Bayesian Optimization Over Iterative Learners with Structured Responses: A Budget-aware Planning Approach

Syrine Belakaria[+] **Janardhan Rao Doppa**[+] **Nicolo Fusi**[†] **Rishit Sheth**[†]
Washington State University[+] Microsoft Research[†]

Abstract

The rising growth of deep neural networks (DNNs) and datasets in size motivates the need for efficient solutions for simultaneous model selection and training. Many methods for hyperparameter optimization (HPO) of iterative learners, including DNNs, attempt to solve this problem by querying and learning a response surface while searching for the optimum of that surface. However, many of these methods make myopic queries, do not consider prior knowledge about the response structure, and/or perform a biased cost-aware search, all of which exacerbate identifying the best-performing model when a total cost budget is specified. This paper proposes a novel approach referred to as **B**udget-**A**ware **P**lanning for **I**terative Learners (BAPI) to solve HPO problems under a constrained cost budget. BAPI is an efficient non-myopic Bayesian optimization solution that accounts for the budget and leverages the prior knowledge about the objective function and cost function to select better configurations and to take more informed decisions during the evaluation (training). Experiments on diverse HPO benchmarks for iterative learners show that BAPI performs better than state-of-the-art baselines in most cases.

1 INTRODUCTION

Hyperparameter optimization (HPO) for machine learning models, and pipelines is the task of automatic tuning of those parameters which affects model selection and training. A variety of HPO approaches have been developed for classical ML models, e.g., SVMs, random forests, utilizing Bayesian optimization (BO) (Snoek et al., 2012; Swersky et al., 2013), meta-learning (Feurer et al., 2015; Fusi et al., 2018), and ensembling (Thornton et al., 2013; Olson and Moore, 2016; LeDell and Poirier, 2020; Erickson et al., 2020) to name a few. For HPO, the inclusion of modern deep neural networks (DNNs) as a pipeline, introduces a temporal dimension to the problem of selecting pipeline queries due to the iterative nature of DNNs. Fixing the number of training epochs per query is clearly inefficient since poorly performing pipelines will waste resources, leading to more promising candidates not being identified when the resource budget is small. This paper considers the problem of HPO for such iterative learners (ILs) under a *fixed total budget*. In this setting, the budget is defined as some measure of resource cost for evaluating queries such as wall-clock time or energy. The key challenge is to reason about the available budget to intelligently select the candidate queries for evaluation to uncover high-quality configurations within the remaining budget. BO is known to be an effective framework to solve such problems. However, most BO algorithms ignore the fact that the cost of different configurations/queries can vary significantly and are unknown prior to their evaluation. We refer to this problem of utilizing BO to select amongst iterative learners under a constrained budget as *budget-aware Bayesian optimization*.

The key idea behind BO for HPO is to learn a response surface (e.g., Gaussian process) which serves as a surrogate for test set performance and use it to perform a sequence of queries by trading off exploration and exploitation. The response and cost modeling, and the planning of querying a sequence of different configurations have individually been addressed in the BO literature. To some extent, the interaction of these two components was studied as well. For ILs, modeling the side-information in the form of the shape of learning curves (i.e., accuracy vs. training epochs) will allow us to make fine-grained decisions such as early stopping to save resources. In our problem setting, we refer to the learning curve as the *structured response*. The structure of responses have been considered and modeled to varying degrees in methods that extrapolate performance to determine good candidates (Klein et al., 2017b; Domhan et al., 2015) as well as in BO over vector-valued responses (Wu et al., 2020; Nguyen et al., 2020). However, none of

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

these settings consider a fixed total budget.

To summarize the drawbacks of prior work for our problem setting: standard approaches for handling structured responses, heterogeneous cost-modeling (i.e., different queries have varying costs), and query selection have inefficiencies or are even inaccurate in some cases, especially when applied to the fixed budget setting. Standard cost-aware modeling for BO utilizes a separate model for cost predictions and then weights the selection of the next candidate query by these predictions (Snoek et al., 2012). However, cost-aware modeling via weighting suffers from a known pathology where the method tends to select low-cost queries with lower accuracy leading to an overall poor performance (Lee et al., 2020b; Astudillo et al., 2021). Fundamentally, the issue is one of mis-calibration between the response and cost models.

For query selection, non-myopic BO methods provide approximations of varying quality to the optimal solution defined by the Bellman recursion for BO (Osborne, 2010; Lam et al., 2016; Jiang et al., 2020a). These non-myopic methods (Lam et al., 2016; Wu and Frazier, 2019; Yue and Kontar, 2020; Jiang et al., 2020b; Lee et al., 2020a, 2021; Astudillo et al., 2021) use a variety of techniques to solve the sequence of nested integrations and maximizations including dynamic programming and rollouts. However, only the two most recent methods of Lee et al. (2021) and Astudillo et al. (2021) take the cost and a finite budget into account. Lee et al. (2021) proposed to leverage the known pathology of standard cost-aware modeling to promote early exploration but the resulting policy *does not adapt its horizon to the remaining budget*. Astudillo et al. (2021) also propose a non-myopic policy, but the *horizon adaptation of their method is post-hoc*, i.e. zero padding is used to fill the horizon after the evaluation budget is exhausted.

Contributions Our proposed solution BAPI executes a non-myopic query selection policy by wrapping standard BO in a layer of budget-aware planning for iterative learners. The key innovations of BAPI include leveraging side-information and expert knowledge such as the objective function's monotonicity, heteorgenity of query costs, and linearity of cost w.r.t training epochs into the planning procedure; overcoming known pathologies of standard cost-aware BO; and principled approach for adapting the horizon to the amount of remaining budget. Therefore, our technical contributions span both budgeted non-myopic BO and hyper-parameter optimization sub-areas. The list of synergistic contributions made by this paper are as follows:

- Development of a new approach for budget-aware non-myopic BO enabling an adaptive horizon to solve HPO problems for iterative learners. To the best of our knowledge, this is the first work that proposes a budgeted non-myopic approach specifically for HPO.
- Refining previous response modeling approaches by leveraging the monotonicity of the objective function through model derivatives to enable: (1) a new conservative stopping estimation approach to decide when a learner becomes ϵ-close to its asymptotic value, and (2) a general modeling approach with minimal assumptions about the shape of the response resulting in accurate extrapolation for improved decision-making.
- Design of a new efficient early termination method aimed to early stop the training of poorly performing HP configurations.
- A new alternative kernel for modeling the training cost of iterative learners while capturing the linearity of the cost w.r.t the number of epochs and its variability across different HPs
- Empirical evaluation on several state-of-the art benchmarks to demonstrate the performance of BAPI compared to algorithms designed for HPO, generic BO, and non-myopic BO and cost-aware non-myopic BO.

2 PROBLEM SETUP AND BACKGROUND

In this section, we state our problem and briefly review Gaussian processes, Bayesian optimization, and myopic vs. non-myopic query selection policies.

Consider the problem of sequentially optimizing a black-box objective function f over the input space $\mathcal{X}$ where the evaluation of each candidate input $\mathbf{x} \in \mathcal{X}$ is expensive and where the cost c of each input is unknown before the evaluation. In the context of HPO for iterative machine learning models, each input candidate $\mathbf{z} := [\mathbf{x}, t]$, where $\mathbf{x}$ represents model/pipeline hyperparameters and $t \in \mathcal{T}$=$[1 \dots t_{max}]$ is the number of training epochs. We let the objective function $f(\mathbf{x}, t)$ be defined as the accuracy [1] and the unknown cost $c(\mathbf{x}, t)$ be defined as the training time. The objective is to identify the maximum of f in a number of queries whose cumulative cost is bounded by a total budget B_T. Let $\mathcal{Z} := \mathcal{X} \times \mathcal{T}$, our problem can be stated as

$$\max_{Z \in P(\mathcal{Z})} \max_{\mathbf{z} \in Z} f(\mathbf{z}), \quad \text{s.t.} \sum_{\mathbf{z} \in Z} c(\mathbf{z}) \leq B_T \tag{1}$$

where $P(\mathcal{Z})$ denotes the power set of $\mathcal{Z}$ and $Z = \{\mathbf{z}_1 \dots \mathbf{z}_h\}$ is the sequence of inputs evaluated until the budget B_T is exhausted. In other words, the optimal HP $\mathbf{z}^*$ is defined as $\mathbf{z}^* \leftarrow \arg\max_{\mathcal{Z}} f(\mathbf{z})$ with $\mathbf{z}^* \in Z$ and $\sum_{\mathbf{z} \in Z} c(\mathbf{z}) \leq B_T$. The problem in Equation (1) is solved using a non-myopic policy, where at each iteration, the algorithm accounts for the sequence of inputs that can be evaluated within the remaining budget, i.e., the horizon h is *adaptive*. We define the non-myopic setting later in this section, which is similar to the setting considered in Lee et al. (2021) and Astudillo et al. (2021).

[1] Any bounded metric can be used (e.g, loss, some cases of reward for RL models etc.)

We focus on problem settings where the objective function is monotonic in the number of epochs t. Specifically, for a fixed hyperparameter $\mathbf{x}$, $f(\mathbf{x}, t) \leq f(\mathbf{x}, t')$ when $t \leq t'$. This is a reasonable assumption for iterative learners. Even if monotonicity does not hold over all training epochs, keeping track of the best model over training epochs is a standard practice (Dai et al., 2019).

