Lawrence Livermore National Laboratory

Deep Kernel Bayesian Optimization

UNCLASSIFIED

February 16, 2021

J Bowden, J Song, Y Chen, Y Yue, T A Desautels*

*LLNL

To appear in Conference on Uncertainty in Artificial Intelligence, July 27-30, 2021. Online.

Disclaimer

Deep Kernel Bayesian Optimization

Abstract

Bayesian optimization using Gaussian processes (GPs) is a well-established framework that is widely studied both theoretically and in applications. One major design decision that often confronts the use of GPs is the choice of kernel function, which can greatly impact the performance of the resulting Bayesian optimization approach. In this paper, we explore the use of deep kernel learning to automatically learn a kernel function "on-the-fly" during Bayesian optimization. We introduce Deep Kernel Bayesian Optimization (DK-BO), a general framework for combining deep neural networks with Bayesian optimization, and investigate different neural network structures, kernel choices, acquisition functions, and posterior sampling methods. Intuitively, DK-BO can learn a kernel (and its corresponding dual feature map) that is lower dimensional than conventional RBF kernels while still retaining flexibility, thus leading to faster convergence. We demonstrate that DK-BO offers significant performance improvements over using RBF-based GPs on multiple real-world datasets and generalizes well with relatively little tuning.

1 INTRODUCTION

The optimization of an unknown function that is expensive to evaluate is a common problem in many domains. Examples include material design [Fleischman et al., 2017], protein design [Romero et al., 2013], clinical therapy optimization [Sui et al., 2017], controller optimization Berkenkamp et al. [2016], personalized recommender systems [Li et al., 2010], hyperparameter tuning [Snoek et al., 2012], amongst many others. Since the goal is to ultimately arrive at the best action or setting of the unknown function, the underlying optimization problem often requires reasoning about uncertainty in order to arrive at effective algorithms.

A popular paradigm for framing the above optimization challenge is Bayesian optimization [Frazier, 2018], whereby one posits a Bayesian prior over the unknown objective function. The most widely used prior distributions are Gaussian processes (GPs) [Rasmussen and Williams, 2006]. A key quantity of interest is how flexible the prior distribution is, which in a GP corresponds to how flexible its kernel function is. More flexible distributions typically require more samples to yield sharp posteriors with low uncertainty, leading to slower convergence. However, less flexible distributions may not properly model the application, leading to convergence to a suboptimal solution. Designing a suitable kernel for Gaussian processes is a notoriously challenging problem [Duvenaud et al., 2013], which typically requires significant trial-and-error. In many cases, one simply resorts to using a generic high-capacity kernel such as RBF or Matern.

In this paper, we propose Deep Kernel Bayesian Optimization (DK-BO), in which we integrate the recent advances in deep kernel learning [Wilson et al., 2016] with BO to automate the laborious process of kernel design. Our approach learns a kernel function "on-the-fly" during Bayesian optimization. We design an acquisition function based on Thompson or posterior sampling [Chapelle and Li, 2011, Russo and Van Roy, 2014] whereby we sample both the (deep) kernel and the objective function from the subsequent GP posterior (from the sampled kernel). Intuitively, one can view DK-BO as learning a kernel (and its corresponding dual feature map) that is lower dimensional than conventional RBF kernels while still retaining flexibility, thus leading to faster convergence without necessarily compromising optimality. We demonstrate the advantage of DK-BO with challenging real-world adaptive experiment design tasks in material science and protein engineering.

Submitted to the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021). **To be used for reviewing only**.

2 BACKGROUND AND RELATED WORK

2.1 GAUSSIAN PROCESSES

A Gaussian process (GP) [Rasmussen and Williams, 2006] models an infinite collection of random variables such that each finite set has a joint Gaussian distribution specified by its mean and covariance, or kernel function. GPs are the favored model for BO as they have a closed form formula for posterior inference. The kernel influences the prior with the radial basis function (RBF) kernel, $k(x, x') = \exp(-\gamma\|x - x'\|^2)$, and the linear kernel, $k(x, x') = x \cdot x'$, being two popular choices.

Custom kernel design is often necessary when standard kernels don't perform well, which is not uncommon when trying to optimize complex objective functions and/or data sets with many features. Kernel design for GPs is very difficult and is often an iterative tuning process [Duvenaud et al., 2013], which is not particularly conducive to real world adaptive experiment design where we'd like to apply BO and have it work without using up expensive queries to tune the model.

2.2 DEEP KERNEL LEARNING

Deep kernel learning [Wilson et al., 2016] uses a deep neural net (DNN) to extract a meaningful embedding upstream of GP inference. DKL combines a DNN with a standard GP model and base kernel and offers an advantage over GPs in that DNNs are capable of being more expressive and thus can learn better representations. We can jointly learn the DNN weights and hyperparameters of the kernel by differentiably optimizing the log likelihood loss function.

There has recently been an increase interest in combining deep learning with BO. Li et al. [2020] uses deep learning for multi-fidelity BO by appending the lower fidelity outputs to the higher fidelity inputs in order to learn some correlation. However, it does not address the more common single-fidelity BO case nor utilize deep kernel learning. Wistuba and Grabocka [2021] uses deep kernel learning to do effective transfer learning for few-shot BO and shows significant performance improvements over standard hyperparameter optimization methods. This is a testament to DKL's potential in a very relevant but specific domain. We aim to provide a more general framework in this paper.

