UQ-Guided Hyperparameter Optimization

Jiesong Liu*, Jiawei Guan+, Feng Zhang+, Xipeng Shen*
* North Carolina State University
+ Renmin University

May 2024

NCSU TR-2024-2

Abstract

Hyperparameter Optimization (HPO) plays a pivotal role in unleashing the potential of machine learning models. This paper addresses a crucial aspect that has largely been overlooked in HPO: the impact of uncertainty in ML model training. The paper introduces the concept of *uncertainty-aware HPO* and presents a novel approach called the *UQ-guided scheme* for quantifying uncertainty. This scheme offers a principled and versatile method to empower HPO techniques in handling model uncertainty during their exploration of the candidate space. By constructing a probabilistic model and implementing probability-driven candidate selection and budget allocation, this approach enhances the quality of the resulting model hyperparameters. It achieves a notable performance improvement of over 50% in terms of accuracy regret and exploration time.

1 Introduction

Hyperparameter optimization (HPO) is essential for unleashing the power of machine learning models [17, 34, 4]. Hyperparameters include traditional parameters like learning rates and more complex ones like neural architectures and data augmentation policies. The main goal of HPO is to explore a vast candidate space to find candidates that lead to optimal model performance.

There are many designs in the literature to solve the HPO problem. Successive Halving (SH) [18], for example, terminates training of candidate configurations with poor performance early so as to save computing resources for more well-behaved candidates. Bayesian optimization [16, 29], another optimization method, uses a surrogate model to guide the selection of candidate configurations for assessment.

There is however a lack of systematic treatment to an important factor in HPO designs, the uncertainty inherent in the dynamics of the training process of machine learning applications. Because of the uncertainty, a model with a candidate hyperparameter configuration (or candidate in short) performing poorly in an early stage of its training could turn out to be the best model after convergence. Such candidates are however likely to be stopped from proceeding further or be completely discarded by existing HPO methods in the early stages, because their selections of candidates are mostly based on the currently observed performance, for lack of a way to treat the uncertainty properly. In 100 experiments of Successive Halving, for instance, the actually best candidates were discarded in the first 8–22 steps of the exploration, causing 48% performance regrets in validation loss (details in Section 3.1 Figure 2 and Section 4 Figure 4).

This paper introduces model uncertainty into the design of HPO methods and establishes the concept of *uncertainty-aware HPO*. At the core of uncertainty-aware HPO is a novel uncertainty quantization (UQ) guided scheme, named *UQ-guided scheme*, which unifies the selection of candidates and the scheduling of training budget—two most important operations in HPO—into a single UQ-based formal decision process. The UQ-guided scheme builds on a probabilistic uncertainty-based model, designed to approximate the statistical effects of discarding a set of candidates at the end of a step in HPO. It uses a lightweight method to efficiently quantify model uncertainty on the fly. It offers a principled, efficient way for HPO to treat model training uncertainty.

As a general scheme, the *UQ-guided scheme* can be integrated into a variety of HPO methods. This paper demonstrates its usefulness and generality by integrating it into four existing HPO methods. Experiments on two widely used HPO benchmarks, NAS-BENCH-201 [9] and LCBench [36], show that the enhanced methods produce models that have 21–55% regret reduction over the models from the original methods at the same exploration cost. And those enhanced methods need only 30–75% time to produce models with accuracy comparable to those by the original HPO methods. The paper further gives a theoretical analysis of the impact of the *UQ-guided scheme* for HPO.

2 Background and Related Work

Many studies are committed to solving the HPO problem efficiently [25, 19, 27, 33, 35]. Bayesian optimization, early stop-based mechanisms, and multi-fidelity optimizations are some important approaches.

Bayesian Optimization (BO). BO is a sequential design strategy used to optimize black-box functions [29, 16, 11]. In HPO scenarios, it can be used as a surrogate model to sample high-quality candidates.

Early Stop Mechanisms. Early stop-based approaches can be effective since they evaluate different candidates during training and make adaptive selections accordingly [30, 8, 2]. The early

stopping mechanism, which stops the training of poorly-performed candidates early, has been widely employed in the HPO community including Successive Halving (SH) [18] and Hyperband (HB) [24]; BOHB [11] combines both BO and HB methods to take advantage of both the BO surrogate model and the early stopping mechanism.

Multi-fidelity Optimizations. Multi-fidelity evaluation focuses on using low-fidelity results trained with small resources to accelerate the evaluation of candidates [30, 8, 2, 21, 22, 32, 3, 20, 31, 13, 25]. Sub-sampling (SS) [14] is proposed mainly using multi-fidelity methods to collect high-quality data to select good configurations without early stopping.

Model Uncertainty in HPO. Various optimization methods in HPO scenarios focus on specific training metrics to assess candidate performance. However, these methods typically overlook the uncertainty in the candidate selection process. Machine learning models inherently have approximation uncertainties [5, 12, 23, 10, 26, 6]. Some HPO designs sample the candidate space based on distributions on the effect of each hyperparameter dimension on the quality of the candidates, but without considering the uncertainty in the model training process. For example, one of the studies [28] separates candidates into "good" or "bad" groups in order to build the distributions. The separation is based on the same deterministic metrics as other HPO methods use, giving no consideration of the uncertainty in model trainings.

3 Uncertainty Quantification (UQ)-Guided Hyperparameter Optimization

This section gives an exploration of model uncertainty, introduces *UQ-guided scheme* for incorporating UQ into the design of HPO, discusses examples of ways to use the UQ-guided scheme to enhance existing HPO methods, and theoretically analyzes its effects.

3.1 Different Forms of Uncertainty

Uncertainty in machine learning originates mainly from two factors: inherent noise in the data and the variability of model predictions due to restricted knowledge [15, 1]. Since data uncertainty is constant, it is the variability in model predictions, referred to as model uncertainty, that primarily influences decisions on HPO.

Consider an observed value $y_o = f(\mathbf{x}) + \epsilon$ for a given $\mathbf{x} \in \mathbb{R}^d$, where ϵ follows a Gaussian distribution $\mathcal{N}(0, \sigma_1^2)$. This implies $y_o \sim \mathcal{N}(f(\mathbf{x}), \sigma_1^2)$, with $f(\mathbf{x})$ being the true target. The prediction distribution, denoted by $y_p \sim N(\hat{y}, \sigma_2^2)$, centers around the mean $\hat{y} = \mathbb{E}\hat{f}(\mathbf{x})$, representing the predicted value of $f(\mathbf{x})$. The noise term σ_1^2, arising from data, is irreducible and signifies data uncertainty. On the other hand, the squared bias term ($[\mathbb{E}\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2$) indicates the difference between estimated and true values, reflecting cognitive limitations due to model properties like hyperparameters and algorithms. The variance term, $\mathbb{E}[\hat{f}(\mathbf{x}) - \mathbb{E}\hat{f}(\mathbf{x})]^2$, relates to the model's sensitivity to training samples. Here, bias together with variance constitutes the model uncertainty. Figure 1 illustrates these concepts. By analyzing model uncertainty and its relation to error components (as shown in Figure 1), appropriate uncertainty quantification methods can be selected to enhance model performance.

