Efficient Hyperparameter Optimization with Adaptive Fidelity Identification

Jiantong Jiang[1], Zeyi Wen[2,3]*, Atif Mansoor[1], Ajmal Mian[1]
[1]The University of Western Australia, [2]HKUST (Guangzhou), [3]HKUST
jiantong.jiang@research.uwa.edu.au, wenzeyi@ust.hk, {atif.mansoor, ajmal.mian}@uwa.edu.au

Abstract

Hyperparameter Optimization and Neural Architecture Search are powerful in attaining state-of-the-art machine learning models, with Bayesian Optimization (BO) standing out as a mainstream method. Extending BO into the multi-fidelity setting has been an emerging research topic in this field, but faces the challenge of determining an appropriate fidelity for each hyperparameter configuration to fit the surrogate model. To tackle the challenge, we propose a multi-fidelity BO method named FastBO, which excels in adaptively deciding the fidelity for each configuration and providing strong performance while ensuring efficient resource usage. These advantages are achieved through our proposed techniques based on the concepts of efficient point and saturation point for each configuration, which can be obtained from the empirical learning curve of the configuration, estimated from early observations. Extensive experiments demonstrate FastBO's superior anytime performance and efficiency in identifying high-quality configurations and architectures. We also show that our method provides a way to extend any single-fidelity method to the multi-fidelity setting, highlighting the wide applicability of our approach.

1. Introduction

Hyperparameters are crucial in machine learning pipelines. Hyperparameter optimization (HPO) [11] and Neural Architecture Search (NAS) [9] aims to find the hyperparameters or architectures that can yield good performance without human experts. Among different HPO and NAS methods, Bayesian Optimization (BO) [2, 14, 40] is an effective model-based method that has shown remarkable success [8, 39]. BO maintains a *surrogate model* of the target performance metric based on past evaluations of hyperparameter configurations, which guides the choice of more promising configurations to evaluate.

Despite its sample efficiency, standard BO requires a full evaluation of each configuration, involving full-scale training and testing of models, which can be highly time-consuming, particularly with the recent trend to larger models. To avoid expensive full evaluations, multi-fidelity methods [4, 16, 25, 26] have been proposed, where the *fidelities* refer to the levels of performance metrics obtained under different resource levels. These methods follow the principle of successive halving (SHA) [16]: initially, they evaluate a set of randomly selected configurations using a small number of resources; then, based on the low-fidelity performances, the poorly-performing ones are successively eliminated, while the well-performing ones continue to be evaluated with increasing resources. Follow-up studies [10, 22, 27, 37, 47] propose model-based multi-fidelity methods, replacing the random configuration selection with a more informed model to improve sample efficiency.

Nevertheless, current model-based multi-fidelity methods face a major limitation: they are built upon the SHA framework, which operates under the assumption that learning curves of different configurations rarely intersect. This assumption does not hold in practice [46], i.e., early performance observations cannot always indicate the final fidelity performance at the full resource level. This leads to a fundamental challenge when extending model-based methods to the multi-fidelity setting: *What is the appropriate fidelity for each configuration to fit the surrogate model?* In other words, which fidelity can provide performance observations that reliably indicate the final fidelity performance? Existing methods struggle to address this fundamental challenge. In particular, BOHB [10] and Hyper-Tune [27] fit separate surrogate models for different fidelities, failing to capture inter-fidelity correlations. FTBO [44] and A-BOHB [22] fit a joint model but require strong assumptions to remain tractable. Another work by Salinas et al. [37] suggests using the last observed fidelity performance to fit the surrogate model. However, it widens the gap between poorly- and well-performing configurations at the early stage, potentially leading to an inaccurate surrogate model.

To this end, we propose a multi-fidelity extension of BO, namely FastBO, which tackles the challenge of deciding the appropriate fidelity for each configuration to fit the surrogate model. FastBO identifies a so-called *efficient point* for

*Zeyi Wen is the corresponding author.

each configuration to be the fidelity. The point balances computational cost and performance quality while capturing valuable learning curve trends. In essence, FastBO selects the fidelity for each configuration instead of evaluating all the configurations at the same fidelity. Additionally, a *saturation point* for each configuration is identified to be an approximation of the final fidelity, leading to high-quality performance while reducing resource wastage. The two crucial points are adaptively derived from the estimated learning curve of each configuration. Furthermore, the warm-up and post-processing stages are carefully designed to enable judicious early-termination detection and efficient saturation level evaluation. Empirical evaluation against the state-of-the-art methods shows that FastBO has strong anytime performance and can considerably save up to 87% of the time required to identify a good configuration or architecture, lowering the barriers for engaging in HPO and NAS. In summary, we make the following major contributions.

1. We propose a multi-fidelity model-based HPO method that adaptively decides the fidelities for configurations and efficiently offers strong performance, thanks to the introduced concepts of efficient and saturation points.
2. We develop a learning curve modeling module to enable adaptive derivation of the key points, a warm-up stage to allow early-termination detection, and a post-processing stage to ensure efficient saturation-level evaluation.
3. We show that our strategy can be used to extend existing single-fidelity methods to the multi-fidelity setting, demonstrating the effectiveness and generality of our method and highlighting promising future opportunities.

2. Related Work

HPO and NAS incur significant costs, especially considering the escalating model evaluation overhead. Despite efforts to directly accelerate computation [17–19, 52], smarter HPO strategies are still required for the widespread adoption of automation. Two crucial directions include model-based and multi-fidelity methods, which can also be combined. Here, we review the methods in these categories.

Model-based methods. BO is the representative of model-based methods. Based on the *surrogate model* constructed by historical evaluation results, BO selects the configurations to evaluate via an *acquisition function* that balances exploration and exploitation. Commonly used surrogate models are Gaussian processes [40], random forests [14], tree-structured Parzen estimator [2], and deep networks [41, 42]. Popular acquisition functions include Expected Improvement [28], Knowledge Gradient [12], Upper Confidence Bound [43], and Predictive Entropy Search [13]. Recent studies on BO have explored the utilization of expert priors [15, 24, 33, 38] and derivative information [1, 34, 49]. There also has been a recent focus on enhancing the interpretability [5, 50, 51] of the HPO process [3, 31, 32].

Multi-fidelity methods. Multi-fidelity methods exploit low and high fidelities for configurations to save the evaluation time. Successive halving (SHA) [16] runs a set of configurations using a small number of resources and promotes only the best-performing half of configurations to continue for twice as many resources. Hyperband [25] calls SHA as a sub-routine with varying maximum resources and introduces a reduction factor to control the fraction of promotion. ASHA [26] extends SHA to the asynchronous setting by aggressive early-stopping. Later, PASHA [4] further extends ASHA through more aggressive early-stopping based on the ranking of configurations during the tuning process.

Combination of model-based and multi-fidelity methods. BOHB [10] and a parallel work [47] first propose to combine model-based and multi-fidelity methods by replacing the random sampling in Hyperband with BO. A-BOHB [22] employs a joint GP surrogate over fidelities and supports asynchronous scheduling. Hyper-Tune [27] improves its Hyperband by a delayed strategy to decrease inaccurate promotions. Salinas et al. [37] proposed to extend methods to multi-fidelity settings by using the performance of the last fidelity in an ASHA running. DyHPO [48] and DPL [20] introduce new surrogates for multi-fidelity BO considering the learning curves; the former uses deep GP kernels while the latter integrates deep power law functions.

3. Problem Formulation

Given an algorithm having hyperparameters $\lambda_1, ..., \lambda_m$ with domains $\Lambda_1, ..., \Lambda_m$, we define its hyperparameter space as $\mathbf{\Lambda} = \Lambda_1 \times ... \times \Lambda_m$. Here, we define the problem and outline the key challenge related to hyperparameter optimization (HPO). Notations are in Supp. 7 for reference.

Single-fidelity setting. For each hyperparameter configuration $\boldsymbol{\lambda}$, we denote $f(\boldsymbol{\lambda})$ as the performance achieved using $\boldsymbol{\lambda}$. For consistency, the metric in this paper refers to descending metrics like validation loss, with ascending metrics being treated similarly. In the single-fidelity HPO setting, we aim to find $\boldsymbol{\lambda}^*$ minimizing function $f(\boldsymbol{\lambda})$, i.e., $\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} f(\boldsymbol{\lambda})$. BO is one of the most popular single-fidelity HPO methods. The vanilla BO has two key components: a *surrogate model* $\mathcal{M}$ to approximate the objective function $f(\boldsymbol{\lambda})$, and an *acquisition function* a to identify a promising configuration from search space. With these ingredients, BO iterates three steps: (i) select a configuration $\boldsymbol{\lambda}_i$ by maximizing the acquisition function; (ii) evaluate $\boldsymbol{\lambda}_i$ to get y_i and add the data $(\boldsymbol{\lambda}_i, y_i)$ into the current observation set $\mathcal{D}_{i-1} = \{(\boldsymbol{\lambda}_1, y_1), ..., (\boldsymbol{\lambda}_{i-1}, y_{i-1})\}$; (iii) update the surrogate model and the acquisition function based on the augmented $\mathcal{D}_i$. In this work, $\mathcal{M}$ is a Gaussian Process and a is Expected Improvement.

Multi-fidelity setting. Multi-fidelity methods consider resource information, such as training epochs or training subset ratios. Evaluations at various resource levels results in different performance levels, known as the *fidelities*. Different fidelities provide a way to balance computational cost and performance quality. In multi-fidelity HPO problems, the target is extended to $\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} f(\boldsymbol{\lambda}, r)$, where $f(\boldsymbol{\lambda}, r)$ is the objective function obtained for $\boldsymbol{\lambda}$ at r. We use r to denote the resource level, which can also be interpreted as the fidelity, and $r \in \{r_{min}, ..., r_{max}\}$.

Extending single-fidelity methods to the multi-fidelity setting. The inefficiency of single-fidelity methods stems from their reliance on the final fidelity evaluation of $f(\boldsymbol{\lambda}, r_{max})$ to be the evaluation of its objective $f(\boldsymbol{\lambda}, r)$. Fitting surrogate models by such final fidelity evaluations incurs high cost due to the full evaluation of the configurations. Notably, low-fidelity evaluations at $r < r_{max}$ provide informative insights into the objective but are computationally cheaper, which is valuable to the optimization process. Therefore, we seek an effective way to extend single-fidelity methods like BO to the multi-fidelity setting. More specifically, recalling the earlier steps of BO, when evaluating the configuration $\boldsymbol{\lambda}_i$ in the second step, we instead acquire its low-fidelity performance $y_i^{r_i}$ at r_i, where r_i denotes the fidelity used for $\boldsymbol{\lambda}_i$ to fit the surrogate model. The observations $\mathcal{D}_i$ then becomes $\{(\boldsymbol{\lambda}_1, y_1^{r_1}), ..., (\boldsymbol{\lambda}_i, y_i^{r_i})\}$. To conclude, in order to extend single-fidelity methods to the multi-fidelity setting, the key challenge to be addressed is to determine r_i for each $\boldsymbol{\lambda}_i$.

4. Methodology

In this section, we propose a novel multi-fidelity model-based algorithm FastBO. We first propose the key concepts of efficient point and saturation point, which are crucial in deciding the fidelity level to fit the surrogate model and to approximate the final fidelity respectively. Secondly, we elaborate on the details of learning curve modeling, where the two crucial points can be extracted. Then, we present the techniques associated with the auxiliary warm-up and post-processing stages. Finally, we summarize FastBO and discuss its wide applicability to any single-fidelity methods.

4.1. Estimation of Efficient and Saturation Points

In our method, we adaptively identify efficient and saturation points for each configuration. The two points are crucial in the optimization process. We first formally define the efficient point as follows.

