Barcelona School of Economics

Master Program in Data Science Methodology

"Bayesian Optimization with Uncertainty Aware Neural Networks"

Ampudia, David; Leung, Clinton

Supervisor: Stojic, Hrvoje

May 2022

ABSTRACT IN ENGLISH:

Bayesian optimization has emerged as an effective and efficient approach for finding the global optimum of highly complex derivative-free black-box functions. It typically models the objective function with Gaussian processes (GP) as a surrogate. Based on this surrogate, an auxiliary acquisition function proposes candidate optima locations to query the objective function at. In this paper, we explore recent developments that may help alleviate two key limitations of GP's: poor performance with large datasets, and non-stationary target functions. To this end, we propose and implement several scalable uncertainty aware neural networks as alternative surrogates. In a series of tests, we showcase the relative performance of ensembles, Bayesian, and direct estimation neural network approaches against that of traditional GP's and state of the art Sparse Variational Gaussian Processes (SVGP) in Bayesian optimization settings. Our results show that not only are neural networks a scalable solution with comparable performance to GP's, but they also hold the potential to outperform SVGP's.

ABSTRACT IN SPANISH:

La optimización bayesiana ha surgido como una alternativa eficaz y eficiente para encontrar el óptimo global de funciones sin derivadas y altamente complejas. Por lo general, la optimización bayesiana modela la función objetivo con procesos gaussianos (PG) como sustituto. En función de este sustituto, una función de adquisición auxiliar propone posibles ubicaciones óptimas para consultar la función objetivo. En este artículo, exploramos los desarrollos recientes que pueden ayudar a aliviar dos limitaciones clave de los PG: bajo rendimiento cuando contamos con grandes cantidades de datos, y funciones objetivo no estacionarias. Con este fin, proponemos e implementamos una serie de alternatives en forma de redes neuronales capaces de cuantificar la incertidumbre del modelo y que permiten trabajar con datos de mayor dimensionalidad. En una serie de pruebas y escenarios de optimización bayesiana, mostramos el rendimiento de modelos neurales de conjuntos (ensembles), bayesianos y de estimación directa de la incertidumbre y los comparamos a los PG tradicionales y a los procesos gaussianos variacionales dispersos (PGVD) de última generación. Nuestros resultados muestran que las redes neuronales implementadas no solo son una solución escalable con un rendimiento comparable al de los PG, sino que también tienen el potencial de superar a los PGVD.

ABSTRACT IN CATALAN:

La optimització bayesiana ha aparegut com una alternativa eficaç i eficient per trobar l'òptim global de funcions sense derivades i molt complexes. Normalment, la optimització bayesiana modela la funció objectiu amb processos gaussians (PG) com a substitut. En funció d'aquest substitut, una funció d'adquisició auxiliar proposa possibles ubicacions òptimes per consultar la funció objectiu. En aquest article, explorem els desenvolupaments recents que poden ajudar a al·leviar dues limitacions clau dels PG: rendiment baix quan disposem de grans quantitats de dades, i funcions objectiu no estacionàries. Amb aquest fi, proposem i implementem una sèrie d'alternatives en forma de xarxes neuronals amb capacitat de quantificar l'incertesa del model i que permeten treballar amb dades de major dimensionalitat. En una sèrie de proves i escenaris d'optimització bayesiana, mostrem el rendiment de models neurals de conjunts (ensembles), bayesians i d'estimació directa de l'incertesa i els comparem amb els PG tradicionals i amb els processos gaussians variacionals dispersos (PGVD) de darrera generació. Els nostres resultats mostren que les xarxes neuronals implementades no només són una solució escalable amb un rendiment comparable als PG, sinó que també tenen el potencial de superar als PGVD.

KEYWORDS IN ENGLISH:

Bayesian optimization, Gaussian processes,
Surrogate models,
Uncertainty aware neural networks,
Ensembles,
Direct estimation neural networks,
Sparse Variational Gaussian Processes (SVGP)

KEYWORDS IN SPANISH:

Optimización bayesiana,
Procesos gaussianos,
Modelos sustitutos,
Redes neuronales capaces de cuantificar la incertidumbre,
Conjuntos,
Estimación directa de la incertidumbre,
Procesos gaussianos variacionales dispersos (PGVD)

KEYWORDS IN CATALAN:

Optimització bayesiana,
Processos gaussians,
Models substituts,
Xarxes neuronals amb capacitat de quantificar l'incertesa,
Conjunts,
Estimació directa de l'incertesa,
Processos gaussians variacionals dispersos (PGVD)

Bayesian Optimization with uncertainty-aware neural networks

Clinton Leung* David Ampudia Vicente*

Supervisor: Hrvoje Stojic

Abstract

Bayesian optimization has emerged as an effective and efficient approach for finding the global optimum of highly complex derivative-free black-box functions. It typically models the objective function with Gaussian processes (GP) as a surrogate. Based on this surrogate, an auxiliary acquisition function proposes candidate optima locations to query the objective function at. In this paper, we explore recent developments that may help alleviate two key limitations of GP's: poor performance with large datasets, and non-stationary target functions. To this end, we propose and implement several scalable uncertainty aware neural networks as alternative surrogates. In a series of tests, we showcase the relative performance of ensembles, Bayesian, and direct estimation neural network approaches against that of traditional GP's and state of the art Sparse Variational Gaussian Processes (SVGP) in Bayesian optimization settings. Our results show that not only are neural networks a scalable solution with comparable performance to GP's, but they also hold the potential to outperform SVGP's.

I Introduction

Bayesian optimization (BO) is a powerful method that has been successfully applied to a diverse array of fields such as hyperparameter tuning for automated machine learning (Snoek et al., 2015), robotics (Lizotte et al., 2007), combinatorial optimization (Hutter et al., 2011), molecular search Hernández-Lobato et al. (2017), and experimental design (Azimi et al. 2012, Greenhill et al. 2020). Simplistic procedures such as systematic or randomized grid search have traditionally been employed (Bergstra and Bengio, 2012), but as objective functions become more complex and more expensive to evaluate, these naive approaches become intractable. Bayesian optimisation sidesteps these difficulties by using a surrogate model to approximate the objective function. Then, through a Bayesian decision framework, it queries the search space in an informed manner to efficiently find the global optimum (Shahriari et al. 2016, Frazier 2018).

Gaussian process-based Bayesian optimisation has been applied successfully in a myriad of small-data and stationary settings due to GP's flexibility and ability to represent model uncertainty. However, recent years have seen a growing demand for optimizing increasingly complex problems,

*Barcelona School of Economics

which typically require a significant number of function evaluations to find a high-quality optimum such as highly noisy problems (Jalali et al., 2017) or complex problems with large search spaces (Griffiths and Hernández-Lobato, 2017). Unfortunately, the computational cost of Gaussian processes is known to scale cubicly with the number of observations, making it prohibitively expensive to use with large evaluation budgets. Additionally, GP's are ill-equipped to deal with non-stationary data (Damianou and Lawrence, 2013). These shortcomings motivate the exploration for alternative easy-to-scale surrogates that are capable of handling non-stationary target functions.

Recently, Vakili et al. (2021) proposed Sparse Variational Gaussian Processes (SVGP) as a scalable alternative to vanilla GP-based BO. Through approximation, SVGP's are able to address the primary concern of high computational cost. However, SVGP's still do not address issues of non-stationarity. On the other hand, artificial neural networks are both highly scalable and flexible allowing them to handle both large data and non-stationarities. Unfortunately, neural networks provide no straightforward approach for quantifying model uncertainty, which has traditionally limited their use in Bayesian optimization. Considered one of the great challenges for machine learning research, the last decade has seen a surge in the literature that studies methods for quantifying uncertainty of black-box functions, both generally (Romano et al., 2019) and specifically in the context of neural networks (Gawlikowski et al., 2021).

In this work, we explore the novel application of four neural networks that are equipped with the necessary tools to estimate model uncertainty, making them valid alternatives to GP's. In particular, we consider deep ensembles, an ensemble method by Lakshminarayanan et al. (2017); Monte Carlo dropout, a pseudo-Bayesian method put forth in Gal and Ghahramani (2016); and two deterministic methods: deep evidential regression (Amini et al., 2020) and direct epistemic uncertainty prediction (Jain et al., 2021). Although other works have compared the varying performance of these models in accurately quantifying uncertainty (Henne et al. 2020, Abdar et al. 2021), it is unclear whether this behavior necessarily translates into better performance in the context of Bayesian optimisation. Outside of Bayesian optimisation, accuracy of uncertainty quantification is compared in absolute terms, but within Bayesian optimisation, this may be superfluous. It may be sufficient to have representations of relative accuracy of uncertainty to sufficiently guide exploration to the optimum.

We benchmark these models on a number of classical BO functions with large and small evaluation budgets to test their scalability and efficacy in handling non-stationary settings. We evaluate their performance relative to a random sampling baseline, traditional GP's and state-of-the-art SVGP's. Our primary contributions can be summarized as follows. We first provide evidence that neural networks are viable substitutes for GP's in Bayesian optimization by showing that they have comparable performance in most settings. Second, we show that neural networks scale better than GP's when evaluation budgets are large. Finally, we find that neural networks outperform SVGP's for some non-stationary noisy functions with large evaluation budgets, suggesting the former are viable alternatives to state-of-the-art surrogate models.

The rest of this article is structured as follows. Section II provides a detailed introduction to Bayesian optimization, the types of uncertainty in neural networks and an overview of the existing literature. Section IV describes the theoretical grounds and practical implementation of the proposed uncertainty-aware artificial neural networks. Section V presents the experimental framework and results of a series of tests on benchmark noiseless and noisy multi-dimensional functions. Section VI concludes.

II Bayesian optimization

Bayesian optimisation consists of a statistical model that defines a distribution over the unknown function $\mathbf{f}$ from the input space $\mathcal{X}$ and an acquisition function that determines the sequence of new input queries $\mathbf{x} \in \mathcal{X}$ that may yield a global optimum $f(\mathbf{x}^*)$, i.e.

$$\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \tag{1}$$

Given a prior over the functional form of $\mathbf{f}$ and a set of n observations of input-output pairs $\mathcal{D}_n = \{(\mathbf{x}_i, \mathbf{f}(\mathbf{x}_i))\}_{i=1}^n$, Bayesian optimization makes use of the posterior of the surrogate function $\widehat{\mathbf{f}}$ to map likely outcomes of $\mathbf{f}$ for some new candidate $\mathbf{x}_{n+1}$, while the acquisition function evaluates the potential of these candidate points to guide the next best step $\mathbf{x}_{n+1}$. After some N number of iterations, the algorithm outputs the set of coordinates $\hat{\mathbf{x}}$ that yield the best estimate of the global optimum. See Algorithm 1 for the pseudo-code of the algorithm.[1]

Algorithm 1 Bayesian optimization

```
1: observe $\mathbf{f}(\mathbf{x})$ at $n_0$ initial points
2: for $n = 1, 2, \dots$ do
3:     select new $\mathbf{x}_{n+1}$ by optimizing acquisition function $\alpha$
           $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$
4:     query objective function to observe $y_{n+1} = \mathbf{f}(\mathbf{x}_{n+1})$
5:     $\mathcal{D}_{n+1} \leftarrow \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
6:     update the posterior distribution on $\mathbf{f}$
7: return $\hat{\mathbf{x}}$ with highest $\mathbf{f}(\hat{\mathbf{x}})$
```

Surrogate models. The default choice for a probabilistic surrogate model are GP's, which are flexible enough to model complex functions, quantify model uncertainty, and exhibit sufficient structure to avoid over-fitting (Jones 2001, Osborne et al. 2009). Two significant shortcomings of GPs are their scalability, and their suitability to non-stationary problems. In particular, inverting the kernel matrix for exact inference with GP's is computationally $O(n^3)$[2], which limits its ability in BO when the dimensionality of the search space or model complexity require a large number of observations to achieve a high quality solution. Furthermore, standard GP's have covariance functions that induce a uniform level of smoothness over the function space, which makes them ineffective for non-stationary functions.