Figure 2 shows how the model uncertainty affects the quality of the returned candidate. In a given SH run, half of the candidates are eliminated at each checkpoint marked by a vertical red dashed line. The solid blue line represents the best validation loss up to the current point, while the orange dashed line signifies the true quality (in terms of validation loss after convergence) of the candidates SH retains at that specific juncture. From the figure, we see that in every round, SH

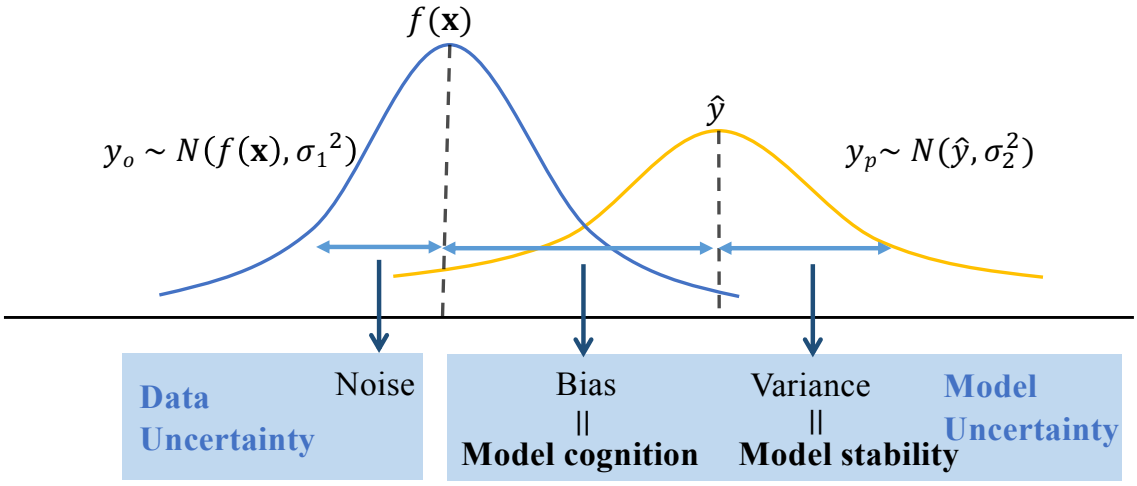


Figure 1: Decomposition of uncertainty.

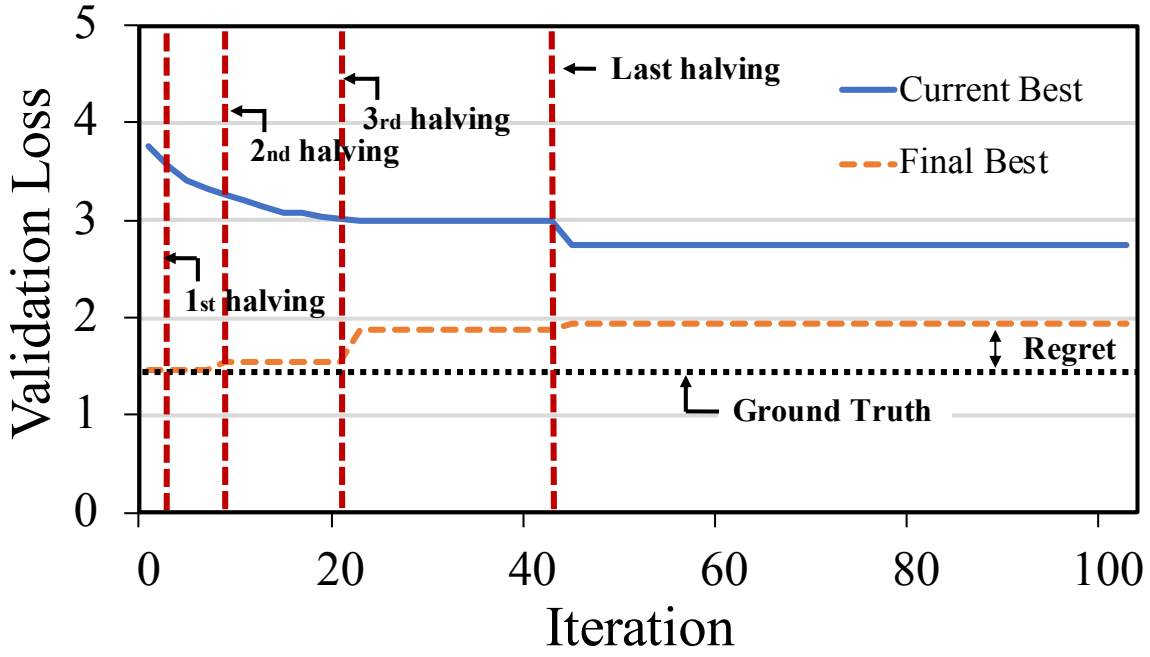


Figure 2: Demonstration of the negative impact from uncertainty on HPO; Successive Halving (SH) is used; the benchmark is NAS-BENCH-2.0 [9]. Due to its overlooking at model uncertainty, at each halving point, SH discards the actual best candidates, causing an increase in the regret.

discards the actually best candidates, causing a continuous increase of the regret. The reason is that the discarding decision of SH is solely based on the current validation loss, but model uncertainty, particularly pronounced in the early stages, disguises the true model potential.

3.2 Quantifying Model Uncertainty

Quantifying model uncertainty is the first step in treating it. High efficiency is essential here as the UQ happens during the HPO process. We employ a lightweight approach to conduct the UQ efficiently on the fly, as explained next.

Let $\gamma_1, \gamma_2, \cdots, \gamma_K \in \Gamma$ be K candidates drawn from the hyperparameter space Γ. Consider a supervised learning setting, where a machine learning model M is trained on some training set $D_T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_{n_t}, y_{n_t})\}$. Let M_γ^t denote the model with hyperparameter γ trained on D_T after t epochs, and M_γ^* the converged model. $M_\gamma^t(\mathbf{x})$ gives the prediction on a certain input $\mathbf{x} \in \mathbb{R}^d$.

We next define $\ell(\cdot, \cdot)$ to be the metric that evaluates the performance of a candidate. During training, a current validation loss for candidate γ_c, $\ell(\mathbf{y}, M_{\gamma_c}^t(\mathbf{X}))$, and model uncertainty are used by our UQ-based scheme in estimating the model converged loss, where t denotes the current epoch.

Following prior practice [18], the modeling assumes the following.

Assumption 3.1 (Converged model metric evaluation limit)**.** $\ell(\mathbf{y}, M_{\gamma_c}^*(\mathbf{X})) = \lim_{t\to\infty} \ell(\mathbf{y}, M_{\gamma_c}^t(\mathbf{X}))$ exists for $\gamma_c \in \Gamma$.

Remark 3.2. Assumption 3.1 says that the machine learning model will eventually converge after enough epochs. Note that the model makes no assumption between $\ell(\mathbf{y}, M_{\gamma_c}^1(\mathbf{X})), \ell(\mathbf{y}, M_{\gamma_c}^2(\mathbf{X})), \cdots, \ell(\mathbf{y}, M_{\gamma_c}^t(\mathbf{X}))$.

Assumption 3.3 (Independent Gaussian distributions)**.** Given a hyperparameter configuration γ_c, the converged loss of a model instance, $\ell(\mathbf{y}, M_{\gamma_c}^*(\mathbf{X}))$, can be affected by training data and other hyperparameters. We assume that the loss of a certain instance from these model variants, after convergence, follows Gaussian distribution: $\ell(\mathbf{y}, M_{\gamma_c}^*(\mathbf{X})) \sim \mathcal{N}(\mu_c, \sigma_c^2)$. Also, assume that $\ell(\mathbf{y}, M_{\gamma_1}^*(\mathbf{X})), \cdots, \ell(\mathbf{y}, M_{\gamma_m}^*(\mathbf{X}))$ are mutually independent.

In our method, at epoch t, we approximate μ_c by $\ell(\mathbf{y}, M_{\gamma_c}^t(\mathbf{X}))$ and model uncertainty σ_c by $\ell(\mathbf{y}, M_{\gamma_c}^b(\mathbf{X}))_{b\in D_t=[t-\delta,t]}$:

$$\begin{aligned}\hat{\sigma}_c^2 &= \frac{1}{\delta}\sum_{b\in D_t}(\mu_c - \mathbb{E}_{D_t}[\mu_c])^2 \\ &\approx \frac{1}{\delta}\sum_{b\in D_t}(\ell(\mathbf{y}, M_{\gamma_c}^b(\mathbf{X})) - \mathbb{E}_{D_t}[\ell(\mathbf{y}, M_{\gamma_c}^b(\mathbf{X}))])^2\end{aligned} \tag{1}$$

We use this approximation because of its simplicity and efficiency. What it needs for the UQ and approximation are only $\ell(\mathbf{y}, M_{\gamma_b}^t(\mathbf{X}))$ in the previous several training epochs before t. All the info is already available in the default HPO process; everything can hence be done on the fly with virtually zero overhead. We had considered some more sophisticated designs (e.g., offline training-based modeling of the relations between intermediate validation loss and the converged loss), but they add extra burden and overhead and hence barriers to adoption. Section 4 will show that this simple method goes a long way in enhancing HPO.

