Hyper-parameter Tuning under a Budget Constraint

Zhiyun Lu[1*], **Liyu Chen**[1], **Chao-Kai Chiang**[2†] and **Fei Sha**[3‡]
[1]University of Southern California
[2]Appier Inc.
[3] Google AI
{zhiyunlu, liyuc}@usc.edu, chaokai@gmail.com, fsha@google.com

Abstract

Hyper-parameter tuning is of crucial importance for real-world machine learning applications. While existing works mainly focus on speeding up the tuning process, we propose to study the problem of hyper-parameter tuning under a budget constraint, which is a more realistic scenario in developing large-scale systems. We formulate the task into a sequential decision making problem and propose a solution, which uses a Bayesian belief model to predict future performances, and an action-value function to plan and select the next configuration to run. With long term prediction and planning capability, our method is able to early stop unpromising configurations, and adapt the tuning behaviors to different constraints. Experiment results show that our method outperforms existing algorithms, including the-state-of-the-art one, on real-world tuning tasks across a range of different budgets.

1 Introduction

Hyper-parameter tuning is of crucial importance to designing and deploying machine learning systems. Broadly, hyper-parameters include the architecture of the learning models, regularization parameters, optimization methods, and other "knobs" to be tuned. It is challenging to explore the vast space of hyper-parameters efficiently to identify the optimal configuration. Quite a few approaches have been proposed: random search, Bayesian Optimization (*BO*) [Snoek *et al.*, 2012; Shahriari *et al.*, 2016], bandits-based *Hyperband* [Jamieson and Talwalkar, 2016; Li *et al.*, 2016], and meta-learning [Chen *et al.*, 2016; Bello *et al.*, 2017; Franceschi *et al.*, 2018].

Most of the prior works have focused on the aspect of reducing as much as possible the computation cost to obtain the optimal configuration. Finding the optimal, however, is not always practical: in deep learning, there is a vast search space of hyper-parameters, and it takes a long time to train each model. In this work, we look at a different but important perspective to hyper-parameter tuning – under a fixed time/computation cost, how we can improve the performance as much as possible. Concretely, we study the problem of hyper-parameter tuning under a hard budget constraint. The budget offers the practitioners a tradeoff: affordable resource balanced with good performance. It is a more realistic scenario in developing large-scale learning systems, and is especially applicable, for example when the practitioner searches for a best model under the pressure of a deadline.

The budget constraint certainly complicates the hyper-parameter tuning strategy. While the strategy without the constraints is to explore and exploit in the hyper-parameter space, a *budget-aware* strategy needs to decide how much to explore and exploit with respect to the resource/time. As most learning algorithms are iterative in nature, a human operator would monitor the training progress of different configurations, and adjust the tuning strategy based on their potential future performances and how much resource remains – we all have had the experience of "killing" a running job as "this training is going nowhere"! *How can we automate this process?*

We formalize this inquiry into a sequential decision making problem, and propose a budgeted hyper-parameter tuning algorithm (*BHPT*) to automatically achieve good resource utilization. The *BHPT* algorithm uses a belief model to predict future performances of configurations, and an action-value function to select the configurations. It automatically balances exploration with exploitation under the budget, and adapts to different constraints. We empirically demonstrate the performance of the proposed algorithm on both synthetic and real-world datasets. *BHPT* outperforms the state-of-the-art tuning algorithms across a wide range of budgets. Besides, it exhibits budget adaptive tuning behaviors.

The rest of the paper is organized as follows. We first discuss related work in Sec. 2. We formally define the problem and introduce the sequential formulation in Sec. 3. We describe the proposed algorithm with analysis in Sec. 4. Experiments and the conclusion can be found in Sec. 5 and 6. respectively.

2 Related Work

Automated design of machine learning system is an important research topic Snoek *et al.* [2012]. Traditionally hyper-parameter optimization (HO) is formulated as a black-box

*Contact Author
†Work done while at USC.
‡On leave from USC (feisha@usc.edu).

optimization and solved by Bayesian optimization (*BO*). It uses a probabilistic model together with an acquisition function to adaptively select and identify the optimal configuration [Shahriari *et al.*, 2016]. However, fully training a single configuration can be expensive. Recent advances focus to exploit cheaper surrogate tasks, like training on subsets of the data (*Fabolas* [Klein *et al.*, 2017]), or partial training curves (*FreezeThaw* [Swersky *et al.*, 2014]), to speed up the tuning.

