Sample-Then-Optimize Batch Neural Thompson Sampling

Zhongxiang Dai[†], **Yao Shu**[†], **Bryan Kian Hsiang Low**[†], **Patrick Jaillet**[§]
Dept. of Computer Science, National University of Singapore, Republic of Singapore[†]
Dept. of Electrical Engineering and Computer Science, MIT, USA[§]
{daizhongxiang,shuyao,lowkh}@comp.nus.edu.sg[†],jaillet@mit.edu[§]

Abstract

Bayesian optimization (BO), which uses a *Gaussian process* (GP) as a surrogate to model its objective function, is popular for black-box optimization. However, due to the limitations of GPs, BO underperforms in some problems such as those with categorical, high-dimensional or image inputs. To this end, recent works have used the highly expressive *neural networks* (NNs) as the surrogate model and derived theoretical guarantees using the theory of *neural tangent kernel* (NTK). However, these works suffer from the limitations of the requirement to invert an extremely large parameter matrix and the restriction to the sequential (rather than batch) setting. To overcome these limitations, we introduce two algorithms based on the *Thompson sampling* (TS) policy named *Sample-Then-Optimize Batch Neural TS* (STO-BNTS) and STO-BNTS-Linear. To choose an input query, we only need to train an NN (resp. a linear model) and then choose the query by maximizing the trained NN (resp. linear model), which is equivalently sampled from the GP posterior with the NTK as the kernel function. As a result, our algorithms sidestep the need to invert the large parameter matrix yet still preserve the validity of the TS policy. Next, we derive regret upper bounds for our algorithms with batch evaluations, and use insights from batch BO and NTK to show that they are *asymptotically no-regret* under certain conditions. Finally, we verify their empirical effectiveness using practical AutoML and reinforcement learning experiments.

1 Introduction

Bayesian optimization (BO), also called *Gaussian process* (GP) bandits, has become a celebrated method for optimizing expensive-to-compute black-box functions, primarily thanks to its practical sample efficiency and theoretically guaranteed convergence [11, 14, 21, 51]. However, there are important problem settings where BO either underperforms or is not even applicable without sophisticated modifications, such as problems with categorical [18], high-dimensional [30], or images inputs. These issues have arisen mainly because GPs (i.e., the surrogate used by BO to model the objective function) are not able to effectively model these types of input space, which therefore calls for the use of alternative surrogate models in BO. To this end, *neural networks* (NNs) serve as a natural candidate owing to their remarkable expressivity [37]. NNs have repeatedly proven their ability to model extremely complicated real-world functions such as those involving categorical, high-dimensional or image inputs, whereas the development of GPs to effectively model these functions still represents active areas of research. In this regard, [69] have adopted NNs as the surrogate model in contextual bandit problems and employed the theoretical framework of *neural tangent kernel* (NTK) [27] to construct a principled algorithm following the well-known policy of *upper confidence bound* (UCB), hence introducing the *Neural UCB* algorithm. More recently, [66] have extended Neural UCB to

Correspondence to: Yao Shu <shuyao@comp.nus.edu.sg>.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

follow the *Thompson sampling* (TS) policy and proposed the *Neural TS* algorithm. Both Neural UCB and Neural TS are equipped with upper bounds on their cumulative regret and perform competitively in real-world contextual bandit experiments.

However, Neural UCB and Neural TS are still faced with several important limitations that may hinder their practical applications. Firstly, these algorithms suffer from the requirement to invert a $p \quad p$ matrix in every iteration, in which p is the number of parameters of the NN surrogate and is usually extremely large since the theory of NTK requires severe overparameterization. In practice, a diagonal approximation is used to avoid the need to invert such a large $p \quad p$ matrix [66, 69], which introduces approximation errors in their practical deployment that are unaccounted for in the theoretical analysis, hence causing a disparity between theory and practice. Secondly, the theoretical analyses of these algorithms are restricted to the sequential setting and are hence not applicable in the batch setting where an entire batch of inputs are selected for querying.[1] To overcome these two limitations, we introduce two algorithms based on the TS policy named *Sample-Then-Optimize Batch Neural Thompson Sampling* (STO-BNTS) and STO-BNTS-Linear, both of which *(1)* sidestep the need to invert the $p \quad p$ matrix and hence close the gap between theory and practice, and *(2)* naturally support batch evaluations while preserving the theoretical guarantees.

To avoid the inversion of the $p \quad p$ matrix while still ensuring the validity of the TS policy, we draw inspirations from sample-then-optimize optimization [42] and Bayesian deep ensembles [24] to efficiently sample functions from the GP posterior (with the NTK as the kernel function) without inverting the $p \quad p$ matrix. Specifically, to choose an input query within a batch, our STO-BNTS (resp. STO-BNTS-Linear) only needs to use the current observation history to train an *NN surrogate* (resp. a linear model defined w.r.t. the random features embedding of the NTK associated with an *NN surrogate*) using randomly initialized parameters, and then choose the input that maximizes the trained NN (resp. trained linear model). As a result, if the NN surrogate has an infinite width, the function of the trained NN (resp. trained linear model) is equivalently sampled from the GP posterior with the NTK as the kernel function. This ensures that our algorithms follow the TS policy and hence lays the foundation for our theoretical analyses. Next, to address the second challenge of deriving theoretical guarantees for our algorithms in the batch setting, we generalize the theoretical analysis of sequential TS [7] to account for batch evaluations and derive a regret upper bound for both of our algorithms when the NN surrogate is infinite-width. Then, we leverage insights from batch BO [17] and NTK [31] to show that the regret upper bound is sub-linear (under some conditions), which implies that our algorithms are *asymptotically no-regret*. Next, when the NN surrogate is *finite-width*, we derive a regret upper bound for our STO-BNTS-Linear by carefully accounting for the approximation error caused by the use of a finite (instead of infinite) NN, and show that the regret upper bound of STO-BNTS-Linear remains sub-linear as long as the NN is wide enough.

Our contributions are summarized as follows:

- Our STO-BNTS and STO-BNTS-Linear algorithms sidestep the inversion of the $p \quad p$ matrix required by Neural UCB and Neural TS, which closes their gap between theory and practice.
- Our algorithms naturally support batch evaluations with theoretical guarantees.
- Our algorithms are equipped with an upper bound on their cumulative regret when the NN surrogate is infinite-width, and are asymptotically no-regret (i.e., their regret upper bound is sub-linear) under certain conditions. Moreover, when the NN surrogate is finite-width, our STO-BNTS-Linear still enjoys a regret upper bound, which remains sub-linear as long as the NN is wide enough.
- We demonstrate our empirical effectiveness in real-world experiments including *automated machine learning* (AutoML) and *reinforcement learning* (RL) tasks, as well as a task on optimization over images. To the best of our knowledge, our experiments (Sec. 5) are the first empirical study to show the advantage of neural bandit over GP bandit algorithms in practical AutoML and RL tasks.

2 Background

Neural Networks and Neural Tangent Kernel. In this work, we adopt the same construction of a neural network (NN) as [2]. We use $f(\mathbf{x}; \theta)$ to denote the scalar output of an $(L + 1)$-layer NN with parameters $\theta \in \mathsf{R}^p$ and input $\mathbf{x}$, and use $\nabla_\theta f(\mathbf{x}; \theta')$ to represent the gradient of the NN evaluated at

[1] The work of [23] focuses on the *contextual bandit* setting and aims to choose a batch of inputs *given a batch of diverse contexts*. So, their methods are not applicable in our setting of BO where the contexts are fixed in all iterations, since they do not explicitly encourage input diversity which is a crucial problem in batch BO [8, 17].

$\theta = \theta'$. For simplicity, we assume every layer of the NN has the same width and represent the width as m. We denote our initializaiton scheme for θ as init( ) which simply independently samples every NN parameter from the standard Gaussian distribution. The NTK [27] provides an explicit connection between NNs trained via gradient descent and kernel regression using NTK as the kernel function [2]. The NTK matrix, denoted as , has been shown to stay constant during the course of training as the width m of the NN approaches infinity [27]. Moreover, can be approximated by an *empirical NTK* $\widetilde{\ }$ [2] calculated using a finite-width NN: $\widetilde{\ }(\mathbf{x}, \mathbf{x}') = \langle \nabla_\theta f(\mathbf{x}; \theta_0), \nabla_\theta f(\mathbf{x}'; \theta_0) \rangle$ $(\mathbf{x}, \mathbf{x}')$, where θ_0 init( ) denotes initial parameters and $\nabla_\theta f(\mathbf{x}; \theta_0)$ is referred to as the *neural tangent features* [66]. We refer the readers to the works of [2, 27] for a more detailed background on NTK.

Problem Setting. We aim to maximize a black-box function $f : \mathcal{X} \to \mathbb{R}$, i.e., find $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, in which the domain $\mathcal{X}$ is a finite subset of the d-dimensional unit ball: $\mathcal{X} \subset \{\mathbf{x} \mid \|\mathbf{x}\|_2 \leq 1\}$. Of note, our theoretical results allow $\mathcal{X}$ to be very large because our regret upper bounds only depend on its cardinality $|\mathcal{X}|$ logarithmically. Moreover, all our theoretical results can be easily extended to problems with continuous input domains with an additional assumption on the Lipschitz continuity of f (Appendix E). We focus on the noisy setting, i.e., for every queried $\mathbf{x}$, we observe a noisy output $y(\mathbf{x}) = f(\mathbf{x}) + \zeta$ where $\zeta \sim \mathcal{N}(0, \sigma^2)$. For simplicity, we focus on the setting of *synchronous* batch BO with a batch size B where a new batch of B inputs are selected only after all evaluations of the previous batch are completed [17]. However, our theoretical results also hold for asynchronous batch BO where a new input query is selected once any pending query is completed (Appendix C). We denote the i^{th} selected input in iteration t as $\mathbf{x}_t^i$. We analyze the *cumulative regret* of our algorithms: $R_T = \sum_{t=1}^{T/B} \sum_{i=1}^{B} (f(\mathbf{x}^*) \quad f(\mathbf{x}_t^i))$, because if R_T is shown to be sub-linear in T, then the simple regret $S_T = \min_{t,i}(f(\mathbf{x}^*) \quad f(\mathbf{x}_t^i)) \quad R_T/T$ goes to 0 asymptotically, which implies that our algorithm is *asymptotically no-regret*.

3 Sample-Then-Optimize Batch Neural Thompson Sampling

Our STO-BNTS and STO-BNTS-Linear algorithms are presented in Algos. 1 and 2. In both algorithms, the NN surrogate $f(\mathbf{x}; \theta)$ can be either infinite-width or finite-width. Both STO-BNTS and STO-BNTS-Linear follow the TS policy to select an input query $\mathbf{x}_t^i$: They firstly (a) obtain a function $f_t^i(\mathbf{x}; \theta_t^i)$ which is equivalently sampled from the GP posterior with the NTK as the kernel: $\mathcal{GP}(\mu_{t-1}(\ ), \beta_t^2 \sigma_{t-1}^2(\ ,\ ))$ [24] (see Appendix A for details), and then (b) maximize the function to select the next query: $\mathbf{x}_t^i = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t^i(\mathbf{x}; \theta_t^i)$. Step (a) is achieved via the sample-then-optimize procedure, i.e., by firstly *sampling* initial parameters (θ_0 and θ_0') to construct a function $f_t^i(\mathbf{x}; \theta)$, and then *optimizing* the function using gradient descent to obtain the resulting function of $f_t^i(\mathbf{x}; \theta_t^i)$.

STO-BNTS (Algo. 1). In every iteration t of STO-BNTS, we firstly construct an NN $f(\mathbf{x}; \theta)$ and multiply its output by $\beta_t = 2\log(\pi^2 t^2 |\mathcal{X}|/(3\delta))$, in which $\delta \in (0, 1)$ (Theorem 1).[2] Next, to choose the i^{th} query $\mathbf{x}_t^i$, we start by sampling initial parameters θ_0 init( ) and θ_0' init( ) independently, and then set the parameters of θ_0' in the last layer to 0 (lines 4-5). Next, we use the resulting θ_0 and θ_0', as well as the NN $f(\mathbf{x}, \theta)$, to construct a function $f_t^i(\mathbf{x}, \theta)$ (line 6). Subsequently, in line 7, using the current history of observations (denoted as $\mathcal{D}_{t-1}$) as the training set, we train $f_t^i(\mathbf{x}, \theta)$ (setting θ_0 as the initial parameters) using gradient descent with the following loss function:

$$L_t(\theta, \mathcal{D}_{t-1}) = \sum_{\tau=1}^{t-1} \sum_{j=1}^{B} (y_\tau^j \quad f_t^i(\mathbf{x}_\tau^j; \theta))^2 + \beta_t^2 \sigma^2 \|\theta \quad \theta_0\|_2^2, \tag{1}$$

in which σ^2 is the observation noise variance (Sec. 2). After the training, we use the resulting function $f_t^i(\mathbf{x}; \theta_t^i)$ as the *acquisition function* to choose the i^{th} query: $\mathbf{x}_t^i = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t^i(\mathbf{x}; \theta_t^i)$ (line 8). This procedure (lines 4-8) is repeated independently for $B \quad 1$ times, after which a batch of B queries $\{\mathbf{x}_t^i\}_{i=1,\ldots,B}$ are selected and then queried to produce the observations $\{y_t^i\}_{i=1,\ldots,B}$. Next, the newly collected input-output pairs $\{(\mathbf{x}_t^i, y_t^i)\}_{i=1,\ldots,B}$ are added to $\mathcal{D}_{t-1}$ and the algorithm proceeds to the next iteration. Importantly, if the NN $f(\mathbf{x}; \theta)$ is infinite-width, the function $f_t^i(\mathbf{x}; \theta_t^i)$ obtained after the training in line 7 is *a sample from the GP posterior with the NTK as the kernel*: $\mathcal{GP}(\mu_{t-1}(\ ), \beta_t^2 \sigma_{t-1}^2(\ ,\ ))$ [24] (Appendix A). This ensures the validity of the TS policy and is crucial for deriving the theoretical guarantee of STO-BNTS (Sec. 4).

STO-BNTS-Linear (Algo. 2). Similar to STO-BNTS, at the beginning of iteration t, STO-BNTS-Linear firstly constructs an NN $f(\mathbf{x}; \theta)$ and multiply its output by β_t. To choose the i^{th} query in

[2] Note that β_t is introduced only for the theoretical analysis, and hence we set $\beta_t = 1$ in our experiments.

Algorithm 1 STO-BNTS

1: **for** $t = 1, 2, \ldots, T/B$ **do**
2: Construct NN $f(\mathbf{x}; \theta)$ and multiply its output by β_t (Theorem 1)
3: **for** $i = 1, 2, \ldots, B$ **do**
4: Sample θ_0 init( )
5: Sample θ'_0 init( ) and set the parameters of θ'_0 in the last layer to 0
6: Set $f_t^i(\mathbf{x}; \theta) = f(\mathbf{x}; \theta) + \langle r_\theta f(\mathbf{x}; \theta_0), \theta'_0 \rangle$
7: Use observation history D_{t-1} to train $f_t^i(\mathbf{x}; \theta)$ with the loss function $L_t(\theta, D_{t-1})$ (1) (setting θ_0 as the initial parameters) using gradient descent till convergence, to obtain $\theta_t^i = \arg\min_\theta L_t(\theta, D_{t-1})$
8: Choose $\mathbf{x}_t^i = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t^i(\mathbf{x}; \theta_t^i)$
9: Query the batch $\{\mathbf{x}_t^i\}_{i=1,\ldots,B}$ to yield the observations $\{y_t^i\}_{i=1,\ldots,B}$, and add them to D_{t-1}

Algorithm 2 STO-BNTS-Linear

1: **for** $t = 1, 2, \ldots, T/B$ **do**
2: Construct NN $f(\mathbf{x}; \theta)$ and multiply its output by β_t (Theorem 1)
3: **for** $i = 1, 2, \ldots, B$ **do**
4: Sample θ'_0 init( ), and define $f_t^i(\mathbf{x}; \theta) = \langle r_\theta f(\mathbf{x}; \theta'_0), \theta \rangle$
5: Sample θ_0 init( )
6: Use observation history D_{t-1} to train $f_t^i(\mathbf{x}; \theta)$ with the loss function $L_t(\theta, D_{t-1})$ (1) (setting θ_0 as the initial parameters) using gradient descent till convergence, to obtain $\theta_t^i = \arg\min_\theta L_t(\theta, D_{t-1})$
7: Choose the query $\mathbf{x}_t^i = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t^i(\mathbf{x}; \theta_t^i)$
8: Query the batch $\{\mathbf{x}_t^i\}_{i=1,\ldots,B}$ to yield the observations $\{y_t^i\}_{i=1,\ldots,B}$, and add them to D_{t-1}

iteration t, STO-BNTS-Linear firstly constructs a linear model $f_t^i(\mathbf{x}; \theta)$ using the neural tangent features (i.e., $r_\theta f(\mathbf{x}; \theta'_0)$ where θ'_0 init( ) are initial parameters) as the input features (line 4). Next, we sample θ_0 init( ) (line 5) and use it as the initial parameters to train $f_t^i(\mathbf{x}; \theta)$ using gradient descent with the same loss function (1) as STO-BNTS, to produce θ_t^i (line 6). After that, the i^{th} query is selected by maximizing the acquisition function $f_t^i(\mathbf{x}; \theta_t^i)$: $\mathbf{x}_t^i = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t^i(\mathbf{x}; \theta_t^i)$ (line 7).