2.3 BAYESIAN OPTIMIZATION

Bayesian optimization is a popular framework used for adaptive experiment design problems when evaluations are expensive. It consists of training a model (usually a GP) and choosing the next point to query by optimizing the model with respect to an acquisition function. This is repeated with an updated posterior until the budget is exhausted or convergence. Example acquisition functions include GP-UCB [Srinivas et al., 2009], entropy search [Hennig and Schuler, 2012], predictive entropy search [Hernández-Lobato et al., 2014] and max-value entropy search [Wang and Jegelka, 2017].

Despite the success of BO in many domains, it is largely limited to problems of moderate dimension. Scaling BO to higher dimensional problems presents a difficult challenge because as the number of dimensions in the input space increases, the amount of queries needed to attain sufficient coverage of the domain increases exponentially. Several recent works tackle this challenge from different perspectives. Kandasamy et al. [2015] and Djolonga et al. [2013] assume a high dimensional function has an inherent low dimensional structure and design algorithms to identify the low dimensional subspace. Wang et al. [2016] uses random projections to reduce the dimensionality. Our DK-BO provides another viable path to automatically identify low dimensional subspace via a DKL model.

2.4 POSTERIOR SAMPLING

Thompson sampling (TS) is used as a Bayesian acquisition function that treats the model's posterior like a distribution and chooses a point to query proportional to the probability that the point is the optimizer of the unknown objective function. It has been applied in parallel setting [Kandasamy et al., 2018] and federated setting [Dai et al., 2020]. Efficiently drawing a function sample from a general GP posterior remains challenging as the complexity scales cubically with the number of queries. Recent work by Wilson et al. [2020] presents a method to reduce the complexity to linear time in exchange for sampling from an approximate posterior. For DK-BO, we sidestep this challenge by using GPs with a linear kernel which admits a simple and efficient posterior sampling method since they are equivalent to Bayesian linear regression models [Duvenaud, 2014].

Monte Carlo dropout (MC dropout, MCD) is a posterior sampling method [Gal and Ghahramani, 2016] that extends dropout, a commonly used regularizer for training neural networks. Dropout is a common practice for avoiding overfitting by zeroing neurons with some probability p during neural network training [Srivastava et al., 2014]. At inference time, MC dropout extends this by sampling from the network weights in the same manner and making all model predictions with this sample. Gal and Ghahramani [2016] show that MC dropout can be interpreted as sampling a model from an approximate posterior from a deep GP model whose structure depends on the neural network layers. As a result, one could obtain predictive uncertainty by drawing multiple posterior samples and obtain a predictive distribution instead of a single point estimation given by a typical

neural network model. In DK-BO, we show that combining MCD with TS is an effective acquisition strategy.

3 PROBLEM STATEMENT

Black-box Function Optimization. We consider the general problem of optimizing a function $f : \mathcal{X} \to \mathbb{R}$ with query access only, i.e., we can obtain information about f only through (noisy) function evaluations $f(x) + \epsilon$ for $x \in \mathcal{X}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Our goal is to maximize f with iterative function evaluations. For applications in adaptive experimental design, each function evaluation call can be expensive. For example, the scientist might need to synthesize a new protein in the lab in order to measure a chemical property. As a result, we would like to optimize f with a small number of function evaluations.

Evaluation Metrics We measure the performance of an optimization algorithm with simple regret, which is a natural quantity measuring the gap between the value of a global optimum and that of the best solution found so far. Specifically, if $x^* = \arg\max_{x \in \mathcal{X}} f(x)$ and we have queried f at $x_1, x_2, \cdots, x_t$ so far. Then the simple regret at round t is defined as $r_t = f(x^*) - \max_{1 \le i \le t} f(x_i)$, which measures the gap between the best global design and the best design we find so far. Ideally, we want r_t to converge to 0 as fast as possible so we can find an approximate global maximizer with a small number of function evaluations.

4 DEEP KERNEL BAYESIAN OPTIMIZATION

We now present a general deep kernel Bayesian optimization (DK-BO) framework. The core contribution is to substitute standard GP models with deep kernel models [Wilson et al., 2016] in the Bayesian optimization process. With the introduction of a more expressive distribution model, there are associated design decisions related to model updates and acquisition function optimization.

4.1 DEEP KERNEL LEARNING

Deep kernel models [Wilson et al., 2016] are compositions of deep neural networks with kernel methods to harvest the representation learning power of neural networks while maintaining the non-parametric property. A standard GP model represents a distribution over functions with the specification of a mean function $\mu(x)$ and a kernel (covariance) function $k(x, x')$ for inputs $x, x' \in \mathcal{X}$. Deep kernel models use a deep neural network g to first transform the input feature into an embedding space, so the mean and kernel function become $\mu(g(x))$ and $k(g(x), g(x'))$, respectively. Wilson et al. [2016] observe that one can jointly learn the neural network parameters as well as the kernel hyperparameters in an end-to-end fashion.

