5th Symposium on Advances in Approximate Bayesian Inference, 2023 1–21

A Study of Bayesian Neural Network Surrogates for Bayesian Optimization

Anonymous Authors
Anonymous Institution

Abstract

Bayesian optimization is a highly efficient approach to optimizing objective functions which are expensive to query. Typically, the objective is represented as a realization of a Gaussian process (GP) surrogate model. However, Bayesian neural networks (BNNs) have recently become practical function approximators, with many benefits over Gaussian processes, such as the ability to naturally handle non-stationarity, and learn representations for high-dimensional data. In this paper, we study Bayesian neural networks as alternatives to Gaussian process surrogates for optimization. We consider a variety of approximate inference procedures for BNNs, ranging from high-quality Hamiltonian Monte Carlo, to stochastic MCMC, to heuristics such as deep ensembles. We also evaluate on diverse problems with varying dimensionality, number of objectives, non-stationarity, and discrete and continuous inputs. While we find that BNNs can outperform GPs when the objective function is non-stationary or the input space high-dimensional, our findings generally support the continued use of GPs for Bayesian optimization, due to their strong priors and exact inference.

1. Introduction

Bayesian optimization (O'Hagan, 1978) is a distinctly compelling success story of Bayesian inference. In Bayesian optimization, we place a prior over the objective we wish to optimize, and infer a posterior predictive *distribution* over the values of the objective at all feasible points in space. We then combine this predictive distribution with an *acquisition function* that trades-off exploration (moving to regions of high uncertainty) and exploitation (moving to regions with a high expected value, for maximization). The resulting approach converges quickly to a global optimum, with strong performance in expensive black-box settings, ranging from experimental design, to learning parameters for simulators, to hyperparameter tuning (Frazier, 2018; Garnett, 2023).

While many acquisition functions have been proposed for Bayesian optimization (e.g., Frazier et al., 2008; Wang and Jegelka, 2017), it is rare to stray from *Gaussian process* (GP) surrogate models for the objective. Indeed, GPs can provide closed-form predictive distributions in regression, without the need for approximate inference. Moreover, GPs are both flexible and can represent strong priors that enable good performance on small datasets (a small number of objective function queries) (Rasmussen and Williams, 2006).

Despite the success of GP surrogate models in Bayesian optimization, it has become timely to consider Bayesian neural network (BNN) surrogates as alternatives to Gaussian processes. Unlike GPs, BNNs require approximate inference, and the priors are not interpretable. However, in recent years there has been extraordinary progress in making BNNs practical (e.g., Wilson and Izmailov, 2020; Daxberger et al., 2021; Khan and Rue, 2021; Tran et al., 2022; Rudner et al., 2022). Moreover, BNNs can learn non-stationary behavior (typical

© A. Authors.

of optimization objectives) without human intervention, discover similarity measures as part of representation learning (useful for higher dimensional inputs), and naturally handle multi-output objectives. In parallel, new *Monte-Carlo* acquisition functions (Balandat et al., 2020) have been developed that only require posterior samples, significantly lowering the barrier to using non-GP surrogates that do not provide closed-form predictive distributions.

In this paper, we evaluate Bayesian neural networks as surrogate models for Bayesian optimization. We consider a variety of approximate inference procedures, ranging from high-quality full-batch Hamiltonian Monte Carlo (HMC) (Neal, 1996, 2010; Izmailov et al., 2021), to stochastic gradient Markov Chain Monte Carlo (Chen et al., 2014), to heuristics such as deep ensembles (Lakshminarayanan et al., 2017), over several architectures. We also exhaustively study a variety of synthetic and real-world objectives—with a wide range of input space dimensionalities, single- and multi-dimensional output spaces, and both discrete and continuous inputs, and non-stationarities.

We find that, in controlled environments, BNNs behave as expected, clearly outperforming GPs in non-stationary and high-dimensional settings. However, across various established synthetic and real-world benchmarking tasks which neither exhibit known non-stationarity nor are sufficiently high-dimensional, the objectives interact with the pros and cons of each approach, leading to mixed results with BNNs and GPs performing similarly well. Curiously, we find that in many instances, there is as much of a difference between different approximate inference procedures for BNNs as there is between BNNs and GPs!

Our findings are optimistic for the future use of BNN surrogates, with the potential to explore challenging problem settings not amenable to GP surrogate models. At the same time, in the relatively data-scarce setting of Bayesian optimization, the convenience and strong priors of standard Gaussian processes remain appealing.

2. Preliminaries & Related Work

We provide a short overview of Bayesian optimization and introduce the two families of surrogate models, Gaussian processes and Bayesian neural networks, in Appendix 1 and discuss related work in Appendix 2.

This paper provides several key contributions in the context of this prior work. GP surrogates are nearly always used with Bayesian optimization. While BNN surrogates have been attempted, they have been limited to SGHMC inference, and applied in specialized settings, without an effort to understand their properties. Little is known about whether BNNs could generally be used as an alternative to GPs for Bayesian optimization, especially in light of more recent general advances in BNN research. This is the first paper to provide a comprehensive study of BNN surrogates, considering a range of approximate inference procedures and experimental settings. We test the utility of BNNs in a variety of contexts, exploring the behavior as we change the dimensionality of the problem and the number of objectives, investigating their performance on nonstationary functions, and also incorporating problems with a mix of discrete and continuous input parameters. Rather than champion a specific approach, we provide an objective assessment, also highlighting the continued benefits of GP surrogates for general Bayesian optimization problems.

3. Empirical Evaluation

We provide an extensive empirical evaluation of BNN surrogates for Bayesian optimization. First, we analyze the effect of key model and inference parameters on the performance of Bayesian optimization in three commonly used synthetic objective functions. Next, to assess how BNNs compare to GPs in relatively simple and well-understood settings, we consider a larger set of commonly used synthetic objective functions and perform an empirical comparison between GPs and BNNs using different inference procedures (HMC, SGHMC, and deep ensembles). To further ascertain whether BNNs may be a suitable alternative to GPs in real-world Bayesian optimization problems, we study six real-world datasets used in prior work on Bayesian optimization with GP surrogates (Maddox et al., 2021; Eriksson et al., 2019; Dreifuerst et al., 2021; Wang et al., 2020; Oh et al., 2019). We conclude our evaluation with a case study of Bayesian optimization tasks where simple Gaussian process models may fail but simple BNN models would be expected to prevail. To this end, we design a set of controlled experiments to assess the performance of GPs and BNNs as a function of the input dimensionality and in settings where the objective function is non-stationary.

