational Bayes [48] and Expectation Propagation-like algorithms [60, 72]. BB-α performs approximate inference by minimizing the following α-divergence [108] $\mathrm{D}_\alpha[p||q]$:

$$\mathrm{D}_\alpha[p||q]=\frac{1}{\alpha(1-\alpha)}\left(1-\int p(\theta)^\alpha q(\theta)^{1-\alpha}d\theta\right).$$

α-divergence is a generic class of divergences that includes the inclusive KL-divergence (α=1, corresponds to EP), Hellinger distance (α=0.5), and the exclusive KL-divergence ($\alpha=0$, corresponds to VI) as special cases. Traditionally power EP [71] optimizes an α-divergence locally with exponential family approximation $q(\theta)\propto\frac{1}{Z}p_0(\theta)\prod_n\tilde{f}_n(\theta)$,$\tilde{f}_n(\theta)\propto\exp\left[\lambda_n^T\phi(\theta)\right]$ via message passing. It converges to a fixed point of the so called *power EP energy*:

$$\begin{aligned}\mathcal{L}_{\text{PEP}}(\lambda_0,\{\lambda_n\})&=\log Z(\lambda_0)+(\frac{N}{\alpha}-1)\log Z(\lambda_q)\\&-\frac{1}{\alpha}\sum_{n=1}^N\log\int p(\mathbf{x}_n|\theta,\tau)^\alpha\exp\left[(\lambda_q-\alpha\lambda_n)^T\phi(\theta)\right]d\theta,\end{aligned}$$

where $\lambda_q=\lambda_0+\sum_{n=1}^N\lambda_n$ is the natural parameter of $q(\theta)$. On the contrary, BB-α directly optimizes $\mathcal{L}_{\text{PEP}}$ with tied factors $\tilde{f}_n=\tilde{f}$ to avoid prohibitive local factor updates and storage on the whole dataset. This means $\lambda_n=\lambda$ for all n and $\lambda_q=\lambda_0+N\lambda$. Therefore instead of parameterizing each factors, one can directly parameterize $q(\theta)$ and replace all the local factors in the power-EP energy function by $\tilde{f}(\theta)\propto(q(\theta)/p_0(\theta))^{1/N}$. After re-arranging terms, this gives the BB-α energy:

$$\mathcal{L}_\alpha(q)=-\frac{1}{\alpha}\sum_n\log\mathbb{E}_q\left[\left(\frac{f_n(\theta)p_0(\theta)^{\frac{1}{N}}}{q(\theta)^{\frac{1}{N}}}\right)^\alpha\right].$$

which can be further approximated by the following if the dataset is large [59]:

$$\mathcal{L}_\alpha(q)=\mathcal{D}_{\text{KL}}[q||p_0]-\frac{1}{\alpha}\sum_n\log\mathbb{E}_q\left[p(\mathbf{x}_n|\theta,\tau)^\alpha\right].$$

The optimization of $\mathcal{L}_\alpha(q)$ could be performed in a black-box manner with reparameterization trick [50] and MC approximation. Empirically, it has been shown that BB-α with $\alpha\neq0$ can return significantly better uncertainty estimation than VB, and has been applied successfully in different scenarios [23, 59]. From the hyperparameter learning (i.e., τ in $p(\mathbf{x}_n|\theta,\tau)$) point of veiw, it is shown in [62] that the BB-α energy $\mathcal{L}_\alpha(q)$ constitutes a better estimation of log marginal likelihood, $\log p(\mathcal{D})$ when compared with the variational free energy. Therefore, for both inference and learning, BB-α energy is extensively used in this paper.

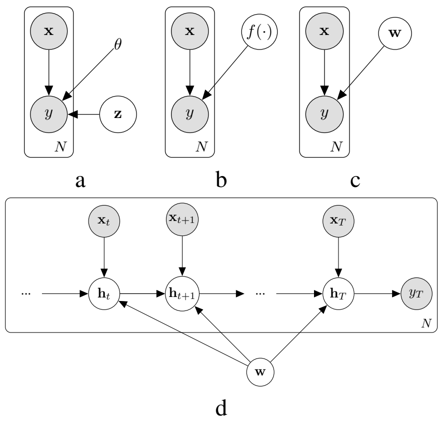

Figure 3: Examples of IPs: (a) Neural samplers; (b) Warped Gaussian Processes (c) Bayesian neural networks; (d) Bayesian RNNs.

C Implicit stochastic processes

C.1 Examples of implicit stochastic processes

IPs are very powerful and form a rich class of priors over functions (see also C.2 and C.3). Indeed, we visualize some examples of IPs in Figure 3 with discussions as follows:

Example 1 (Data simulators)**.** *Simulators, e.g. physics engines and climate models, are omnipresent in science and engineering. These models encode laws of physics in* $g_\theta(\cdot,\cdot)$*, use* $\mathbf{z} \sim p(\mathbf{z})$ *to explain the remaining randomness, and evaluate the function at input locations* $\mathbf{x}$*:* $f(\mathbf{x}) = g_\theta(\mathbf{x}, \mathbf{z})$*. We define the* ***neural sampler*** *as a specific instance of this class. In this case* $g_\theta(\cdot,\cdot)$ *is a neural network with weights* θ*, i.e.,* $g_\theta(\cdot,\cdot) = \mathrm{NN}_\theta(\cdot,\cdot)$*, and* $p(\mathbf{z}) =$ *Uniform*$([-a, a]^d)$*.*

Example 2 (Warped Gaussian Processes)**.** *Warped Gaussian Processes [96] is also an interesting example of IPs. Let* $z(\cdot) \sim p(z)$ *be an GP prior, and* $g_\theta(\mathbf{x}, z)$ *is defined as* $g_\theta(\mathbf{x}, z) = h(z(\mathbf{x}))$*, where* $h(\cdot)$ *is a one dimensional monotonic function that mapping on to the whole of the real line.*

Example 3 (Bayesian neural network)**.** *In a Bayesian neural network the synaptic weights* W *are random variables (i.e.,* $\mathbf{z} = W$*) with a prior* $p(W)$ *on them. A function is sampled by* $W \sim p(W)$ *and then setting* $f(\mathbf{x}) = g_\theta(\mathbf{x}, W) = \mathrm{NN}_\mathrm{W}(\mathbf{x})$ *for all* $\mathbf{x} \in \mathbf{X}$*. In this case* θ *could include, e.g., the network architecture and additional hyper-parameters.*

Example 4 (Bayesian RNN)**.** *Similar to Example 3, a Bayesian recurrent neural network (RNN) can be defined by considering its weights as random variables, and taking as function evaluation an output value generated by the RNN after processing the last symbol of an input sequence.*

C.2 Well-definedness of implicit processes (finite dimensional case)

Proposition 1 (Finite dimension case)**.** *Let* $\mathbf{z}$ *be a finite dimensional vector. Then there exists a unique stochastic process, such that any finite collection of random variables has distribution implicitly defined by Definition (1).*

Proof Generally, consider the following noisy IP model:

$$f(\cdot) \sim \mathcal{IP}(g_\theta(\cdot,\cdot), p_\mathbf{z}), \;\; y_n = f(\mathbf{x}_n) + \epsilon_n, \; \epsilon_n \sim \mathcal{N}(0, \sigma^2).$$

For any finite collection of random variables $y_{1:n} = \{y_1, ..., y_n\}$, $\forall n$ we denote the induced distribution as $p_n(y_{1:n})$. Note that $p_{1:n}(y_{1:n})$ can be represented as $\mathbb{E}_{p(\mathbf{z})}[\prod_{i=1}^n \mathcal{N}(y_i; g(\mathbf{x}_i; \mathbf{z}), \sigma^2)]$. Therefore for any $m < n$, we have

$$\begin{aligned}&\int p_{1:n}(y_{1:n}) dy_{m+1:n} \\ &= \int \int \prod_{i=1}^n \mathcal{N}(y_i; g(\mathbf{x}_i, \mathbf{z}), \sigma^2) p(\mathbf{z}) d\mathbf{z} dy_{m+1:n}\end{aligned}$$

$$= \int \int \prod_{i=1}^{n} \mathcal{N}(y_i; g(\mathbf{x}_i, \mathbf{z}), \sigma^2) p(\mathbf{z}) dy_{m+1:n} d\mathbf{z}$$
$$= \int \prod_{i=1}^{m} \mathcal{N}(y_i; g(\mathbf{x}_i, \mathbf{z}), \sigma^2) p(\mathbf{z}) d\mathbf{z} = p_{1:m}(y_{1:m}).$$