Algorithm 1 Deep Kernel Bayesian Optimization

1: **Input:** a function f, a parameter space $\mathcal{X}$, a budget B, an initialization budget b for fitting a starting model
2: Sample b points from $\mathcal{X}$ uniformly at random and query f at those locations
3: Fit a deep kernel model on the b points
4: **for** $i = b + 1, \cdots, B$ **do**
5: Formulate an acquisition function $a : \mathcal{X} \to \mathbb{R}$
6: Find $x_i = \arg\max_{x \in \mathcal{X}} a(x)$
7: Query f at x_i and receive $f(x_i)$
8: Update the deep kernel model with the new data point
9: **end for**

4.2 DEEP KERNEL BAYESIAN OPTIMIZATION

Algorithm 1 provides an overview of the deep kernel Bayesian optimization framework. It follows closely the standard Bayesian optimization loop while substituting a standard GP with a deep kernel model. This modification introduces additional flexibility at the acquisition function formulation step (Line 5). More concretely, Gal and Ghahramani [2016] develops a method to draw posterior samples from a trained neural network with the technique of Monte-Carlo dropout. A natural question to ask is: can we incorporate this distributional view with existing sampling-based acquisition techniques, e.g., Thompson sampling? In Section 5, we provide a formal look at the implication of such combinations.

4.3 ACQUISITION FUNCTION OPTIMIZATION

Once we have computed an acquisition function, we need to maximize it to find a new candidate query. Regardless of the specific form of the acquisition function (e.g., posterior sampling and upper-confidence bound), it is a mapping a from the input domain $\mathcal{X}$ to some real value. If the domain is discrete, such as the case when designing proteins as a sequence of amino acids, the maximization process can be done by enumerating all possible input values. If the domain is continuous, the end-to-end differentiability enjoyed by the deep kernel model is also applicable to $a(x)$. We could use gradient descent for optimization to obtain local optima. Practically, automatic differentiation tools such as PyTorch [Paszke et al., 2019] make this procedure easy to implement. As a could be non-convex, repeated random restarts can be used to improve the optimization outcome.

5 THEORETICAL DISCUSSION & INTERPRETATION

In this section, we seek to analyze the effect of combining Monte-Carlo dropout in neural network with Thompson sampling in posterior.

DKL with Dropout as Variational Inference on Deep GP Posterior. In the following, we show that training a DKL model with dropout applied before each weight layer is equivalent to an approximation of a variational approximation to a deep Gaussian process [Damianou and Lawrence, 2013] by extending the analysis in Gal and Ghahramani [2016] to DKL.

Within a DKL model with L feedforward layers, assume weights for each layer $W_i \in \mathbb{R}^{K_i \times K_{i-1}}$. Let $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_L\}$ denote the neural network weights. We also implement dropout for layer i with probability p_i. Finally, (μ, k_θ) denote the mean and kernel function for the last GP layer, θ are hyperparameters for a kernel function, e.g., length scale for a RBF kernel. The analysis in Gal and Ghahramani [2016] defines a deep GP with L layers and a covariance function defined with respect to layer weights. For a DKL model, we extend such a deep GP with an extra GP layer. Next, we briefly describe the connection between training DKL with log-likelihood loss and approximating the posterior of a deep GP model.

Given n observations $(\mathbf{X}, \mathbf{Y}) = (\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n)$, we are interested in the posterior predictive distribution at a new point x,

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, \mathbf{W}, \theta) p(\mathbf{W}, \theta|\mathbf{X}, \mathbf{Y}) d\mathbf{W} d\theta \tag{1}$$

$$p(y|x, \mathbf{W}, \theta) = \mathcal{N}(y; \mu(\hat{x}), k_\theta(\hat{x}, \hat{x})) \tag{2}$$

where $\hat{x} = f(x, \mathbf{W})$ is the embedding after the neural network layers. The notation in Equation 2 denotes that $y \sim \mathcal{N}(\mu(\hat{y}(x, \mathbf{W})), k_\theta(\hat{y}(x, \mathbf{W}), \hat{y}(x, \mathbf{W})))$. The posterior distribution $p(\mathbf{W}, \theta|\mathbf{X}, \mathbf{Y})$ is intractable and we wish to approximate it with a tractable distribution $q(\mathbf{W}, \theta)$. For simplicity, we assume q decomposes to a product of $q(\mathbf{W})q(\theta)$. Following the same treatment for dropout in Gal and Ghahramani [2016], we define $q(\mathbf{W})$ as:

$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}([z_{i,j}]_{j=1}^{K_{i-1}})$$
$$z_{i,j} \sim \text{Bernoulli}(p_i), 1 \leq i \leq L, 1 \leq j \leq K_{i-1}$$

with $\mathbf{M}_i$s being variational parameters. We place a Gaussian prior on $q(\theta) = \mathcal{N}(\theta; \mu_\theta, \sigma_\theta^2)$. By minimizing the KL divergence

$$\begin{aligned}&\text{KL}(q(\mathbf{W})q(\theta)||p(\mathbf{W}, \theta|\mathbf{X}, \mathbf{Y})) \\ &= -\int q(\mathbf{W})q(\theta) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}, \theta) d\mathbf{W} d\theta \\ &+ \text{KL}(q(\mathbf{W})q(\theta)||p(\mathbf{W})p(\theta))\end{aligned}$$

We can obtain an unbiased estimator for the first term by sampling a $\hat{\mathbf{W}} \sim q(\mathbf{W}), \hat{\theta} \sim q(\theta)$. Then $\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}, \theta)$ is precisely the log-likelihood loss function we use to train the DKL model. We name this part of the loss as $\mathcal{L}_{\text{DKL}}$.