Sparse Variational Gaussian Processes (SVGP's) are the current state-of-the-art method for addressing scalability, bypassing the cubic limitation by approximating the GP posterior through a fixed set of inducing points (Hensman et al., 2013). Recently, Vakili et al. (2021) showed that sparse processes can achieve the same regret bounds as classical processes with Thompson sampling, and

[1]See Garnett (2022) for an in-depth review of the literature on Bayesian optimization.

[2]Details about this result are derived in Appendix Section VII.A.

showed in their experiments that Bayesian optimization using SVGP outperforms their classical counterparts when the number of function evaluations is unbounded.

Nonetheless, SVGP's do not address issues of non-stationarity. One classical approach is to formulate the covariance function as a non-stationary kernel, but this requires a high degree of parametrization which may make it unsuitable for higher dimensional problems quickly. Another approach is to use multiple local stationary covariance functions to capture the non-stationarities. A third approach is to warp the input space with a non-linear mapping into a more stationary environment. These two latter methods are limited when dealing with computationally expensive or sparse data settings common to Bayesian optimisation (Hebbal et al., 2019).

On the other hand, neural networks are known to both scale and handle non-stationarities well. In this article, we study the use of several uncertainty aware neural network, presented in Section IV, and their efficacy as surrogate models.

Acquisition functions. An optimization policy guides sequential exploration of the objective function using the surrogate model. In each step, the acquisition policy uses the surrogate model and its current beliefs about the objective function to propose the next point to evaluate. The acquisition function seeks to trade off exploration and exploitation: a policy that over-explores may be prohibitively inefficient and delay convergence, while over-exploitation may lead to excessive evaluation of local optima. In order to balance between the two, the policy requires some reasonable estimate of out-of-sample model uncertainty to help guide exploration beyond the space of initial points. The literature has put forth many acquisition functions, and in this paper we consider two popular candidates: Expected Improvement and Thompson Sampling. Both approaches are easy to evaluate, which allow for more efficient optimization over the original unknown function.

Expected Improvement combines a greedy one-step look-ahead policy with a simple reward utility function, where for a posterior mean function of the surrogate $\mu_{\mathcal{D}_n}$ we define the utility as $u\left(\mathcal{D}_n\right) = \max \mu_{\mathcal{D}_n}(\mathbf{x})$. The Expected Improvement acquisition function then chooses x such that it maximizes Equation 2, ie. it selects $x \in \underset{x' \in \mathcal{X}}{\arg\max}\, \alpha_{\mathrm{EI}}\left(x'; \mathcal{D}_n\right)$ as the next query point.

$$\alpha_{\mathrm{EI}}(x; \mathcal{D}_n) = \int \left[\max \mu_{\mathcal{D}_{n+1}}\left(\mathbf{x}'\right)\right] p(y \mid x, \mathcal{D}_n) \mathrm{d}y - \max \mu_{\mathcal{D}_n}(\mathbf{x}) \tag{2}$$

Alternatively, Thompson sampling takes a non-deterministic sampling approach. Instead of trying to maximize over the entire posterior distribution of the objective function, this approach samples random realizations of the unknown objective function by drawing from its posterior distribution given the available data $\mathcal{D}_n$ (Garnett, 2022), see Equation 3. The next query point is acquired over the sampled candidates, using the most promising realization.

$$\alpha_{\mathrm{TS}}(x; \mathcal{D}_n) \sim p(f \mid \mathcal{D}_n) \tag{3}$$

Conveniently, Thompson sampling can be easily expanded to batch sampling, where some k realizations of the posterior are drawn and evaluated simultaneously, yielding a batch of k new query locations. Since, there is significant overhead for training a neural network, it is quite wasteful to optimize with only a single new observation, as would be the case with vanilla Thompson sampling.

Hence, the combination of batch sampling and neural networks can provide significant computational advantages. Although this precipitates a coarser schedule of optimization, the loss in accuracy is typically made up for by the gains in computational costs.

Using the surrogate model to capture beliefs about the unknown distribution, and some easy-to-evaluate acquisition function that optimizes the search, Bayesian optimization efficiently seeks out the global optimum. It sequentially updates its posterior beliefs in a Bayesian fashion, and upon termination infers the optimum using the posterior, which can be written as:

$$p(f|\mathcal{D}) = \int p(f \mid \mathcal{X}, \phi)\, p(\phi|\mathcal{D})\, d\phi \tag{4}$$

where ϕ represents the surrogate model. In the following sections, we seek to identify recent advances in neural networks that are most useful in the above framework, and provide evidence on their relative performance based on a series of tests on different data.

III Related Work

The literature on Bayesian optimization originated with work by Kushner (1964), A.G. (1975) and Mockus et al. (1978), but the algorithm was first popularized by Jones et al. (1998) in their work on the Efficient Global Optimization schedule. Their work paved the way for many innovations in the field, including multi-objective optimization (Knowles 2006, Keane 2006), multi-fidelity optimization (Sobester et al., 2004) and the analysis of convergence rates (Calvin and Zilinskas, 2000). In turn, Bayesian optimization impacted a broad range of other literature strains, including robotics (Lizotte et al., 2007), automatic machine learning (Bergstra and Bengio 2012, Swersky et al. 2013, Snoek et al. 2015), experimental design (Azimi et al., 2012), reinforcement learning (Brochu et al., 2010) and model simulation (Salemi et al. 2014, Kleijnen 2015, Mehdad and Kleijnen 2018).

Standard deep neural networks may scale substantially better than GP's (Bottou and Bousquet, 2008), but are agnostic about model uncertainty. In a bid to address the latter, Mullachery et al. (2018) developed Bayesian Neural Networks (BNN's). Unfortunately, although BNN's are theoretically well founded, they are typically computationally intractable to compute the posterior over any sizeable network, giving them limited usefulness. Snoek et al. (2015), building from work by Neal (1995) and Williams (1996), put forth a solution that consisted of replacing only the final hidden layer in a deep neural network with a Bayesian linear regressor. This theoretically amounted to forming an adaptive basis function with the first $K-1$ layers to feed into a simple Bayesian linear regression, forming the so-called Deep Networks for Global Optimization (DNGO) method. In practical terms, this meant the computational cost was reduced to growing linearly with the number of observations and cubicly with the dimensionality of the $K-1$ basis function which determine the size of the matrix to be inverted in the added Bayesian linear regression layer. The authors show this is computationally cheaper than GP's, and is also able to explicitly trade off between computational cost and model expressivity by adjusting the size of the basis function. Experiments showed that with a small budget of 200 evaluations, BNN-based BO in place of vanilla GP achieved comparable results.

Alternative work by Hernandez-Lobato and Adams (2015) showed that probabilistic backpropagation (PBP) is more cost-efficient than Bayesian neural networks. PBP works by propagating probabilities forward to obtain the marginal likelihood, and then backpropagating the gradients of the marginal likelihood with respect to the parameters of the posterior approximation. In a follow-up paper, Hernández-Lobato et al. (2017) are able to further improve the computational tractability of PBP through parallelization of Thompson Sampling, which does not depend on the underlying model structure. Although, intrinsic computational time is not reduced, distributed Thompson Sampling is able to reduce the wall clock time.

Beyond the scope of Bayesian optimization, extensive research has been conducted in recent years in broader attempts at equipping deep learning models with the tools of uncertainty quantification, see Abdar et al. (2021) for a survey of recent developments. In the next section we consider salient examples of uncertainty-aware deep learning methods, which we study and contrast in subsequent sections.

IV Uncertainty-Aware Neural Networks

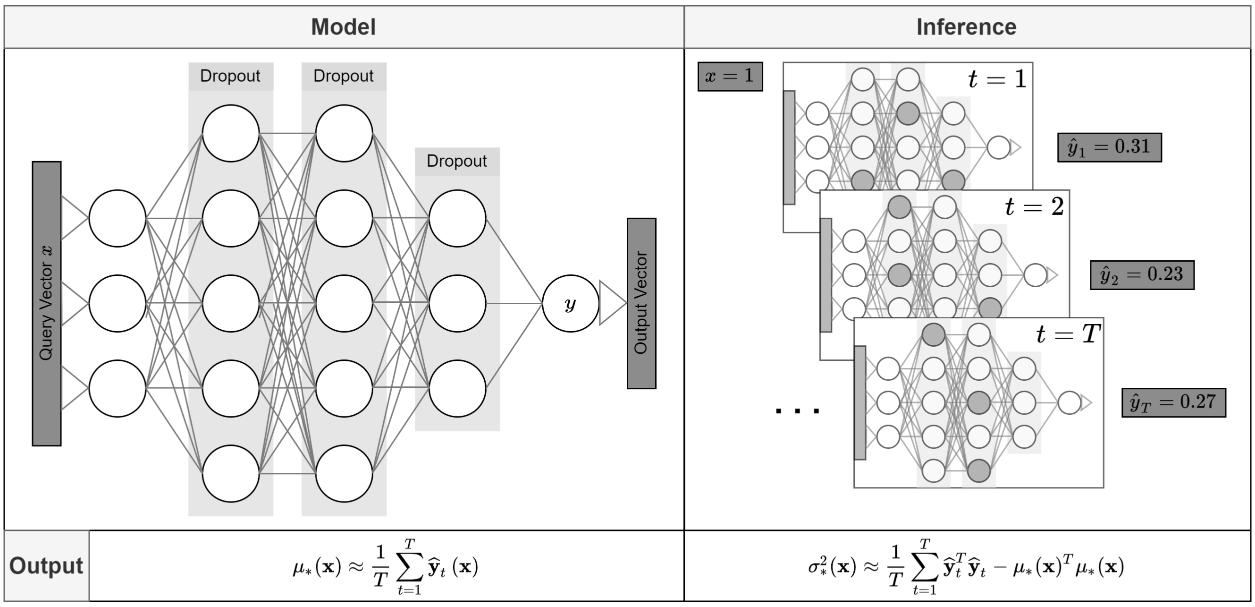
Neural networks scale much better than vanilla Gaussian processes, making them an attractive choice for Bayesian optimisation with large evaluation budgets. Moreover, while GP's are not apt for modelling non-stationary target functions, neural networks' high degree of expressivity make them quite suitable. An additional advantage of using neural networks as surrogate models is that they are particularly capable of learning in batches, which facilitates the use of parallel acquisition functions (Hernández-Lobato et al., 2017). Despite these attractive features, the use of neural networks as surrogate models in Bayesian optimisation has long been considered infeasible due to the lack of tractable strategies to quantify uncertainty.