3.1. Neural Network Surrogate Models

First, we conduct a sensitivity study into the effect of key design choices for BNN surrogate models. Gaussian processes involve relatively few design choices — essentially only the covariance function, which is often simply set to the RBF kernel. By contrast, with BNNs we must consider the architecture, the prior over parameters, and the approximate inference procedure. Moreover, while certain architectures have powerful inductive biases for computer vision and language tasks, generic regression tasks, such as Bayesian optimization tend to be well-suited for fully-connected multi-layer perceptrons (MLPs), which have relatively mild inductive biases (loose assumptions about the structure of the function we are modeling).

For Gaussian process models, we have an intuitive understanding of the induced distributions over functions for different choices of covariance functions, but it is less clear how different modeling choices in BNNs affect the inferred posterior predictive distributions. To illustrate these differences for varying network, prior, and variance parameters, we plot the inferred posterior predictive distributions over functions for different network widths and depths, the activation functions, and likelihood and variance parameters in Figure A.1 and evaluate the performance under different model choices for three synthetic data problems in Figure A.5 (see Appendix 5.1).

3.2. Synthetic Benchmarks

We evaluate BNN and GP surrogates on a variety of synthetic benchmarks. The results can be found in Appendix 3.1.

3.3. Real-World Benchmarks

To provide an evaluation of BNN surrogates in more realistic optimization problems, we consider a diverse selection of real-world applications which span a variety of domains such as solving differential equations and monitoring cellular network coverage (Maddox et al., 2021; Eriksson et al., 2019; Dreifuerst et al., 2021; Wang et al., 2020; Oh et al., 2019). Many

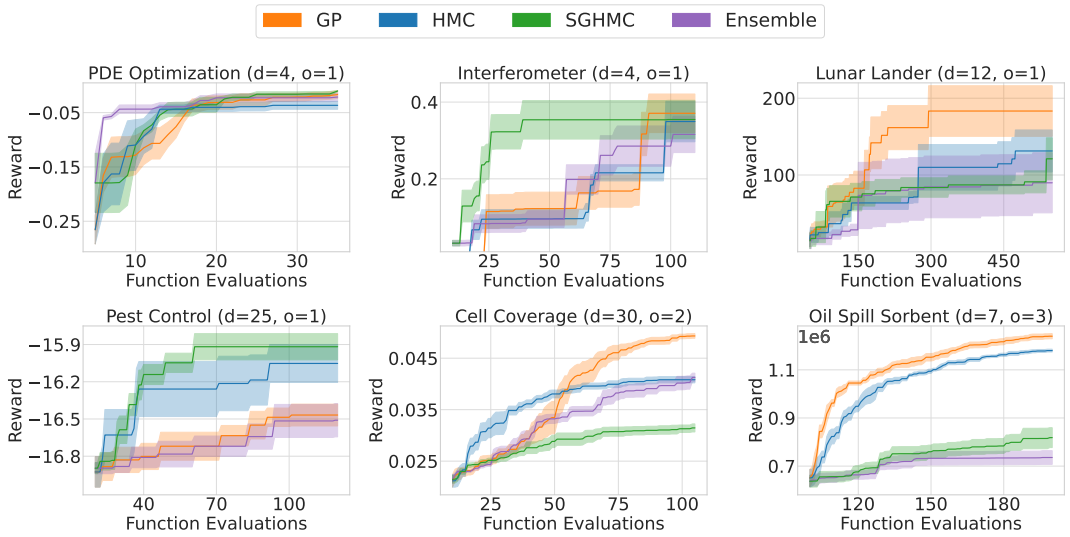


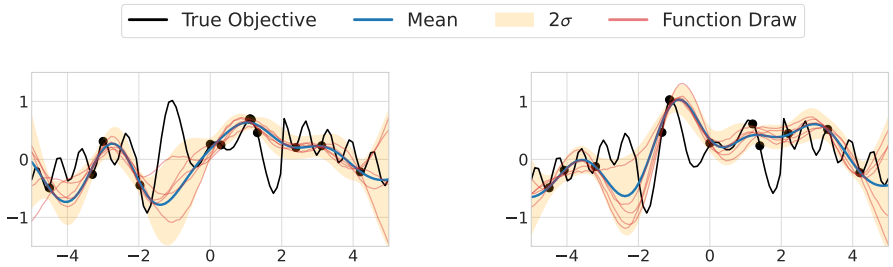
Figure 1: **Real world benchmarks show mixed results**. There does not seem to be a noticeable preference for GPs or BNNs as we increase the number of input dimensions. Additionally, there does not appear to be a clear separation between the top row of experiments, which optimize over continuous parameters, and the bottom row of experiments, which also include some discrete inputs. We plot the mean and one standard error of the mean over 5 trials.

of these problems, such as the development of materials to clean oil spills, have consequential applications; however, these objectives are often multi-modal and are difficult to optimize globally. Additionally, unlike the synthetic benchmarks, many real-world applications consist of input data with ordinal or categorical values, which may be difficult for GPs to handle. Several of the problems require multiple objectives to be optimized. Detailed problem descriptions are provided in Appendix 4.

We share the results of our experiments in Figure 1, and details about the experiment setup can be found in Appendix 5. The results are mixed: There are some problems where BNNs outperform GPs, and others where GPs find significantly larger rewards. The pest control, cell coverage, and oil spill sorbent experiments all include discrete input parameters; however, there is no clear pattern in the behavior of the different surrogate models. Similar to the findings from the synthetic benchmarks, we see that the different approximate inference methods lead to significantly different Bayesian optimization performance: SGHMC outperforms the other BNNs in two experiments, while performing worse than full HMC in others. Additionally, it appears that GPs outperform BNNs in the two multi-objective problems, although that may be due to confounding factors.

3.4. Limitations of GP Surrogate Models

Although popular, GPs suffer from well-known limitations that directly impact their usefulness as surrogate models. To contrast BNN and GP surrogates, we explore two failure modes of GPs and demonstrate that the increased flexiblity provided by BNN surrogate models can overcome these issues and improve performance in Bayesian optimization.



(a) GP (b) HMC

Figure 2: **GPs struggle to find the global maximum for nonstationary functions**. After 20 function evaluations, the GP does not accurately model the true function around $x = -1$ because is unable to account for the sudden increase in scale due to its assumption of stationarity. BNNs do not suffer from the same pathologies and are able to find the true maximum.