Definition 3.4 (UQ-guided comparison of candidates)**.** UQ-guided comparison of candidates compares two candidates based on the probability that the validation loss of the converged model γ_{c_1} is lower than that of γ_{c_2}, represented as follows based on the approximation from the current validation losses and uncertainty of the two candidates:

$$\begin{aligned}P = \Pr(&\ell(\mathbf{y}, M_{\gamma_{c_1}}^*(\mathbf{X})) > \ell(\mathbf{y}, M_{\gamma_{c_2}}^*(\mathbf{X})) \\ &\Big|\ell(\mathbf{y}, M_{\gamma_{c_1}}^t(\mathbf{X})), \hat{\sigma}_{c_1}, \ell(\mathbf{y}, M_{\gamma_{c_2}}^t(\mathbf{X})), \hat{\sigma}_{c_2}).\end{aligned} \tag{2}$$

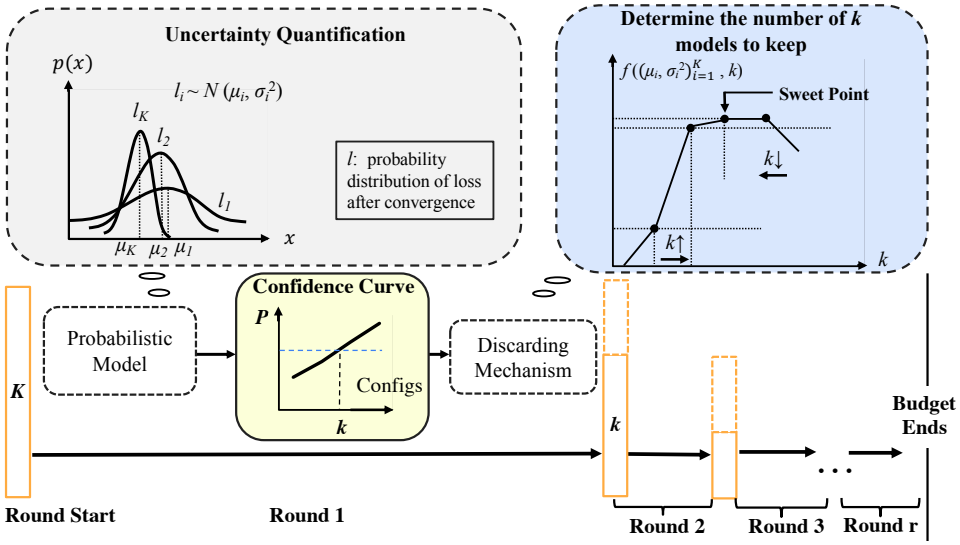


Figure 3: Illustration of using UQ-guided scheme to enhance Successive Halving. The goal is to select an optimal hyperparameter configuration from K candidates. It involves multiple rounds. R is the predefined budget resources (e.g., training epochs) for each round. For the first round, K candidates each get trained for $\frac{R}{K}$ epochs. Based on the observed validation loss and the quantified uncertainty for each candidate, the UQ-guided HPO represents each candidate's converged loss with a probability distribution. From that, it constructs a *confidence curve*, capturing the probability that the best configuration is among the current top k candidates for $1 \leq k \leq K$. From the curve, it then calculates $f((\mu_i, \sigma_i^2)_{i=1}^K, k)$, which captures the effects of keeping k top candidates ($1 \leq k \leq K$) for the next round, by considering the tradeoff between the risks of discarding the best candidate and the training budget each top candidate can get. From that, it identifies the best k value, discards the least promising $K - k$ candidates, and enters the next round. The process continues until the total budget is used up.

The main idea of Definition 3.4 is to use the current validation loss and quantified uncertainty to approximate the converged validation loss, so that we compare two candidates – more precisely, we compute the probability that one candidate is better than the other – based on the probability distribution of their converged validation loss. For example, if the current validation loss and uncertainty of two candidates γ_j and γ_k, at epoch t, are (μ_j, σ_j) and (μ_k, σ_k), using converged validation loss as the metric, we have $\Pr(\ell(\mathbf{y}, M_{\gamma_j}^t(\mathbf{X})) > \ell(\mathbf{y}, M_{\gamma_k}^t(\mathbf{X}))) = \Phi\left(\frac{\mu_k - \mu_j}{\sqrt{\sigma_j^2 + \sigma_k^2}}\right)$, where Φ denotes the cumulative distribution function (CDF) of the standard normal distribution. We next present *UQ-guided scheme*, a principled way to use UQ to guide HPO.

3.3 UQ-Guided Scheme

Figure 3 illustrates how UQ-guided scheme works in HPO. For the purpose of clarity, we base our explanation of the scheme on HPO that uses early stop mechanisms, but will show in Section 3.5 that the scheme is general, applicable to other HPO methods as well.

The original HPO has several *working rounds* and drops some candidates regarded as unpromising at the end of each round. With our UQ-guided scheme, at the end of each round, the scheme derives a *confidence curve* from the current probabilistic model, and uses a *discarding mechanism* to drop candidates that are unlikely to perform well after convergence. In contrast to the original HPO that

drops a fixed amount (or fraction) of candidates in each round, the UQ-guided scheme carefully calculates the number of candidates to drop in a round based on the probabilistic model such that the expected quality of the HPO outcomes can be maximized, as explained later.

The UQ-guided scheme respects the HPO budget—that is, the total amount of time usable by the HPO for identifying the best candidate. By default, it works around the given budget constraint: the budget for each round (R) equals the total budget divided by the number of rounds. We next discuss each step.

3.3.1 Confidence Curve Derived from Uncertainty Quantification

The concept of *confidence curve* is central in UQ-guided HPO. Define $[n] = \{1, 2, \cdots, n\}$.

Definition 3.5 (Confidence curve)**.** At epoch t, we evaluate each candidate's performance and sort them based on validation loss. A *confidence curve* $\mathcal{C}$ is a trajectory of a series of probabilities, $\{P_k | k \in [n]\}$, that depicts the probability that the optimal configuration (with the lowest loss after convergence) is among the first k configurations. For $k \in [n]$, P_k can be expressed as

$$\begin{aligned} P_k = \Pr(\, &\min(\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M^*_{\gamma_k}(\mathbf{X}))) \\ &\leq \min(\ell(\mathbf{y}, M^*_{\gamma_{k+1}}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X})))). \end{aligned}$$

The *confidence curve* is derived based on joint probability distribution in the following way. Suppose there are n candidates. At the end of a certain round, the probabilistic model returns n pairs of $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \cdots, (\mu_n, \sigma_n)$ as estimations for $\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X}))$, $\ell(\mathbf{y}, M^*_{\gamma_2}(\mathbf{X}))$, $\cdots$, $\ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X}))$. For simplicity, assume that $\mu_1 < \mu_2 < \cdots < \mu_n$, and $\sigma_1 = \sigma_2 = \cdots = \sigma$.

Let Φ and ϕ be the CDF and PDF of the standard normal distribution. For $k = m$, we can calculate P_m as

$$P_m = \int_{-\infty}^{\infty} \frac{1}{\sigma} \sum_{i=1}^{m} \frac{\phi(\frac{y+\mu_i}{\sigma})}{\Phi(\frac{y+\mu_i}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy. \tag{3}$$

The details of obtaining Equation 3 are in Appendix A.1.