Gaussian Processes GPs are characterized by a mean function m and a covariance or kernel function K. If a function f is sampled from GP(m, K), then $f(\mathbf{x}, t)$ is distributed normally $\mathcal{N}(m(\mathbf{x}, t), K([\mathbf{x}, t], [\mathbf{x}, t]))$ for a finite set of inputs from $[\mathbf{x}, t] \in \mathcal{X} \times \mathcal{T}$. The predictive mean and uncertainty for a GP for a new input $\mathbf{z}_* \in \mathcal{Z}$ is defined as:

$$\mu(\mathbf{z}_*) = K_{\mathbf{z}_*,Z}[K_{Z,Z} + \sigma^2 I]^{-1}(Y - m(Z)) + m(\mathbf{z}_*)$$
$$\sigma^2(\mathbf{z}_*) = K_{\mathbf{z}_*,\mathbf{z}_*} - K_{\mathbf{z}_*,Z}[K_{Z,Z} + \sigma^2 I]^{-1} K_{Z,\mathbf{z}_*}$$

where $K_{\mathbf{z}_*,\mathbf{z}_*} = K(\mathbf{z}_*, \mathbf{z}_*)$, $K_{Z,Z} = K(Z, Z)$, $K_{\mathbf{z}_*,Z} = [K(\mathbf{z}_*, \mathbf{z}_i)]_{\forall i}$, Z is the set of evaluated inputs and Y is their corresponding function values. A typical choice to model blackbox functions with a temporal component is using a product kernel $K([\mathbf{x}, t], [\mathbf{x}', t']) = K_{\mathbf{x}}(\mathbf{x}, \mathbf{x}') \times K_t(t, t')$. $K_{\mathbf{x}}$, defined over the input space $\mathbf{x} \in \mathcal{X}$, is often selected to be an RBF or a Matern kernel. For the temporal component K_t, previous work for GPs over iterative learners (Swersky et al., 2014) proposed an exponential decay (ED) kernel, defined as $K_t(t, t') = \beta^\alpha / (t + t' + \beta)^\alpha$, to model decreasing covariance with increasing time. However, this kernel does not guarantee that the predictive mean of GP or sampled functions would necessarily follow a desired monotonic shape. Nguyen et al. (2020) argued that the use of ED is not appropriate for reinforcement learning models where the reward might follow a logistic shape and proposed the use of an RBF kernel for K_t.

Bayesian Optimization And Non-Myopic Query Policies BO is a sequential, model-based approach for optimizing blackbox functions (Shahriari et al., 2015; Belakaria et al., 2019; Deshwal et al., 2021). BO is often performed with the specification of a GP prior over the function, and an acquisition function. The GP posterior acts as a surrogate for the true unknown response. The potential or utility of points in the input space to be the optimizer is scored by the acquisition function. Two examples of acquisition functions are expected improvement (EI) and upper confidence bound (UCB) and both are considered myopic since they only aim to maximize the function for the next query without accounting for the future queries.

We review some standard facts for optimal sequential decision-making (Osborne, 2010; Jiang et al., 2020a). Consider having collected a set of i responses D_i and let u denote the utility of D_i for maximizing $f(\mathbf{z}) = y$, i.e., $u(D_i) = \max_{(\mathbf{z},y) \in D_i} y$. The marginal gain in utility of the query $\mathbf{z}$ w.r.t. D_i is expressed as:

$$u(y|\mathbf{z}, D_i) = u(D_i \cup (\mathbf{z}, y)) - u(D_i) \tag{2}$$

The one-step expected marginal gain is equivalent to the expected improvement (EI) strategy (Močkus, 1975):

$$\mathcal{U}_1(\mathbf{z}|D_i) = \mathbb{E}_y[u(D_i \cup (\mathbf{z}, y)) - u(D_i)|\mathbf{z}, D_i] \tag{3}$$

Now, consider the case where r steps are remaining. The r-steps expected marginal gain can be expressed through the Bellman recursion as (Jiang et al., 2020a):

$$\mathcal{U}_r(\mathbf{z}|D_i) = \mathbb{E}_y[u(y|\mathbf{z}, D_i)] + \mathbb{E}_y[\max_{\mathbf{z}'} \mathcal{U}_{r-1}(\mathbf{z}'|D_i \cup (\mathbf{z}, y))] \tag{4}$$

Maximizing (4) w.r.t. $\mathbf{z}$ results in the optimal r-steps "lookahead" . Being a sequence of r nested integrals of maximizations, optimizing (4) is intractable for even small r. **Lower Bound To Optimal Policy** The previous discussion focused on the optimality of selecting single queries. We review recent work by (Jiang et al., 2020a) which makes a connection between single selection and batch selection of size r, $Z = \{\mathbf{z}_1 \dots \mathbf{z}_r\}$. Assuming parallel evaluation, the optimal set of selected points Z^* maximizes the expected marginal utility of the new associated evaluations $Y = \{y_1 \dots y_r\}$:

$$\begin{aligned} Z^* &= \arg\max_{Z \in \mathcal{Z}} \mathcal{U}(Z|D_i) \\ \text{with} \quad & \mathcal{U}(Z|D_i) = \mathbb{E}_Y[u(Y|Z, D_i)] \end{aligned} \tag{5}$$

Jiang et al. (2020a) showed that choosing a query $\mathbf{z}^* \in Z^*$ is equivalent to solving $\arg\max_{\mathbf{z}} V(\mathbf{z}|D)$ where

$$\begin{aligned} V(\mathbf{z}|D_i) = {} & \mathbb{E}_y[u(y|\mathbf{z}, D_i)] \\ & + \max_{Z' : |Z'| = r-1} \mathbb{E}_y[\mathcal{U}(Z'|D_i \cup (\mathbf{z}, y))] \end{aligned} \tag{6}$$

and that the second term of (6) is a lower bound to the second term in (4):

$$\begin{aligned} & \max_{Z' : |Z'| = r-1} \mathbb{E}_y[\mathcal{U}(Z'|D_i \cup (\mathbf{z}, y))] \\ & \leq \mathbb{E}_y[\max_{\mathbf{z}'} \mathcal{U}_{r-1}(\mathbf{z}'|D_i \cup (\mathbf{z}, y))] \end{aligned} \tag{7}$$

Given this observation, Jiang et al. (2020a) proposed approximating the optimal policy (4) by optimizing its lower bound (6) which is equivalent to optimizing the batch EI known as q-EI. Jiang et al. (2020a) proposed using joint q-EI which is budget-unaware and scales poorly with increased dimensions (Wilson et al., 2018).

3 PROPOSED APPROACH

In this section, we start by providing a high-level overview of the proposed BAPI algorithm and briefly explain its key components. Next, we provide complete details of each component. First, we describe how to perform an efficient budget-aware non-myopic search. Second, we explain our approach to model structured response for iterative learning which can be used to estimate conservative stopping for increased resource-efficiency. Finally, after describing an alternative kernel for the cost model, we provide the full BAPI approach with all its component coherently put together.

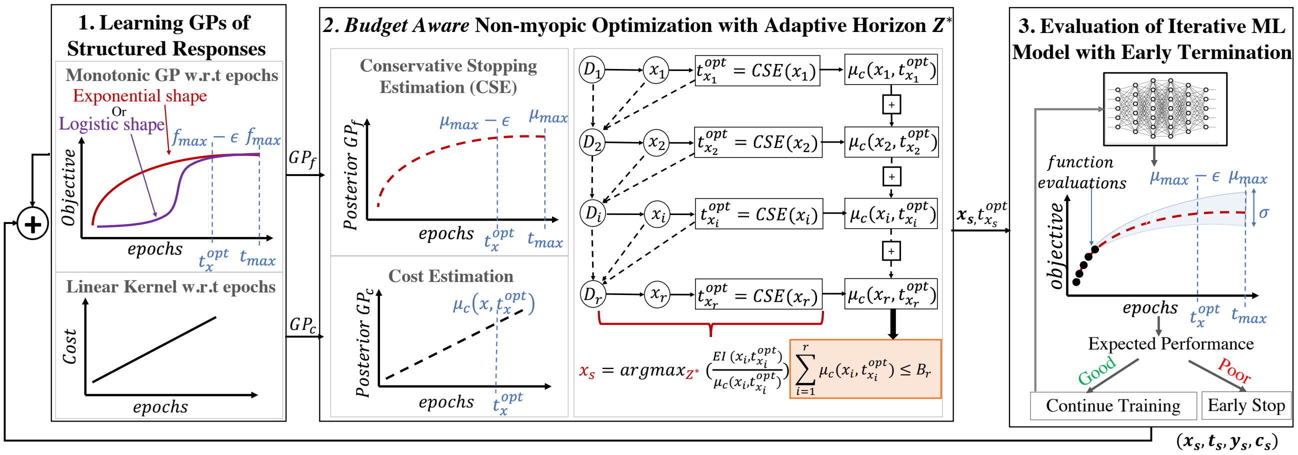

1. Learning GPs of Structured Responses
Monotonic GP w.r.t epochs
Exponential shape
Or
Logistic shape
$f_{max} - \epsilon$ f_{max}
Objective
epochs
t_x^{opt} t_{max}
Linear Kernel w.r.t epochs
Cost
epochs
GP_f
GP_c
2. Budget Aware Non-myopic Optimization with Adaptive Horizon Z^*
Conservative Stopping Estimation (CSE)
$\mu_{max} - \epsilon$ μ_{max}
Posterior GP_f
epochs
t_x^{opt} t_{max}
Cost Estimation
$\mu_c(x, t_x^{opt})$
Posterior GP_c
epochs
t_x^{opt}
D_1 x_1 $t_{x_1}^{opt} = CSE(x_1)$ $\mu_c(x_1, t_{x_1}^{opt})$
D_2 x_2 $t_{x_2}^{opt} = CSE(x_2)$ $\mu_c(x_2, t_{x_2}^{opt})$
D_i x_i $t_{x_i}^{opt} = CSE(x_i)$ $\mu_c(x_i, t_{x_i}^{opt})$
D_r x_r $t_{x_r}^{opt} = CSE(x_r)$ $\mu_c(x_r, t_{x_r}^{opt})$
+
$x_s = argmax_{Z^*} \left(\frac{EI(x_i, t_{x_i}^{opt})}{\mu_c(x_i, t_{x_i}^{opt})}\right)$
$\sum_{i=1}^{r} \mu_c(x_i, t_{x_i}^{opt}) \leq B_r$
$x_s, t_{x_s}^{opt}$
3. Evaluation of Iterative ML Model with Early Termination
function evaluations
$\mu_{max} - \epsilon$ μ_{max}
σ
objective
epochs
t_x^{opt} t_{max}
Expected Performance
Good
Poor
Continue Training
Early Stop
(x_s, t_s, y_s, c_s)

Figure 1: Overview of BAPI algorithm illustrating its three key components explained in section 3.1

3.1 Overview Of BAPI Algorithm

Let GP_f and GP_c be the surrogate probabilistic models learned given a set of observed data points D_i of the objective function f and the cost function c respectively. Let μ_c and σ_c^2 be the predictive mean and variance of GP_c and μ and σ^2 be the predictive mean and variance of GP_f .
As shown in Figure 1, BAPI is a sequential algorithm with three key components listed below:

1. Learning Surrogate Models We build two surrogate models GP_f for the objective function and GP_c for the cost function by fitting independent GPs using queries evaluated in the past. We enforce shape constraint on the posterior of GP_f with respect to t (epoch number) to incorporate prior knowledge about the monotonicity of the function. We use a special kernel for GP_c to leverage our knowldge about the variability of the cost across different HPs and its linearity with respect to t.