Lines 4-6 of STO-BNTS-Linear can be interpreted as a sample-then-optimize method [42] using the neural tangent features as the input features. As a result, same as STO-BNTS, if an infinite-width NN is used, the function $f_t^i(\mathbf{x}; \theta_t^i)$ obtained after the training in line 6 is a sample from the GP posterior with the NTK as the kernel: $GP(\mu_{t-1}(\,), \beta_t^2 \sigma_{t-1}^2(\,,\,))$ (Appendix A). However, in contrast to STO-BNTS, if the NN is *finite-width*, the function $f_t^i(\mathbf{x}; \theta_t^i)$ derived after line 6 of Algo. 2 still corresponds to a sample from the GP posterior with the *empirical NTK* $\tilde{}(\mathbf{x}, \mathbf{x}') = \langle r_\theta f(\mathbf{x}; \theta'_0),\ r_\theta f(\mathbf{x}'; \theta'_0) \rangle$ as the kernel. As a result, for infinite-width NNs, STO-BNTS-Linear and STO-BNTS enjoy the same sub-linear (under some conditions) upper bound on their cumulative regret (Sec. 4.1); however, for finite-width NNs, unlike STO-BNTS, STO-BNTS-Linear still enjoys a regret upper bound which is sub-linear as long as the NN is wide enough (Sec. 4.2).

Although STO-BNTS-Linear (Algo. 2) has the theoretical advantage of a theoretically guaranteed convergence also for finite-width NNs (Sec. 4.2), we expect STO-BNTS (Algo. 1) to perform better in practice. This is because STO-BNTS explicitly trains an NN surrogate model in every iteration and is hence able to *directly exploit the strong representation power of NNs* to model the objective function f. In contrast, STO-BNTS-Linear derives its representation power entirely from the neural tangent features of NTK. However, it has been shown [1, 69] that *neural tangent features of NTK can not completely realize the representation power of NNs*. As a result, STO-BNTS is expected to be more competitive in practice, especially in problems where the strong representation power of NNs is essential for accurately modeling the complex objective functions. We empirically verify this practical advantage of STO-BNTS in our experiments (Sec. 5.5).

4 Theoretical Results

We firstly prove a regret upper bound for both STO-BNTS and STO-BNTS-Linear using infinite-width NNs, and show that both algorithms are asymptotically no-regret under certain conditions (Sec. 4.1). Next, we derive a regret upper bound for STO-BNTS-Linear when the NN is finite-width, and show that the regret upper bound remains sub-linear as long as the NN is wide enough (Sec. 4.2).

4.1 Infinite-width NNs

Here we use $\mathbf{y}_{1:t}$ to denote the output observations from iterations 1 to t (i.e., $t \quad B$ observations in total) and use $\mathbf{y}_A$ to denote the vector of observations at a set of inputs $A \quad X$. Theorem 1 (proof in Appendix C) gives a regret upper bound for both STO-BNTS and STO-BNTS-Linear:

Theorem 1 (Infinite-width NNs)**.** *Assume that f is sampled from a GP with the NTK as the kernel function, that $|f(\mathbf{x})| \quad B' \; 8\mathbf{x} \; 2 \; X$ for some $B' > 0$, and that $(\mathbf{x}, \mathbf{x}') \quad K_0 \; 8\mathbf{x}, \mathbf{x}' \; 2 \; X$ for some $K_0 > 0$. Let $\delta \; 2 \; (0, 1)$ and $\beta_t = 2\log(\pi^2 t^2 |X| / (3\delta))$. Then with probability of $1 \quad \delta$,*

$$R_T = \widetilde{O}(e^C \sqrt{T}(\sqrt{\gamma_T} + 1))$$

where $\widetilde{O}$ ignores all logarithmic factors, γ_T is the max information gain about f from any set of T observations, and C is an absolute constant s.t. $\max_{A \subset \mathcal{X}, |A| \leq B-1} I(f; \mathbf{y}_A | \mathbf{y}_{1:t}) \quad C, 8t \quad 1$.

The assumption of $(\mathbf{x}, \mathbf{x}') \quad K_0$ is natural and in line with [31], and $|f(\mathbf{x})| \quad B'$ is also a mild assumption since most practical objective functions have bounded values. Our primary assumption that f is sampled from a GP with the NTK as the kernel function is a common assumption in the analysis of BO algorithms [17, 57]. Of note, compared with the assumptions from previous works using other commonly used kernels such as the squared exponential (SE) kernel [17, 57], our assumption on f holds for more complicated objective functions. Specifically, the class of functions sampled from a GP with the NTK as the kernel subsumes more non-smooth and sophisticated objective functions compared with the other commonly used kernels. As shown in [17], C can be chosen to be an absolute constant that is independent of B and T as long as we initialize our algorithm using the uncertainty sampling method, which entails choosing the initial inputs by sequentially maximizing the GP posterior variance (more details in Appendix B). Specifically, for any chosen constant $C > 0$, as long as we run the uncertainty sampling initialization stage for T_{init} iterations such that $((B \quad 1)\gamma_{T_{\text{init}}})/T_{\text{init}} \quad C$, then $\max_{A \subset \mathcal{X}, |A| \leq B-1} I(f; \mathbf{y}_A | \mathbf{y}_{1:t}) \quad C, 8t \quad 1$ is guaranteed to be satisfied. Since it has been shown by the work of [31] that $\gamma_T = \widetilde{O}(T^{(d-1)/d})$ grows sub-linearly for the NTK,[3] therefore, $((B \quad 1)\gamma_{T_{\text{init}}})/T_{\text{init}}$ is decreasing as T_{init} increases. As a result, for any chosen C, we are able to choose a finite T_{init} such that the condition $((B \quad 1)\gamma_{T_{\text{init}}})/T_{\text{init}} \quad C$ is satisfied.

For example, if we choose $C = 1$, then the required number of initial iterations is approximately $T_{\text{init}} = \quad ((B \quad 1)^d)$. As a result, the regret upper bound will be a summation of $2B'T_{\text{init}}$ (i.e., the regrets incurred during the initializatioin stage, because the regret at every step is upper-bounded by $2B'$) and the regret upper bound from Theorem 1 with $C = 1$. Since both B' and T_{init} are constants independent of T (assuming that B is independent of T), the asymptotic regret upper bound can be simplified into $R_T = \widetilde{O}(\sqrt{T}(1 + \sqrt{\gamma_T}))$. Plugging in $\gamma_T = \widetilde{O}(T^{(d-1)/d})$, the final regret upper bound becomes $R_T = \widetilde{O}(T^{1/2} + T^{(2d-1)/(2d)}) = \widetilde{O}(T^{(2d-1)/(2d)})$ which is sub-linear in T and hence implies that our STO-BNTS and STO-BNTS-Linear are both *asymptotically no-regret* when the NN surrogate is infinite-width.

Moreover, when $T \quad B$ and B is a constant which is independent of T, Theorem 1 gives us insights on the benefit of batch over sequential evaluations. In this case, the regrets incurred during initialization (i.e., $\widetilde{O}((B \quad 1)^d)$) is negligible. Therefore, our analysis above suggests that our algorithms with batch evaluations ($B > 1$) enjoys the same asymptotic regret upper bound as its sequential counterpart ($B = 1$) since the resulting regret upper bound of $R_T = \widetilde{O}(\sqrt{T}(1 + \sqrt{\gamma_T}))$ does not depend on the batch size B. This demonstrates the advantage of batch evaluations because when $B > 1$, some of our evaluations can be run in parallel, which is not supported by the sequential setting with $B = 1$. As a simple illustration, for a large T, both the sequential and batch settings achieve a simple regret of the order $\widetilde{O}(T^{-1/(2d)})$ after T function evaluations. However, since our batch setting can evaluate every $B > 1$ selected inputs in parallel in every iteration, our batch setting with $B > 1$ achieves this simple regret after only T/B iterations, which is only a fraction $(1/B)$ of the T iterations required by the sequential setting. This also shows that we enjoy more benefit with a larger batch size B.

[3] Note that in order to quote the results from [31], we need follow their assumption to assume that the domain $\mathcal{X}$ is a subset of the d-dimensional unit hyper-sphere: $\mathcal{X} \subset \{\mathbf{x} | \|\mathbf{x}\|_2 = 1\}$, which is more strict than our main assumption (Sec. 2) that $\mathcal{X}$ is a subset of the unit hyper-ball: $\mathcal{X} \subset \{\mathbf{x} | \|\mathbf{x}\|_2 \leq 1\}$.

4.2 Finite-width NNs

Here we prove a regret upper bound for STO-BNTS-Linear with *finite-width* NN. The main technical challenge in the proof lies in the disparity between the exact and empirical NTKs, i.e., the function f is assumed to be sampled from a GP with the exact NTK yet our acquisition function $f_t^i(\mathbf{x}; \theta_t^i)$ (line 7 of Algo. 2) is obtained using the empirical NTK $\widetilde{}$ when the NN is finite-width. To this end, we make use of the following theoretical guarantee on the approximation error between and $\widetilde{}$.

Proposition 1 (Theorem 3.1 of [2]). *Choose $\varepsilon > 0$ and $\delta \in (0, 1)$. If the width of every layer in an NN satisfies $m =$ $(\frac{L^6}{\varepsilon^4} \log(4LjXj^2/\delta))$, then $\forall \mathbf{x}, \mathbf{x}' \in X$, we have with probability $1 - \delta/4$ that*

$$\left|\langle r_\theta f(\mathbf{x}, \theta_0), r_\theta f(\mathbf{x}', \theta_0)\rangle - (\mathbf{x}, \mathbf{x}')\right| \leq (L+1)\varepsilon.$$

Proposition 1 ensures that we can reduce the upper bound $(L+1)\varepsilon$ on the approximation error between and $\widetilde{}$ by increasing the width m of the NN. For example, let $m = C_{\text{ntk}} L^6 \varepsilon^{-4} \log(4LjXj^2/\delta)$ for some constant $C_{\text{ntk}} > 0$, then the approximation quality of Proposition 1 can be expressed as $(L+1)\varepsilon = C_{\text{ntk}}(L+1)L^{3/2}\log^{1/4}(4LjXj^2/\delta)m^{-1/4}$ which decreases as the width m increases. In our theoretical analysis, we assume that $(L+1)\varepsilon \leq 1$, which can be satisfied as long as m is large enough. The regret upper bound for STO-BNTS-Linear is given in Theorem 2 (proof in Appendix D):

Theorem 2 (Finite-width NNs). *Let $\delta \in (0, 1)$ and $\beta_t = 2\log(2\pi^2 t^2 jXj/(3\delta))$. Then we have*

$$R_T = \widetilde{O}(e^C \sqrt{\overline{T}(\sqrt{\overline{\gamma_T}} + 1)} + T^3 m^{-1/8}(L+1)^{5/4}),$$

with probability of $\geq 1 - \delta$. Here γ_T and C are the same as those defined in Theorem 1.

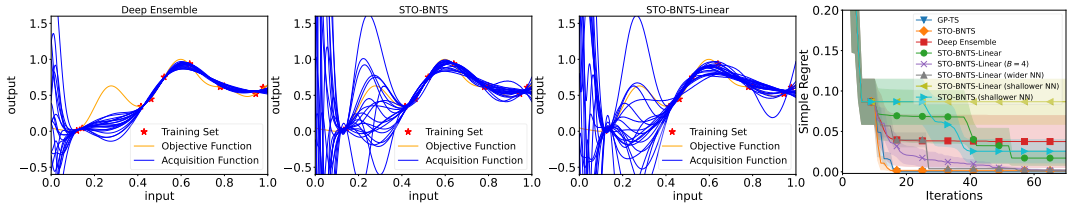
The first term in the regret upper bound in Theorem 2 is the same as that of Theorem 1 for infinite-width NNs and can hence be made sub-linear by using uncertainty sampling as the initialization stage: $R_T = \widetilde{O}(\sqrt{\overline{T}(1 + \sqrt{\overline{\gamma_T}})}) = \widetilde{O}(T^{(2d-1)/(2d)})$. The second term in the regret upper bound represents the additional regrets incurred by the use of finite-width NNs, which can also be made sub-linear by choosing $m = \Omega(T^{24})$. In other words, if the width m of the NN is chosen to be large enough (i.e., if $m = \Omega(T^{24})$), then the cumulative regret of STO-BNTS-Linear scales sub-linearly in T.

4.3 Discussion

The assumption on the function f for our theoretical analyses in Theorems 1 and 2 differs from those made by the previous works on neural contextual bandits [66, 69]. Specifically, these previous works have relied on the assumption of a positive definite NTK Gram matrix to approximate the value of f (only evaluated at the observed contexts up to iteration T) using a function that is linear in the neural tangent features. When translated into our setting (which is equivalent to the contextual bandit setting where all contexts are fixed for all $t \geq 1$), the assumption of a positive definite NTK Gram matrix can be easily violated as long as any input $\mathbf{x}$ is queried more than once, thus rendering this assumption unrealistic in our setting. In contrast, we have assumed that f is sampled from a GP with the NTK as the kernel, which is a common assumption in the analysis of BO [17, 57] and allows us to derive a sub-linear regret upper bound. As a result of the different assumptions, our regret upper bounds are not directly comparable with those from the previous works on neural contextual bandits [66, 69].

5 Experiments

We compare our STO-BNTS and STO-BNTS-Linear with the baselines of Neural UCB [69] and Neural TS [66], as well as GP-UCB and GP-TS which use GPs (with the SE kernel) instead of NNs as their surrogate models. The original implementations of Neural UCB [69] and Neural TS [66] are only applicable to discrete domains. So, for a fair comparison in those tasks with continuous domains, we have modified their implementations to maximize their acquisition functions in the same way as our methods (i.e., through a combination of random search and L-BFGS-B, refer to Appendix F for more details). We firstly explore some interesting insights about our algorithms using a synthetic experiment in Sec. 5.1. Next, we apply our algorithms to real-world AutoML (Sec. 5.2) and RL (Sec. 5.3) problems, as well as an optimization task over images (Sec. 5.4). Finally, we discuss some interesting insights from our experiments in Sec. 5.5. We plot the simple regret (or the best found observation till an iteration) when presenting the experimental results, which is the common practice in BO [8, 29]. We have deferred some experimental details to Appendix F due to space limitation. Our code is available at `https://github.com/daizhongxiang/sto-bnts`.



(a) (b) (c) (d)

Figure 1: (a-c) Acquisition functions and (d) performances in the synthetic experiment (Sec. 5.1).

5.1 Synthetic Experiment

Here, we sample a smooth function from a GP with an SE kernel (with a length scale of 0.1), defined on a discrete 1-dimensional domain within the range of $[0, 1]$. For all methods, we use an NN architecture with a depth of $L = 8$ and a width of $m = 64$ unless specified otherwise.

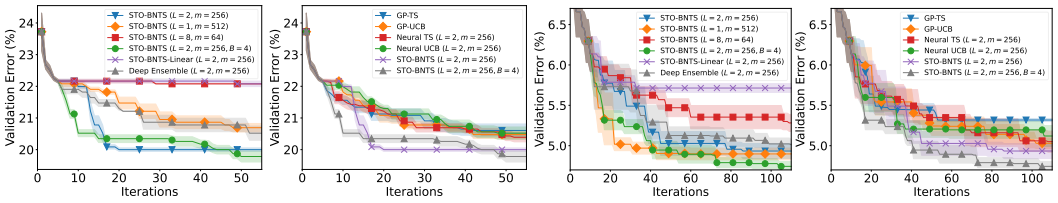
Figs. 1 (a-c) illustrate the acquisition functions $f_t^i(\mathbf{x}; \theta_t^i)$ (line 8 of Algo. 1 and line 7 of Algo. 2) of different algorithms. The *Deep Ensemble* method [36] in Fig. 1a can be regarded as a reduced version of our STO-BNTS algorithm (Algo. 1) in which the term $\hbar r_\theta f(\mathbf{x}; \theta_0), \theta_0' i$ (i.e., the second term in line 6 of Algo. 1) is removed. As a result, Deep Ensemble does not enjoy the theoretical guarantees of our algorithms (Sec. 4). In Figs. 1 (a-c), given the same training set (red stars), every method constructs a batch of $B = 100$ acquisition functions. E.g., STO-BNTS repeats lines 4-7 of Algo. 1 independently for $B = 100$ times to produce acquisition functions $f_t^i(\mathbf{x}; \theta_t^i)$ for $i = 1, \ldots, 100$, which are plotted as the blue lines in Fig. 1b. Note that every acquisition function (blue line) is maximized to select an input query. The figures show that compared with the naive baseline of Deep Ensemble, our STO-BNTS and STO-BNTS-Linear are able to display more exploratory behaviors in unexplored regions (e.g., the interval of $[0.2, 0.4]$). This may be explained by our theoretical guarantees (Sec. 4) which imply that both of our algorithms are able to perform exploration in a principled way and hence naturally handle the the exploration-exploitation trade-off. Moreover, it has also been justified by [24] that the addition of the term $\hbar r_\theta f(\mathbf{x}; \theta_0), \theta_0' i$ improves the ability of the NN to characterize the uncertainty of predictions, which corroborates our findings here.