Note that the swap+ of the order of integration relies on that the integral is finite. Therefore, the marginal consistency condition of Kolmogorov extension theorem is satisfied. Similarly, the permutation consistency condition of Kolmogorov extension theorem can be proved as follows: assume $\pi(1:n) = \{\pi(1), ..., \pi(n)\}$ is a permutation of the indices $1:n$, then

$$\begin{aligned} &p_{\pi(1:n)}(y_{\pi(1:n)}) \\ &= \int \prod_{i=1}^{n} \mathcal{N}(y_{\pi(i)}; g(\mathbf{x}_{\pi(i)}, \mathbf{z}), \sigma^2) p(\mathbf{z}) d\mathbf{z} \\ &= \int \prod_{i=1}^{n} \mathcal{N}(y_i; g(\mathbf{x}_i, \mathbf{z}), \sigma^2) p(\mathbf{z}) d\mathbf{z} = p_{1:n}(y_{1:n}). \end{aligned}$$

Therefore, by Kolmogorov extension theorem, there exists a unique stochastic process, with finite marginals that are distributed exactly according to Definition 1.

□

C.3 Well-definedness of implicit processes (infinite dimensional case)

Proposition 2 (Infinite dimension case)**.** *Let $z(\cdot) \sim \mathcal{SP}(0, C)$ be a centered continuous stochastic process on $\mathcal{L}^2(\mathbb{R}^d)$ with covariance function $C(\cdot, \cdot)$. Then the operator $g(\mathbf{x}, z) = O_k(z)(\mathbf{x}) := h(\int_{\mathbf{x}} \sum_{l=0}^{M} K_l(\mathbf{x}, \mathbf{x}') z(\mathbf{x}') d\mathbf{x}'),\ 0 < M < +\infty$ defines a stochastic process if $K_l \in \mathcal{L}^2(\mathbb{R}^d \times \mathbb{R}^d)$, h is a Borel measurable, bijective function in $\mathbb{R}$ and there exist $0 \leq A < +\infty$ such that $|h(x)| \leq A|x|$ for $\forall x \in \mathbb{R}$.*

Proof Since $\mathcal{L}^2(\mathbb{R}^d)$ is closed under finite summation, without loss of generality, we consider the case of $M = 1$ where $O(z)(\mathbf{x}) = h(\int K(\mathbf{x}, \mathbf{x}') z(\mathbf{x}') d\mathbf{x}')$. According to Karhuhen-Loeve expansion (K-L expansion) theorem [67], the stochastic process z can be expanded as the *stochastic* infinite series,

$$z(\mathbf{x}) = \sum_{i}^{\infty} Z_i \phi_i(\mathbf{x}), \quad Z_i \sim \mathcal{N}(0, \lambda_i), \quad \sum_{i}^{\infty} \lambda_i < +\infty.$$

Here $\{\phi_i\}_{i=1}^{\infty}$ is an orthonormal basis of $\mathcal{L}^2(\mathbb{R}^d)$ that are also eigen functions of the operator $O_C(z)$ defined by $O_C(z)(\mathbf{x}) = \int C(\mathbf{x}, \mathbf{x}') z(\mathbf{x}') d\mathbf{x}'$. The variance λ_i of Z_i is the corresponding eigen value of $\phi_i(\mathbf{x})$.

Apply the linear operator

$$O_K(z)(\mathbf{x}) = \int K(\mathbf{x}, \mathbf{x}') z(\mathbf{x}') d\mathbf{x}'$$

on this K-L expansion of z, we have:

$$\begin{aligned} O_K(z)(\mathbf{x}) &= \int K(\mathbf{x}, \mathbf{x}') z(\mathbf{x}') d\mathbf{x}' \\ &= \int K(\mathbf{x}, \mathbf{x}') \sum_{i}^{\infty} Z_i \phi_i(\mathbf{x}') d\mathbf{x}' \\ &= \sum_{i}^{\infty} Z_i \int K(\mathbf{x}, \mathbf{x}') \phi_i(\mathbf{x}') d\mathbf{x}', \end{aligned} \tag{C.1}$$

where the exchange of summation and integral is guaranteed by Fubini's theorem. Therefore, the functions $\{\int_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}') \phi_i(\mathbf{x}') d\mathbf{x}'\}_{i=1}^{\infty}$ forms a new basis of $\mathcal{L}^2(\mathbb{R}^d)$. To show that the stochastic series C.1 converge:

$$
\begin{aligned}
&||\sum_i^\infty Z_i \int K(\mathbf{x},\mathbf{x}')\phi_i(\mathbf{x}')d\mathbf{x}||_{\mathcal{L}^2}^2 \\
&\leq ||O_K||^2 ||\sum_i^\infty Z_i\phi_i(\mathbf{x}')||_{\mathcal{L}^2}^2 \\
&= ||O_K||^2 \sum_i^\infty ||Z_i||_2^2,
\end{aligned}
$$

where the operator norm is defined by

$$||O_K|| := \inf\{c \geq 0 : ||O_k(f)||_{\mathcal{L}^2} \leq c||f||_{\mathcal{L}^2},\ \forall f \in \mathcal{L}^2(\mathbb{R}^d)\}.$$

This is a well defined norm since O_K is a bounded operator ($K \in \mathcal{L}^2(\mathbb{R}^d \times \mathbb{R}^d)$). The last equality follows from the orthonormality of $\{\phi_i\}$. The condition $\sum_i^\infty \lambda_i < \infty$ further guarantees that $\sum_i^\infty ||Z_i||^2$ converges almost surely. Therefore, the random series (C.1) converges in $\mathcal{L}^2(\mathbb{R}^d)$ a.s..

Finally we consider the nonlinear mapping $h(\cdot)$. With $h(\cdot)$ a Borel measurable function satisfying the condition that there exist $0 \leq A < +\infty$ such that $|h(x)| \leq A|x|$ for $\forall x \in \mathbb{R}$, it follows that $h \circ O_K(z) \in \mathcal{L}^2(\mathbb{R}^d)$. In summary, $g = O_k(z) = h \circ O_K(z)$ defines a well-defined stochastic process on $\mathcal{L}^2(\mathbb{R}^d)$.

□

Note that for infinite dimensional case, the operator defined in Proposition 2 can be recursively applied to build many powerful models [33, 36, 56, 98, 105] that even possesses universal approximation ability to nonlinear operators [36]. In the recent example [36], the so called Deep Function Machines (DFMs) that possess universal approximation ability to nonlinear operators:

Definition 3 (Deep Function Machines [36]). *A deep function machine $g = O_{DFM}(z, S)$ is a computational skeleton S indexed by I with the following properties:*

- *Every vertex in S is a Hilbert space $\mathbb{H}_l$ where $l \in I$.*
- *If nodes $l \in A \subset I$ feed into l' then the activation on l' is denoted $y^l \in \mathbb{H}_l$ and is defined as*

$$y^{l'} = h \circ (\sum_{l \in A} O_{K_l}(y^l))$$

Therefore, by Proposition 2, we have proved:

Corollary 2 *Let $z(\cdot) \sim \mathcal{SP}(0, C)$ be a centered continuous stochastic process on $\mathbb{H} = \mathcal{L}^2(\mathbb{R}^d)$. Then the Deep function machine operator $g = O_{DFM}(z, S)$ defines a well-defined stochastic process on $\mathbb{H}$.*

D Details of the Wake-Sleep procedure for VIPs

D.1 Sleep phase: GP posterior as variational distribution

This section proposes an approximation to the IP posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)$. A naive approach based on variational inference [48] would require computing the joint distribution $p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \theta)$ that is again intractable. However, sampling from this joint distribution is straightforward. Therefore, we leverage the idea of the *sleep phase* in the wake-sleep algorithm to approximate the joint distribution $p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \theta)$ instead.