2. Budget Aware Non-Myopic Optimization With Adaptive Horizon We perform non-myopic optimization to approximate the optimal lookahead horizon Z^* defined as the potential sequence of inputs that can be evaluated until their conservative stopping t_x^{opt} without violating the remaining budget B_r: $Z^* = \{(\mathbf{x}_1, t_{\mathbf{x}_1}^{opt}) \dots (\mathbf{x}_r, t_{\mathbf{x}_r}^{opt})\}$ such that $\sum_{\mathbf{z} \in Z^*} c(\mathbf{z}) \approx \sum_{i=1}^{r} \mu_c(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt}) \leq B_r$. While constructing the horizon Z^*, each input $\mathbf{x}_i$ is selected based on its expected improvement. The associated conservative stopping $t_{\mathbf{x}_i}^{opt}$ and cost $\mu_c(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt})$, are estimated upon the input selection.

3. Evaluation With Early Termination From Z^* obtained from the second step, we select the input with highest expected improvement per unit cost *at its estimated conservative stopping* for evaluation. After training the model for a fraction of the maximum number of epochs, we re-estimate the performance of the input at its conservative stopping epoch. We early terminate the training if the expected performance is poor with high certainty.

3.2 Budget Aware BO With Adaptive Horizon

Approximating the non-myopic optimization with the batch expected improvement q-EI, where the batch size q is equal to the horizon of the lookahead optimization r, is an efficient approach (Jiang et al., 2020a). However, the *joint* q-EI via reparametrization trick and Monte Carlo sampling proposed in Wang et al. (2016a) and used in Jiang et al. (2020a) requires the size of the batch to be fixed and solves a *joint one-shot* optimization problem of $(d \times q)$ dimensions.

Challenges 1) In the context of budgeted non-myopic optimization, the horizon of remaining queries r is unknown: it depends on remaining budget B_r and expected costs of horizon queries $\mathbf{z}_i, i \in \{1 \dots r\}$. An efficient method should allow the horizon to be adaptive to the budget. Therefore, the *joint* q-EI is not a suitable solution. 2) Given an optimization problem with reasonable medium-size dimension and a medium length horizon, the dimensionality of the joint optimization problem can significantly increase. Wilson et al. (2018) showed that the performance of the joint q-EI deteriorates for large optimization dimension.

Proposed Alternative To overcome the above two challenges, we propose to employ the sequential greedy q-EI via reparametrization trick and Monte Carlo sampling proposed in Wilson et al. (2018). Wilson et al. (2018) showed that q-EI is a submodular acquisition function, which guarantees a near-optimal maximization via a sequential greedy approach. This incremental version of the acquisition function has several distinct advantages over the joint one: 1) It is amenable to an adaptive horizon, where we can stop adding points to the batch based on the remaining budget. 2) It is more efficient and produces better performance when the value of $d \times q$ is high (Wilson et al., 2018). After the batch approximation returns a sequence (horizon) of inputs, we select one input to query its expensive function evaluation. We discuss an input selection strategy, that is relevant to iterative machine learning models optimization, in section

3.4. Note that our approximation can clearly extend to the use of any other batch acquisition function that satisfies the submodularity condition and have a sequential greedy approach, namely the q-UCB and q-PI (Wilson et al., 2018).

It is important to highlight that in practice, our approach can naturally extend to parallel BO evaluation (batch setting). The user can choose more than one point from the approximated horizon and evaluate them in parallel as long as the horizon length is fairly larger than the number of selected points for evaluation. Even though in this paper we focus on the sequential setting, we enable this option in our implementation. We provide a summary of the budget-aware BO approach in Algorithm 1.

Algorithm 1 Budget Aware Non-myopic BO

```
Input: $\mathcal{Z}$, $f(\mathbf{z}), c(\mathbf{z})$, models $GP_f, GP_c$, utility function $u(y|\mathbf{z}, D)$, a total budget $B_T$
Output: $D, \mathbf{z}^*, f(\mathbf{z}^*)$
Initialize the remaining budget $B_r \leftarrow B_T$
while $B_r \geq 0$: do
    Approximate the optimal horizon via adaptive optimal batch computation $Z^*$ of size $r$ such that $\sum_{i=0}^{r} \mu_c(\mathbf{z}_i) \leq B_r$
    Select a candidate input $\mathbf{z}^* \in Z^*$ and observe its evaluation $f(\mathbf{z}^*) = y^*$ and cost $c(\mathbf{z}^*) = y_c^*$
    Update the remaining budget $B_r \leftarrow B_r - c(\mathbf{z}^*)$
    Update data $D = D \cup \{(\mathbf{x}^*, y^*, y_c^*)\}$
```

3.3 Structured Responses

In this section, we describe our proposed approaches to leverage prior knowledge about the structure of the responses, namely, the monotonicity and shape of the function f and the linearity of the cost c.

We propose to use a GP with monotonicity constraint over the t variable to model the function f. Recent work (Agrell, 2019) proposed an efficient approach to introduce linear operator inequality constraints to GPs. Let f be the function modeled by a GP and $\mathcal{L}$ be a linear operator. The proposed approach enables the posterior prediction to account for inequality constraints defined as $a(\mathbf{z}) \leq \mathcal{L}f(\mathbf{z}) \leq b(\mathbf{z})$. The derivative operator is a linear operator. Hence, to apply monotonicity, this condition can be seen as the partial derivative of the model of f with respect to t is positive. For this condition to hold, Agrell (2019) proposed to define a set of *virtual observation locations* $Z^v = \{\mathbf{z_1}^v, \ldots, \mathbf{z_s}^v\}$ where the condition is guaranteed to be satisfied.

The posterior predictive distribution of the monotonic GP is $\mathbf{f}^*|Y, C$, which is the distribution of $\mathbf{f}^* = f(\mathbf{z}_*)$ for some new inputs $\mathbf{z}_* = [\mathbf{x}_*, t_*]$, conditioned on the observed data Y and the constraint C defined as $a(Z^v) \leq \mathcal{L}f(Z^v) \leq b(Z^v)$. The final derivation of the predictive distribution, provided by Agrell (2019), is defined as follows:

$$\mathbf{f}^*|Y, C \sim \mathcal{N}(\mu^* + A(\mathbf{C} - \mathcal{L}\mu^v) + B(Y - \mu), \Sigma)$$

$$\mathbf{C} = \widetilde{C}|Y, C \sim \mathcal{TN}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1, a(Z^v), b(Z^v))$$

where $\mathcal{TN}(\cdot, \cdot, a, b)$ is the truncated Gaussian $\mathcal{N}(\cdot, \cdot)$ conditioned on the hyper-rectangle $[a_1, b_1] \times \cdots \times [a_k, b_k]$, $\mu^v = m(Z^v), \mu^* = m(\mathbf{z}_*), \mu = m(Z)$. The definition of the matrices A, B, A_1, B_1 and Σ can be found in Appendix A. The computation of the posterior of the monotonic GP requires the definition of derivatives of the kernel function. In this work we consider monotonicity with respect to one dimension t. Therefore, we need the first order derivatives.

In cases where the function is known to be exponentially decaying (e.g., neural network training), the kernel over dimension t should be defined as an ED kernel. However, in cases where the shape of learning curve is monotonic but not necessarily exponentially decaying (e.g., cumulative and average reward for RL models), an RBF kernel with monotonicity over dimension t should be used. Leveraging monotonicity in the modeling allows flexibility and the generalization of our approach for several types of ILs. We provide the derivatives for both kernels and the details about the specification of the location of virtual observations with each kernel in Appendix A and in our implementation. We additionally provide insights about the efficient posterior computation of the monotonic GP. For more details, we refer the reader to Agrell (2019).

Conservative Stopping Estimation Previously proposed BO approaches for HPO consider a maximum number of epoch t_{max} at which the objective function will reach its best value. However, in practice, different HPs do not need to necessarily run to the maximum number of epochs to reach their optimal value as the objective stops improving (reaches a plateau) Kaplan et al. (2020). Therefore, running them for longer epochs can be a waste of limited resource budget with diminishing returns. Existing work proposed early stopping of HPs based on their performance compared to previously evaluated data points (Li et al., 2017; Dai et al., 2019; Swersky et al., 2014) or based on the expected improvement per unit cost (Nguyen et al., 2020) which leads to the selection of very low number of epoch due to the high cost of t_{max}. We propose to define a conservative stopping $t^{opt}_{\mathbf{x}}$ for each HP $\mathbf{x}$ as the smallest number of epoch needed to reach the best function value at $\mathbf{x}$. Our approach enables the estimation of when a learner becomes ϵ-close to its asymptotic value. To the best of our knowledge, no previous work used the estimation of the function values at another location to reason about the HP selection and optimal early termination before reaching that epoch. The problem of estimating $t^{opt}_{\mathbf{x}}$ for a HP $\mathbf{x}$ based on the GP posterior is defined as below and efficiently solved using binary search.

$$\begin{aligned} t^{opt}_{\mathbf{x}} &\leftarrow \arg min_{t \in [t_{min}, t_{max}]}\ t \\ s.t\ \mu(\mathbf{x}, t_{max}) &- \mu(\mathbf{x}, t) \leq \epsilon \end{aligned} \tag{8}$$

Cost Modeling The cost prediction is an important component in our algorithm. Therefore, it is important to have an accurate and informative model for the cost. We propose to model the cost by an independent Gaussian process GP_c that captures two important characteristics: 1) The cost of the training of different HPs for the same number of iterations t can vary significantly. 2) The cost of training of a fixed HP $\mathbf{x}$ increases linearly with the number epochs t. We propose to use the product kernel $K_c([\mathbf{x}, t], [\mathbf{x}', t']) = K_{c_x}(\mathbf{x}, \mathbf{x}') \times K_{c_t}(t, t')$, where K_{c_x} is an RBF kernel over $\mathbf{x}$ and K_{c_t} is a linear kernel over t. Note that previous work assumes the cost is same for different HP $\mathbf{x}$ and linear with respect to t. This might lead to an inaccurate estimation of the cost especially if some of the dimensions of $\mathbf{x}$ represent architectural variables (e.g number of layers, number of hidden nodes etc.)