3.3.2 Discarding Mechanism

The next step is to decide, at the end of each round, the appropriate value of k, which determines how many $(n-k)$ lowest-ranked candidates will be discarded in this round. Our scheme decides k based on the confidence curve: choosing the smallest k that satisfies $P_k \geq \tau$, where τ is a parameter determined by our scheme adaptively as follows.

Choosing τ. At the end of round i, we have the *confidence curve* $\mathcal{C}_i(P^i_1, P^i_2, \cdots, P^i_n)$ that is the trajectory of a series of probabilities. We quantify how τ influences the probability for the HPO to select the best candidate.

Let τ_i be the value of τ for round i, k_i be $\min\{k : \mathcal{C}_i(P_k) \geq \tau_i\}$. As the scheme discards the worst $n - k_i$ candidates and further trains the best k_i candidates in round $i+1$, we can derive the *confidence curve* of round $i+1$ as $\mathcal{C}_{i+1}(P^{i+1}_1, P^{i+1}_2, \cdots, P^{i+1}_{k_i})$ based on those selected k_i candidates. Since we want to quantify the effect of τ on the probability that the scheme returns the best candidate (that is, to suppose round $i+1$ is our final round), P^{i+1}_1 is the target we desire to maximize. Define $\boldsymbol{\xi}_i$ to be the current condition $(\mu_i, \sigma_i)^n_{i=1}$. Let $f(\cdot, \cdot)$ be a mapping such that $f(\boldsymbol{\xi}_i, \tau_i) = P^{i+1}_1$. We want to use a selector function $\Psi : D \to [0, 1]$ where $D = ([0, \infty) \times [0, \infty))^n \times [0, 1]$. Ψ takes $\boldsymbol{\xi}_i$ as input and returns an optimal τ_i:

$$\Psi(\boldsymbol{\xi}_i) = \arg \max_{\tau_i \in [0,1]} \{f(\boldsymbol{\xi}_i, \tau_i)\}. \tag{4}$$

The effect of τ_i on f manifests through its influence on the number of candidates k_i retained in the subsequent round, and can be ultimately broken down into the influence of (1) *exploration*, meaning keeping more candidates in the next round can reduce this round's discarding error, and (2) *exploitation*, meaning keeping fewer candidates in the next round can allow each candidate to receive more training time (recall that the training time budget is fixed for each round) and hence will increase the reliability of the validation at the end of the next round.

Exploration. If k_i drops by 1 to k_i', according to the definition of the *confidence curve*, the probability that the final optimal configuration is among the remaining candidates we keep drops by $\Delta c_{\downarrow} = P_{k_i} - P_{k_i'}$:

$$\Delta c_{\downarrow} = \int_{-\infty}^{\infty} \frac{1}{\sigma} \cdot \frac{\phi(\frac{y+\mu_{k_i}}{\sigma})}{\Phi(\frac{y+\mu_{k_i}}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy. \tag{5}$$

Exploitation. At the same time, a drop in k_i leads to an increase in the individual training budget b. Let ζ be the coefficient that relates the increase in the number of training epochs to its corresponding effect on confidence. Using an approach similar to that employed in formulating the *confidence curve*, we have

$$\begin{aligned} \zeta = &\int_{-\infty}^{\infty} \frac{1}{\sigma - \Delta_t\sigma} \phi(\frac{y+\mu_1}{\sigma - \Delta_t\sigma}) \cdot \prod_{i=2}^{k_i} \Phi(\frac{y+\mu_i}{\sigma - \Delta_t\sigma}) dy \\ &- \int_{-\infty}^{\infty} \frac{1}{\sigma} \phi(\frac{y+\mu_1}{\sigma}) \cdot \prod_{i=2}^{k_i} \Phi(\frac{y+\mu_i}{\sigma}) dy \end{aligned} \tag{6}$$

where t represents the current total number of epochs and $\Delta_t\sigma$ represents the reduction in the uncertainty σ that would result from training each candidate for one additional epoch. The specifics for Equations 5 and 6 can be found in Appendix A.2. Given that b increases by $\frac{R}{k_i'} - \frac{R}{k_i}$, the overall influence of *exploitation* on the probability of selecting an optimal candidate is $\Delta c_{\uparrow} = \frac{R}{k_i(k_i-1)}\zeta$.

Let $\zeta(k_i, \boldsymbol{\xi}_i)$ be the confidence increase, given condition $\boldsymbol{\xi}_i$, when each of the k_i candidates gets a unit extra training budget. $\mathcal{C}_i(P_k), k \in [n]$ are the *confidence curves*. Balancing *exploration* and *exploitation* leads to a sweet point where $\Delta c_{\downarrow} = \Delta c_{\uparrow}$. That gives the way to derive the appropriate value for τ, which just needs to make the following hold:

$$P_{k_i} - P_{k_i - 1} = \frac{R}{k_i(k_i - 1)} \zeta(k_i, \boldsymbol{\xi}_i). \tag{7}$$

3.4 Theoretical Analysis

We consider how the method performs in terms of identifying the best candidate. For all $i \in [n], k \geq 1$, let $\ell_{i,k} \in \mathbb{R}$ be an i.i.d. sample from the Gaussian distribution $\mathcal{N}$. For each i, assume $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$ exists. The goal is to identify $\arg\min_i \nu_i$. Without loss of generality, assume that $\nu_1 < \nu_2 \leq \cdots \leq \nu_n$. The assumption that $\lim_{\tau \to \infty} \ell_{i,\tau}$ exists implies that as τ grows, the overall gap between $\ell_{i,\tau}$ and ν_i decreases. Let $\sigma_t = f(t)$ be the model uncertainty at epoch t. We then introduce a random variable that characterizes the approximation error of $\ell_{i,t}$ relative to ν_i, modeling it as a distribution that incorporates t as a parameter:

$$X_t = |\ell_{i,t} - \nu_i|, X_t \sim \mathcal{N}(0, \sigma_t^2) \quad \forall t.$$

By applying Chebyshev inequality, we have

$$\begin{aligned}\Pr(|\ell_{i,t}-\nu_i|>\frac{\nu_i-\nu_1}{2}) &\le \frac{4\sigma_t^2}{(\nu_i-\nu_1)^2}\\ &= \frac{4f(t)^2}{(\nu_i-\nu_1)^2} \quad i=2,\cdots,n.\end{aligned} \tag{8}$$

Let $\mathcal{A}$ denote the event $\ell_{i,t}>\ell_{1,t}$, then by Equation 8

$$\begin{aligned}\Pr(\mathcal{A}) &= \Pr((\ell_{i,t}-\nu_i)+(\nu_1-\ell_{1,t})+2\cdot(\frac{\nu_i-\nu_1}{2})>0)\\ &\ge 1-\Big(\frac{4f(t)^2}{(\nu_i-\nu_1)^2}\Big)^2.\end{aligned} \tag{9}$$

Equation 9 tells us that $\ell_{i,t}>\ell_{1,t}$ has a high probability with respect to t if $f(t)\in O(t^{-1/4})$ (see Lemma in Appendix B). That is, comparing the intermediate values at a certain time t is likely to establish an order similar to the order of the final values of ν_i and ν_1.

The following theorem is stated in terms of the abovementioned quantities with proofs in Appendix B.1.