The simple regrets of different algorithms are plotted in Fig. 1d.[4] The first interesting observation is that our STO-BNTS (orange) significantly outperforms Deep Ensemble (red), which corroborates the insight discussed above positing that the addition of the term $\hbar r_\theta f(\mathbf{x}; \theta_0), \theta_0' i$ leads to more principled exploration and hence better performances. Due to its lack of exploration as illustrated in Fig. 1a, Deep Ensemble fails to reach zero regret in Fig. 1d. Furthermore, the discrepancy between the green and purple curves shows that batch evaluations ($B = 4$) lead to significant performance improvement. Moreover, compared with the green curve for which $m = 64$, *using a wider NN* (gray curve, $m = 512$) *substantially improves the performance of STO-BNTS-Linear* yet employing a shallower NN (yellow curve, $m = 16$) significantly degrades the performance. These observations agree with Theorem 2 which states that a larger width m reduces the regret of STO-BNTS-Linear. Similarly, the NN surrogate model of STO-BNTS should also be wide enough since the use of a narrower NN (light blue curve, $m = 16$) also leads to a worse performance for STO-BNTS. Lastly, GP-TS (blue) performs competitively in this experiment with a very smooth objective function. However, as we will show in the next two sections, in real-world experiments with more complicated objective functions, GP-based methods are consistently outperformed by our algorithms.

5.2 Real-world Experiments on Automated Machine Learning (AutoML)

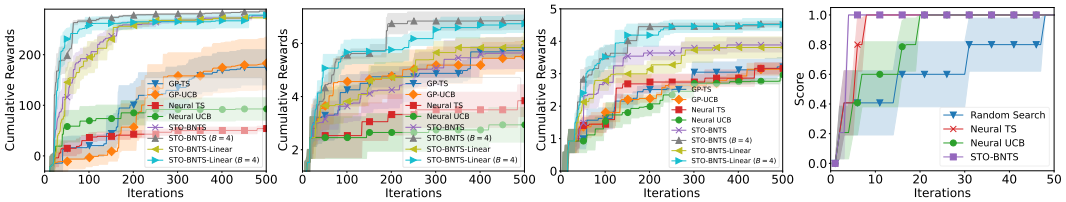
Here, we adopt 3 hyperparameter tuning tasks. We use tasks involving categorical hyperparameters to highlight the advantage of our NN-based over GP-based methods. We use a diabetes diagnosis dataset to tune 6 categorical hyperparameters of random forest (RF), and then use the MNIST dataset to tune 9 hyperparameters (4 continuous and 5 categorical) of XGBoost and 9 hyperparameters (2 continuous and 7 categorical) of convolutional neural networks (CNNs). Fig. 2 plots the results for the RF (a,b) and XGBoost (c,d) tasks, and the results for CNN are shown in Fig. 4 (Appendix F.2).

[4]To show the benefit of batch evaluations, in all experiments (including real-world experiments), we use the *iterations* t as the horizontal axis and in every iteration t, we report the largest $f(\mathbf{x}_t^i)$ (y_t^i in real-world experiments) within a batch (when $B > 1$) as the observation in this iteration.



(a) (b) (c) (d)

Figure 2: Validation errors for hyperparameter tuning of RF (a,b), XGBoost (c,d). $B = 1$ by default.



(a) (b) (c) (d)

Figure 3: Results for (a) 12-D Lunar-Lander, (b) 14-D robot pushing, (c) 20-D rover trajectory planning, and (d) optimization over images in Sec. 5.4. $B = 1$ unless specified otherwise.

For each task, we firstly compare the performances of different variants of our algorithms in Figs. 2a, c and Fig. 4a, which demonstrate a number of interesting insights. Firstly, our STO-BNTS outperforms Deep Ensemble, which is consistent with the results in Fig. 1d and further emphasizes the practical significance of the improved exploration performed by our STO-BNTS (Sec. 5.1). Secondly, STO-BNTS-Linear is significantly outperformed by STO-BNTS in Figs. 2 a and c, which can likely be attributed to its inability to fully leverage the representation power of NNs as we have discussed in Sec. 3 (see more discussions in Sec. 5.5). Next, Figs. 2a, c and Fig. 4a also show that the performance of STO-BNTS tends to suffer when the NN surrogate model is either overly shallow ($L = 1, m = 512$, orange curves) or overly deep ($L = 8, m = 64$, red curves), that is, both the orange and red curves underperform significantly in two of the three experiments. This is likely because an overly shallow NN lacks the representation power to model complicated objective functions, whereas an overly deep NN may be prone to overfitting since the size of the training set here is much smaller than typical deep learning applications. In contrast, the NN architecture of $L = 2, m = 256$ (blue curves) consistently performs well in all three experiments, therefore, we will use it as the default architecture in the experiments in the next section. Moreover, the benefit of batch evaluations can also be further confirmed by comparing the blue ($B = 1$) and green ($B = 4$) curves in Figs. 2a, c and Fig. 4a.

Figs. 2b, d and Fig. 4b show that for the same NN architecture of $L = 2, m = 256$, our STO-BNTS is the best-performing method since it performs better than all baselines in Figs. 2b and d and comparably with them in Fig. 4b. The undesirable performances of Neural UCB and Neural TS may be explained by the errors due to their diagonal matrix approximation (Sec. 1), and the underwhelming performances of GP-UCB and GP-TS may result from the ineffectiveness of GP in modeling categorical inputs (more on this in Sec. 5.5).

5.3 Real-world Experiments on Reinforcement Learning (RL)

Here we optimize the control parameters of 3 RL problems: we tune $d = 12$ parameters of a heuristic controller for the Lunar-Lander task from OpenAI Gym [4], $d = 14$ parameters of a controller for a robot pushing task [62], and $d = 20$ parameters for a rover trajectory planning problem [62]. We use $L = 2, m = 256$ for all methods. The results in these three RL tasks are plotted in Figs. 3a, b and c. Our STO-BNTS and STO-BNTS-Linear (purple and yellow curves) consistently perform the best among all methods with sequential evaluations ($B = 1$), and our methods with batch evaluations ($B = 4$, gray and light blue curves) achieve further performance improvements over their sequential counterparts. Of note, despite underperforming in Sec. 5.2, STO-BNTS-Linear achieves comparable performances with STO-BNTS in all three experiments here for both the sequential and batch evaluations, outperforming the other baselines. The inefficacy of GP-TS and GP-UCB here may result from the inability of GPs to effectively model high-dimensional input space [30] (Sec. 5.5).

5.4 Optimization over Images

Some real-world applications require optimizing over an input domain of *images*. For example, an image recommender system sequentially recommends different images to a user in order to select the image with the best rating/score [65]. In these applications, GP-based BO methods require sophisticated techniques such as convolutional kernels in order to model the inputs of images [59], in contrast, our methods can be easily applied by simply replacing the NN surrogate model with a CNN. Here, we simulate these applications by maximizing a score over the domain of images from the MNIST dataset. For all CNN-based methods, we use a CNN with a convolutional layer followed by a fully connected layer (both with $m = 64$) as the surrogate model (more details in Appendix F.4). The results (Fig. 3d) show that our STO-BNTS performs the best. Moreover, when using the same CNN architecture, STO-BNTS-Linear fails to achieve comparable performances to the other CNN-based methods since STO-BNTS-Linear is not able to fully leverage the representation power of CNN (Sec. 3). However, again consistent with Theorem 2, increasing the width of the CNN can improve the performance of STO-BNTS-Linear to be comparable with STO-BNTS (Fig. 6 in Appendix F.4).

5.5 Discussion

Secs. 5.2, 5.3 and 5.4 show that our algorithms, although *without any special design for a specific type of problem* (e.g., problems involving categorical, high-dimensional or image inputs), are *competitive in all these problems* thanks to the ability of NNs to model complicated real-world functions. Compared with GP-based BO methods, our algorithms may incur more computation due to the need to train an NN surrogate model. However, the additional computation can be easily overshadowed by the cost of function evaluations since BO is usually used to optimize expensive-to-compute functions.

STO-BNTS vs. STO-BNTS-Linear. Our experimental results have demonstrated some interesting insights on the comparison between our STO-BNTS (Algo. 1) and STO-BNTS-Linear (Algo. 2). As we have discussed in Sec. 3, since STO-BNTS-Linear is unable to explicitly take advantage of the representation power of NNs, it is expected to be less competitive than STO-BNTS in practice especially in problems where the strong representation power of NNs is crucial for accurately modeling the objective function. Examples of such problems include those with categorical (Sec. 5.2) or image (Sec. 5.4) inputs. Interestingly, our results in Secs. 5.2 and 5.4 indeed corroborate this insight by showing that STO-BNTS-Linear is consistently outperformed by STO-BNTS in these experiments. However, note that in such problems, the performance of STO-BNTS-Linear can be significantly improved by further increasing the width of the NN (CNN) as we have shown in Fig. 6. In addition, the practical efficacy of STO-BNTS-Linear can also be seen from the experiments in Sec. 5.3, in which STO-BNTS-Linear performs comparably with STO-BNTS and *consistently outperforms all other baselines*. This demonstrates the empirical competence of STO-BNTS-Linear in some real-world problems such as RL. Moreover, also note that compared with STO-BNTS, STO-BNTS-Linear enjoys the theoretical advantage of having a guaranteed convergence for finite-width NNs (Sec. 4.2).

Baseline Methods. The underwhelming performances of Neural UCB and Neural TS in our experiments are likely caused by the errors introduced by the diagonal matrix approximation that is used to avoid the inversion of the $p \quad p$ matrix (Sec. 1). The inadequate performances of GP-UCB and GP-TS may be explained by the ineffectiveness of GPs to model objective functions involving categorical [15, 18] or high-dimensional inputs [30, 43], which correspond to the experiments in Secs. 5.2 and 5.3, respectively. Nevertheless, we expect GP-based methods to achieve better performances (than their performances here) in problems with lower-dimensional and purely continuous input space (e.g., GP-TS performs competitively in the synthetic experiment as shown in Fig. 1d).

Depth L and Width m of the NN Surrogate Model. Our experimental results have provided some guidelines on the choices of the depth L and width m of the NN surrogate model in our algorithms. Regarding the depth L, our experiments in Sec. 5.2 have shown that an overly shallow NN usually hurts the performance due to its lack of expressive power, whereas an excessively deep NN is also likely to deteriorate the performance due to overfitting. Therefore, we discourage the use of NNs which are either exceedingly shallow or overly deep, and recommend shallower NNs for simpler tasks to prevent overfitting and deeper NNs for more complicated tasks to gain enough representation power. The width m should be chosen to be large enough since our experiments in Sec. 5.1 (Fig. 1d) and Sec. 5.4 (Fig. 6 in Appendix F.4) suggest that a larger width usually improves the performance. Moreover, we have shown that the choice of $L = 2, m = 256$ consistently leads to competitive performances in a wide range of experiments (i.e., all experiments in Sec. 5.2 and Sec. 5.3). Therefore, we recommend it as the default choice for real-world problems.

6 Related Works

The NTK provides a theoretical tool to study the training dynamics of NNs by drawing connections with kernel methods [2, 5, 13, 27, 38], and it has been successfully applied to a number of practical problems such as neural architecture search [52, 54], active learning [61], data valuation [63], among others. The pioneering work of [69] exploited the theory of NTK to introduce the Neural UCB algorithm for contextual bandits, which uses an NN to learn the reward (objective) function and leverages neural tangent features for exploration following the UCB principle. Neural UCB has been followed up by other works which mainly aimed to make Neural UCB more practical. [64] proposed to improve the computational efficiency of Neural UCB through the additional assumption that the reward function is linear in the feature mapping of the last layer of the NN, [23] also aimed to reduce the computational cost of Neural UCB by limiting the number of updates of the NN surrogate model, and [44] proposed a method to reduce the memory requirement of generic neural bandit algorithms. The recent work of [41] performed an empirical study of neural bandit algorithms, and discovered that they tend to perform competitively in problems that require learning complicated representations. Above all, the work that is most closely related to our paper is [66], which introduced the Neural TS algorithm. Following similar principles as Neural UCB, Neural TS learns the reward function using an NN surrogate model and constructs the exploration term through the neural tangent features, which are used as, respectively, the mean and variance of the Gaussian distribution from which the reward is sampled for running the TS routine. GP-based BO methods [3, 33, 48, 49] have achieved impressive performances in recent years, and have been extended to various problem settings such as high-dimensional BO [19, 26, 30], multi-fidelity BO [14, 28, 67, 68], meta-BO [10, 50], risk-averse BO [46, 47, 58], multi-agent/collaborative BO [9, 56], non-myopic BO [34, 40], among others. More importantly, they have also been extended to the batch setting, based on either GP-UCB [8, 16, 17, 32, 60] or GP-TS [25, 29, 45, 60].

7 Conclusion

We propose STO-BNTS and STO-BNTS-Linear, both of which sidestep the requirement to invert a large parameter matrix in existing neural bandit algorithms, and naturally support batch evaluations while preserving their theoretical guarantees. Both algorithms are asymptotically no-regret under certain conditions if the NN surrogate is infinite-width, and STO-BNTS-Linear still enjoys sub-linear regret for a finite-width NN if it is wide enough. A potential limitation is that our theoretical analysis for finite-width NNs (Sec. 4.2) only holds for STO-BNTS-Linear but not for STO-BNTS, which we will explore in future works. Another promising future topic is to leverage our ability to exploit the strong representation power of NNs to apply our algorithms to other challenging optimization tasks with sophisticated search spaces, such as chemical design [22, 35], neural architecture search [53, 55], etc. Moreover, it is also an interesting future direction to extend our algorithms to handle other problem settings such as those which have been considered by BO (Sec. 6), e.g., multi-fidelity optimization [67, 68], risk aversion/fault tolerance [20, 46, 47], etc. A potential negative societal impact is that our work may promote more adoption of deep learning methods and hence cause more electricity consumption.

Acknowledgments and Disclosure of Funding

This research/project is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund – Pre Positioning (IAF-PP) (Award A19E4a0101).

References

[1] Z. Allen-Zhu and Y. Li. What can ResNet learn efficiently, going beyond kernels? In *Proc. NeurIPS*, 2019.

[2] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. arXiv:1904.11955, 2019.

[3] S. Balakrishnan, Q. P. Nguyen, B. K. H. Low, and H. Soh. Efficient exploration of reward functions in inverse reinforcement learning via Bayesian optimization. In *Proc. NeurIPS*, pages 4187–4198, 2020.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. arXiv:1606.01540, 2016.

[5] Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Proc. NeurIPS*, volume 32, pages 10836–10846, 2019.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[7] S. R. Chowdhury and A. Gopalan. On kernelized multi-armed bandits. In *Proc. ICML*, pages 844–853, 2017.

[8] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[9] Z. Dai, Y. Chen, B. K. H. Low, P. Jaillet, and T.-H. Ho. R2-B2: Recursive reasoning-based Bayesian optimization for no-regret learning in games. In *Proc. ICML*, pages 2291–2301, 2020.

[10] Z. Dai, Y. Chen, H. Yu, B. K. H. Low, and P. Jaillet. On provably robust meta-Bayesian optimization. In *Proc. UAI*, 2022.

[11] Z. Dai, B. K. H. Low, and P. Jaillet. Federated Bayesian optimization via Thompson sampling. In *Proc. NeurIPS*, pages 9687–9699, 2020.

[12] Z. Dai, B. K. H. Low, and P. Jaillet. Differentially private federated Bayesian optimization with distributed exploration. In *Proc. NeurIPS*, pages 9125–9139, 2021.

[13] Z. Dai, Y. Shu, A. Verma, F. X. Fan, B. K. H. Low, and P. Jaillet. Federated neural bandit. arXiv:2205.14309, 2022.

[14] Z. Dai, H. Yu, B. K. H. Low, and P. Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML*, pages 1496–1506, 2019.

[15] E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. Mixed-variable Bayesian optimization. In *Proc. IJCAI*, 2020.

[16] E. A. Daxberger and B. K. H. Low. Distributed batch Gaussian process optimization. In *Proc. ICML*, pages 951–960, 2017.

[17] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

[18] A. Deshwal, S. Belakaria, and J. R. Doppa. Bayesian optimization over hybrid spaces. In *Proc. ICML*, 2021.