Recently *Hyperband* [Li *et al.*, 2016] formulates the HO as a best-arm identification problem. It adaptively evaluates a configuration's intermediate results and quickly eliminates poor arms. Lately, Falkner *et al.* [2018] combines the benefits of *BO* and *Hyperband* to achieve fast convergence to the optimal. Other HO work includes gradient-based methods [Maclaurin *et al.*, 2015; Franceschi *et al.*, 2017], meta-learning [Jaderberg *et al.*, 2017; Franceschi *et al.*, 2018], the spectral approach [Hazan *et al.*, 2017], and adaptive data allocation [Sabharwal *et al.*, 2016].

However, none of them has an explicit notion of *hard* budget constraint for the tuning process. Neither do they consider to adapt the tuning strategy across different budgets. In this work, we propose to take the resource constraint as an input to the tuning algorithm.

Table. 1 summarizes the comparisons of popular tuning algorithms to our work from three perspectives: whether it uses a (probabilistic) model to adaptively predict and identify good configurations; whether it supports early stop and resume during training; and whether it adapts to different budgets.

algorithm	early stop and resume	adaptive (future) prediction	budget aware
random search	×	×	×
BO (GP-EI/SMAC)	×	✓	×
Fabolas	✓	✓	×
FreezeThaw	✓	✓	×
Hyperband	✓	×	×
BHPT (ours)	✓	✓	✓

Table 1: Comparison of tuning algorithms

3 Problem Statement

We start by introducing the notation, and formally define the budgeted hyper-parameter tuning problem. Then we formulate the task into a sequential decision making problem.

3.1 Preliminaries

Configuration (arm). Configuration denotes the hyper-parameters, e.g. the architecture, the optimization method. We use $[K] = \{1, ..., K\}$ to index the set of configurations. The term configuration and arm are used interchangeably. (See the supplementary material[1] Sec. A for a full notation table.)

Model. Model refers to the (intermediate) training outcome, e.g. the weights of neural nets, of a particular configuration. We evaluate the model on a heldout set periodically, for example every epoch. We consider loss or error rate as the evaluation metric. We keep track of $\nu_b^k \in [0, 1)$, the *minimum* loss of configuration k among $1, \dots, b$ epochs. Note that ν_b^k is a non-increasing function in b. We always use superscript to denote the configuration, and the subscript for the budget/epoch.

Budget. The budget defines a computation constraint imposed on the tuning procedure. In this paper, we consider training time, or epoch, as the budget unit. Epoch is an abstract notion of computation resource in most iterative learning algorithms. Given a total budget $B \in \mathbb{N}_+$, a strategy $\boldsymbol{b} = (b^1, \dots, b^K) \in \mathbb{N}^K$ allocates the budget among K configurations, i.e. it runs configuration k for b^k epochs. $\boldsymbol{b}$ should satisfy that the total epochs from K configurations add up to B: $\boldsymbol{b}^\top \mathbf{1}_K = B$. We use epoch and budget interchangeably when there is no confusion.

Constrained Optimization. The goal of the budgeted hyper-parameter tuning task is to obtain a well-optimized model under the constraint. Under the allocation strategy $\boldsymbol{b}$, arm k returns a model with loss $\nu_{b^k}^k$. We search for the strategy which optimizes the loss of the *best* model out of K configurations, $\ell_B = \min\{\nu_{b^1}^1, \dots, \nu_{b^K}^K\}$. Concretely, the constrained optimization problem is

$$\min_{\boldsymbol{b}} \ell_B, \quad \text{s.t. } \boldsymbol{b}^\top \mathbf{1}_K = B. \tag{1}$$

3.2 Optimal Solution with Perfect Information

Despite a combinatorial optimization, if we *know* the training curves of all configurations, Solving Eq. 1 is simply to put all budgets to the configuration c that achieves the smallest loss.

$$\ell_B^* = \min_k \nu_B^k, \quad c = \operatorname*{argmin}_k \nu_B^k. \tag{2}$$

Namely $\nu_B^c = \ell_B^*$. Note that this is infeasible because it would require KB epochs to obtain $\{\nu_b^k\}_{k\in[K], b\in[B]}$.

However, since $\nu_1^k, ..., \nu_B^k$ are dependent variables, it is feasible to predict the future losses using historical information, which helps to select the next configuration. This motivates us to formulate the budgeted tuning as a finite-horizon sequential decision making problem.