3.4.1. Non-Stationary Objective Functions

To use GP surrogates, we must specify a kernel function class that governs the covariance structure over data points. This kernel has typically has the form $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$, because it is easier to describe the functional form and learn the hyperparameters. Because the covariance between two values only depends on their distance and not on the values themselves, this setup assumes the function is stationary and has similar mean and smoothness throughout. Unfortunately, this assumption does not hold true in many real-world settings. For example, in the common Bayesian optimization application of choosing hyperparameters of a neural network, the true loss function landscape may have vastly different behavior in one part of the input space compared to another. BNN surrogates, in contrast to GP surrogates, are able to learn non-stationary functions without similar constraints.

In Figure 2(a), we show the performance of Bayesian optimization over a synthetic nonstationary function with both BNNs and GP surrogate models for a non-stationary objective function. Because the GP assumes that the behavior of the function is the same throughout the input domain, it cannot accurately model the input-dependent variation of the function and leads to underfitting around the true optimum.

In contrast, BNN surrogates do not suffer from this limitation and can learn non-stationary functions without similar constraints. As expected, in Figure 2(b) we see that, unlike the GP, the BNN surrogate is able to accurately find the global minimum and closely follow the contours of the function.

3.4.2. High-Dimensional Input Spaces

Due to the curse of dimensionality, GPs do not scale well to high-dimensional input spaces without careful human intervention. Common covariance functions may fail to faithfully represent high-dimensional input data, making the design of custom-tailored kernel functions necessary. In contrast, neural networks are well-suited for modeling high-dimensional input data (Krizhevsky et al., 2012).

To measure the effects of dimensionality on the performance of GPs and BNNs, we create a synthetic objective function using a function draw from a BNN with 2 hidden layers of 256

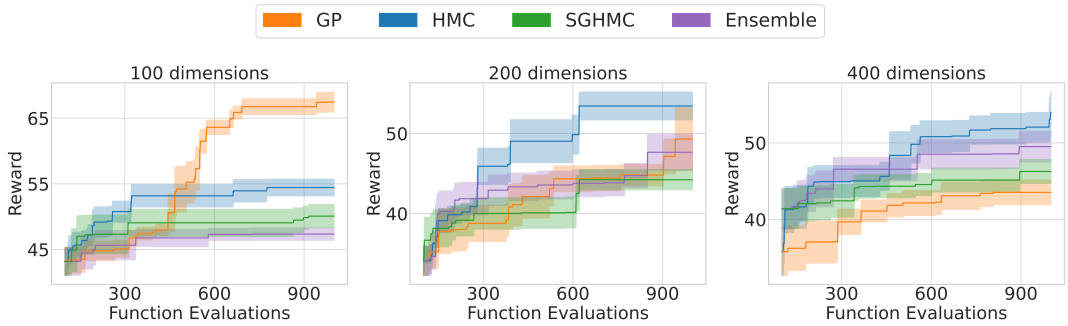


Figure 3: **BNNs outperform GPs in high-dimensional settings**. For each experiment, we use a fixed prior function draw from a neural network as the objective function. This objective function is high-dimensional and may be non-stationary. Although GPs perform well in 100 dimensions, they are not able to find good rewards in 200 and 400 dimensions. We plot the mean and its standard error over five trials.

parameters each. With this setup, we are able to easily create problems with high-dimensional inputs. Additionally, we expect these function draws to exhibit non-stationarity.

We experiment with 100, 200, and 400 input dimensions to see how the performance of GP and BNNs as surrogate models varies as we increase the dimensionality of the data, and we share results in Figure 3 and Figure A.4. We see that although GP surrogates outperform BNNs for 100-dimensional inputs, BNNs consistently find better values than GPs for 200- and 400-dimensional data. This result supports the hypothesis that HMC can perform well in higher-dimensional input spaces and provides a compelling motivation for BNNs as surrogate models for Bayesian optimization.

4. Discussion

In this work, we provide the first comprehensive study of Bayesian neural network surrogate models for Bayesian optimization, comparing three approximate inference procedures. We evaluate GP and BNN surrogate models on synthetic benchmarking tasks, real-world optimization problems, and controlled experiments designed to highlight potential shortcomings of GP versus BNN surrogates.

We show that BNNs with full HMC perform similarly to GP surrogates on many synthetic and real-world optimization problems, but that the design of BNN surrogates can significantly impact performance, and different inference methods may lead to significantly different results. While GP surrogates outperform BNNs on some problems, and BNN surrogates outperform GPs on others, our results show that BNN surrogates (especially with full HMC) are a competitive surrogate model choice for many real-world, and especially for high-dimensional and non-stationary, Bayesian optimization problems.

Our findings are optimistic for the future of BNN surrogates for Bayesian optimization and unlock new applications over high-dimensional input spaces. Our findings also highlight the sustained appeal of GP surrogates, which—due to their strong priors and exact inference—remain relatively effective on a wide range of Bayesian optimization tasks.

References

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20, 2007.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Tianqi Chen, Emily B Fox, Carlos Guestrin, and Michael I Jordan. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pages 1683–1691, 2014.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 2021.

Ryan M Dreifuerst, Samuel Daulton, Yuchen Qian, Paul Varkey, Maximilian Balandat, Sanjay Kasturia, Anoop Tomar, Ali Yazdan, Vish Ponnampalam, and Robert W Heath. Optimizing coverage and capacity in cellular networks using machine learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8138–8142. IEEE, 2021.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Peter I Frazier. A tutorial on bayesian optimization. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 2008.

Jacob Gardner, Chuan Guo, Kilian Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for bayesian optimization. In *Artificial Intelligence and Statistics*, 2017.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, May 2013. ISSN 1532-4435.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, 2015.

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljacic. Deep learning for bayesian optimization of scientific problems with high-dimensional structure. *Transactions of Machine Learning Research*, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25:*, pages 1106–1114, 2012.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Wesley J Maddox, Maximilian Balandat, Andrew G Wilson, and Eytan Bakshy. Bayesian optimization with high-dimensional outputs. *Advances in Neural Information Processing Systems*, 34:19274–19287, 2021.

Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. Boss: Bayesian optimization over string spaces. *Advances in Neural Information Processing Systems*, 2020.

Radford M Neal. Bayesian Learning for Neural Networks. 1996.

Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th international conference on, intelligent systems application to power systems*, pages 84–91. IEEE, 2005.

Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

A O'Hagan. On curve fitting and optimal design for regression. *J. Royal Stat. Soc. B*, 40: 1–32, 1978.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32*. 2019.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

Tim G. J. Rudner, Freddie Bickford Smith, Qixuan Feng, Yee Whye Teh, and Yarin Gal. Continual Learning via Sequential Function-Space Variational Inference. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2022.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

Dmitry Sorokin, Alexander Ulanov, Ekaterina Sazhina, and Alexander Lvovsky. Interferobot: aligning an optical interferometer by a reinforcement learning agent. *Advances in Neural Information Processing Systems*, 33:13238–13248, 2020.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29, 2016.

Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *International Conference on Machine Learning*, 2022.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Dustin Tran, Jeremiah Liu, Michael W. Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Karan Singhal, Zachary Nado, Joost van Amersfoort andAndreas Kirsch, Rodolphe Jenatton, Nithum Thain, Honglin Yuan, Kelly Buchanan, Kevin Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan. Plex: Towards Reliability Using Pretrained Large Model Extensions. 2022.

Boqian Wang, Jiacheng Cai, Chuangui Liu, Jian Yang, and Xianting Ding. Harnessing a novel machine-learning-assisted evolutionary algorithm to co-optimize three characteristics of an electrospun oil sorbent. *ACS Applied Materials & Interfaces*, 12(38):42842–42849, 2020.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, 2016.

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. *Advances in neural information processing systems*, 30, 2017.

Supplementary Material

1. Preliminaries

We provide a short overview of Bayesian optimization and introduce the two families of surrogate models: Gaussian processes and Bayesian neural networks.

1.1. Bayesian Optimization

Bayesian optimization is a global optimization method commonly used for black-box functions (for which we do not have a closed-form expression) that are costly to evaluate. More specifically, Bayesian optimization seeks the solution $\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$, where $\mathbf{x}$ represents all possible inputs to objective function f, using as few queries of the objective function f as possible. To do so, Bayesian optimization iteratively selects new function evaluations using the following procedure: (1) specify a *surrogate model* to find $p(f \mid \mathcal{D})$, where $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^t$ and y_i is a noisy observation of $f(\mathbf{x}_i)$; (2) select the next $\mathbf{x}_{t+1}$ to query by maximizing an easy-to-evaluate *acquisition function* $a(\cdot)$, which typically uses the mean and variance of the posterior $p(f \mid \mathcal{D})$; and (3) query the expensive-to-evaluate objective function f at value $\mathbf{x}_{t+1}$ to observe $y_{t+1} \sim \mathcal{N}(f(\mathbf{x}_{t+1}), \sigma^2_{\text{obs}})$ and add the result to the dataset $\mathcal{D} := \mathcal{D} \cup \{\mathbf{x}_{t+1}, y_{t+1}\}$. (See tutorials Frazier (2018) or Brochu et al. (2010).)

In multi-objective Bayesian optimization (Swersky et al., 2013), the goal is to find the input $\mathbf{x}^*$ that maximizes k related objective functions $\mathcal{F} = \{f_1(\mathbf{x}), \cdots, f_k(\mathbf{x})\}$. Since there is often no single solution that maximizes all K objectives simultaneously, performance is typically compared using *Pareto dominance* (Ngatchou et al., 2005), where one solution Pareto-dominates another if it performs equally well or better on all objectives. Pareto dominance can be measured using *hypervolume*, which indicates how much of a bounded region is dominated by a solution.

The choice of acquisition function and surrogate model will greatly impact the performance of Bayesian optimization. The acquisition function determines the trade-off between exploitation and exploration, and uses criteria such as the probability of improvement or the expected improvement over the currently observed maximum (Jones et al., 1998).

The selected surrogate model will also have a significant impact on the performance of Bayesian optimization. Since the true function f may take a variety of forms, the surrogate model should be flexible enough to represent it accurately. Additionally, the uncertainty estimates that the surrogate model provides should be well-calibrated to ensure that the input that maximizes f can be found in a limited number of Bayesian optimization iterations. Gaussian processes are currently the canonical surrogate model for Bayesian optimization (Shahriari et al., 2016).

1.2. Gaussian Processes

GPs Gaussian processes (GPs) (Rasmussen and Williams, 2006) are distributions over functions entirely specified by a mean function $\mu(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. For regression tasks with GPs, we assume $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$ with $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is the likelihood variance. The functions $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and kernel functions of the GP that govern the functional properties and generalization performance of

the model. In practice, we tune the hyperparameters of $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ on the training data to optimize the *marginal likelihood* of the GP, which maximizes the probability that the GP model will have generated the data.

By the definition of a GP, for any finite collection of inputs $\mathbf{x} = [\mathbf{x}_{\text{train}}, \mathbf{x}_{\text{test}}]$, $f(\mathbf{x})$, and thus $y(\mathbf{x})$ are jointly normal. Therefore, we can apply the rules of conditioning partitioned multivariate normals to form a posterior distribution $p\left(f(\mathbf{x}_{\text{test}}) \,|\, y(\mathbf{x}_{\text{train}})\right)$, which will also be Gaussian.

In the context of Bayesian optimization, this simple conditioning procedure means that given some set of points at which we have already queried the objective function, we can quickly generate a posterior over potential candidate points and, with the help of an acquisition function, select the next points to query the objective.

1.3. Bayesian Neural Networks

Bayesian neural networks (BNNs) are neural networks with stochastic parameters for which a posterior distribution is inferred using Bayesian inference. More specifically, for regression tasks, we can specify a Gaussian observation model, $p(y(\mathbf{x}) \mid \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(h(\mathbf{x} \mid \boldsymbol{\theta}), \sigma^2)$, where $y(\mathbf{x})$ represents a noisy observed value and $h(\mathbf{x} \mid \boldsymbol{\theta})$ represents the output of a neural network with parameter realization $\boldsymbol{\theta}$ for input $\mathbf{x}$. For BNNs, a prior distribution is placed over the stochastic parameters $\boldsymbol{\theta}$ of the neural network, and the corresponding posterior distribution is given by $p(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\mathcal{D} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) / p(\mathcal{D})$. The posterior predictive distribution for any evaluation point $\mathbf{x}_{\text{test}}$ is $p(y(\mathbf{x}_{\text{test}}) \,|\, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(y(\mathbf{x}_{\text{test}}) \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}) d\boldsymbol{\theta}$. However, since neural networks are non-linear in their parameters, exact inference over the stochastic network parameters is analytically intractable. To enable Bayesian inference in neural network models, approximate inference methods can be used to find approximations to the exact posterior $p(\boldsymbol{\theta} \,|\, \mathcal{D})$.