Recent advances in uncertainty aware neural networks has made their use in Bayesian optimisation potentially feasible. Although some of the recent approaches have been shown to produce superior uncertainty estimates over others (Osband et al., 2021), the extent to which this may affect performance in Bayesian optimization is not explored in the literature. In this section, we present four different neural network architectures that we will evaluate in the Bayesian optimisation setting: deep ensembles, an ensemble method by Lakshminarayanan et al. (2017); Monte Carlo dropout, a pseudo-Bayesian method put forth in Gal and Ghahramani (2016); deep evidential regression (Amini et al., 2020); and direct epistemic uncertainty prediction (Jain et al., 2021). Details of their implementation are in Appendix Section VII.E.

IV.A Monte Carlo Dropout

Dropout is a standard regularisation technique that is employed to prevent neural networks from over-fitting (Srivastava et al., 2014). At training time, each input of a feed-forward layer will either be dropped with probability p or used with probability $(1 - p)$. This is analogous to bootstrapping the model into exponentially many different architectures. Normally, dropout is disabled at inference time to provide consistent predictions, but Gal and Ghahramani (2016) suggest that dropout may be repurposed to approximate a probabilistic deep Gaussian process (Damianou and Lawrence, 2013).

In practice, this implies making T stochastic forward passes with dropout enabled at inference time, and estimating the first two empirical moments of the distribution by using the sample mean and variance of the T forward passes (Figure 1).



Model
Inference
Dropout
Dropout
Dropout
Query Vector x
y
Output Vector
$x = 1$
$t = 1$
$t = 2$
$t = T$
$\hat{y}_1 = 0.31$
$\hat{y}_2 = 0.23$
$\hat{y}_T = 0.27$
Output
$\mu_*(\mathbf{x}) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{y}}_t(\mathbf{x})$
$\sigma_*^2(\mathbf{x}) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{y}}_t^T \hat{\mathbf{y}}_t - \mu_*(\mathbf{x})^T \mu_*(\mathbf{x})$

Figure 1: Monte Carlo Dropout Architecture

Monte Carlo Dropout approximates Bayesian inference by maintaining Bernoulli dropout at both training and inference time. The mean and variance of the posterior distribution can be characterised by the sample mean and sample variance of T stochastic samples.

Monte Carlo Dropout has proven particularly successful in the applied literature due to its ease of implementation. Although MC Dropout has been shown to produce poorer estimates of uncertainty than some of the alternative approaches considered here, it is not entirely clear that it would not be effective in a Bayesian optimisation setting. Furthermore, since samples can be easily drawn by repeatedly conducting forward passes, it is cheap and straightforward apply large batch sampling with Thompson sampling.

IV.B Deep Ensembles

Deep ensembles are ensembles of deep neural networks that have been found to have a good representation of uncertainty. Lakshminarayanan et al. (2017) has shown that deep ensembles performs at least as well as Bayesian neural network approaches, including Monte Carlo Dropout. Beluch et al. (2018) and Gustafsson et al. (2020) similarly find ensembles to be more reliable and applicable than Bayesian networks in practical scenarios. Deep ensembles consists of M fully connected multi-layer probabilistic networks that are used as base learners and whose last layer has a Gaussian distribution. A negative log-likelihood loss is used to train the base networks, and the model relies on differences in the random initialization of these to learn about the model uncertainty. See Figure 2 for details about the architecture.

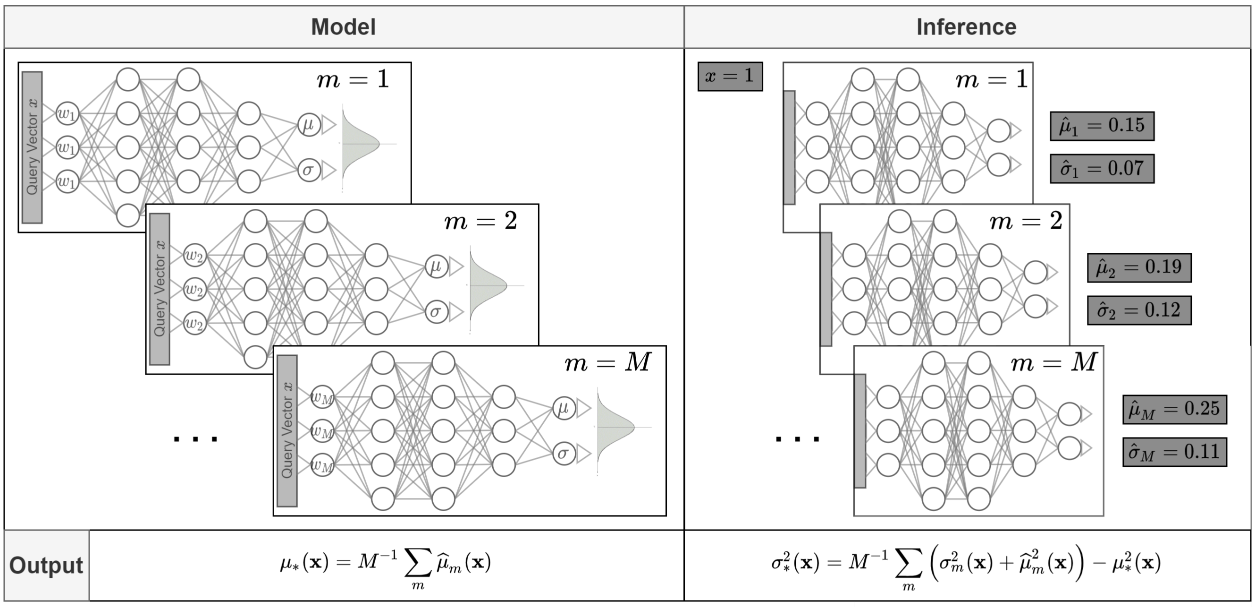


Figure 2: Deep Ensembles Architecture

Deep ensembles capture model uncertainty by considering differences in prediction across base learners.

Despite their proven ability to quantify model uncertainty, employing neural network ensembles incurs high computational overheads. Additionally, their usefulness may be limited when used with acquisition functions that exploit sampling of the surrogate model, as in principle the number of samples at any given x is upper bounded by the number of base models.

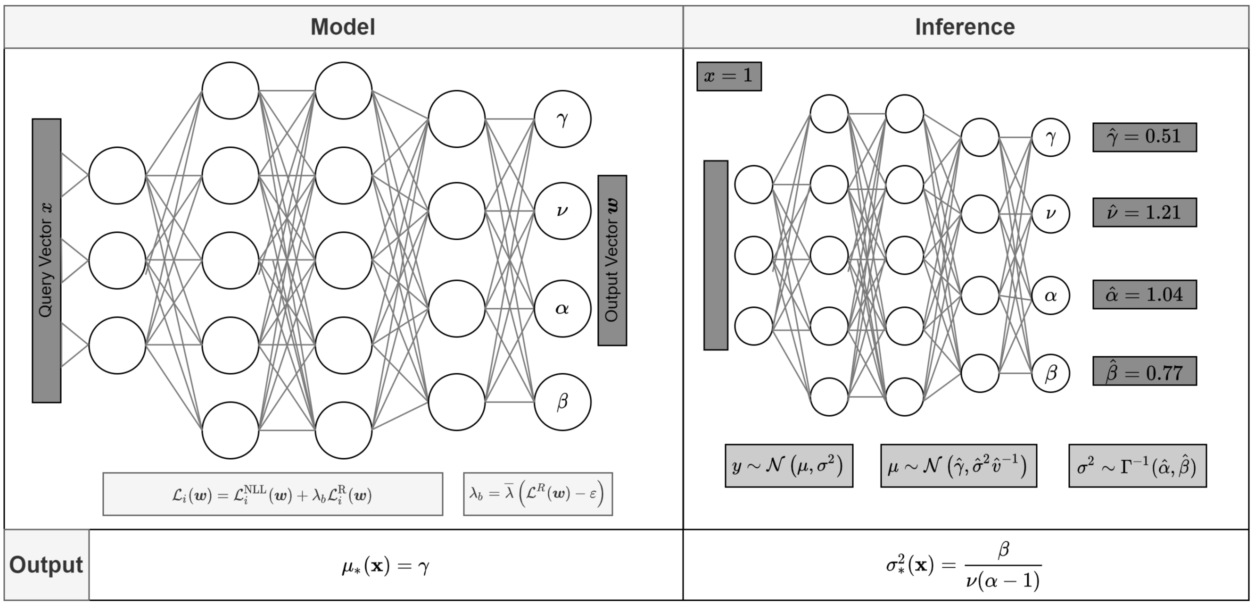
IV.C Deep Evidential Regression

Deep Evidential Regression (DER) is a deterministic method that trains non-Bayesian neural networks to learn the parameters of a higher order evidential regressions. This permits the model to directly estimate the target, the epistemic uncertainty, and the aleatoric uncertainty (Amini et al., 2020). Unlike MC Dropout and Deep Ensembles, DER can analytically disentangle the reducible (epistemic) uncertainty from the irreducible (aleatoric). By assuming a Gaussian likelihood function on the data and evidential priors, a Gaussian prior is placed on the mean and an Inverse Gamma prior is placed on the variance (equation 5). By further assuming that the approximating distributions can be factorized: $q(\mu, \sigma^2) = q(\mu)q(\sigma^2)$, we get the so-called Normal Inverse-Gamma (NIG) distribution.

$$(y_1, y_2, \ldots, y_N) \sim \mathcal{N}(\mu, \sigma^2) \tag{5}$$

$$\mu \sim \mathcal{N}(\gamma, \sigma^2\nu^{-1}) \qquad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta) \tag{6}$$

The neural network is then trained to output the four evidential parameters γ, ν, α, and β of the NIG distribution (Figure 3). The prediction of the NIG distribution is simply $\mathbf{E}(\mu) = \gamma$; the epistemic uncertainty is $var(\mu) = \frac{\beta}{\nu(\alpha-1)}$; and the aleatoric uncertainty is $\mathbf{E}(\sigma^2) = \frac{\beta}{(\alpha-1)}$.



Model
Inference
Query Vector $\boldsymbol{x}$
Output Vector $\boldsymbol{w}$
γ
ν
α
β
$\mathcal{L}_i(\boldsymbol{w}) = \mathcal{L}_i^{\mathrm{NLL}}(\boldsymbol{w}) + \lambda_b \mathcal{L}_i^{\mathrm{R}}(\boldsymbol{w})$
$\lambda_b = \bar{\lambda}\left(\mathcal{L}^R(\boldsymbol{w}) - \varepsilon\right)$
$x = 1$
$\hat{\gamma} = 0.51$
$\hat{\nu} = 1.21$
$\hat{\alpha} = 1.04$
$\hat{\beta} = 0.77$
$y \sim \mathcal{N}\left(\mu, \sigma^2\right)$
$\mu \sim \mathcal{N}\left(\hat{\gamma}, \hat{\sigma}^2 \hat{v}^{-1}\right)$
$\sigma^2 \sim \Gamma^{-1}(\hat{\alpha}, \hat{\beta})$
Output
$\mu_*(\mathbf{x}) = \gamma$
$\sigma_*^2(\mathbf{x}) = \frac{\beta}{\nu(\alpha - 1)}$

Figure 3: Deep Evidential Regression Architecture

Deep Evidential Regression trains the model to produce a 4-dimensional output for every observation. At inference time, these are used to characterize completely the Normal Inverse Gamma distribution.