The second term decomposes to $\text{KL}(q(\mathbf{W})||p(\mathbf{W})) + \text{KL}(q(\theta)||p(\theta))$. Typically we impose standard Gaussian priors on $p(\mathbf{W})$ and $p(\theta)$. Then the first KL-divergence term is present in the analysis in Gal and Ghahramani [2016] and reduces to l_2 regularization terms on weights and biases and the second KL term is between two Gaussian priors. Putting everything together, we see that the loss function for the variational inference is $\mathcal{L}_{\text{VI}} = \mathcal{L}_{\text{DKL}} + \sum_{i=1}^L p_i \|M_i\|^2/2 - \log \sigma_\theta + (\sigma_\theta^2 + \mu_\theta^2 - 1)/2$. As a result, optimizing the log-likelihood for a DKL model with dropout is approximately equivalent to variational inference on a deep GP posterior.

DK-BO with MC-dropout as Approximate Posterior Sampling. The above analysis shows that DKL results in an approximate deep GP posterior; furthermore, Gal and Ghahramani [2016] suggests that MC-dropout on a deep neural network could be interpreted as sampling from the resulting approximate deep GP posterior. Therefore, with MC dropout and Thompson sampling on the GP layer, it could be interpreted as an approximate posterior sampling procedure for deep kernel Bayesian optimization.

Riquelme et al. [2018] studies performances of various deep Bayesian models with Thompson sampling from approximate posteriors. In their experiments on contextual bandits, they find that the NeuralLinear model which applies Bayesian linear regression on features learned through neural network performs consistently well. Separately, models trained with dropout also perform decently when the dropout rate is tuned. These results that our DK-BO approach is empirically sound.

The theoretical aspect is less clear. Recently, Phan et al. [2019] studies the k-armed bandit problem, and analyzes the asymptotic behavior of the Bayesian and frequentist regret for posterior sampling with approximate inference. They show that even small divergence from the exact posterior can lead to linear regret. We wish to note that examples leading to the undesired behaviors are highly stylized and consist of 2 arms so their practical implication is unknown.

Representation Learning Automates Kernel Design. The representation learning provided by a deep neural network automates the kernel design process. We show in the experiments (Section 6.4) that such capacity ensures that a simple linear kernel provides excellent empirical performance. As pointed out in [Duvenaud, 2014], kernel design is usually an art that has significant impact in GP model performance. Misspecification of kernel can lead to undesirable results [Sollich, 2001, Beckers et al., 2018]. A DKL model with a linear kernel provides an elegant way to address this issue by following Mercer's theorem [Mercer, 1909]: the

linear kernel provides a generic inner product form while representation learning explores the best way to map the feature. A natural question is whether this setup limits the type of kernels. A subtle point about Mercer's theorem is that the feature map for a kernel might be infinite dimensional (e.g., the radial basis function (RBF) kernel). Given the neural network maps only to a finite dimension embedding, this might restrict learnable kernels. However, the classical work by Rahimi and Recht [2007] shows that for any shift-invariant kernel, that is a kernel $k(x, y)$ that only depends on the difference $x - y$ (e.g., RBF kernel, Matern kernel), there exists a low dimensional feature map z such that the inner product $\langle z(x), z(y) \rangle$ approximates $k(x, y)$ to arbitrary precision. As a result, DKL models are able to recover commonly used kernels and have the flexibility to fit additional forms of kernels based on the representation learning.

Another benefit of using a linear kernel is the ease to apply Thompson sampling in the BO loop. Sampling a function from a Gaussian posterior is hard for generic kernels [Wilson et al., 2020]. A workaround is to discretize the parameter space [Kandasamy et al., 2018], which limits the accuracy of Thompson sampling. However, GP with a linear kernel is equivalent to Bayesian linear regression [Duvenaud, 2014] and has an easy posterior sampling procedure. The sampled function is a linear function, which is simple to optimize to compute a query point required by Thompson sampling. The representation learning can be generalized to fundamentally different input types such as images and graphs as well and the modular property of DKL provides flexibility in expressing kernels.

6 EXPERIMENTS

In this section, we show that DK-BO models outperform BO with standard GPs (GP-BO) on 3 real-world datasets. We study different practical scenarios for designing probabilistic models. In Section 6.3, we show that DK-BO performs better than GP-BO *on average*. This result caters to the practical setting where one might not have much budget for tuning during optimization. In Section 6.4, we study three important design decisions in further optimizing the DK-BO performance: 1) the final layer GP kernel and the acquisition function; 2) the embedding dimension of the neural network; 3) the inclusion of dropout and Monte Carlo dropout as posterior sampling.