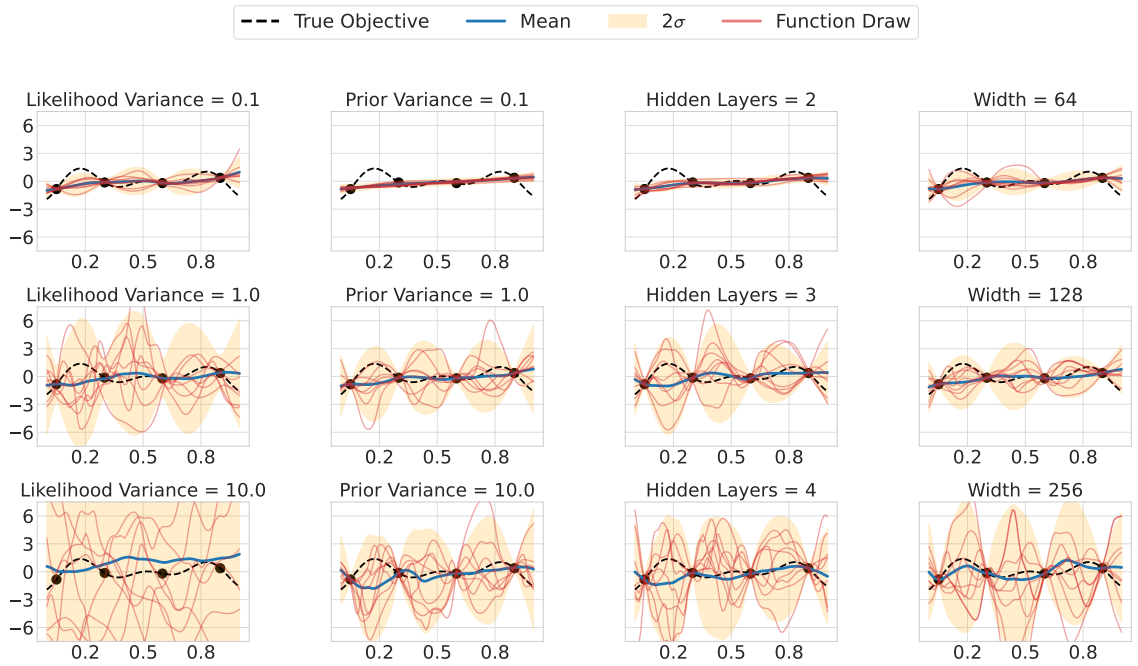
In this work, we focus on three types of approximate inference: Hamiltonian Monte Carlo (HMC; Neal (2010)), stochastic-gradient HMC (Chen et al., 2014), and deep ensembles of deterministic neural networks (Lakshminarayanan et al., 2017).

Hamiltonian Monte Carlo. Hamiltonian Monte Carlo is a Markov Chain Monte Carlo method that produces asymptotically exact samples from the posterior distribution (Neal, 2010) and is commonly referred to as the "gold standard" for inference in BNNs. At test time, HMC approximates the integral $p(y(\mathbf{x}_{\text{test}}) \,|\, \mathcal{D}) \approx \frac{1}{M} \sum_{i=1}^{M} p(y(\mathbf{x}_{\text{test}}) \,|\, \boldsymbol{\theta}_i)$ using samples $\boldsymbol{\theta}_i$ drawn from the posterior over parameters. Full-batch HMC provides the most accurate approximation of the posterior distribution but is computationally expensive and in practice limited to models with only a few hundred thousand parameters (Izmailov et al., 2021). Full-batch HMC allows us to study the performance of BNNs in Bayesian optimization without many of the confounding factors of inaccurate approximations of the predictive distribution.

Stochastic Gradient HMC. Stochastic gradient methods (Welling and Teh, 2011; Hoffman et al., 2013) are commonly used as an inexpensive approach to sampling from the posterior distribution. Unlike full-batch HMC, which computes the gradients over the full dataset, stochastic gradient HMC instead samples a mini-batch to compute a noisy estimate of the gradient (Chen et al., 2014). While these methods may seem appealing when working

with large models or datasets, they can result in inaccurate approximations and posterior predictive distribution with unfaithful uncertainty representations.

Deep Ensembles. Deep ensembles are ensembles of several deterministic neural networks, each trained using maximum a posteriori estimation using a different random seed and initialization (and sometimes using different subsets of the training data). The ensemble components can be viewed as forming an efficient approximation to the posterior predictive distribution, by choosing parameters that represent typical points in the posterior and have high functional variability (Wilson and Izmailov, 2020). Deep ensembles are easy to implement in practice and have been shown to provide highly accurate predictions and a good predictive uncertainty estimate (Lakshminarayanan et al., 2017; Ovadia et al., 2019).



(*a*) Likelihood Variance (*b*) Prior Variance (*c*) Network Depth (*d*) Network Width

Figure A.1: **The design of the BNN has a significant impact on the uncertainty estimates.** We visualize the uncertainty estimates and function draws produced by full-batch HMC on a simple toy objective function with four function queries (denoted in black). For each set of experiments, we fix all other design choices with the following base parameters: likelihood variance = 1, prior variance = 1, number of hidden layers = 3, and width = 128. We see that varying the different aspects of the model leads to significantly different posterior predictive distributions.

2. Related Work

There is a large body of literature on improving the performance of Bayesian optimization. For a comprehensive review, see Garnett (2023).

An overwhelming majority of this research uses Gaussian process surrogate models, focusing instead on developing new acquisition functions (e.g., Frazier et al., 2008; Wang and Jegelka, 2017), additive covariance functions (Gardner et al., 2017; Kandasamy et al., 2015), the inclusion of gradient information (Wu et al., 2017), multi-objectives (Swersky et al., 2013), trust region methods that use input partitioning for higher dimensional and non-stationary data (Eriksson et al., 2019), and covariance functions for discrete inputs and strings (Moss et al., 2020).

Besides Gaussian process surrogates, random forests (Hutter et al., 2011) and tree-structured Parzen estimators have been used (Bergstra et al., 2013). Moreover, Snoek et al. (2015) apply a Bayesian linear regression model to the last layer of a deterministic neural network for computational scaling that can be helpful for the added number of objective queries associated with higher dimensional inputs. Deep kernel learning (Wilson et al., 2016), which transforms the inputs of a Gaussian process kernel with a deterministic neural network, is sometimes also used with Bayesian optimization, especially in higher dimensional settings where it is not clear what covariance function to use, such as protein engineering (Stanton et al., 2022).