Although the prior distribution permits an analytical solution giving the Student t-distribution, $St(y; \gamma, \frac{\beta(1+\nu)}{\nu\alpha}, 2\alpha)$, training solely on the negative log likelihood of the Student's t-distribution is ill-defined and a regularizer to penalize model evidence on incorrect predictions is required. For this, Amini et al. (2020) formulates a novel evidence regularizer (7) to be used in conjunction with the negative log likelihood of the Student's t-distribution (8) as the complete loss function of the neural network (9).[3]

$$\mathcal{L}_i^R(\mathbf{w}) = |y_i - \gamma| \cdot (2\nu + \alpha) \tag{7}$$

$$\mathcal{L}_i^{NLL}(\mathbf{w}) = \frac{1}{2} log\left(\frac{\pi}{\nu}\right) - \alpha \log(\Omega) + \left(\alpha + \frac{1}{2}\right) \log\left((y_i - \gamma)^2 \nu + \Omega\right) + \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha + \frac{1}{2})}\right) \tag{8}$$

$$\mathcal{L}_i(\mathbf{w}) = \mathcal{L}_i^{NLL}(\mathbf{w}) + \lambda \mathcal{L}_i^R(\mathbf{w}) \tag{9}$$

where $\Omega = 2\beta(1 + \nu)$ and λ is a hyperparameter that adjusts the weight attributed to the loss regularizer. A large value of λ can cause uncertainty to explode, whereas a small value of λ can lead to overconfidence in the model. In practice, to help with tuning this hyperparameter, we follow the work of Amini et al. (2020) and implement an iterative procedure to adjust the λ parameter throughout training.[4] Constraints are placed on ν, α, and β to be strictly positive (with $\alpha > 1$) to ensure that the distributions and variances are well behaved.

Since the deep evidential framework amounts to a simple feed-forward network, it is the computationally cheapest of the neural networks we consider. Although, Amini et al. (2020) has shown that it is capable of representing uncertainty well, in practice, we find that its representations of uncertainty is not particularly robust to initialisations and highly sensitive to its many hyperparameters. But, through experimentation, we find that taking a logarithmic transform of the epistemic uncertainty for use with either expected improvement or Thompson sampling produced better and

[3]$\mathbf{w}$ are the weights of the neural network.

[4]https://github.com/aamini/evidential-deep-learning

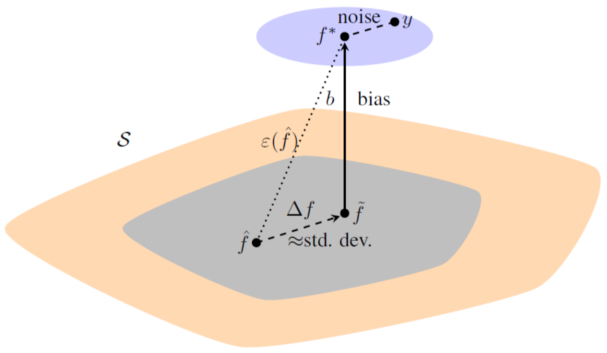
more reliable performance in the context of Bayesian optimisation.

IV.D Direct Epistemic Uncertainty Prediction

The algorithms described above exploit model variance to approximate epistemic uncertainty, but are agnostic about the additional error bias that may stem from parsimonious network architectures and mechanisms used to prevent over-fitting. In light of this, Jain et al. (2021) propose an alternative measure of uncertainty, which is estimated in terms of the loss of the learner as a decomposition into reducible and irreducible uncertainty. Considering a learning algorithm $\mathbf{L}$ mapping a dataset $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ to a predictive function $\hat{f} = \mathbf{L}(\mathcal{D}_n)$ which tries to minimize the expected value of a loss $\mathcal{L}(f(x), y)$, the authors express total uncertainty as in Equation 10.

$$\mathcal{U}(\hat{f}, x) = \int \mathcal{L}(\hat{f}(x), y) dP(y \mid x) = \mathcal{E}(\hat{f}, x) + \mathcal{A}(x) = \mathcal{E}(\hat{f}, x) + \mathcal{U}(f^*, x) \tag{10}$$

where $\mathcal{E}(\hat{f}, x)$ and $\mathcal{A}(x)$ represent epistemic and aleatoric uncertainty. Given some generalization error caused by the choice of network parameters, the best one may aspire to estimate is $\tilde{f}$, the orthogonal projection of $\hat{f}$ on f^* in the subspace of the network. Previous methods explore the model variance around $\hat{f}$ to estimate the reducible error, whereas DEUP additionally accounts for the generalization error, or distance between $\hat{f}^*$ on f^*, when estimating epistemic uncertainty.



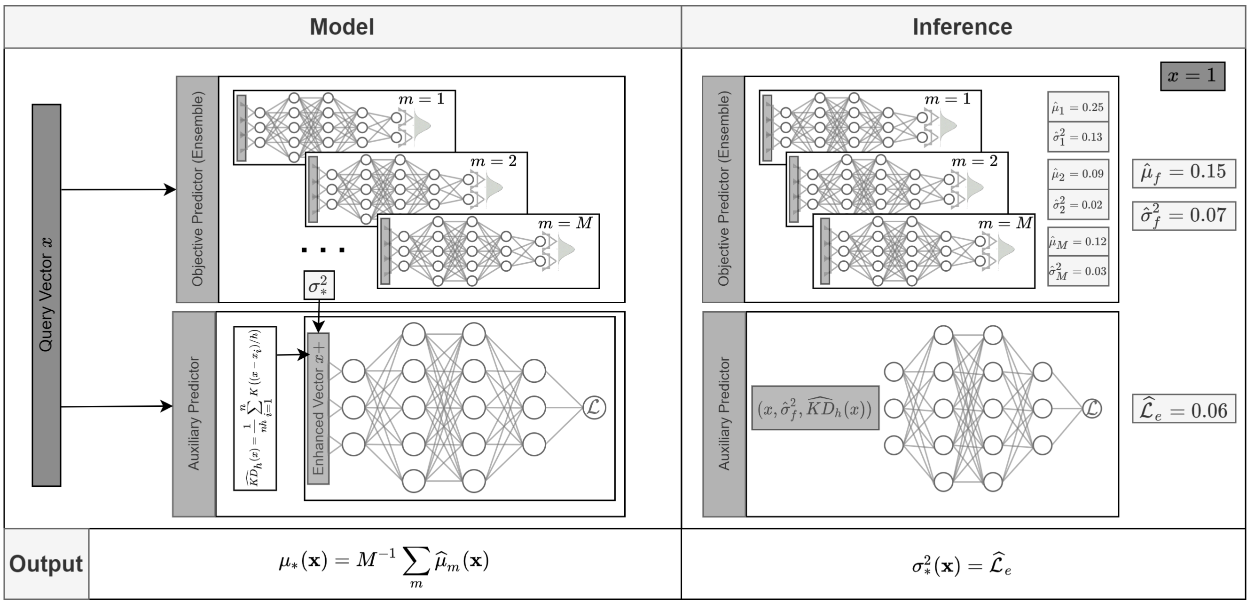
Direct Epistemic Uncertainty Prediction (DEUP) employs two predictors to approximate $\mathcal{E}(\hat{f}, x)$: a main predictor $\hat{f}$ that learns about the original task, and an auxiliary error predictor u which is used to predict the epistemic uncertainty of the main predictor. The core principle of DEUP is to use observed out-of-sample errors in order to train u, which can be used to estimate epistemic uncertainty elsewhere. The error predictor u is trained using the original coordinates of x as well as additional features to predict the mean squared error loss values of the main predictor. These auxiliary variables aim to provide contextual information about the input space and are themselves proxies for epistemic uncertainty. We use both kernel density estimates (Charpentier et al., 2020) and the main predictor variance (Gal and Ghahramani, 2016) at each point x. These variables address non-stationarity concerns of the error predictor targets, which arise from the changing behavior of uncertainty around some x point as the model explores the neighboring space.

Additionally, we give the error predictor u a warm start by creating a synthetic dataset of cross-validated observations, which improves early prediction of the epistemic uncertainty in the model. We use cross-validation to train the main predictor $\hat{f}$, and append the loss values to the synthetic dataset. See Algorithm 2 for the complete pseudo-code of the algorithm.

Algorithm 2 Training procedure for DEUP in a BO setting

```
1:  data: $\mathcal{D}_{init}$ initial dataset with pairs $(x, y) \in \mathcal{X} \times \mathbf{R}$
2:  $\hat{f} : \mathcal{X} \to \mathbf{R}$, main predictor (of $y$ given $x$)
3:  $u : \mathcal{X} \to \mathbf{R}$, total uncertainty estimator
4:  $\phi : \mathcal{X} \to \mathbf{R}^k$, auxiliary stationarizing features
5:  $acq$: acquisition function
6:  $x_{acq}$ from $\mathcal{X}$, using the current $\hat{f}$ and $u$ estimates
7:  $\mathcal{D}_u \leftarrow \emptyset$, training dataset for $u$
8:  $\mathcal{D} \leftarrow \mathcal{D}_{init}$, dataset of training points for $\hat{y}$ seen so far
9:  optional: pre-fill $\mathcal{D}_u$ using shuffled sets of initial points and fits of $u$
10: $x_{acq} \leftarrow \emptyset, \quad y_{acq} \leftarrow \emptyset$
11: while $n = 1, 2, \ldots$ do
12:     fit predictor $\hat{f}$ and features $\phi$ on $\mathcal{D}$
13:     $\mathcal{D}_u \leftarrow \mathcal{D}_u \cup \left\{\phi(x_{acq}), \left(y_{acq} - \hat{f}(x_{acq})\right)^2\right\}$ if $x_{acq} \neq \emptyset$
14:     fit auxiliary predictor $u$ on $\mathcal{D}_u$
15:     $x_{acq} \leftarrow acq(\mathcal{X}, \hat{f}, u), \quad y_{acq} \sim P(\cdot \mid x_{acq})$
16:     $\mathcal{D}_u \leftarrow \mathcal{D}_u \cup \left\{\phi(x_{acq}), \left(y_{acq} - \hat{f}(x_{acq})\right)^2\right\}$
17:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_{acq}, y_{acq}\}$
```

Although any probabilistic model can be used as a main predictor, we consider Deep Ensembles in the experiments below because it is known to have good uncertainty quantification. The auxiliary predictor u is a standard feed-forward network, and Gaussian density estimates are computed in each iteration following a grid search of the optimal bandwidth parameter. This optimization schedule makes DEUP the most computationally expensive models among the ones considered in this article, and the visual description of this specification is shown in Figure 4.



