Be Greedy – a Simple Algorithm for Blackbox Optimization using Neural Networks

Biswajit Paria BPARIA@CS.CMU.EDU
Barnabàs Pòczos BAPOCZOS@CS.CMU.EDU
Pradeep Ravikumar PRADEEPR@CS.CMU.EDU
Jeff Schneider JEFF4@ANDREW.CMU.EDU
Carnegie Mellon University

Arun Sai Suggala ARUNSS@GOOGLE.COM
Google Research

(ordered alphabetically)

Abstract

Neural network based methods have recently gained popularity for the problem of optimization of expensive blackbox functions. Many of such methods are based on extensions of the classical ideas of Thompson Sampling (TS), Upper Confidence Bound (UCB), Expected Improvement (EI), to neural networks. However, these techniques lack either in achieving the right balance between exploration and exploitation, or are computationally expensive, resulting in poor practical performance. In this work, we empirically demonstrate that explicitly performing EI, TS, or UCB on neural networks is not essential. A simple greedy algorithm that fits a neural network to the current set of observations, and uses the learned network as the acquisition function to determine the next query point, often performs quite well. Our key insight is that such a greedy algorithm implicitly samples from a Gaussian process in wide neural networks. We use this insight to derive regret bound of our algorithm in the infinite-width limit and show that it depends on the eigen-spectrum of the Neural Tangent Kernel. Through extensive experiments, we show that the greedy algorithm compares favorably to GPs and outperforms several neural network based blackbox optimization techniques.

1. Introduction

Global optimization of expensive blackbox functions is an important problem with applications in a number of engineering and scientific disciplines (Snoek et al., 2012; Romero et al., 2013; Ueno et al., 2016; Papalexopoulos et al., 2021). For example, hyper-parameter tuning (Snoek et al., 2012; Falkner et al., 2018) and neural architecture search (Kandasamy et al., 2018). Owing to its importance, numerous techniques have been proposed to find the global optimum of blackbox functions (Mockus, 1982; Huang et al., 2006; Kleinberg et al., 2008; Srinivas et al., 2009). Of these, Bayesian Optimization (BO), and in particular GP optimization, has been quite successful in practice, and is the de facto technique in the industry (Golovin et al., 2017), owing to the *simplicity* and *expressivity* of GPs. They are easy to use, and impose minimal structural assumptions on the unknown blackbox function.

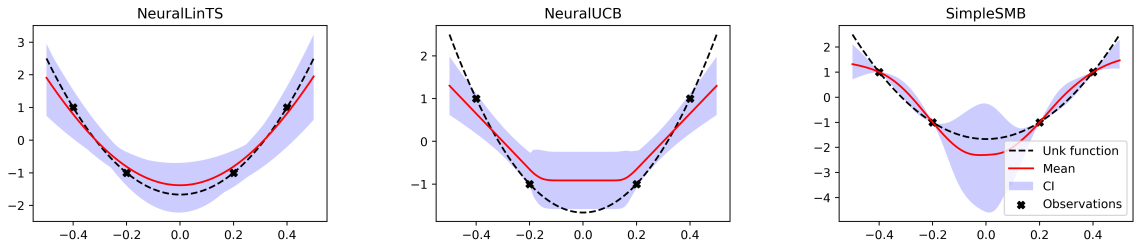


Figure 1: Plots show the confidence bands estimated by various neural network based BBO techniques on a $1D$ (noiseless) blackbox function. NeuralLinTS, NeuralUCB are the techniques developed by Snoek et al. (2015); Zhou et al. (2020) respectively. SimpleSMB is our greedy technique. It can be seen that our confidence bands are narrower for points close to observations.

Despite its success, GP optimization faces issues with *scale* and *flexibility*. In particular, the computational complexity of GPs scales cubically with the number of function evaluations (Liu et al., 2020). This makes GPs less appealing in high-dimensional search spaces when a large number of function evaluations are often needed to identify a good solution. In addition, GPs are not easily extendable to structured domains involving objects such as images, audio, text, and graphs (Van der Wilk et al., 2017; Kandasamy et al., 2018). The design of appropriate kernels for such structured objects is problem specific and is still an ongoing research problem with state-of-the-art structured kernels not as performant as modern neural feature representations (Kim et al., 2021).

An emerging line of work has tried to address these issues by developing neural network (NN) based blackbox optimization techniques (Springenberg et al., 2016; Riquelme et al., 2018; Zhou et al., 2020; Kim et al., 2021). These approaches use NNs as their surrogate models, and rely on standard acquisition functions such as EI (Jones et al., 1998), TS (*a.k.a* posterior sampling) (Thompson, 1933; Russo and Van Roy, 2014), UCB (Agrawal, 1995) to determine the next query point. The computational complexity of these approaches typically grows linearly with the number of function evaluations, making them attractive for high-dimensional problems. In addition, these techniques can be easily extended to structured domains involving objects such as images, audio, and graphs, for which we have neural architectures that can encode priors from the domain (for instance, convolutional neural networks).

One of the key challenges in designing NN based approaches lies in extending the classical ideas of EI, TS, UCB to neural networks. For instance, UCB and EI require construction of valid confidence intervals for the unknown function. While this is easy for simple models such as linear models (Chu et al., 2011), constructing such intervals is highly non-trivial for complex models such as neural networks. TS, on the other hand, require computing the posterior distribution of Bayesian Neural Networks (BNNs), which, again, is non-trivial (Springenberg et al., 2016). Existing approaches overcome these issues by relying on a variety of heuristics. Snoek et al. (2015); Xu et al. (2020) perform LinearUCB/LinearTS on top of the representation of the last layer of the neural network. Zhang et al. (2020); Zhou et al. (2020) work in the infinite-width limit of NNs, where the networks can be viewed as linear models, and perform LinearUCB/LinearTS on the "linearized networks". However, both these heuristics have

several drawbacks: (a) they often lead to conservative confidence bands, which in turn reduces the speed of convergence of the algorithm (see Figure 1), (b) some of these techniques are computation and memory intensive as they involve manipulation of large matrices with sizes quadratic in the number of NN parameters, in each iteration (Zhang et al., 2020; Zhou et al., 2020), (c) some are geared towards finite search spaces and their extension to continuous domains is non-trivial (Zhang et al., 2020). Another heuristic that is often used involves mimicking the posterior distribution of BNNs using ensembles of NNs (Riquelme et al., 2018; Kim et al., 2021). However, this heuristic requires a large number of models in the ensemble to get a good approximation of the posterior distribution.

In this work, we show that one need not explicitly estimate the confidence sets or posterior distributions for NN based blackbox optimization. We show that a simple greedy algorithm which fits a NN to the current set of observations, and uses the learned network as the acquisition function achieves better performance than existing NN based approaches. A crucial aspect of our algorithm is that we train our neural network surrogate model from scratch in each iteration, and rely on stochastic gradient descent with randomly initialized weights. Our key insight is that such an approach mimics GP sampling and Thompson sampling in wide neural networks (He et al., 2020; Lee et al., 2019).

Blackbox Optimization. In blackbox optimization (BBO), our aim is to minimize a (potentially non-convex) function f^* over an action space $\mathcal{X}$, given only zeroth-order oracle access to the function. That is, the only information about f^* comes via querying the oracle at a point $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and observing $y = f^*(x)+\xi$. Here, ξ is the independent noise, sampled from the Gaussian distribution $\mathcal{N}(0, \sigma_*^2)$. The special case of $\sigma_*^2 = 0$ is called *noiseless* BBO. Further background and related work on Gaussian processes, Bayesian optimization, and wide neural networks can be found in Appendices A and B.

2. Greedy Algorithm

We now present our greedy algorithm for Blackbox Optimization (BBO) (shown in Algorithm 1). Our algorithm has three key hyper-parameters: initialization weight variance γ, noise variance σ^2, and scale parameter ν. γ dictates the kind of surrogate models we fit to the data. Smaller values of γ lead to smoother surrogate models and larger values lead to less smoother models. The scale parameter ν controls the exploration-exploitation trade-off (set to 1 in our experiments). Larger values lead to more exploration.

There are two key steps in our algorithm, namely surrogate model building step (line 7 of Algorithm 1) and the acquisition step (line 8). Our acquisition step simply involves minimizing the learned surrogate model. The first surrogate model builder we consider is SimpleSMB which is described in Algorithm 2. For the noiseless case, where $\sigma^2 = 0$, this algorithm simply fits a neural network to the observed data by minimizing the squared ℓ_2 loss. The network is learned using (stochastic) gradient descent with the parameters randomly initialized using NTK initialization scheme described in Appendix A. In the noisy case, the algorithm regularizes the learned network to prevent over-fitting. Without regularization, the learned network interpolates the data, which leads to undesirable behavior. To avoid this, we use two forms of regularization in our algorithm. The first one involves perturbing the response variables $\{y_i\}_{i=1}^n$ with Gaussian noise. The second one is the "distance-from-initialization"

term (line 3) which biases the learned network to stay close to the initialization. Both these forms of regularization play a key role in our algorithm. In Appendix C, we show that training the network in this way is equivalent to sampling from GPs in the infinite-width limit.

Algorithm 1 Neural Greedy

Input: NN initialization weight variance γ, noise variance σ^2, scale parameter ν, budget T, surrogate model building sub-routine: SurrModelBuilder

1: Initialization: $\mathcal{D}_0 = \{\}$
2: **for** $t = 1, \cdots, T_e$ **do** ▷ Random Exploration Phase
3: Sample x_t randomly from the domain
4: Query blackbox function at x_t and obtain y_t
5: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$
6: **for** $t = T_e + 1, \cdots, T$ **do** ▷ Greedy Phase
7: $\tilde{f}_t \leftarrow$ SurrModelBuilder$(\mathcal{D}_{t-1}, \gamma, \sigma^2, \nu)$
8: $x_t \leftarrow \text{argmin}_{x \in \mathcal{X}} \tilde{f}_t(x)$
9: Query blackbox function at x_t to obtain y_t
10: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$
11: **return** $y_{\text{best}} = \min(\{y_t\}_{t=1}^T)$

Algorithm 2 Simple Surrogate Model Builder (SimpleSMB)

Input: Data $\{(x_i, y_i)\}_{i=1}^{t-1}$, initialization weight variance γ, noise variance σ^2, scale parameter ν

1: Add i.i.d noise to targets: $y_i' = y_i + \nu\epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
2: Initialize $\theta_0 \sim$ NTK-Init(γ) ▷ Random weight initialization
3: Solve $\min_\theta \sum_{i=1}^{t-1}(y_i' - \nu f(x_i, \theta))^2 + \sigma^2\nu^2\|\theta - \theta_0\|_2^2$ using GD/SGD with θ_0 as initialization, and obtain θ_t
4: **return** $\nu \times f(\cdot, \theta_t)$

Discussion. While we show that Algorithm 1 results in a sample from a GP, and works well in practice (as shown in Section 3), it doesn't perform posterior sampling with respect to any kernel. We propose a slight modification to SimpleSMB called PosteriorSMB based on recent developments in wide neural network theory (He et al., 2020) which results in a posterior sample with respect to the neural tangent kernel in the infinite-width limit. At a high level, this method perturbs the surrogate model of SimpleSMB with a random function that helps us sample from the posterior distribution. Further details about this method can be found in Appendix D.