Precisely, we approximate $p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \theta)$ with a simpler distribution $q(\mathbf{y}, \mathbf{f}|\mathbf{X}) = q(\mathbf{y}|\mathbf{f})q(\mathbf{f}|\mathbf{X})$ instead. We use a GP prior for $q(\mathbf{f}|\mathbf{X})$ with mean and covariance functions $m(\cdot)$ and $\mathcal{K}(\cdot, \cdot)$, respectively, and write the prior as $q(\mathbf{f}|\mathbf{X}) = q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, m, \mathcal{K})$. The sleep-phase update minimizes the KL divergence between the two joint distributions, which reduces to the following constrained optimization problem:

$$q^\star_{\mathcal{GP}} = \underset{m,\mathcal{K}}{\operatorname{argmin}} \mathcal{U}(m, \mathcal{K}), \quad \mathcal{U}(m, \mathcal{K}) = \mathrm{D}_{\mathrm{KL}}[p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}, \mathbf{f}|\mathbf{X}, m, \mathcal{K})]. \tag{D.1}$$

We further simplify the approximation by using $q(\mathbf{y}|\mathbf{f}) = p(\mathbf{y}|\mathbf{f})$, which reduces $\mathcal{U}(m, \mathcal{K})$ to $\mathrm{D}_{KL}[p(\mathbf{f}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, m, \mathcal{K})]$, and the objective is minimized when $m(\cdot)$ and $\mathcal{K}(\cdot, \cdot)$ are equal to the mean and covariance functions of the IP, respectively:

$$\begin{aligned} m^\star(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ \mathcal{K}^\star(\mathbf{x}_1, \mathbf{x}_2) &= \mathbb{E}[(f(\mathbf{x}_1) - m^\star(\mathbf{x}_1))(f(\mathbf{x}_2) - m^\star(\mathbf{x}_2))]. \end{aligned} \tag{D.2}$$

In the following we also write the optimal solution as $q^\star_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \theta) = q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, m^\star, \mathcal{K}^\star)$ to explicitly specify the dependency on prior parameters θ. In practice, the mean and covariance functions are estimated by by Monte Carlo, which leads to *maximum likelihood* training for the GP with *dreamed* data from the IP. Assume S functions are drawn from the IP: $f_s(\cdot) \sim \mathcal{IP}(g_\theta(\cdot, \cdot), p_{\mathbf{z}}), s = 1, \dots, S$. The optimum of $\mathcal{U}(m, \mathcal{K})$ is then estimated by the *MLE solution*:

$$m^\star_{\text{MLE}}(\mathbf{x}) = \frac{1}{S} \sum_s f_s(\mathbf{x}), \tag{D.3}$$

$$\mathcal{K}^\star_{\text{MLE}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{S} \sum_s \Delta_s(\mathbf{x}_1)\Delta_s(\mathbf{x}_2), \tag{D.4}$$

$$\Delta_s(\mathbf{x}) = f_s(\mathbf{x}) - m^\star_{\text{MLE}}(\mathbf{x}).$$

To reduce computational costs, the number of dreamed samples S is often small. Therefore, we perform *maximum a posteriori/posterior mean estimation* instead, by putting an inverse Wishart process prior [92] $\mathcal{IWP}(\nu, \Psi)$ over the GP covariance function (Appendix G.1).

The original sleep phase algorithm in [43] also find a posterior approximation by minimizing (D.1). However, the original approach would define the q distribution as $q(\mathbf{y}, \mathbf{f}|\mathbf{X}) = p(\mathbf{y}|\mathbf{X}, \theta)q_{\mathcal{GP}}(\mathbf{f}|\mathbf{y}, \mathbf{X})$, which builds a *recognition model* that can be directly transfered for later inference. By contrast, we define $q(\mathbf{y}, \mathbf{f}|\mathbf{X}) = p(\mathbf{y}|\mathbf{f})q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X})$, which corresponds to an approximation of the IP prior. In other words, we approximate an intractable generative model using another generative model with a GP prior and later, the resulting GP posterior $q^\star_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y})$ is employed as the variational distribution. Importantly, we never explicitly perform the sleep phase updates as there is an analytic solution readily available, which can potentially save an enormous amount of computation.

Another interesting observation is that the sleep phase's objective $\mathcal{U}(m, \mathcal{K})$ also provides an upper-bound to the KL divergence between the posterior distributions,

$$\mathcal{J} = \mathrm{D}_{\text{KL}}[p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y})].$$

One can show that $\mathcal{U}$ is an upper-bound of $\mathcal{J}$ according to the non-negativity and chain rule of KL divergence:

$$\mathcal{U}(m, \mathcal{K}) = \mathcal{J} + \mathrm{D}_{\text{KL}}[p(\mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X})] \geq \mathcal{J}. \tag{D.5}$$

Therefore $\mathcal{J}$ is also decreased when the mean and covariance functions are optimized during the sleep phase. This justifies $\mathcal{U}(m, \mathcal{K})$ as a valid variational objective for posterior approximation.

D.2 Wake phase: Bayesian linear regression over random functions

In the wake phase of traditional wake-sleep, the prior parameters θ are optimized by maximizing the variational lower-bound [48] to the log marginal likelihood $\log p(\mathbf{y}|\mathbf{X}, \theta)$. Unfortunately, this requires evaluating the IP prior $p(\mathbf{f}|\mathbf{X}, \theta)$ which is again intractable. But recall from (D.5) that during the sleep phase $\mathrm{D}_{\text{KL}}[p(\mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X})]$ is also minimized. Therefore we directly approximate the log marginal likelihood using the optimal GP from the sleep phase, i.e.

$$\log p(\mathbf{y}|\mathbf{X}, \theta) \approx \log q^\star_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta). \tag{D.6}$$

This again demonstrates the key advantage of the proposed sleep phase update via generative model matching. Also it becomes a sensible objective for predictive inference as the GP returned by wake-sleep will be used at prediction time anyway.

Similar to GP regression, optimizing $\log q^\star_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta)$ can be computationally expensive for large datasets. Therefore sparse GP approximation techniques [15, 40, 95, 99] are applicable, but we leave them to future work and consider an alternative approach that is related to random feature approximations of GPs [3, 29, 31, 54, 83]. Note that $\log q^\star_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta)$ can be approximated by the log marginal likelihood of a Bayesian linear regression model with S randomly sampled dreamed functions, and coefficient $\mathbf{a} = (a_1, ..., a_S)$:

$$\log q^\star_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta) \approx \log \int \prod_n q^\star(y_n|\mathbf{x}_n, \mathbf{a}, \theta) p(\mathbf{a}) d\mathbf{a}, \tag{D.7}$$

$$\begin{aligned} q^\star(y_n|\mathbf{x}_n, \mathbf{a}, \theta) &= \mathcal{N}\left(y_n; \mu(\mathbf{x}_n, \mathbf{a}, \theta), \sigma^2\right), \\ p(\mathbf{a}) &= \mathcal{N}(\mathbf{a}; 0, \mathbf{I}), \\ \mu(\mathbf{x}_n, \mathbf{a}, \theta) &= \frac{1}{\sqrt{S}} \sum_s (f_s(\mathbf{x}_n) - m^\star(\mathbf{x}_n)) a_s. \end{aligned} \tag{D.8}$$

Then it is straightforward to apply variational inference again for scalable stochastic optimization, and we follow [42, 59, 62] to approximate (D.7) by the α-energy (see Appendix B):

$$\begin{aligned} \log q^\star_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta) &\approx \mathcal{L}^\alpha_{\mathcal{GP}}(\theta, q(\mathbf{a})) \\ &= \frac{1}{\alpha} \sum_n^N \log \mathbb{E}_{q(\mathbf{a})}\left[q^\star(y_n|\mathbf{x}_n, \mathbf{a}, \theta)^\alpha\right] - \mathrm{D}_{\mathrm{KL}}[q(\mathbf{a})||p(\mathbf{a})]. \end{aligned} \tag{D.9}$$

When $\alpha \to 0$ the α-energy reduces to the variational lower-bound, and empirically the α-energy has better approximation accuracy when $\alpha > 0$ [42, 59, 62]. Also since the prior $p(\mathbf{a})$ is conjugate to the Gaussian likelihood $q^\star(y_n|\mathbf{x}_n, \mathbf{a}, \theta)$, the exact posterior of $\mathbf{a}$ can be reached by a correlated Gaussian $q(\mathbf{a})$. Stochastic optimization is applied to the α-energy wrt. θ and $q(\mathbf{a})$ jointly, making our approach scalable to big datasets.