Definition 1 (Efficient point)**.** *For a given learning curve $\mathcal{C}_i(r)$ of hyperparameter configuration $\boldsymbol{\lambda}_i$, where r represents the resource level (also referred to as fidelity), the efficient point e_i of $\boldsymbol{\lambda}_i$ is defined as: $e_i = \min\{r \mid \mathcal{C}_i(r) - \mathcal{C}_i(2r) < \delta_1\}$, where δ_1 is a predefined small threshold.*

The semantic of Definition 1 is that starting from the efficient point onwards, when the resources are doubled (i.e., from r to $2r$), the performance improvement falls below a small threshold δ_1. Consequently, this point characterizes the fidelity at which a configuration demonstrates strong performance while still efficiently utilizing resources. In simpler terms, it signifies an appropriate fidelity of performance that can be achieved with comparably efficient resource usage. Therefore, we make the following remark.

Remark 1. *The efficient points of the hyperparameter configurations can serve as their appropriate fidelities used for fitting the surrogate model. This is due to their (i) optimal resource-to-performance balance, (ii) ability to capture valuable learning curve trends, and (iii) customization for different hyperparameter configurations.*

We elaborate on the reasons in Remark 1 as follows. Firstly, efficient points balance the trade-off between computational cost and result quality. Beyond the efficient point of a given configuration, allocating additional resources to that configuration becomes less efficient. Secondly, efficient points capture valuable trends within the learning curves. For example, the learning rate influences the shape of learning curves; the identification of efficient points for configurations with smaller learning rates often occurs at later stages. The insights into learning curve behaviors enable more informed decision-making. Thirdly, the ability to customize the fidelity for each specific configuration is a significant advantage. This adaptive approach is more reasonable than previous studies that use a fixed fidelity for all configurations, as it better accounts for the unique characteristics of individual learning curves.

This insight leads us to use the efficient point e_i identified for each configuration $\boldsymbol{\lambda}_i$ as its fidelity used to fit the surrogate model. Specifically, we evaluate $\boldsymbol{\lambda}_i$ until reaching e_i and obtain the observed performance $y_i^{e_i}$. The resulting data point $(\boldsymbol{\lambda}_i, y_i^{e_i})$ is then added into the current observation set $\mathcal{D}_{i-1}$ to refit the surrogate model. We proof the superiority of FastBO over SHA-based methods in Supp. 8.

Besides efficient points, we identify saturation points for all configurations from their learning curves as well. We provide the definition of the saturation point as follows.

Definition 2 (Saturation point)**.** *For a given learning curve $\mathcal{C}_i(r)$ of configuration $\boldsymbol{\lambda}_i$, where r represents the resource level (also referred to as fidelity), the saturation point s_i of $\boldsymbol{\lambda}_i$ is defined as: $s_i = \min\{r \mid \forall r' > r, |\mathcal{C}_i(r') - \mathcal{C}_i(r)| < \delta_2\}$, where δ_2 is a predefined small threshold.*

The semantic of Definition 2 is that beyond the saturation point, the observed performance no longer exhibits notable variations with more resources. Thus, this point characterizes the fidelity at which the performance of a configuration stabilizes. The concept of saturation point is well-recognized within the machine learning community. Building on the above definition, we make the following remark.

Remark 2. *The saturation points of the hyperparameter configurations can serve as their approximate final fidelities, as they provide performance results that meet predefined quality thresholds while reducing resource wastage.*

This insight leads us to use the saturation point s_i identified for each configuration $\boldsymbol{\lambda}_i$ as its final fidelity approximation. The point is used in the post-processing stage for promoting some well-performing configurations to get higher-fidelity performances. In essence, when aiming for a full evaluation of the configurations, we suggest that terminating the evaluation at the saturation point is sufficient. A more intuitive illustration of the concepts of efficient and saturation points is provided in Supp. 9.

4.2. Learning Curve Modeling

From Definitions 1 and 2, we can extract the efficient and saturation points of configurations from their learning curves. The curve $\mathcal{C}_i(r)$ corresponds to configuration $\boldsymbol{\lambda}_i$ and describes the predictive performance with $\boldsymbol{\lambda}_i$ as a function of the fidelity r. Here, r can be either the number of training instances or the number of training epochs. In the context of learning curves, the former is referred to as observation learning curves, while the latter is iteration learning curves [29]. Both types are applicable to FastBO, so we use the term learning curve to encompass both. Given the observation set $\mathcal{O}_i^w = \{(r, y_i^r)\}_{r=r_{min},...,w}$ for $\boldsymbol{\lambda}_i$, which comprises pairs of data points representing fidelities $r \in \{r_{min}, ..., w\}$ and the corresponding evaluations y_i^r, where w is a pre-defined warm-up point to stop collecting data, FastBO can estimate a learning curve for $\boldsymbol{\lambda}_i$ based on $\mathcal{O}_i^w$ by first constructing a parametric learning curve model, then estimating the parameters.

Constructing a parametric learning curve model. Empirical learning curves can be modeled with function classes relying on some parameters. Viering and Loog [46] comprehensively summarized the parametric models studied in machine learning. In practice, different problems have different learning curves; even under the same problem, different hyperparameter configurations (*e.g.*, learning rate, regularization, etc.) may lead to significantly different learning curves. Since one single parametric model is not enough to characterize all the learning curves by itself, we consider combining different parametric models into a single model. Specifically, we consider three parametric models POW3, EXP3 and LOG2, as listed in Tab. 1, which have shown good fitting and predicting performance in previous empirical studies [29, 46]. We provide detailed discussions on the choice of parametric models in Supp. 10.

Table 1. Parametric learning curve models used.

Model	Formula	Family
POW3	$y = d + ax^{-\alpha}$	Power law
EXP3	$y = d + e^{-ax+b}$	Exponential
LOG2	$y = d + a\log(x)$	Logarithmic

Here, we denote each parametric model as $c_j(r|\boldsymbol{\theta}_j)$ with parameters $\boldsymbol{\theta}_j$, where the independent variable r represents the fidelity. We combine three models into one model via a weighted linear combination:

$$\mathcal{C}(r|\boldsymbol{\phi}) = \sum\nolimits_{j\in\{1,2,3\}} \omega_j c_j(r|\boldsymbol{\theta}_j), \tag{1}$$

where $\boldsymbol{\phi} = \{\omega_1, \omega_2, \omega_3, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$ is the parameter of the combined model, which consists of parameters $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$ and weight $\{\omega_1, \omega_2, \omega_3\}$ of every single model. Therefore, each pair of observations (r, y_i^r) in O_i^w can be modeled by the combined model as $y_i^r = \mathcal{C}(r|\boldsymbol{\phi}) + \epsilon$, where y_i^r is the observed dependent variable and ϵ represents the error term.

Estimating parameters in the parametric learning curve model. We employ maximum likelihood estimation to estimate the parameters $\boldsymbol{\phi}$ in the parametric model $\mathcal{C}(r|\boldsymbol{\phi})$. Assuming that $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the probability of an observed performance y_i^r under parameters is given by $p(y_i^r|\boldsymbol{\phi}, \sigma^2) = \mathcal{N}(y_i^r; \mathcal{C}(r|\boldsymbol{\phi}), \sigma^2)$. Given the observations $\mathcal{O}_i^w$ of $\boldsymbol{\lambda}_i$ that contains a set of observed data points (r, y_i^r), the likelihood function can be expressed as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}, \sigma^2; \mathbf{r}, \mathbf{y}_i^r) &= \prod p(y_i^k|\boldsymbol{\phi}, \sigma^2) \\ &= \prod_{k=r_{min}}^{w} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i^r - \mathcal{C}(r=k|\boldsymbol{\phi}))^2}{2\sigma^2}\right).\end{aligned} \tag{2}$$

We estimate $\boldsymbol{\phi}$ by maximizing log-likelihood function, which is easily calculated given Eq. 2.

An existing model-free method [7] also considers using learning curves for the HPO problem. However, it targets predicting the high-fidelity performance from the low-fidelity observations and thus stopping configurations that are unlikely to beat the current best values, which is different from our main target of identifying appropriate fidelity levels for the configurations to fit the surrogate model from their estimated learning curves.

4.3. Warm-up And Post-processing Stages

In addition to its core components, FastBO incorporates two auxiliary stages: the warm-up and post-processing stages. For the completeness of our method, we provide an overview of these stages, outlining their targets and presenting the key techniques of early-termination detection and saturation-level evaluation that are applied within.

Warm-up stage. The warm-up stage prepares the early observation set $\mathcal{O}_i^w$ for each configuration $\boldsymbol{\lambda}_i$ that is used to estimate its learning curve, as discussed in § 4.2. Here $w \in (r_{min}, r_{max})$ is a pre-determined fidelity, denoted as warm-up point. Specifically, we initiate the evaluation of each newly selected $\boldsymbol{\lambda}_i$, proceeding until reaching w. During this process, we record each fidelity r and its evaluation result y_i^r, forming pairs (r, y_i^r). Upon reaching w, we pause the evaluation for $\boldsymbol{\lambda}_i$ and obtain its early observation set $\mathcal{O}_i^w = \{(r, y_i^r)\}_{r=r_{min},\ldots,w}$, and start modeling the learning curve. During the warm-up stage, we monitor the performance changes across every two continuous fidelities. If we detect that the performance of $\boldsymbol{\lambda}_i$ has consecutively dropped twice by more than a ratio α, i.e., $(y_i^{r-1} - y_i^{r-2}) > \alpha y_i^{r-2}$ and $(y_i^r - y_i^{r-1}) > \alpha y_i^{r-1}$, we promptly terminate the evaluation for $\boldsymbol{\lambda}_i$ at its current fidelity r, because such consecutive performance deterioration indicates $\boldsymbol{\lambda}_i$ is unlikely to achieve satisfactory performance. Once terminated, we directly incorporate the current performance y_i^r of $\boldsymbol{\lambda}_i$ into $\mathcal{D}_{i-1}$ that is used for updating the surrogate model. Thus, further operations like learning curve modeling are discontinued for $\boldsymbol{\lambda}_i$. Moreover, if we observe a single case of performance drop without subsequent occurrences, i.e, $y_i^{r-1} - y_i^{r-2} > \alpha y_i^{r-2}$ and $y_i^r - y_i^{r-1} \leq \alpha y_i^{r-1}$, we opt not to include data from fidelity $r-1$ in $\mathcal{O}_i^w$. This is to manually filter out potential noise in the data that may adversely affect the fitting of the learning curve.

Post-processing stage. The post-processing stage aims at two tasks: promoting the well-performing configurations for saturation-level evaluations and identifying the best configuration and its performance. Firstly, FastBO promotes the top-k well-performing configurations and evaluates them to their saturation points to ensure high-quality performance while maintaining efficient resource utilization. We set k to be always less than or equal to the number of parallel workers available, ensuring a manageable overhead of saturation-level evaluations. It is worth noting that the additional time required is factored into the overall time. Secondly, FastBO finds the best configuration along with its performance achieved so far, which is a standard final step in most HPO methods. However, an increase in fidelities does not always result in performance improvement, possibly due to overfitting, resource saturation, or problem complexity. Therefore, we treat the evaluation at each fidelity as an individual task and record all these intermediate evaluation results, which is also a common practice in recent implementations. In this way, FastBO finds the best performance by considering all the results, rather than relying solely on the highest-fidelity performances of the configurations. In the parallel setting, treating each fidelity evaluation as an individual task offers an added benefit due to its finer granularity. More specifically, when a worker is idle, it takes on a new task of evaluating a configuration at a specific fidelity, rather than evaluating an entire configuration.