[19] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. volume 32, pages 5496–5507, 2019.

[20] F. X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low. Fault-tolerant federated reinforcement learning with theoretical guarantee. In *Proc. NeurIPS*, 2021.

[21] P. I. Frazier. A tutorial on Bayesian optimization. arXiv:1807.02811, 2018.

[22] R.-R. Griffiths and J. M. Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

[23] Q. Gu, A. Karbasi, K. Khosravi, V. Mirrokni, and D. Zhou. Batched neural bandits. arXiv:2102.13028, 2021.

[24] B. He, B. Lakshminarayanan, and Y. W. Teh. Bayesian deep ensembles via the neural tangent kernel. In *Proc. NeurIPS*, 2020.

[25] J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proc. ICML*, pages 1470–1479. PMLR, 2017.

[26] T. N. Hoang, Q. M. Hoang, and B. K. H. Low. Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proc. AAAI*, pages 3231–3238, 2018.

[27] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proc. NeurIPS*, 2018.

[28] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proc. ICML*, pages 1799–1808. PMLR, 2017.

[29] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *Proc. AISTATS*, pages 133–142. PMLR, 2018.

[30] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *Proc. ICML*, pages 295–304. PMLR, 2015.

[31] P. Kassraie and A. Krause. Neural contextual bandits without regret. arXiv:2107.03144, 2021.

[32] T. Kathuria, A. Deshpande, and P. Kohli. Batched Gaussian process bandit optimization via determinantal point processes. In *Proc. NeurIPS*, volume 29, pages 4206–4214, 2016.

[33] D. Kharkovskii, Z. Dai, and B. K. H. Low. Private outsourced Bayesian optimization. In *Proc. ICML*, pages 5231–5242, 2020.

[34] D. Kharkovskii, C. K. Ling, and B. K. H. Low. Nonmyopic Gaussian process optimization with macro-actions. In *Proc. AISTATS*, pages 4593–4604, 2020.

[35] K. Korovina, S. Xu, K. Kandasamy, W. Neiswanger, B. Poczos, J. Schneider, and E. Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *Proc. AISTATS*, pages 3393–3403. PMLR, 2020.

[36] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc. NeurIPS*, 2017.

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[38] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. volume 32, pages 8572–8583, 2019.

[39] Z. Li and J. Scarlett. Gaussian process bandit optimization with few batches. In *Proc. AISTATS*, pages 92–107, 2022.

[40] C. K. Ling, B. K. H. Low, and P. Jaillet. Gaussian process planning with Lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. In *Proc. AAAI*, pages 1860–1866, 2016.

[41] M. Lisicki, A. Afkanpour, and G. W. Taylor. An empirical study of neural kernel bandits. In *NeurIPS Workshop on Bayesian Deep Learning*, 2021.

[42] A. G. d. G. Matthews, J. Hron, R. E. Turner, and Z. Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. In *NeurIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

[43] M. Mutnỳ and A. Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Proc. NeurIPS*, pages 9005–9016. Curran, 2019.

[44] O. Nabati, T. Zahavy, and S. Mannor. Online limited memory neural-linear bandits with likelihood matching. In *Proc. ICML*, 2021.

[45] E. Nava, M. Mutny, and A. Krause. Diversified sampling for batched bayesian optimization with determinantal point processes. In *Proc. AISTATS*, pages 7031–7054. PMLR, 2022.

[46] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. Optimizing conditional value-at-risk of black-box functions. In *Proc. NeurIPS*, pages 4170–4180, 2021.

[47] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. Value-at-risk optimization with Gaussian processes. In *Proc. ICML*, pages 8063–8072, 2021.

[48] Q. P. Nguyen, S. Tay, B. K. H. Low, and P. Jaillet. Top-k ranking Bayesian optimization. In *Proc. AAAI*, pages 9135–9143, 2021.

[49] Q. P. Nguyen, Z. Wu, B. K. H. Low, and P. Jaillet. Trusted-maximizers entropy search for efficient Bayesian optimization. In *Proc. UAI*, pages 1486–1495, 2021.

[50] D. Salinas, H. Shen, and V. Perrone. A quantile-based approach for hyperparameter transfer learning. In *Proc. ICML*, pages 8438–8448. PMLR, 2020.

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

[52] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low. NASI: Label- and data-agnostic neural architecture search at initialization. In *Proc. ICLR*, 2022.

[53] Y. Shu, Y. Chen, Z. Dai, and B. K. H. Low. Neural ensemble search via Bayesian sampling. In *Proc. UAI*, 2022.

[54] Y. Shu, Z. Dai, Z. Wu, and B. K. H. Low. Unifying and boosting gradient-based training-free neural architecture search. In *Proc. NeurIPS*, 2022.

[55] Y. Shu, W. Wang, and S. Cai. Understanding architectures learnt by cell-based neural architecture search. In *Proc. ICLR*, 2020.

[56] R. H. L. Sim, Y. Zhang, B. K. H. Low, and P. Jaillet. Collaborative Bayesian optimization with fair regret. In *Proc. ICML*, pages 9691–9701, 2021.

[57] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pages 1015–1022, 2010.

[58] S. S. Tay, C. S. Foo, D. Urano, R. C. X. Leong, and B. K. H. Low. Efficient distributionally robust Bayesian optimization with worst-case sensitivity. In *Proc. ICML*, 2022.

[59] M. Van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian processes. In *Proc. NeurIPS*, 2017.

[60] A. Verma, Z. Dai, and B. K. H. Low. Bayesian optimization under stochastic delayed feedback. In *Proc. ICML*, pages 22145–22167. PMLR, 2022.

[61] Z. Wang, P. Awasthi, C. Dann, A. Sekhari, and C. Gentile. Neural active learning with performance guarantees. *Proc. NeurIPS*, 34:7510–7521, 2021.

[62] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *Proc. AISTATS*, pages 745–754. PMLR, 2018.

[63] Z. Wu, Y. Shu, and B. K. H. Low. Davinz: Data valuation using deep neural networks at initialization. In *International Conference on Machine Learning*, pages 24150–24176. PMLR, 2022.

[64] P. Xu, Z. Wen, H. Zhao, and Q. Gu. Neural contextual bandits with deep representation and shallow exploration. arXiv:2012.01780, 2020.

[65] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[66] W. Zhang, D. Zhou, L. Li, and Q. Gu. Neural Thompson sampling. In *Proc. ICLR*, 2021.

[67] Y. Zhang, Z. Dai, and B. K. H. Low. Bayesian optimization with binary auxiliary information. In *Proc. UAI*, pages 1222–1232, 2019.

[68] Y. Zhang, T. N. Hoang, B. K. H. Low, and M. Kankanhalli. Information-based multi-fidelity Bayesian optimization. In *Proc. NIPS Workshop on Bayesian Optimization*, 2017.

[69] D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with UCB-based exploration. In *Proc. ICML*, pages 11492–11502. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Refer to Section 7.

(c) Did you discuss any potential negative societal impacts of your work? [Yes] Refer to Section 7.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] Refer to Sections 4.1 and 4.2.

(b) Did you include complete proofs of all theoretical results? [Yes] Refer to Appendices C and D.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is uploaded as part of the Supplementary Material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Refer to Appendix F.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Refer to Appendix F.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Refer to Appendix F.

(b) Did you mention the license of the assets? [Yes] Refer to Appendix F.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Refer to Appendix F.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Refer to Appendix F.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Justification for The Acquisition Functions Being Sampled from GPs

For our Algo. 1, the work of [24] has shown that if $\beta_t = 1$, then after running lines 4-7 of Algo. 1, the resulting function $f_t^i(\mathbf{x}; \theta_t^i)$ corresponds to a function sampled from the GP posterior with the NTK as the kernel function: $GP(\mu_{t-1}(\ ), \sigma_{t-1}^2(\ ,\ ))$ conditioned on the $(t \quad 1) \quad B$ observations from the first $t \quad 1$ iterations. The GP posterior mean and covariance function are expressed as:

$$\mu_{t-1}(\mathbf{x}) \ , \ \mathbf{k}_{t-1}(\mathbf{x})^\top (\mathbf{K}_{t-1} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_{t-1} \ , \tag{2}$$

$$\sigma_{t-1}^2(\mathbf{x}, \mathbf{x}') \ , \ k(\mathbf{x}, \mathbf{x}') \quad \mathbf{k}_{t-1}(\mathbf{x})^\top (\mathbf{K}_{t-1} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{t-1}(\mathbf{x}') \tag{3}$$

where $\mathbf{k}_{t-1}(\mathbf{x}) \quad , \quad (k(\mathbf{x}, \mathbf{x}_\tau^i))_{\tau=1,\ldots,t-1, i=1,\ldots,B}^\top$ which is a $(t \quad 1)B$ dimensional vector, $\mathbf{y}_t \quad , \quad (y_\tau^i)_{\tau=1,\ldots,t-1,i=1,\ldots,B}^\top$ which is also a $(t \quad 1)B$ dimensional vector, and $\mathbf{K}_t \quad , \quad (k(\mathbf{x}_\tau^i, \mathbf{x}_{\tau^0}^{i^0}))_{\tau=1,\ldots,t-1,i=1,\ldots,B;\tau^0=1,\ldots,t-1,i^0=1,\ldots,B}$ which is a $(t \quad 1)B \quad (t \quad 1)B$ dimensional squared matrix.

For our Algo. 2, when $\beta_t = 1$, because every run of the procedure in lines 4-6 corresponds to running the sample-then-optimize method [42] while treating the neural tangent features as the input features, therefore, the resulting linear function $f_t^i(\mathbf{x}; \theta_t^i)$ w.r.t. θ_t^i also corresponds to a sampled function from the GP posterior $GP(\mu_{t-1}(\ ), \sigma_{t-1}^2(\ ,\ ))$ with the NTK (if the NN is infinite-width) or empirical NTK (if the NN if finite-width) as the kernel function according to the work of [42].

Next, for both Algo. 1 and Algo. 2, when $\beta_t = 2\log(\pi^2 t^2 |X| / \delta)$, since we have multiplied the output of NN by β_t which corresponds to multiplying the gradient of the NN by β_t, therefore, the resulting NTK will be multiplied by β_t^2. Also note that we have also multiplied the noise variance σ^2 by β_t^2 in (1). As a result, after plugging these two changes into the equations for GP posterior mean (2) and variance (3), it is easy to verify that the GP posterior variance will be multiplied by β_t^2 while the GP posteior mean is unchanged.

B GP Posterior Variance with NTK

When using the NTK as the kernel function, the GP posterior variance (3) at any input $\mathbf{x}$ can be easily approximated by

$$\sigma_{t-1}^2(\mathbf{x}, \mathbf{x}) \quad r_\theta f(\mathbf{x}; \theta_0)^\top \left[\quad _{t-1} + \sigma^2 I \right]^{-1} r_\theta f(\mathbf{x}; \theta_0), \tag{4}$$

in which

$$_{t-1} = \sum_{\tau=0}^{t-1} \sum_{i=1}^{B} r_\theta f(\mathbf{x}_\tau^i; \theta_0) \, r_\theta f(\mathbf{x}_\tau^i; \theta_0)^\top, \tag{5}$$

and $\theta_0 \quad$ init( ) are randomly initialized parameters. Therefore, to run the uncertainty sampling algorithm as the initialization stage, we simply need to sequentially maximize equation (4), i.e., in iteration t of the initialization stage, we simply choose the next initial input $\mathbf{x}$ by maximizing equation (4).

C Proof of Theorem 1

Here, to simplify the analysis, we follow the work of [17] and reparameterize the iterations to view our algorithms in the sequential setting. Specifically, in the main text (Algos. 1 and 2), every B function evaluations are counted as an iteration t; however, we reparameterize the iterations such that every query selection is counted as an iteration t. That is, every time an input query is selected, we increment the number of iterations by 1. As a result, before the reparameterization, the cumulative regret is expressed as $R_T = \sum_{t=1}^{T/B} \sum_{i=1}^{B} (f(\mathbf{x}^*) \quad f(\mathbf{x}_t^i))$; after reparameterization, the same cumulative regret is now expressed as $R_T = \sum_{t=1}^{T} (f(\mathbf{x}^*) \quad f(\mathbf{x}_t))$. Note that when $B = 1$, the two parameterizations are the same. Therefore, in the entire proof in this section, we index the iterations sequentially by $1, 2, \ldots, t, t+1, \ldots, T$.

At iteration t, we use fb[t] to denote the largest iteration index whose observation has been collected. For example, if the batch size is $B = 3$, assuming that after the most recent batch of inputs

have been collected, we have in total gathered t 1 observations; then when selecting the input queries in iterations t, t + 1 and t + 2, we have that fb[t] = fb[t + 1] = fb[t + 2] = t 1, because the index of the most recent observation is fixed at t 1 since we do not collect any new observations during this process. Next, when choosing the input query at iterations t + 3, t + 4 and t + 5, we have that fb[t + 3] = fb[t + 4] = fb[t + 5] = t + 2. As a result of our reparameterization here, the requirement on the constant C from Theorem 1 should be slightly modified into: $\max_{A\subset\mathcal{X},|A|\leq B-1} \mathrm{I}(f;\mathbf{y}_A|\mathbf{y}_{1:\mathrm{fb}[t]})$ $C, 8t$ 1, where $\mathbf{y}_{1:\mathrm{fb}[t]}$ represents the output observations from iterations 1 to fb[t].

Our reparameterization mentioned above allows us to derive more general theoretical results which hold for both synchronous and asynchronous batch BO. In the setting of synchronous batch evaluations which we have focused on in the main text, $\max\{t$ $B, 0\}$ fb[t] t 1. In this setting, fb[t] is a deterministic function of t, which is determined before the algorithm starts. Of note, although we focus on the setting of synchronous batch BO in our theoretical analysis, the only requirement of our theoretical analysis on fb[t] is that t fb[t] B, i.e., the number of pending observations t fb[t] 1 should be upper-bounded by B 1. Therefore, our theoretical results also hold in the setting of asynchronous batch BO, because the number of pending observations in asynchronous batch BO is always equal to B 1 [17].

Denote as $\mu_{\mathrm{fb}[t]}$ and $\sigma_{\mathrm{fb}[t]}$ the GP posterior mean and standard deviation conditioned on the observations from iteration 1 to fb[t]. Define F_{t-1} = $\{\mathbf{x}_1, y_1, \ldots, \mathbf{x}_{\mathrm{fb}[t]}, y_{\mathrm{fb}[t]}, \mathbf{x}_{\mathrm{fb}[t]+1}, \ldots, \mathbf{x}_{t-1}\}$ as the history of selected inputs and observed outputs for those completed observations, as well as the selected inputs of those pending observations. Define β_t = $2\log(\pi^2 t^2 |\mathcal{X}|/(3\delta))$, and c_t = $\beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$.

Lemma 1. *Choose δ 2 (0, 1). Define $E^f(t)$ as the event that $|\mu_{fb[t]}(\mathbf{x})$ $f(\mathbf{x})|$ $\beta_t\sigma_{fb[t]}(\mathbf{x}), 8\mathbf{x}$ 2 $\mathcal{X}$. We have that $\mathrm{P}(E^f(t))$ 1 $\delta/2, 8t$ 1.*

The proof of Lemma 1, which follows from the proof of Lemma 5.1 of [57], makes use of our assumption that f is sampled from a GP and relies on simple applications of the concentration of Gaussian distributions and union bounds. Denote by f_t the acquisition function in iteration t, which is sampled from the GP posterior with the NTK as the kernel function: f_t $GP(\mu_{\mathrm{fb}[t]}(\ ), \beta_t^2\sigma^2_{\mathrm{fb}[t]}(\ ,\ ))$ as we have justified in Appendix A. Note that the query in iteration t is selected by $\mathbf{x}_t$ = $\arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$, which corresponds to line 8 of Algo. 1 and line 7 of Algo. 2 respectively.

Lemma 2. *Define $E^{f_t}(t)$ as the event that $|\mu_{fb[t]}(\mathbf{x})$ $f_t(\mathbf{x})|$ $\beta_t\sqrt{2\log(|\mathcal{X}|t^2)}\sigma_{fb[t]}(\mathbf{x}), 8\mathbf{x}$ 2 $\mathcal{X}$. We have that $\mathrm{P}(E^{f_t}(t))$ 1 $1/t^2, 8t$ 1.*

The proof of Lemma 2 follows from Lemma 5 of the work of [7]. Importantly, conditioned on both events $E^f(t)$ and $E^{f_t}(t)$, we have that

$$|f(\mathbf{x}) \quad f_t(\mathbf{x})| \quad c_t\sigma_{\mathrm{fb}[t]}(\mathbf{x}). \tag{6}$$

We next define the set of *saturated points*, which can be understood as the set of undesirable points in every iteration.