Both the forms of regularization used in Algorithm 2 have been studied in the literature in various contexts. For instance, Nagarajan and Kolter (2019) showed that controlling $\|\theta - \theta_0\|_2$ helps in better generalization of the learned network. Kveton et al. (2020) studied reward perturbations in the context of GLM bandits. However, reward perturbation alone doesn't suffice for NNs. Not using $\|\theta - \theta_0\|_2$ can lead to networks with poor generalization and wider confidence bands which slows down the convergence of bandit algorithm (see Section C for more details).

Kannan et al. (2018) studied a greedy algorithm for linear contextual bandits. Unlike our

algorithm which perturbs the response variables, their algorithm perturbs the context vectors (this corresponds to perturbing the actions $x \in \mathcal{X}$ in BBO setting). Riquelme et al. (2018) studied a neural greedy algorithm for contextual bandits. Their algorithm constructs the surrogate model by minimizing the following objective $\sum_{i=1}^{n}(y_i - f(x_i, \theta))^2$, and determines the next query point by minimizing the learned surrogate model. While this might look similar to our algorithm, their algorithm differs from ours in two crucial aspects: (a) their algorithm doesn't differentiate noisy and noiseless settings and uses the same surrogate model builder for both, (b) they only update their surrogate model once every 20 steps, and more importantly, they don't train the model from scratch in each iteration. They instead warm-start the training with the previous surrogate model. We note that training the surrogate model from scratch in each iteration, with random initialization, is crucial as it helps us explore the search space better (see Section C). In a recent work, Papalexopoulos et al. (2021) used a neural greedy algorithm to solve constrained, discrete BBO problems. However, they only considered noiseless setting and did not provide any theoretical insights for the algorithm.

In Algorithm 1 we have an exploration phase that lasts for T_e rounds. We note that this is not the same as the exploration phase of classical explore-then-commit (ETC) algorithms (Slivkins, 2019). T_e in our algorithm is independent of T (typically ≤ 10). Whereas, the number of exploration rounds needed in ETC algorithms to achieve non-trivial regret guarantees (*i.e.,* $o(T)$ regret) scales polynomially with T.

Theory. Theoretical analysis of SimpleSMB and PosteriorSMB can be found in Appendix Appendices C and D.1.

3. Experiments

In this section, we present experimental results showing the effectiveness of the proposed BBO algorithms. Due to space constraints, our focus here is on the noiseless setting. Experimental results for the noisy case can be found in the Appendix F.

Experiment Setup. We run our experiments on a set of synthetic functions that are commonly used in blackbox optimization benchmarking and competitions (Loshchilov and Glasmachers, 2015; Hansen et al., 2021). The functions we chose vary in dimensionality, modality, smoothness, and structure (more details about them can be found in Appendix F.3). The budget T for each function was set based on the number of dimensions: functions of higher dimensions were allocated more rounds.

Compared algorithms. We compare our proposed Algorithms (SimpleSMB, PosteriorSMB) with several baselines: *GP-EI* (Jones et al., 1998), *NeuralLinTS* (Xu et al., 2020), *NeuralUCB* (Zhou et al., 2020), *NeuralEnsembles* (Kim et al., 2021). We perform a grid search for each compared algorithm and present the results with the best found hyper-parameters. A further description about the baselines, implementation and hyper-parameter search can be found in Appendix F.

Results. The plots for the minimum value observed over iterations is shown in Fig 2 for the noiseless case. The results for the noisy case can be found in Appendix E. It can be seen that the SimpleSMB has the best performance over all the neural network based techniques, and has similar (if not better) performance as GP-EI. Moreover, while PosteriorSMB is

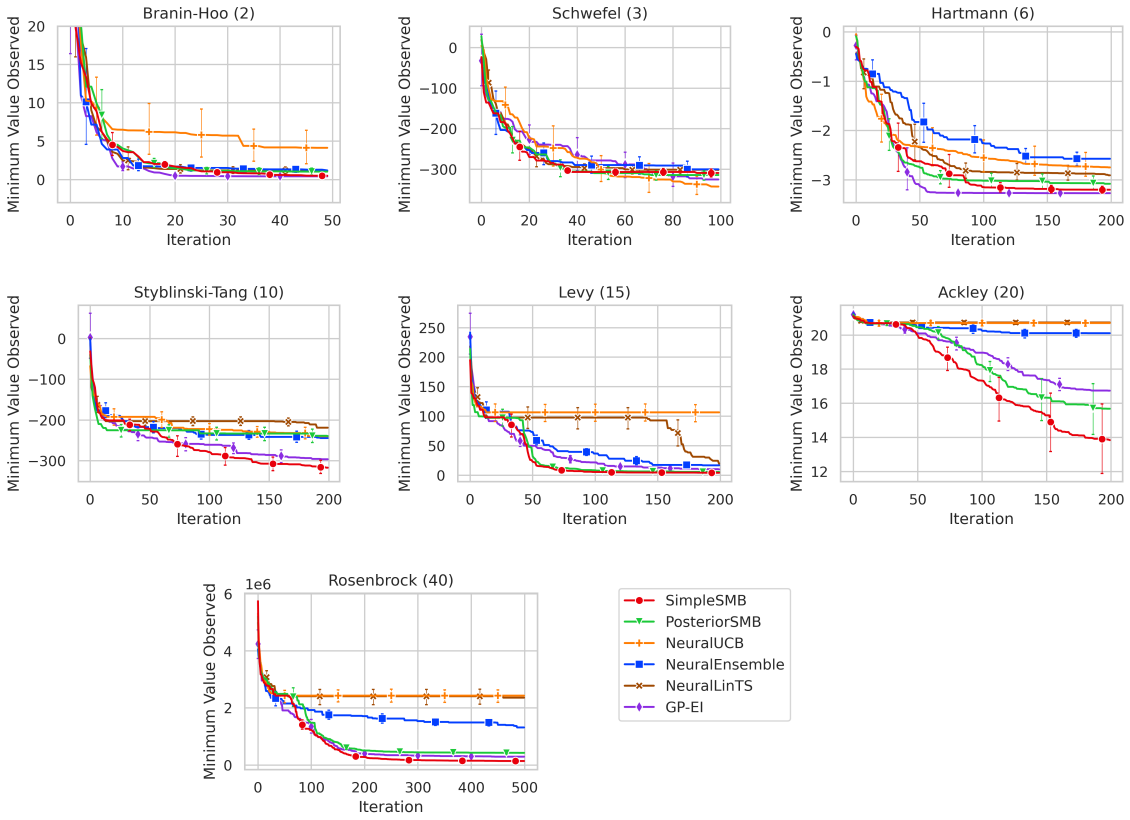


Figure 2: The plots show the minimum value observed over iterations for various baselines on the blackbox functions. The dimensionality of the blackbox functions is provided in parenthesis.

better than all the neural baselines, it has slightly worse performance than SimpleSMB. The experimental results indicate the effectiveness of neural networks in BBO, especially in higher dimensions.

4. Conclusion

In this work, we presented two simple neural-greedy algorithms for global optimization of expensive BB functions. Unlike existing works, our algorithms do not explicitly construct confidence intervals or perform posterior sampling. They instead greedily fit a surrogate model to the observed data and use the learned model as the acquisition function to determine the next query point. Our main approach SimpleSMB adds random perturbations to the target variable, whereas PosteriorSMB adds random perturbations to both the target variable and the output. We connect our algorithms to GPs and Thompson sampling in the infinite-width limit of NNs to derive regret bounds. Lastly, experimental results show that our algorithms have better performance than other neural baselines and achieve similar performance as GP-EI, which is the gold standard for blackbox optimization. Future work and limitations of our approach are discussed in Appendix G

Appendix A. Preliminaries

Blackbox Optimization. In blackbox optimization (BBO), our aim is to minimize a (potentially non-convex) function f^* over an action space $\mathcal{X}$, given only zeroth-order oracle access to the function. That is, the only information about f^* comes via querying the oracle at a point $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and observing $y = f^*(x) + \xi$. Here, ξ is the independent noise, sampled from the Gaussian distribution $\mathcal{N}(0, \sigma_*^2)$. The special case of $\sigma_*^2 = 0$ is called *noiseless* BBO. The performance of a BBO algorithm is measured using one of the following criteria

$$\textbf{Simple Regret:} \quad \min_{t\in[T]} f^*(x_t) - \min_{x\in\mathcal{X}} f^*(x), \quad \textbf{Cumulative Regret:} \quad \sum_{t=1}^{T} f^*(x_t) - \min_{x\in\mathcal{X}} f^*(x),$$

where $x_1, x_2, \ldots, x_T$ are the sequence of points queried by the algorithm.

Gaussian Processes. A GP over $\mathcal{X}$, denoted by $\mathrm{GP}(\mu, \mathcal{K})$, is a collection of random variables $\{f(x)\}_{x\in\mathcal{X}}$ such that the joint distribution of every finite subset $\{f(x_i)\}_{i=1}^n$ of them is multivariate Gaussian with mean $\mathbb{E}[f(x_i)] = \mu(x_i)$ and covariance $\mathrm{Cov}(f(x_i), f(x_j)) = \mathcal{K}(x_i, x_j)$ (Rasmussen, 2003). Here, $\mu, \mathcal{K}$ are the mean and covariance functions of the GP. In this work, we assume $\mu = 0$ for GPs not conditioned on the data.