D.3 Computational complexity and scalable predictive inference

Assume the evaluation of a sampled function value $f(\mathbf{x}) = g_\theta(\mathbf{x}, \mathbf{z})$ for a given input $\mathbf{x}$ takes $\mathcal{O}(C)$ time. The VIP has time complexity $\mathcal{O}(CMS + MS^2 + S^3)$ in training, where M is the size of a mini-batch, and S is the number of random functions sampled from $\mathcal{IP}(g_\theta(\cdot,\cdot), p_\mathbf{z})$. Note that approximate inference techniques in $\mathbf{z}$ space, e.g. mean-field Gaussian approximation to the posterior of Bayesian neural network weights [10, 42, 59], also takes $\mathcal{O}(CMS)$ time. Therefore when C is large (typically the case for neural networks) the additional cost is often negligible, as S is usually significantly smaller than the typical number of inducing points in sparse GP ($S = 20$ in the experiments).

Predictive inference follows the standard GP equations to compute $q^\star_{\mathcal{GP}}(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}, \theta^\star)$ on test data $\mathbf{X}_*$ that has K datapoints: $\mathbf{f}_* \sim \mathcal{N}(\mathbf{f}_*; \mathbf{m}_*, \mathbf{\Sigma}_*)$,

$$\begin{aligned} \mathbf{m}_* &= m^\star(\mathbf{x}_*) + \mathbf{K}_{*\mathbf{f}}(\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - m^\star(\mathbf{X})), \\ \mathbf{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_{*\mathbf{f}}(\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{\mathbf{f}*}. \end{aligned} \tag{D.10}$$

Recall that the optimal variational GP approximation has mean and covariance functions defined as (D.3) and (D.4), respectively, which means $\mathbf{K}_{\mathbf{ff}}$ has rank S. Therefore predictive inference requires both function evaluations and matrix inversion, which costs $\mathcal{O}(C(K+N)S + NS^2 + S^3)$ time. This complexity can be further reduced: note that the computational cost is dominated by the inversion of matrix $\mathbf{K}_{\mathbf{ff}} + \sigma^2\mathbf{I}$. Denote the Cholesky decomposition of the kernel matrix $\mathbf{K}_{\mathbf{ff}} = \mathbf{B}\mathbf{B}^\top$ as before. It is straightforward to show that in the Bayesian linear regression problem (D.8) the exact posterior of $\mathbf{a}$ is $q(\mathbf{a}|\mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}, \mathbf{\Sigma})$, with $\boldsymbol{\mu} = \frac{1}{\sigma^2}\mathbf{\Sigma}\mathbf{B}^\top(\mathbf{y} - \mathbf{m}), \sigma^2\mathbf{\Sigma}^{-1} = \mathbf{B}^\top\mathbf{B} + \sigma^2\mathbf{I}$. Therefore the parameters of the GP predictive distribution in (D.10) are reduced to:

$$\begin{aligned} \mathbf{m}_* = \boldsymbol{\phi}_*^\top \boldsymbol{\mu}, \quad \mathbf{\Sigma}_* &= \boldsymbol{\phi}_*^\top \mathbf{\Sigma} \phi_*, \\ (\boldsymbol{\phi}_*)_s &= \frac{1}{\sqrt{S}}(f_s(\mathbf{x}_*) - m^\star(\mathbf{x}_*)). \end{aligned} \tag{D.11}$$

This reduces the prediction cost to $\mathcal{O}(CKS + S^3)$, which is on par with e.g. conventional predictive inference techniques for Bayesian neural networks that also cost $\mathcal{O}(CKS)$. In practice we use the mean and covariance matrix from $q(\mathbf{a})$ to compute the predictive distribution. Alternatively one can directly sample $\mathbf{a} \sim q(\mathbf{a})$ and compute $\mathbf{f}_* = \sum_{s=1}^S a_s f_s(\mathbf{X}_*)$, which is also an $\mathcal{O}(CKS + S^3)$ inference approach[3] but is liable for higher variance.

[3]If $q(\mathbf{a})$ is a mean-field Gaussian distribution then the cost is $\mathcal{O}(CKS)$.

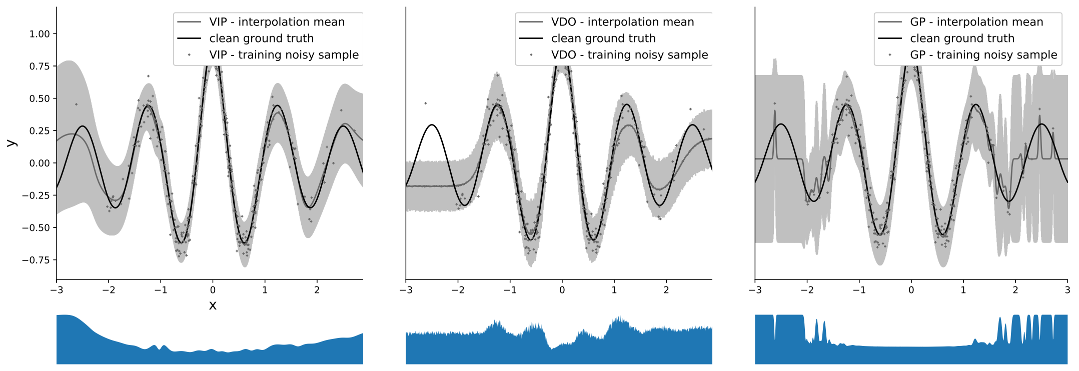

Figure 4: First row: Predictions returned from VIP (**left**), VDO (**middle**) and exact GP (**right**), respectively. **Dark grey dots**: noisy observations; **dark line**: clean ground truth function; **dark gray line**: predictive means; **Gray shaded area**: confidence intervals with 2 standard deviations. **Second row**: Corresponding predictive uncertainties.

E Further experiments

We evaluate VIPs for regression using real-world data. For small datasets we use the posterior GP equations for prediction, otherwise we use the $\mathcal{O}(S^3)$ approximation. We use $S = 20$ for VIP unless noted otherwise. When the VIP is equipped with a Bayesian NN/LSTM as prior over functions, the prior parameters over each weight are untied, thus can be individually tuned. Fully Bayesian estimates of the prior parameters are used in experiments E.2 and E.4. We focus on comparing VIPs to other fully Bayesian approaches, with detailed experimental settings presented in the next section.

E.1 Synthetic example

We first evaluate the predictive inference and uncertainty estimate of our method on a toy regression example. The training set is generated by first sampling 300 inputs x from $\mathcal{N}(0,1)$. Then, for each x obtained, the corresponding target y is simulated as $y = \frac{\cos 5x}{|x|+1} + \epsilon,\ \epsilon \sim \mathcal{N}(0, 0.1)$. The test set consists of 1000 evenly spaced points on $[-3, 3]$. We use an IP with a Bayesian neural network (1-10-10-1 architecture) as the prior. We use $\alpha = 0$ for the wake-step training. We also compare VIP with the full exact GP with RBF kernel, and another Bayesian neural network with identical architecture but trained using variational Bernoulli dropout (VDO) with dropout rate $p = 0.99$ and length scale $l = 0.001$. The (hyper-)parameters are optimized using 500 epochs (batch training) with Adam optimizer (learning rate = 0.01).