Definition 1. *Define the set of saturated inputs in iteration t as*

$$S_t = \{\mathbf{x} \in \mathcal{X} : \quad (\mathbf{x}) > c_t\sigma_{fb[t]}(\mathbf{x})\},$$

in which $(\mathbf{x})$ = $f(\mathbf{x}^*)$ $f(\mathbf{x})$.

An important consequence of the definition above is that $\mathbf{x}^*$ is always unsaturated, because $(\mathbf{x}^*)$ = $0 < c_t\sigma_{\mathrm{fb}[t]}(\mathbf{x}^*)$.

Lemma 3. *For any F_{t-1}, conditioned on the events $E^f(t)$, we have that $8\mathbf{x}$ 2 $\mathcal{X}$,*

$$\mathrm{P}\left(f_t(\mathbf{x}) > f(\mathbf{x})|F_{t-1}\right) \quad p, \tag{7}$$

in which p = $\frac{1}{4e\sqrt{\pi}}$.

Proof.

$$\begin{aligned}
\mathsf{P}\left(f_t(\mathbf{x}) > f(\mathbf{x}) \middle| F_{t-1}\right) &= \mathsf{P}\left(\frac{f_t(\mathbf{x}) \quad \mu_{\mathrm{fb}[t]}(\mathbf{x})}{\beta_t \sigma_{\mathrm{fb}[t]}(\mathbf{x})} > \frac{f(\mathbf{x}) \quad \mu_{\mathrm{fb}[t]}(\mathbf{x})}{\beta_t \sigma_{\mathrm{fb}[t]}(\mathbf{x})} \middle| F_{t-1}\right) \\
&\quad \mathsf{P}\left(\frac{f_t(\mathbf{x}) \quad \mu_{\mathrm{fb}[t]}(\mathbf{x})}{\beta_t \sigma_{\mathrm{fb}[t]}(\mathbf{x})} > \frac{|f(\mathbf{x}) \quad \mu_{\mathrm{fb}[t]}(\mathbf{x})|}{\beta_t \sigma_{\mathrm{fb}[t]}(\mathbf{x})} \middle| F_{t-1}\right) \\
&\overset{(a)}{\quad} \mathsf{P}\left(\frac{f_t(\mathbf{x}) \quad \mu_{\mathrm{fb}[t]}(\mathbf{x})}{\beta_t \sigma_{\mathrm{fb}[t]}(\mathbf{x})} > 1 \middle| F_{t-1}\right) \\
&\overset{(b)}{\quad} \frac{e^{-1}}{4\sqrt{\pi}}.
\end{aligned} \tag{8}$$

(a) follows from Lemma 1, which holds because we condition on the event $E^f(t)$ here. (b) follows since $f_t(\mathbf{x}) \quad \mathcal{N}(\mu_{\mathrm{fb}[t]}(\mathbf{x}), \beta_t^2 \sigma^2_{\mathrm{fb}[t]}(\mathbf{x}))$ and makes use of the Gaussian anti-concentration inequality, i.e., $\mathsf{P}(Z > a) \quad \frac{e^{-\frac{a^2}{2}}}{4\sqrt{\pi}a}$ where Z follows a standard Gaussian distribution. □

The next Lemma shows that the probability that the selected input is unsaturated (i.e., desirable according to Definition 1) can be lower-bounded.

Lemma 4. *For any F_{t-1}, conditioned on the event $E^f(t)$, we have that*

$$\mathsf{P}\left(\mathbf{x}_t \in X \setminus S_t \middle| F_{t-1}\right) \quad p \quad 1/t^2.$$

Proof. To begin with, we can lower-bounded the probability that the selected $\mathbf{x}_t$ is unsaturated as follows:

$$\mathsf{P}\left(\mathbf{x}_t \in X \setminus S_t | F_{t-1}\right) \quad \mathsf{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | F_{t-1}\right). \tag{9}$$

The inequality above holds because the event on the right hand sight implies the event on the left hand side. Specifically, because $\mathbf{x}^*$ is always unsaturated (Definition 1), therefore, as long as $f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t$, then the selected $\mathbf{x}_t$ is guaranteed to be unsaturated because it is selected as $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t(\mathbf{x})$.

Next, we assume that both events $E^f(t)$ and $E^{f_t}(t)$ holds, which allows us to derive an upper bound on $f_t(\mathbf{x})$ for all $\mathbf{x} \in S_t$:

$$f_t(\mathbf{x}) \overset{(a)}{\quad} f(\mathbf{x}) + c_t \sigma_{\mathrm{fb}[t]}(\mathbf{x}) \overset{(b)}{\quad} f(\mathbf{x}) + \quad (\mathbf{x}) = f(\mathbf{x}) + f(\mathbf{x}^*) \quad f(\mathbf{x}) = f(\mathbf{x}^*), \tag{10}$$

in which (a) results from Lemma 1 and Lemma 2 and (b) follows from Definition 1. As a result,equation (10) implies that when both both events $E^f(t)$ and $E^{f_t}(t)$ hold, we have that

$$\mathsf{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | F_{t-1}\right) \quad \mathsf{P}\left(f_t(\mathbf{x}^*) > f(\mathbf{x}^*) | F_{t-1}\right). \tag{11}$$

Next, combining equations (9) and (11) and separately considering the cases where the event $E^{f_t}(t)$ is true or false, we have that

$$\begin{aligned}
\mathsf{P}\left(\mathbf{x}_t \in X \setminus S_t | F_{t-1}\right) &\quad \mathsf{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | F_{t-1}\right) \\
&\overset{(a)}{\quad} \mathsf{P}\left(f_t(\mathbf{x}^*) > f(\mathbf{x}^*) | F_{t-1}\right) \quad \mathsf{P}\left(\overline{E^{f_t}(t)} | F_{t-1}\right) \\
&\overset{(b)}{\quad} p \quad 1/t^2.
\end{aligned} \tag{12}$$

This completes the proof. □

We use $\sigma_{t-1}(\ )$ to represent the GP posterior standard deviation conditioned on all selected input queries from iterations 1 to $t \quad 1$.

Lemma 5. *We have for all $t \quad 1$ and all $\mathbf{x} \in X$ that*

$$\frac{\sigma_{\mathrm{fb}[t]}(\mathbf{x})}{\sigma_{t-1}(\mathbf{x})} \quad e^C.$$

Proof. We use $\mathbf{y}_{1:\text{fb}[t]}$ to denote the output observations from iterations 1 to fb$[t]$, use $\mathbf{y}_{\text{fb}[t]+1:t-1}$ to represent the the output observations from iterations fb$[t]$ + 1 to t 1, and use $\mathbf{y}_A$ to denote the vector of observations at a set of inputs A X. We use $H(\ )$ to represent the entroy of a random variable.

To begin with, we establish the relationship between the following conditional information gain and the ratio of GP posterior standard deviations which we intend to upper-bound:

$$\begin{aligned} \mathsf{I}(f(\mathbf{x}); \mathbf{y}_{\text{fb}[t]+1:t-1} | \mathbf{y}_{1:\text{fb}[t]}) &= H(f(\mathbf{x}) | \mathbf{y}_{1:\text{fb}[t]}) \quad H(f(\mathbf{x}) | \mathbf{y}_{1:t-1})) \\ &= \frac{1}{2} \log(2\pi e \sigma^2_{\text{fb}[t]}(\mathbf{x})) \quad \frac{1}{2} \log(2\pi e \sigma^2_{t-1}(\mathbf{x})) \\ &= \log \frac{\sigma_{\text{fb}[t]}(\mathbf{x})}{\sigma_{t-1}(\mathbf{x})}. \end{aligned} \tag{13}$$

The first equality comes from the definition of conditional information gain and the second equality follows immediately from the entroy of Gaussian random variables. The equation above allows us to upper-bound the ratio of GP posterior standard deviations as follows:

$$\begin{aligned} \frac{\sigma_{\text{fb}[t]}(\mathbf{x})}{\sigma_{t-1}(\mathbf{x})} &= \exp(\mathsf{I}(f(\mathbf{x}); \mathbf{y}_{\text{fb}[t]+1:t-1} | \mathbf{y}_{1:\text{fb}[t]})) \\ &\overset{(a)}{} \exp(\mathsf{I}(f; \mathbf{y}_{\text{fb}[t]+1:t-1} | \mathbf{y}_{1:\text{fb}[t]})) \\ &\overset{(b)}{} \exp\Big(\max_{A \subset \mathcal{X}, |A| \leq B-1} \mathsf{I}(f; \mathbf{y}_A | \mathbf{y}_{1:\text{fb}[t]}) \Big) \overset{(c)}{} e^C, \end{aligned} \tag{14}$$

in which (a) is because the information gain about f is larger than that of $f(\mathbf{x})$, (b) follow since the size of $\mathbf{y}_{\text{fb}[t]+1:t-1}$ is at most B 1 in our batch setting with a batch size of B, and (c) is a result of the definition of the constant C. This completes the proof. □

Next, we are ready to prove an upper bound on the expected instantaneous regret $r_t = f(\mathbf{x}^*)$ $f(\mathbf{x}_t)$.

Lemma 6. *For any $\mathcal{F}_{t-1}$, conditioned on the event $E^f(t)$, we have that*

$$\mathsf{E}\left[r_t | \mathcal{F}_{t-1}\right] \quad c_t e^C \left(1 + \frac{10}{p}\right) \mathsf{E}\left[\sigma_{t-1}(\mathbf{x}_t) | \mathcal{F}_{t-1}\right] + \frac{2B'}{t^2}.$$

Proof. To begin with, define

$$\overline{\mathbf{x}}_t \ , \ \arg\min_{\mathbf{x} \in \mathcal{X} \setminus S_t} \sigma_{\text{fb}[t]}(\mathbf{x}). \tag{15}$$

That is, $\overline{\mathbf{x}}_t$ is the unsaturated input with the smallest GP posterior standard deviation. Note that given a $\mathcal{F}_{t-1}$, $\overline{\mathbf{x}}_t$ is deterministic. Next, we have that

$$\begin{aligned} \mathsf{E}[\sigma_{\text{fb}[t]}(\mathbf{x}_t) | \mathcal{F}_{t-1}] \quad & \mathsf{E}\left[\sigma_{\text{fb}[t]}(\mathbf{x}_t) | \mathcal{F}_{t-1}, \mathbf{x}_t \ 2 \ X \ n \ S_t\right] \mathsf{P}\left(\mathbf{x}_t \ 2 \ X \ n \ S_t | \mathcal{F}_{t-1}\right) \\ & \sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t)(p \quad 1/t^2), \end{aligned} \tag{16}$$

where the second inequality makes use of Lemma 4, which holds here because we have also conditioned on the event $E^f(t)$ in Lemma 4. Next, conditioned on both $E^f(t)$ and $E^{f_t}(t)$, we have that

$$\begin{aligned} r_t = f(\mathbf{x}^*) \quad f(\mathbf{x}_t) &= f(\mathbf{x}^*) \quad f(\overline{\mathbf{x}}_t) + f(\overline{\mathbf{x}}_t) \quad f(\mathbf{x}_t) \\ &\overset{(a)}{} (\overline{\mathbf{x}}_t) + f_t(\overline{\mathbf{x}}_t) + c_t \sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) \quad f_t(\mathbf{x}_t) + c_t \sigma_{\text{fb}[t]}(\mathbf{x}_t) \\ &\overset{(b)}{} c_t \sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + c_t \sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + c_t \sigma_{\text{fb}[t]}(\mathbf{x}_t) + f_t(\overline{\mathbf{x}}_t) \quad f_t(\mathbf{x}_t) \\ &\overset{(c)}{} c_t \left(2\sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + \sigma_{\text{fb}[t]}(\mathbf{x}_t)\right), \end{aligned} \tag{17}$$

in which (a) makes use of Lemma 1 and Lemma 2, (b) follows since $\overline{\mathbf{x}}_t$ is unsaturated, and (c) follows from the way in which $\mathbf{x}_t$ is selected: $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t(\mathbf{x})$. Next, the expected value of

r_t can be upper-bounded as follows:

$$\begin{aligned}
\mathsf{E}\left[r_t | F_{t-1}\right] &\leq \mathsf{E}\left[c_t\left(2\sigma_{\mathrm{fb}[t]}(\overline{\mathbf{x}}_t) + \sigma_{\mathrm{fb}[t]}(\mathbf{x}_t)\right) | F_{t-1}\right] + 2B'\mathsf{P}\left(\overline{E^{f_t}(t)} | F_{t-1}\right) \\
&\overset{(a)}{\leq} \mathsf{E}\left[c_t\left(\frac{2}{p - 1/t^2}\sigma_{\mathrm{fb}[t]}(\mathbf{x}_t) + \sigma_{\mathrm{fb}[t]}(\mathbf{x}_t)\right) | F_{t-1}\right] + \frac{2B'}{t^2} \\
&= \mathsf{E}\left[c_t\left(1 + \frac{2}{p - 1/t^2}\right)\sigma_{\mathrm{fb}[t]}(\mathbf{x}_t) | F_{t-1}\right] + \frac{2B'}{t^2} \\
&\overset{(b)}{\leq} c_t\left(1 + \frac{2}{p - 1/t^2}\right)\mathsf{E}\left[e^C\sigma_{t-1}(\mathbf{x}_t) | F_{t-1}\right] + \frac{2B'}{t^2} \\
&\overset{(c)}{\leq} c_t e^C\left(1 + \frac{10}{p}\right)\mathsf{E}\left[\sigma_{t-1}(\mathbf{x}_t) | F_{t-1}\right] + \frac{2B'}{t^2},
\end{aligned} \tag{18}$$

where (a) follows from equation (16) and (b) makes use of Lemma 5. (c) follows since $2/(p - 1/t^2) \leq 10/p$, which holds because (i) $p - 1/t^2 < 0$ for $t < 5$, (ii) $2/(p - 1/t^2) \leq 10/p$ for $t = 5$, and (iii) $2/(p - 1/t^2)$ is decreasing as t increases when $t \geq 5$.

□

Definition 2. *Define $Y_0 = 0$, and for all $t = 1, \ldots, T$,*

$$\begin{aligned}
\overline{r}_t &= r_t \mathbb{I}\{E^f(t)\}, \\
X_t &= \overline{r}_t - c_t e^C\left(1 + \frac{10}{p}\right)\sigma_{t-1}(\mathbf{x}_t) - \frac{2B'}{t^2} \\
Y_t &= \sum_{s=1}^{t} X_s.
\end{aligned}$$

Lemma 7. *Conditioned on the event $E^f(t)$, $(Y_t : t = 0, \ldots, T)$ is a super-martingale with respect to the filtration F_t.*

Proof.

$$\begin{aligned}
\mathsf{E}[Y_t - Y_{t-1} | F_{t-1}] &= \mathsf{E}[X_t | F_{t-1}] \\
&= \mathsf{E}[\overline{r}_t - c_t e^C\left(1 + \frac{10}{p}\right)\sigma_{t-1}(\mathbf{x}_t) - \frac{2B'}{t^2} | F_{t-1}] \\
&= \mathsf{E}[\overline{r}_t | F_{t-1}] - \left(c_t e^C\left(1 + \frac{10}{p}\right)\mathsf{E}[\sigma_{t-1}(\mathbf{x}_t) | F_{t-1}] + \frac{2B'}{t^2}\right) \leq 0.
\end{aligned} \tag{19}$$

If the event $E^f(t)$ holds, then $\overline{r}_t = r_t$ and the inequality follows from Lemma 6. If $E^f(t)$ does not hold, $\overline{r}_t = 0$ and the inequality holds trivially. □

Lastly, we can apply the Azuma-Hoeffding's inequality to the martingale $(Y_t : t = 0, \ldots, T)$ to derive the upper bound on the cumulative regret R_T.