3.4 Budget-Aware Planning For Iterative Learners (BAPI)

In this section, we describe the overall budget-aware non-myopic BO algorithm for HPO of iterative learners. The main idea is to use the reparametrized iterative greedy q-EI proposed in Wilson et al. (2018) to approximate the optimal sequence of selections with respect to the available budget. q-EI will have an adaptive batch size with budget exhaustion as a stopping criteria. We propose to adaptively add inputs to the horizon based on their expected improvement at their conservative stopping iteration without normalizing the utility function by the cost during the optimization. The details of execution can be found in the Non-Myopic Optimization (NMO) function described in Algorithm 4. This function returns a set of inputs representing the optimal horizon Z^*.

Input Selection From Horizon Given the set of inputs Z^*, how to select the next input to evaluate? We propose to select the input with the highest immediate expected reward per unit cost *at its conservative stopping iteration*. We note here that this is *different* from optimizing the utility function per unit cost and the issue of selecting low non-informative number of iterations would not arise. In this case, the number of iterations is already fixed to an optimal high value for each input $\mathbf{x}^*$.

Early Termination After selecting the next candidate HP to evaluate, the function evaluation will return a y_t value after each epoch. Based on the function values of the initial p epochs, we can re-estimate the final performance of $\mathbf{x}$ and its new conservative stopping $t_{\mathbf{x}}^{opt_n}$. The algorithm makes a decision to continue model training with the current HP or early-stop it. If both 1. $\mu(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq y_{best}$, and 2. $\sigma(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq \tau\sigma(\mathbf{x}, t)$, then model training will be early stopped in epoch t, otherwise, the conditions will be verified again after running another set of p epochs or when it reaches the estimated $t_{\mathbf{x}}^{opt_n}$, whichever happens earlier. The first condition will recommend stopping the training if the predicted function value at $t_{\mathbf{x}}^{opt_n}$ will not be higher than the current best value achieved across all evaluated HP. The second condition recommends the early stopping only if the uncertainty of the model about the predicted function value at the estimated conservative stopping is no more than a factor $\tau \geq 1$ of the uncertainty of the model about the last evaluated epoch. In another word, condition 2 will prevent the early stopping if the model is not certain enough about its prediction of the function value at $t_{\mathbf{x}}^{opt_n}$. Algorithm 5 summarizes evaluation with early termination.

Algorithm 2 BAPI

```
Input: $\mathcal{X}$; $f(\mathbf{x}, t)$;$c(\mathbf{x}, t)$; $t_{max}$; $B_T$
Output:$\mathbf{x}^*, t_{\mathbf{x}^*}^{opt}, f(\mathbf{x}^*, t_{\mathbf{x}^*}^{opt})$
Initialize with $N_0$ initial points
Fit the models: $GP_f, GP_c$
$B_r \leftarrow B_T - \sum_{i=0}^{N_0} c(\mathbf{x}_i, t_{\mathbf{x}_i})$
while $B_r > 0$ do
    # Find the budget constrained horizon and their corresponding conservative stopping
    $Z^* : \{(\mathbf{x}_1, t_{\mathbf{x}_1}^{opt}) \cdots (\mathbf{x}_r, t_{\mathbf{x}_r}^{opt})\} \leftarrow NMO(GP_f, GP_c, B_r)$
    # Select one point for evaluation
    $\mathbf{x}, t_{\mathbf{x}}^{opt} \leftarrow \arg max_{Z^*} \frac{EI(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt})}{\mu_c(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt})}$
    $\mathbf{y}, \mathbf{y}_c \leftarrow Evaluate(f(\mathbf{x}, t_{\mathbf{x}}^{opt}))$
    Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, \mathbf{y}, \mathbf{y}_c)\}$
    Update Models: $GP_f, GP_c$
    $B_r \leftarrow B_r - c(\mathbf{x}, t_{\mathbf{x}})$
```

Algorithm 3 Conservative Stopping Estimation

```
$ConservativeStopping(GP_f, \mathbf{x})$
$t_{\mathbf{x}}^{opt} \leftarrow \arg min_{t \in [t_{min}, t_{max}]} t$
    s.t $\mu(\mathbf{x}, t_{max}) - \mu(\mathbf{x}, t) < \epsilon$
Return $t_{\mathbf{x}}^{opt}$
```

Algorithm 4 Adaptive Horizon q-EI AFO

```
$NMO(GP_f, GP_c, B_r)$
$Z^* = \{\}$
while $B_r > 0$ do
    # Add $\mathbf{x}$ based on the highest number of epochs
    $\mathbf{x} \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}} q\text{-EI}(\mathbf{x}, t_{max})$
    # Estimate the conservative stopping for $\mathbf{x}$
    $t_{\mathbf{x}}^{opt} \leftarrow ConservativeStopping(GP_f, \mathbf{x})$
    Deduct estimate cost at $t_{\mathbf{x}}^{opt}$ from budget
    $B_r \leftarrow B_r - \mu_c(\mathbf{x}, t_{\mathbf{x}}^{opt})$
    $Z^* = Z^* \cup \{(\mathbf{x}, t_{\mathbf{x}}^{opt}\}$
Return $Z^*$
```

Algorithm 5 Evaluate Function

```
$Evaluate(f(\mathbf{x}, t_{\mathbf{x}}^{opt}))$
$t \leftarrow p$
while $t \leq t_{max}$ and $Continue$ do
    $\mathbf{y} = f(\mathbf{x}, t)$ ; $\mathbf{y}_c = c(\mathbf{x}, t)$
    $t_{\mathbf{x}}^{opt_n} = ConservativeStopping(GP_f, \mathbf{x})$
    if $\mu(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq y_{best}$ and $\sigma(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq \tau\sigma(\mathbf{x}, t)$:
        $Continue \leftarrow False$
    else:
        $Continue \leftarrow True$
        $t \leftarrow min(t_{\mathbf{x}}^{opt_n}, t + p)$
Return $\mathbf{y}, \mathbf{y}_c$
```

Data Points Selection From Learning Curve Iterative machine learning models evaluated with an input configuration $\mathbf{x}^*$ and a number of epochs t^* return a vector of t^* function values and a vector of t^* cost values associated with each iteration $t \leq t^*$. Most of existing work, do not utilize these data points and use only the function value at the last epoch. However, leveraging part of this data can help the learning of the monotonic shape of the objective function and result in a more accurate extrapolation. We select, from each curve, the points with the highest model uncertainty(variance) following the approach proposed in Nguyen et al. (2020).

Practical Considerations Considering a perfect model of the function, querying a complete lookahead horizon in each iteration would be optimal. However, as pointed out by previous work on non-myopic BO (Jiang et al., 2020a; Lee et al., 2021, 2020a; Yue and Kontar, 2020), the model is usually uncertain about long term predictions. Consequently querying a long horizon can hurt the optimization by evaluating misleading points and causing a higher computational cost. Therefore, we follow previous work (Astudillo et al., 2021) and set a maximum horizon length as an additional stopping condition to the size of the horizon. Given this mitigation, we expect that the horizon adaptation to the budget to occurs depending on the remaining budget. Additionally, selected points would always be within the limits of the remaining budget.

Cost of a Restarted Hyperparameter In iterative learning, the optimization algorithm might select a configuration that was previously evaluated for a lower number of epochs. However, the cost will always be estimated with respect to the evaluation iteration. For an accurate optimization, our algorithm handles this special case by assigning a cost that only reflect the additional epochs to be run. This is accounted for in the non-myopic optimization function, input selection, and budget deduction after function evaluation.

4 RELATED WORK

Our problem setting and the proposed BAPI solution have many intersections with previous work in hyperparameter optimization, Bayesian optimization, non-myopic optimization, and sequential decision making which we attempt to summarize here.

HPO/BO For Iterative Learning Domhan et al. (2015) proposed learning curve prediction in order to allow early termination of non-promising candidates. This approach utilizes approximate Bayesian inference w.r.t. a pre-defined finite set of learning curve models to perform extrapolation to a fixed horizon. Klein et al. (2017b) built on this method by showing Bayesian neural networks could be used for learning curve prediction. Swersky et al. (2014) proposed a hierarchical GP model for HP tuning that includes learning curve prediction upon which decisions for exploration (freeze current and test new candidate) and exploitation (thaw current and continue learning) are based. More recently, Dai et al. (2019) proposed an optimal stopping procedure for increasing the sample efficiency of BO and showed competitive performance with Domhan et al. (2015) for iterative learners. While this procedure obtains theoretical guarantees, it must generate a sample of large size from the GP in order to make a reliable decision. Moreover, solving for the stopping rule requires an approximate backward induction technique after each epoch. Our proposal for early termination is conceptually simpler and far less computationally demanding.

The method of Lu et al. (2019) considers a finite set of learners modeled with freeze/thaw (Swersky et al., 2014) selecting between exploration and exploitation with a heuristic ϵ-greedy rule. Wu et al. (2020) extend the knowledge gradient acquisition function (Frazier et al., 2008) to trace-valued observations that occur in multi-fidelity applications. BOIL (Nguyen et al., 2020) also considers trace-valued observations, but compresses the trace via learned weighted sum as well as adding carefully-chosen intermediate trace values as observations. The setting for BOIL includes reinforcement learning problems with reward functions taking non-exponentially decaying shapes. Our BAPI approach uses a product kernel to jointly model correlations between HPs and epochs within the iterative training procedure.

Kandasamy et al. (2017) developed BOCA, an extension of UCB to general multi-fidelity BO setting.

We note that there exist orthogonal approaches that focus on the ability to extrapolate responses based on smaller datasets where the cost is varied based on the size or fraction of the dataset used for training. These methods propose algorithms based on multi-task BO (Swersky et al., 2013; Klein et al., 2017a) and importance sampling (Ariafar et al., 2021).

Non-Myopic Policies While there were early attempts at non-myopic selection for length-two horizons (e.g., Osborne, 2010), most work on proposing practical methods for longer horizons is very recent. Wu and Frazier (2019) developed gradient estimates for two-step EI admitting gradient-based search for the optimal two-step selection. Lam et al. (2016) utilized a Markov decision process (MDP) formalism and performed rollouts with a predefined base policy to estimate the value function. GLASSES (González et al., 2016b) approximates the solution for the optimal non-myopic selection by a combination of approximate integration given future selections and approximating the future selections by a diversity-promoting batch selection procedure from González et al. (2016a)).