4.4. FastBO and Generalization

Algorithm 1 summarizes our proposed FastBO. It takes surrogate model $\mathcal{M}$, acquisition function a, warm-up point w, performance decrease ratio α, promotion number k, and thresholds δ_1, δ_2 as inputs, and output the best-founded configuration $\boldsymbol{\lambda}^*$ and its performance y^*. FastBO follows a similar iterative process of model-based methods but replaces the expensive full evaluations with a more intelligent alternative (*cf*. Lines 4-10). Specifically, each configuration $\boldsymbol{\lambda}_i$ first enters a warm-up stage to collect its early observation set $\mathcal{O}_i^w$ and to be detected and terminated if it exhibits consecutive performance deterioration (*cf*. Line 4). If $\boldsymbol{\lambda}_i$ is not terminated, FastBO then estimates a learning curve $\mathcal{C}_i(r)$ for $\boldsymbol{\lambda}_i$ based on $\mathcal{O}_i^w$ (*cf*. Line 6), and thus the efficient point and saturation point of $\boldsymbol{\lambda}_i$ can be obtained (*cf*. Line 7). After that, $\boldsymbol{\lambda}_i$ continues to be evaluated until reaching e_i (*cf*. Line 8); the result is added to the observation set $\mathcal{D}$ (*cf*. Line 11) that is used for updating $\mathcal{M}$ (*cf*. Line 12). On the other hand, the poorly-performing configuration will be terminated early at fidelity t with its result being added directly to $\mathcal{D}$ (*cf*. Lines 10, 11). Finally, the post-processing stage promotes the most promising configurations to their saturation points and finds the best-founded configuration $\boldsymbol{\lambda}^*$ and its performance y^* (*cf*. Line 14).

Algorithm 1: FastBO algorithm

```
input  : $\mathcal{M}$, $a$, $w$, $\alpha$, $k$, $\delta_1$, $\delta_2$.
output: $\boldsymbol{\lambda}^*$, $y^*$
1  $i \leftarrow 0$, $\mathcal{D} \leftarrow \emptyset$
2  while not meet the stop criterion do
3      find $\boldsymbol{\lambda}_i \leftarrow \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} a(\boldsymbol{\lambda}, \mathcal{M}_{i-1})$
4      $\mathcal{O}_i^w$, $t \leftarrow$ warm-up given $w$, $\alpha$    // cf. §4.3
5      if $\mathcal{O}_i^w$ is not empty then
6          fit $\mathcal{C}_i(r)$ to $\mathcal{O}_i^w$    // cf. §4.2
7          find $e_i$, $s_i$ given $\mathcal{C}_i(r)$, $\delta_1$, $\delta_2$    // cf. §4.1
8          $y_i^{e_i} \leftarrow$ continue evaluating $\boldsymbol{\lambda}_i$ to $e_i$
9      else
10         $e_i \leftarrow t$, $s_i \leftarrow r_{max}$
11     $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup (\boldsymbol{\lambda}_i, y_i^{e_i})$
12     refit $\mathcal{M}_i$ to $\mathcal{D}_i$
13     $i \leftarrow i + 1$
14 $\boldsymbol{\lambda}^*$, $y^* \leftarrow$ post-process given $\mathbf{s} = \{s_i\}$, $k$ // cf. §4.3
```

Generalizing FastBO to single-fidelity methods. The core of FastBO is to tackle the key challenge of deciding an appropriate fidelity for each configuration to fit the surrogate model by adaptively identifying its efficient point. This strategy of using the efficient point performances for surrogate model fitting also provides a simple but effective way to bridge the gap between single- and multi-fidelity methods. While it is primarily described in the context of model-based methods, the strategy can be generalized to various single-fidelity methods. For example, when evaluating configurations within the population for an evolution-

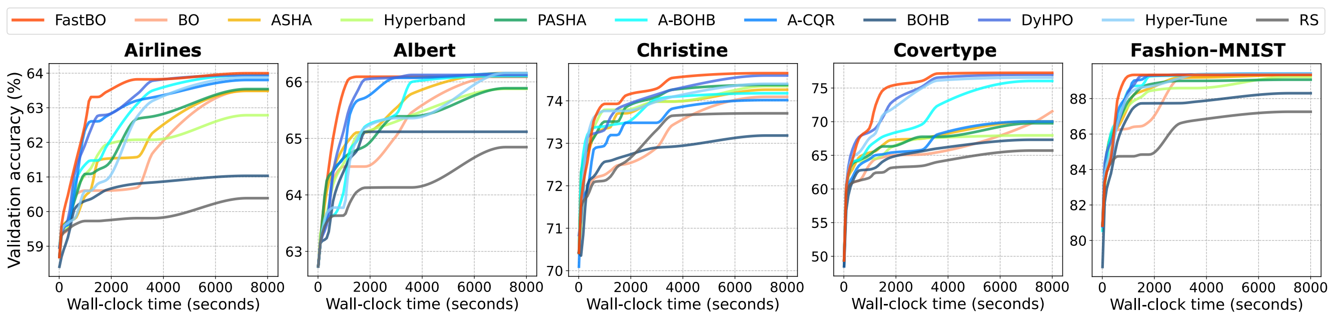
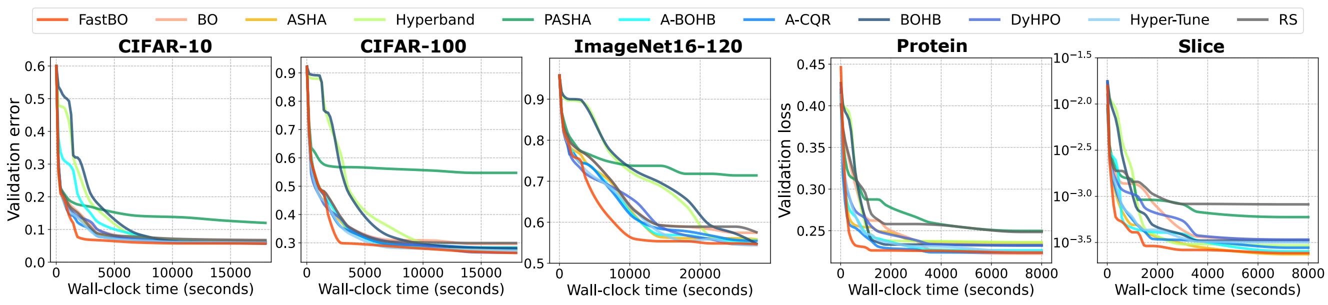


Figure 1. Performance of average validation accuracy on the LCBench benchmark.



(a) NAS-Bench-201 benchmark **(b) FCNet benchmark**

Figure 2. Performance of **(a)** average validation error on NAS-Bench-201 and **(b)** average validation loss on FCNet.

ary algorithm-based HPO method, we can similarly evaluate the efficient point performances instead of the final performances of these configurations and integrate the performances in the subsequent processes, such as selection and variation. Relying on the efficient point rather than the final fidelity or all fidelities available simplifies the extension of the single-fidelity methods to the multi-fidelity setting. The rationale behind this adaptive fidelity identification strategy is discussed in Remark 1. We also demonstrate in our experiments the efficacy of this strategy in extending a range of single-fidelity methods to the multi-fidelity setting.

5. Experiments

We empirically evaluate the performance of FastBO and compare it with the random search baseline (RS) and 9 competitive baselines from 3 related categories, including (i) model-based methods: standard Gaussian Process-based BO [40]; (ii) multi-fidelity methods: ASHA [26], Hyperband [25], PASHA [4]; and (iii) model-based multi-fidelity methods: A-BOHB [22], A-CQR [37], BOHB [10], DyHPO [48], Hyper-Tune [27]. RS and BO are single-fidelity baselines, while the others are multi-fidelity ones.

Our experiments are conducted on 10 datasets from 3 popular benchmarks LCBench [53], NAS-Bench-201 [8] and FCNet [21]. Detailed information on the benchmarks is provided in Supp. 13.1. All the experiments are evaluated with four parallel workers and 10 random seeds. We allocate 20% total budget for warm-up, i.e., $w = r_{min} + 0.2 \cdot (r_{max} - r_{min})$. Ratio α is set to 0.1; thresholds δ_1 and δ_2 are set to 0.001 and 0.0005 [1]. We set k based on the number of workers $\#w$ and the number of started configurations $\#c$: $k = \max\{\lceil \#c/10 \rceil, \#w\}$. We provide more experiments and discussions on the hyperparameters in Supp. 12. We use implementations of the baselines in Syne Tune [36]. Details of the baseline settings are in Supp. 13.2.

5.1. Anytime Performance

To evaluate the anytime performance, we compare FastBO against the baselines on wall-clock time. For fair comparisons, all the baselines, even single-fidelity BO and RS, are extended to consider intermediate results at all the fidelities when identifying the configuration, akin to FastBO as discussed in § 4.3. Consequently, all the baselines are able to achieve their best possible anytime performance.

The results on LCBench, NAS-Bench-201, and FCNet are shown in Figs. 1 and 2. We report the validation accuracy, error, and loss over wall-clock time for the three benchmarks, as provided by the benchmarks. We provide the results on NAS-Bench-301 [39] in Supp. 11.1. Overall, FastBO can handle various performance metrics and shows strong anytime performance. We can observe that FastBO gains an advantage earlier than other methods, rapidly con-

[1]Parameters δ_1 and δ_2 given here are derived after standardizing metrics to a uniform scale from 0 to 1.

Table 2. Comparison of relative efficiency on configuration identification. FastBO is set as the baseline with a relative efficiency of 1.00. Wall-clock time (abbr. WC time) reports the elapsed time spent for each method on finding configurations with similar performance metrics, i.e., validation error ($\times 10^{-2}$) for Covertype and ImageNet16-120 and validation loss ($\times 10^{-5}$) for Slice.

Dataset	Metric \ Method	**FastBO**	BO	PASHA	A-BOHB	A-CQR	BOHB	DyHPO	Hyper-Tune
Covertype	Val. error	**22.9**$_{\pm 0.2}$	23.0$_{\pm 0.3}$	25.1$_{\pm 2.5}$	23.5$_{\pm 1.1}$	31.6$_{\pm 1.9}$	32.5$_{\pm 0.8}$	23.0$_{\pm 0.3}$	23.0$_{\pm 0.2}$
	WC time (h)	**0.7**$_{\pm 0.3}$	2.9$_{\pm 0.7}$	3.9$_{\pm 1.0}$	2.0$_{\pm 1.0}$	3.9$_{\pm 0.2}$	2.5$_{\pm 1.0}$	1.7$_{\pm 0.6}$	1.8$_{\pm 0.7}$
	Rel. efficiency	**1.00**	0.25	0.18	0.37	0.19	0.29	0.41	0.40
ImageNet 16-120	Val. error	**55.3**$_{\pm 0.2}$	57.4$_{\pm 1.2}$	55.7$_{\pm 0.3}$	55.8$_{\pm 1.6}$	55.5$_{\pm 0.9}$	55.5$_{\pm 1.1}$	55.5$_{\pm 1.0}$	55.3$_{\pm 2.0}$
	WC time (h)	**2.2**$_{\pm 0.7}$	6.6$_{\pm 0.9}$	2.5$_{\pm 1.2}$	5.9$_{\pm 1.1}$	6.0$_{\pm 1.3}$	3.2$_{\pm 0.7}$	4.3$_{\pm 1.0}$	3.4$_{\pm 1.1}$
	Rel. efficiency	**1.00**	0.34	0.90	0.38	0.37	0.68	0.51	0.67
Slice	Val. loss	**26.3**$_{\pm 2.6}$	26.4$_{\pm 4.4}$	26.8$_{\pm 9.5}$	26.3$_{\pm 6.3}$	27.1$_{\pm 4.2}$	26.8$_{\pm 5.6}$	27.4$_{\pm 2.3}$	28.7$_{\pm 1.3}$
	WC time (h)	**0.4**$_{\pm 0.1}$	3.1$_{\pm 0.7}$	1.2$_{\pm 0.9}$	2.1$_{\pm 0.7}$	2.5$_{\pm 0.7}$	2.2$_{\pm 0.9}$	2.5$_{\pm 0.5}$	1.8$_{\pm 0.6}$
	Rel. efficiency	**1.00**	0.13	0.35	0.20	0.17	0.19	0.17	0.24

verging to the global optimum after the initial phase.