3.3 Sequential Decision Making

We define a sequential tuning procedure, where at each step a model of one configuration is selected to train for one epoch. At the n-th step, the tuning algorithm selects the action/configuration $a_n \in [K]$, and the corresponding loss $z_n \in [0, 1)$ is returned. We want to find a policy π, which selects the action at each step based on the history and the remaining budget $r = B - n$: $a_n = \pi\big(r, (a_1, z_1, \dots, a_{n-1}, z_{n-1})\big)$. This process is repeated for B steps and the final tuning output is

$$\ell_B^\pi = \min_{1\le n\le B} z_n. \tag{3}$$

Solving Eq. 3 is equivalent to solving Eq. 1 in a sequential manner. However, the advantages are that the tuning algorithm can stop and resume the training of a configuration at any time. Besides, it can leverage the information of partial training curves and remaining budget r to make more economical decisions in terms of budget.

The framework can be extended to other settings, e.g. multiple configurations in parallel, resource of heterogenous

[1] Supplementary material link: https://bit.ly/2ICQF0K.

types (like cpu, gpu) etc. While some extensions is straightforward, some requires more careful design and is left as future work.

Regret. The goal of the sequential problem is to optimize the policy π, such that the regret is minimized,

$$\min_{\pi} \ell_B^{\pi} - \ell_B^* = \min_{\pi} \Big(\min_{1 \le n \le B} z_n - \nu_B^c \Big). \tag{4}$$

where c is the best arm in Eq. 2. Note ℓ_B^* assumes knowledge of all curves, hence infeasible to achieve in practice.

Challenges. The sequential decision making problem introduced in our paper is an instance of extreme bandits [Nishihara *et al.*, 2016]. This leads to the well-known exploitation-exploration tradeoff: on one hand, it tempts to pick arms that have achieved small losses (exploitation). On the other hand, there is also an incentive to pick other arms to see whether they can admit even smaller losses (exploration). However, Eq. 4 is different than the typical bandits, because ℓ_B^{π} is the *minimum* over all losses, instead of the sum. This leads to two differences: firstly the optimal policy/configuration depends on the horizon B. Secondly at every step, the optimal policy for the remaining horizon depends on the history. The key of our method to address this challenge is to *re-plan* at every step using historical information.

4 Approach

In this section, we describe the *BHPT* algorithm. The main idea is that we use a Bayesian belief model to *predict* the future performance of configurations, and follow a policy to *plan* and select the next configuration.

At step n, *BHPT* does the following two steps:

$$a_n \leftarrow \pi(r, \mathrm{S}_{n-1}) \qquad \textit{plan},$$
$$\mathrm{S}_n \leftarrow \text{ Belief update}(\mathrm{S}_{n-1}, a_n, z_n) \qquad \textit{predict}.$$

S_n is the Bayesian belief model, which predicts future performances and gets updated at every step with new observations. $\pi(r, \mathrm{S}_{n-1})$ is the policy to pick the next configuration, which takes both the remaining budget r, and the future predictions of the belief S_{n-1} as input. The core of policy π is an action-value function $\mathrm{Q}_r[a|\mathrm{S}_{n-1}]$, which outputs a score of expected future performance of a configuration.

BHPT algorithm can adaptively predict and identify good configurations through the Bayesian belief model S_n, and adapt the tuning behaviors to different budgets due to the dependency of $\pi(r, \mathrm{S}_{n-1})$ on r, as summarized in Table 1.

We describe the action-value function Q in Sec. 4.1, and the Bayesian belief model in Sec. 4.2. We discuss how Q balances the exploration-exploitation tradeoff in Sec. 4.3, and present the *BHPT* algorithm in Sec. 4.4.

4.1 Action-value Function Q

Consider we are at the n-th step, with remaining budget $r = B - n$. Our goal is to find the policy which minimizes the future losses : $\ell_r = \min_{n \le s \le B} z_s$. We use a belief state/model S_{n-1} to estimate the unknown training curves, which is derived from the past history $(a_1, z_1, \ldots, a_{n-1}, z_{n-1})$ and allows us to simulate, and predict future outcomes.

We compute an action-value function $\mathrm{Q}_r[a|\mathrm{S}_{n-1}]$ for each arm, and select the next action that minimizes Q,

$$a_n = \pi\big(r, (a_1, z_1, \ldots, a_{n-1}, z_{n-1})\big) = \operatorname*{argmin}_{a \in [K]} \mathrm{Q}_r[a|\mathrm{S}_{n-1}]. \tag{5}$$

The action-value function Q[2] is defined as follow:

$$\mathrm{Q}_r[a] = \mathbb{E}\left[\min\{\nu_r^a, \min_{k \ne a} \mu_r^k\} \right] \tag{6}$$

where ν_r^k is the minimum loss of configuration k in the *future* r epochs *starting from the current epoch* (recall Sec. 3.2). and $\mu_r^k = \mathbb{E}\left[\nu_r^k | \mathrm{S}_{n-1}\right]$ is the expected loss.