Lemma 8. *Define $C_1 := \frac{2}{\log(1+\sigma^{-2})}$. With probability of $1 - \delta$, we have that*

$$R_T \leq c_T e^C\left(1 + \frac{10}{p}\right)\sqrt{C_1 T \gamma_T} + \frac{B'\pi^2}{3} + \left[4B' + c_T e^C\left(1 + \frac{10}{p}\right)K_0\right]\sqrt{2T\log(2/\delta)}. \tag{20}$$

Proof. To begin with, note that

$$\begin{aligned}
|Y_t - Y_{t-1}| = |X_t| &\leq |\overline{r}_t| + c_t e^C\left(1 + \frac{10}{p}\right)\sigma_{t-1}(\mathbf{x}_t) + \frac{2B'}{t^2} \\
&\leq 2B' + c_t e^C\left(1 + \frac{10}{p}\right)K_0 + 2B' \\
&= 4B' + c_t e^C\left(1 + \frac{10}{p}\right)K_0,
\end{aligned} \tag{21}$$

where we have made use of our assumption that $(\mathbf{x}, \mathbf{x}') \quad K_0$ in the second inequality. Next, applying the Azuma-Hoeffding's inequality to $(Y_t : t = 0, \ldots, T)$ with a probability of $\delta/2$, we have with probability $\quad 1 \quad \delta/2$ that

$$\sum_{t=1}^{T} \overline{r}_t \quad \sum_{t=1}^{T} c_t e^C \left(1 + \frac{10}{p}\right) \sigma_{t-1}(\mathbf{x}_t) + \sum_{t=1}^{T} \frac{2B'}{t^2} + \sqrt{2 \log(2/\delta) \sum_{t=1}^{T} \left(4B' + c_t e^C (1 + \frac{10}{p}) K_0\right)^2}$$

$$\overset{(a)}{} c_T e^C \left(1 + \frac{10}{p}\right) \sum_{t=1}^{T} \sigma_{t-1}(\mathbf{x}_t) + \frac{B'\pi^2}{3} + \left[4B' + c_T e^C \left(1 + \frac{10}{p}\right) K_0\right] \sqrt{2T \log(2/\delta)}$$

$$\overset{(b)}{} c_T e^C \left(1 + \frac{10}{p}\right) \sqrt{C_1 T \gamma_T} + \frac{B'\pi^2}{3} + \left[4B' + c_T e^C \left(1 + \frac{10}{p}\right) K_0\right] \sqrt{2T \log(2/\delta)}. \tag{22}$$

(a) follows since c_t is increasing in t, and (b) follows from the proof of Lemma 5.4 in the work of [57]. Next, note that $\overline{r}_t = r_t, \forall t \quad 1$ with probability of $\quad 1 \quad \delta/2$ according to Lemma 1. Therefore, the upper bound derived in the equation above is an upper bound on $R_T = \sum_{t=1}^{T} r_t$ (with probability of $\quad 1 \quad \delta$), and the proof is completed.

□

Note that $c_T = O(\log^2 T)$. From Lemma 8, we have that

$$R_T = O\left(e^C (\log^2 T) \sqrt{p}\, \overline{T} \left(1 + \sqrt{p}\, \overline{\gamma_T}\right)\right) = \widetilde{O}\left(e^C \sqrt{p}\, \overline{T} \left(1 + \sqrt{p}\, \overline{\gamma_T}\right)\right). \tag{23}$$

D Proof of Theorem 2

In this section, the main technical challenge is to rigorously account for the mismatch between the kernel with which we assume the objective function f is sampled (i.e., the exact NTK $\quad$) and the kernel with which the acquisition function is sampled (i.e., the empirical NTK $\widetilde{\quad}$). For ease of exposition, we use k and $\widetilde{k}$ (instead of $\quad$ and $\widetilde{\quad}$) to represent the exact and empirical NTK in the proof in this section. Similarly, we also use $\widetilde{\quad}$ to indicate that a term is associated with the empirical NTK $\widetilde{k}$. For example, we use $\widetilde{\mu}_{\text{fb}[t]}(\ )$ and $\widetilde{\sigma}^2_{\text{fb}[t]}(\ )$ to represent the GP posterior mean and variance calculated using the empirical NTK $\widetilde{k}$.

For simplicity, we assume that the event in Proposition 1 holds throughout the entire proof, which happens with probability of $\quad 1 \quad \delta/4$. That is, the approximation error between exact and empirical NTKs is bounded:

$$|\widetilde{k}(\mathbf{x}, \mathbf{x}') \quad k(\mathbf{x}, \mathbf{x}')| = \left|\langle \nabla_\theta f(\mathbf{x}, \widetilde{\theta}),\ \nabla_\theta f(\mathbf{x}', \widetilde{\theta})\rangle \qquad (\mathbf{x}, \mathbf{x}')\right| \quad (L+1)\varepsilon, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \tag{24}$$

To begin with, we use the following lemma to bound the difference between the GP posterior standard deviations calculated using the exact and empirical NTKs. Here, to simplify the derivations and results, we assume that $(L+1)\varepsilon \quad 1$ and $\sigma^2 \quad 1$. Note that these assumptions are not essential to the proof but are only used get cleaner expressions. Here, again for ease of expositions, we define $\hat{K}_0 \triangleq \max\{1, K_0\}$, and $\hat{K}_0^2 \triangleq \max\{1, K_0^2\}$.

Lemma 9. *We have* $\forall t \quad 1, \forall \mathbf{x} \in \mathcal{X}$ *that*

$$|\sigma_{\text{fb}[t]}(\mathbf{x}) \quad \widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})| \quad \sqrt{(L+1)\varepsilon \left(1 + \frac{4\hat{K}_0^2 t^2}{\sigma^4}\right)}.$$

Proof. Denote by $\mathbf{K}_t$ the $\text{fb}[t] \quad \text{fb}[t]$-dimensional gram matrix of exact NTK covariance values calculated using all $\text{fb}[t]$ observations up to iteration $\text{fb}[t]$, and use $\widetilde{\mathbf{K}}_t$ to represent the corresponding

gram matrix calculated using the empirical NTK $\widetilde{k}$. If we define $A = (K_t + \sigma^2)^{-1}$ or $A = (\widetilde{K}_t + \sigma^2)^{-1}$, then for both values of A, we have that

$$\|A\|_2 = \sqrt{\max[\mathrm{eig}(A^\top A)]} = \sqrt{\max[\mathrm{eig}(A)^2]} \quad \frac{1}{\sigma^2}. \tag{25}$$

This allows us to derive the following equation:

$$\begin{aligned} &\left\|(K_t + \sigma^2)^{-1} \quad (\widetilde{K}_t + \sigma^2)^{-1}\right\|_2 \quad \left\|(K_t + \sigma^2)^{-1}\right\|_2 \left\|(\widetilde{K}_t + \sigma^2)^{-1}\right\|_2 \left\|K_t \quad \widetilde{K}_t\right\|_2 \\ &\qquad\qquad \frac{1}{\sigma^2} \quad \frac{1}{\sigma^2} \quad t(L+1)\varepsilon = \frac{t(L+1)\varepsilon}{\sigma^4}. \end{aligned} \tag{26}$$

Define the fb$[t]$-dimensional vectors $k_t(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_\tau)]_{\tau=1,\ldots,\mathrm{fb}[t]}$ and $\widetilde{k}_t(\mathbf{x}) = [\widetilde{k}(\mathbf{x}, \mathbf{x}_\tau)]_{\tau=1,\ldots,\mathrm{fb}[t]}$. Then making use of the approximation guarantee from equation (24), we define $\widetilde{k}_t(\mathbf{x}) = k_t(\mathbf{x}) + (L+1)\varepsilon \ (\mathbf{x})$, where $(\mathbf{x})$ is an fb$[t]$ dimensional vector where every element satisfies $|(\mathbf{x})_i| \leq 1, \forall i \in [\mathrm{fb}[t]]$. Now we can use these definitions to derive the following upper bound.

$$\begin{aligned} &|\sigma^2_{\mathrm{fb}[t]}(\mathbf{x}) \quad \widetilde{\sigma}^2_{\mathrm{fb}[t]}(\mathbf{x})| = |k(\mathbf{x},\mathbf{x}) \quad k_t(\mathbf{x})^\top (K_t + \sigma^2 I)^{-1} k_t(\mathbf{x}) \\ &\qquad \widetilde{k}(\mathbf{x},\mathbf{x}) + \widetilde{k}_t(\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1} \widetilde{k}_t(\mathbf{x})| \\ &\quad |k(\mathbf{x},\mathbf{x}) \quad \widetilde{k}(\mathbf{x},\mathbf{x})| + |k_t(\mathbf{x})^\top (K_t + \sigma^2 I)^{-1} k_t(\mathbf{x}) \quad \widetilde{k}_t(\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1} \widetilde{k}_t(\mathbf{x})| \\ &\quad (L+1)\varepsilon + \left| k_t(\mathbf{x})^\top \left((K_t + \sigma^2 I)^{-1} \quad (\widetilde{K}_t + \sigma^2 I)^{-1} \right) k_t(\mathbf{x}) \right| \\ &\quad 2(L+1)\varepsilon \ \ (\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1} k_t(\mathbf{x}) \quad (L+1)^2\varepsilon^2 \ \ (\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1} \ \ (\mathbf{x})| \\ &\quad (L+1)\varepsilon + \|k_t(\mathbf{x})\|_2 \left\|(K_t + \sigma^2)^{-1} \quad (\widetilde{K}_t + \sigma^2)^{-1}\right\|_2 \|k_t(\mathbf{x})\|_2 + \\ &2(L+1)\varepsilon \| \ \ (\mathbf{x})\|_2 \left\|(\widetilde{K}_t + \sigma^2)^{-1}\right\|_2 \|k_t(\mathbf{x})\|_2 + (L+1)^2\varepsilon^2 \| \ \ (\mathbf{x})\|_2 \left\|(\widetilde{K}_t + \sigma^2 I)^{-1}\right\|_2 \| \ \ (\mathbf{x}\|_2) \\ &\quad (L+1)\varepsilon + K_0 \sqrt{t} \frac{t(L+1)\varepsilon}{\sigma^4} K_0 \sqrt{t} + 2(L+1)\varepsilon \sqrt{t} \frac{1}{\sigma^2} K_0 \sqrt{t} + (L+1)^2\varepsilon^2 \sqrt{t} \frac{1}{\sigma^2} \sqrt{t} \\ &= (L+1)\varepsilon + K_0^2 \frac{t^2(L+1)\varepsilon}{\sigma^4} + 2K_0(L+1)\varepsilon \frac{t}{\sigma^2} + (L+1)^2\varepsilon^2 \frac{t}{\sigma^2} \\ &\quad (L+1)\varepsilon + 4\hat{K}_0^2 \frac{t^2(L+1)\varepsilon}{\sigma^4} \\ &\quad (L+1)\varepsilon \left(1 + \frac{4\hat{K}_0^2 t^2}{\sigma^4}\right). \end{aligned} \tag{27}$$

Elementary calculation tells us that for $a, b, c > 0$, if $a^2 \quad b^2 \quad c^2$, then $a \quad \sqrt{b^2 + c^2} \quad b + c$, which leads to $a \quad b \quad c$. As a result, the equation above tells us that $|\sigma_{\mathrm{fb}[t]}(\mathbf{x}) \quad \widetilde{\sigma}_{\mathrm{fb}[t]}(\mathbf{x})| \quad \sqrt{(L+1)\varepsilon \left(1 + \frac{4\hat{K}_0^2 t^2}{\sigma^4}\right)}$. □

The next Lemma gives an upper bound on the difference between the GP posterior means calculated using the exact and empirical NTKs.

Lemma 10. *With probability of* $1 \quad \delta/4$*, we have* $\forall t \quad 1, \forall \mathbf{x} \in \mathcal{X}$ *that*

$$|\mu_{fb[t]}(\mathbf{x}) \quad \widetilde{\mu}_{fb[t]}(\mathbf{x})| \quad 2\hat{K}_0 \frac{t^2(L+1)\varepsilon}{\sigma^4} \left(B' + \sigma\sqrt{2\log(4T/\delta)}\right).$$

Proof. Define $\mathbf{y}_t = [y_\tau]_{\tau=1,\ldots,\mathrm{fb}[t]}$. We have that $y_\tau = f(\mathbf{x}_\tau) + \epsilon$ where $\epsilon \quad \mathcal{N}(0, \sigma^2)$. Standard Gaussian concentration tells us that $|\epsilon| \quad z\sigma$ with probability of $1 \quad \exp(\quad z^2/2)$. Substituting $z = \sqrt{2\log(4T/\delta)}$ and making use of the assumption that $|f(\mathbf{x})| \quad B', \forall \mathbf{x} \in \mathcal{X}$, we have that $|y_\tau| \quad B' + \sigma\sqrt{2\log(4T/\delta)}$ with probability of $1 \quad \delta/(4T)$. Now taking a union bound over all T

iterations, we have that $|y_\tau| \leq B' + \sigma\sqrt{2\log(4T/\delta)}, \forall \tau = 1, \ldots, T$ with probability of $\geq 1 - \delta/4$. This further implies that $\|\mathbf{y}_t\|_2 = \sqrt{\sum_{\tau=1}^{\text{fb}[t]} y_\tau^2} \leq \sqrt{t}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right)$. Now we are ready to bound the term in question:

$$
\begin{aligned}
&|\mu_{\text{fb}[t]}(\mathbf{x}) - \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})| = |\boldsymbol{k}_t(\mathbf{x})^\top (K_t + \sigma^2 I)^{-1}\mathbf{y}_t - \widetilde{\boldsymbol{k}}_t(\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1}\mathbf{y}_t| \\
&= |\boldsymbol{k}_t(\mathbf{x})^\top (K_t + \sigma^2 I)^{-1}\mathbf{y}_t - \boldsymbol{k}_t(\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1}\mathbf{y}_t + (L+1)\varepsilon - (\mathbf{x})^\top (\widetilde{K}_t + \sigma^2 I)^{-1}\mathbf{y}_t| \\
&\leq \|\boldsymbol{k}_t(\mathbf{x})\|_2 \left\|(K_t + \sigma^2 I)^{-1} - (\widetilde{K}_t + \sigma^2 I)^{-1}\right\| \|\mathbf{y}_t\|_2 + \|(\widetilde{K}_t + \sigma^2 I)^{-1}\|_2 \|\mathbf{y}_t\|_2 \\
&\leq K_0 \sqrt{t}\frac{t(L+1)\varepsilon}{\sigma^4}\sqrt{t}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + (L+1)\varepsilon\sqrt{t}\frac{1}{\sigma^2}\sqrt{t}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) \\
&\leq 2K_0\frac{t^2(L+1)\varepsilon}{\sigma^4}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right).
\end{aligned}
\tag{28}
$$

□

Next, similar to the proof in Appendix C, here we also need a lemma showing the concentration of the function f. Define $\beta_t = 2\log(2\pi^2 t^2 |\mathcal{X}|/(3\delta))$, and $c_t = \beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$. Note that the value of β_t defined here is slightly different due to the use of different error probabilities (i.e., we have used an error probability of $\delta/2$ in the proof in Appendix C yet $\delta/4$ in this section).

Lemma 11. *$|\mu_{fb[t]}(\mathbf{x}) - f(\mathbf{x})| \leq \beta_t\sigma_{fb[t]}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$, with probability of $\geq 1 - \delta/4, \forall t \geq 1$.*

The proof of Lemma 11 is the same as that of Lemma 1. The next Lemma proves the concentration of the objective function f around the GP posterior mean calculated using the empirical NTK $\widetilde{k}$, which consists of an additional error term $\epsilon_{m,t}$ due to the use of the empirical NTK compared with Lemma 11 above.

Lemma 12. *Define*

$$
\epsilon_{m,t} := 2K_0\frac{t^2(L+1)\varepsilon}{\sigma^4}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + \beta_t\sqrt{(L+1)\varepsilon\left(1 + \frac{4K_0^2 t^2}{\sigma^4}\right)}.
$$

Define $E^{\widetilde{f}}(t)$ as the event that $|\widetilde{\mu}_{fb[t]}(\mathbf{x}) - f(\mathbf{x})| \leq \beta_t\widetilde{\sigma}_{fb[t]}(\mathbf{x}) + \epsilon_{m,t}, \forall \mathbf{x} \in \mathcal{X}$. We have that $\mathbb{P}(E^{\widetilde{f}}(t)) \geq 1 - \delta/2, \forall t \geq 1$.

Proof.

$$
\begin{aligned}
&|\widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}) - f(\mathbf{x})| \leq |\widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}) - \mu_{\text{fb}[t]}(\mathbf{x})| + |\mu_{\text{fb}[t]}(\mathbf{x}) - f(\mathbf{x})| \\
&\overset{(a)}{\leq} 2K_0\frac{t^2(L+1)\varepsilon}{\sigma^4}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + \beta_t\sigma_{\text{fb}[t]}(\mathbf{x}) \\
&\overset{(b)}{\leq} 2K_0\frac{t^2(L+1)\varepsilon}{\sigma^4}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + \beta_t\left(\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + \sqrt{(L+1)\varepsilon\left(1 + \frac{4K_0^2 t^2}{\sigma^4}\right)}\right) \\
&= \beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + 2K_0\frac{t^2(L+1)\varepsilon}{\sigma^4}\left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + \beta_t\sqrt{(L+1)\varepsilon\left(1 + \frac{4K_0^2 t^2}{\sigma^4}\right)} \\
&= \beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m,t}
\end{aligned}
\tag{29}
$$

(a) follows from Lemma 10 and Lemma 11 and hence holds with probability of $\geq 1 - \delta/4 - \delta/4 = 1 - \delta/2$, and (b) results from Lemma 9. □

Denote by $\widetilde{f}_t$ the sampled function in iteration t using the empirical NTK, i.e., $\widetilde{f}_t \sim GP(\widetilde{\mu}_{\text{fb}[t]}(\cdot), \beta_t^2\widetilde{\sigma}^2_{\text{fb}[t]}(\cdot, \cdot))$.