The superiority can be attributed to two main factors. Firstly, FastBO maintains, and in some cases even surpasses, the sample efficiency of vanilla BO, thanks to our techniques that enable quick and precise identification of the fidelities for configurations to update the surrogate model. We provide more explanations and experiments on sample efficiency in Supp. 11.2. Secondly, the multi-fidelity extension speeds up the evaluations, contributing to its overall efficiency. In contrast, the single-fidelity baselines tend to waste more time on the full evaluations. While the multi-fidelity baselines efficiently explore numerous configurations, they limit their evaluations to only constrained fidelities for some time, thus struggling to provide relatively high performance in a short time. This issue in multi-fidelity methods is particularly pronounced in PASHA when applied to NAS-Bench-201 and FCNet, as shown in Fig. 2. In Supp. 11.3, we further provide the ranks of all methods and statistically show FastBO's superiorty on an early stage. It is worth noting that all the additional overhead introduced by FastBO is taken into account in the wall-clock time.

Regarding the final performance, most methods are able to converge to satisfactory solutions, with negligible differences among them in most cases. Although our goal is not to offer the best final performance as we limit the evaluations to at most the saturation point even for those we consider most promising, FastBO still achieves top-2 final performance on 8 out of 10 datasets. In contrast, model-free methods sometimes cannot obtain a satisfactory final performance because they randomly select the configurations. For example, on the "Covertype" dataset, only 3 out of 2000 configurations yield a validation accuracy exceeding 75%. As a result, all the model-free methods face challenges in converging to a satisfactory final performance.

5.2. Efficiency on Configuration Identification

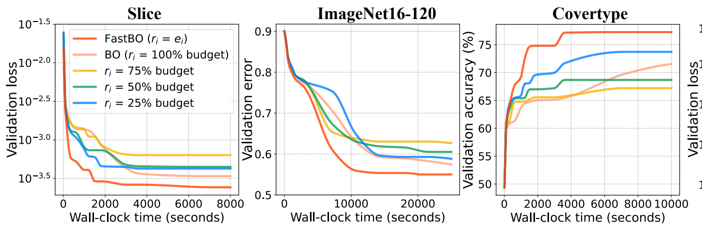
One explanation for PASHA's suboptimal anytime performance (*cf*. Fig. 2) lies in its primary goal [4]: the goal of PASHA is not high accuracy but to identify the best configuration more quickly. To ensure equitable comparisons, we report the time spent for each method on identifying a satisfactory configuration, consistent with the experiments described in PASHA [4]. Results on three expensive datasets "Covertype"[2], "ImageNet16-120", and "Slice" of the three benchmarks are shown in Tab. 2. Similar results on additional datasets can be found in Supp. 11.4. Besides PASHA, results of other model-free multi-fidelity methods are not included, as PASHA demonstrates its superiority over them.

Tab. 2 shows that FastBO saves 10% to 87% wall-clock time over other methods when achieving up to 9.6% better performance values. It can be observed from the "rel. efficiency" rows, where we set FastBO as the baseline with a relative efficiency of 1.00 and report the efficiency of other methods relative to ours. When compared with vanilla BO, FastBO significantly shortens the time in identifying a good configuration by a factor of 3 to 8, because FastBO pauses a configuration earlier at an appropriate fidelity and fits the surrogate model to guide the next configuration search. This advantage creates opportunities to efficiently explore more configurations. Another observation is that PASHA always gets a relatively high variance in wall-clock time. This is due to the fact that different random seeds can have a larger impact on such model-free methods.

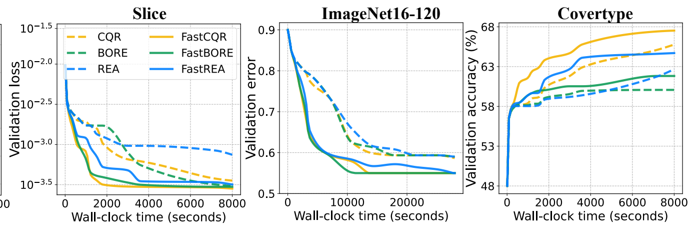
5.3. Effectiveness of Adaptive Fidelity Identification

As discussed in § 4.1, FastBO is able to adaptively identify the efficient point e_i for each configuration $\boldsymbol{\lambda}_i$ and serves

[2]We convert the accuracy of "Covertype" into error for readability.



(a) Impact of adaptive fidelity identification



(b) Generality of FastBO

Figure 3. Performance comparison: **(a)** Performance of FastBO that adaptively sets $r_i = e_i$ with the schemes that use fixed r_i for all configurations. **(b)** Performance of single-fidelity methods CQR, BORE, REA and their multi-fidelity variants using our extension method.

e_i as its fidelity r_i for surrogate model fitting. To investigate the effectiveness of the adaptive fidelity identification strategy, we conduct an ablation study to compare the performance achieved with and without applying this strategy. Specifically, we compare FastBO, where r_i is adaptively set to e_i, with the partial evaluation schemes that employ fixed predefined values as the fidelity for all the configurations to fit the surrogate model. We consider three representative fixed fidelities, including 25%, 50%, and 75% of the total resource budget. In addition, we include a comparison with vanilla BO that can be viewed as using 100% resource budget as the fixed fidelity for all configurations.

We provide the results on three representative datasets in Fig. 3(a), with more results available in Supp. 11.5. We have three main observations. Firstly, FastBO always outperforms the partial evaluation baselines that use a fixed fidelity, indicating the effectiveness of the adaptive strategy. Secondly, FastBO shows stronger performance than vanilla BO. The limitation of vanilla BO lies in the additional time required for full evaluations. Secondly, compared to the vanilla BO, partial evaluation schemes with fixed r_i converge faster in the initial stage due to their ability to evaluate more configurations promptly, but this advantage is gradually offset over time because they fail to find appropriate fidelities to create an accurate surrogate model. This causes a suboptimal final performance compared to vanilla BO, as shown in the first two figures in Figs. 3(a). In the case of the last one, we can observe a noticeable upward trend exhibited by the vanilla BO towards the end of the evaluation, indicating its potential to improve the final performance given abundant time. The comparison between the partial evaluation baselines and vanilla BO also demonstrates the importance of our adaptive strategy, which ensures that the fidelities align optimally with each configuration.

5.4. Generality of The Proposed Extension Method

The adaptive fidelity identification strategy provides a simple way to extend single-fidelity methods to the multi-fidelity setting, as discussed in § 4.4. To examine the ability of our extension method, we conduct experiments using three popular single-fidelity methods CQR [37], BORE [45] and REA [35], extending them to the multi-fidelity variants with our extension method, referred to as FastCQR, FastBORE, and FastREA respectively. Similar to FastBO, all the multi-fidelity extensions evaluate the configurations to the adaptively identified efficient point and use the corresponding performances for the subsequent operations. The results on three datasets are illustrated in Fig. 3(b) and similar results on other datasets are in Supp. 11.6. We can clearly observe that the multi-fidelity variants with our extension method always outperform their single-fidelity counterparts. It is worth noting that REA is an evolutionary algorithm-based HPO method and is also significantly improved by our extension. The observation highlights the ability of the proposed adaptive strategy to extend any single-fidelity method to the multi-fidelity setting. It also suggests future opportunities to extend other advanced single-fidelity techniques into the multi-fidelity setting.

6. Conclusion

In this paper, we propose a model-based multi-fidelity HPO method FastBO, which adaptively identifies the appropriate fidelity for each configuration to fit the surrogate model and offers high-quality performance while ensuring efficient resource utilization. The advantages are achieved through our concepts of efficient and saturation point, the proposed techniques of learning curve modeling, and well-designed warm-up and post-processing stages with judicious early-termination detection and efficient saturation-level evaluation. Moreover, the proposed adaptive fidelity identification strategy provides a simple way to extend any single-fidelity method to the multi-fidelity setting. Experiments demonstrate the effectiveness and wide generality of our proposed techniques. FastBO source code is freely available at https://github.com/jjiantong/FastBO.

Acknowledgment

This research was funded by ARC Grant number DP190102443. Ajmal Mian is the recipient of an Australian Research Council Future Fellowship Award (project number FT210100268) funded by the Australian Government.

References

[1] Sebastian E Ament and Carla P Gomes. Scalable first-order Bayesian Optimization via structured automatic differentiation. In *International Conference on Machine Learning*, pages 500–516. PMLR, 2022. 2

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011. 1, 2

[3] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484, 2023. 2

[4] Ondrej Bohdal, Lukas Balles, Martin Wistuba, Beyza Ermis, Cédric Archambeau, and Giovanni Zappella. PASHA: efficient HPO and NAS with progressive resource allocation. In *International Conference on Learning Representations*. OpenReview.net, 2023. 1, 2, 6, 7, 4, 9

[5] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 9

[7] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence*, 2015. 4

[8] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. 1, 6, 8

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 1

[10] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018. 1, 2, 6, 9

[11] Matthias Feurer and Frank Hutter. Hyperparameter optimization. *Automated Machine Learning: Methods, Systems, Challenges*, pages 3–33, 2019. 1

[12] 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. 2

[13] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing systems*, 27, 2014. 2

[14] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011. 1, 2

[15] Carl Hvarfner, Danny Stoll, Artur L. F. Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. πBO: Augmenting acquisition functions with user beliefs for bayesian optimization. In *International Conference on Learning Representations*. OpenReview.net, 2022. 2

[16] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248. PMLR, 2016. 1, 2, 9

[17] Kasra Jamshidi, Harry Xu, and Keval Vora. Accelerating graph mining systems with subgraph morphing. In *European Conference on Computer Systems*, pages 162–181, 2023. 2

[18] Jiantong Jiang, Zeyi Wen, Zeke Wang, Bingsheng He, and Jian Chen. Parallel and distributed structured svm training. *IEEE Transactions on Parallel and Distributed Systems*, 33 (5):1084–1096, 2021.

[19] Jiantong Jiang, Zeyi Wen, and Ajmal Mian. Fast parallel bayesian network structure learning. In *IEEE International Parallel and Distributed Processing Symposium*, pages 617–627. IEEE, 2022. 2

[20] Arlind Kadra, Maciej Janowski, Martin Wistuba, and Josif Grabocka. Scaling laws for hyperparameter optimization. In *Advances in Neural Information Processing Systems*, 2023. 2

[21] Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*, 2019. 6, 8

[22] Aaron Klein, Louis C Tiao, Thibaut Lienart, Cedric Archambeau, and Matthias Seeger. Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*, 2020. 1, 2, 6, 9

[23] Prasanth Kolachina, Nicola Cancedda, Marc Dymetman, and Sriram Venkatapathy. Prediction of learning curves in machine translation. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22–30, 2012. 2

[24] Cheng Li, Santu Rana, Sunil Gupta, Vu Nguyen, Svetha Venkatesh, Alessandra Sutti, David Rubin de Celis Leal, Teo Slezak, Murray Height, Mazher Mohammed, and Ian Gibson. Accelerating experimental design by incorporating experimenter hunches. In *International Conference on Data Mining*, pages 257–266. IEEE Computer Society, 2018. 2

[25] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017. 1, 2, 6, 9

[26] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020. 1, 2, 6, 9

[27] Yang Li, Yu Shen, Huaijun Jiang, Wentao Zhang, Jixiang Li, Ji Liu, Ce Zhang, and Bin Cui. Hyper-tune: Towards

efficient hyper-parameter tuning at scale. *Proceedings of the VLDB Endowment*, 15(6):1256–1265, 2022. 1, 2, 6, 9

[28] Jonas Mockus. The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 2:117, 1998. 2

[29] Felix Mohr and Jan N van Rijn. Learning curves for decision making in supervised machine learning–a survey. *arXiv preprint arXiv:2201.12150*, 2022. 4, 2

[30] Felix Mohr, Tom J Viering, Marco Loog, and Jan N van Rijn. Lcdb 1.0: An extensive learning curves database for classification tasks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 3–19. Springer, 2022. 2

[31] Julia Moosbauer, Julia Herbinger, Giuseppe Casalicchio, Marius Lindauer, and Bernd Bischl. Explaining hyperparameter optimization via partial dependence plots. *Advances in Neural Information Processing Systems*, 34:2280–2291, 2021. 2