Intuitively, Q measures the potential of arm a compared to the expected performances of other arms. Note that this is an optimistic estimation of arm a's performance since it does not account for the risk when $\nu_r^a > \min_{k \ne a} \mu_r^k$. The design of Q leverages the optimal planning structure – it directly predicts the final loss after r epochs, which is an estimate of the optimal solution with perfect information (Sec. 3.2). In what follows, we show that minimizing Q is equivalent to maximizing a notion of surprise, which measures the potential gain of pulling the arm for the remaining budgets.

Specifically, what is a good action/arm which improves the estimate of future performances? There are two scenarios where the outcomes contradict to our prior belief: (a) an arm previously considered sub-optimal turns out to be the best arm, and (b) the arm previously considered best is actually inferior to others. Denote $\widehat{c} = \operatorname*{argmin}_k \mu_r^k$ as the predicted top configuration. $\widehat{c}$'s predicted loss is $\mu_r^{\text{1st}} = \min_k \mu_r^k$, and similarly the loss of the runner-up μ_r^{2nd}. Now consider for a sub-optimal arm $a \ne \widehat{c}$, Eq. 6 becomes

$$\mathrm{Q}_r[a] = \mathbb{E}\big[\min\{\nu_r^a, \mu_r^{\text{1st}}\}\big] = \mu_r^{\text{1st}} - \mathbb{E}\Big[\big(\mu_r^{\text{1st}} - \nu_r^a\big)^+\Big]. \tag{7}$$

Note that case (a) happens when: $\nu_r^a < \mu_r^{\text{1st}}$, and we expect to gain $\mu_r^{\text{1st}} - \nu_r^a$ by running a instead of $\widehat{c}$. The second term on the r.h.s., $\mathbb{E}\left[\left(\mu_r^{\text{1st}} - \nu_r^a\right)^+\right]$, computes the area when ν_r^a falls smaller than μ_r^{1st}, which quantifies the expected surprise of a. This quantity is called the value of perfect information in decision theory [Howard, 1966], a numerical value that measures the reduction of uncertainty. Minimizing $\mathrm{Q}_r[a]$ favors the arm with large surprise.

Similarly in case (b) when we consider the predicted top arm $a = \widehat{c}$, there is a surprise when $\nu_r^{\widehat{c}} > \mu_r^{\text{2nd}}$: the expected best arm falls behind with other candidates. Q in this case is

$$\mathrm{Q}_r[\widehat{c}] = \mathbb{E}\Big[\min\{\nu_r^{\widehat{c}}, \mu_r^{\text{2nd}}\}\Big] = \mu_r^{\text{1st}} - \mathbb{E}\left[\left(\nu_r^{\widehat{c}} - \mu_r^{\text{2nd}}\right)^+\right], \tag{8}$$

where the second term computes the area when surprisingly $\nu_r^{\widehat{c}}$ is larger than μ_r^{2nd}. Pulling $\widehat{c}$ in this case is favorable since it has large gain of surprise. Refer to Fig.1 in the supplementary material Sec. B for visualization of Q.

We will come back to the discussion of the properties and behaviors of Q in Sec. 4.3, after introducing the details of the belief model in the next section.

[2] We drop the S_{n-1} in Q to simplify the notation when it's clear.

4.2 Belief Model

In this section, we briefly describe the Bayesian belief model S_n, and explain how Eq. 6 is computed with the posterior distribution. The belief model captures our current knowledge of the training curves to predict future ν_r^k, and gets updated as new observations become available. In this paper, we adopt the Freeze-Thaw Gaussian Process [Swersky *et al.*, 2014].

We use (k, t) to index the hyper-parameters and epochs respectively, and the loss of k from the t-th epoch is $y^k(t)$. The joint distribution of losses from all configurations and epochs, $\boldsymbol{y} = [y^1(1), y^1(2), \ldots, y^1(n_1), \ldots, y^K(n_K)]^\top$(arm k has n_k epochs/losses) is Gaussian, with a specific Freeze-Thaw covariance kernels. We use the joint distribution to predict future performances of $y^k(t),\ \forall k, t$, and update the posteriors with new observations, by applying Bayes' rule. Details of the belief model can be found in the supplementary material Sec. E.