Model
Inference
Query Vector $\boldsymbol{x}$
Objective Predictor (Ensemble)
$m = 1$
$m = 2$
$m = M$
σ^2_*
Auxiliary Predictor
$\widehat{KD}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K((x - x_i)/h)$
Enhanced Vector $x+$
$\mathcal{L}$
$\boldsymbol{x} = 1$
$\hat{\mu}_1 = 0.25$
$\hat{\sigma}^2_1 = 0.13$
$\hat{\mu}_2 = 0.09$
$\hat{\sigma}^2_2 = 0.02$
$\hat{\mu}_M = 0.12$
$\hat{\sigma}^2_M = 0.03$
$\hat{\mu}_f = 0.15$
$\hat{\sigma}^2_f = 0.07$
$(x, \hat{\sigma}^2_f, \widehat{KD}_h(x))$
$\widehat{\mathcal{L}}_e = 0.06$
Output
$\mu_*(\mathbf{x}) = M^{-1}\sum_m \widehat{\mu}_m(\mathbf{x})$
$\sigma^2_*(\mathbf{x}) = \widehat{\mathcal{L}}_e$

Figure 4: Direct Epistemic Uncertainty Prediction Architecture

Direct Epistemic Uncertainty Prediction trains two separate models to estimate uncertainty in an auxiliary model as the reducible loss of the main model. The uncertainty model is trained with observations that are included before and after retraining the main model, and additional variables are used to account for the resulting non-stationary behavior.

V Experiments

To test the scalability of these novel models, we incorporate Gaussian noise into the observations of already difficult functions: Shekel-4, Ackley-5, and Hartmann-6. The Shekel and Ackley functions are highly non-stationary whereas the Hartmann, although stationary, is a higher dimension function that is difficult to optimize. Hence, a large number of function evaluations is required to achieve low final regret. In order to reduce the computational overhead of BO in this large evaluation budget setting we use large batches, leveraging the highly parallelizable TS. We run a BO schedule of 50 steps using batches of 100 samples with discrete Thompson sampling for all of the models. We compare the performance of neural networks against GP's and SVGP's using minimum regret.

We also evaluate these models' abilities to handle non-stationary functions with small evaluation budgets using Branin-2, Michalwicz-2, Dropwave-2, Eggholder-2, Goldstein-Price-2, Hartmann-3, Rosenbrock-4, Shekel-4, Ackley-5, and Hartmann-6 functions (Figure 5). These classical BO synthetic functions have varying degrees of non-stationarity and often exhibit many local optima which make them challenging for GP-based BO (Adorio and Diliman 2005, Surjanovic and Bingham 2013). We compare the models to random sampling and Gaussian process baselines by running BO schedules for 500 steps. We allow for early stopping when the minimum is found.

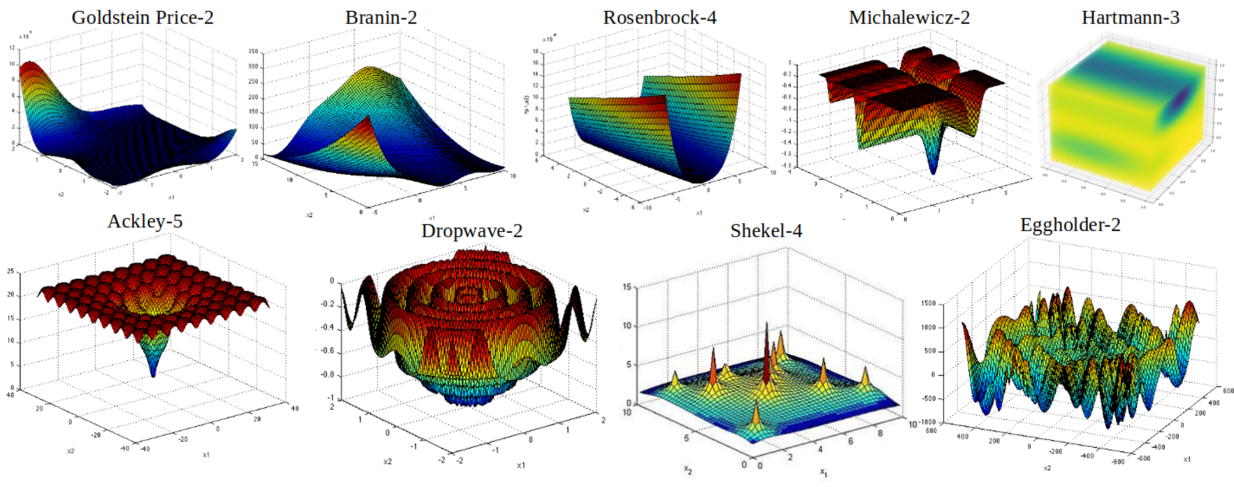


Figure 5: Synthetic objective functions approximated in 3D

We manually tune the hyperparameters of the networks for these functions, but additional fine-tuning could further improve their performance. Throughout all tests we maintain the same network architectures for all models, the details of which are included in Appendix Section VII.F. All simulations are averaged over at least ten randomly selected seeds. In the figures and tables below, minimum regret is standardized by the output range of the functions to range between zero and one. Confidence bands are represented in the figures by the shaded regions.

V.A Large Evaluation Budgets

To increase the difficulty of Shekel-4, Ackley-5, and Hartmann-6 we add observational Gaussian noise proportional to the output ranges of these functions, resulting in variance of 0.6 for each of these functions. This gives them a similar degree of difficulty to those used in Vakili et al. (2021). We compare our models with SVGP using 500 inducing points following the advice of Vakili et al. (2021). The results are presented in Figure A3 and Table 1.

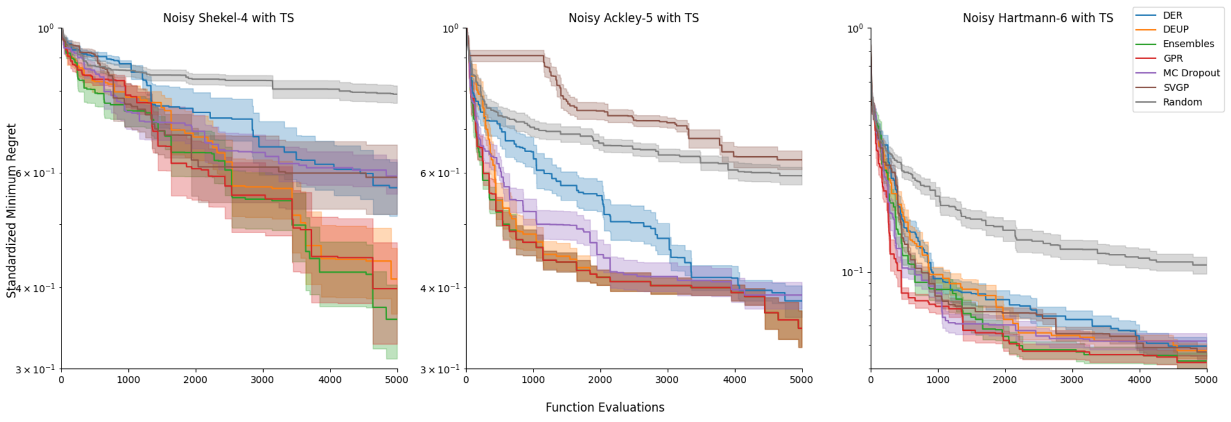


Figure 6: Comparison of Minimum Regret in Noisy Functions

Performance of neural networks is mostly comparable with GPR and superior to SVGP in the case of Shekel-4 and Ackley-5.

Figure A3 shows that GP maintains good performance and finds high quality solutions on all of the noisy functions. Nonetheless, we find that Deep ensembles and DEUP continue to perform comparably, and in fact, Deep Ensembles finds a better final regret in the noisy Shekel-4 setting. Moreover, DER and MC Dropout are also able to achieve comparable performance to GP in the noisy Ackley and Hartmann settings. On the other hand, SVGP maintains good performance with the stationary Hartmann function, but is outperformed in both the noisy Shekel and noisy Ackley settings. In particular, it performs worse than random sampling for the Ackley function, showcasing its lack of expressivity and its limitations with these non-stationary functions.

Table 1: *Minimum Regret on Noisy Functions*

	Average Minimum Regret						
Acquisition	Thompson Sampling						Random
Model	DE	DER	DEUP	MC	SVGP	*GPR*	
Shekel-4	**0.36**	0.57	0.41	0.59	0.59	*0.40*	0.95
Ackley-5	**0.35**	0.38	**0.35**	0.39	0.63	*0.35*	0.85
Hartmann-6	**0.04**	0.05	0.05	0.05	0.05	*0.04*	0.38

Deep Ensembles provides comparable performance to GPR, which are both superior to SVGP across the board. The neural network model with the lowest achieved regret for each target function is highlighted in bold.

Although GP's are able to find good solutions to the noisy functions, average runtimes showcase expectedly cubic behaviour, as seen in Figure 7. Conversely, the computational cost of neural networks is shown to increase only linearly; MC Dropout and DER scale particularly well. SVGP was far superior to the rest in terms of computational speed and scalability, in line with results in Vakili et al. (2021). Nonetheless, in all of these scenarios GP is more computationally expensive than the neural network models with the exception of DEUP, but its cubic cost suggests it will also surpass DEUP in situations with larger evaluation budgets.

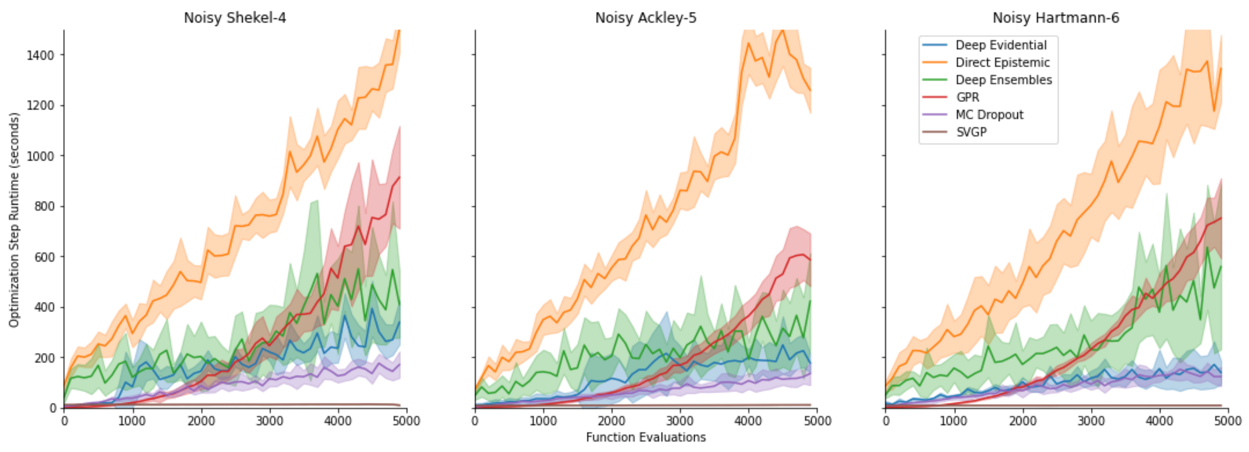


Figure 7: Comparison of Optimisation time

With the exception of DEUP, after 40 BO steps, GPR surpasses neural networks in optimisation time and continues to scale cubicly while neural networks scale linearly.

Table 2: *Total Optimization Runtime on Noisy Functions*

	Average Bayesian Optimization Runtime (Minutes)					
Acquisition	Thompson Sampling					
Model	DE	DER	DEUP	MC	*SVGP*	*GPR*
Shekel-4	221.9	143.1	586.9	**74.8**	*10.7*	*221.0*
Ackley-5	176.5	102.3	610.5	**53.7**	*8.2*	*157.2*
Hartmann-6	215.1	77.8	570.6	**66.9**	*7.7*	*189.5*

Both DER and MC Dropout have much faster total runtimes than GPR. The fastest neural network model per target function is highilighted in bold.