Theorem 3.6. *Let n be the number of total candidates, and $\nu_i=\lim_{\tau\to\infty}\ell_{i,\tau}$. For a given $c>0$, there exists a $T>0$ s.t. $\prod_{i=2}^{n}(1-(\frac{4f(T)^2}{(\nu_i-\nu_1)^2})^2)>1-c$. If the round budget $R>T\cdot n$, then the best candidate is returned with probability $P>(1-\lfloor\frac{B}{R}\rfloor c)(1-c)$, where B is the total budget.*

In comparison, the bound in the UQ-oblivious approach is as follows:

Theorem 3.7. *Let $\delta>0,\nu_i=\lim_{\tau\to\infty}\ell_{i,\tau}$ and assume $\nu_1\le\nu_2\le\cdots\le\nu_n$. Let $\gamma^{-1}(\epsilon,\delta)=\min\{t\in\mathbb{N}:\frac{f(t)}{\epsilon}\le\delta^{\frac{1}{4}}\}$, and*

$$\begin{aligned}z_{ob} &= 2\lceil\log_2(n)\rceil\max_{i=2,\dots,n} i\left(1+\gamma^{-1}(\tfrac{\nu_i-\nu_1}{2},\delta)\right)\\ &\le 2\lceil\log_2(n)\rceil(n+\sum_{i=1,\dots,n}\gamma^{-1}(\tfrac{\nu_i-\nu_1}{2},\delta)).\end{aligned}$$

If the UQ-oblivious early stopping method is run with any budget $B_{ob}>z_{ob}$ then the best candidate is returned with probability $P_{ob}>1-n\delta$.

Example 3.8. *Consider $f(t)=\frac{1}{t}$. According to Theorem 3.7, if $B_{ob}>2\lceil\log_2(n)\rceil(n+\sum_{i=1,\dots,n}\gamma^{-1}(\frac{\nu_i-\nu_1}{2},\delta))$, the UQ-oblivious method can return the best candidate with probability over $1-n\delta$. But if $B_{UQ}\simeq\gamma^{-1}(\frac{\nu_2-\nu_1}{2},\delta)\cdot n$ [1], the UQ method can return the best candidate with probability over $1-n\delta$. As shown in Appendix B.2, Theorems 3.6 and 3.7 together show that the UQ approach guarantees the same probability of identifying the optimal candidate as the UQ-oblivious counterpart with a smaller budget lowerbound B (see Corollary B.2).*

We have been concerned with identifying the best candidate, while in practice, it is often sufficient to consider a situation where the difference between the result of candidate i_ϵ (ν_{i_ϵ}) and the result of the best candidate (ν_1) is less than or equal to a small value ϵ. We obtain the following theorem with proofs in Appendix B.3.

[1] $f\simeq g$ if there are constants c,c' s.t. $cg(x)\le f(x)\le c'g(x)$.

Theorem 3.9. *For a budget $B > R$ and a set of n candidates, let $\hat{i}$ be the output of the UQ-guided approach. Then*

$$\Pr(\nu_{\hat{i}} - \nu_1 > \epsilon) \leq \frac{2\lfloor \frac{B}{R} \rfloor f(R)^2}{\epsilon}.$$

In comparison, $\hat{i}_D$, the output of the UQ-oblivious counterpart, satisfies

$$\Pr(\nu_{\hat{i}_D} - \nu_1 \geq \epsilon) \leq \frac{2\lceil \log_2(n) \rceil f\left(\lfloor \frac{B}{n\lceil \log_2(n) \rceil} \rfloor \right)^2}{\epsilon}.$$

Example 3.10. *Consider $f(t) = \frac{1}{t}$. Substitution of $f(t)$ in Theorem 3.9 can clearly show a smaller upperbound of the UQ-guided approach than that of the UQ-oblivious counterpart (see Appendix B.3 for details).*

The theorems provide some insights into the theoretical benefits of the UQ-guided scheme. But it is worth noting that neither this bound comparison nor the budget bound comparison in Example 3.8 is sufficient to prove that the UQ-guided approach definitely would outperform the UQ-oblivious approach, a reason for the empirical comparisons in Section 4.

[tb] **Input:** Total budget B, the set of K configurations $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_K\}$, minimum round budget R **Output:** The configuration with the best performance $b = \lfloor \frac{R}{K} \rfloor$ $i = 1$ **to** K Evaluate $M_{\gamma_i}^{t_i}$ with budget b and get $M_{\gamma_i}^{t_i+b}$ $t_i += b$ Rank according to performance and obtain new $M_{\gamma_1}^{t_1}$, $M_{\gamma_2}^{t_2}, \cdots, M_{\gamma_K}^{t_K}$ $K = OracleModel(M_{\gamma_1}^{t_1}, M_{\gamma_2}^{t_2}, \cdots, M_{\gamma_K}^{t_K})$ Keep top K candidates total budget B runs out

[tb] **Input:** K instances of Machine Learning models $M_{\gamma_1}^{t_1}$, $M_{\gamma_2}^{t_2}, \cdots, M_{\gamma_K}^{t_K}$ **Output:** A new K that tells the model how many candidates to keep Get a τ from the probabilistic model Construct the *confidence curve* $\mathcal{C}(P_1, P_2, \cdots, P_K)$ based on $\ell(\mathbf{y}, M_{\gamma_1}^{t_1}(\mathbf{X})), \ell(\mathbf{y}, M_{\gamma_2}^{t_2}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M_{\gamma_K}^{t_K}(\mathbf{X}))$ **return** $\min\{k : P_k > \tau\}$

3.5 UQ-Guided HPO Family

The *UQ-guided scheme* is a general approach to enhancing HPO with uncertainty awareness. We next explain how it is integrated into several existing HPO methods to transform them into UQ-guided ones, yielding a UQ-guided HPO family. In the following, we use the suffix "plus (+)" to indicate the UQ-guided HPO methods.

Successive Halving plus (SH+) is derived from the early stop-based HPO design Successive Halving (SH) [18]. Algorithms 3.4 and 3.4 show the pseudo-code. Given total budget B and round budget R and an initial K, SH+ first trains K candidates each with the initial $b = \lfloor \frac{R}{K} \rfloor$ units of budget, and ranks them by the evaluation performance. Then SH+ updates K based on Section 3.3.2 and keeps the top K configurations according to the UQ-guided scheme ($OracleModel$ in Algorithms 3.4 and 3.4), and continues the process until the budget runs out.

Hyperband plus (HB+) originates from the popular HPO design Hyperband (HB). HB is an HPO method trying to better balance exploration and exploitation than SH does [24] by adding an outer loop for grid search of the value of K. HB+ simply extends HB by using SH+ rather than SH as its inner loop, changing the target of the grid search to the initial value of K.

Bayesian Optimization and Hyperband plus (BOHB+) is developed from BOHB [11]. BOHB is similar to HB except that it replaces the random sampling from the uniform distribution with BO-based sampling. BOHB+ makes the corresponding changes from HB+ by adopting BO-based sampling for its outer loop.

Sub-sampling plus (SS+) is derived from the Sub-sampling (SS) algorithm [14]. It showcases the applicability of the UQ-guided scheme to non-early stop–based methods. Similar to other

methods, in each round, SS also chooses candidates for further training based on its assessment of the potential of those candidates. But unlike the other methods, SS does not discard any candidates, but keeps all in play throughout the entire HPO process. In each round, the candidates it chooses are those that show smaller validation loss than the most trained candidate shows. If there is none, it trains only the most trained candidate in that round. SS+ integrates the UQ-guided scheme into the candidate selection process of SS. When SS+ compares a candidate (c_i) against the most trained candidate (c_m), rather than checking their validation losses, it uses the UQ-guided scheme to compute the probability for the convergence loss of c_i to be smaller than that of c_m and checks whether the probability is over a threshold τ (0.9 in our experiments), that is, $\Pr(\ell(y, M^*_{\gamma_{c_m}}(\mathbf{x})) \geq \ell(y, M^*_{\gamma_{c_i}}(\mathbf{x}))) \geq \tau$.

4 Experiments

We conduct a series of experiments on the four UQ-guided HPO methods to validate the efficacy of the UQ-guided scheme for HPO.

4.1 Experimental Setup

Methodology. To check the benefits of the UQ-guided scheme for HPO, we apply the proposed UQ-guided HPO family to different HPO benchmarks, including NAS-BENCH-201 and LCBench, to measure the performance for different hyperparameter optimization tasks, and compared those with their original UQ-oblivious versions.