Computing Q**.** Note that the future best loss is a random variable, $\nu_r^k = \min_{1\le t\le r} y^k(t_0+t)$. However, ν_r^k's distribution is non-trivial to compute, as it is the minimum of r correlated Gaussians. To simplify the computation[3], we approximate: $\nu_r^k \approx y^k(t_0+\tau^k)$ where we fix the time index τ^k deterministically, to be the one which achieves the minimum loss in expectation, $\tau^k = \operatorname{argmin}_{1\le t\le r} \mathbb{E}\left[y^k(t_0+t)\right]$. [4] The advantage is that $\mathrm{Q}_r[a]$ can be computed efficiently in closed-form:

$$\mathrm{Q}_r[a] = \mathbb{E}_\nu[\min\{\nu_r^a, \mu\}] = \mu - \sigma[\nu_r^a]\big(s\Phi(s) + \phi(s)\big) \qquad (9)$$

where $s = \dfrac{\mu - \mathbb{E}\left[\nu_r^a\right]}{\sigma\left[\nu_r^a\right]}$ is the normalized distance of ν_r^a to μ, and μ is a constant, either μ_r^{1st} or μ_r^{2nd} depending on whichever a we are looking at. $\sigma\left[\cdot\right]$ is the standard deviation, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the cdf and pdf of standard Gaussian respectively.

4.3 Behavior of Q

In this section, we analyze the behaviors of the action-value function. With Gaussian distributed ν_r^ks, there are nice properties of the proposed Q.

Q balances the exploration with exploitation: both arms with smaller mean (exploitation), and larger variance (exploration) are preferred—in either case the surprise area is large (2nd term on the r.h.s. of Eq. 7 and 8). We provide asymptotic analysis of the behaviors as the budget goes to infinity. Proofs can be found in the supplementary material Sec. C.

Proposition 1. Q *is asymptotically consistent, that is,* $\lim_{B\to\infty} \ell_B^\pi - \ell_B^* \to 0$. *The regret (Eq. 4) goes to 0, and the best configuration will be discovered* $B \to \infty$.

[3] We can always use a Monte Carlo approximation to compute this quantity. Asymptotically our simplification does not affect the behavior of the Q, see Sect. 4.3.

[4] The intuition is that for different random draws of the curve, ν_r^k can be any one of $y^k(t_0+t)$ for $1 \le t \le r$. In most cases, $\Pr\left[\nu_r^k = y^k(t_0+\tau^k)\right] \ge \Pr\left[\nu_r^k = y^k(t_0+t)\right], \forall t$. Thus we use $y^k(t_0+\tau^k)$ as ν_r^k.

Proposition 2. *As* $B \to \infty$*, the limiting ratio of the number of pulls between the top and the second best arm approaches 1 assuming the same observation noise parameter.*

4.4 Budgeted Hyper-parameter Tuning Algorithm

In this section, we discuss the practical use of Q, and present the *BHPT* algorithm. Imagine the behavior of Q when applied to the hyper-parameter tuning: all configurations get selected often at the beginning, due to the high uncertainty. Gradually as our prediction gets more accurate, the actions focus on a few promising arms. Finally, Q mostly allocates budget among the top two arms, according to Prop. 2. This in practice can be a waste of resource, because we aim to finalize on one model. We propose the following heuristics.

ε-Greedy policy. In practice, we can cut down the exploration in Q, and exploit more in the top arm. Recall $\widehat{c}$ is the predicted best arm. We design the following ε-greedy policy:

$$a_n = \pi^\varepsilon(r, \mathrm{S}_{n-1}) = \begin{cases} \widehat{c}, & \text{w.p. } \varepsilon, \\ \operatorname{argmin}_{a\neq\widehat{c}} \mathrm{Q}_r[a|\mathrm{S}_{n-1}], & \text{otherwise.} \end{cases} \qquad (10)$$

With probability ε we select $\widehat{c}$, and follow Q for the rest of the time. We set $\varepsilon = 0.5$ in the experiment[5].

Budget exhaustion. When the budget is running out, it is desirable to focus mainly on the top arm $\widehat{c}$. Define $\tau^\star = \operatorname{argmin}_{1\le t\le r} \mathbb{E}\left[y^{\widehat{c}}(t_0+t)\right]$, the number of epoch $\widehat{c}$ short of achieving its minimum. We use the condition $\tau^\star < r$, where r is the remaining budget, to keep track of the budget exhaustion. We pick $\widehat{c}$ when it is false.

The final algorithm is presented in Alg. 1, which we will refer to as *BHPT*, and *BHPT*-ε in the experiments.