[32] Julia Moosbauer, Giuseppe Casalicchio, Marius Lindauer, and Bernd Bischl. Improving accuracy of interpretability measures in hyperparameter optimization via Bayesian algorithm execution. *arXiv preprint arXiv:2206.05447*, 2022. 2

[33] ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, pages 3868–3877. PMLR, 2018. 2

[34] Misha Padidar, Xinran Zhu, Leo Huang, Jacob Gardner, and David Bindel. Scaling gaussian processes with derivative information using variational inference. *Advances in Neural Information Processing Systems*, 34:6442–6453, 2021. 2

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4780–4789, 2019. 8, 5, 9

[36] David Salinas, Matthias Seeger, Aaron Klein, Valerio Perrone, Martin Wistuba, and Cedric Archambeau. Syne tune: A library for large scale hyperparameter tuning and reproducible research. In *International Conference on Automated Machine Learning*, pages 16–1. PMLR, 2022. 6, 9

[37] David Salinas, Jacek Golebiowski, Aaron Klein, Matthias W. Seeger, and Cédric Archambeau. Optimizing hyperparameters with conformal quantile regression. In *International Conference on Machine Learning*, pages 29876–29893. PMLR, 2023. 1, 2, 6, 8, 5, 9

[38] Bobak Shahriari, Alexandre Bouchard-Côté, and Nando Freitas. Unbounded Bayesian Optimization via regularization. In *Artificial intelligence and statistics*, pages 1168–1176. PMLR, 2016. 2

[39] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *CoRR*, abs/2008.09777, 2020. 1, 6, 3

[40] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012. 1, 2, 6, 9

[41] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180. PMLR, 2015. 2

[42] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in Neural Information Processing Systems*, 29, 2016. 2

[43] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 2

[44] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. 1

[45] Louis C Tiao, Aaron Klein, Matthias W Seeger, Edwin V Bonilla, Cedric Archambeau, and Fabio Ramos. BORE: Bayesian optimization by density-ratio estimation. In *International Conference on Machine Learning*, pages 10289–10300. PMLR, 2021. 8, 5, 9

[46] Tom Viering and Marco Loog. The shape of learning curves: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 4, 2

[47] Jiazhuo Wang, Jason Xu, and Xuejun Wang. Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning. *arXiv preprint arXiv:1801.01596*, 2018. 1, 2

[48] Martin Wistuba, Arlind Kadra, and Josif Grabocka. Supervising the multi-fidelity race of hyperparameter configurations. *Advances in Neural Information Processing Systems*, 35:13470–13484, 2022. 2, 6, 9

[49] Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. *Advances in neural information processing systems*, 30, 2017. 2

[50] Peiyu Yang, Naveed Akhtar, Zeyi Wen, Mubarak Shah, and Ajmal Mian. Re-calibrating feature attributions for model interpretation. In *International Conference on Learning Representations*, 2022. 2

[51] Peiyu Yang, Naveed Akhtar, Zeyi Wen, and Ajmal Mian. Local path integration for attribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3173–3180, 2023. 2

[52] Xueying Zhu, Jie Sun, Zhenhao He, Jiantong Jiang, and Zeke Wang. Staleness-reduction mini-batch k-means. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 2

[53] Lucas Zimmer, Marius Thomas Lindauer, and Frank Hutter. Auto-Pytorch: Multi-fidelity metalearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:3079–3090, 2021. 6, 8

Efficient Hyperparameter Optimization with Adaptive Fidelity Identification

Supplementary Material

7. Notation

In Table 3, we provide a comprehensive summary of the notations utilized throughout the paper, along with their detailed definitions and explanations.

8. Proof of FastBO

HPO methods generally do not provide theoretical guarantees or rely on strong assumptions. In § 4.1, we provide formal definitions for the efficient point and propose to use the efficient point e_i of $\boldsymbol{\lambda}_i$ as its fidelity to fit the surrogate model. While it is challenging to prove FastBO's efficiency in reaching the optimal configuration, we provide a proof showing the superiority of FastBO over SHA-based methods (*e.g.*, Hyperband, ASHA, PASHA, BOHB, A-BOHB, A-CQR, Hyper-Tune). We show that fidelities in FastBO more reliably indicate final fidelity performance than those in SHA-based methods.

Proof. Given two learning curves $\mathcal{C}_1(r)$, $\mathcal{C}_2(r)$. $\mathcal{C}_2(r)$ descends more rapidly initially, while $\mathcal{C}_1(r)$ descends more slowly initially but finally converges to a lower loss, as shown in Figure 4. Let c be the crossing point.

SHA-based methods: they use a set of fixed fidelities $\{r\}$ for both $\mathcal{C}_1(r)$ and $\mathcal{C}_2(r)$. If $r \leq c$, then $\mathcal{C}_1(r) \geq \mathcal{C}_2(r)$, failing to indicate final performance.

FastBO: FastBO uses fidelities e_1 and e_2 for $\mathcal{C}_1(r)$, $\mathcal{C}_2(r)$. Clearly, $e_1 > e_2$, leading to $\mathcal{C}_2(e_1) < \mathcal{C}_2(e_2)$. In what follows, we discuss two cases.

Case 1: $e_1 \geq c$ (including $c \leq e_2 < e_1$ and $e_2 < c \leq e_1$): It follows that $\mathcal{C}_1(e_1) \leq \mathcal{C}_2(e_1)$. Thus, we have $\mathcal{C}_1(e_1) \leq \mathcal{C}_2(e_1) < \mathcal{C}_2(e_2)$. Then, $\mathcal{C}_1(e_1) < \mathcal{C}_2(e_2)$ holds true, aligning with the final performance.

Case 2: $e_2 < e_1 < c$: Based on Definition 1, $\mathcal{C}_1(e_1) - \mathcal{C}_1(2e_1) \approx \delta_1$, $\mathcal{C}_2(e_2) - \mathcal{C}_2(2e_2) \approx \delta_1$. Subtracting yields $\mathcal{C}_1(e_1) - \mathcal{C}_2(e_2) = \mathcal{C}_1(2e_1) - \mathcal{C}_2(2e_2) + \delta_1'$, where δ_1' is a small threshold around δ_1. As $2e_1 \geq c$ exists, it implies $\mathcal{C}_1(2e_1) < \mathcal{C}_2(2e_2)$ based on Case 1, so $\mathcal{C}_1(e_1) < \mathcal{C}_2(e_2)$.

Therefore, FastBO offers better fidelities that can more reliably indicate final fidelity performance, including scenarios even when $e_1, e_2 < c$. □

9. Illustration on Efficient Point and Saturation Point

In § 4.1, we provide formal definitions for the efficient point and saturation point. Here, we provide a more intuitive understanding of the concepts.

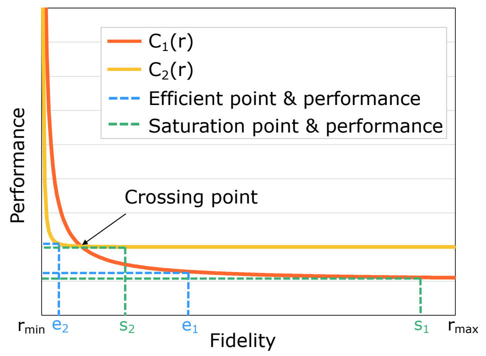


Figure 4. Illustration of efficient point and saturation point associated with learning curves.

Figure 4 shows an intuitive visualization of two learning curves $\mathcal{C}_1(r)$, $\mathcal{C}_2(r)$, together with their respective efficient points e_1, e_2 and saturation points s_1, s_2. We can easily grasp that the saturation points signify that the performance has nearly reached full convergence, while the efficient points, located at a relatively earlier stage, represent a position where performance can be achieved with high efficiency. From Figure 4, we can see a significant difference in the shapes of the two learning curves. $\mathcal{C}_2(r)$ experiences rapid initial descent and quick convergence; while $\mathcal{C}_1(r)$ experiences a slower initial descent, but eventually converges to a better performance than $\mathcal{C}_2(r)$. Due to this difference, we can find a crossing point where the two curves meet.

Suppose that $\mathcal{C}_1(r)$ and $\mathcal{C}_2(r)$ correspond to configurations $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ respectively, we can know $\boldsymbol{\lambda}_1$ outranks $\boldsymbol{\lambda}_2$ in terms of configuration performance ranking. Since FastBO utilizes efficient points as the fidelities for fitting the surrogate model, it is able to capture the distinctive trends in the learning curves. This ensures that the observed performance $y_1^{e_1}$ surpasses $y_2^{e_2}$, i.e., consistent with the configuration performance ranking. In contrast, existing successive halving-based methods may fail to maintain ranking consistency. Specifically, they are susceptible to erroneous termination of $\boldsymbol{\lambda}_1$ if the decision is made before the crossing point. Even with the aid of surrogate models, fitting before the crossing point leads to an inaccurate surrogate model.

Furthermore, we can observe that there is often a gap between the saturation point s_i and the final fidelity r_{max}, which becomes more pronounced on curves that converge rapidly, such as $\mathcal{C}_2$. FastBO utilizes the saturation point s_1 and s_2 as the approximation for the final fidelity r_{max}. Intuitively, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ can achieve performances $y_1^{s_1}$ and $y_2^{s_2}$ that are very close to their performances at the final fidelity while saving a considerable amount of computational cost.

Table 3. The notations used throughout the paper and the corresponding definitions.

Notation	Definition
a	Acquisition function.
$c_j(r\|\boldsymbol{\theta}_j)$, $\mathcal{C}(r\|\boldsymbol{\phi})$	One of, and the combined parametric learning curve model.
$\mathcal{C}_i(r)$	Empirical learning curve for $\boldsymbol{\lambda}_i$.
$\mathcal{D}_i$	Observation set that used to fit the surrogate model, containing i pairs of data points.
e_i	The efficient point of $\boldsymbol{\lambda}_i$.
$f(\boldsymbol{\lambda})$, $f(\boldsymbol{\lambda}, r)$	Performance with configuration $\boldsymbol{\lambda}$ in the single-fidelity and multi-fidelity settings.
k	The number of configurations to be promoted.
$\mathcal{M}$	Surrogate model.
$\mathcal{O}_i^w$	Early observation set of $\boldsymbol{\lambda}_i$ across different fidelities, with a maximum level w considered.
r, r_{max}, r_{min}	Fidelity; the maximum and minimum fidelity.
s_i	The saturation point of $\boldsymbol{\lambda}_i$.
w	Warm-up point for all the configurations.
y_i, y_i^r	Evaluation results of $f(\boldsymbol{\lambda}_i)$ and $f(\boldsymbol{\lambda}_i, r)$ in the single-fidelity and multi-fidelity settings.
y_{max}, y_{min}	Best and worse possible evaluation performance.
α	Performance decrease ratio.
δ_1, δ_2	Small thresholds used in identifying efficient points and saturation points.
$\boldsymbol{\theta}_j$, $\boldsymbol{\phi}$	Parameters in one of, and the combined parametric learning curve model.
λ_i, $\boldsymbol{\lambda}$	A hyperparameter and a hyperparameter configuration.
Λ_i, $\boldsymbol{\Lambda}$	Domain of λ_i and search space of $\boldsymbol{\lambda}$.
ω_j	The weight of a parametric learning curve model.

10. Discussion on Choice of Parametric Learning Curve Models

In § 4.2, we construct the parametric learning curve model by combining three parametric models POW3, EXP3 and LOG2. Here, we provide detailed discussions on the choice.

Overall, POW3, EXP3 and LOG2, especially POW3, have shown good fitting and predicting performance in previous empirical studies [29, 46]. In order to capture the diversity in learning curve shapes, we explore different families of parametric models, including the power law, exponential, and logarithmic families. However, parametric models from the sigmoidal family, like MMF and Weibull, are not being considered, since they tend to fit well if enough observations are used for fitting; but in situations like ours where observations are limited, their performance is suboptimal [30]. Moreover, existing studies have discussed the underfitting of the power law and exponential models with two parameters and the overfitting of those with four or more parameters [23]. Therefore, we opt for the POW3 and EXP3 (i.e., power law and exponential models with 3 parameters respectively).