Figure 4 visualizes the results. Compared with VDO and GP, VIP's predictive mean is closer to the ground truth function. Moreover, VIP provides the best predictive uncertainty, especially when compared with VDO: it increases smoothly when $|x| \to 3$, where training data is sparse around there. Although the uncertainty estimate of GP also increases when data is sparser, it slightly over-fits to the training data, and tends to extrapolate a zero mean function around $|x| \approx 3$. Test Negative Log-likelihood (NLL) and RMSE results reveal similar conclusions (see Table 1), where VIP significantly outperforms VDO and GP.

Table 1: Interpolation performance on toy dataset.

Method	VIP	VDO	GP
Test NLL	**-0.60±0.01**	-0.07 ± 0.01	-0.48 ± 0.00
Test RMSE	**0.140±0.00**	0.161±0.00	0.149±0.00

E.2 Multivariate regression

We perform experiments on real-world multivariate regression datasets from the UCI data repository [64]. We train a VIP with a Bayesian neural net as the prior over latent functions (VIP-BNN), and compare it with VDO, GP, SVGP, and additionally other popular approximate inference methods

Table 2: Regression experiment: Average test negative log likelihood

Dataset	N	D	**VIP-BNN**	**VIP-NS**	**VI**	**VDO**	$\alpha = 0.5$	**SVGP**	**exact GP**
boston	506	13	**2.45±0.04**	**2.45±0.03**	2.76±0.04	2.63±0.10	**2.45±0.02**	2.63±0.04	2.46±0.04
concrete	1030	8	**3.02±0.02**	3.13±0.02	3.28±0.01	3.23±0.01	3.06±0.03	3.4±0.01	3.05±0.02
energy	768	8	0.60±0.03	**0.59±0.04**	2.17±0.02	1.13±0.02	0.95±0.09	2.31±0.02	0.57±0.02
kin8nm	8192	8	**-1.12±0.01**	-1.05±0.00	-0.81±0.01	-0.83±0.01	-0.92±0.02	-0.76±0.00	N/A±0.00
power	9568	4	2.92±0.00	2.90±0.00	2.83±0.01	2.88±0.00	**2.81±0.00**	2.82±0.00	N/A±0.00
protein	45730	9	**2.87±0.00**	2.96±0.02	3.00±0.00	2.99±0.00	2.90±0.00	3.01±0.00	N/A±0.00
red wine	1588	11	**0.97±0.02**	1.20±0.04	1.01±0.02	**0.97±0.02**	1.01±0.02	0.98±0.02	0.26±0.03
yacht	308	6	**-0.02±0.07**	0.59±0.13	1.11±0.04	1.22±0.18	0.79±0.11	2.29±0.03	0.10±0.05
naval	11934	16	-5.62±0.04	-4.11±0.00	-2.80±0.00	-2.80±0.00	-2.97±0.14	**-7.81±0.00**	N/A±0.00
Avg.Rank			**1.77±0.54**	2.77±0.57	4.66±0.28	3.88±0.38	2.55±0.37	4.44±0.66	N/A±0.00

Table 3: Regression experiment: Average test RMSE

Dataset	N	D	**VIP-BNN**	**VIP-NS**	**VI**	**VDO**	$\alpha = 0.5$	**SVGP**	**exact GP**
boston	506	13	2.88±0.14	**2.78±0.12**	3.85±0.22	3.15±0.11	3.06±0.09	3.30±0.21	2.95±0.12
concrete	1030	8	**4.81±0.13**	5.54±0.09	6.51±0.10	6.11±0.10	5.18±0.16	7.25±0.15	5.31±0.15
energy	768	8	**0.45±0.01**	**0.45±0.05**	2.07±0.05	0.74±0.04	0.51±0.03	2.39±0.06	0.45±0.01
kin8nm	8192	8	**0.07±0.00**	0.08±0.00	0.10±0.00	0.10±0.00	0.09±0.00	0.11±0.01	N/A±0.00
power	9568	4	4.11±0.05	4.11±0.04	4.11±0.04	4.38±0.03	4.08±0.00	**4.06±0.04**	N/A±0.00
protein	45730	9	**4.25±0.07**	4.54±0.03	4.88±0.04	4.79±0.01	4.46±0.00	4.90±0.01	N/A±0.00
red wine	1588	11	**0.64±0.01**	0.66±0.01	0.66±0.01	**0.64±0.01**	0.69±0.01	0.65±0.01	0.62±0.01
yacht	308	6	**0.32±0.06**	0.54±0.09	0.79±0.05	1.03±0.06	0.49±0.04	2.25±0.13	0.35±0.04
naval	11934	16	**0.00±0.00**	**0.00±0.00**	0.38±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**	N/A±0.00
Avg.Rank			**1.33±0.23**	2.22±0.36	4.66±0.33	4.00±0.44	3.11±0.42	4.44±0.72	N/A±0.00

for Bayesian neural nets: variational Gaussian inference with reparameterization tricks (VI, [10]) and variational dropout (VDO, [29]), variational alpha inference by dropout ($\alpha = 0.5$, [59]). We further train a VIP with neural sampler prior (VIP-NS), as defined in section C. All neural networks use a [dim($\mathbf{x}$)-10-10-1] architecture with two hidden layers of size 10. All the models are trained for 1,000 epochs of full batch training using Adam (learning rate = 0.01). Observational noise variance for VIP and VDO are tuned using fast grid search over validation set, as detailed in Appendix H. The α value for both VIP and alpha-variational inference are fixed to 0.5, as suggested in [42, 59]. The experiments are repeated for 10 times on all datasets except *Protein Structure*, on which the experiments are repeated for 5 times.

Results are shown in Table 2 and 3 with the best performances boldfaced. Note that exact (full) GP models are only trained for small datasets due to its prohibitive cubic computational costs, therefore it is not included for the overall ranking. VIP-based methods consistently outperforms other methods, obtaining the best test-NLL in 7 datasets, and the best test RMSE in 8 out of the 9 datasets. In addition, VIP-BNN obtains the best ranking among 6 methods. It is also encouraging to note that, despite its general form, the VIP-NS achieves the second best average ranking in RMSE, outperforming many specifically designed BNN algorithms. Despite that only $S = 20$ samples are used for VIP-based methods, VIP out performs exact GP 3/5 in terms of test NLL, and 4/5 in terms of test RMSE.

E.3 ABC example: the Lotka–Volterra model

We apply our VIP approach on an Approximate Bayesian Computation (ABC) example with the Lotka–Volterra (L-V) model that models the continuous dynamics of stochastic population of a predator-prey system. An L-V model consists of 4 parameters $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$ that controls the rate of four possible random events in the model:

$$\dot{y} = \theta_1 xy - \theta_2 y, \quad \dot{x} = \theta_3 x - \theta_4 xy,$$

where x is the population of the predator, and y is the population of the prey. Therefore the L-V model is an implicit model, which allows the simulation of data but not the evaluation of model density. We follow the experiment setup of [77] to select the ground truth parameter of the L-V model, so that the model exhibit a natural oscillatory behavior which makes posterior inference difficult. Then the L-V model is simulated for 25 time units with a step size of 0.05, resulting in 500 training observations. The prediction task is to extrapolate the simulation to the $[25, 30]$ time interval.