In BBO, GPs are often used to place a prior distribution over the unknown blackbox function f^*. Suppose, we observe n datapoints $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $y_i = f^*(x_i) + \xi_i$ is the output of the zeroth-order oracle when queried at x_i. Conditioned on $\mathcal{D}$, the posterior distribution over f^* is again a GP with the following mean and covariance functions

$$\mu_n(x) = \mathcal{K}_{x\mathcal{X}_n} \left(\mathcal{K}_{\mathcal{X}_n\mathcal{X}_n} + \sigma^2 I\right)^{-1} \mathcal{Y}_n, \quad \mathcal{K}_n(x, x') = \mathcal{K}_{xx'} - \mathcal{K}_{x\mathcal{X}_n} \left(\mathcal{K}_{\mathcal{X}_n\mathcal{X}_n} + \sigma^2 I\right)^{-1} \mathcal{K}_{\mathcal{X}_n x'},$$

where $\mathcal{X}_n = [x_i]_{i=1}^n, \mathcal{Y}_n = [y_i]_{i=1}^n$. Here, $\mathcal{K}_{x\mathcal{X}_n} \in \mathbb{R}^{1\times n}$ with i^{th} entry given by $\mathcal{K}(x, x_i)$, and $\mathcal{K}_{\mathcal{X}_n\mathcal{X}_n} \in \mathbb{R}^{n\times n}$ with $(i,j)^{th}$ entry given by $\mathcal{K}(x_i, x_j)$.

Acquisition functions play a key role in BBO, as optimizing them helps us decide the next query point. The following acquisition functions are commonly used with GPs, and are known to achieve a good balance between exploration and exploitation: (a) $\alpha_{\mathrm{UCB}}(x; \mathcal{D}) = \mu_n(x) + \nu_n \sqrt{\mathcal{K}_n(x, x)}$, (b) $\alpha_{\mathrm{TS}}(x; \mathcal{D}) = f(x)$, where f is randomly sampled from the posterior $\mathrm{GP}(\mu_n, \nu_n^2 \mathcal{K}_n)$, and (c) $\alpha_{\mathrm{EI}}(x; \mathcal{D}) = \mathbb{E}_{f\sim\mathrm{GP}(\mu,\mathcal{K})}\left[\max(0, y_{\mathrm{best}} - f(x)) | \mathcal{D}\right]$. Here, ν_n is a scale parameter that controls the exploration, and $y_{\mathrm{best}} = \min(\{y_i\})_{i=1}^n$.

Neural Networks. In this work, we study neural network based BBO algorithms. We consider feed-forward networks $f(x, \theta)$ with L hidden layers and d_l neurons in the l^{th} hidden layer. Such a neural network can be defined using the following recurrence relation

$$\alpha^{(l+1)}(x, \theta) = \phi\left(\frac{\sigma_W}{\sqrt{d_l}} W^{(l)} \alpha^{(l)}(x, \theta) + \sigma_b b^{(l)}\right), \quad f(x, \theta) = \frac{\sigma_W}{\sqrt{d_L}} W^{(L)} \alpha^{(L)}(x, \theta) + \sigma_b b^{(L)},$$

with $\alpha^{(0)}(x, \theta) = x$. Here, $W^{(l)} \in \mathbb{R}^{d_{l+1}\times d_l}$, $b^{(l)} \in \mathbb{R}^{d_{l+1}}$ are the weights of layer l, with $d_0 = d$, $d_{L+1} = 1$. ϕ is the elementwise nonlinearity (we set it to `tanh` in our experiments), and the hyper-parameters $\gamma = (\sigma_W, \sigma_b)$ are the weight and bias variances. This particular parameterization of neural network is called Neural Tangent Kernel (NTK) parameterization (Jacot

et al., 2018). In this parameterization, the weights $\theta = \{W^{(\leq L)}, b^{(\leq L)}\}$ are initialized by generating i.i.d samples from $\mathcal{N}(0,1)$. We consider this particular initialization scheme in this work.

Infinite-width limit of NNs. Suppose the weights of a NN are initialized to θ_0 using the random initialization scheme described above. For sufficiently wide networks (*a.k.a.* NTK regime), the outputs $\{f(x,\theta_0)\}_{x\in\mathcal{X}'}$, for any finite set $\mathcal{X}'$, converge to a multivariate Gaussian distribution (Lee et al., 2017). To be precise, such randomly initialized networks correspond to a GP with mean 0 and the following covariance function $\mathcal{K}^{\text{NN}}(x,x') = \lim_{\min(d_1,d_2\ldots d_L)\to\infty} \mathbb{E}\left[f(x,\theta_0)f(x',\theta_0)\right]$.

Now, let's suppose we train the network to minimize the following squared loss over the training dataset $\{(x_i,y_i)\}_{i=1}^n$: $\sum_{i=1}^n (f(x_i,\theta)-y_i)^2$. Suppose we randomly initialize the network at θ_0 and use gradient descent (GD) to minimize the objective. For sufficiently wide networks and small enough step-size, the GD iterates $\{\theta_t\}_{t>0}$ stay close to θ_0 (Lee et al., 2019). Consequently, the NN can be well approximated using the following linear model: $f(x,\theta_t) \approx f(x,\theta_0) + \langle \nabla_\theta f(x,\theta_0), \theta_t - \theta_0\rangle$. Letting $\phi(x) = \nabla_\theta f(x,\theta_0)$, the kernel $\hat{\Theta}(x,x') = \langle \phi(x), \phi(x')\rangle$ is called the empirical NTK kernel. Note that $\hat{\Theta}$ is a random quantity. In the NTK regime, $\hat{\Theta}(x,x')$ converges in probability to a deterministic quantity $\Theta(x,x')$, which is called the NTK kernel (Arora et al., 2019b). As a consequence, GD on wide networks can be viewed as performing kernel regression in the Reproducing Kernel Hilbert Space (RKHS) associated with Θ. For large t, Lee et al. (2019) showed that the neural network $f(\cdot,\theta_t)$ (which is a random function that depends on θ_0), can be viewed as being sampled from a GP with the following mean and covariance functions

$$\mu_n(x) = \Theta_{x\mathcal{X}_n}\Theta^{-1}_{\mathcal{X}_n\mathcal{X}_n}\mathcal{Y}_t, \quad \mathcal{K}_n^{\text{NNGD}}(x,x') = \mathcal{K}^{\text{NN}}(x,x') + \Theta_{x\mathcal{X}_n}\Theta^{-1}_{\mathcal{X}_n\mathcal{X}_n}\mathcal{K}^{\text{NN}}_{\mathcal{X}_n\mathcal{X}_n}\Theta^{-1}_{\mathcal{X}_n\mathcal{X}_n}\Theta_{\mathcal{X}_n x'} - \left(\Theta_{x\mathcal{X}_n}\Theta^{-1}_{\mathcal{X}_n\mathcal{X}_n}\mathcal{K}^{\text{NN}}_{\mathcal{X}_n x'} + h.c.\right),$$

where "+h.c." is the Hermitian conjugate of its preceding term, and $\mathcal{X}_n = [x_i]_{i=1}^n, \mathcal{Y}_n = [y_i]_{i=1}^n$ are the features and response variables in the training dataset. Note that $\mathcal{K}^{\text{NN}}, \Theta$, and $\mathcal{K}_n^{\text{NNGD}}$ all depend on the variance hyper-parameter γ.

Appendix B. Related Work

BBO. Global optimization of expensive blackbox functions is an extremely well studied problem. Numerous techniques have been proposed for this problem over the years. We review some of the relevant literature below.

Structural Assumptions. One category of works impose structural assumptions on the unknown blackbox function f^*. For instance, Filippi et al. (2010); Agrawal and Goyal (2013) assume f^* is a linear function. Kveton et al. (2020) assume f^* can be modeled using a generalized linear model (GLM). Several works assume convexity of f^* (Agarwal et al., 2011; Shamir, 2013; Belloni et al., 2015; Bach and Perchet, 2016).

No Structural Assumptions. The second category of works impose minimal assumptions on f. Kleinberg et al. (2008); Bubeck et al. (2009) impose Lipschitz continuity on f^*. In GPs, it is typically assumed that f^* lies in an RKHS (Srinivas et al., 2009; Chowdhury and Gopalan, 2017). As previously mentioned GPs do not scale well to high-dimensional

problems. Consequently, several works have attempted to speed-up GP inference (Liu et al., 2020; Calandriello et al., 2020)

NN based approaches. A recent line of work assumes f^* is a neural network. While this might look like a structural assumption, it actually is not because NNs have the power to arbitrarily approximate any continuous function. Early works relied on Bayesian neural networks (BNN) to impose a prior over the unknown blackbox function (Springenberg et al., 2016). These techniques relied on exact posterior sampling to compute acquisition functions such as EI. A variety of algorithms have been developed for posterior sampling on BNNs. These include Hamiltonian Monte Carlo (Neal, 2012), stochastic gradient Langevin MCMC (Korattikara Balan et al., 2015), variational inference methods (Graves, 2011). A drawback with these techniques is that they are very complex to implement in practice with a large number of hyper-parameters. In addition, many of these techniques provide conservative estimates of the uncertainty for points that are far from the observed data (Springenberg et al., 2016). Recent works considered wide neural networks and relied on NTK theory to develop UCB and TS algorithms for neural networks (Zhang et al., 2020; Zhou et al., 2020). Another class of approaches have relied on heuristics such as performing LinearUCB, LinearTS using features from a neural network (Snoek et al., 2015; Xu et al., 2020) or relying on ensembles to mimic posterior sampling (Kim et al., 2021; Lakshminarayanan et al., 2017).

Infinite-width limit of Neural Networks. Wide NNs and their infinite-width limits have gained attention of late. Lee et al. (2017); Matthews et al. (2018) showed that wide NNs at initialization are equivalent to GPs. Jacot et al. (2018); Arora et al. (2019b) showed that training wide NNs (with random initialization that leads to zero or small initial outputs) using gradient descent is equivalent to performing kernel ridge regression with NTK kernel. Lee et al. (2019) showed that gradient descent on randomly initialized wide NNs is equivalent to sampling from GPs. NTK theory was extended from feed forward networks to other architectures such as convolutional networks (Arora et al., 2019b), graph neural networks (Du et al., 2019). Arora et al. (2019a) derived generalization bounds for 2 layer wide networks, that rely on NTK kernel. Eldan et al. (2021) derived non-asymptotic rates for the speed at which finite-width NN training approaches the NTK regime.