Considering the goal of high efficiency in HPO, we simplify the choice of the parametric learning curve model to strike a balance between capturing general learning curve shapes and prioritizing computational efficiency. We avoid considering complex models, since the computational complexity of the subsequent parameter estimation is proportional to the number of parameters. The increase in the number of parameters translates to an increase in the time required for each hyperparameter configuration during the optimization process, which runs counter to the fundamental objective of designing efficient HPO algorithms.

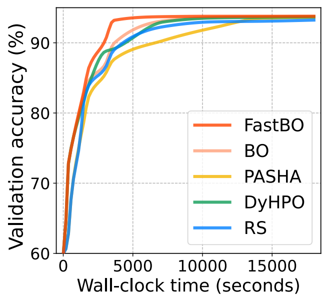


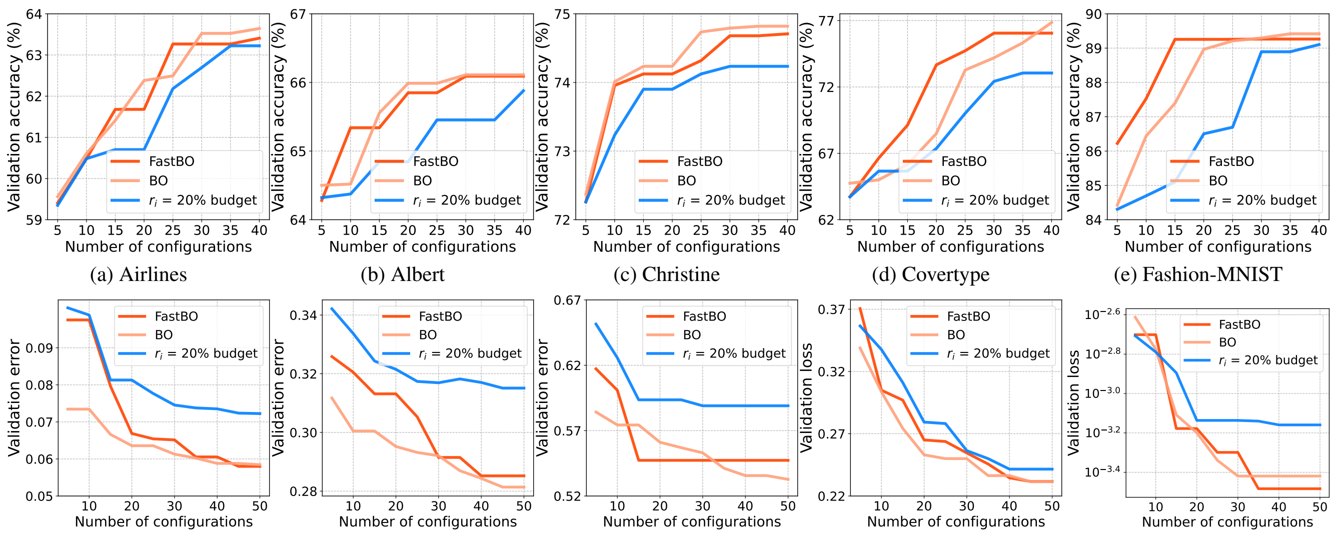
Figure 5. Performance of average validation accuracy on CIFAR-10 of the NAS-Bench-301 benchmark.

11. Extended Experiments

In this section, we provide additional experimental results and discussions.

11.1. Extended Experiments on NAS-Bench-301

Besides the comparison on the LCBench, NAS-Bench-201 and FCNet benchmark in § 5.1, we compare the anytime performance for the HPO methods on the NAS-Bench-301



(a) Airlines (b) Albert (c) Christine (d) Covertype (e) Fashion-MNIST

(f) CIFAR-10 (g) CIFAR-100 (h) ImageNet16-120 (i) Protein (j) Slice

Figure 6. Performance of **(a)-(e)**: average validation accuracy against the number of evaluated configurations on the LCBench benchmark. **(f)-(h)**: average validation error against the number of evaluated configurations on the NAS-Bench-201 benchmark and **(i)-(j)**: average validation loss against the number of evaluated configurations on the FCNet benchmark.

benchmark [39] that has up to 10^{21} architectures on the DARTS/FBNet search space. The results on the CIFAR-10 dataset are shown in Figure 5. We can observe that FastBO still shows strong anytime performance on NAS-Bench-301, demonstrating the scalability of FastBO on large search spaces.

11.2. Extended Experiments of Sample Efficiency

In § 5.1, we show the anytime performance of a wide range of HPO methods. One reason for FastBO's good anytime performance is its good sample efficiency. Sample efficiency refers to the ability of an algorithm to find the optimal solution with the minimum number of samples. In the context of HPO, sample efficiency quantifies how effectively the algorithm explores the hyperparameter space and identifies promising configurations while minimizing the number of evaluated configurations. Methods with higher sample efficiency, such as BO, are capable of identifying satisfactory configurations with fewer evaluations.

To investigate the sample efficiency of FastBO, we conduct experiments using the same settings as the experiments in § 5.1 but plotting the achieved performance as a function of the number of evaluated configurations. Figure 6 shows the results obtained on the three benchmarks. We can observe that FastBO is able to achieve comparable, and in some cases, even superior performance to vanilla BO. It is particularly noteworthy considering that FastBO only performs partial evaluations of the configurations and is unsure about their performance at the final fidelity. The results demonstrate that FastBO has the ability to identify the appropriate fidelity for each configuration that can reliably indicate its performance. This ability is achieved by our adaptive strategy that adaptively finds the efficient point for each configuration as its fidelity r_i for surrogate model fitting.

In order to facilitate a clearer comparison, we also incorporate the results on an additional baseline: a partial evaluation scheme that replaces the adaptive strategy with the adoption of a fixed value as the fidelity for all the configurations to fit the surrogate model. We set the fixed fidelity to 20% of the total resource budget and present the results in Figure 6. We can see that this partial evaluation baseline consistently lags behind both FastBO and vanilla BO. It underscores the challenge of using a fixed fidelity value for all configurations in reflecting their final fidelity performance, which highlights the importance of the adoption of our adaptive strategy.

11.3. Extended Experiments of Anytime Performance

In § 5.1, we compare the anytime performance for the HPO methods. Here, we present the critical difference diagrams to summarize the ranks of all methods and provide information on the statistical difference.

Due to the potential inconsistencies in performance metric differences among different datasets within the same benchmark, which may affect the critical difference diagram, we first employ normalized regret to standardize each evaluation result y across datasets. The normalized regret for each y is defined as $(y - y_{min})/(y_{max} - y_{min})$, where y_{max} and y_{min} represents the best and worse possible eval-

Table 4. Configuration Evaluation Unit (CEU) of each dataset.

Benchmark	Dataset	CEU(second)
LCBench	Airlines	1187
	Albert	1297
	Christine	1715
	Covertype	1942
	Fashion-MNIST	831
NAS-Bench-201	CIFAR-10	3879
	CIFAR-100	3879
	ImageNet16-120	11150
FCNet	Protein	303
	Slice	547

uation performance can be found. Moreover, since different datasets require varying time to evaluate a single configuration, it is not fair or meaningful to use the evaluation results at a fixed time for all the datasets for comparison. Considering the varying dataset workloads, we introduce one Configuration Evaluation Unit (CEU) as the average time required to perform a complete evaluation of a single configuration on a given dataset. The CEU of each dataset shown in Table 4 is easy to obtain for the tabular benchmark.

With these ingredients, we provide the critical difference diagrams of LCBench, NAS-Bench-201 and FCNet in Figure 7. The critical difference diagrams are based on Wilcoxon-Holm post-hoc analysis. The results correspond to the results at one CEU[3], which represents relatively early evaluated performances. We can observe that FastBO consistently outperforms the baseline methods on all the benchmarks at one CEU, showing its capacity for an early advantage gain during the optimization process.

From Figure 7, we observe that the model-based multi-fidelity HPO methods, including FastBO, A-BOHB, A-CQR, BOHB, DyHPO and Hyper-Tune, outperform the other methods in most cases, highlighting the promising direction of integrating model-based approaches with multi-fidelity techniques. Among them, DyHPO also considers the learning curves of hyperparameter configurations. Both FastBO and DyHPO are able to gain an advantage at a relatively early stage, indicating the significant value of learning curve information in addressing HPO problems. However, we observe that DyHPO exhibits inferior performance on the FCNet benchmark, suggesting a potential limitation in dealing with the validation loss metric.

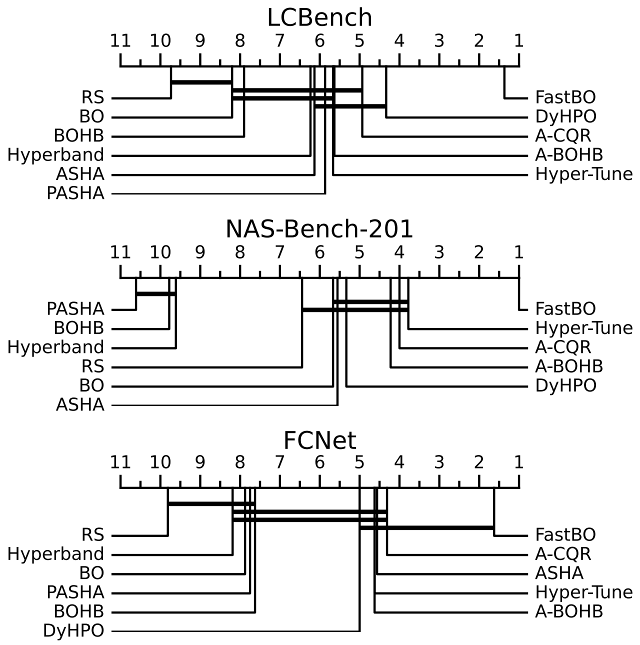


Figure 7. Critical difference diagram for LCBench, NAS-Bench-201 and FCNet at one CEU. The ranks indicate the sorted position in terms of normalized regret (the lower the better). Connected ranks indicate that differences are not statistically significant.

11.4. Extended Experiments of Efficiency on Configuration Identification

In § 5.2, we compare the time spent for the HPO methods on identifying a good configuration. Here, we report additional results on the datasets from the LCBench, NAS-Bench-201 and FCNet benchmarks in Table 5. We conduct experiments following the same settings as the experiments in § 5.2.

The experimental results shown in Table 5 are consistent with those shown in § 5.2. FastBO saves considerable wall-clock time over the baseline methods when achieving similar or better performance values, demonstrating the high efficiency of FastBO in identifying a good configuration. The model-free PASHA method often gets a high variance in wall-clock time because different random seeds can have a larger impact on it. Results of other model-free methods are not included in Table 5, since PASHA demonstrates its superiority over them [4].

11.5. Extended Experiments of Effectiveness of Adaptive Fidelity Identification

In § 5.3, we examine the effectiveness of the proposed adaptive fidelity identification strategy. Here, we provide additional results on more datasets.

We show the results on LCBench, NAS-Bench-201 and

[3]Note that the CEU is measured under one sequential worker, while FastBO and the baselines are evaluated under 4 parallel workers.

Table 5. Comparison of relative efficiency for configuration identification. Wall-clock time (abbr. WC time) reports the elapsed time spent for each method on finding configurations with similar performance metrics, i.e., validation error ($\times 10^{-2}$) and validation loss ($\times 10^{-2}$). Regarding relative efficiency, FastBO is set as the baseline with a relative efficiency of 1.00.