Workloads. We evaluate the UQ-guided methods on two real-world benchmarks. Nas-Bench-201 [9] encompasses three heavyweight neural architecture search tasks (NAS) on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets. In addition, we investigate the performance of optimizing traditional ML pipelines, hyperparameters, and neural architecture in LCBench [36]. For example, we optimized 7 parameters for the Fashion-MNIST dataset [7], where the resource type is determined by the number of iterations. Additional information regarding these benchmarks can be found in Appendix D. In this context, one unit of budget equates to a single training epoch, and by default, the total HPO budget (B) allocated for each method is 4 hours.

4.2 Experimental Results

Figure 4 illustrates the results of NAS-BENCH-2.0 trained on ImageNet16-120.

It shows the results of four UQ-guided methods compared to their original ones. For each comparison, we show three metrics, namely top-1 rank on different trials, top-1 rank on different fractions of budgets, and regret on different fractions of budgets. Top-1 rank refers to the real ranking of the candidate ultimately chosen by the method. Regret (%) refers to the accuracy difference between the returned candidate and the real best candidate. The benefits of the UQ-guided scheme are obvious, both for individual trials and across different fractions of budgets. It brings a 21-55% regret reduction. Similar results are observed on other benchmarks (LCBench results shown in Appendix E).

Table 1 provides the fraction of the total exploration time needed for the UQ-guided methods to achieve comparable model accuracy as the original methods do. The UQ-guided methods need much less time than their counterparts to obtain a similar performance. For instance, SH+ achieves the same average regret of 5% on NAS with only half of the budgets required by SH. These results indicate that the UQ technique can conduct HPO efficiently and effectively.

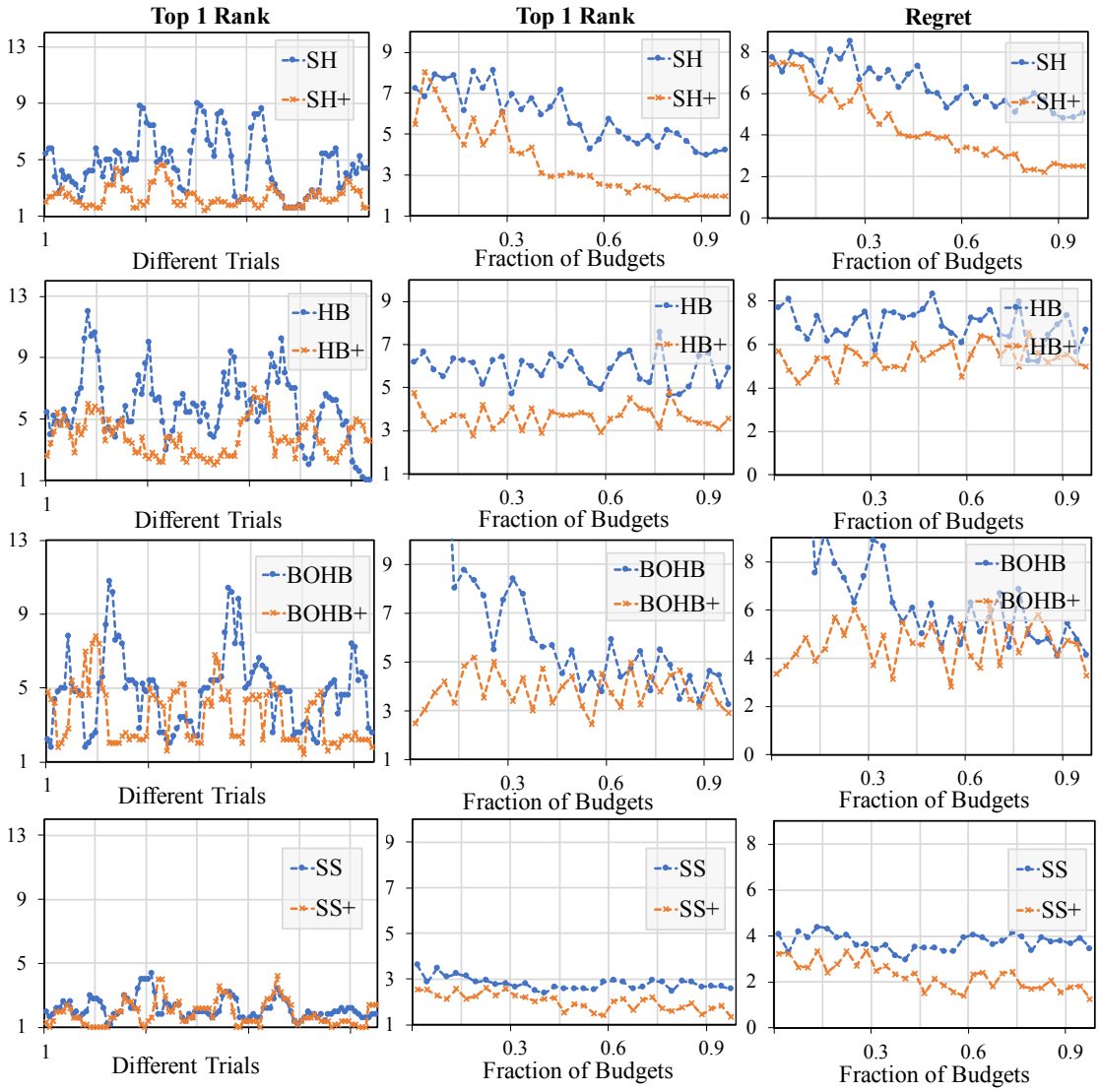


Figure 4: Experimental results of UQ-oblivious HPO methods and their UQ-guided enhancements on NAS-BENCH-2.0.

5 Conclusion

This paper points out the importance of systematic treatment to the uncertainty in model trainings for HPO. It introduces a novel scheme named *UQ-guided scheme*, which offers a general way to enhance HPO methods with uncertainty-awareness. Experiments demonstrate that the UQ-guided scheme can be easily integrated into various HPO methods. The enhanced methods achieve 21–55% reduction of regret over their original versions, and requires only 30–75% time to identify a candidate with a matching performance as the original methods do. The paper in addition provides theoretical analysis of the effects of the UQ-guided scheme for HPO.

Overall, this study concludes that UQ is important for HPO to consider, simple on-the-fly UQ goes a long way for HPO, and the *UQ-guided scheme* can serve as a general effective scheme for enhancing HPO designs.

Table 1: Fraction of time (%) required for the UQ-guided methods to achieve comparable model performance as the original HPO methods do.

Methods	NAS-BENCH-201	LCBench
SH+	50.78	43
HB+	75	60
BOHB+	68.4	53.34
SS+	47.84	30.93

References

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*, 2017.

[3] Hadrien Bertrand, Roberto Ardon, Matthieu Perrot, and Isabelle Bloch. Hyperparameter optimization of deep neural networks: Combining hyperband with bayesian model selection. In *Conférence sur l'Apprentissage Automatique*, 2017.

[4] 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.

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[6] Danruo Deng, Guangyong Chen, Yang Yu, Furui Liu, and Pheng-Ann Heng. Uncertainty estimation by fisher information-based evidential deep learning. In *International conference on machine learning*. PMLR, 2023.

[7] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[8] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[9] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[10] Vincent Dumont, Casey Garner, Anuradha Trivedi, Chelsea Jones, Vidya Ganapati, Juliane Mueller, Talita Perciano, Mariam Kiran, and Marc Day. Hyppo: A surrogate-based multi-level parallelism tool for hyperparameter optimization. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pages 81–93. IEEE, 2021.

[11] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR, 10–15 Jul 2018.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[13] Yi-Qi Hu, Yang Yu, Wei-Wei Tu, Qiang Yang, Yuqiang Chen, and Wenyuan Dai. Multi-fidelity automatic hyper-parameter tuning via transfer series expansion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3846–3853, 2019.