Closely-related to our work is BINOCULARS Jiang et al. (2020a) which was discussed in Section 2. The major differences to our proposed BAPI approach are that (a) BINOCULARS uses *joint* batch expected improvement q-EI Wang et al. (2016a) while we use the sequential greedy selection Wilson et al. (2018), (b) BINOCULARS uses a fixed horizon that is budget agnostic while we use a budget-adaptive horizon, and (c) BINOCULARS does not take cost into account when returning its non-myopic selected query while our method does factor cost in. Lee et al. (2021) consider the general cost-aware setting and frame the problem as a constrained MDP. Their method approximately solves the intractable problem by performing rollouts of a base policy that does not adapt the horizon to the budget. Moreover, it's base policy is normalized by cost leading to a bias towards low-cost queries. Building upon the efficient one-shot multi-step tree approach of Jiang et al. (2020b), Astudillo et al. (2021) introduce cost-modeling and develop a budget-aware method. However, this method's adaptation to the horizon is post-hoc in the sense that the horizon has to be fixed in advance and cannot be adaptive to the remaining budget due to utility function formulation and optimization. This leads to an unnecessary higher dimensional optimization and a manual zero padding technique to handle cases where the selected horizon violates the remaining budget.

Bandit Algorithms Given that the objective in (1) is equivalent to optimizing for simple regret, there is a large amount of relevant work within the multi-armed bandits literature. Audibert et al. (2010) developed the upper confidence bound exploration (UCB-E) policy, for the best arm identification (BAI) in the budgeted setting by providing conditions under which simple regret decays exponentially with increasing budget. Hoffman et al. (2014) considered linear bandits and proposed the BayesGap algorithm which is an exploration policy within budget constraints. Later, Jamieson and Talwalkar (2016) analyzed successive halving as an instance of non-stochastic multi-armed bandits in the setting where the budget is greater than the number of learners. HyperBand (Li et al., 2017) is an implementation of successive halving running this algorithm in multiple successive rounds and is a very general algorithm for HPO including non-iterative learners. Most recently, BOHB (Falkner et al., 2018) modified HyperBand by utilizing BO within the successive halving procedure which guides the selection process for the learners that will be trained for longer budgets.

5 EXPERIMENTS AND RESULTS

In this section, we first provide details about our experimental setup. Next, we evaluate the performance of BAPI approach and compare it to several state-of-the-art baselines.

Baselines. We evaluate state-of-the-art baselines, described in the related work: from cost-aware non-myopic BO literature `BMS-EI`[2] (Astudillo et al., 2021), from non-myopic BO BINOCULARS `(BINOC)` (Jiang et al., 2020a)[3]and `MS-EI` (Jiang et al., 2020b)[3], from general BO literature `EI` (Jones et al., 1998), from HPO for iterative learners literature `BOHB` (Falkner et al., 2018)[4] and HyperBand `(HB)` (Li et al., 2017)[4], from multi-fidelity BO for HPO literature `BOIL` Nguyen et al. (2020)[5]. Each baseline implementation uses settings recommended by the original authors and publicly available code. We also evaluated GLASSES (González et al., 2016b)[3] and random search. However, both of them performed always poorly when compared to all other baselines. Therefore, for clarity of the figures, we do no report them. We note that previous work on non-myopic BO do not include HB and BOHB as baselines but given their competitive performance in iterative learning settings, we recommend they become standard in future work in this problem setting.

Experimental Setup All experiments were averaged over 10 runs with different random seeds. The code of our BAPI implementation is publicly available[6]. We considered several state-of-the-art HPO benchmarks: 1) Logistic regression with MNIST dataset; 2) Multi-layer perceptron with Olivetti dataset; 3) Multi-layer perceptron with Covtype dataset ; 4) Fully connected network with MNIST dataset with two different t_{max} setups 5) CNN on image dataset CIFAR10 with two different t_{max} setups; 6) CNN on SVHN dataset with two different t_{max} setups; 7) Resnet on CIFAR100 dataset; 8) A Dueling DQN (DDQN) agent in the CartPole-v0 environment; 9) An Advantage Actor Critic (A2C) agent in the Reacher-v2 environment; and 10) An Advantage Actor Critic (A2C) agent in the InvertedPendulum-v2 environment. Full experimental details are listed in Appendix B. We report the validation error as the evaluation metric for consistency across datasets. We evaluate two different variants of our algorithm: BAPI-4 and BAPI-8, where the maximum horizon is set to 4 and 8 respectively. BAPI-8 was evaluated on two benchmarks (LR with MNIST and MLP with Olivetti) to demonstrate the effect of varying the maximum horizon on the performance. The uncertainty threshold τ is set to 2 for all experiments. The parameter p is set to 20% of the maximum number of epochs for all experiments except for CNN-SVHN, where it is set to 10% due to the high cost of each epoch. We select at most three data points from each learning curve.

Setting ϵ For experiments 1 to 7, ϵ is set to 0.01 (interpreted as at most 1% degradation in accuracy) except for CNN-SVHN, where it is set to 0.005 due the small variation in the validation error. In the case of a loss/reward function where ϵ cannot be easily set (e.g experiments 8 to 10), it is automatically set as the smallest degradation in the function value at t_{max} in the evaluated data: $\epsilon = min\{f(x, t_{max}) - f(x, t) \, \forall (x, t) \in D\}$. In general, ϵ is not required to be fixed. A strategy for updating it, that balances exploration and exploitation (e.g., a wider value and therefore earlier stopping in the beginning), can be set by the practitioner and interfaced with our code easily. *We provide additional results and discussion in* **Appendix B**.

Results and Discussion Figure 2 shows the results (best validation error as a function of wall-clock time) of all methods on four HPO tasks. We make the following observations. **1)** BAPI identifies better candidates with less total cost than

[2] github.com/RaulAstudillo06/BudgetedBO

[3] github.com/shalijiang/bo/tree/main/enbo

[4] github.com/automl/HpBandSter [5] github.com/ntienvu/BOIL

[6] github.com/belakaria/BAPI

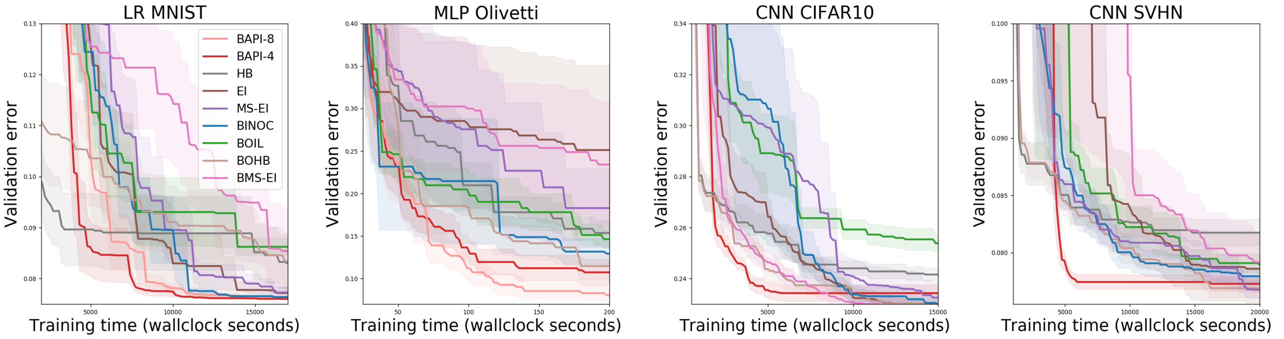


Figure 2: Results of validation error ± standard error for different baselines and our proposed approach on multiple iterative learners against training budget.

Table 1: Average ranking of BAPI and baseline methods across all experiments.

Algorithm	BAPI	HB	BOHB	EI	MS-EI	BINOC	BOIL	BMS-EI
Average ranking	2.9 ±0.51	6.1±0.51	3.6±0.67	4.9±0.53	3.4±0.50	4.3±0.38	6.3±0.63	4.2±0.73

the baselines due to its ability to plan selections while accounting for budget and early terminating non-promising candidates. **2)** A longer horizon for BAPI was tested on the relatively cheaper experiments LR MNIST and MLP Olivetti datasets and shows some performance degradation (LR MNIST) and some improvement (MLP Olivetti) suggesting that optimal horizon length is problem-dependent but clearly helpful in some cases. **3)** BMS-EI had an unstable performance and was not able to uncover good candidate in several experiments. We speculate that it is due to the approach being conservative about which points would satisfy the remaining horizon constraint. **4)** HB and BOHB can identify good candidates faster than most algorithms in the beginning, mainly because their strategy forces initial evaluations to be low-epoch trained runs. In the mid-range, their performance slows down, perhaps a consequence of their exploitation behavior and reliability on successive halving which might limit their extrapolation ability and stop promising candidates very early. Similar analysis has been reported in previous work (Dai et al., 2019). With longer search times, BOHB can catch up. However, both BOHB and HB performance degrades significantly in RL settings since successive halving cannot extrapolate accurately when the function might take a sigmoid or logit shape. **5)** BINOCULARS and MS-EI, are slower to uncover promising candidates due to spending more budget in evaluating all selected candidates to the maximum number of epochs. However, they both arrive at a competitive performance towards the end that can be attributed to their planning capabilities. **6)** BOIL is worse than most baselines across all benchmarks. As discussed in Section 1, BOIL selects the next candidates by weighting EI by the cost and might suffer from the cost miscalibration pathology. Similar observations were made in Astudillo et al. (2021).

Additional results included in Appendix B show that BAPI becomes less competitive when the total budget is significantly increased, but also provides results suggesting that performance loss can be brought back by adjusting the pruning behavior, a topic of on-going work.

We compared our approach to a wide range of baselines on 13 different experiments. All the baselines had inconsistent performance across the different experiments while our algorithm performed fairly well across all of them. The gain was significant in the case of limited budget, which is the desired behavior since a planning approach is more needed when the budget is limited. The gain was less significant in experiments with higher budgets but our approach was still competitive. Therefore, in Table 1, we provide the average ranking of each algorithm over all experiments based on their final performance.