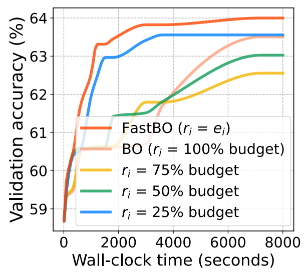
Dataset	Metric \ Method	**FastBO**	BO	PASHA	A-BOHB	A-CQR	BOHB	DyHPO	Hyper-Tune
Airlines	Val. error	**36.2**$_{\pm \mathbf{0.1}}$	36.3$_{\pm 0.5}$	**36.2**$_{\pm \mathbf{0.1}}$	36.3$_{\pm 0.3}$	38.9$_{\pm 0.5}$	38.5$_{\pm 0.1}$	36.3$_{\pm 0.1}$	**36.2**$_{\pm \mathbf{0.1}}$
	WC time (h)	**0.5**$_{\pm \mathbf{0.3}}$	2.4$_{\pm 1.3}$	1.1$_{\pm 0.7}$	1.1$_{\pm 0.6}$	2.7$_{\pm 0.6}$	2.2$_{\pm 0.4}$	1.3$_{\pm 0.3}$	1.1$_{\pm 0.6}$
	Rel. efficiency	**1.00**	0.23	0.51	0.50	0.20	0.25	0.38	0.48
Albert	Val. error	**33.9**$_{\pm \mathbf{0.1}}$	34.0$_{\pm 0.1}$	34.3$_{\pm 0.1}$	34.0$_{\pm 0.0}$	34.8$_{\pm 0.7}$	34.7$_{\pm 0.2}$	**33.9**$_{\pm \mathbf{0.2}}$	34.0$_{\pm 0.3}$
	WC time (h)	**0.5**$_{\pm \mathbf{0.3}}$	1.0$_{\pm 0.7}$	1.2$_{\pm 0.8}$	1.6$_{\pm 1.0}$	3.2$_{\pm 0.4}$	1.9$_{\pm 1.4}$	1.0$_{\pm 0.4}$	1.2$_{\pm 1.1}$
	Rel. efficiency	**1.00**	0.48	0.39	0.28	0.14	0.24	0.49	0.39
Christine	Val. error	**25.3**$_{\pm \mathbf{0.1}}$	25.5$_{\pm 0.1}$	25.6$_{\pm 0.1}$	25.5$_{\pm 0.1}$	26.7$_{\pm 0.0}$	26.8$_{\pm 0.2}$	25.5$_{\pm 0.1}$	25.4$_{\pm 0.0}$
	WC time (h)	**0.8**$_{\pm \mathbf{0.3}}$	2.4$_{\pm 1.3}$	2.4$_{\pm 2.2}$	2.1$_{\pm 1.2}$	1.6$_{\pm 2.1}$	1.5$_{\pm 0.9}$	1.6$_{\pm 0.6}$	2.9$_{\pm 0.8}$
	Rel. efficiency	**1.00**	0.33	0.33	0.37	0.48	0.54	0.47	0.27
Fashion-MNIST	Val. error	**10.7**$_{\pm \mathbf{0.1}}$	**10.7**$_{\pm \mathbf{0.1}}$	**10.7**$_{\pm \mathbf{0.1}}$	**10.7**$_{\pm \mathbf{0.1}}$	11.6$_{\pm 0.3}$	11.4$_{\pm 0.2}$	**10.7**$_{\pm \mathbf{0.1}}$	**10.7**$_{\pm \mathbf{0.1}}$
	WC time (h)	**0.2**$_{\pm \mathbf{0.1}}$	0.8$_{\pm 0.7}$	1.8$_{\pm 1.4}$	0.5$_{\pm 0.2}$	2.5$_{\pm 1.1}$	3.2$_{\pm 0.8}$	0.6$_{\pm 0.2}$	0.6$_{\pm 0.4}$
	Rel. efficiency	**1.00**	0.21	0.10	0.34	0.07	0.19	0.28	0.27
CIFAR-10	Val. error	**6.2**$_{\pm \mathbf{0.4}}$	6.5$_{\pm 0.4}$	6.4$_{\pm 0.7}$	**6.2**$_{\pm \mathbf{0.2}}$	6.3$_{\pm 0.4}$	6.3$_{\pm 0.2}$	6.3$_{\pm 0.4}$	**6.2**$_{\pm \mathbf{0.2}}$
	WC time (h)	**0.6**$_{\pm \mathbf{0.4}}$	3.9$_{\pm 2.0}$	1.3$_{\pm 0.6}$	2.3$_{\pm 1.1}$	2.6$_{\pm 0.9}$	2.1$_{\pm 0.5}$	2.5$_{\pm 0.8}$	1.6$_{\pm 0.8}$
	Rel. efficiency	**1.00**	0.16	0.49	0.27	0.25	0.31	0.26	0.39
CIFAR-100	Val. error	**28.7**$_{\pm \mathbf{1.3}}$	29.6$_{\pm 1.4}$	32.8$_{\pm 8.9}$	**28.7**$_{\pm \mathbf{1.2}}$	28.8$_{\pm 1.5}$	28.8$_{\pm 0.7}$	28.8$_{\pm 1.1}$	29.4$_{\pm 1.1}$
	WC time (h)	**1.2**$_{\pm \mathbf{0.9}}$	2.4$_{\pm 1.6}$	1.6$_{\pm 1.4}$	2.8$_{\pm 1.2}$	2.8$_{\pm 1.3}$	1.7$_{\pm 0.4}$	2.3$_{\pm 1.0}$	1.7$_{\pm 0.5}$
	Rel. efficiency	**1.00**	0.50	0.73	0.43	0.42	0.72	0.52	0.72
Protein	Val. loss	**22.6**$_{\pm \mathbf{0.4}}$	22.9$_{\pm 0.7}$	23.6$_{\pm 0.9}$	**22.6**$_{\pm \mathbf{0.3}}$	22.7$_{\pm 0.5}$	23.2$_{\pm 0.4}$	22.8$_{\pm 0.7}$	22.7$_{\pm 0.7}$
	WC time (h)	**0.3**$_{\pm \mathbf{0.1}}$	1.2$_{\pm 0.7}$	0.7$_{\pm 0.6}$	0.8$_{\pm 0.5}$	0.6$_{\pm 0.3}$	1.3$_{\pm 0.7}$	1.2$_{\pm 0.4}$	1.1$_{\pm 0.5}$
	Rel. efficiency	**1.00**	0.23	0.38	0.32	0.42	0.21	0.23	0.25

FCNet in Figure 8. FastBO with the adaptive fidelity identification strategy sets the efficient point e_i for each configuration $\boldsymbol{\lambda}_i$ as its fidelity r_i to fit the surrogate model. In contrast, the vanilla BO is a full evaluation scheme that uses 100% of the total resource budget as r_i. The other three baselines are also partial evaluation schemes like FastBO but they replace the adaptive choice of $r_i = e_i$ with a fixed fidelity, including 25%, 50%, and 75% of the total resource budget, for all the configurations to fit the surrogate model.

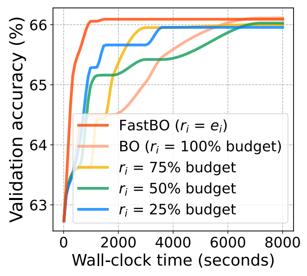
The results shown in Figures 8 are consistent with those shown in § 5.3. We have two main observations. Firstly, FastBO outperforms the other partial evaluation schemes that remove the adaptive fidelity identification strategy, showing the effectiveness of the proposed adaptive strategy. Secondly, although the partial evaluation schemes with fixed r_i are able to converge faster than the full evaluation counterpart (i.e., the vanilla BO) in the initial stage, this early advantage diminishes progressively over time. Finally, these partial evaluation baselines show significant differences in their final performance on 4 out of 7 datasets when compared to vanilla BO. The main reason is that these partial evaluation schemes naively use a fixed r_i for all the configurations and thus fail to create an accurate surrogate model to identify more promising configurations. This observation also highlights the importance of the adoption of our adaptive fidelity identification strategy.

11.6. Extended Experiments of Generality of The Proposed Extension Method

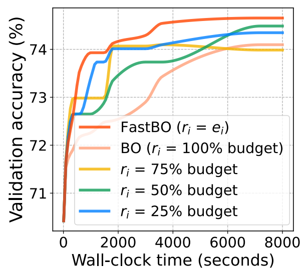
In § 5.4, we investigate the ability of our proposed extension method. Here, we provide additional results in Figure 9. We run three well-known single-fidelity methods CQR [37], BORE [45], and REA [35], and extend them to the multi-fidelity setting using our extension method, denoted as FastCQR, FastBORE, and FastREA respectively. More specifically, all the multi-fidelity variants evaluate the configurations to their efficient points and use the corresponding performances for the subsequent operations, i.e.,



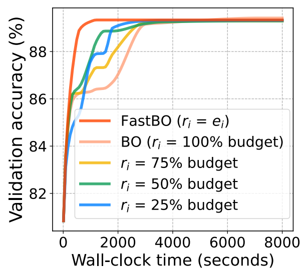
(a) Airlines



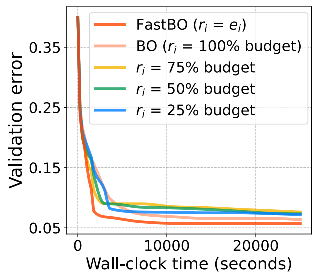
(b) Albert



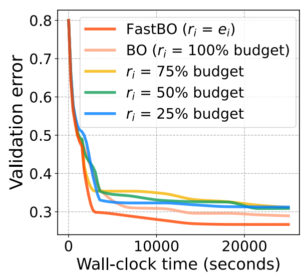
(c) Christine



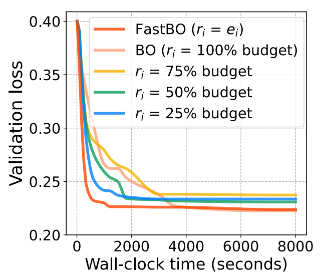
(d) Fashion-MNIST



(e) CIFAR-10

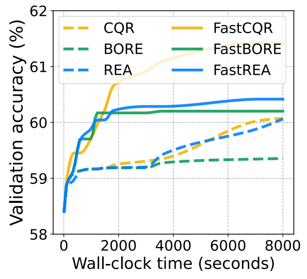


(f) CIFAR-100

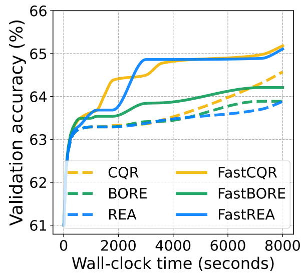


(g) Protein

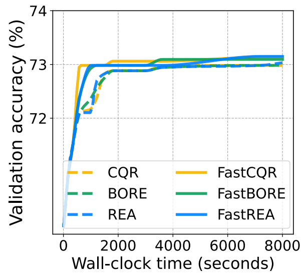
Figure 8. Average validation accuracy on the LCBench benchmark (**(a)-(d)**), average validation error on the NAS-Bench-201 benchmark (**(e)-(f)**), and average validation loss on the FCNet benchmark (**g**) of (i) FastBO that set $r_i = e_i$, (ii) the schemes that use fixed 25%, 50%, 75% of the total resource budget as r_i for all configurations, and (iii) vanilla BO that uses 100% total resource budget as r_i.



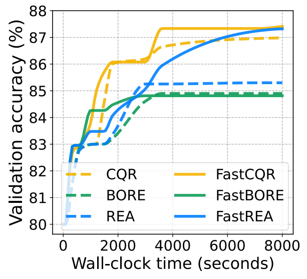
(a) Airlines



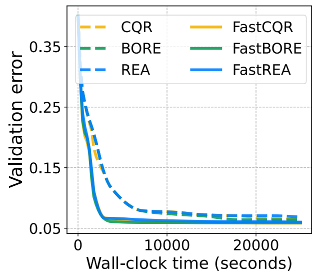
(b) Albert



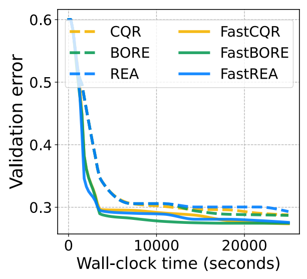
(c) Christine



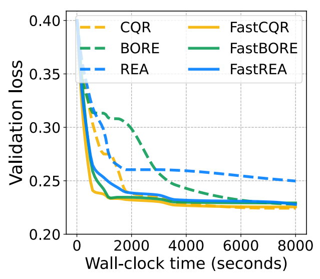
(d) Fashion-MNIST



(e) CIFAR-10



(f) CIFAR-100



(g) Protein

Figure 9. Performance of single-fidelity methods CQR, BORE, REA and their multi-fidelity variants FastCQR, FastBORE, FastREA using our extension method: average validation accuracy on the LCBench benchmark (**(a)-(d)**), average validation error on the NAS-Bench-201 benchmark (**(e)-(f)**), and average validation loss on the FCNet benchmark (**(g)**).

fitting the surrogate model for FastCQR and FastBORE, selection and variation for FastREA.