Appendix C. Theoretical analysis of SimpleSMB in the infinite-width limit

In this section, we study our algorithm in the infinite-width limit of NNs. We show that in each iteration of Algorithm 1, the learned network θ_t is sampled from a GP.

Proposition 1 *Suppose the width of the NNs used in the Algorithm 1 approaches infinity; that is,* $\min(d_1, d_2 \dots d_L) \to \infty$. *Suppose GD with small enough step size is used to optimize the surrogate model objective (*i.e., *line 3 of Algorithm 2). Consider the* $(t+1)^{th}$ *iteration of the algorithm, for any* $t \geq T_e$. *Conditioned on* $\mathcal{D}_t$ *and the past randomness, the surrogate model* $\tilde{f}_{t+1}$ *can be viewed as being randomly sampled from a* GP *with the following mean and*

covariance functions

$$\mu_t(x) = \Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\mathcal{Y}_t,$$
$$\mathcal{K}_t^{NNGD}(x,x') = \nu^2\mathcal{K}^{NN}(x,x') - \nu^2\left(\Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\mathcal{K}^{NN}_{\mathcal{X}_t x'} + h.c.\right)$$
$$+ \nu^2\Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\left(\mathcal{K}^{NN}_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\Theta_{\mathcal{X}_t x'},$$

where $\mathcal{K}^{NN}, \Theta$ *are defined in Section A. Here* $\mathcal{X}_t = \{x_i\}_{i=1}^t$, *and* $\mathcal{Y}_t = \{y_i\}_{i=1}^t$.
Discussion. The variance function $\mathcal{K}_t^{\text{NNGD}}(x,x)$ is usually large at points far away from the observed points $\mathcal{X}_t$. This helps our algorithm explore unobserved areas of the action space.

Instead of retraining the surrogate model from scratch at each iteration, suppose we warm-start Algorithm 2 with the surrogate model from the previous step (as done in Riquelme et al. (2018)). Then the variance of θ_{t+1} conditioned on the history is 0, and the algorithm wouldn't perform any exploration. This would in turn lead to poor performance. This is also evident in the experiments of Riquelme et al. (2018). Next, let's suppose we do not use the regularization term $\|\theta - \theta_0\|_2$ in Algorithm 2. Then a simple calculation shows that the mean function $\mu_t(x)$ is equal to $\Theta_{x\mathcal{X}_t}\Theta^{-1}_{\mathcal{X}_t\mathcal{X}_t}\mathcal{Y}_t$. When evaluated at the observed points $\mathcal{X}_t$, this gives us $\mu_t(\mathcal{X}_t) = \mathcal{Y}_t$. This shows that without the regularization term, the mean function interpolates the noisy data and leads to over-fitted models. Next, let's suppose we do not perturb the targets $\{y_i\}_{i=1}^n$ in Algorithm 2. The covariance function $\mathcal{K}_t^{\text{NNGD}}(x,x')$ in this case is similar to the one in Proposition 1, except for one difference: $\left(\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)$ in the last term is replaced by $\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}_t}$. That is, not perturbing the targets results in narrow confidence bands which can lead to poor exploration. This shows that both forms of regularization are important for the algorithm to achieve good performance.

While our algorithm samples from a GP in each iteration, it doesn't perform Thompson/posterior sampling. TS would require us to sample from a GP with the following mean and covariance functions

$$\mu_t(x) = \Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\mathcal{Y}_t, \quad \mathcal{K}_t(x,x') = \nu^2\Theta_{xx'} - \nu^2\Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\Theta_{\mathcal{X}_t x'}.$$

In section D, we present a modification to our algorithm which lets us sample from the above GP. Nevertheless, despite the lack of correspondence between our algorithm and GP-TS, our algorithm achieves better performance than existing neural network based BBO techniques (see Section 3).

C.1 Proof of Proposition 1

Since we are in the infinite width limit, the iterates of GD with small enough step size stay close to the initialization θ_0 (Lee et al., 2019). So, the following first order approximation is accurate: $f(x,\theta) = f(x,\theta_0) + \langle\phi(x), \theta - \theta_0\rangle$, where $\phi(x) = \nabla_\theta f(x,\theta_0)$. Under this approximation, it suffices to study the following objective

$$\min_\theta \sum_{i=1}^t \left(y_i' - \nu f(x_i,\theta_0) - \nu\left\langle\phi(x_i), \theta - \theta_0\right\rangle\right)^2 + \sigma^2\nu^2\|\theta - \theta_0\|_2^2.$$

This objective can equivalently be written as

$$\min_{\theta} \sum_{i=1}^{t} \left(\frac{y_i}{\nu} + \epsilon_i - f(x_i, \theta_0) - \langle \phi(x_i), \theta - \theta_0 \rangle \right)^2 + \sigma^2 \|\theta - \theta_0\|_2^2.$$

Let $\phi(\mathcal{X}_t) = [\phi(x_1)^T; \phi(x_2)^T \dots]^T$ be the matrix obtained by stacking the vectors $\{\phi(x_i)\}_{i=1}^t$ vertically. Let $f(\mathcal{X}_t, \theta) = [f(x, \theta)]_{x \in \mathcal{X}_t}$, $\mathcal{Y}_t = [y_i]_{i=1}^t$, and $\epsilon = [\epsilon_i]_{i=1}^t$. The minimizer θ_{t+1} of the above objective satisfies the following first order optimality conditions

$$\phi(\mathcal{X}_t)^T \left(\phi(\mathcal{X}_t)(\theta_{t+1} - \theta_0) + f(\mathcal{X}_t, \theta_0) - \epsilon - \frac{\mathcal{Y}_t}{\nu} \right) + \sigma^2(\theta_{t+1} - \theta_0) = 0.$$

Rearranging the terms, we get

$$\theta_{t+1} = \theta_0 + \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I \right)^{-1} \phi(\mathcal{X}_t)^T \left(\frac{\mathcal{Y}_t}{\nu} + \epsilon - f(\mathcal{X}_t, \theta_0) \right).$$

Combining this with the linear approximation above, gives us the following expression for our surrogate model $\tilde{f}_{t+1}(x) = \nu f(x, \theta_{t+1})$

$$\nu f(x, \theta_0) + \left\langle \phi(x), \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I \right)^{-1} \phi(\mathcal{X}_t)^T \left(\mathcal{Y}_t + \nu\epsilon - \nu f(\mathcal{X}_t, \theta_0) \right) \right\rangle.$$

Expectation. Consider any finite set $\mathcal{X}' \subseteq \mathcal{X}$. We now compute the mean function $\mu_t(\mathcal{X}') = \mathbb{E}\left[\tilde{f}_{t+1}(\mathcal{X}') | \mathcal{D}_t, \mathcal{H}_t \right]$, where $\mathcal{H}_t$ denotes the randomness from the past iterations.

$$\begin{aligned}
\mu_t(\mathcal{X}') = \nu \mathbb{E}\left[f(\mathcal{X}', \theta_0) | \mathcal{D}_t, \mathcal{H}_t \right] \\
&+ \nu \phi(\mathcal{X}') \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I \right)^{-1} \phi(\mathcal{X}_t)^T \mathbb{E}\left[\epsilon - f(\mathcal{X}_t, \theta_0) | \mathcal{D}_t, \mathcal{H}_t \right] \\
&+ \phi(\mathcal{X}') \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I \right)^{-1} \phi(\mathcal{X}_t)^T \mathcal{Y}_t.
\end{aligned}$$

Since $f(\cdot, \theta_0) \sim \mathrm{GP}(0, \mathcal{K}^{\mathrm{NN}})$ and ϵ is independent zero-mean Gaussian noise, the first two terms in the RHS above are 0. This shows that

$$\begin{aligned}
\mu_t(\mathcal{X}') &= \phi(\mathcal{X}') \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I \right)^{-1} \phi(\mathcal{X}_t)^T \mathcal{Y}_t \\
&\overset{(a)}{=} \phi(\mathcal{X}') \phi(\mathcal{X}_t)^T \left(\sigma^2 I + \phi(\mathcal{X}_t) \phi(\mathcal{X}_t)^T \right)^{-1} \mathcal{Y}_t \\
&\overset{(b)}{=} \Theta_{\mathcal{X}'\mathcal{X}_t} (\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1} \mathcal{Y}_t,
\end{aligned}$$

where (a) follows from Woodbury matrix identity[1], and (b) follows from the fact that $\langle \phi(x), \phi(x') \rangle$ converges in probability to the NTK kernel $\Theta(x, x')$. This proves the first part of the Proposition.

1. $(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)VA^{-1}$. Here, $A \in \mathbb{R}^{n \times n}, C \in \mathbb{R}^{k \times k}, U, V^T \in \mathbb{R}^{n \times k}$

Covariance. To simplify the notation, we define $M_t = \left(\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1}$. Consider any finite set $\mathcal{X}' \subseteq \mathcal{X}$. The covariance function is given by

$$\begin{aligned}
\mathcal{K}_t^{\text{NNGD}}(\mathcal{X}', \mathcal{X}') &= \mathbb{E}\left[\left(\tilde{f}_{t+1}(\mathcal{X}') - \mu_t(\mathcal{X}')\right)\left(\tilde{f}_{t+1}(\mathcal{X}') - \mu_t(\mathcal{X}')\right)^T |\mathcal{D}_t, \mathcal{H}_t\right] \\
&\overset{(a)}{=} \nu^2\mathbb{E}\Big[\left(f(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f(\mathcal{X}_t, \theta_0)\right) \\
&\qquad\qquad \left(f(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f(\mathcal{X}_t, \theta_0)\right)^T |\mathcal{D}_t, \mathcal{H}_t\Big] \\
&\quad + \sigma^2\nu^2\phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T,
\end{aligned}$$

where (a) follows from the fact that θ_0, ϵ are independent of each other and $\mathbb{E}[\epsilon\epsilon^T] = \sigma^2 I$. Consequently, the cross terms involving θ_0, ϵ are equal to 0. First consider the second term in the RHS above. Using Woodbury matrix identity on M_t, together with the fact that $\langle\phi(x), \phi(x')\rangle$ converges in probability to $\Theta(x, x')$, it can be written as

$$\phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T = \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-2}\Theta_{\mathcal{X}_t\mathcal{X}'}.$$

Next, consider the first term

$$\begin{aligned}
&\mathbb{E}\left[\left(f(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f(\mathcal{X}_t, \theta_0)\right)\left(f(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f(\mathcal{X}_t, \theta_0)\right)^T |\mathcal{D}_t, \mathcal{H}_t\right] \\
&= \mathbb{E}\left[f(\mathcal{X}', \theta_0)f(\mathcal{X}', \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\qquad + \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathbb{E}\left[f(\mathcal{X}_t, \theta_0)f(\mathcal{X}_t, \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right]\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\qquad - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathbb{E}\left[f(\mathcal{X}_t, \theta_0)f(\mathcal{X}', \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\qquad - \mathbb{E}\left[f(\mathcal{X}', \theta_0)f(\mathcal{X}_t, \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right]\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\overset{(a)}{=} \mathcal{K}^{\text{NN}}_{\mathcal{X}'\mathcal{X}'} + \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}_t}\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\qquad - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}'} - \mathcal{K}^{\text{NN}}_{\mathcal{X}'\mathcal{X}_t}\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\overset{(b)}{=} \mathcal{K}^{\text{NN}}_{\mathcal{X}'\mathcal{X}'} + \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}'} \\
&\qquad - \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\mathcal{K}^{\text{NN}}_{\mathcal{X}_t\mathcal{X}'} - \mathcal{K}^{\text{NN}}_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}'},
\end{aligned}$$

where (a) follows from the definition of $\mathcal{K}^{\text{NN}}$ (recall, $\mathcal{K}^{\text{NN}}(x, x') = \mathbb{E}[f(x, \theta_0)f(x', \theta_0)]$), and (b) follows from Woodbury matrix identity. Substituting the above two expressions in the expression for $\mathcal{K}_t^{\text{NNGD}}(\mathcal{X}', \mathcal{X}')$ gives us the required result.