V.B Small Evaluation Budgets

As expected, GP's showcase a superior performance in a majority of the functions (Table 3). With the exception of Dropwave-2, Shekel-4, and Ackley-5, GP is able to achieve minimum final regret reasonably fast. Even when neural networks perform comparably, they tend to have slower runtimes, as shown in table 4. Notable instances were neural networks appear to struggle with finding the true minimum where GP did not are Hartmann-6 and Eggholder-2. Here we report the most salient results, the rest of the results can be found in Appendix Section VII.C.

Table 3: *Minimum Regret on Noise-Free Functions*

	Average Minimum Regret										
Acquisition	Expected Improvement					Thompson Sampling					Random
Model	DE	DER	DEUP	MC	*GPR*	DE	DER	DEUP	MC	*GPR*	
Log Goldstein-Price-2	**(48)**	0.10	(89)	0.07	*(10)*	**(26)**	(39)	(66)	(80)	*(22)*	0.02
Scaled Branin-2	**(58)**	(128)	(141)	(87)	*(17)*	**(11)**	(36)	(41)	(28)	*(27)*	0.01
Michalwicz-2	**(116)**	0.14	0.14	(129)	*(25)*	**(54)**	0.03	(123)	(68)	*(23)*	0.02
Dropwave-2	**0.01**	0.17	0.06	0.05	*0.19*	**0.04**	0.06	**0.04**	0.06	*0.06*	0.06
Eggholder-2	0.06	0.07	**0.03**	0.07	*0.02*	**0.01**	**0.01**	**0.01**	0.02	*0.02*	0.02
Hartmann-3	**(170)**	0.01	0.01	0.03	*(46)*	**(78)**	(127)	(315)	(232)	*(81)*	0.01
Rosenbrock-4	**(135)**	(396)	0.01	0.01	*(36)*	**(318)**	(428)	0.01	(433)	*(444)*	0.01
Shekel-4	**0.08**	0.70	0.68	0.66	*0.11*	0.26	0.34	**0.25**	0.27	*0.26*	0.83
Ackley-5	0.14	0.19	0.19	**0.11**	*0.19*	0.28	0.28	0.28	0.28	*0.27*	0.65
Hartmann-6	**0.01**	0.04	0.02	0.03	*(150)*	0.03	0.04	**0.02**	0.03	*0.03*	0.14

Deep Ensembles and MC Dropout achieve somewhat comparable performance to GPR, albeit often slower. The table reports the average performance of each model under different objective-acquisition function pairings. Reported results correspond to the minimum regret after 500 function evaluations, standardized by the output range of the function. For instances where a model find the minimum in less than 500 evaluations, we report the number of steps taken in parentheses instead. Best performing neural network models per target and acquisition function are reported in bold.

Otherwise, neural networks achieve similar final regret to GP-based BO, although often requiring more steps. In particular, Deep Ensembles and DEUP performed comparatively well, although the latter is orders of magnitude more computationally expensive. MC Dropout's performance was often similar to Deep Ensembles, while DER's was not consistent.

Table 4: *Total Optimization Runtime on Noise-Free Functions*

	Average Bayesian Optimization Runtime (Minutes)									
Acquisition	Expected Improvement					Thompson Sampling				
Model	DE	DER	DEUP	MC	*GPR*	DE	DER	DEUP	MC	*GPR*
Log Goldstein-Price-2	**4.1**	43.9	20.9	19.9	*0.2*	**4.1**	5.6	33.8	11.9	*0.5*
Scaled Branin-2	**4.3**	21.0	48.0	5.5	*0.4*	**0.9**	6.5	21.8	1.7	*0.6*
Michalwicz-2	12.6	42.0	158.0	**9.0**	*0.5*	10.6	40.0	98.9	**9.0**	*0.5*
Dropwave-2	152.2	76.6	296.6	**76.1**	*19.8*	526.2	296.6	935.8	**263.1**	*90.3*
Eggholder-2	104.0	**57.9**	222.3	69.0	*7.0*	482.6	421.3	1066.3	**299.6**	*76.2*
Hartmann-3	**22.5**	33.3	208.9	49.1	*1.1*	**21.7**	35.9	674.8	81.4	*2.6*
Rosenbrock-4	**22.2**	60.3	369.7	84.0	*0.7*	288.1	**231.4**	1428.6	232.6	*85.3*
Shekel-4	189.6	93.1	373.3	**73.4**	*21.3*	603.4	485.3	1075.7	**291.1**	*124.9*
Ackley-5	209.7	125.9	582.2	**92.2**	*12.3*	538.4	380.2	1470.9	**220.0**	*182.8*
Hartmann-6	214.8	**75.9**	351.9	95.8	*4.9*	619.8	523.4	1201.6	**292.3**	*108.5*

Although MC Dropout is the fastest neural network per step, a strong model such as Deep Ensembles will stop early when the minimum is found reducing its total runtime. Nonetheless, GPR is still the fastest model across the board in this small data regime. The fastest neural network for each target and acquisition function pairing is highlighted in bold.

Considering the noiseless versions of the Shekel, Ackley, and Hartmann functions, GP's underperform with Ackley and Shekel relative to Deep Ensembles. In the Ackley case, all neural networks achieve similar or better performance than GP's. Here, MC Dropout actually outperforms Deep Ensembles. Figure 8 plots the regret curves.

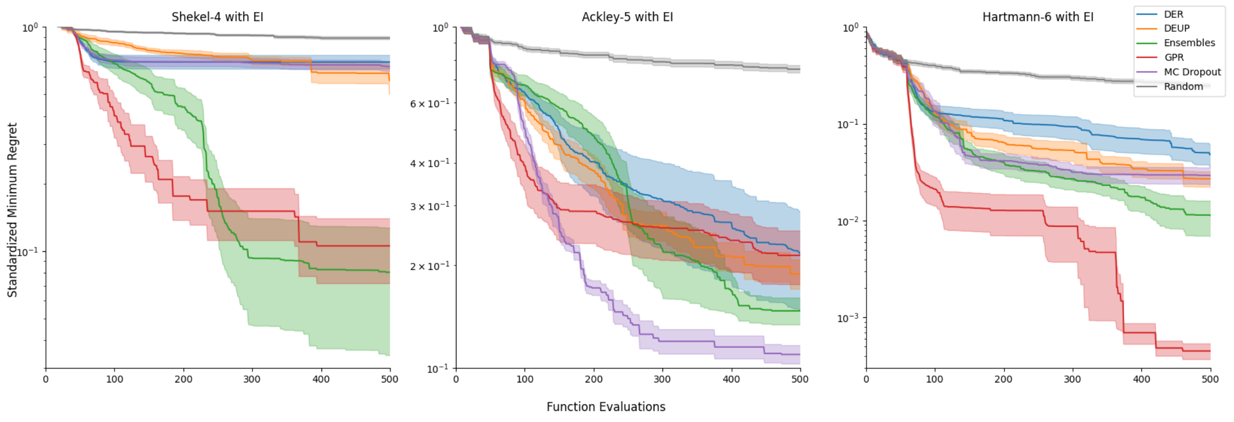


Figure 8: Most Challenging Synthetic Functions

GP is outperformed by neural networks, particularly Deep Ensembles in the Shekel and Ackley functions.

Interestingly, Dropwave was one of the few instances where GP performed particularly poorly: even random sampling was able to produce better minima. The evolution of minimum regret across models for Dropwave is plotted in Figure 9. This lackluster performance reflects the fact that the function is highly non-stationary. MC Dropout, DEUP, and particularly Deep Ensembles are able to produce better minima under both acquisition function regimes.

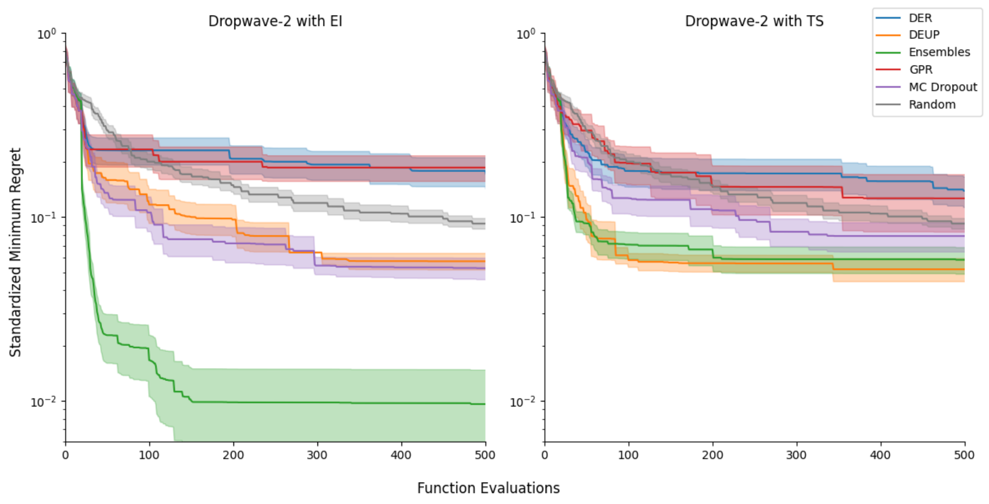


Figure 9: Minimum Regret for Dropwave-2

GPR performs exceptionally poorly (worse than random) with the highly non-stationary Dropwave-2 function.

The empirical results suggest that neural networks work better in conjunction with Thompson sampling for most functions. While Dropwave is an exception to this observation, examples that especially align with the observation above include Hartmann-3 and Goldstein-Price (Figure 10). These are revealing examples where switching acquisition functions yields significantly improved results.