Lemma 13. *Define $E^{\widetilde{f}_t}(t)$ as the event that $j\widetilde{\mu}_{fb[t]}(\mathbf{x}) \quad \widetilde{f}_t(\mathbf{x})j \quad \beta_t\sqrt{2\log(jXjt^2)}\widetilde{\sigma}_{fb[t]}(\mathbf{x}), 8\mathbf{x}\ 2\ X$. We have that $\mathsf{P}(E^{\widetilde{f}_t}(t)) \quad 1 \quad 1/t^2, 8t \quad 1$.*

Lemma 13 is the counterpart to Lemma 2 in Appendix C and can be proved using the same techniques. Of note, conditioned on both events $E^{\widetilde{f}}(t)$ and $E^{\widetilde{f}_t}(t)$, we have that

$$\begin{aligned} jf(\mathbf{x}) \quad \widetilde{f}_t(\mathbf{x})j \quad jf(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})j + j\widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}) \quad \widetilde{f}_t(\mathbf{x})j \\ \beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m,t} + \beta_t\sqrt{2\log(jXjt^2)}\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) \\ = c_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m,t}. \end{aligned} \tag{30}$$

Next, we similarly define the set of saturated inputs.

Definition 3. *Define the set of saturated inputs in iteration t as*

$$S_t = \{\mathbf{x}\ 2\ X : \quad (\mathbf{x}) > c_t\widetilde{\sigma}_{fb[t]}(\mathbf{x}) + 2\epsilon_{m,t}\},$$

in which $(\mathbf{x}) = f(\mathbf{x}^*) \quad f(\mathbf{x})$.

Again, $\mathbf{x}^*$ is always unsaturated.

Lemma 14. *For any F_{t-1}, conditioned on the events $E^{\widetilde{f}}(t)$, we have that $8\mathbf{x}\ 2\ X$,*

$$\mathsf{P}\left(\widetilde{f}_t(\mathbf{x}) + \epsilon_{m,t} > f(\mathbf{x})jF_{t-1}\right) \quad p, \tag{31}$$

in which $p = \frac{1}{4e\sqrt{\pi}}$.

Proof.

$$\begin{aligned} \mathsf{P}\left(\widetilde{f}_t(\mathbf{x}) + \epsilon_{m.t} > f(\mathbf{x})jF_{t-1}\right) &= \mathsf{P}\left(\frac{\widetilde{f}_t(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m.t}}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})} > \frac{f(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})}\Big|F_{t-1}\right) \\ &\quad \mathsf{P}\left(\frac{\widetilde{f}_t(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m.t}}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})} > \frac{jf(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})j}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})}\Big|F_{t-1}\right) \\ &= \mathsf{P}\left(\frac{\widetilde{f}_t(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})} > \frac{jf(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})j \quad \epsilon_{m.t}}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})}\Big|F_{t-1}\right) \\ &\overset{(a)}{} \mathsf{P}\left(\frac{\widetilde{f}_t(\mathbf{x}) \quad \widetilde{\mu}_{\text{fb}[t]}(\mathbf{x})}{\beta_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x})} > 1\Big|F_{t-1}\right) \\ &\overset{(b)}{} \frac{\exp(\quad 1)}{4\sqrt{\pi}}. \end{aligned} \tag{32}$$

(a) follows from Lemma 12, and (b) follows because $\widetilde{f}_t(\mathbf{x}) \quad N(\widetilde{\mu}_{\text{fb}[t]}(\mathbf{x}), \beta_t^2\widetilde{\sigma}^2_{\text{fb}[t]}(\mathbf{x}))$ and makes use of the Gaussian anti-concentration inequality.

□

Next, we again prove a lower bound on the probability that the selected input is unsaturated.

Lemma 15. *For any F_{t-1}, conditioned on the event $E^{\widetilde{f}}(t)$, we have that*

$$\mathsf{P}\left(\mathbf{x}_t\ 2\ X\ n\ S_t, jF_{t-1}\right) \quad p \quad 1/t^2.$$

Proof. The proof here follows similar steps as the proof of Lemma 4. To begin with, we have the following relationship.

$$\mathsf{P}\left(\mathbf{x}_t\ 2\ X\ n\ S_tjF_{t-1}\right) \quad \mathsf{P}\left(\widetilde{f}_t(\mathbf{x}^*) > \widetilde{f}_t(\mathbf{x}), 8\mathbf{x}\ 2\ S_tjF_{t-1}\right), \tag{33}$$

The validity of this equation can be justified in a similar way as equation (9) in the proof of Lemma 4, i.e., the event on the right hand sight implies the event on the left hand side.

Next, we assume that both events $E^{\widetilde{f}}(t)$ and $E^{\widetilde{f}}(t)$ are true, which allows us to derive an upper bound on $\widetilde{f}_t(\mathbf{x})$ for all $\mathbf{x}$ 2 S_t:

$$\widetilde{f}_t(\mathbf{x}) \overset{(a)}{} f(\mathbf{x}) + c_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}) + \epsilon_{m,t} \overset{(b)}{} f(\mathbf{x}) + \quad (\mathbf{x}) \quad \epsilon_{m,t} = f(\mathbf{x}^*) \quad \epsilon_{m,t}, \tag{34}$$

in which (a) follows from Lemmas 12 and 13, and (b) is a result of Definition 3.

Therefore, (34) implies that when both both events $E^{\widetilde{f}}(t)$ and $E^{\widetilde{f}_t}(t)$ hold,

$$\mathsf{P}\left(\widetilde{f}_t(\mathbf{x}^*) > \widetilde{f}_t(\mathbf{x}), 8\mathbf{x}\ 2\ S_t | F_{t-1}\right) \quad \mathsf{P}\left(\widetilde{f}_t(\mathbf{x}^*) > f(\mathbf{x}^*) \quad \epsilon_{m,t} | F_{t-1}\right). \tag{35}$$

Next, we can show that

$$\begin{aligned}\mathsf{P}\left(\mathbf{x}_t\ 2\ X\ n\ S_t | F_{t-1}\right) \quad & \mathsf{P}\left(\widetilde{f}_t(\mathbf{x}^*) > \widetilde{f}_t(\mathbf{x}), 8\mathbf{x}\ 2\ S_t | F_{t-1}\right) \\ & \mathsf{P}\left(\widetilde{f}_t(\mathbf{x}^*) > f(\mathbf{x}^*) \quad \epsilon_{m,t} | F_{t-1}\right) \quad \mathsf{P}\left(\overline{E^{\widetilde{f}_t}(t)} | F_{t-1}\right) \\ & p \quad 1/t^2,\end{aligned} \tag{36}$$

where the last inequality makes use of Lemma 14. □

The next Lemma derives an upper bound on the expected instantaneous regret $r_t = f(\mathbf{x}^*) \quad f(\mathbf{x}_t)$.

Lemma 16. *For any F_{t-1}, conditioned on the event $E^{\widetilde{f}}(t)$, we have that*

$$\mathsf{E}\left[r_t | F_{t-1}\right] \quad c_t e^C \left(1 + \frac{10}{p}\right) \mathsf{E}\left[\sigma_{t-1}(\mathbf{x}_t) | F_{t-1}\right] + \epsilon'_{m,t} + \frac{2B'}{t^2},$$

where

$$\epsilon'_{m,t} \ , \ 3c_t\sqrt{(L+1)\varepsilon\left(1 + \frac{4\hat{K}_0^2 t^2}{\sigma^4}\right)} + 4\epsilon_{m,t}. \tag{37}$$

Proof. Define

$$\overline{\mathbf{x}}_t \ , \ \arg\min_{\mathbf{x}\in\mathcal{X}\setminus S_t}\sigma_{\text{fb}[t]}(\mathbf{x}). \tag{38}$$

Note that given a F_{t-1}, $\overline{\mathbf{x}}_t$ is deterministic. Next, this definiton also leads to:

$$\begin{aligned}\mathsf{E}[\sigma_{\text{fb}[t]}(\mathbf{x}_t) | F_{t-1}] \quad & \mathsf{E}\left[\sigma_{\text{fb}[t]}(\mathbf{x}_t) | F_{t-1}, \mathbf{x}_t\ 2\ X\ n\ S_t\right] \mathsf{P}\left(\mathbf{x}_t\ 2\ X\ n\ S_t | F_{t-1}\right) \\ & \sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t)(p \quad 1/t^2),\end{aligned} \tag{39}$$

where the last inequality makes use of Lemma 15.

Next, conditioned on both $E^{\widetilde{f}}(t)$ and $E^{\widetilde{f}_t}(t)$, we have that

$$\begin{aligned} r_t &= f(\mathbf{x}^*) \quad f(\mathbf{x}_t) = f(\mathbf{x}^*) \quad f(\overline{\mathbf{x}}_t) + f(\overline{\mathbf{x}}_t) \quad f(\mathbf{x}_t) \\ &\overset{(a)}{} (\overline{\mathbf{x}}_t) + \widetilde{f}_t(\overline{\mathbf{x}}_t) + c_t\widetilde{\sigma}_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + \epsilon_{m,t} \quad \widetilde{f}_t(\mathbf{x}_t) + c_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}_t) + \epsilon_{m,t} \\ &\overset{(b)}{} c_t\widetilde{\sigma}_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + 2\epsilon_{m,t} + c_t\widetilde{\sigma}_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + c_t\widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}_t) + 2\epsilon_{m,t} + \widetilde{f}_t(\overline{\mathbf{x}}_t) \quad \widetilde{f}_t(\mathbf{x}_t) \\ &\overset{(c)}{} c_t\left(2\widetilde{\sigma}_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + \widetilde{\sigma}_{\text{fb}[t]}(\mathbf{x}_t)\right) + 4\epsilon_{m,t} \\ &\overset{(d)}{} c_t\left(2\sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + \sigma_{\text{fb}[t]}(\mathbf{x}_t)\right) + 3c_t\sqrt{(L+1)\varepsilon\left(1 + \frac{4\hat{K}_0^2 t^2}{\sigma^4}\right)} + 4\epsilon_{m,t} \\ &= c_t\left(2\sigma_{\text{fb}[t]}(\overline{\mathbf{x}}_t) + \sigma_{\text{fb}[t]}(\mathbf{x}_t)\right) + \epsilon'_{m,t}. \end{aligned} \tag{40}$$

(a) follows from Lemmas 12 and 13, (b) follows from the definition of unsaturated inputs (Definition 3), (c) results from the way in which $\mathbf{x}_t$ is selected: $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \widetilde{f}_t(\mathbf{x})$, (d) makes use of Lemma 9.

Next, we can upper-bound the expected instantaneous regret:

$$\begin{aligned}
\mathsf{E}\left[r_t | \mathcal{F}_{t-1}\right] &\leq \mathsf{E}\left[c_t\left(2\sigma_{\mathrm{fb}[t]}(\overline{\mathbf{x}}_t) + \sigma_{\mathrm{fb}[t]}(\mathbf{x}_t)\right) + \epsilon'_{m,t} | \mathcal{F}_{t-1}\right] + 2B'\mathsf{P}\left(\overline{E^{\widetilde{f}_t}(t)} | \mathcal{F}_{t-1}\right) \\
&\overset{(a)}{\leq} \mathsf{E}\left[c_t\left(\frac{2}{p\sqrt{1/t^2}}\sigma_{\mathrm{fb}[t]}(\mathbf{x}_t) + \sigma_{\mathrm{fb}[t]}(\mathbf{x}_t)\right) + \epsilon'_{m,t} | \mathcal{F}_{t-1}\right] + \frac{2B'}{t^2} \\
&= \mathsf{E}\left[c_t\left(1 + \frac{2}{p\sqrt{1/t^2}}\right)\sigma_{\mathrm{fb}[t]}(\mathbf{x}_t) + \epsilon'_{m,t} | \mathcal{F}_{t-1}\right] + \frac{2B'}{t^2} \\
&\overset{(b)}{\leq} c_t\left(1 + \frac{2}{p\sqrt{1/t^2}}\right)\mathsf{E}\left[e^C\sigma_{t-1}(\mathbf{x}_t) | \mathcal{F}_{t-1}\right] + \epsilon'_{m,t} + \frac{2B'}{t^2} \\
&\overset{(c)}{\leq} c_t e^C\left(1 + \frac{10}{p}\right)\mathsf{E}\left[\sigma_{t-1}(\mathbf{x}_t) | \mathcal{F}_{t-1}\right] + \epsilon'_{m,t} + \frac{2B'}{t^2}.
\end{aligned} \tag{41}$$

(a) follows from equation (39), (b) makes use of Lemma 5, and (c) follows since $2/(p\sqrt{1/t^2}) \leq 10/p$. This completes the proof. □

We similarly define the following stochastic process, which will be shown to be a super-martingale in the subsequent Lemma.

Definition 4. *Define $Y_0 = 0$, and for all $t = 1, \ldots, T$,*

$$\overline{r}_t = r_t \mathbb{I}\{E^{\widetilde{f}}(t)\},$$

$$X_t = \overline{r}_t - c_t e^C\left(1 + \frac{10}{p}\right)\sigma_{t-1}(\mathbf{x}_t) - \epsilon'_{m,t} - \frac{2B'}{t^2}$$

$$Y_t = \sum_{s=1}^{t} X_s.$$

Lemma 17. *Conditioned on the event $E^f(t)$, $(Y_t : t = 0, \ldots, T)$ is a super-martingale with respect to the filtration $\mathcal{F}_t$.*

The proof of Lemma 17 above follows closely the proof of Lemma 7 and is hence omitted.

Lemma 18. *Define $C_1 = \frac{2}{\log(1+\sigma^{-2})}$. With probability of $1 - \delta$,*

$$R_T \leq c_T e^C\left(1+\frac{10}{p}\right)\sqrt{C_1 T\gamma_T} + T\epsilon'_{m,T} + \frac{B'\pi^2}{3} + \left(4B' + c_T e^C\left(1+\frac{10}{p}\right)K_0 + \epsilon'_{m,T}\right)\sqrt{2T\log(4/\delta)}. \tag{42}$$

Proof. To begin with, we have that

$$\begin{aligned}
|Y_t - Y_{t-1}| = |X_t| &\leq |\overline{r}_t| + c_t e^C\left(1 + \frac{10}{p}\right)\sigma_{t-1}(\mathbf{x}_t) + \epsilon'_{m,t} + \frac{2B'}{t^2} \\
&\leq 2B' + c_t e^C\left(1 + \frac{10}{p}\right)K_0 + \epsilon'_{m,t} + 2B' \\
&= 4B' + c_t e^C\left(1 + \frac{10}{p}\right)K_0 + \epsilon'_{m,t}.
\end{aligned} \tag{43}$$

Using the Azuma-Hoeffding's inequality with an error probability of $\delta/4$, we have that

$$\begin{aligned}
\sum_{t=1}^{T} \overline{r}_t \quad & \sum_{t=1}^{T} c_t e^C \left(1 + \frac{10}{p}\right) \sigma_{t-1}(\mathbf{x}_t) + \sum_{t=1}^{T} \epsilon'_{m,t} + \sum_{t=1}^{T} \frac{2B'}{t^2} + \\
& \sqrt{2\log(4/\delta) \sum_{t=1}^{T} \left(4B' + c_t e^C \left(1 + \frac{10}{p}\right) K_0 + \epsilon'_{m,t}\right)^2} \\
& \overset{(a)}{\leq} c_T e^C \left(1 + \frac{10}{p}\right) \sum_{t=1}^{T} \sigma_{t-1}(\mathbf{x}_t) + \sum_{t=1}^{T} \epsilon'_{m,t} + \frac{B'\pi^2}{3} + \\
& \left(4B' + c_T e^C \left(1 + \frac{10}{p}\right) K_0 + \epsilon'_{m,T}\right) \sqrt{2T\log(4/\delta)} \\
& \overset{(b)}{\leq} c_T e^C \left(1 + \frac{10}{p}\right) \sqrt{C_1 T \gamma_T} + \sum_{t=1}^{T} \epsilon'_{m,t} + \frac{B'\pi^2}{3} + \\
& \left(4B' + c_T e^C \left(1 + \frac{10}{p}\right) K_0 + \epsilon'_{m,T}\right) \sqrt{2T\log(4/\delta)} \\
& c_T e^C \left(1 + \frac{10}{p}\right) \sqrt{C_1 T \gamma_T} + T\epsilon'_{m,T} + \frac{B'\pi^2}{3} + \\
& \left(4B' + c_T e^C \left(1 + \frac{10}{p}\right) K_0 + \epsilon'_{m,T}\right) \sqrt{2T\log(4/\delta)}.
\end{aligned} \tag{44}$$

(a) follows since c_t is increasing in t, and (b) follows from the proof of Lemma 5.4 in the work of [57]. Next, note that $\overline{r}_t = r_t, \delta t \quad 1$ with probability of $\quad 1 \quad \delta/2$ according to Lemma 12. Also recall that throughout the entire proof in this section, we have conditioned on the event in Proposition 1, which also holds with probability of $\quad 1 \quad \delta/4$. Therefore, also taking into account the error probability of $\delta/4$ from the Azuma-Hoeffding's inequality, the upper bound derived above is an upper bound on the cumulative regret $R_T = \sum_{t=1}^{T} r_t$ with probability of $\quad 1 \quad \delta/2 \quad \delta/4 \quad \delta/4 = 1 \quad \delta$. □