6.1 DATASET DESCRIPTIONS

We selected several real-world problems to assess DK-BO's performance on high-dimensional, difficult-to-optimize tasks. Note that these tasks have discrete input spaces, as is often the case in reality.

The first is a protein engineering dataset, for which we wish to maximize stability fitness predictions for the Guanine nucleotide-binding protein GB1 given different sequence mutations in a target region of 4 residues [Wu et al., 2019]. There are 26^4 possible orderings given 26 amino acids and 4 positions, and this problem has a 2048-dimension input created using a Transformer model [Rao et al., 2019]. GB1 has been well studied by biologists and its domain is known to be highly rugged and dominated by "dead" variants with very low fitness scores [Wittmann et al., 2020]. Because of its high input dimensionality and enormous input space, this dataset is very challenging for traditional GP kernels.

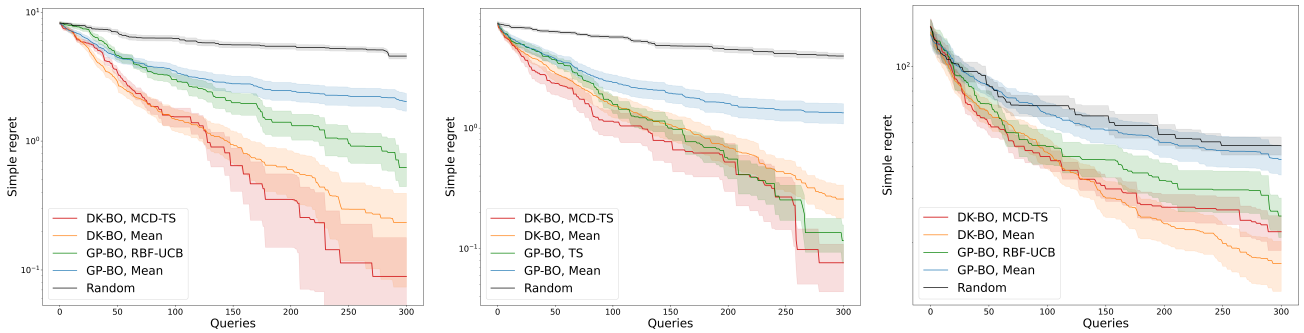
The next is another protein engineering dataset describing a set of antigen/antibody binding calculations. These calculations, executed using supercomputing resources, estimate the change in binding free energy between 71769 modified antibodies and the SARS-CoV-2 spike protein, as compared to a reference antibody. A variant antibody with better (lower) binding free energy will putatively be a stronger binder. These calculations are executed via FoldX, a structure-based protein binding code [Schymkowitz et al., 2005], at a cost of several CPU hours each, and are produced during an antibody design process [Desautels et al., 2020]. Inputs are described with an 80-dimensional feature vector that, relative to the reference sequence, describes changes in the interface between the antibody and the corresponding target region on the SARS-CoV-2 spike. In practice, 20 to 30 of these dimensions are static across the dataset and thus can be trimmed prior to optimization. This is a particularly relevant problem setting when trying to rapidly choose antibody candidates to respond to a new disease in a timely fashion.

The third is a nanophotonics dataset, for which we wish to optimize a weighted figure of merit quantifying the fitness of the transmission spectrum of a possible device design as assessed by a numerical solver [Song et al., 2018b]. This problem has a 5-dimensional input corresponding to the physical design dimensions of a potential filter. Although the input dimension is not very large, the function represents a discrete solution of Maxwell's equations and has a complex value landscape.

6.2 EXPERIMENTAL SETUP

Experiments were conducted by running 20 trials for each model and plotting average simple regret for each model with an error bar of one standard error over all runs. Pseudo-random seeds were fixed for reproducibility across models in a given trial run and we initialize all models in a given trial with 10 pseudo-random data points. All models were implemented using GPyTorch [Gardner et al., 2018] and given an optimization budget of 300 additional queries.

We use $\beta = 0.1$ as the exploration factor for GP-UCB and DKL-UCB; Thompson sampling does not require such a



(a) GB1 dataset; 2048-D (b) FoldX dataset; 58-D (c) Nanophotonics dataset; 5-D

Figure 1: Simple regret on real datasets for DK-BO with Monte Carlo dropout and Thompson sampling, mean DK-BO and GP-BO, the best GP model, and random choice. Error bar shows one standard error over 20 runs for each experiment.

parameter. For DK-BO, we use the leaky ReLU activation function in the deep network and set DNN architectures of [2048-500-150-50], [58-500-100-20], and [5-300-150-50] for the GB1, FoldX, and nanophotonics datasets respectively (see 6.1). We discuss architecture choice in depth in Section 6.4. When using the standard train time dropout (denoted DO), we dropout nodes with probability p during training only. When using MC dropout, we follow the aforementioned procedure and *also* sample a single set of weights from the network with probability p for use during prediction.