Despite the extraordinary recent practical advances in developing Bayesian neural networks (BNNs) for many tasks (e.g., Wilson and Izmailov, 2020), and recent Monte-Carlo acquisition functions which make it easier to use surrogates like BNNs that do not provide closed-form predictive distributions (Balandat et al., 2020), there is a vanishingly small body of work that uses BNNs as surrogates for Bayesian optimization. This is surprising since we would indeed expect BNNs to have properties naturally aligned with Bayesian optimization, such as the ability to learn non-stationary functions without explicit modeling interventions, and gracefully handle high-dimensional input and output spaces.

The possible first attempt to use a Bayesian neural network surrogate for Bayesian optimization (Springenberg et al., 2016) came before most of these advances in BNN research, and used a form of stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) for inference. Like Snoek et al. (2015), the focus was largely on scalability advantages over Gaussian processes; however, the reported gains were marginal, and puzzling in that they were largest for a *small* number of objective function queries (where the neural net would not be able to learn a rich representation, and scalability would not be required). Kim et al. (2021) used the same method for BNNs with Bayesian optimization, also with SGHMC, targeted at scientific problems with known structures and high dimensionality. In these applications, BNNs leverage auxiliary information, domain knowledge, and intermediate data, which would not typically be available in Bayesian optimization.

3. Further Empirical Results

3.1. Synthetic Benchmarks

We evaluate BNN and GP surrogates on a variety of synthetic benchmarks and we choose problems with a wide span of input dimensions to understand how the performance differs as we increase the dimensionality of the data. We include the details of the experiment setup in Appendix 5.

In the single-objective case, **Branin** is a function with two-dimensional inputs with three global minima, **Hartmann** is a six-dimensional function wtih six local minima, and **Ackley** is a multi-dimensional function with many local minima which is commonly used to test optimization algorithms. We convert all problems to maximization problems, and the goal of Bayesian optimization in single-objective settings is to find the maximum value of the objective function.

For multi-objective benchmarks, **BraninCurrin** is a two-dimensional input and two-objective problem composed of the Branin and Currin functions, and **DTLZ1** and **DTLZ5** are multi-dimensional and multi-objective test functions which are used to measure an optimization algorithm's ability to converge to the Pareto-frontier. Here, the goal is to find an input corresponding to the multi-dimensional objective value with the maximum hypervolume from a problem-specific reference point.

As shown in Figure A.2, we found that the choice of approximate inference method plays a critical role in the performance of BNNs. In general, BNNs using full HMC show promising results and are comparable to GPs, performing very similarly in many of the experiments, slightly underperforming in Hartmann and BraninCurrin, and outperforming standard GPs in the 10-dimensional Ackley experiment. However, the performance of SGHMC varies significantly, matching the performance of HMC and GPs in some experiments while failing to approach the maximum value in others, and deep ensembles underperform the other surrogate models, plateauing at low objective values, on all six benchmarking tasks.

3.2. Limitations of Gaussian Process Surrogate Models

Multi-objective: GPs have been successfully extended to a wide range of multi-objective problems. However, in the interest of making the approaches scalable, there are many assumptions placed onto the kernel. In the most naive setting, we can model each objective independently, so an n-objective problem would be modeled using n independent GPs. While this approach is highly scalable, it completely ignores the relationship between objective values and has no notion of shared structure, so it is unable to take advantage of all of the information in the problem.

Multi-objective covariance functions can also be decomposed as Kronecker product kernels (Bonilla et al., 2007). Thus we compute the covariance as $k(x_t, x'_{t'}) = k_T(t, t') \otimes k_X(x, x')$, where k_T is a kernel function that describes the covariance over tasks, and k_X is a kernel describing the covariance over inputs. While this approach has significant computational advantages, it requires each objective to itself be modeled with the same underlying kernel.

Thus, this method of modeling multiple objectives will fail to capture the nuances of each particular objective when the functions have differing properties. In Figure A.3 we show the result of twenty iterations of Bayesian optimization over a toy multi-task example, where we

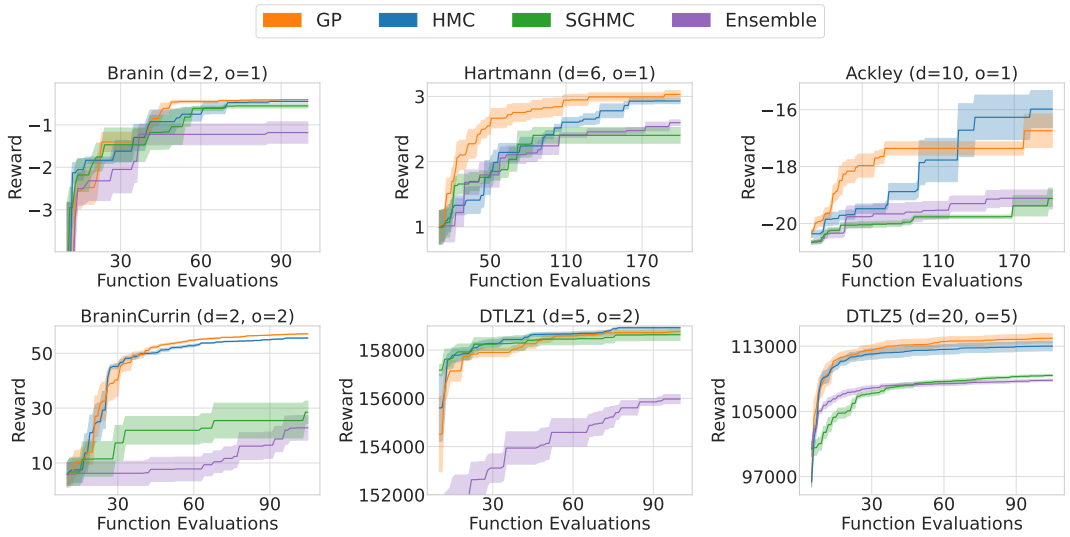
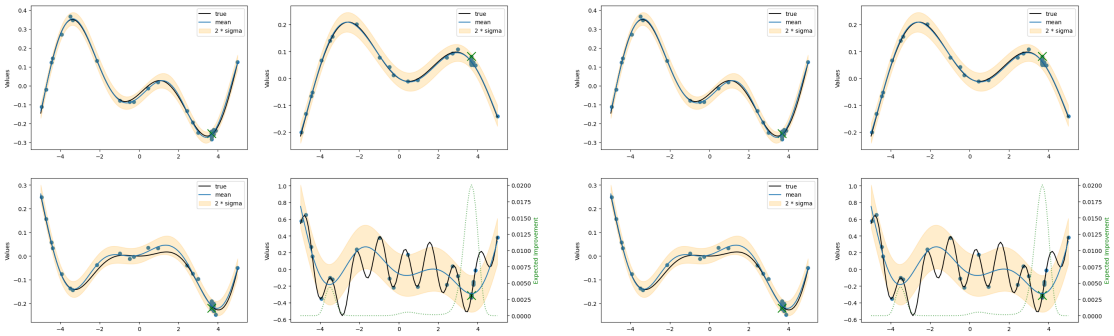