Appendix D. Posterior Corrected Greedy Algorithm

In this section, we present a slight modification to Algorithm 2 that let's us perform posterior sampling in the infinite-width limit. This modification was originally proposed by He et al. (2020) for constructing deep Bayesian ensembles. In this work, we use it for BBO. This modification involves adding a random perturbation $\delta(x) = \left\langle \nabla_\theta f(x, \theta_0), \tilde{\theta}_0 \right\rangle$ to the neural network $f(x, \theta)$. Here, $\theta_0, \tilde{\theta}_0$ are random weights generated using NTK initialization, with

the last layer weights of $\tilde{\theta}_0$ set to 0. Observe that $\delta(x)$ doesn't have any trainable parameters. All the trainable parameters in $f(x,\theta)+\delta(x)$ come from the first term. The rest of the algorithm for building the surrogate model remains the same as Algorithm 2, and involves finding a θ that minimizes the regularized least squares objective (see Algorithm 3 for details).

Algorithm 3 Posterior Corrected Surrogate Model Builder (PosteriorSMB)

Input: Data $\{(x_i, y_i)\}_{i=1}^{t-1}$, initialization weight variance γ, noise variance σ^2, scale parameter ν

1: Add i.i.d noise to targets $y_i' = y_i + \nu\epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$
2: Initialize $\theta_0, \tilde{\theta}_0 \sim$ NTK-Init(γ)
3: Set the last layer parameters in $\tilde{\theta}_0$ to 0.
4: Let $\delta(x) = \left\langle \nabla_\theta f(x,\theta_0), \tilde{\theta}_0 \right\rangle$
5: Solve $\min_\theta \sum_{i=1}^{t-1}(y_i' - \nu f(x_i,\theta) - \nu\delta(x_i))^2 + \sigma^2\nu^2\|\theta-\theta_0\|_2^2$ using GD/SGD with θ_0 as initialization, and obtain θ_t
6: **return** $\nu \times (f(\cdot,\theta_t) + \delta(\cdot))$

D.1 Theoretical analysis of PosteriorSMB in the infinite-width limit

Similar to section C, we first present the following result which connects our algorithm to GPs and Thompson sampling. The proof of this proposition follows from a similar result proved in He et al. (2020).

Proposition 2 *Suppose Algorithm 1 is run with Algorithm 3 as the surrogate model builder. Suppose the width of the NNs used in the algorithm approaches infinity; that is,* $\min(d_1, d2 \dots d_L) \to \infty$. *Suppose GD with small enough step size is used to optimize the surrogate model objective (*i.e., *line 5 of Algorithm 3). Consider the* $(t+1)^{th}$ *iteration of the algorithm, for any* $t \geq T_e$. *Conditioned on* $\mathcal{D}_t$ *and the past randomness, the surrogate model* $\tilde{f}_{t+1}$ *can be viewed as being randomly sampled from a* GP *with the following mean and covariance functions*

$$\mu_t(x) = \Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\mathcal{Y}_t,$$
$$\mathcal{K}_t^{NNGD\text{-}PC}(x,x') = \nu^2\Theta_{xx'} - \nu^2\Theta_{x\mathcal{X}_t}\left(\Theta_{\mathcal{X}_t\mathcal{X}_t} + \sigma^2 I\right)^{-1}\Theta_{\mathcal{X}_t x'}.$$

This shows that our algorithm is equivalent to performing GP-TS with NTK kernel, in the infinite-width limit. We rely on this equivalence to derive regret bounds of our algorithm (Theorem 3). For the purpose of the theorem, we let the scale parameter ν vary with iteration, and let ν_t denote the scale parameter at iteration t. We assume that $T_e = 0$; that is, there is no exploration phase in the algorithm. If $T_e \neq 0$, our regret would have a $O(T_e)$ additive term. Next, we assume the NTK kernel Θ is bounded and satisfies $\sup_{x\in\mathcal{X}}\|\Theta(x,x)\| \leq 1$. This assumption is not very restrictive. If instead, $\sup_{x\in\mathcal{X}}\|\Theta(x,x)\| \leq c$, for some $c > 1$, our regret bounds would increase by a factor of c. Finally, we let $I_t = \max_{\mathcal{A}\subset\mathcal{X},|\mathcal{A}|=t} I(y_\mathcal{A}; f_\mathcal{A})$ be the information gain between $f_A = [f(x)]_{x\in\mathcal{A}}$, and $y_\mathcal{A} = f_\mathcal{A} + \xi_\mathcal{A}$, where $f \sim \mathrm{GP}(0,\Theta), \xi_\mathcal{A} \sim \mathcal{N}(0,\sigma^2\nu_t^2 I)$.

Theorem 3 *Consider the setting of Proposition 2. Let $\mathcal{X} \subseteq [0,r]^d$ be a compact and convex set, and let γ be the initialization variance hyper-parameter used in Algorithm 1. Finally, let $\mathcal{H}$ be the RKHS associated with the NTK kernel Θ. Suppose the true blackbox function f^* is Lipschitz and satisfies $\|f^*\|_{\mathcal{H}} \leq B$. Suppose Algorithm 1 is run with the following hyper-parameters: $\sigma^2 = 1 + \frac{2}{T}$, $\nu_t = B + \sigma_* \sqrt{2\left(I_{t-1} + 1 + \log(2/\delta)\right)}$. Then with probability at least $1-\delta$, the cumulative regret of our algorithm is upper bounded by $O\left(\sqrt{(I_T + \log(2/\delta))\, d \log(BdT)}\left(\sqrt{TI_T} + B\sqrt{T\log(2/\delta)}\right)\right)$.*

The proof of this Theorem can be found in the Appendix and relies on similar proof techniques as in Chowdhury and Gopalan (2017). Notice the regret bound depends on information gain I_T. This quantity can be bounded in terms of the eigen-spectrum of the NTK kernel Θ (Vakili et al., 2021). Let $\{\lambda_1 \geq \lambda_2 \geq \dots\}$ be the eigen-spectrum of Θ w.r.t uniform measure over $\mathcal{X}$. Then I_T depends on the tail function $B(n) = \sum_{t>n} \lambda_t$. Recent works have studied the tail function of NTK kernel for 2-layer networks (Cao et al., 2019; Geifman et al., 2020). These results can be used to derive concrete regret bounds for Algorithm 1.

One can derive non-asymptotic versions of Theorem 3 using results of Arora et al. (2019b); Eldan et al. (2021) which characterize the rate at which wide NN training converges to the infinite-width limit. Recent works have extended NTK theory to other architectures such as convolution and graph neural networks. These results can be used to derive regret bounds of our algorithm in the infinite-width limit of CNNs, GNNs.

Remark 4 *For NTK kernel of a 1-hidden layer network, one can rely on the results of Geifman et al. (2020) to show that $I_T = O(T^{1-d^{-1}})$. Plugging this into the bound of Theorem 3, we obtain a regret bound of $O(T^{3/2-d^{-1}})$ which is vacuous. We note that the regret bounds of Zhang et al. (2020); Zhou et al. (2020) for NeuralUCB, NeuralTS also face this issue. Deriving a TS style algorithm that achieves non-vacuous regret bound in this setting is still an open problem.*

D.2 Proof of Proposition 2

The proof of the Proposition relies on similar arguments as in the proof of Proposition 1. Since we are in the infinite width limit, the iterates of GD with small enough step size stay close to the initialization θ_0. So, the following first order approximation is accurate: $f(x,\theta) = f(x,\theta_0) + \langle \phi(x), \theta - \theta_0 \rangle$, where $\phi(x) = \nabla_\theta f(x,\theta_0)$. Under this approximation, it suffices to study the following objective

$$\min_{\theta} \sum_{i=1}^{t} \left(y_i' - \nu f(x_i,\theta_0) - \nu\delta(x_i) - \nu \left\langle \phi(x_i), \theta - \theta_0 \right\rangle\right)^2 + \sigma^2\nu^2 \|\theta - \theta_0\|_2^2.$$

To simplify the notation, we define $f^{\text{pc}}(x,\theta_0) = f(x,\theta_0) + \delta(x)$. Using similar analysis as in Proposition 1, we get the following expression for our surrogate model $\tilde{f}_{t+1}(x) = \nu f(x,\theta_{t+1}) + \nu\delta(x)$

$$\nu f^{\text{pc}}(x,\theta_0) + \left\langle \phi(x), \left(\phi(\mathcal{X}_t)^T \phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1} \phi(\mathcal{X}_t)^T \left(\mathcal{Y}_t + \nu\epsilon - \nu f^{\text{pc}}(\mathcal{X}_t,\theta_0)\right)\right\rangle.$$

The key result we now use is that the function $f^{\text{pc}}(\cdot,\theta_0)$ can be viewed as being sampled from $\text{GP}(0,\Theta)$ (He et al., 2020). In contrast, $f(\cdot,\theta_0)$ can be viewed as being sampled from

$\mathrm{GP}(0, \mathcal{K}^{\mathrm{NN}})$. The posterior correction term $\delta(x)$ essentially changes the covariance function from $\mathcal{K}^{\mathrm{NN}}$ to Θ.