6.3 MAIN RESULTS

In Figure 1, we plot simple regret for all three real-world datasets discussed above. DK-BO with a linear kernel and MC dropout and Thompson sampling (MCD-TS) is shown in bold red and outperforms the best GP model (shown in green) on all three datasets. An average of DK-BO models (shown in orange) outperforms the average of GP-BO models (shown in blue) on all three datasets as well. Both averages include: linear kernel with TS, linear kernel with UCB, and RBF kernel with UCB. The DK-BO average also includes MCD-TS. These are compared with random choice (shown in black), which performs poorly on all datasets.

On the GB1 dataset (Figure 1a), the DK-BO average consistently outperforms both the GP-BO average and the best GP model and exhibits a relative final regret improvement of 88.4% and 62.4% respectively. MCD-TS uses $p = 0.3$ as its dropout rate at both test time and train time because the input space is very high dimensional and thus can afford to be sampled from at a high rate. MCD-TS improves upon the DK-BO average by 62.0% by the end of the optimization.

On the FoldX dataset (Figure 1b), the DK-BO average consistently outperforms the GP-BO average with a relative final regret improvement of 80.7%. We use a dropout rate of $p = 0.2$ for MCD-TS. Although the best GP model (GP-UCB, RBF kernel) outperforms the DK-BO average, MCD-TS offers a relative regret improvement of 34.5% over the best GP model.

On the nanophotonics dataset (Figure 1c), the DK-BO average improves upon final regret of the average GP-BO and best GP model by 61.3% and 35.1% respectively. The optimal dropout rate for MCD-TS was found to be $p = 0.1$, and although it improves upon the best GP model by 13.3%, it fails to outperform the DK-BO mean. This is consistent with the intuition that low dimensional problems can't afford very much dropout, discussed in Section 6.4 In this problem setting the GP-BO average does not significantly outperform random choice, representing a nontrivial subset of adaptive experiment designs in which we are better off choosing randomly than doing BO. However, DK-BO does much better than random choice with relatively little tuning, offering a clear alternative.

6.4 DESIGN CHOICES

Kernel and Acquisition Function We consider kernel and acquisition function choice in Figure 2. Note that the GP approximation used by TS assumes a linear kernel [Wilson et al., 2020], meaning we can only use the UCB acquisition function for the RBF kernel. For both DK-BO and GP-BO, we examine the linear kernel with each of the TS and UCB acquisition functions, and the RBF kernel with UCB. These represent some of the most popular, "go-to" kernels used in Bayesian optimization and are likely the first kernels one would try when working on a new adaptive experiment design problem. This choice results in significant performance differences across the various GP models. DK-BO is largely unaffected by such kernel and acquisition choices and all three configurations have similar regret curves with overlapping error bounds, demonstrating its stability. Although DK-BO with RBF kernel and UCB achieves the lowest regret here, the linear kernel and TS combination is not significantly worse. Thus, we focus on the linear kernel for the other experiments and analyses as this is in line with

our theoretical interpretation (Section 5).

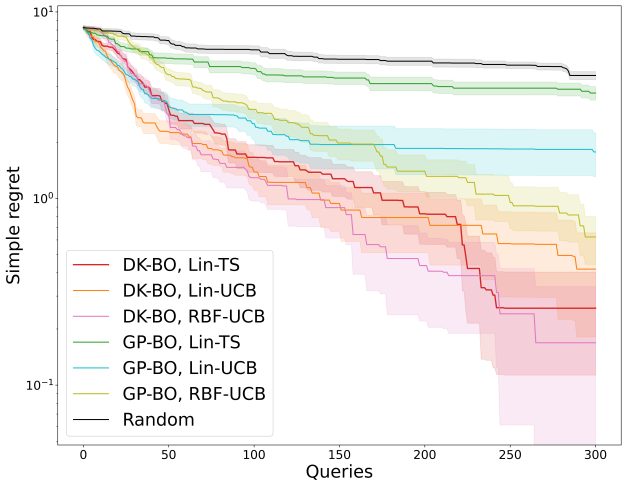


Figure 2: Simple regret for GB1 protein optimization task with various kernel and acquisition function combinations. Error bar shows one standard error over 20 runs for each experiment.

Neural Network Architecture Neural network architecture is one of the most important design choices for any deep learning model and also one of the most difficult . We focus on the final embedding dimension of the DNN output as this has the most influence on what representation is passed into the GP kernel during DKL and how much information content it can hold. Wilson et al. [2016] suggest using a $[d - 1000 - 500 - 50 - 2]$ architecture for datasets with less than 6000 points (in the context of regression tasks); this architecture is denoted by '2-D DK-BO' in Figure 3. We compare this with DK-BO models using a linear kernel with TS acquisition function and architectures of $[d - 500 - 150 - e_f]$, where $e_f \in [10, 50, 100, 200]$. These architectures have less complex hidden layers given that in a BO setting we often have a training set with much fewer than 6000 points. e_f is larger on the other hand, because a larger output dimension allows for more flexibility and expressiveness to be passed along to the GP base kernel regardless of the objective's complexity. It is much more likely that relevant information will be lost when mapping to a smaller latent space. Figure 3 illustrates this intuition: for smaller embedding dimensions like 2 and 10, regret performance is poor, whereas increasing dimension to 50 or 100 results in significant improvements, but an embedding dimension that is too large (such as 200) likely results in inefficient feature extraction, potentially making it more challenging for the GP base kernel to fit the data. Embedding choice is especially important in the case of the TS acquisition function, since the embedding is used to compute a posterior distribution and a 50D embedding has more flexibility than a 2D one. As it is worse to use an embedding dimension too small and lose relevant information, we recommend erring on the safe side and starting with a default embedding dimension of 50. Some problems may perform better with smaller embeddings; this can be tuned if budget allows.