From Figures 9, we can clearly observe that the multi-fidelity variants with our extension method always outperform their single-fidelity counterparts. For the relatively simple task presented by the "Christine" dataset, the distinctions are not as pronounced as they are in the case of other datasets. However, it is still evident that the multi-fidelity methods are able to converge towards a higher accuracy more rapidly. Moreover, the evolutionary algorithm REA

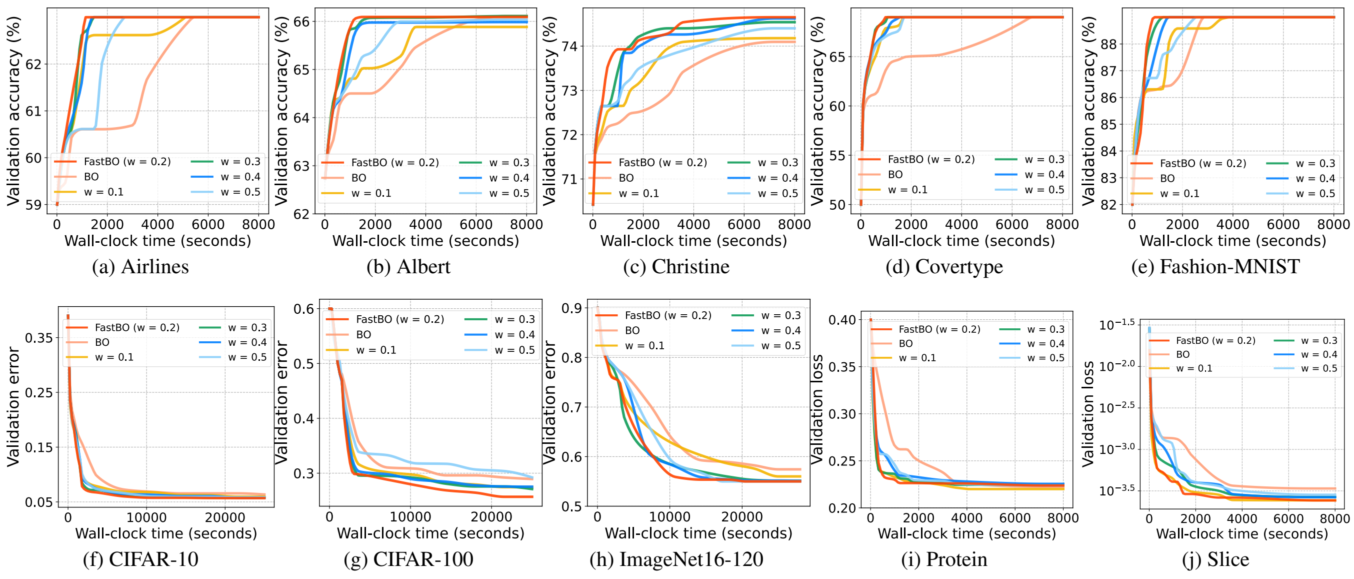


Figure 10. Performance of vanilla BO and the schemes with different w. The default setting of w in FastBO is 20% of the total resource budget (abbr. w = 0.2).

can also be enhanced by our extension method. The results are consistent with the observations shown in § 5.4 and highlight the wide applicability of the proposed adaptive strategy to extend any single-fidelity method to the multi-fidelity setting.

12. FastBO Hyperparameter Setting, Experiments, and Discussions

Here, we present the hyperparameter settings in FastBO, and provide experimental results and discussions on the hyperparameter settings.

12.1. Hyperparameter Setting

FastBO uses a Matérn $\frac{5}{2}$ kernel with automatic relevance determination parameters and the expected improvement acquisition function. We allocate 20% total resource budget for the warm-up stage, i.e., $w = r_{min} + 0.2 \cdot (r_{max} - r_{min})$. Ratio α is set to 0.1; thresholds δ_1 and δ_2 are set to 0.001 and 0.0005 [4]. We set k based on the number of parallel workers $\#workers$ and the number of started configurations $\#configurations$: $k = \max\{\lceil \#configurations/10 \rceil, \#workers\}$.

12.2. Experiments of Hyperparameter Setting

We compare the anytime performance of FastBO with different values of w, the warm-up point for all the configurations. We set w to 10%, 20%, 30%, 40%, and 50% of the total resource budget and examine their performances, where 20% one is the default setting of FastBO. In addition, we include a comparison with vanilla BO. In this section, we simply use $w = 0.1, ..., 0.5$ for abbreviation.

The results are shown in Figure 10. Overall, the default setting works quite well across different datasets. The results show that FastBO is not highly sensitive to the values of w, particularly within a reasonable range of 0.1 to 0.4, showing the robustness of our method.

Specifically, setting w to 0.2 and 0.3 always performs better on all the datasets. For $w = 0.5$, we can often observe a delayed performance improvement, as it requires more time to obtain additional evaluation observations for each configuration. Although this setting has the possibility of modeling more accurate learning curves, it wastes much time on expensive evaluations. The suitable values for w vary slightly across different benchmarks. For LCBench, the datasets have a relatively small maximum fidelity level of 50. Setting $w = 0.1$ cannot perform well, since there are only 5 observations for each configuration that can be used to fit its learning curve. While for NAS-Bench-201 and FCNet that have larger maximum fidelity levels, we can often see a delayed performance improvement when setting $w = 0.4$.

12.3. Discussion on Hyperparameter Setting

In order to avoid introducing extra efforts on tuning hyperparameters in FastBO, we intentionally set the hyperparameters in a simple way. We encourage the practitioners to directly use our default setting. Fine-tuning them is also a possibility and, if explored, may lead to further optimization on performance.

[4] Parameters δ_1 and δ_2 given here are derived after standardizing metrics to a uniform scale from 0 to 1.

Table 6. Detailed information of LCBench, NAS-Bench-201 and FCNet benchmarks.

Benchmark	#Evaluations	#Hyperparameters	#Fidelities
LCBench	2,000	7	50
NAS-Bench-201	15,625	6	200
FCNet	62,208	9	100

Table 7. Hyperparameters and configuration spaces for benchmarks.

Benchmark	Hyperparameter	Configuration space
LCBench	num_layers	[1, 5]
	max_units	[64, 512]
	batch_size	[16, 512]
	learning_rate	[1e-4, 1e-1]
	weight_decay	[1e-5, 0.1]
	momentum	[0.1, 0.99]
	max_dropout	[0.0, 1.0]
NAS-Bench-201	x0	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
	x1	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
	x2	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
	x4	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
	x3	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
	x5	[avg_pool_3x3, nor_conv_3x3, skip_connect, nor_conv_1x1, none]
FCNet	activation_1	[tanh, relu]
	activation_2	[tanh, relu]
	batch_size	[8, 16, 32, 64]
	dropout_1	[0.0, 0.3, 0.6]
	dropout_2	[0.0, 0.3, 0.6]
	init_lr	[0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]
	lr_schedule	[cosine, const]
	n_units_1	[16, 32, 64, 128, 256, 512]
	n_units_2	[16, 32, 64, 128, 256, 512]

13. Experimental Setup

Here we provide more details on the experimental setup, including details of the used benchmarks and choice of parameters on the baseline methods.

13.1. Benchmark Details

In our experiments, we use 3 well-known tabular benchmarks: LCBench [53], NAS-Bench-201 [8], and FCNet [21]. We conclude detailed information on these benchmarks in Tables 6, including the number of provided evaluations, the number of hyperparameters, and the number of fidelities. Table 7 provides information on the hyperparameters in the benchmarks and their corresponding configuration spaces.

LCBench. LCBench is a neural network benchmark that consists of 2000 hyperparameter configurations. LCBench features a search space of 7 numerical hyperparameters of neural networks, including the number of layers, the maximum number of units per layer, batch size, learning rate, weight decay, momentum, and dropout. The fidelity refers to the number of epochs in LCBench and each hyperparameter configuration is trained for 50 epochs. LCBench contains 35 datasets and we run the 5 most expensive ones.

NAS-Bench-201. NAS-Bench-201 is a benchmark that consists of 15625 hyperparameter configurations. NAS-Bench-201 features a search space of 6 categorical hyperparameters that correspond to 6 operations within the macro architecture cell. The fidelity refers to the number of epochs in NAS-Bench-201 and each hyperparameter configuration, which represents a network architecture, is trained for 200 epochs. NAS-Bench-201 contains the image classification datasets cifar-10, cifar-100 and ImageNet16-120.

FCNet. FCNet is a benchmark that consists of 62208 hyperparameter configurations. FCNet features a search space of 4 architectural choices (i.e., the number of units and activation functions for two layers) and 5 hyperparameters (i.e., dropout rates per layer, batch size, initial learning rate and learning rate schedule). The fidelity refers to the number of epochs in FCNet and each hyperparameter configuration is trained for 100 epochs. FCNet uses 4 popular UCI datasets for regression and we run the 2 most expensive ones.

13.2. Choice of Parameters on Baseline Methods

We use implementations of all the baseline HPO methods provided in Syne Tune [36]. We here list the parameters used for running the baselines in our experiments. In general, we follow the default settings in Syne Tune which are also recommended in the previous work.

- Vanilla Bayesian Optimization (BO) [40] uses a Matérn $\frac{5}{2}$ kernel with automatic relevance determination parameters and the expected improvement (EI) acquisition function.
- ASHA [26], Hyperband [25] and PASHA [4] follow the successive halving (SHA) [16] framework and sample new configurations at random. We use the reduction factor η of 3 in all of them. In other words, the evaluations are stopped after 1, 3, 9, 27, ... resource levels.
- A-BOHB [22] follows the SHA framework with $\eta = 3$. It uses a stopping variant asynchronous scheduling, which is different from the promotion variant asynchronous scheduling used in ASHA. New configurations are selected as in the vanilla BO.
- A-CQR [37] follows the SHA framework with $\eta = 3$ and uses the promotion variant asynchronous scheduling as ASHA. It uses BO to select the configuration and uses the last observed values from the SHA framework to fit the surrogate model. It uses a conformal quantile regression-based surrogate model.
- BOHB [10] follows the SHA framework with $\eta = 3$ and uses synchronous scheduling. It uses BO with a multivariate kernel density estimator (KDE) to select new hyperparameter configurations.
- DyHPO [48] uses the introduced deep kernel Gaussian Process surrogate and multi-fidelity EI. It uses an RBF kernel and the dense layers of the transformation function have 128 and 256 units. It uses a convolutional layer with a kernel size of three and four filters.
- Hyper-Tune [27] follows the SHA framework with $\eta = 3$ and uses the promotion variant asynchronous scheduling as ASHA. It fits independent Gaussian process models at different fidelities.

The experiments in § 5.4 and Supplementary Material 11.6 contain three HPO methods and we use implementations of them provided in Syne Tune. We also provide the parameter settings of the three methods as follows.

- CQR [37] uses BO with a conformal quantile regression-based surrogate model to select new configurations.
- BORE [45] is evaluated with XGBoost [6] as the classifier with its default setting. We set $\gamma = 1/4$, consistent with BORE's default hyperparameter setting.
- REA [35] is an evolutionary algorithm that uses a population size of 10, and 5 samples are drawn to select a mutation from.