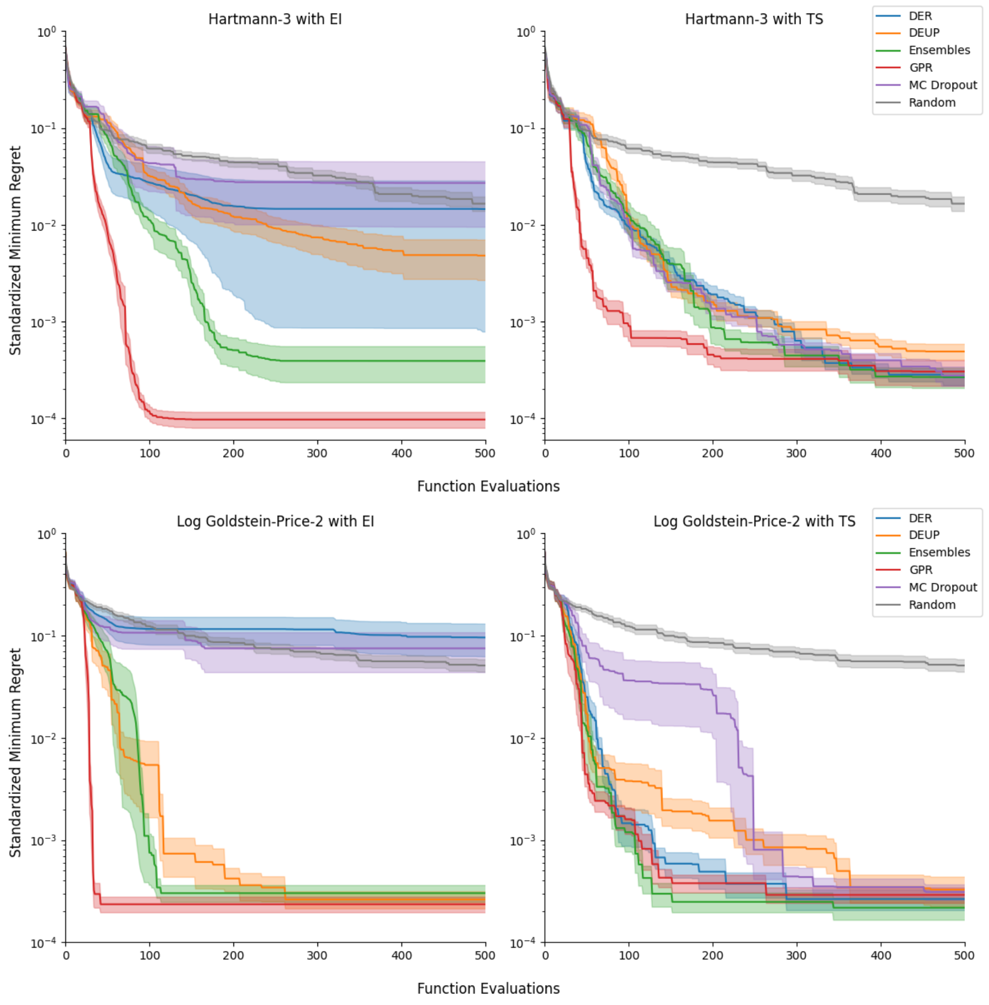


Figure 10: Comparison of Expected Improvement and Thompson Sampling

Thompson sampling often drastically improves results for neural networks over the use of Expected Improvement.

A possible reason for this behavior is that Expected Improvement can over-exploit (Berk et al., 2019), whereas Thompson Sampling has a natural mechanism to induce exploration. Coupled with the fact that neural networks are highly prone to over-fitting and predicting with excessive confidence, a potential conjecture is that the models evaluated are best used in tandem with acquisition functions that lean towards exploration.

V.C Predictive Accuracy and Uncertainty Quantification

Without specifically testing within Bayesian optimisation, a common approach to judge the potential effectiveness of a surrogate model is to consider its representation of uncertainty. Osband et al. (2021) has shown that Deep Ensembles have better uncertainty quantification than MC Dropout, and more recently developed models such as DEUP and DER will likely have varying degrees of performance. To tractably investigate this further, we take a Sobol sample of 100,000 points in the search space at the termination of each experiment and compute the negative log probability density assuming a Gaussian distribution. We take the Gaussian kernel-weighted sum of these points to the nearest true minimum to account for the fact that areas around the true minimum are most important.

Table 5: *Correlation between prediction and optimization*

	Correlations between NLPD and Minimum Regret					
Model	**All**	DE	DEUP	DER	MC	GPR
NLPD$_{samples}$	**-0.029**	-0.084	0.021	-0.029	-0.057	-0.073
NLPD$_{min}$	**0.013**	-0.070	0.029	0.026	-0.042	-0.038

Whether considering NLPD at the minimum or weighted samples across the search space, there is no apparent correlation between NLPD and final minimum regret achieved. NLPD is computed assuming a Gaussian distribution and weighted by distance to the true minimum by a Gaussian kernel.

We see in Table 5, that there is no apparent correlation to final regret. Although this method of quantifying overall accuracy is a simple heuristic, we would have hoped to see at least some weak correlations here. In light of this, it would seem that looking at uncertainty quantification and predictive accuracy outside of the context of BO as a proxy for BO performance might not be as informative as hoped for.

VI Conclusion

We have shown that neural networks can provide comparable performance to standard GP-based models when objective functions are complex, and a large number of function evaluations are required. The advantage of neural networks in this setting is that they are more scalable and tractable in the long run than GP's. Clearly, with truly large evaluation budgets, GP's would quickly become infeasible. In this scenario, the current state of the art tool is SVGP, but although SVGP's can provide superior scalability by leveraging approximation, it comes at a high cost of expressivity. We have shown that neural network based BO has the potential to maintain a high degree of expressivity and performance whilst staying computationally tractable with large evaluation budgets.

Neural networks are clearly a viable alternative to traditional GP-based routines or state-of-the-art SVGP's.

Our experiments in smaller data regimes suggest that neural networks have the potential for comparable performance to the gold standard of GP-based BO if hyperparameters are sufficiently tuned. Techniques that have been proposed such as using a combination of regularization techniques (Kadra et al., 2021) or Bayesian optimisation itself over architecture hyperparameters (Kandasamy et al., 2018), and whether their computational overhead would be worthwhile in a Bayesian optimisation context, would be venues for future research.

Furthermore, since performance in Bayesian optimisation cannot simply be inferred or extrapolated, future work would also include testing these models against SVGP in significantly larger data regimes. Additional work would also explore the use of a continuous Thompson sampler with large batch trajectory sampling, which would likely provide significant computational advantages with neural network based Bayesian optimisation (Zhang et al., 2019).

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.

Adorio, E. P. and Diliman, U. P. (2005). Multivariate test functions library in c for unconstrained global optimization.

A.G., Z. (1975). Single-step bayesian search method for an extremum of functions of a single variable. *Cybernetics*, 11:160–166.

Amini, A., Schwarting, W., Soleimany, A., and Rus, D. (2020). Deep evidential regression. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14927–14937. Curran Associates, Inc.

Azimi, J., Jalali, A., and Fern, X. (2012). Hybrid batch bayesian optimization. arXiv.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305.

Berk, J., Nguyen, V., Gupta, S., Rana, S., and Venkatesh, S. (2019). Exploration enhanced expected improvement for bayesian optimization. In Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., and Ifrim, G., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 621–637, Cham. Springer International Publishing.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 161–168. NIPS Foundation (http://books.nips.cc).

Brochu, E., Cora, V. M., and de Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.

Calvin, J. and Zilinskas, A. (2000). One-dimensional p-algorithm with convergence rate o(n3+) for smooth functions. *Journal of Optimization Theory and Applications*, 106:297–307.

Charpentier, B., Zügner, D., and Günnemann, S. (2020). Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1356–1367. Curran Associates, Inc.

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In Carvalho, C. M. and Ravikumar, P., editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA. PMLR.

Frazier, P. I. (2018). A tutorial on bayesian optimization.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the*

33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org.

Garnett, R. (2022). *Bayesian Optimization*. Cambridge University Press. in preparation.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A. M., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. (2021). A survey of uncertainty in deep neural networks. *CoRR*, abs/2107.03342.

Greenhill, S., Rana, S., Gupta, S., Vellanki, P., and Venkatesh, S. (2020). Bayesian optimization for adaptive experimental design: A review. *IEEE Access*, 8:13937–13948.

Griffiths, R.-R. and Hernández-Lobato, J. M. (2017). Constrained bayesian optimization for automatic chemical design.

Gustafsson, F. K., Danelljan, M., and Schön, T. B. (2020). Evaluating scalable bayesian deep learning methods for robust computer vision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1289–1298.

Hebbal, A., Brevault, L., Balesdent, M., Talbi, E.-G., and Melab, N. (2019). Bayesian optimization using deep gaussian processes.

Henne, M., Schwaiger, A., Roscher, K., and Weiß, G. (2020). Benchmarking uncertainty estimation methods for deep learning with safety-related metrics.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data.

Hernandez-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France. PMLR.

Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. (2017). Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1470–1479. PMLR.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello, C. A. C., editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jain, M., Lahlou, S., Nekoei, H., Butoi, V., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. (2021). DEUP: direct epistemic uncertainty prediction. *CoRR*, abs/2102.08501.

Jalali, H., Nieuwenhuyse, I. V., and Picheny, V. (2017). Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. *Eur. J. Oper. Res.*, 261:279–301.

Jones, D. (2001). A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21:345–383.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. (2021). Well-tuned simple nets excel on tabular datasets.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. (2018). Neural architecture search with bayesian optimisation and optimal transport.

Keane, A. J. (2006). Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44(4):879–891.

Kleijnen, J. (2015). *Design and Analysis of Simulation Experiments.* International Series in Operations Research amp; Management Science. Springer Verlag, Germany, 2nd edition.

Knowles, J. (2006). Parego: A hybrid algorithm with on-line landscape approximation for expension multiobjective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 10:50 – 66.

Kushner, H. J. (1964). A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Lizotte, D., Wang, T., Bowling, M., and Schuurmans, D. (2007). Automatic gait optimization with gaussian process regression. pages 944–949.

Mehdad, E. and Kleijnen, J. P. C. (2018). Efficient global optimisation for black-box simulation via sequential intrinsic kriging. *Journal of the Operational Research Society*, 69(11):1725–1737.

Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2.

Mullachery, V., Khera, A., and Husain, A. (2018). Bayesian neural networks.

Neal, R. M. (1995). Bayesian learning for neural networks.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Van Roy, B. (2021). Epistemic neural networks.

Osborne, M., Garnett, R., and Roberts, S. (2009). Gaussian processes for global optimization.

Romano, Y., Patterson, E., and Candes, E. (2019). Conformalized quantile regression. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Salemi, P., Nelson, B. L., and Staum, J. (2014). Discrete optimization via simulation using gaussian markov random fields. In *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, page 3809–3820. IEEE Press.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. (2015). Scalable bayesian optimization using deep neural networks. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France. PMLR.

Sobester, A., Leary, S., and Keane, A. (2004). A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27:371–383.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. 15(1):1929–1958.

Surjanovic, S. and Bingham, D. (2013). Virtual library of simulation experiments: Test functions and datasets.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-task bayesian optimization. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Vakili, S., Picheny, V., and Artemev, A. (2021). Scalable thompson sampling using sparse gaussian process models. In *NeurIPS*.

Williams, C. K. I. (1996). Computing with infinite networks. In *NIPS*.

Zhang, R., Wen, Z., Chen, C., and Carin, L. (2019). Scalable thompson sampling via optimal transport.

VII Appendix

VII.A Exact Gaussian processes

Consider the collection of n input-output pairs $\mathcal{D}_n = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i$ is D dimensional and $\mathbf{y}_i$ is a n-dimensional vector of evaluations on $\mathbf{f}$. Under the assumption of sub-Gaussian distributed disturbances, the following generative model is defined

$$\widehat{\mathbf{f}}_{1:n} \mid x_{1:n} \sim \mathcal{N}(m, K)$$
$$\mathbf{y} \mid \widehat{\mathbf{f}}_{1:n}, \sigma^2 \sim \mathcal{N}\left(\widehat{\mathbf{f}}_{1:n}, \sigma^2 I\right)$$

where the mean and covariance matrix elements are defined as $m_i = \mu_0(\mathbf{x}_i)$ and $K_{i,j} = k(x_i, x_j)$ for some Gaussian kernel function $k(\cdot,\cdot)$. Given the existing history $\mathcal{D}_n$, the posterior of the Gaussian process $\widehat{\mathbf{f}}_{1:n}$ is another Gaussian process with the following mean μ_n and variance σ_n^2 expressions

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + k_n(\mathbf{x}_{1:n}, \mathbf{x})^{\mathbf{T}} \left(k_n(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + \sigma^2 I\right)^{-1} (\mathbf{y} - m)$$
$$\sigma_n^2(\mathbf{x}) = k_n(\mathbf{x}, \mathbf{x}) - k_n(\mathbf{x}_{1:n}, \mathbf{x})^{\mathbf{T}} \left(k_n(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + \sigma^2 I\right)^{-1} k_n(\mathbf{x}_{1:n}, \mathbf{x})$$

where $k_n(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})$ is the covariance matrix of the initial n points and $k_n(\mathbf{x}_{1:n}, \mathbf{x})$ is a vector of covariance terms between a new value $\mathbf{x}$ and $\mathbf{x}_{1:n}$. The posterior means and variances represent the model's prediction and uncertainty in the unknown function at the query point $\mathbf{x}$, which can be used to select the next point in the iteration, $\mathbf{x}_{n+1}$.