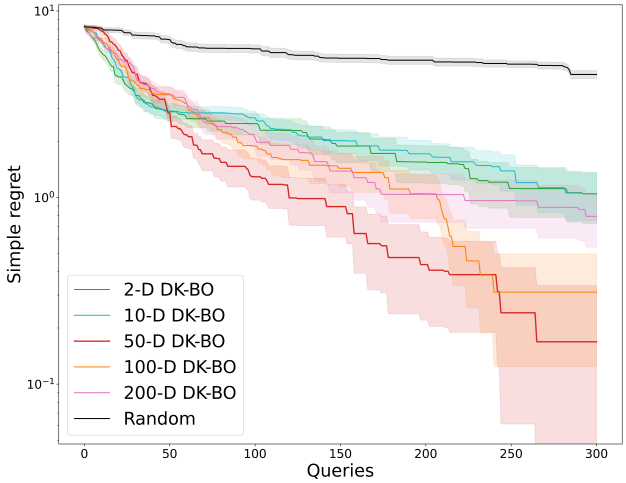


Figure 3: Simple regret for GB1 protein optimization task using a linear kernel and TS with various DNN embedding dimensions. Error bar shows one standard error over 20 runs for each experiment.

Monte Carlo Dropout and Thompson Sampling Using DKL for BO introduces the option of neural network dropout. Training dropout (denoted DO) can be interpreted as regularization via variational inference and may provide performance improvements by protecting against overfitting. Monte Carlo dropout (denoted MCD) combines training dropout with test time dropout and can be considered sampling from the set of neural network weights for prediction [Gal and Ghahramani, 2016]. We can interpret MCD combined with Thompson sampling as approximate posterior sampling, as discussed in Section 5.

In Figure 4, we compare MCD-TS with only train-time dropout (DO-TS) and DK-BO without any dropout (TS) on the GB1 dataset. A dropout rate of $p = 0.3$ is used for both MCD and DO. MCD-TS consistently performs better than DO-TS and TS alone with a 71.6% and 65.6% relative improvement in regret by the end of the optimization. This demonstrates that MCD-TS offers empirical benefits over both normal dropout models and naive DK-BO in addition to its theoretical interpretation (Section 5).Section 5

The dropout rate p used for MCD-TS on each dataset was tuned empirically by testing each $p \in \{0.1, 0.2, 0.3\}$ as dropping weights out at a higher rate likely means losing a significant amount of information. However, we can also correlate p to dimensionality of the dataset: GB1 has 2048 di-

mensions and an optimal p of 0.3; FoldX has 58 dimensions and an optimal p of 0.2, and Nanophotonics has 5 dimensions and an optimal p of 0.1. We see the largest gains for MCD-TS on the highest dimensional dataset (Figure 1a) and the smallest gains on the lowest dimensional dataset (Figure 1c) where MCD-TS is outperformed by the average of DK-BO models that don't use dropout at all. This suggests that for problems where raw dimension is larger than the true dimension (which encompasses most problems), MCD-TS with higher dropout rates can offer substantial gains because we can afford to lose a lot of the given information. Conversely, when raw dimension is likely representative of the true dimension , higher dropout rates are more likely to be detrimental as we cannot afford to discard very much information, suggesting the use of plain DK-BO without MC dropout for such problems. These results demonstrate that DK-BO with MCD-TS does indeed offer significant empirical gains in addition to its theoretical basis, particularly on challenging, high-dimensional datasets.

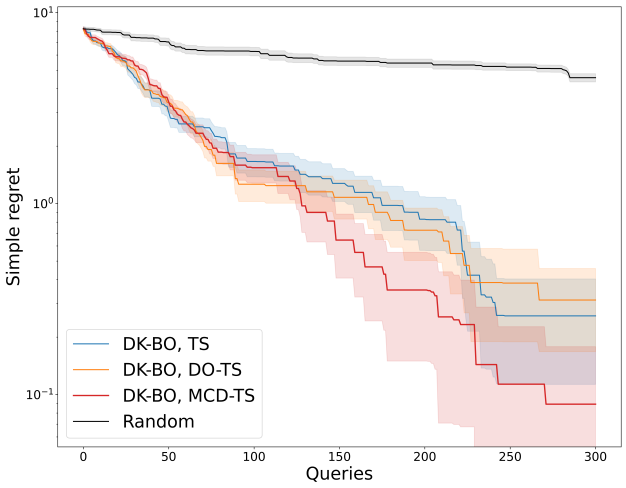


Figure 4: Simple regret for GB1 protein optimization task with and without dropout and MC dropout. Error bar shows one standard error over 20 runs for each experiment.