6 SUMMARY

This paper considered the problem of hyperparameter optimization (HPO) for iterative learners under a constrained cost budget. The proposed BAPI approach addressed gaps in prior work including modeling of structured responses and mis-calibration between response and cost models leading to biased search. More importantly, our planning-based BAPI approach allows for non-myopic candidate selection over horizons adaptive to the budget. Combined with subset selection and early termination procedures, our experimental evaluation on a variety of HPO benchmarks shows BAPI's efficacy over previous methods in finding high-performing candidates with less cost budget.

References

Christian Agrell. Gaussian processes with linear operator inequality constraints. *JMLR*, 2019.

Setareh Ariafar, Zelda Mariet, Dana Brooks, Jennifer Dy, and Jasper Snoek. Faster & more reliable tuning of neural networks: Bayesian optimization with importance sampling. In *AISTATS*, pages 3961–3969. PMLR, 2021.

Raul Astudillo, Daniel Jiang, Maximilian Balandat, Eytan Bakshy, and Peter Frazier. Multi-step budgeted Bayesian optimization with unknown evaluation costs. *NeurIPS*, 34, 2021.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, pages 41–53. Citeseer, 2010.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization. In *Conference on Neural Information Processing Systems*, pages 7823–7833, 2019.

Zdravko I Botev. The normal law under linear restrictions: simulation and estimation via minimax tilting. *Journal of the Royal Statistical Society*, 79(1):125–148, 2017.

Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *ICML*. PMLR, 2019.

Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Bayesian optimization over hybrid spaces. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 2632–2643. PMLR, 2021.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.

Matthias Feurer, Aaron Klein, Jost Eggensperger, Katharina Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *NeurIPS*, pages 2962–2970, 2015.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47 (5):2410–2439, 2008.

Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic matrix factorization for automated machine learning. *NeurIPS*, 31:3348–3357, 2018.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *AISTATS*, pages 648–657. PMLR, 2016a.

Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In *AISTATS*. PMLR, 2016b.

Matthew Hoffman, Bobak Shahriari, and Nando Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *AISTATS*, pages 365–374. PMLR, 2014.

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

Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. BINOCULARS for efficient, nonmyopic sequential experimental design. In *ICML*. PMLR, 2020a.

Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. In *NeurIPS*, 2020b.

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

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *ICML*. PMLR, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *AISTATS*, pages 528–536. PMLR, 2017a.

Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with Bayesian neural networks. In *ICLR*, 2017b.

Jayesh H Kotecha and Petar M Djuric. Gibbs sampling approach for generation of truncated multivariate Gaussian random variables. In *ICASSP*, volume 3, pages 1757–1760. IEEE, 1999.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Remi R Lam, Karen E Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *NeurIPS*. Citeseer, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Erin LeDell and Sebastien Poirier. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, 2020.

Eric Lee, David Eriksson, David Bindel, Bolong Cheng, and Mike Mccourt. Efficient rollout strategies for Bayesian optimization. In *UAI*. PMLR, 2020a.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware Bayesian optimization. *arXiv preprint arXiv:2003.10870*, 2020b.

Eric Hans Lee, David Eriksson, Valerio Perrone, and Matthias Seeger. A nonmyopic approach to cost-constrained Bayesian optimization. In *UAI*, pages 568–577. PMLR, 2021.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(1):6765–6816, 2017.

Zhiyun Lu, Chao-Kai Chiang, and Fei Sha. Hyper-parameter tuning under a budget constraint. *arXiv preprint arXiv:1902.00532*, 2019.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*. Springer, 1975.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Vu Nguyen, Sebastian Schulze, and Michael Osborne. Bayesian optimization for iterative learning. *NeurIPS*, 33, 2020.

Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR, 2016.

Michael A Osborne. *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature*. PhD thesis, Oxford University, UK, 2010.

Jaakko Riihimäki and Aki Vehtari. Gaussian processes with monotonicity information. In *AISTATS*, pages 645–652. JMLR Workshop and Conference Proceedings, 2010.

Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *WACV*, pages 138–142. IEEE, 1994.

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, 2015.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *NeurIPS*, 25, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. *NeurIPS*, 26:2004–2012, 2013.

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

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, pages 847–855, 2013.

Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016a.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016b.

James T Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *NeurIPS*, 2018.

Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. *NeurIPS*, 32, 2019.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *UAI*. PMLR, 2020.

Xubo Yue and Raed AL Kontar. Why non-myopic Bayesian optimization is promising and how far should we look-ahead? a study via rollout. In *AISTATS*. PMLR, 2020.

Bayesian Optimization Over Iterative Learners with Structured Responses: A Budget-aware Planning Approach: Supplementary Materials

A Details of the monotonic Gaussian process

The posterior predictive distribution of the monotonic GP is $\mathbf{f}^*|Y, C$ which is the distribution of $\mathbf{f}^* = f(\mathbf{z}_*)$ for some new inputs $\mathbf{z}_* = [\mathbf{x}_*, t_*]$, conditioned on the observed data Y and the constraint C defined as $a(Z^v) \leq \mathcal{L}f(Z^v) \leq b(Z^v)$. The final derivation of the predictive distribution is defined as follow:

$$\mathbf{f}^*|Y, C \sim \mathcal{N}(\mu^* + A(\mathbf{C} - \mathcal{L}\mu^v) + B(Y - \mu), \Sigma) \tag{9}$$

$$\mathbf{C} = \widetilde{C}|Y, C \sim \mathcal{TN}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1, a(Z^v), b(Z^v)) \tag{10}$$

where $\mathcal{TN}(\cdot, \cdot, a, b)$ is the truncated Gaussian $\mathcal{N}(\cdot, \cdot)$ conditioned on the hyper-rectangle $[a_1, b_1] \times \cdots \times [a_k, b_k]$, $\mu^v = m(Z^v), \mu^* = m(\mathbf{z}_*), \mu = m(Z)$. The matricies A, B, A_1, B_1 and Σ are defined as follow:

$$A_1 = (\mathcal{L}K_{Z^v,X})(K_{Z,Z} + \sigma^2 I)^{-1} \tag{11}$$
$$A_2 = K_{\mathbf{z}_*,Z}(K_{Z,Z} + \sigma^2 I)^{-1} \tag{12}$$
$$B_1 = \mathcal{L}K_{Z^v,Z^v}\mathcal{L}^T + \sigma_v^2 I - A_1 K_{Z,Z^v}\mathcal{L}^T \tag{13}$$
$$B_2 = K_{\mathbf{z}_*,\mathbf{z}_*} - A_2 K_{Z,\mathbf{z}_*} \tag{14}$$
$$B_3 = K_{\mathbf{z}_*,Z^v}\mathcal{L}^T - A_2 K_{Z,Z^v}\mathcal{L}^T \tag{15}$$
$$A = B_3 B_1^{-1} \tag{16}$$
$$B = A_2 - AA_1 \tag{17}$$
$$\Sigma = B_2 - AB_3^T \tag{18}$$

Additionally, the probability that the unconstrained version of $\mathbf{C}$ falls within the constraint region, $p(C|Y)$, is defined as follow:

$$p(C|Y) = p\left(a(Z^v) \leq \mathcal{N}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1) \leq b(Z^v)\right) \tag{19}$$

and the unconstrained predictive distribution is

$$\mathbf{f}^*|Y \sim \mathcal{N}(\mu^* + A_2(Y - \mu), B_2).$$

Sampling from the posterior distribution with constraints has been a challenging task in previous work Riihimäki and Vehtari (2010). However, Agrell (2019) proposed to use a new method based on simulation via minimax tilting proposed by Botev (2017). This sampling approach was proposed for high-dimensional exact sampling and was shown to efficient and fast compared to previous approaches like rejection sampling and Gibb sampling Kotecha and Djuric (1999).

The specification of the location of virtual observations Z^v can have an important effect on the efficiency and scalability of the monotonic Gaussian process. Agrell (2019) proposed to have a suboptimization problem to find the optimal location. The idea is to iteratively place virtual observation locations where the probability that the constraint holds is low. However, this optimization becomes suboptimal when the dimension of the problem grows. Given that our function is monotonic with respect to only one dimension, we chose to define linearly spaced locations with respect to dimension t. The number of points is defined based on the kernel:

- ED kernel: The virtual observations locations will mainly enforce the direction of the monotonicity, therefore adding only two locations is sufficient.
- RBF Kernel: The number and location of virtual observations depends on the smoothness (lengthscale) of the kernel. The distance between every two virtual observations should be smaller than the lengthscale in order to maintain the monotonicity and avoid any fluctuations.

For more details about the the efficient posterior computation of the monotonic GP we refer the reader to Agrell (2019).

kernels derivatives

The computation of the posterior of the monotonic Gaussian process requires the definition of derivatives of the kernel function. In this work we consider monotonicity with respect to one dimension t. Therefore, kernel derivatives would be defined as follow

$$\frac{\partial}{\partial t}K([\mathbf{x},t],[\mathbf{x}',t']) = K_x(x,x') \times \frac{\partial}{\partial t}K_t(t,t') \tag{20}$$

$$\frac{\partial}{\partial t \partial t'}K([\mathbf{x},t],[\mathbf{x}',t']) = K_x(x,x') \times \frac{\partial}{\partial t \partial t'}K_t(t,t') \tag{21}$$

In this work t is a single dimensional variables. However, for sake of generality , we provide the kernel derivatives for the general case where t can be multi-dimensional. We define d_t as the t. In our experiments, we focus mainly on cases where the kernel over dimension t is an ED kernel. However, Our proposed method is not restrictive. In cases where the learning curve is not exponentially decaying, an RBF kernel with monotonicity over dimension t can be used. We provide the derivatives for both kernels

Exponential Decay Kernel

$$K_t(\mathbf{t},\mathbf{t}') = w + (\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha} \tag{22}$$

$$\frac{\partial}{\partial t'_j}K_t(\mathbf{t},\mathbf{t}') = -\frac{\alpha}{\beta_j}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-1} \tag{23}$$

$$\frac{\partial}{\partial t_j \partial t'_j}K_t(\mathbf{t},\mathbf{t}') = \frac{\alpha(\alpha+1)}{\beta_j^2}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-2} \tag{24}$$

$$\frac{\partial}{\partial t_i \partial t'_j}K_t(\mathbf{t},\mathbf{t}') = \frac{\alpha(\alpha+1)}{\beta_i \beta_j}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-2} \tag{25}$$

Radial basis function Kernel

$$K_t(\mathbf{t},\mathbf{t}') = exp(\frac{-1}{2}\sum_{i=1}^{d_t}\frac{(t_i - t'_i)^2}{l_i}) \tag{26}$$

$$\frac{\partial}{\partial t'_j}K_t(\mathbf{t},\mathbf{t}') = \frac{t_j - t'_j}{l_j^2}K_t(\mathbf{t},\mathbf{t}') \tag{27}$$

$$\frac{\partial}{\partial t_j \partial t'_j}K_t(\mathbf{t},\mathbf{t}') = \frac{1}{l_j^2}(1 - \frac{t_j - t'_j}{l_j^2})K_t(\mathbf{t},\mathbf{t}') \tag{28}$$

$$\frac{\partial}{\partial t_i \partial t'_j}K_t(\mathbf{t},\mathbf{t}') = -\frac{t_j - t'_j}{l_j^2}\frac{t_i - t'_i}{l_i^2}K_t(\mathbf{t},\mathbf{t}') \tag{29}$$

B Experimental Setup and Additional Results

B.1 Experimental Setup Details

Logistic Regression with MNIST:
We train the logistic regression classifier on the MNIST image dataset LeCun et al. (1998). The dataset consists of 70,000 images categorized into 10 classes. We use 80% for training and 10% for validation. We optimize the model over three

hyperparameter: the learning rate $\in [10^{-6}, 1]$, the L_2 regularization $\in [0, 1]$ and the batch size $\in [20, 2000]$. We apply a log transformation to the learning rate and batch size. We set the maximum number of epochs to 100.