Algorithm 1: Budgeted Hyperparameter Tuning (*BHPT*)

Input: Budget B, and configurations $[K]$.
for $n = 1, 2, \ldots B$ **do**
  if $\tau^\star < r$ **then**
    $a_n = \pi(r, \mathrm{S}_{n-1})$, or $a_n = \pi^\varepsilon(r, \mathrm{S}_{n-1})$; // Eq. 5 and 6, or 10
  else
    $a_n = \widehat{c}$. // budget exhaustion
  Run a_n and obtain loss $z_n = y^{a_n}(t)$ (for some t).
  $\mathrm{S}_n \leftarrow$ Belief update$(\mathrm{S}_{n-1}, a_n, z_n)$. // Sec. 4.2
  Update $\tau^\star$ and r.
Output $\min_{1\le n\le B} z_n$.

5 Experiment

In this section, we first compare and validate the conceptual advantage of the *BHPT* algorithm over other methods, by analyzing the exploration exploitation tradeoff under different budgets on synthetic data. Then we demonstrate the performance of the *BHPT* algorithm on real-world hyper-parameter tuning tasks. Particularly, we include a task to tune network

[5] Despite the fact that we do not tune ε, π^ε performs well as demonstrated in the experiments.

dataset	K # arms	# hyper-params.	evaluation
synthetic	84	NA	Eq. 11
ResNet on CIFAR-10	96	5	error rate
FCNet on MNIST	50	10	error rate
VAE on MNIST	49	4	ELBO
ResNet/AlexNet on CIFAR-10	49	6	error rate

Table 2: Data and Evaluation Metric

architectures because selecting the optimal architecture under a budget constraint is of great practical importance.

We start by describing the experimental setup, i.e. the data, evaluation metric and baseline methods, in Sec. 5.1, and then provide results and analysis in Sec. 5.2 and 5.3.

5.1 Experiment Description

Data. For the data preparation, we generate and save the learning curves of all configurations, to avoid repeated training when tuning under different budget constraints. For synthetic data, we generate 100 sets of training curves drawn from a Freeze-Thaw GP. For real-world data, we create 4 tuning tasks as summarized in Table 2. For more details, please refer to the supplementary material Sec. F.

Evaluation Metric. For evaluation metric on synthetic data, we define the *normalized regret*

$$\mathcal{R}_B = \frac{\ell_B^\pi - \ell_B^*}{\ell_0 - \ell_B^*}, \tag{11}$$

where ℓ_0 is the maximal initial loss of all arms; ℓ_B^π is the tuning output; ℓ_B^* is the optimal solution with perfect information of training curves. We normalize the regret over the range $\ell_0 - \ell_B^*$, which can be substantially different from different sets. For real-world task, see Table 2 for the evaluation metrics.

Baseline Methods. We compare to *Hyperband*, *BO* and its variants (*Fabolas* and *FreezeThaw*), and *Rollout* [Lam *et al.*, 2016], which is an approximate dynamic programming approach to solve for the Bayes optimal solution. Please refer to the supplementary material Sec. D for the full descriptions of methods and the implementation details. Table 1 in Sec. 2 summarizes the key differences between methods.

5.2 Results on Synthetic Data

On synthetic data, since we use the correct model, we expect that the belief model learns to accurately predict the future as the budget increases. We examine the behaviors of the proposed algorithms under different budgets. All results are averaged over 100 synthetic sets.

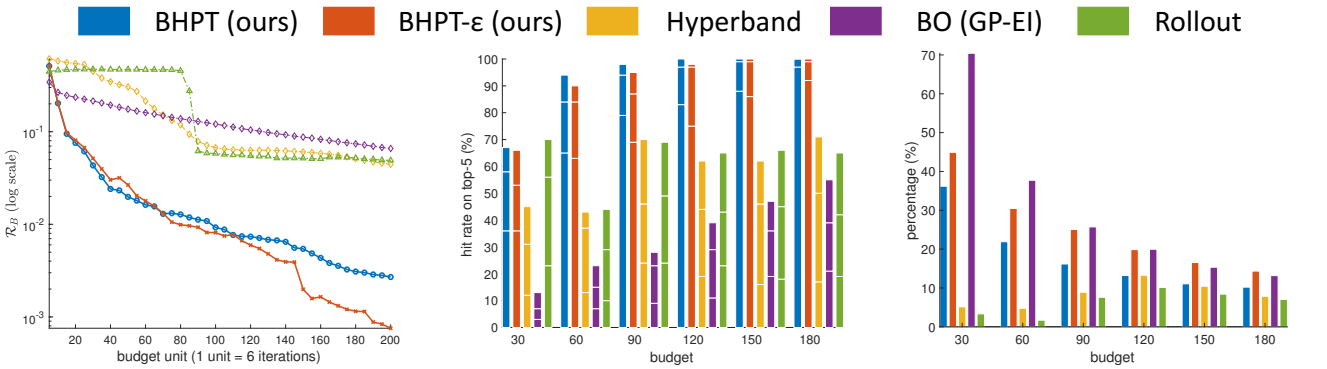
First, we plot the normalized regrets $\mathcal{R}_B$ (Eq. 11) over budgets in Fig. 1(a). Our algorithms consistently outperform competing methods under different budgets. As B increases, the regret of *BHPT* $\mathcal{R}_B \to 0$ as discussed in Sec. 4.3.