We consider (approximate) posterior inference using two types of approaches: regression-based methods (VIP-BNN, VDO-BNN and SVGP), and ABC methods (MCMC-ABC [68] and SMC-ABC [5, 11]). ABC methods first perform posterior inference in the parameter space, then use the

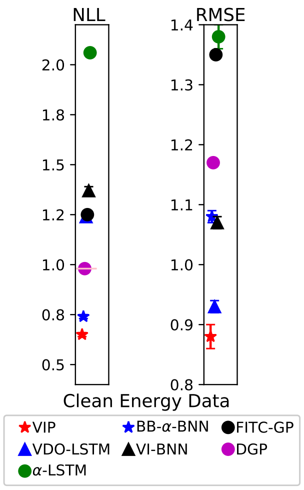

Figure 5: Performance on clean energy dataset

L-V simulator with posterior parameter samples for prediction. On the contrary, regression-based methods treat this task as an ordinary regression problem, where VDO-BNN fits an approximate posterior to the NN weights, and VIP-BNN/SVGP perform predictive inference directly in function space. Results are shown in Table 4, where VIP-BNN outperforms others by a large margin in both test NLL and RMSE. More importantly, VIP is the only regression-based method that outperforms ABC methods, demonstrating its flexibility in modeling implicit systems.

Table 4: ABC with the Lotka–Volterra model

Method	VIP-BNN	VDO-BNN	SVGP	MCMC-ABC	SMC-ABC
Test NLL	**0.485**	1.25	1.266	0.717	0.588
Test RMSE	**0.094**	0.80	0.950	0.307	0.357

E.4 Bayesian LSTM for predicting power conversion efficiency of organic photovoltaics molecules

Finally we perform experiments with data from the Harvard Clean Energy Project, the world's largest materials high-throughput virtual screening effort [37]. It has scanned a large number of molecules of organic photovoltaics to find those with high power conversion efficiency (PCE) using quantum-chemical techniques. The target value within this dataset is the PCE of each molecule, and the input is the variable-length sequential character data that represents the molecule structures. Previous studies have handcrafted [13, 42, 79] or learned finger-print features [27] that transforms the raw string data into fixed-size feature for prediction.

We use a VIP with a prior defined by Bayesian LSTM (200 hidden units) and $\alpha = 0.5$. We replicate the experimental settings in [13, 42], except that our method directly takes raw sequential molecule structure data as input. We compare our approach with variational dropout for LSTM (VDO-LSTM) [30], alpha-variational inference LSTM (α-LSTM, [59]), BB-α on BNN [42], VI on BNN [10], FITC GP and deep GP trained with expectation propagation (DGP, [13]). Results for the latter 4 methods are directly obtained from [13, 42]. Results in Figure 5 show that VIP significantly outperforms other baselines and hits a state-of-the-art result in both test likelihood and RMSE.

Table 5: Performance on clean energy dataset

Metric	**VIP**	**VDO-LSTM**	**α-LSTM**	**BB-α**	**VI-BNN**	**FITC-GP**	**EP-DGP**
Test NLL	**0.65±0.01**	1.24±0.01	2.06±0.02	0.74±0.01	1.37±0.02	1.25±0.00	0.98±0.00
Test RMSE	**0.88±0.02**	0.93±0.01	1.38±0.02	1.08±0.01	1.07±0.01	1.35±0.00	1.17±0.00

F Related research

In the world of nonparametric models, Gaussian Processes (GPs) [84] provides a principled and flexible probabilistic framework for Bayesian inference over functions. The Bayesian nature enables GPs to provide accurate uncertainty estimates on unseen data, all wrapped up in a single, *exact* closed form solution of posterior inference. Despite the success and popularity of GPs in the past decades, their $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space complexities make them impractical for large-scale datasets. Therefore, people often resort to complicated approximate methods [14,15,18,40,81,87,91,95,99,102]. Another intrinsic issue is the limited representational power of GP kernels. It has been argued that stationary kernels commonly used for nonlinear regressions are not able to obtain hierarchical representations for high dimensional data, which limits the usefulness of GP methods [8].

On the contrary, in the world of parametric modeling, it is well-known that deep neural networks [7, 25,44,52,52,88,94] are extremely flexible function approximators that enable learning from very high-dimensional and structured data. Nowadays Deep learning has been widely spread to an enormous amount of tasks, including computer vision [25, 52, 94] and speech recognition [17, 49, 55, 74]. As people starts to apply deep learning techniques to critical applications such as automatic driving and health care, uncertainty quantification of neural networks has become increasingly important. Although decent progress has been made [4, 10, 22, 35, 41, 46, 59, 76], uncertainty in deep learning still remains an open challenge.

Research in the *Gaussian process-neural net correspondance* has been extensively explored in order to improve the understandings of both worlds. Bayesian neural nets (BNNs) with infinitely wide hidden layers and certain prior distributions have been studied from a GP perspective. e.g. [75,76,105] for single-hidden layer, and [29, 38, 58, 69] for deeper nets. Notably, in [29, 75] a one-layer Bayesian neural network with non-linearity $\sigma(\cdot)$ and mean-field Gaussian prior is approximately equivalent to a Gaussian process with kernel function

$$\mathcal{K}_{\mathrm{VDO}}(\mathbf{x}_1, \mathbf{x}_2) = \mathbb{E}_{p(\mathbf{w})p(b)}[\sigma(\mathbf{w}^\top \mathbf{x}_1 + b)\sigma(\mathbf{w}^\top \mathbf{x}_2 + b)]. \tag{F.1}$$

Later [58] and [69] generalized this result and proved that deep Bayesian neural networks is approximately equivalent to a Gaussian process with a *compositional kernel* [16, 21, 39, 78] that mimic the deep net. These approaches allow us to construct expressive kernels for GPs [51], or conversely, exploit the *exact* Bayesian inference on GPs to perform exact Bayesian prediction for deep neural nets [58]. We will compare the above kernel with equation (D.4) in Appendix F.1.

A number of alternative schemes have also been investigated to exploit deep structures for GP model design. These include: (1) *deep GPs* [13, 19, 20], where compositions of GP priors are proposed to represent prior over compositional functions; (2) the search and design of kernels for accurate and efficient learning [3, 6, 26, 90, 100, 103], and (3) *deep kernel learning* that uses deep neural net features as the inputs to GPs [1, 12, 45, 47, 106]. Frustratingly the first two approaches still struggle to model high-dimensional structured data such as texts and images; and the third approach is not fully Bayesian, i.e. the model is only Bayesian wrt. the last output layer.

Our work is in different spirit of the above two: the intension is not to understand BNNs as GPs nor to use the deep learning concept to help GP design. Instead we directly model a BNN as an instance of implicit processes (IPs), and the GP is used as a *variational distribution* to assist predictive inference.[4] Therefore it also retains some of the benefits of Bayesian nonparametric approaches. This variational approximation does not require previous assumptions in the GP-BNN correspondence literature (infinite hidden units, i.i.d. weight priors, etc) [58, 69] nor the conditions in compositional kernel literature. Instead the optimal kernel (D.4) in the sleep phase applies to *any* IP that includes BNNs and *beyond*. A very recent work [28] also minimizes the sleep phase KL divergence (D.1) but wrt. the BNN prior, and their goal is to regularize BNN priors and implant some smoothness properties of GPs to BNNs. By contrast, our approach takes the advantage from the BNN prior over functions to better encode rich structures. Also [28] still performs variational posterior inference in weight space, and our inference method in function space also allows better uncertainty quantification.

From the practical point of view, the proposed inference method is computationally cheap, and it allows scalable learning of hyper-parameters. The $\mathcal{O}(S^3)$ additional cost is negligible when the computation is dominated by the evaluation of e.g. BNN function samples. In the case where GP

[4]In principle any process can be used here as the variational distribution, and we use GPs here for the convenience of analytic approximations.

approximations dominate the cost, our approach does not require expensive matrix inversions, nor complicated kernel compositions that only have analytic forms under restricted cases [51, 58].

Finally, concurrent work of neural processes [32] studies a special case of the IP, which corresponds to the neural sampler in our experiments in Section E.2. However, inference is conducted in the latent variable $\mathbf{z}$ space using the variational auto-encoder approach [50, 86], with the inference network parameterized in a similar way as PointNet [80]. By contrast, the proposed VIP approach applies to any implicitly defined process, and performs inference in function space. In the experiments we also show improved accuracies of the VIP approach on neural samplers over many existing Bayesian approaches.