Figure A.2: **HMC is often comparable to GPs on synthetic benchmarks.** In contrast, SGHMC and deep ensembles have less reliable performance and are often times unable to effectively find the maximum. For all problems, the choice of approximate inference method has a significant impact on the performance of BNNs. We plot the mean and one standard error of the mean over 5 trials.

care about optimizing over the fourth function but provide additional information through the other three datasets. Such settings are common when we are taking measurements in the real world: We may primarily care about optimizing one objective, but we may have access to a variety of related data and objectives. These objectives may have correlation or structure, and it is sensible to use a multi-objective model.

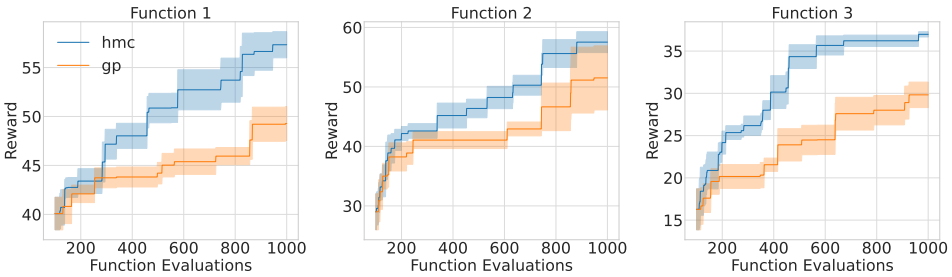
In our experiment, although the GP is able to learn the proper length scale and variance over the three additional functions with similar length scales, it struggles to accurately fit the fourth objective. Because the GP is unable to account for the differences between the four functions, it does not find the global optimum. Unlike GPs, BNNs are not restricted to stationary covariance structures and are able to produce well-calibrated uncertainty estimates in multi-objective settings. As shown in Figure A.3(b), BNNs are able to accurately fit all four functions, including the fourth objective function which has a much smaller length scale compared to the others.

High-dimensional: We include additional high-dimensional results in Figure A.4 which compare the performance of GPs and HMC in 200 and 400 dimensions. We use a function draw from a neural network as the objective function.

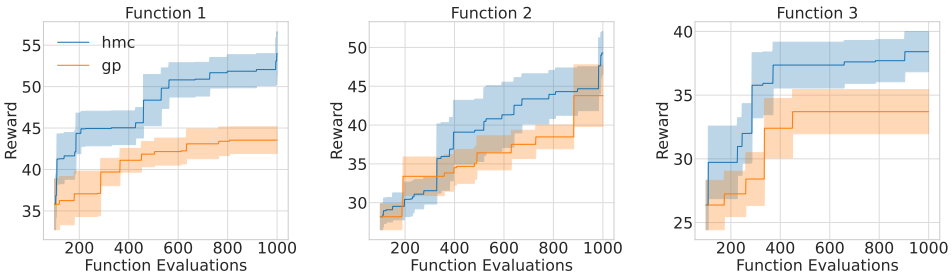


(*a*) GP (*b*) HMC

Figure A.3: **GPs have a hard time finding the global minimum in multi-objective settings**. Multi-task GPs learn one length scale across all objectives, which may not be suitable for many datasets. In this example, it does not find the global minimum in the 4th objective because it treats the shorter length scale as noise. In contrast, BNNs are able to appropriately model the uncertainty for the 4th objective and find its true minimum.



(*a*) 200 dimensions



(*b*) 400 dimensions

Figure A.4: **BNNs outperform GPs in high-dimensional settings**. We use a prior function draw from a neural network as the objective function. This objective function is high-dimensional and may be non-stationary. HMC consistently outperforms GPs in 200 and 400 dimensions.

4. Real-World Dataset Descriptions

Interferometer In this problem, the goal is to tune an optical interferometer through the alignment of two mirrors. We use the simulator in Sorokin et al. (2020) to replicate the Bayesian optimization problem in Maddox et al. (2021). Each mirror has a continuous x and y coordinate, which should be optimized so that the two mirrors reflect light with minimal amounts of interference.

Lunar Lander Following Eriksson et al. (2019), we aim to learn the parameters of a controller for a lunar lander as implemented in OpenAI gym. The controller has 12 continuous input dimensions, corresponding to events such as "change the angle of the rover if it is tilted more than x degrees." The objective is to maximize the average final reward over 50 randomly generated environments.

Cellular Coverage We want to optimize the cellular network coverage and capacity from 15 cell towers (Dreifuerst et al., 2021). Each tower has a continuous parameter corresponding to transmit power parameter and an ordinal parameter with 6 values corresponding to tilt, for a total of 15 continuous and 15 ordinal input parameters. There are two different objectives: maximize the cellular coverage, and minimize the total interference between cell towers.

Oil Spill Sorbent In this problem, we optimize the properties of a material to maximize its performance as a sorbent for marine oil spills (Wang et al., 2020). We tune 5 ordinal parameters and 2 continuous parameters which control the manufacturing process of electrospun materials, and optimize over three objectives: water contact angle, oil absorption capacity, and mechanical strength.

Pest Control Minimizing the spread of pests while minimizing the prevention costs of treatment is an important problem with many parallels (Oh et al., 2019). In this experiment, we define the setting as a categorical optimization problem with 25 categorical variables corresponding to stages of intervention, with 5 different values at each stage. We optimize over two objectives: minimizing the spread of pests and minimizing the cost of prevention.