7 DISCUSSION AND FUTURE WORK

In this paper, we demonstrate that DK-BO offers significant performance improvements over current GP methods on complex, high-dimensional, real-world datasets representative of the adaptive experiment design domain. We provide both theoretical and empirical reasoning for the design choices that follow, including neural network architecture and kernel choice. We also introduce the combination of DK-BO, MC dropout, and Thompson sampling as a deep kernel method that can be interpreted in a Bayesian manner. We also identify a set of satisfactory model parameters such that DK-BO can be used on a variety of functions and input dimensions without much tuning, and release an accompanying software package constructed using GPyTorch [Gardner et al., 2018], available here <github link to be put after accepted>.

There is potential to ensemble DKL models with different kernels or acquisition strategies or to train multiple DNNs, potentially with a subset of the acquired dataset as suggested by BESA [Baransi et al., 2014], and sample networks in a Bayesian fashion. These ensembles also have the option of being randomly sampled, sequential (rotate through set of models in some order), averaged, or combined into a weighted average via meta-learning [Fort et al., 2019]. Integrating a deep kernel with our Bayesian optimization framework opens up many interesting extensions on the network front.

We plan to expand our implementation to multi-fidelity BO and potentially multi-task and multi-objective versions as well [Song et al., 2018a], using multi-fidelity versions of the nanophotonics and coronavirus binding datasets described above. Each of these expansions of BO requires learning multiple objective functions and is likely to require additional regret formulations and analysis. More traditional GP models often require complex kernel design here and have not been able to perform very well. In contrast, deep kernel based approaches have the advantage of being flexible and thus may be able to learn meaningful underlying embeddings for these and other complex Bayesian optimization settings involving multiple functions.

References

Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. Sub-sampling for multi-armed bandits. *Proceedings of the European Conference on Machine Learning*, 2014.

Thomas Beckers, Jonas Umlauft, and Sandra Hirche. Mean square prediction error of misspecified gaussian process models. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1162–1167. IEEE, 2018.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24:2249–2257, 2011.

Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33, 2020.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215, 2013.

Thomas Desautels, Adam Zemla, Edmond Lau, Magdalena Franco, and Daniel Faissol. Rapid in silico design of antibodies targeting sars-cov-2 using machine learning and supercomputing. *BioRxiv*, 2020.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. In *Neural Information Processing Systems*, number CONF, 2013.

David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174. PMLR, 2013.

Dagny Fleischman, Luke A Sweatlock, Hirotaka Murakami, and Harry Atwater. Hyper-selective plasmonic color filters. *Optics Express*, 25(22):27386–27395, 2017.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv:1912.02757*, 2019.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *arXiv:1809.11165*, 2018.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012. URL https://bit.ly/2x5KMQC.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pages 918–926, 2014.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR, 2015.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142. PMLR, 2018.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

Shibo Li, Wei Xing, Robert Kirby, and Shandian Zhe. Multi-fidelity bayesian optimization via deep neural networks. *Neural Information Processing Systems*, 2020.

J Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

My Phan, Yasin Abbasi Yadkori, and Justin Domke. Thompson sampling and approximate inference. *Advances in Neural Information Processing Systems*, 32:8804–8813, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1177–1184, 2007.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with tape. *arXiv:1906.08230v1*, 2019.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://bit.ly/2tYpBix.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3):E193–E201, 2013.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

Joost Schymkowitz, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. The foldx web server: an online force field. *Nucleic Acids Research*, 33(suppl-2):W382–W388, 07 2005.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Peter Sollich. Gaussian process regression with mismatched models. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pages 519–526, 2001.

Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. *arXiv:1811.00755*, 2018a.

Jialin Song, Yury Tokpanov, Yuxin Chen, Dagny Fleischman, Kate Fountaine, Harry Atwater, and Yisong Yue. Optimizing photonic nanostructures via multi-fidelity gaussian processes. *arXiv:1811.07707*, 2018b.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv:0912.3995*, 2009.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yanan Sui, Yisong Yue, and Joel W Burdick. Correlational dueling bandits with application to clinical treatment in large decision spaces. In *IJCAI*, 2017.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. *arXiv preprint arXiv:1703.01968*, 2017. URL https://arxiv.org/pdf/1703.01968.pdf.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR, 2020.

Martin Wistuba and Wistuba Grabocka. Few-shot bayesian optimization with deep kernel surrogates. *International Conference on Learning Representations*, 2021.

Bruce J Wittmann, Yisong Yue, and Frances H Arnold. Machine learning-assisted directed evolution navigates a combinatorial epistatic fitness landscape with minimal screening burden. 2020.

Zachary Wu, S. B. Jennifer Kan, Russell D. Lewis, Bruce J. Wittmann, and Frances H. Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *PNAS*, 2019.

A APPENDIX

A.1 ADDITIONAL EXPERIMENTAL DETAILS

We train all models for a constant 100 episodes and choose the maximizer of the acquisition function at each episode. For each model and at each time step, we calculate the acquisition function on all possible inputs because optimizing a discrete domain (as many real-world problems require) isn't as straightforward as the continuous case. Note that DKL (and GP) models are reinitialized during each episode of BO to avoid weights being biased toward earlier points.