F.1 Further discussions on Bayesian neural networks

Following previous section, we provide a further discussion on the comparison between our kernel in equation (D.4), and the kernel proposed in [29], which is the most similar one found in the literature. Notably, consider a Gaussian process $\mathcal{GP}(0, \mathcal{K}_{\text{VDO}}(\cdot,\cdot))$, where $\mathcal{K}_{\text{VDO}}$ is defined as in F.1. [29] considered approximating this GP with a one-hidden layer BNN $\hat{y}(\cdot) = \text{BNN}(\cdot, \theta)$ with θ collecting the weights and bias vectors of the network. Denote the weight matrix of the first layer as $\boldsymbol{W} \in \mathbb{R}^{D\times K}$, i.e. the network has K hidden units, and the kth column of $\boldsymbol{W}$ as $\mathbf{w}_k$. Similarly the bias vector is $\boldsymbol{b} = (b_1, ..., b_K)$. We further assume the prior distributions of the first-layer parameters are $p(\boldsymbol{W}) = \prod_{k=1}^K p(\mathbf{w}_k)$ and $p(\boldsymbol{b}) = \prod_{k=1}^K p(b_k)$, and use mean-field Gaussian prior for the output layer. Then this BNN constructs an approximation to the GP kernel as:

$$\tilde{\mathcal{K}}_{\text{VDO}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{K}\sum_k \sigma(\mathbf{w}_k^\top \mathbf{x}_1 + b_k)\sigma(\mathbf{w}_k^\top \mathbf{x}_2 + b_k), \quad \mathbf{w}_k \sim p(\mathbf{w}), b_k \sim p(b).$$

This approximation is equivalent to the empirical estimation (D.4) when $S = K$ and the implicit process is defined by

$$f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \mathbf{z} = \{\mathbf{w}, b\}, p(\mathbf{z}) = p(\mathbf{w})p(b). \tag{F.2}$$

In such case, the output layer of that one-hidden layer BNN corresponds to the Bayesian linear regression “features” in our final approximation. However, the two methods are motivated in different ways. [29] used this interpretation to approximate a GP with kernel (F.1) using a one-hidden layer BNN, while our goal is to approximate the implicit process F.2 by a GP (note that the implicit process is defined as the output of the *hidden* layer, not the output of the BNN). Also this coincidence only applies when the IP is defined by a Bayesian logistic regression model, and our approximation is applicable to BNN and beyond.

G Further details on the derivations

G.1 Inverse Wishart process as prior for kernel function

Definition 4 (Inverse Wishart processes [92]). *Let Σ be random function $\Sigma(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A stochastic process defined on such functions is called the inverse Wishart process on $\mathcal{X}$ with parameter ν and base function $\Psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, if for any finite collection of input data $\mathbf{X} = \{\mathbf{x}_s\}_{1\le s\le N_s}$, the corresponding matrix-valued evaluation $\Sigma(\mathbf{X}, \mathbf{X}) \in \Pi(N_s)$ is distributed according to an inverse Wishart distribution $\Sigma(\mathbf{X}, \mathbf{X}) \sim IW_S(\nu, \Psi(\mathbf{X}, \mathbf{X}))$. We denote $\Sigma \sim \mathcal{IWP}(v, \Psi(\cdot,\cdot))$.*

Consider the problem in Section D.1 of minimizing the objective

$$\mathcal{U}(m, \mathcal{K}) = \mathcal{D}_{\text{KL}}[p(\mathbf{f}, \mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{f}, \mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot))]$$

Since we use $q(\mathbf{y}|\mathbf{f}) = p(\mathbf{y}|\mathbf{f})$, this reduces $\mathcal{U}(m, \mathcal{K})$ to $\text{D}_{KL}[p(\mathbf{f}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, m, \mathcal{K})]$. In order to obtain optimal solution wrt. $\mathcal{U}(m, \mathcal{K})$, it sufficies to draw S fantasy functions (each sample is a random function $f_s(\cdot)$) from the prior distribution $p(\mathbf{f}|\mathbf{X}, \theta)$, and perform moment matching, which gives exactly the MLE solution, i.e., empirical mean and covariance functions

$$m^\star_{\text{MLE}}(\cdot) = \sum_s \frac{1}{S} f_s(\cdot) \tag{G.1}$$

$$\mathcal{K}^\star_{\text{MLE}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{S}\sum_s [f_s(\mathbf{x}_1) - m^\star(\mathbf{x}_1)][f_s(\mathbf{x}_2) - m^\star(\mathbf{x}_2)] \tag{G.2}$$

In practice, in order to gain computational advantage, the number of fantasy functions S is often small, therefore we further put an inverse wishart process prior over the GP covariance function, i.e. $\mathcal{K}(\cdot,\cdot) \sim \mathcal{IWP}(\nu, \Psi)$. By doing so, we are able to give MAP estimation instead of MLE estimation. Since inverse Wishart distribution is conjugate to multivariate Gaussian distribution, the Maximum A Posteriori(MAP) solution is given by

$$\mathcal{K}^\star_{\text{MAP}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\nu + S + N + 1}\{\sum_s [f_s(\mathbf{x}_1) - m^\star(\mathbf{x}_1)][f_s(\mathbf{x}_2) - m^\star(\mathbf{x}_2)] + \Psi(\mathbf{x}_1, \mathbf{x}_2)\} \tag{G.3}$$

Where N is the number of data points in the training set $\mathbf{X}$ where $m(\cdot)$ and $\mathcal{K}(\cdot,\cdot)$ are evaluated. Alternatively, one could also use Posterior Mean Estimator (which is an example of Bayes estimator that minimizes posterior expected squared loss)

$$\mathcal{K}^\star_{\text{PM}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\nu + S - N - 1}\{\sum_s [f_s(\mathbf{x}_1) - m^\star(\mathbf{x}_1)][f_s(\mathbf{x}_2) - m^\star(\mathbf{x}_2)] + \Psi(\mathbf{x}_1, \mathbf{x}_2)\} \tag{G.4}$$

In the implementation of this paper, we choose $\mathcal{K}_{\text{PM}}$ estimator with $\nu = N$ and $\Psi(\mathbf{x}_1, \mathbf{x}_2) = \psi\delta(\mathbf{x}_1, \mathbf{x}_2)$. The hyper parameter ψ is trained using fast grid search using the same procedure for the noise variance parameter, as detailed in Appendix H.

G.2 Derivation of the upper bound $\mathcal{U}(m, \mathcal{K})$or sleep phase

Applying the chaine rule of KL-divregence, we have

$$\begin{aligned}\mathcal{J}(m, \mathcal{K}) =& \mathcal{D}_{\text{KL}}[p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y}, m(\cdot), \mathcal{K}(\cdot,\cdot))] \\ =& \mathcal{D}_{\text{KL}}[p(\mathbf{f}, \mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{f}, \mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot))] \\ & - \mathcal{D}_{\text{KL}}[p(\mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot))]\end{aligned}$$

Therefore, by the non-negative property of KL divergence, we have

$$\begin{aligned}\mathcal{J}(m, \mathcal{K}) =& \mathcal{D}_{\text{KL}}[p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta)||q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}, \mathbf{y}, m(\cdot), \mathcal{K}(\cdot,\cdot))] \\ \leq& \mathcal{U}(m, \mathcal{K}) = \mathcal{D}_{\text{KL}}[p(\mathbf{f}, \mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{f}, \mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot))]\end{aligned}$$

Finally, since m and $\mathcal{K}$ are the optimal solution of $\mathcal{U}(m, \mathcal{K})$, it is also optimal for $-\mathcal{D}_{\text{KL}}(p(\mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot)))$ under the same samples $\{f_s(\cdot)\}_{s=1}^S$. Therefore not only the upper bound $\mathcal{U}$ is optimized in sleep phase, the gap $-\mathcal{D}_{\text{KL}}(p(\mathbf{y}|\mathbf{X}, \theta)||q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, m(\cdot), \mathcal{K}(\cdot,\cdot)))$ is also decreased when the mean and covariance functions are optimized.