Now let's analyze the asymptotic scaling of the regret upper bound derived above. Firstly, note that $c_T = O(\log^2 T)$. Next, recall that we have in the main text that $(L+1)\varepsilon = C_{\text{ntk}}(L+1)L^{3/2}\log^{1/4}(4LjXj^2/\delta)m^{-1/4}$. This allows us to analyze the scaling of $\epsilon'_{m,T}$.

$$\begin{aligned}
\epsilon'_{m,T} &= 3c_T \sqrt{(L+1)\varepsilon \left(1 + \frac{4\hat{K}_0^2 T^2}{\sigma^4}\right)} + 4\Big(2\hat{K}_0 \frac{T^2(L+1)\varepsilon}{\sigma^4} \left(B' + \sigma\sqrt{2\log(4T/\delta)}\right) + \\
&\qquad \beta_T \sqrt{(L+1)\varepsilon \left(1 + \frac{4\hat{K}_0^2 T^2}{\sigma^4}\right)}\Big) \\
&= \widetilde{O}\left((\log T)^2 \sqrt{(L+1)\varepsilon} T + T^2(L+1)\varepsilon + \log T \sqrt{(L+1)\varepsilon} T\right) \\
&= \widetilde{O}\left(T^2 \sqrt{(L+1)\varepsilon}\right) \\
&= \widetilde{O}\left(T^2 m^{-1/8} (L+1)^{5/4}\right)
\end{aligned} \tag{45}$$

This allows us to analyze the asymptotic scaling of our regret upper bound (ignoring all log factors)

$$\begin{aligned}
R_T &= \widetilde{O}\left(e^C \sqrt{T\gamma_T} + T\epsilon'_{m,T} + (e^C + \epsilon'_{m,T})\sqrt{T}\right) \\
&= \widetilde{O}\left(e^C \sqrt{T}(\sqrt{\gamma_T} + 1) + T\epsilon'_{m,T} + \sqrt{T}\epsilon'_{m,T}\right) \\
&= \widetilde{O}\left(e^C \sqrt{T}(\sqrt{\gamma_T} + 1) + T^3 m^{-1/8} (L+1)^{5/4}\right).
\end{aligned} \tag{46}$$

E Extension to Continuous Input Domains

To extend our theoretical results to cases where the input domain $\mathcal{X}$ is continuous, we can follow the techniques discussed in Section 3.1 of the work of [39]. We assume that $\mathcal{X} \subseteq [0, 1]^d$. To begin with, we need to additionally assume that the objective function f is Lipschitz continuous with a Lipschitz constant $L > 0$. Next, we can construct a finite sub-domain $\widetilde{\mathcal{X}}$ of the continuous domain $\mathcal{X}$, where $\widetilde{\mathcal{X}}$ has equal spacing of $\frac{1}{\sqrt{T}}$ in each dimension. As a result, the finite sub-domain $\widetilde{\mathcal{X}}$ contains $T^{d/2}$ points, i.e., $|\widetilde{\mathcal{X}}| = T^{d/2}$. Then, we can simply run our algorithms (Algo. 1 and Algo. 2) on this finite sub-domain $\widetilde{\mathcal{X}}$.

As a consequence, for **Theorem 1**, we only need to make two changes to our theoretical results. Firstly, we need to modify β_t to be $\beta_t = 2\log(\pi^2 t^2 |\widetilde{\mathcal{X}}|/(3\delta)) = 2\log(\pi^2 t^2 T^{d/2}/(3\delta))$, which will only introduce an additional dependence on $\mathcal{O}(d \log T)$ into c_T (Appendix C) and hence *an additional multiplicative factor of* $\mathcal{O}(d \log T) = \widetilde{\mathcal{O}}(d)$ into the regret upper bound in Theorem 1. Secondly, due to the Lipschitz continuity of f and the fact that every input $\mathbf{x} \in \mathcal{X}$ has a neighbor in the finite sub-domain $\widetilde{\mathcal{X}}$ whose distance to it is less than $\frac{d}{\sqrt{T}}$, we have that $f(\mathbf{x}^*) \leq \max_{\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}} f(\widetilde{\mathbf{x}}) + \mathcal{O}(\frac{Ld}{\sqrt{T}})$. As a result, this will introduce *an additional additive term of* $\mathcal{O}(T \frac{Ld}{\sqrt{T}}) = \mathcal{O}(Ld\sqrt{T})$ to the final upper bound on the cumulative regret.

For **Proposition 1**, an additional multiplicative factor of $d \log T$ will be introduced into the condition on m. For **Theorem 2**, to begin with, same as the analysis of Theorem 1 in the paragraph above, *an additional additive factor of* $\mathcal{O}(Ld\sqrt{T})$ will be introduced, and *an additional multiplicative factor of* $\mathcal{O}(d \log T) = \widetilde{\mathcal{O}}(d)$ will be introduced into the first term in Theorem 2. Moreover, as a result of the additional factor of $d \log T$ in the condition on m (Proposition 1), an additional multiplicative factor of $d \log T$ will also be introduced into the approximation quality of $(L + 1)\varepsilon$ (Sec. 4.2). As a result, in the proof of Theorem 2 (Appendix D), *an additional multiplicative factor of* $\sqrt{d}$ will be introduced into the term $\epsilon'_{m,T}$ (see (45)) and hence into the second term in the upper bound in Theorem 2.

Of note, the modified results discussed above do not affect the scaling of our theoretical results (Theorem 1 and Theorem 2) in T (we ignore all dependencies on $\log T$), because the only additional term depending on T for both theorems is an additive term of $\widetilde{\mathcal{O}}(\sqrt{T})$.

F More Experimental Details

In all experiments, for simplicity, we set $\beta_t = 1, \forall t \geq 1$, which is consistent with many previous papers on BO which have found the theoretical values of β_t to be overly conservative [57]. For fair comparisons, in every experiment, all methods under comparison use the same set of initial inputs which are selected by random search. We use the ERF activation function in the synthetic experiment (Sec. 5.1) because the synthetic function we have adopted is very smooth. In all real-world experiments (Secs. 5.2, 5.3 and 5.4), we use the ReLU activation function since it has been found to be very effective in modeling complicated real-world functions.

For all methods under comparison, when maximizing the acquisition function to choose an input query, if the domain is discrete, we simply evaluate the acquisition function value at every input in the domain and then choose the input that maximizes it. When the domain is continuous, we firstly use random search to randomly sample $10,000$ inputs in the domain to evaluate their acquisition function values, and then use L-BFGS-B with 100 random restarts to refine the search. When the domain is mixed (i.e., consisting of both continuous and discrete inputs), we treat it as a continuous domain and after finding the input the maximizes the acquisition function, we round the discrete inputs to the nearest integer. For Neural UCB [69], we treat the UCB value calculated for each arm (input) as the acquisition function; for Neural TS, we treat the reward sampled for each arm (input) as the acquisition function [66].

We implement the training of the surrogate model $f_t^i(\mathbf{x}; \theta)$ for both Algos. 1 and 2 based on the implementations from the work of [24], and we adopt all their default parameter settings (refer to the implementations of [24] for the specific parameter settings, available at `https://github.com/bobby-he/bayesian-ntk`) and only vary the architecture of the NN surrogate model (e.g., the depth and width of the NN, we replace the NN with a CNN for our experiments in Sec. 5.4) as we

have mentioned in the main text. For both Neural UCB and Neural TS, we adopt the implementations from the work of [66], use all their default parameter settings, and only modify the architecture of their NN surrogate model for a fair comparison with our methods. As we have mentioned in the main text, to apply Neural UCB and Neural TS for problems with continuous domains, we adapt their implementations such that we optimize their acquisition functions in the same way as our methods (i.e., through a combination of random search and L-BFGS-B as discussed above). Our experiments are performed using a computing cluster where each machine has an NVIDIA A100 GPU and 96 CPUs.

F.1 Synthetic Experiments

In the synthetic experiment, the objective function f is sampled from a GP with an SE kernel using a lengthscale of 0.1. The domain of f is a uniform grid of size $1,000$ in the interval of $[0, 1]$.

F.2 Real-world Experiments on Automated ML

In this section, we give more details on the three hyperparameter tuning experiments in Sec. 5.2.

Hyperparamter Tuning of Random Forest. Here we tune 6 categorical hyperparameters of random forest:

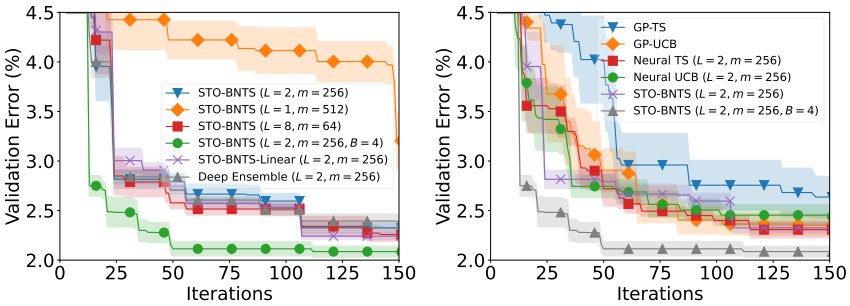
- the maximum depth of any individual tree (integer within $[1, 10]$),
- the minimum number of samples required to split an internal node (integer within $[2, 10]$),
- the minimum number of samples required to be at a leaf node (integer within $[1, 10]$),
- the maximum number of features to consider when looking for the best split (integer within $[1, 8]$),
- the criterion to measure the quality of a split (binary, "entropy" or "gini"),
- whether bootstrap samples are used when building trees (binary, True or False).

We use the publicly available diabetes prediction dataset which can be accessed from `https://www.kaggle.com/uciml/pima-indians-diabetes-database` and has the CC0 License. This dataset does not contains personally identifiable information or offensive content. It consists of 768 data instances, each containing 8 input features. We use 70% of the dataset as the training set and the remaining 30% as the validation set. We use random search to choose 5 initial inputs as the set of initialization, which is shared among all methods under comparison.

Hyperparameter Tuning of XGBoost. The MNIST dataset is publicly available and associated with the GNU General Public License, and can be obtained from the Keras Package[5]. It does not contain personally identifiable information or offensive content. In this experiment, we use the MNIST dataset to tune 9 hyperparameters of XGBoost [6]:

- gamma which represents the minimum loss reduction required to make a further partition on a leaf node of the tree (continuous, $[0, 10]$),
- the learning rate (continuous, $[10^{-6}, 1]$),
- the maximum depth of any individual tree (integer, $[1, 15]$),
- which booster to use (binary, "dart" or "gbtree"),
- the grow policy which controls the way new nodes are added to the tree (binary, "depthwise" or "lossguide"),
- the objective (bianry, "multi:softprob" or "multi:softmax"),
- the tree construction method (binary, "exact" or "hist"),
- alpha which is the L1 regularization term on the weights (continuous, $[0, 10]$), and
- lambda which is the L2 regularization term on the weights (continuous, $[0, 10]$).

[5] `https://keras.io/`



(a) (b)

Figure 4: Validation errors for hyperparameter tuning of CNN. $B = 1$ unless specified otherwise.

Hyperparameter Tuning of Convolutional Neural Networks. Here we use the MNIST dataset to tune 9 hyperparameters of convolutional neural networks (CNN). The CNN consists of one convolutional layer, followed by a max pooling layer and subsequently a fully connected layer. The 9 hyperparameters are:

- the learning rate (continuous, $[10^{-4}, 0.1]$),
- the weight decay (continuous, $[10^{-6}, 10^{-2}]$),
- the batch size (integer, $[64, 512]$),
- the max pooling size (integer, $[3, 5]$),
- the number of neurons in the convolutional layer (integer, $4, 16$),
- the size of the convolutional kernel (integer, $[3, 5]$),
- the number of neurons in the fully connected layer (integer, $[4, 16]$),
- which activation function to use (binary, ReLU or Tanh),
- which optimization method to use (binary, ADAM or RMSprop).

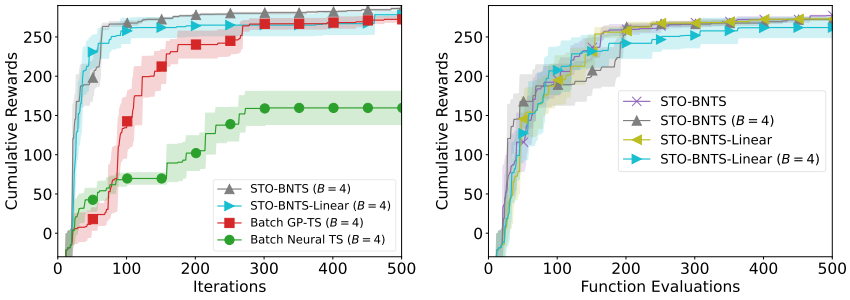
F.3 Real-world Experiments on Reinforcement Learning

The lunar lander task involves tuning 12 parameters of a heuristic controller which is used to control the LunarLander-v2 environment from OpenAI Gym [4]. The heuristic controller is provided by OpenAI Gym and can be found at `https://github.com/openai/gym/blob/8a96440084a6b9be66b2216b984a1c170e4a061c/gym/envs/box2d/lunar_lander.py#L447`. OpenAI Gym[6] is open-sourced and under the MIT License. The (14-dimensional) robots pushing and (20-dimensional) rover trajectory planning tasks were firstly introduced by the work of [62] where the detailed experimental settings can be found. Both tasks are publicly available at `https://github.com/zi-w/Ensemble-Bayesian-Optimization` and are under the MIT license. Due to the large number of iterations (500) of these three experiments (which is necessary as a result of the high dimensionality of the input spaces), standard GP-UCB and GP-TS become too computationally costly to run. Therefore, we applied random Fourier features approximations [11, 12] to the GP using a large number $1,000$ of random features, with which GP-UCB and GP-TS perform well and are still computationally feasible to run.

Using the Lunar-Lander experiment, we have also compared our STO-BNTS and STO-BNTS-Linear with batch versions of GP-TS and Neural TS. The results are shown in Fig. 5 (a), in which all methods use the same batch size of $B = 4$. The figure shows that our STO-BNTS and STO-BNTS-Linear are still able to significantly outperform the other baseline methods when the same batch size is used.

We also use the Lunar-Lander experiment to show an alternative visualization of the performances of our algorithms with batch evaluations in Fig. 5 (b). Specifically, the horizontal axis in Fig. 5 (b) is the number of function evaluations, in contrast to iterations in Fig. 3a. Same as Fig. 3a, this figure also shows the benefit of batch evaluations, because compared with the sequential algorithms ($B = 1$,

[6] `https://github.com/openai/gym`



(a) (b)

Figure 5: (a) Comparison of different algorithms with the same batch size of $B = 4$, including batch versions of our STO-BNTS and STO-BNTS-Linear, as well as batch GP-TS and batch Neural TS. (b) An alternative visualization of the performance of our algorithms with batch evaluations, using the 12-D Lunar-Lander experiment (Fig. 3a). The horizontal axis here is the number of function evaluations, in contrast to iterations in Fig. 3a.

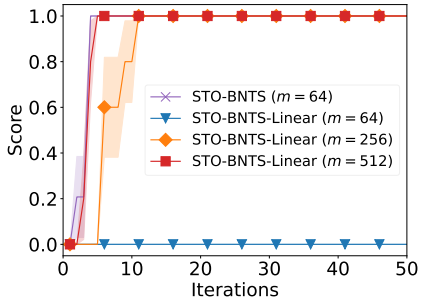


Figure 6: Results of STO-BNTS-Linear in the experiments on optimization over images (Sec. 5.4).

purple and yellow curves), our algorithms with a batch size of $B = 4$ only suffer slight degradations of the per-function evaluation performances.

F.4 Optimization over Images

In this experiment, to construct the score function (Fig. 3d), we firstly use the training set of the MNIST dataset (consisting of $60,000$ images) to train a CNN, and then use the trained CNN to predict the class probabilities for the 10 different classes using the testing set of $10,000$ images. Next, we use the predicted probability of class 0 for the $10,000$ testing images as the score function. As a result of our construction of the score function, similar images in general have similar score values since they share similar representations from the CNN, and images of 0 overall have much larger scores than images from the other classes. This can simulate the real-world scenario of image recommender system, in which the user may prefer a certain type of images and hence give higher ratings to them.

For all three CNN-based methods in Fig. 3d, we use a CNN with one convolutional layer (with convolutional kernels size of 3), followed by a max pooling layer (with a pooling size of 3), and then followed by a fully connected layer. We have used the ReLU activation function. The width of both the convolutional and fully connected layers are $m = 64$. Fig. 6 plots the results for STO-BNTS-Linear in this experiment, which shows that its performance can be dramatically improved if we increase the width m of the NN surrogate model (Sec. 5.4).