Exploration vs. Exploitation. There are two factors that determine the performance of an algorithm: whether it correctly identifies the optimal arm, and whether it spends sufficient budget to achieve small loss on such arm. The first task reflects if the algorithm achieves effective exploration, such that it can accurately estimate the curves and identify the top arm, and the second task indicates sufficient exploitation. To examine the "exploration", we plot the hit rate of the output arm on the top 5 arms (out of all 84) across different budgets in Fig. 1(b)[6]. To check the "exploitation", we visualize the percentage of the budget spent on the output arm in Fig. 1(c). In both (b) and (c), the higher of the bar the better.

In (b), the proposed *BHPT* and *BHPT*-ε do better in exploration than all baselines. The column "adaptive prediction" in Table. 1 explains the exploration behavior of different methods. Specifically, the Bayesian belief model in *BO* and *BHPT* improve with more budgets, which leads to the increase in hit rate as budget increases. In (c), our methods perform well in terms of exploitation. The "early stop" column in Table. 1 partially explains the exploitation behavior: *BO* does strong exploitation (and poor exploration) under small budget because it does not early stop configurations.

Although *Rollout* has a belief model and does future predictions as *BHPT*, it doesn't perform well on neither task: the hit rate does not improve with more budget, nor does it exploit sufficiently on the output arm. Our conjecture is that the *Rollout* truncates the planning horizon due to the computa-

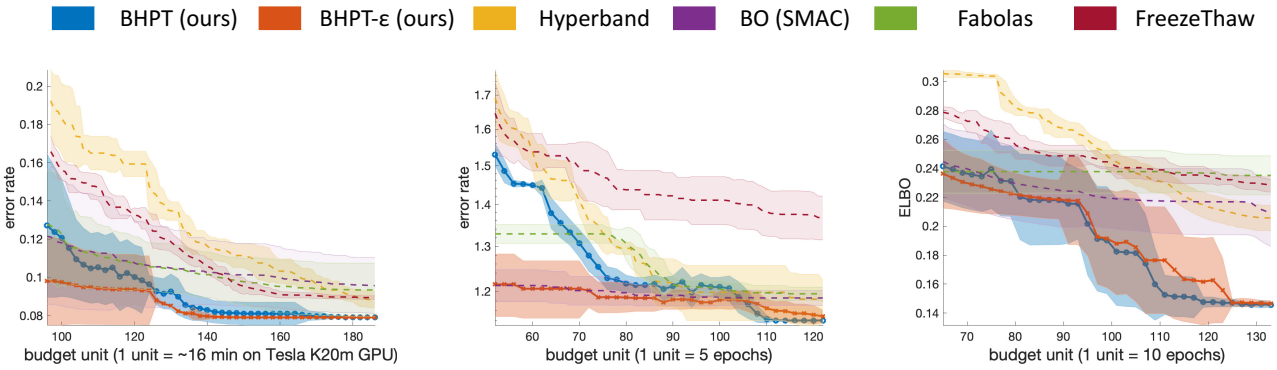
[6]The three stacks in each bar are the hit rate @ top-1, top-3, and top-5 in ground-truth respectively. The numerical values are reported in the supplementary material Sec. F.



(a) normalized regret $\mathcal{R}_B$ (b) exploration: output arm hit rate @ top 5 (c) exploitation: % budget on the output arm

In (a) *BHPT* outperforms all other methods, and $\mathcal{R}_B \to 0$ as B increases for *BHPT*. In (b) the hit rate of *BHPT* is the highest, i.e. correctly identifies the best arm.

Figure 1: budgeted optimization on 100 synthetic sets



(a) ResNet for classification (b) FCnet for classification (c) VAE for generative model

BHPT (blue) converges to the global optimal model at the rightmost budget for all tasks. *BHPT*-ε is better under small budgets, while *BHPT* is better under large budgets.

Figure 2: Real-world Hyper-parameter Tuning Tasks

tion challenge, which leads to myopic behaviors and the poor results. Indeed in all three subplots of Fig. 1, *Rollout* performs and behaves similarly to *Hyperband*, which only uses current performance to select actions. This demonstrates the importance of *long-term* predictions and planning in the budgeted tuning task.