MLP with Olivetti and Covtype:
We train a multi-layer perceptron with two fully connected layers on the Olivetti dataset Samaria and Harter (1994) and Covtype dataset. We use 10% of the data for the validation set. We optimize four hyperparameters learning rate $\in [10^{-6}, 1]$, batch size $\in [8, 128]$ for Olivetti and $\in [32, 1024]$ for Covtype , the L_2 regularization $\in [10^{-7}, 10^{-3}]$ and the momentum $\in [0.1, 0.9]$. We apply a log transformation to the learning rate, the batch size and the L_2 regularization. We set the maximum number of epochs to 100. The experiment with Covtype dataset was run on Tesla V100 GPU machine and the experiment on Olivetti was run on a CPU machine with Intel(R) Core(TM) i9-7960X CPU 2.80GHz.

FCNET MNIST:
We train a fully connected network with on the MNIST dataset. We use 50,000 images for the training set and 10,000 images for the validation set. We optimize six hyperparameters learning rate $\in [10^{-6}, 0.1]$, batch size $\in [32, 1024]$, , the L_2 regularization $\in [10^{-7}, 10^{-3}]$, the momentum $\in [0.1, 0.9]$, the number of hidden layers $\in [1, 4]$ and the size of hidden layers $\in [100, 1000]$ We apply a log transformation to the learning rate and the batch size. We set evaluate all algorithms on two different setups where in figure 3 we report the results with the maximum number of epochs set to $t_{max} = 25$ and in figure 4 we report the results with the maximum number of epochs set to $t_{max} = 50$. The total wallclock time budget is extended accordingly. These experiments were run on Tesla V100 GPU machine.

CNN with CIFAR10 and SVHN:
We train a CNN model on two image datasets CIFAR10 (Krizhevsky et al., 2009) and the Street View House Numbers (SVHN) (Netzer et al., 2011). For CIFAR10 we use 40,000 image for the training set and 10,000 for the validations set. For SVHN 63,257 image for the training set and 10,000 for the validations set. We optimize six hyperparameters: the batch size$\in [32, 1024]$, the learning rate$\in [10^{-6}, 0.1]$, the momentum$\in [0.1, 0.9]$, the L_2 regularization$\in [10^{-7}, 10^{-3}]$, the number of convolutional filters$\in [32, 256]$, and the number of dense units $\in [64, 512]$. We apply a log transformation to the learning rate, the batch size. We set evaluate all algorithms on two different setups where in figure 2 we report the results with the maximum number of epochs set to $t_{max} = 25$ and in figure 4 we report the results with the maximum number of epochs set to $t_{max} = 50$. The total wallclock time budget is extended accordingly. These experiments were run on Tesla P100 GPU machine.

Resnet with CIFAR100:
We train a ResNet model on a the image dataset CIFAR100 (Krizhevsky et al., 2009). We employ 40,000 images for the training set and 10,000 for the validations set. We optimize six hyperparameters: the batch size $\in [32, 512]$, the learning rate $\in [1e-6, 1e-1]$, the momentum $\in [0.1, 0.9]$, the L_2 regularization$\in [1e-7, 1e-3]$, the number of convolutional filters$\in [32, 256]$, and the number of layers $\in [10, 18]$. We report the results with the maximum number of epochs set to $t_{max} = 100$ in Figure 3. The total wall-clock time budget is extended accordingly. These experiments were run on a Tesla V100 GPU machine.

DQN CartPole:
We train a Dueling DQN (DDQN) (Wang et al., 2016b) agent in the CartPole-v0 environment. We employ the same setting proposed by Nguyen et al. (2020). We optimize two hyperparameters: the discount factor $\in [0.8, 1]$ and the learning rate for the model $\in [1e-6, 0.01]$. We vary the number of episodes from 200 to 500. We map the episodes into epochs with each three episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

A2C Reacher:
We train a Advantage Actor Critic (A2C) (Mnih et al., 2016) agent in the Reacher-v2 environment. We employ the same setting proposed by Nguyen et al. (2020). We optimize three hyperparameters: the discount factor $\in [0.8, 1]$, the learning rate for the actor $\in [1e-6, 0.01]$, and the learning rate for the critic $\in [1e-6, 0.01]$. We vary the number of episodes from 200 to 500. We map the episodes into epochs with each three episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

A2C Inverted Pendulum:
We train a Advantage Actor Critic (A2C) (Mnih et al., 2016) agent in the InvertedPendulum-v2 environment. We employ the same setting proposed by Nguyen et al. (2020). We optimize three hyperparameters: the discount factor $\in [0.8, 1]$, the learning rate for the actor $\in [1e-6, 0.01]$, and the learning rate for the critic $\in [1e-6, 0.01]$. We vary the number of

episodes from 700 to 1500. We map the episodes into epochs with each eight episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

B.2 Additional Results and Discussion

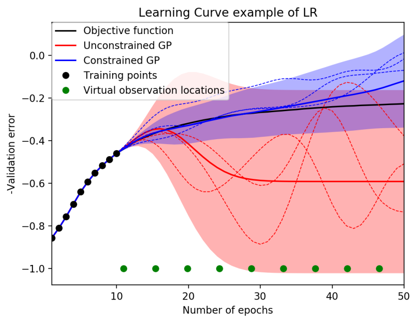
We report additional results of our BAPI approach and existing baselines. We report an additional variant of our algorithm that we name BAPI-4-L. We test the case where we do not add additional points from each curve but rather use only the last epoch. We notice that this variant performs competitively and sometimes better than adding additional points to the curve. This opens a discussion about the utility of leveraging additional data points from each curve especially while using the monotonic GP. It is important to note that previous methods built for HPO frequently suggest using additional points. This includes approaches proposed in the papers by Nguyen et al. (2020); Dai et al. (2019), and Wu et al. (2020). We plan to work on investigating this problem further to develop a sound theoretical understanding of this phenomenon.

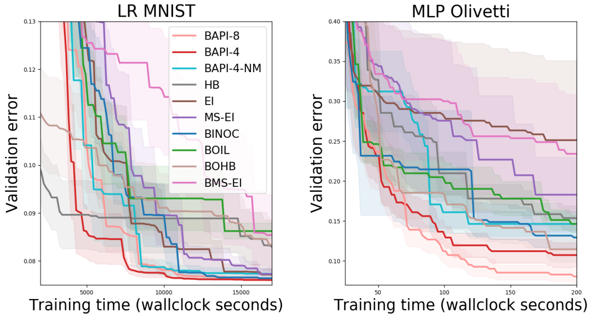
In Figure 4, we test all algorithms on settings with extended budget and a higher number of maximum epochs $t_{max} = 50$. We observe that given a sufficiently large budget, most of the baselines converge to statistically comparable results. We notice that HB, in most of the experiments, is able to reduce the validation error in the beginning but does not always converge to good results. However, BOHB performance was remarkably stronger with a higher number of maximum epochs. Increasing the maximum number of epoch enables BOHB to evaluate a larger number of configurations at a low budget and therefore we can see a significant drop in the validation error earlier than all baselines. The results in Figure 3 and Figure 4 show that BAPI-4 becomes less competitive when the total budget is significantly increased and the maximum number of epochs is higher, but also provides results suggesting that performance loss can be brought back by adjusting pruning behavior in BAPI-4-L.

We report additional results for reinforcement learning experiments optimized with RBF kernel over the number of epochs. Figure 4 shows the increasing discounted cumulative reward with discount factor 0.9 as suggested by Dai et al. (2019). The results show that BAPI-4 performs better or similar to the baselines. We observe that HB and BOHB performance degrades significantly with RL experiments most likely because they do not account for the possibility that the learning curve can be flat in the middle. We additionally notice that BAPI-4-L performance is competitive but worse than BAPI-4. One candidate reason for this behavior is due to the use of the RBF kernel, where adding intermediate points from the curve can be more crucial to avoid fluctuations.

B.3 Ablation: GP without enforced monotonicity

We provide an ablation study where we run our algorithm using a GP without enforced monotonicity to show the benefit of using a monotonic GP. The first figure illustrates differences in learning curve extrapolation between a **GP with enforced monotonicity and vanilla RBF GP**. The RBF GP fluctuates further from evaluated points rendering extrapolation highly uncertain (as well as inaccurate). Basing an estimation of optimal stopping time on this vanilla model directly affects budgeted HP optimization performance as shown in the **ablation study** displayed in the second and third figures. These show the performance of BAPI with a Non-Monotonic GP (BAPI-4-NM) is inferior to BAPI with monotonic GP. However, BAPI-4-NM shows competitive performance that might be associated with its budget-aware planning strategy.