Expectation. Consider any finite set $\mathcal{X}' \subseteq \mathcal{X}$. We now compute the mean function $\mu_t(\mathcal{X}') = \mathbb{E}\left[\tilde{f}_{t+1}(\mathcal{X}')|\mathcal{D}_t, \mathcal{H}_t\right]$, where $\mathcal{H}_t$ denotes the randomness from the past iterations.

$$\begin{aligned}
\mu_t(\mathcal{X}') &= \nu\mathbb{E}\left[f^{\mathrm{pc}}(\mathcal{X}', \theta_0)|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\quad + \nu\phi(\mathcal{X}')\left(\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1}\phi(\mathcal{X}_t)^T\mathbb{E}\left[\epsilon - f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\quad + \phi(\mathcal{X}')\left(\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1}\phi(\mathcal{X}_t)^T\mathcal{Y}_t.
\end{aligned}$$

Since $f^{\mathrm{pc}}(\cdot, \theta_0) \sim \mathrm{GP}(0, \Theta)$ and ϵ is independent zero-mean Gaussian noise, the first two terms in the RHS above are 0. This shows that

$$\begin{aligned}
\mu_t(\mathcal{X}') &= \phi(\mathcal{X}')\left(\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1}\phi(\mathcal{X}_t)^T\mathcal{Y}_t \\
&\overset{(a)}{=} \phi(\mathcal{X}')\phi(\mathcal{X}_t)^T\left(\sigma^2 I + \phi(\mathcal{X}_t)\phi(\mathcal{X}_t)^T\right)^{-1}\mathcal{Y}_t \\
&\overset{(b)}{=} \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\mathcal{Y}_t,
\end{aligned}$$

where (a) follows from Woodbury matrix identity, and (b) follows from the fact that $\langle\phi(x), \phi(x')\rangle$ converges in probability to the NTK kernel $\Theta(x, x')$. This proves the first part of the Proposition.

Covariance. Using similar arguments as in Proposition 1, we get the following expression for the covariance function

$$\begin{aligned}
\mathcal{K}_t^{\text{NNGD-PC}}(\mathcal{X}', \mathcal{X}') = \nu^2\mathbb{E}\Big[&\left(f^{\mathrm{pc}}(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)\right) \\
&\left(f^{\mathrm{pc}}(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)\right)^T |\mathcal{D}_t, \mathcal{H}_t\Big] \\
&+ \sigma^2\nu^2\phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T,
\end{aligned}$$

where $M_t = \left(\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t) + \sigma^2 I\right)^{-1}$. Consider the first term in the RHS above

$$\begin{aligned}
&\mathbb{E}\big[\left(f^{\mathrm{pc}}(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)\right) \\
&\qquad \left(f^{\mathrm{pc}}(\mathcal{X}', \theta_0) - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)\right)^T |\mathcal{D}_t, \mathcal{H}_t\big] \\
&= \mathbb{E}\left[f^{\mathrm{pc}}(\mathcal{X}', \theta_0)f^{\mathrm{pc}}(\mathcal{X}', \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\qquad + \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathbb{E}\left[f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)^T|\mathcal{D}_t, \mathcal{H}_t\right]\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\qquad - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\mathbb{E}\left[f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)f^{\mathrm{pc}}(\mathcal{X}', \theta_0)|\mathcal{D}_t, \mathcal{H}_t\right] \\
&\qquad - \mathbb{E}\left[f^{\mathrm{pc}}(\mathcal{X}', \theta_0)f^{\mathrm{pc}}(\mathcal{X}_t, \theta_0)|\mathcal{D}_t, \mathcal{H}_t\right]\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\overset{(a)}{=} \Theta_{\mathcal{X}'\mathcal{X}'} + \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\Theta_{\mathcal{X}_t\mathcal{X}_t}\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\qquad - \phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\Theta_{\mathcal{X}_t\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}_t}\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T \\
&\overset{(b)}{=} \Theta_{\mathcal{X}'\mathcal{X}'} + \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}'} \\
&\qquad - \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-1}\Theta_{\mathcal{X}_t\mathcal{X}'},
\end{aligned}$$

where (a) follows from the fact that $f^{\text{pc}}(\cdot, \theta_0)$ is sampled from $\text{GP}(0, \Theta)$, and (b) follows from Woodbury matrix identity. Next, consider the second term. Using similar arguments as in Proposition 1, it can be rewritten as

$$\phi(\mathcal{X}')M_t\phi(\mathcal{X}_t)^T\phi(\mathcal{X}_t)M_t\phi(\mathcal{X}')^T = \Theta_{\mathcal{X}'\mathcal{X}_t}(\sigma^2 I + \Theta_{\mathcal{X}_t\mathcal{X}_t})^{-2}\Theta_{\mathcal{X}_t\mathcal{X}'}.$$

Substituting the above two expressions in the expression for $\mathcal{K}_t^{\text{NNGD-PC}}(\mathcal{X}', \mathcal{X}')$ gives us the required result.

D.3 Proof of Theorem 3

The proof follows from the regret bound of GP-TS derived by Chowdhury and Gopalan (2017) (similar proof techniques have been used to derive regret bounds of TS in various contexts including linear, generalized linear models (Kveton et al., 2020)). We provide a high level argument here for the sake of completeness. For technical reasons, we consider a discretization of $\mathcal{X}$, instead of directly working with $\mathcal{X}$ (*i.e.,* we replace the domain $\mathcal{X}$ with its discretized version in the algorithm). In particular, in the t^{th} iteration of the algorithm, we consider the discretization $\mathcal{X}^{(t)} \subset \mathcal{X}$ which satisfies the following property: $\forall x \in \mathcal{X},\ |f^*(x) - f^*([x]_t)| \leq \frac{1}{t^2}$, where $[x]_t$ is the closest point to x in $\mathcal{X}^{(t)}$. Such a discretization can be achieved because of the fact that f^* is Lipschitz continuous.

The instantaneous regret at any iteration t is given by

$$f^*(x^*) - f^*(x_t) = f^*(x^*) - f^*([x^*]_t) + f^*([x^*]_t) - f^*(x_t) \leq \frac{1}{t^2} + \Delta_t(x_t),$$

where $[x^*]_t$ is the point closest to x^* in the discretization $\mathcal{X}^{(t)}$, and $\Delta_t(x_t) = f^*([x^*]_t) - f^*(x_t)$. To bound the regret, it suffices to bound $\sum_{t=1}^T \Delta_t(x_t)$. To this end, we partition the action space $\mathcal{X}^{(t)}$ into two sets namely, saturated and unsaturated sets. The saturated set consists of actions which satisfy the following condition: $\Delta_t(x) \geq c_t\sigma_{t-1}(x)$, where $\sigma_t(x) = \sqrt{\mathcal{K}_t^{\text{NNGD-PC}}(x, x)}$ (intuitively this consists of actions that are clearly sub-optimal). The unsaturated sets consist of points which satisfy the following condition: $\Delta_t(x) < c_t\sigma_{t-1}(x)$. The key technical part of the proof involves showing that the probability of playing a saturated action is small. Once we have this result, the regret of the algorithm can be bounded as $O(\sum_{t=1}^T c_t\sigma_{t-1}(x_t))$. As proved by Chowdhury and Gopalan (2017), this quantity is upper bounded by $O(\sqrt{I_T T})$, which is our desired regret bound.

Appendix E. Experiments with noisy oracles

We also perform experiments with noisy versions of the blackbox functions. For all the points sampled by the algorithm, we consider the true value of the point (in contrast to the noisy value observed by the algorithm), and plot the minimum observed true value over iterations in Figure 3. We notice similar trends as the noiseless plots shown in Figure 2, with the notable exception of the Ackley function. We observe that neither of the methods are able to reliably find the optimum of the Ackley function within the specified budget. We hypothesize that this is due to the fact that the Ackley function has a sharp global minima, making it even harder to find in the presence of noise.

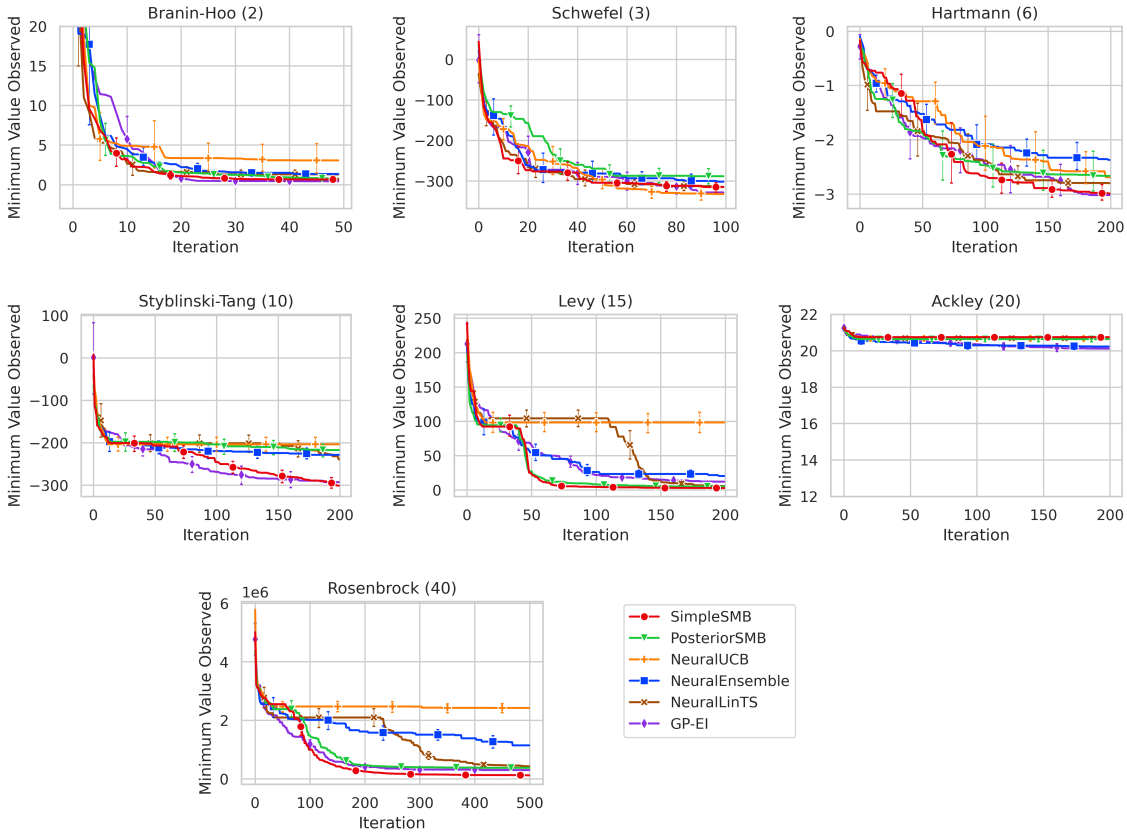


Figure 3: The plots show the minimum *true* value observed over iterations for all the compared methods on noisy version of the blackbox functions. The *true* value denotes the noise-free value of the black box function. The dimensionality of the blackbox functions is provided in parenthesis. The confidence intervals are based on 10 independent runs. While most of the plots follow similar trends as the noiseless plots in Figure 2, we observe that none of the methods are able to reliably minimize the Ackley function.