Comparing (b) and (c), there is a clear tradeoff between exploration and exploitation: the hit rate decreases as the exploitation percentage increases. Note that *BHPT* adjusts this tradeoff automatically across different budgets. Compare the *BHPT* against its ε-greedy variant, the *BHPT*-ε does slightly better in exploitation, and worse in exploration.

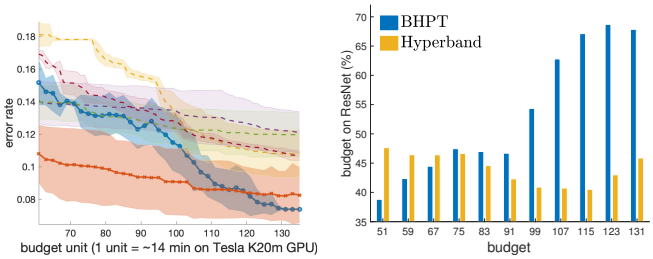
5.3 Results on Real-world Data

In this section, we report the tuning performances on real-world tuning tasks across different budget constraints. We plot the tuning outcomes (error rate or ELBO) over budgets in Fig. 2 and Fig. 3(a). Each curve is the average of 10 runs from different random seeds, and the mean with one standard deviation is shown in the figures.

BHPT methods work well under a wide range of budgets, and outperform *BO*, *Hyperband*, *FreezeThaw* and the state-of-the-art algorithm *Fabolas*, across 4 tuning tasks. The trend of different methods across budgets is consistent with the observations on the synthetic data.

As explained in the synthetic experiment, the vanilla *BHPT* does better in exploration while the ε-greedy variant does more exploitation. This explains the superior performance of the ε-greedy under small budgets. However, the lack of exploration results in worse belief model and damages the performance as the budget increases. This phenomenon is more salient on the architecture selection task Fig. 3(a), where the belief modeling is more challenging due to different learning curve patterns between ResNet and AlexNet.

Budget Adaptive Behaviors. An important motivation to study the budgeted tuning problem is that it is practically desirable to have adaptive strategy under different constraints. For example, the optimal network architecture might change under different budgets. We would like to examine whether the proposed *BHPT* exhibits such adaptive behavior. We take the architecture selection task between ResNet and AlexNet



(a) architecture selection between AlexNet and ResNet (b) budget allocation

Figure 3: On an architecture selection task, *BHPT* adapts its behavior to different budgets, while *Hyperband* employs a fixed strategy.

on CIFAR-10[7], and visualize the ratio of the resource spent on ResNet configurations over budgets in Fig. 3(b). ResNet configurations converge slower than AlexNet, but reach smaller error rates. Thus it is rewarding to focus the tuning on AlexNet under small budgets, and vice versa. Indeed, the *BHPT* allocates more resource to ResNet as the budget increases, compared to the baseline *Hyperband*, which samples the two networks more or less uniformly at random.

6 Conclusion

In this paper, we study the budgeted hyper-parameter tuning problem. We formulate a sequential decision making problem, and propose the *BHPT* algorithm, which uses long-term predictions with an action-value function to allocate the resource efficiently. It exhibits budget adaptive behavior, and achieves the state-of-the-art performance on real-world tasks.

Acknowledgments

We appreciate the feedback from the reviewers. This work is partially supported by NSF Awards IIS-1513966/1632803/1833137, CCF-1139148, DARPA Award#: FA8750-18-2-0117, DARPA-D3M-Award UCB-00009528, Google Research Awards, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

[7]The hyper-parameters include architecture type as well as others, like learning rate and etc. There are 49 configurations in total, with 24 ResNets and 25 AlexNets.

References

Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*, 2017.

Yutian Chen, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P Lillicrap, and Nando de Freitas. Learning to learn for global optimization of black box functions. *arXiv preprint arXiv:1611.03824*, 2016.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*, 2018.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. *arXiv preprint arXiv:1703.01785*, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*, 2018.

Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: a spectral approach. *arXiv preprint arXiv:1706.00764*, 2017.

Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248, 2016.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, pages 528–536, 2017.

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Advances in Neural Information Processing Systems*, pages 883–891, 2016.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

Robert Nishihara, David Lopez-Paz, and Léon Bottou. No regret bound for extreme bandits. In *AISTATS*, pages 259–267, 2016.

Ashish Sabharwal, Horst Samulowitz, and Gerald Tesauro. Selecting near-optimal learners via incremental data allocation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

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

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.