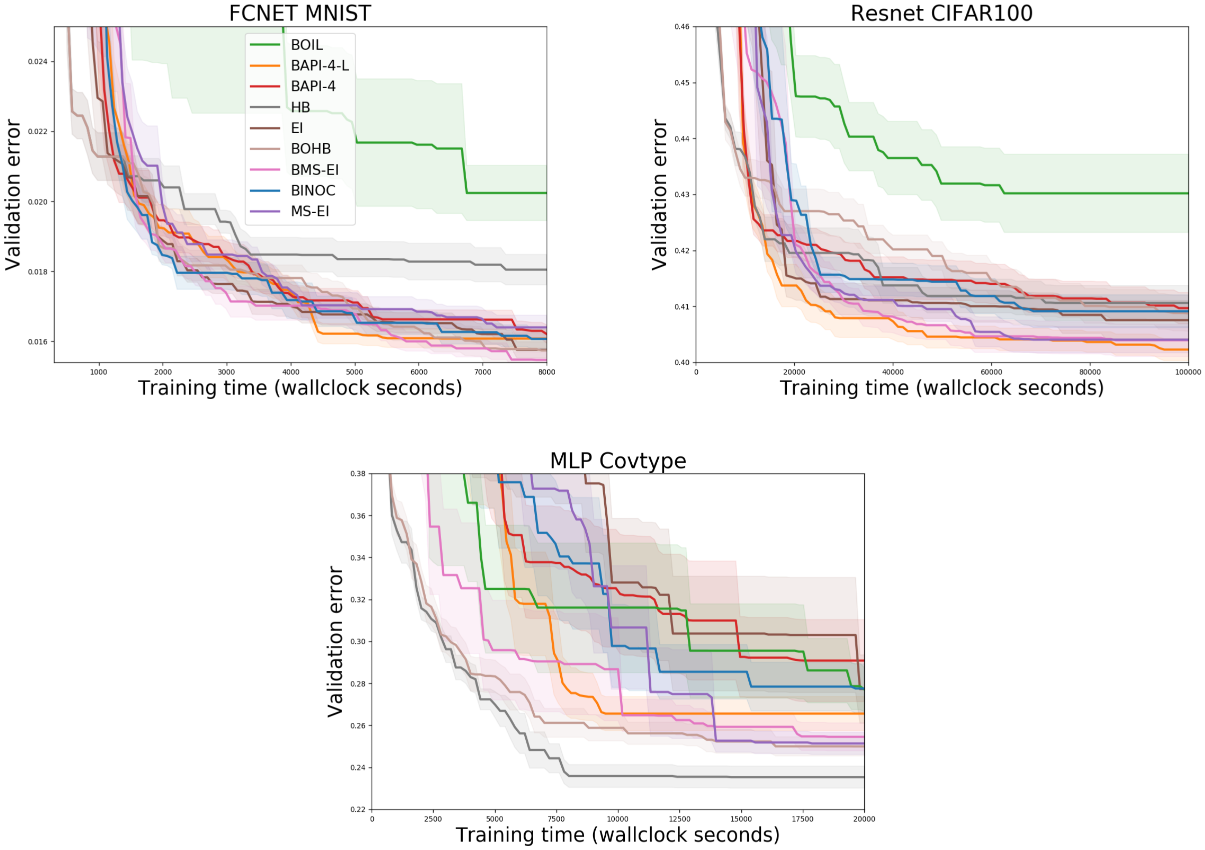


Figure 3: Results of validation error ± standard error for different baselines and our proposed approach on ResNet with $t_{max} = 100$, FCNET with $t_{max} = 25$ and MLP-Covtype with $t_{max} = 100$

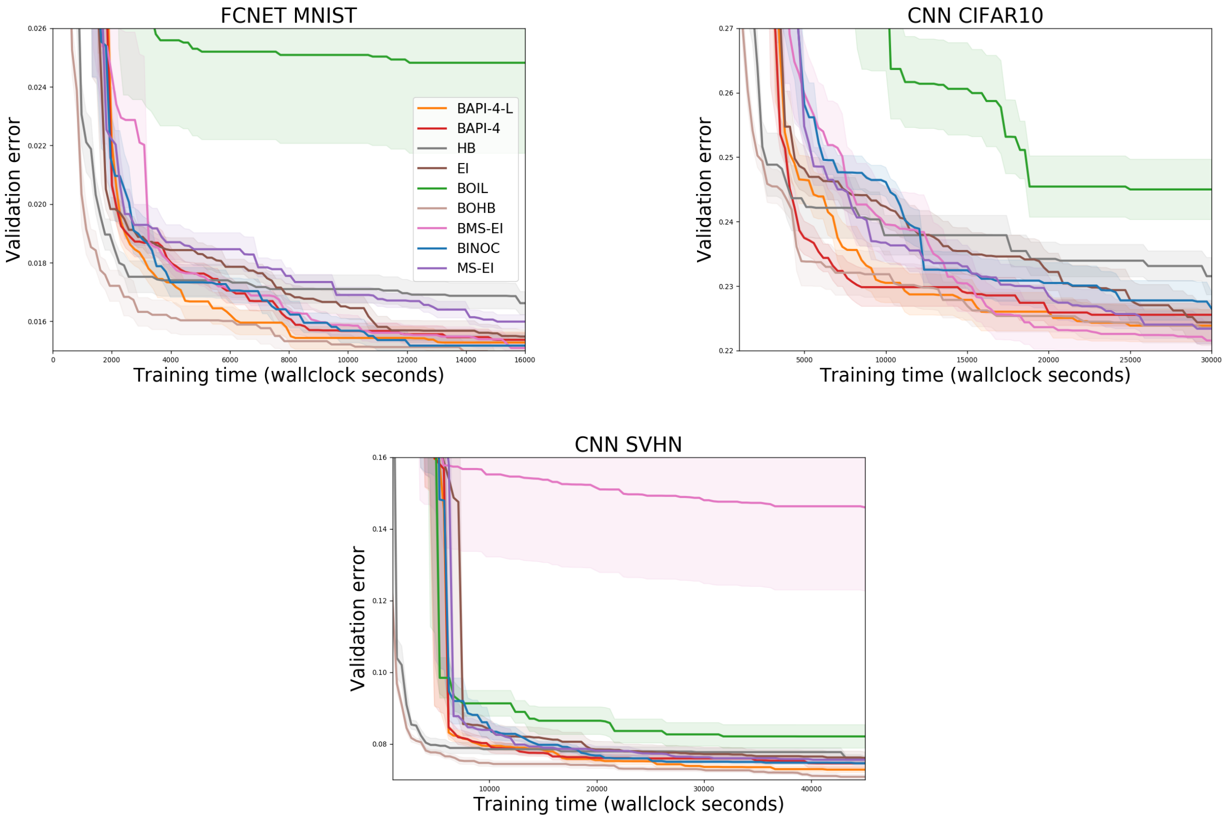


Figure 4: Results of validation error ± standard error for different baselines and our proposed approach on FCNET-MNIST, CNN-CIFAR10 and CNN-SVHN with $t_{max} = 50$

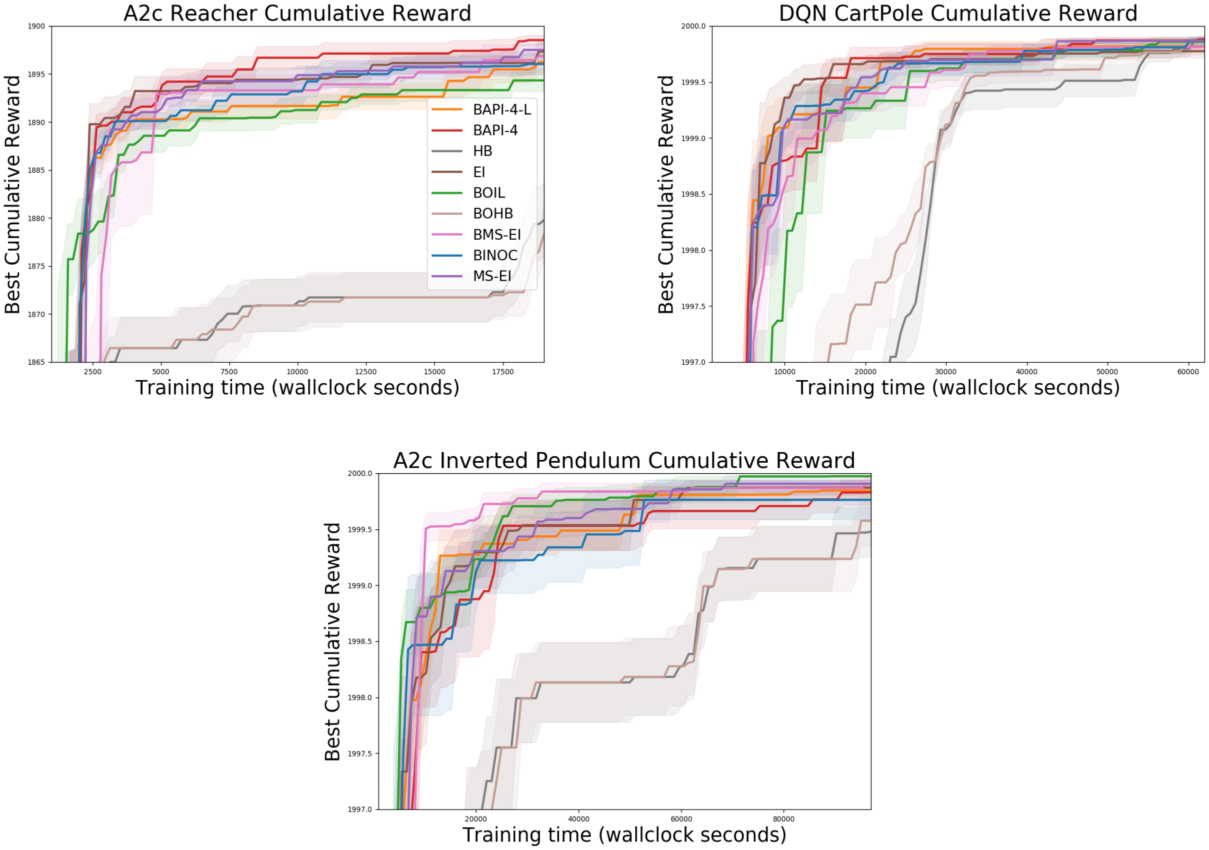


Figure 5: Results of Cumulative discounted reward ± standard error for different baselines and our proposed approach on A2C Reacher, DQN Cartpole, and A2C Inverted Pendulum