Appendix F. Additional Experimental Details

F.1 Further details about the baselines

We consider the following baselines in our experiments: (a) **GP-EI**: this performs GP optimization with EI as the acquisition function. We chose EI over UCB and TS because EI has fewer hyper-parameters and is known to achieve similar performance as the other two (Snoek et al., 2012). We used squared-exponential kernel in our experiments, and relied on GPflow for the implementation (Matthews et al., 2017). GPflow automatically sets the kernel hyper-parameters using maximum likelihood estimation, (b) **NeuralLinTS (Xu et al., 2020)**: this baseline performs linear TS on top of the feature representations of NN. It has two hyper-parameters: variance of the prior (diagonal) covariance matrix γ and the observation noise variance σ^2. We do a grid search for γ over $\{10, 100, 1000\}$ and for σ^2 over $\{0.001, 0.01, 0.1\}$, (c) **NeuralUCB (Zhou et al., 2020)**: this baseline constructs confidence bands that are motivated from NTK theory. It has two hyper-parameters: γ used to scale the size of the confidence interval, and λ a regularization parameter. We do grid search for γ over $\{0.01, 0.1\}$, and λ over $\{10^{-5}, \cdots, 10^{-2}\}$, (d) **NeuralEnsembles (Kim et al., 2021)**: this baseline builds an ensemble of m neural networks with identical architectures but different random initializations. This ensemble acts as a surrogate model. The different predictions of networks in the ensemble can be interpreted as samples from the posterior distribution. We use EI as the acquisition function over the predictions from the ensembles. Similar to Kim et al. (2021) at each iteration, we warm-start the networks at their previous values and update the model using GD. In our experiments, we set $m = 10$ and train the model to convergence at each iteration. For NeuralLinTS, NeuralEnsembles, we use a 1-hidden layer networks of widths 500 and 1000 respectively, as surrogate models. We use a smaller width for NeuralLinTS, since it requires inversion of a $d_f \times d_f$ matrix, where d_f is the dimension of the feature extracted from the neural network. For NeuralUCB, we can not use wide networks as it involves inversion of a $p \times p$ matrix, where p is the number of parameters in the NN. This can lead to computational and memory overflow issues. So we use smaller but deeper models. In particular, we use a 2-hidden layer network of width 20 as surrogate model. More details about our implementation can be found in the Appendix. Finally, we note that we neither implement NeuralTS (Zhang et al., 2020) as it is geared towards finite search spaces, nor NeuralLinUCB (Xu et al., 2020) as it has similar performance as NeuralLinTS.

F.2 Implementation details

Exploration budget. The exploration budget T_e was set to $5d$ where d is the input dimension, we also clip the exploration budget to lie between 2.5% and 7.5% of the total budget, so as to not exhaust most of the budget on exploration. All experiments in this paper are repeated 10 times with different seeds, and the average and standard deviation of the results are reported.

Neural network implementation details. We primarily use 1-hidden layer neural architectures with a *large* number (> 1000) of hidden nodes, with the exception of neural UCB, for which we use a 2-layer and 20-wide neural networks as described in (Zhou et al., 2020). The weights of the neural network are initialized with independent samples from a normal distribution $\mathcal{N}(0, \gamma^2/\text{layer input size})$. We initialize the biases of the hidden layers

with independent samples from a normal distribution $\mathcal{N}(0, \gamma^2)$. We initialize the bias of the final layer to 0, as is standard in neural network training. As described in the main text, γ is a hyper-parameter that we tune for our proposed approaches SimpleSMBand PosteriorSMB. We set $\gamma = 1$ for the baselines (as done in the original papers).

We implemented the above in python using Jax (Bradbury et al., 2018). The networks (*i.e.,* surrogate models) were trained using the Adam optimizer (Kingma and Ba, 2014) with a fixed learning rate of 0.001. We did not notice any practical difference from SGD other than faster convergence. All our experiments were run on a server with a 40 core Intel Xeon Silver 4210 CPU @ 2.20GHz and 187 GB memory.

Acquisition optimization. Optimizing the acquisition function is an essential step in all of the considered approaches. The acquisition function is built from the observed data. It takes as input a point from the input domain and produces a scalar value denoting the informativeness of the point. The acquisition function is optimized to select the most informative point to query next. Examples of acquisition functions include the upper confidence bound (Srinivas et al., 2009) and expected improvement (Jones et al., 1998).

Acquisition functions over continuous domains are typically differentiable and amenable to gradient based optimizers. We used standard gradient descent to optimize the acquisition function. We initialize the starting point with a uniform random sample from the input domain and perform 500 gradient descent steps with a fixed learning rate of 0.01. We repeat this process 10 times and return the best point found.

Hyper-parameters and acquisition functions. For each of these methods, we fix the number of hidden layers and the width. The rest of the tunable hyper-parameters are described below. Each hyper-parameter is tuned w.r.t. the cumulative regret over the last $T/2$ steps of the algorithm, where T is the total budget.

SimpleSMB *and* PosteriorSMB*:* We set $\nu = 1, \sigma^2 = 0$, and set $\sigma_W = \sigma_b$ within γ. The only tuned hyper-parameter is the initialization variance parameter γ. We tune γ in the range $[0.5, 5.0]$. For our methods, we use 1-hidden layer networks of fixed width 5000 as surrogate models. The effect of the initialization variance on the performance of SimpleSMBis summarized in Figures 4 and 5 for the noiseless and noisy cases respectively. The aquisition function in this case is the surrogate function returned from Algorithms 2 and 3.

NeuralUCB: The hyper-parameters for this method include γ, the scale of the confidence interval (Algorithm 1 in (Zhou et al., 2020)) and λ the regularization parameter while training the surrogate model on observations (Algorithm 2 in (Zhou et al., 2020)). We tune the hyper-parameters over the sets $\gamma \in \{0.01, 0.1, 1.0\}$ and $\lambda \in \{10^{-e}\}_{e=2}^{5}$.

NeuralEnsemble: This approach requires no hyper-parameters. We train 10 independent neural networks to convergence in each iteration. The neural networks are warm started with the weights from the previous iteration.

NeuralLinTS: Following Riquelme et al. (2018), in this method we first train the neural network on the observed data, extract the final hidden layer activations, and use them for Bayesian linear regression. We assume a diagonal prior covariance $\gamma\mathbb{I}$ on the linear regression

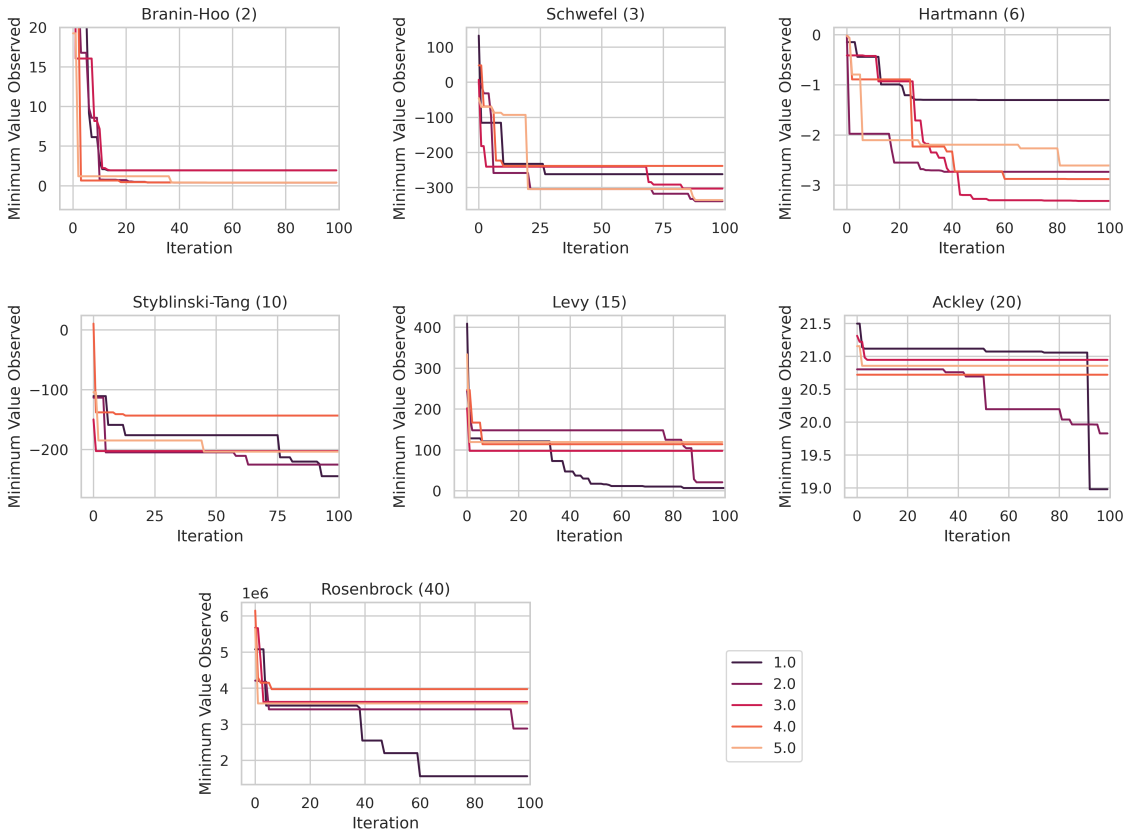


Figure 4: Hyper-parameter search plots for SimpleSMB: The plots show the minimum value observed over iterations for various initialization variances γ in SimpleSMB, for noiseless oracles. The dimensionality of the blackbox functions is provided in parenthesis. These runs were not averaged over multiple runs due to lack of computational resources.

weights, and a Gaussian zero mean noise with variance σ^2. We tune the hyper-parameters over the sets $\gamma \in \{10, 100, 1000\}$ and $\sigma^2 \in \{0.001, 0.01, 0.1\}$. We vary γ over relatively large values in order to not regularize the regression weights too much.