[14] Yimin Huang, Yujun Li, Hanrong Ye, Zhenguo Li, and Zhihua Zhang. Improving model training with multi-fidelity hyperparameter evaluation. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 485–502, 2022.

[15] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

[16] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[17] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[18] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR.

[19] Jie Jiang, Jiawei Jiang, Bin Cui, and Ce Zhang. Tencentboost: A gradient boosting tree system with parameter server. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 281–284. IEEE, 2017.

[20] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, pages 1799–1808. PMLR, 2017.

[21] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pages 528–536. PMLR, 2017.

[22] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *International Conference on Learning Representations*, 2016.

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

[24] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, jan 2017.

[25] Yang Li, Yu Shen, Jiawei Jiang, Jinyang Gao, Ce Zhang, and Bin Cui. Mfes-hb: Efficient hyperband with multi-fidelity quality measurements. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8491–8500, 2021.

[26] Siyan Liu, Pei Zhang, Dan Lu, and Guannan Zhang. PI3NN: Out-of-distribution-aware prediction intervals from three neural networks. In *International Conference on Learning Representations*, 2022.

[27] Jingwei Ma, Jiahui Wen, Mingyang Zhong, Weitong Chen, and Xue Li. Mmm: multi-source multi-net micro-video recommendation with clustered hidden item representation learning. *Data Science and Engineering*, 4:240–253, 2019.

[28] Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Gonzalo Nápoles. Multi-objective hyperparameter optimization with performance uncertainty. *Communications in Computer and Information Science*, 1684:37–46, 2022.

[29] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[30] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

[31] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity bayesian optimization with max-value entropy search and its parallelization. In *International Conference on Machine Learning*, pages 9334–9345. PMLR, 2020.

[32] 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.

[33] Shiwen Wu, Yuanxing Zhang, Chengliang Gao, Kaigui Bian, and Bin Cui. Garg: anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *Data Science and Engineering*, 5:433–447, 2020.

[34] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

[35] Wentao Zhang, Jiawei Jiang, Yingxia Shao, and Bin Cui. Snapshot boosting: a fast ensemble framework for deep neural networks. *Science China Information Sciences*, 63:1–12, 2020.

[36] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

A Method Details

A.1 Formulation of *Confidence Curve*

The following contents detail how to compute the *confidence curve* $\mathcal{C}(P_1, P_2, \cdots, P_n)$ based on current validation loss and quantified uncertainty of the n candidates.

Let Y be the random variable denoting the negation of the lowest converged validation loss:

$$Y = \max(-\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X})), -\ell(\mathbf{y}, M^*_{\gamma_2}(\mathbf{X})), \cdots, -\ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X}))).$$

Since $\ell(\mathbf{y}, M^*_{\gamma_i}(\mathbf{X})) \sim \mathcal{N}(\mu_i, \sigma_i^2)$, the cumulative distribution function (CDF) of Y, $F_Y(y)$, is

$$\begin{aligned} F_Y(y) &= \Pr(Y \le y) = \Pr(-\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X})) \le y, -\ell(\mathbf{y}, M^*_{\gamma_2}(\mathbf{X})) \le y, \cdots, -\ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X})) \le y) \\ &= \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) = \exp(\sum_{i=1}^{n} \ln \Phi(\frac{y+\mu_i}{\sigma})). \end{aligned}$$

Accordingly, the probability density function (PDF) of Y, $f_Y(y)$, is

$$f_Y(y) = \frac{dF_Y(y)}{dy} = \frac{1}{\sigma} \sum_{i=1}^{n} \frac{\phi(\frac{y+\mu_i}{\sigma})}{\Phi(\frac{y+\mu_i}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}).$$

Now we can construct the *confidence curve* by calculating each P_k ($k \in [n]$). For $k = m$, P_k can be expressed as

$$P_m = \Pr(\min(\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M^*_{\gamma_m}(\mathbf{X}))) \le \min(\ell(\mathbf{y}, M^*_{\gamma_{m+1}}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X})))).$$

According to Assumption 3.3, $\ell(\mathbf{y}, M^*_{\gamma_1}(\mathbf{X})), \cdots, \ell(\mathbf{y}, M^*_{\gamma_n}(\mathbf{X}))$ are mutually independent, thus

$$\begin{aligned} P_m &= \int_{-\infty}^{\infty} f_Y(y) \Pr(-\mathcal{N}(\mu_{m+1}, \sigma_{m+1}) \le y, \cdots, -\mathcal{N}(\mu_n, \sigma_n) \le y) dy \\ &= \int_{-\infty}^{\infty} f_Y(y) \Phi(\frac{y+\mu_{m+1}}{\sigma}) \times \cdots \times \Phi(\frac{y+\mu_n}{\sigma}) dy \\ &= \int_{-\infty}^{\infty} \frac{1}{\sigma} \sum_{i=1}^{m} \frac{\phi(\frac{y+\mu_i}{\sigma})}{\Phi(\frac{y+\mu_i}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy. \end{aligned} \tag{10}$$

A.2 Computing ζ

Equation 5 can be calculated by directly subtracting P_{k_i} by P_{k_i-1}:

$$\begin{aligned} \Delta c_{\downarrow} &= \int_{-\infty}^{\infty} \frac{1}{\sigma} \sum_{i=1}^{k_i} \frac{\phi(\frac{y+\mu_i}{\sigma})}{\Phi(\frac{y+\mu_i}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy - \int_{-\infty}^{\infty} \frac{1}{\sigma} \sum_{i=1}^{k_i-1} \frac{\phi(\frac{y+\mu_i}{\sigma})}{\Phi(\frac{y+\mu_i}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy \\ &= \int_{-\infty}^{\infty} \frac{1}{\sigma} \cdot \frac{\phi(\frac{y+\mu_{k_i}}{\sigma})}{\Phi(\frac{y+\mu_{k_i}}{\sigma})} \cdot \prod_{i=1}^{n} \Phi(\frac{y+\mu_i}{\sigma}) dy. \end{aligned}$$

ζ is the coefficient that relates the increase in the number of training epochs to its corresponding effect on confidence. Consider a working round that starts with k candidates. We have approximations at the end of the round for the converged loss $\ell(y, M^*_{\gamma_i}(\mathbf{x})) \sim \mathcal{N}(\mu_i, \sigma^2)$ for $i \in [k]$. Here,

t denotes the epochs. Letting each candidate train one extra unit of resource results in lower uncertainty, thus increasing the f score. First compute $f(\boldsymbol{\xi}, \tau)$ at epoch t:

$$f(\boldsymbol{\xi}, \tau)_t = \int_{-\infty}^{\infty} \frac{1}{\sigma}\phi(\frac{y+\mu_1}{\sigma}) \cdot \prod_{i=2}^{k} \Phi(\frac{y+\mu_i}{\sigma})dy. \tag{11}$$

If each configuration is trained with one extra unit of resource (a total of $t+1$ epochs), model uncertainty would be reduced. We use the same μ_i to approximate the converged validation loss for $t+1$ epochs, and use $\sigma - \Delta_t\sigma$ as an approximation for model uncertainty. Here $\Delta_t\sigma$ is the decrease in model uncertainty that is determined through offline profiling with details in Section C. This approximation leads us to $f(\boldsymbol{\xi}, \tau)$ at epoch $t+1$ as

$$f(\boldsymbol{\xi}, \tau)_{t+1} = \int_{-\infty}^{\infty} \frac{1}{\sigma - \Delta_t\sigma}\phi(\frac{y+\mu_1}{\sigma - \Delta_t\sigma}) \cdot \prod_{i=2}^{k} \Phi(\frac{y+\mu_i}{\sigma - \Delta_t\sigma})dy. \tag{12}$$

Subtracting Equation 12 by Equation 11 gives us the result in Equation 6:

$$\begin{aligned}\zeta = f(\boldsymbol{\xi}, \tau)_{t+1} - f(\boldsymbol{\xi}, \tau)_t = &\int_{-\infty}^{\infty} \frac{1}{\sigma - \Delta_t\sigma}\phi(\frac{y+\mu_1}{\sigma - \Delta_t\sigma}) \cdot \prod_{i=2}^{k} \Phi(\frac{y+\mu_i}{\sigma - \Delta_t\sigma})dy \\ &- \int_{-\infty}^{\infty} \frac{1}{\sigma}\phi(\frac{y+\mu_1}{\sigma}) \cdot \prod_{i=2}^{k} \Phi(\frac{y+\mu_i}{\sigma})dy.\end{aligned}$$

B Proofs

In this section, we provide proofs for the theorems presented in Section 3.4. At the end of the proof, we let $f(t) = \frac{1}{t}$ and obtained the results in Example 3.8 and Example 3.10.