G.3 Fully Bayesian approximation

Here, We further present a fully Bayesian treatment, by introducing a prior distribution $p(\theta)$ on the prior parameters θ, and fitting a variational approximation $q(\theta)$ to the posterior. Sleep phase updates remain the same when conditioned on a given configuration of θ. The α-energy term in wake phase learning becomes

$$\begin{aligned}\log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}) &= \log \int_\theta q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta)p(\theta)d\theta \approx \mathcal{L}^\alpha_{\mathcal{GP}}(q(\mathbf{a}), q(\theta)), \\ \mathcal{L}^\alpha_{\mathcal{GP}}(q(\mathbf{a}), q(\theta)) &= \frac{1}{\alpha}\sum_n^N \log \mathbb{E}_{q(\mathbf{a})q(\theta)}\left[q^\star(y_n|\mathbf{x}_n, \mathbf{a}, \theta)^\alpha\right] - \text{D}_{\text{KL}}[q(\mathbf{a})||p(\mathbf{a})] - \text{D}_{\text{KL}}[q(\theta)||p(\theta)].\end{aligned} \tag{G.5}$$

Therefore, compared with the point estimate case, the only extra term needs to be estimated is $-\mathcal{D}_{\text{KL}}[q(\theta)||p(\theta)]$. Note that, introducing $q(\theta)$ will double the number of parameters. In the case of Bayesian NN as an IP, where θ contains means and variances for weight priors, then a simple Gaussian $q(\theta)$ will need two sets of means and variances variational parameters (i.e., posterior means of means, posterior variances of means,posterior means of variances, posterior variances of variances). Therefore, to make the representation compact, we choose $q(\theta)$ to be a Dirac-delta function $\delta(\theta_q)$.

Another issue of fully Bayesian approach is, one need to specify and tune the form and hyperparameters for $p(\theta)$. To alleviate this, notice that from standard variational lower bound

$$\log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}) \approx \langle \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta) \rangle_{q(\theta)} - \mathcal{D}_{\mathrm{KL}}[q(\theta)||p(\theta)].$$

Then $\mathcal{D}_{\mathrm{KL}}[q(\theta)||p(\theta)]$ can be approximated by

$$\begin{aligned} -\mathcal{D}_{\mathrm{KL}}[q(\theta)||p(\theta)] &\approx - \langle \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta) \rangle_{q(\theta)} + \text{constant} \\ &= - \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta_q) + \text{constant} \end{aligned}$$

Therefore, this gives us a new regulation term $- \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta_q)$, which penalizes the parameter configurations that returns a *full* marginal log likelihood (as opposed to the diagonal likelihood in the original BB-α energy $\frac{1}{\alpha} \sum_n^N \log \mathbb{E}_{q(\mathbf{z})q(\theta)} q_{\mathcal{GP}}(y_n|\mathbf{x}_n, \mathbf{z}, \theta)^\alpha$) that is too high, especially the contribution from non-diagonal covariances. We refer this as *likelihood regularization*. In practice, $- \log q_{\mathcal{GP}}(\mathbf{y}|\mathbf{X}, \theta_q)$ is estimated on each mini-batch.

H Further experimental details

We provide further details as supplementary to the main text.

General settings For small datasets we use the posterior GP equations for prediction, otherwise we use the $\mathcal{O}(S^3)$ approximation. We use $S = 20$ for VIP unless noted otherwise. When the VIP is equipped with a Bayesian NN/LSTM as prior over functions, the prior parameters over each weight are untied, thus can be individually tuned. Fully Bayesian estimates of the prior parameters are used in experiments E.2 and E.4.

H.1 Further implementation details for multivariate regression experiments

- Variational Gaussian inference for BNN (VI-BNN): we implement VI for BNN using the Bayesian deep learning library, ZhuSuan [93]. VI-BNN employs a mean-field Gaussian variational approximation but evaluates the variational free energy using the reparameterisation trick [50]. We use a diagonal Gaussian prior for the weights and fix the prior variance to 1. The noise variance of the Gaussian noise model is optimized together with the means and variances of the variational approximation using the variational free energy.
- Variational implicit process-Neural Sampler regressor (VIP-NS): we use neural sampler with two hidden layers of 10 hidden units. The input noise dimension is 10 or 50, which is determined using validation set.
- Variational dropout (VDO) for BNN: similar to [29], we fix the length scale parameter $0.5 * l^2 = 10e^{-6}$. Since the network size is relatively small, dropout probability is set as 0.005 or 0.0005. We use 2000 forward passes to evaluate posterior likelihood.
- α-dropout inference for BNN: suggested by [59], we fix $\alpha = 0.5$ which often gives high quality uncertainty estimations, possibility due to it is able to achieve a balance between reducing training error and improving predictive likelihood. We use $K = 10$ for MC sampling.
- Variational sparse GPs and exact GPs: we implement the GP-related algorithms using GPflow [70]. variational sparse GPs uses 50 inducing points. Both GP models use the RBF kernel.
- About noise variance parameter grid search for VIPs (including VIP-BNNs and VIP-NSs), VDOs and α-dropout: we start with random noise variance parameter, run optimization on the model parameters, and then perform a (thick) grid search over noise variance parameter on validation set. Then, we train the model on the entire training set using this noise variance parameter value. This coordinate ascent like procedure does not require training the model for multiple times as in Bayesian optimization, therefore can speed up the learning process. The same procedure is used to search for optimal hyperparameter ψ of the inverse-Wishart process of VIPs.

H.2 Additional implementation details for ABC experiment

Following the experimental setting of [77], we set the ground truth L-V model parameter to be $\theta_1 = 0.01, \theta_2 = 0.5, \theta_3 = 1.0, \theta_4 = 0.01$. We simulate population data in the range of $[0, 30]$ with step size 0.05, which result in 600 gathered measurements. We use the first 500 measurements as training data, and the remaining as test set. For MCMC-ABC and SMC-ABC setup, we also follow the implementation[5] of [77] . MCMC-ABC is ran for 10000 samples with tolerance ϵ set to be 2.0 which is manually tuned to give the best performance. In MCMC-ABC, last 100 samples are taken as samples. Likewise SMC-ABC uses 100 particles. Model likelihood is calculated based on Gaussian fit. VIP ($\alpha = 0$) is trained for 10000 iterations with Adam optimizer using 0.001 learning rate.

H.3 Additional implementation details for predicting power conversion efficiency of organic photovoltaics molecules

For Bayesian LSTMs, we put Gaussian prior distributions over LSTM weights. The output prediction is defined as the final output at the last time step of the input sequence. We use $S = 10$ for VIP. All methods use Adam with a learning rate of 0.001 for stochastic optimization. Noise variance parameter are not optimized, but set to suggested value according to [42].To match the run time of the fingerprint-based methods, all LSTM methods are trained for only 100 epochs with a batch size of 250. Among different models in the last few iterations of optimization, we choose the one with the best training likelihood for testing. Note that in the original paper of variational dropout and α-dropout inference, K sample paths ($K = 1$ for VDO and $K = 10$ for α-dropout) are created for *each* training data, which is too prohibitive for memory storage. Therefore, in our implementation, we enforce all training data to share K sample paths. This approximation is accurate since we use a small dropout rate, which is 0.005.

H.4 Solar irradiance prediction results in table

Table 6: Interpolation performance on solar irradiance.

Method	VIP	VDO	SVGP	GP
Test NLL	**0.08±0.02**	0.21 ± 0.04	0.56 ± 0.23	0.832 ± 0.00
Test RMSE	**0.28±0.00**	0.29±0.01	0.55±0.08	0.650±0.0

[5]`https://github.com/gpapamak/epsilon_free_inference`