GP-EI: For this, we estimate the GP kernel parameters from the data by maximizing the marginal likelihood of the observations (Rasmussen, 2003). We use the expected improvement acquisition function (Jones et al., 1998) with the estimated kernel parameters to select the next observation point.

F.3 Benchmark black-box functions

We use a set of commonly used benchmark synthetic black-box functions to evaluate our algorithm. Low-dimensional visualizations of the function are shown in Figure 6. We refer the reader to the excellent repository of benchmarking functions by Surjanovic and Bingham (2013) available at `https://www.sfu.ca/~ssurjano/optimization.html`, for the exact expression of these functions.

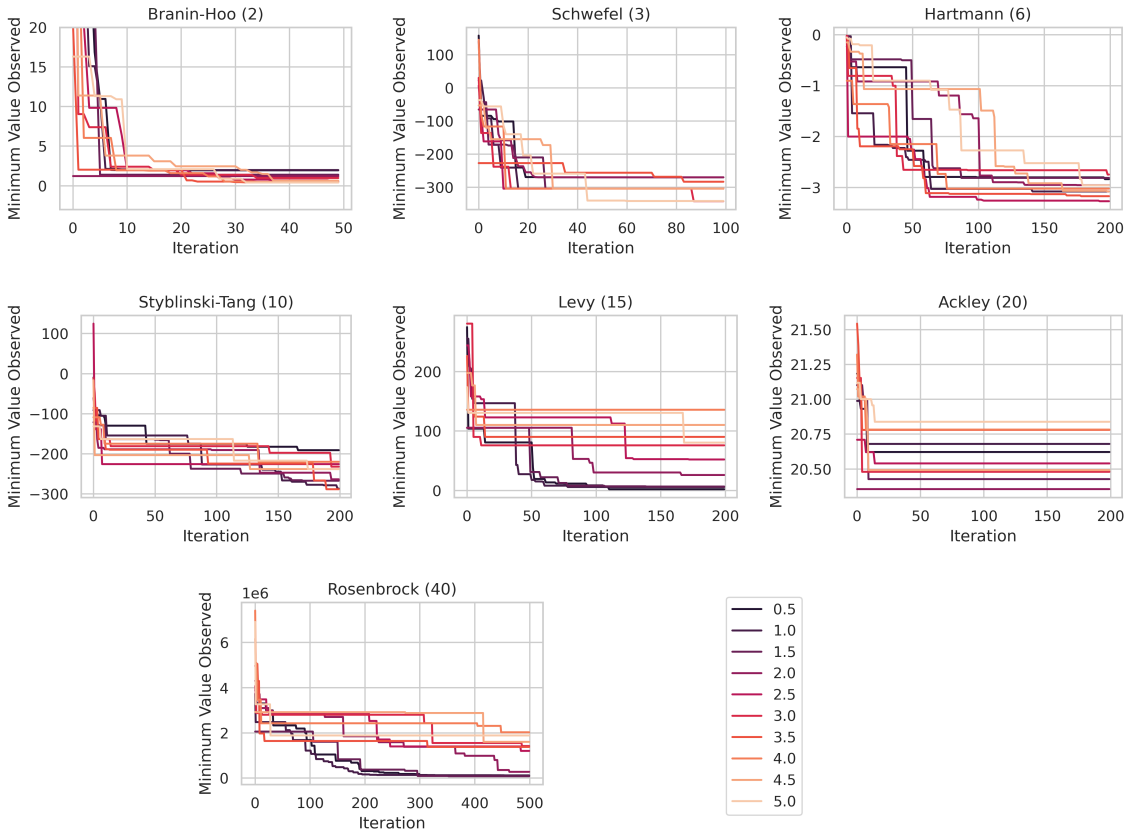
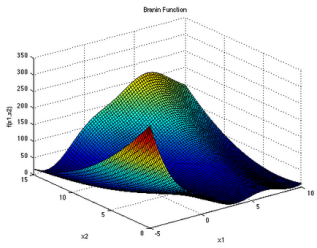


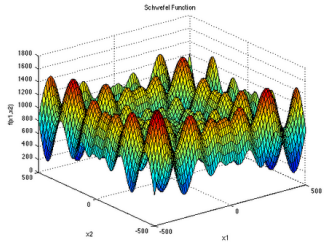
Figure 5: Noisy hyper-parameter search plots for SimpleSMB. The plots show the minimum *true* value observed over iterations for various initialization variances γ in SimpleSMB, for the noisy versions of the blackbox functions. The *true* value denotes the noise-free value of the black box function. The dimensionality of the blackbox functions is provided in parenthesis. These runs were not averaged over multiple runs due to lack of computational resources.

Appendix G. Future Work

In the future, we aim to derive regret bounds for SimpleSMB and understand the settings in which it achieves better performance than PosteriorSMB. There have been criticisms of the NTK regime as not capturing the behavior of regular neural networks (Ghorbani et al., 2021). An interesting future direction would be to study our algorithms in the non-NTK regime and derive their regret bounds. Another important problem is to propose an approach for automatically tuning γ, the initialization variance of the neural network parameters. We noticed a significant impact of γ on the performance of our algorithm. In future, we aim to rely on the connection between our algorithms and GPs to automatically set this hyper-parameter using maximum likelihood estimation.



(a) Branin-Hoo (2)



(b) Schwefel (3)

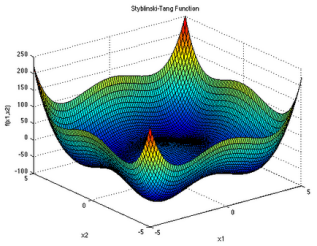
$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right), \text{ where}$$

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T$$

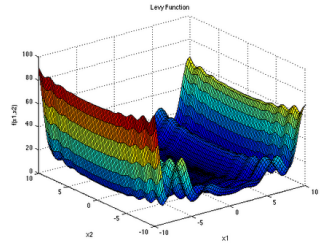
$$\mathbf{A} = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

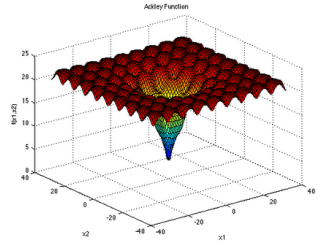
(c) Hartmann (6)



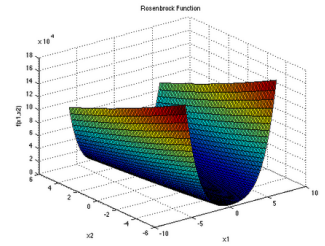
(d) Styblinski-Tang (10)



(e) Levy (15)



(f) Ackley (20)



(g) Rosenbrock (40)

Figure 6: The figures show 2-dimensional versions of the respective benchmark functions wherever available. Hartmann 6 does not have a 2-dimensional version, and hence the equation is displayed instead. All figures are credited to Surjanovic and Bingham (2013).

References

Alekh Agarwal, Dean P Foster, Daniel J Hsu, Sham M Kakade, and Alexander Rakhlin. Stochastic convex optimization with bandit feedback. In *Advances in Neural Information Processing Systems*, pages 1035–1043, 2011.

Rajeev Agrawal. Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078, 1995.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019a.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019b.

Francis Bach and Vianney Perchet. Highly-smooth zero-th order online optimization. In *Conference on Learning Theory*, pages 257–283. PMLR, 2016.

Alexandre Belloni, Tengyuan Liang, Hariharan Narayanan, and Alexander Rakhlin. Escaping the local minima via simulated annealing: Optimization of approximately convex functions, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Sébastien Bubeck, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos. Online optimization in x-armed bandits. In *Advances in Neural Information Processing Systems*, pages 201–208, 2009.

Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Near-linear time gaussian process optimization with adaptive batching and resparsification. In *International Conference on Machine Learning*, pages 1295–1305. PMLR, 2020.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2019.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Ronen Eldan, Dan Mikulincer, and Tselil Schramm. Non-asymptotic approximations of neural networks by gaussian processes. In *Conference on Learning Theory*, pages 1754–1775. PMLR, 2021.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.

Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594, 2010.

Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, and Basri Ronen. On the similarity between the laplace and neural tangent kernels. *Advances in Neural Information Processing Systems*, 33:1451–1461, 2020.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *The Annals of Statistics*, 49(2):1029–1054, 2021.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:1010–1022, 2020.

Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 34(3):441–466, 2006.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

Sampath Kannan, Jamie H Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. *Advances in neural information processing systems*, 31, 2018.

Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Scalable and flexible deep bayesian optimization with auxiliary information for scientific problems. *arXiv preprint arXiv:2104.11667*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690, 2008.

Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. *Advances in Neural Information Processing Systems*, 28, 2015.

Branislav Kveton, Manzil Zaheer, Csaba Szepesvari, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. Randomized exploration in generalized linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 2066–2076. PMLR, 2020.

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

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

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

I Loshchilov and T Glasmachers. Black-box optimization competition (bbcomp), 2015.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18 (40):1–6, apr 2017. URL `http://jmlr.org/papers/v18/16-537.html`.

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

Jonas Mockus. The bayesian approach to global optimization. In *System Modeling and Optimization*, pages 473–481. Springer, 1982.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

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

Theodore Papalexopoulos, Christian Tjandraatmadja, Ross Anderson, Juan Pablo Vielma, and David Belanger. Constrained discrete black-box optimization using mixed-integer programming. *arXiv preprint arXiv:2110.09569*, 2021.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 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.

Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory*, pages 3–24. PMLR, 2013.

Aleksandrs Slivkins. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

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

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

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

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: test functions and datasets. *Simon Fraser University, Burnaby, BC, Canada*, 13:2015, 2013. URL `https://www.sfu.ca/~ssurjano/hart6.html`.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. Combo: An efficient bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2021.

Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. *Advances in Neural Information Processing Systems*, 30, 2017.

Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780*, 2020.

Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.