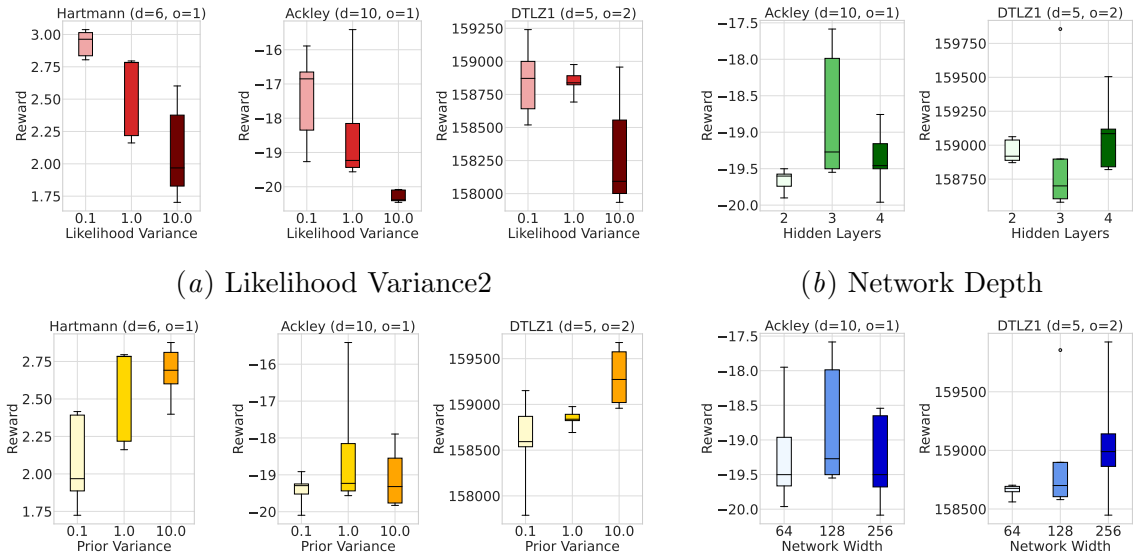
PDE Optimization Here, following Maddox et al. (2021), we optimize 4 continuous parameters corresponding to the diffusivity and reaction rates of a Brusselator with spatial coupling. The objective of the problem is to minimize the variance of the PDE output.

5. Experiment Details

5.1. Model Design

Model Hyperparameters. We consider isotropic priors over the neural network parameters with zero mean and variance parameters 0.1, 1, and 10. Similarly, we consider Gaussian likelihood functions with variance parameters 0.1, 1, and 10. The corresponding posterior predictive distributions for full-batch HMC are shown in Figure A.1(a). As would be expected, an increase in the likelihood variance results in a poor fit of the data and virtually no posterior collapse. In contrast, increasing the prior variance results in a higher predictive variance between data points with a good fit to the data points, whereas a prior variance that is too small leads to over-regularization and uncertainty collapse. As shown in Figure A.5(a) and Figure A.5(c), lower likelihood variance parameters and larger prior variance parameters tended to perform best across three synthetic data experiments.

Network Width and Depth. To better understand the effects of the network size on inference, we explore the differences in performance when varying the number of hidden layers and the number of parameters per layer, each corresponding to an increase in model complexity. In Figure A.1(c) and Figure A.1(d), we can see the increase in uncertainty as we increase the width and the depth of the network. However, the best size to choose seems to be problem-dependent, as shown in Figure A.5(b) and Figure A.5(d).



(a) Likelihood Variance2

(b) Network Depth

(c) Prior Variance

(d) Network Width

Figure A.5: **There is no single BNN design that performs the best across all problems.** We compare the impact of the design on the BO performance for different benchmark problems. For each set of experiments, we fix all other aspects of the design and plot the values of the maximum reward found using HMC after 100 function evaluations over 10 trials.

5.2. General Setup

For all datasets, we normalize input values to be between [0, 1] and standardize output values to have mean 0 and variance 1.

GP: For single-objective problems, we use GPs with a Matérn 5/2 kernel, adaptive scale, a length-scale prior of Gamma$(3, 6)$, and an output-scale prior of Gamma$(2.0, 0.15)$. For multi-objective problems, we use the same GP to independently model each objective.

HMC: We use HMC with an adaptive step size, and we search over different architectures and model choices.

SGHMC: We use SGHMC with minibatch size of 5 using the same model parameters as found in our full SGHMC search.

Deep ensembles: We use an ensemble of 10 MLPs with 3 hidden layers, 128 parameters wide, tanh activation, likelihood variance of 1.0, and prior variance of 1.0. Each model is trained on a random 80% of the function evaluations.

We run multiple trials for all experiments, where each trial starts with a different set of initial function evaluations drawn using a sobol sampler.

5.3. Synthetic Benchmarks

Branin We ran 5 trials using batch size 5 with 10 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 10.0.

Hartmann We ran 5 trials using batch size 10 with 10 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 1.0.

Ackley We ran 5 trials using batch size 10 with 10 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 0.1.

BraninCurrin We ran 5 trials using batch size 10 with 10 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 10.0.

DTLZ1 We ran 5 trials using batch size 5 with 10 initial points. Our SGHMC model was an MLP with 2 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 10.0.

DTLZ5 We ran 5 trials using batch size 1 with 10 initial points. Our SGHMC model was an MLP with 5 hidden layers, 10 parameters per layer, tanh activation, likelihood variance of 0.001, and prior variance of 10.0.

5.4. Real-World Benchmarks

PDE Optimization We ran 5 trials using batch size 1 with 5 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 1.0, and prior variance of 1.0.

Interferometer We ran 10 trials using batch size 10 with 10 initial points. Our SGHMC model was an MLP with 5 hidden layers, 100 parameters per layer, tanh activation, likelihood variance of 10.0, and prior variance of 10.0.

Lunar Lander We ran 5 trials using batch size 50 with 50 initial points. Our SGHMC model was an MLP with 5 hidden layers, 100 parameters per layer, tanh activation, likelihood variance of 10.0, and prior variance of 10.0.

Pest Control We ran 5 trials using batch size 4 with 20 initial points. Our SGHMC model was an MLP with 5 hidden layers, 100 parameters per layer, tanh activation, likelihood variance of 10.0, and prior variance of 10.0.

Cell Coverage We ran 5 trials using batch size 5 with 10 initial points. Our SGHMC model was an MLP with 3 hidden layers, 128 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 0.1.

Oil Spill Sorbent We ran 5 trials using batch size 10 with 10 initial points. Our SGHMC model was an MLP with 5 hidden layers, 100 parameters per layer, tanh activation, likelihood variance of 0.1, and prior variance of 10.0.