Note that accessing the posterior expressions as described above requires an $O\left(n^3\right)$ matrix inversion of kernel and covariance values, rendering this approach intractable when scaling the algorithm to large evaluation budgets.

VII.B Cubic tests

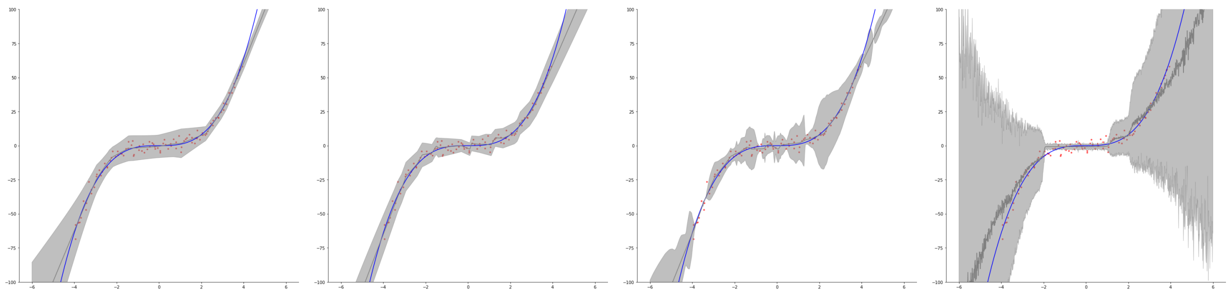

a Deep Ensembles **b** Deep Evidential **c** Direct Epistemic **d** Monte Carlo

Figure A1: Cubic function with out-of-distribution predictions
This figure plots the point predictions and epistemic uncertainty bands under the different neural network models proposed.

In Figure A1, we compare the ability of the different methods considered to predict uncertainty in a toy one dimensional cubic function. The models are all capable of quantifying uncertainty to varying degrees and with different compositions of epistemic and aleatoric noise.

It is worth noting that not all models are equally robust, and in practice some of the neural networks can yield poor representations of uncertainty at times. This does not seem to materialize in the experiments, and we believe this may not necessarily translate to poor performance in Bayesian optimisation.

VII.C Regret Plots

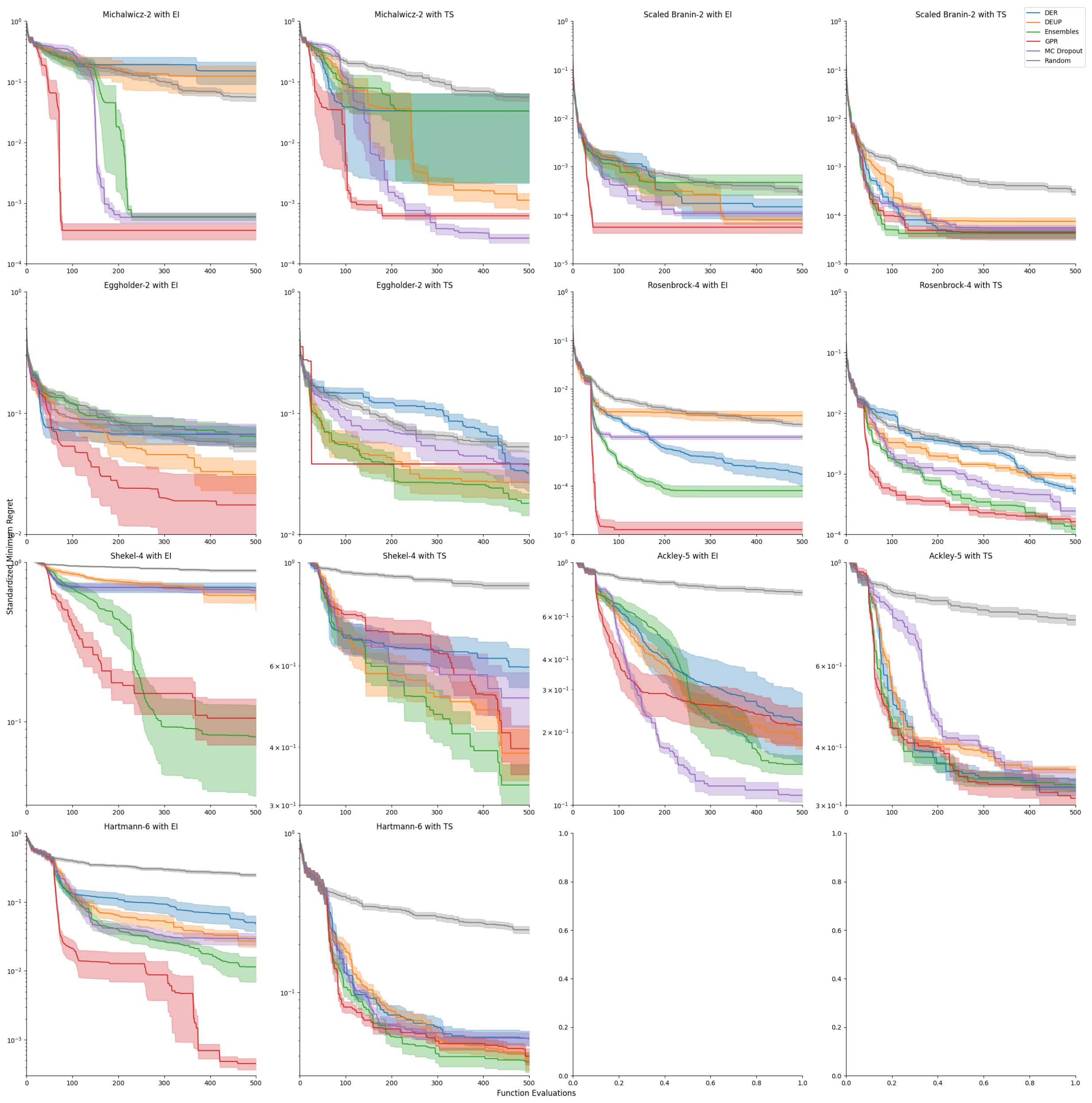


Figure A3: Minimum Regret of Synthetic Functions with Small Evaluation Budgets

VII.D Optimisation Times

The reported optimization times both in Section V and in the Appendix refer to only the time taken to update and optimize the surrogate model, and do not include the time taken by the acquisition function. The previous tables showed the overall runtime across models and for all target-acquisition function pairs. We complement those findings with runtime results per Bayesian Optimization step, showing results below for both noise-free and noisy functions.

Table A1: *Average Step Runtime on Noise-Free Functions*

	Average Runtime per BO step (Minutes)									
Acquisition	Expected Improvement					Thompson Sampling				
Model	DE	DER	DEUP	MC	*GPR*	DE	DER	DEUP	MC	*GPR*
Log Goldstein-Price-2	4.86	6.96	11.02	4.16	*1.03*	7.36	6.87	24.01	6.68	*1.28*
Scaled Branin-2	3.71	5.15	14.73	2.94	*1.23*	4.94	10.24	21.77	2.79	*1.27*
Michalwicz-2	5.68	7.00	24.55	4.03	*1.04*	9.36	10.29	38.19	6.35	*1.24*
Dropwave-2	18.23	9.18	35.52	9.12	*2.37*	63.02	35.52	112.08	31.51	*10.82*
Eggholder-2	13.49	7.50	30.71	8.67	*1.39*	57.80	50.45	127.70	36.88	*9.12*
Hartmann-3	7.57	7.11	31.72	6.98	*1.38*	13.81	14.32	99.59	14.81	*1.77*
Rosenbrock-4	8.78	8.56	44.28	10.06	*1.10*	45.83	30.90	171.10	29.70	*10.56*
Shekel-4	22.70	11.46	44.70	8.79	*2.55*	72.27	58.12	128.82	34.87	*14.96*
Ackley-5	25.11	15.08	69.73	11.04	*1.47*	64.47	45.53	176.15	26.34	*21.89*
Hartmann-6	25.72	10.21	43.35	11.47	*1.58*	74.23	62.68	143.91	35.01	*12.99*

Note: The table shows the mean per step runtime of the Bayesian optimization schedules under different models and for different acquisition and target function combinations, computed in minutes.

Table A2: *Average Step Runtime on Noisy Functions*

	Average Runtime per BO step (Minutes)					
Acquisition	Thompson Sampling					
Model	DE	DER	DEUP	MC	SVGP	*GPR*
Shekel-4	221.9	143.1	586.9	74.8	10.7	*221.0*
Ackley-5	176.5	102.3	610.5	53.7	8.2	*157.2*
Hartmann-6	215.1	77.8	570.6	66.9	7.7	*189.5*

Note: The table shows the mean per-step runtime of the Bayesian optimization schedules under different models and for different noisy target functions, computed in minutes.

VII.E Implementation Details

All neural networks are implemented by us to be used within the framework of trieste's Bayesian optimization package with the exception of Deep Ensembles which was implemented by Hrvoje Stojic. Trieste's implementations of GP-based Bayesian optimisation, Sparse Variational Gaussian Processes, and all other Bayesian optimisation related routines were used otherwise. For the source code to our contributions see github.com/clinton0313/trieste.

VII.F Experiment Details

Experiments were run on a Linux cluster server, with model-specific simulations running in parallel on 16 CPU nodes of 16 GB RAM each. Simulations were CUDA accelerated using NVIDIA Quadro RTX 4000 GPUs. We manually tune the hyperparameters of the neural networks after performing a grid search of candidate combinations tested on small data functions. Final parameter choices are defined as in Table , with parameter choices defined as in Table A3.

Table A3: *Architecture parameters*

Model	DE	DER	DEUP	MC
Number of Hidden Layers	5	4	5 (5)	5
Number of Units	50	100	50 (128)	100
Learning Rate	0.001	0.001	0.001	0.001
Ensemble Size	7	-	7	-
Regulation Weight	-	0.001	-	-
Dropout Rate	-	-	-	0.1
Number passes	-	-	-	50

The plot shows cubic functions and both in-distribution and out-of-distribution predictions under the different neural networks considered in the model. The network parameters do not necessarily correspond to the ones used in the experiments.

In the case of GP, we employ a Matérn 5/2 kernel and constant mean function, following recommended settings in the trieste implementation. For SVGP we choose the same combination of priors, and select 500 inducing points to make results comparable to those in Vakili et al. (2021).

The small and medium size functions employed throughout this paper are drawn from Adorio and Diliman (2005) and Surjanovic and Bingham (2013).