The Lemma stated next will prove to be useful.

Lemma B.1. *For $i > 1$, if $\min\{t_1, t_i\} > t$, then we have a high probability that $\ell_{i,t_i} > \ell_{1,t_1}$ with respect to t if $f(t) \in O(t^{-1/4})$.*

Proof. In Section 3.4, we come to the conclusion that

$$\Pr(\ell_{i,t} > \ell_{1,t}) \geq 1 - (\frac{4f(t)^2}{(\nu_i - \nu_1)^2})^2 = 1 - (\frac{2}{\nu_i - \nu_1})^4 \cdot f(t)^4 > 1 - O\left(\frac{1}{t}\right).$$

This shows that the event $\ell_{i,t} > \ell_{1,t}$ happens with high probability with respect to t.

Now consider a more general setting, where each ℓ_i has its own t_i:

$$\Pr(|\ell_{i,t_i} - \nu_i| > \frac{\nu_i - \nu_1}{2}) \leq \frac{4f(t_i)^2}{(\nu_i - \nu_1)^2} \;\; i = 1, \cdots, n.$$

Comparing ℓ_{i,t_i} and ℓ_{1,t_1} for a particular $i \in [n]$ gives us the following:

$$\Pr(\ell_{i,t_i} > \ell_{1,t_1}) = \Pr((\ell_{i,t_i} - \nu_i) + (\nu_1 - \ell_{1,t_1}) + 2 \cdot \frac{\nu_i - \nu_1}{2} > 0) \geq 1 - \frac{4f(t_1)^2}{(\nu_i - \nu_1)^2} \cdot \frac{4f(t_i)^2}{(\nu_i - \nu_1)^2}. \tag{13}$$

Since $t_1 > t$ and $t_i > t$,

$$(13) > 1 - O\left(\frac{1}{t}\right).$$

□

B.1 Proof of Theorem 3.6

Proof. Let S_i be the set of candidates the UQ scheme evaluates at the beginning of the i-th round. We assume that the n infinitely long loss sequences $[\ell_{i,t}]$ with limits $\{\nu_i\}_{i=1}^n$ satisfy Assumption 3.1 and 3.3.

We compute the probability that the algorithm includes the best candidate in the last round, namely, $1 \in S_{\lfloor \frac{B}{R} \rfloor}$, and the probability that the UQ scheme returns the best candidate in $S_{\lfloor \frac{B}{R} \rfloor}$.

Let r_k be the round budget for each candidate in S_k. $R_k = \sum_{j=0}^{k} r_k$. The probability that the best candidate is among the final kept candidate set is

$$\begin{aligned}
\Pr(1 \in S_{\lfloor \frac{B}{R} \rfloor}) &= 1 - \sum_{k=1}^{r=\lfloor \frac{B}{R} \rfloor} \Pr(1 \notin S_k, 1 \in S_{k-1}) \\
&= 1 - \sum_{k=0}^{r=\lfloor \frac{B}{R} \rfloor - 1} (\Pr(1 \in S_{k-1}) - \Pr(1 \in S_k, 1 \in S_{k-1})) \qquad (14) \\
&\geq 1 - \sum_{k=0}^{r=\lfloor \frac{B}{R} \rfloor - 1} \left(1 - \Pr\Big(\bigwedge_{i \in S_k \setminus \{1\}} \ell_{i,R_k} > \ell_{1,R_k}\Big)\right).
\end{aligned}$$

Since the probability that $\ell_{1,t}$ is the smallest among all $\ell_{k,t}$ ($k \in [n]$) is greater than

$$\prod_{i=2}^{n} \Pr(\ell_{i,t} > \ell_{1,t}) \geq \prod_{i=2}^{n} (1 - \Big(\frac{4f(t)^2}{(\nu_i - \nu_1)^2}\Big)^2),$$

we have

$$\begin{aligned}
(14) &\geq 1 - \sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} (1 - \prod_{i=2}^{n} (1 - (\frac{4f(R_k)^2}{(\nu_i - \nu_1)^2})^2)) \\
&= 1 - \sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} (1 - \prod_{i=2}^{n} (1 - (\frac{4f\big(\sum_{j=0}^{k} \frac{R}{|S_j|}\big)^2}{(\nu_i - \nu_1)^2})^2)) \\
&\geq 1 - \lfloor \frac{B}{R} \rfloor c.
\end{aligned}$$

Therefore, the probability that the scheme returns the best candidate is no less than

$$\Pr(1 \in S_{\lfloor \frac{B}{R} \rfloor}) \cdot \Pr\Big(\bigwedge_{i \in S_{\lfloor \frac{B}{R} \rfloor}} \ell_{i,R_{\lfloor \frac{B}{R} \rfloor}} > \ell_{1,R_{\lfloor \frac{B}{R} \rfloor}}\Big) \geq (1 - \lfloor \frac{B}{R} \rfloor c)(1 - c). \qquad (15)$$

□

That is to say, for a given confidence threshold c, there exists a T such that as long as $\min\{t_1, t_2, ..., t_n\} > T$, then $\Pr(\wedge_{i=2,...,n}(\ell_{i,t_i} > \ell_{1,t_1})) > 1 - c$. Recall that in the UQ scheme design, the goal is to select an optimal hyperparameter configuration from n candidates, and each round is allocated for R resources. This means that if choose $R \geq T \cdot n$, then the best candidate is returned from the algorithm with probability $P > (1 - \lfloor \frac{B}{R} \rfloor c)(1 - c)$.

B.2 Proof of Theorem 3.7

Proof. First we show that, given budget $z = z_{ob}$, round budget for round k satisfies

$$\begin{aligned} r_k &\geq \frac{z}{|S_k|\lceil \log_2 n \rceil} - 1 \\ &= \frac{2}{|S_k|} \max_{i=2,\dots,n} i \left(1 + \gamma^{-1}(\tfrac{\nu_i - \nu_1}{2}, \delta)\right) - 1 \\ &\geq \frac{2}{|S_k|}(\lfloor |S_k|/2 \rfloor + 1) \left(1 + \gamma^{-1}(\tfrac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}, \delta)\right) - 1 \\ &\geq \gamma^{-1}(\tfrac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}, \delta). \end{aligned}$$

The last inequality is derived because $\lfloor |S_k|/2 \rfloor \geq |S_k|/2 - 1$.

Let $\tau_i := \gamma^{-1}(\frac{\nu_i - \nu_1}{2}, \delta)$. We then show that, for a time t:

$$\begin{aligned} t \geq \tau_i \Rightarrow t &\geq \gamma^{-1}(\tfrac{\nu_i - \nu_1}{2}, \delta) \\ &\Leftrightarrow 1 - \left(\frac{4f(t)^2}{(\nu_i - \nu_1)^2}\right)^2 \geq 1 - \delta \\ &\Rightarrow \Pr(\ell_{i,t} > \ell_{1,t}) \geq 1 - \delta. \end{aligned}$$

The second line follows by the definition of $\gamma^{-1}(\epsilon, \delta)$. Since $r_k \geq \tau_{\lfloor |S_k|/2 \rfloor + 1}$, we can compute the following probability

$$\begin{aligned} \Pr(1 \in S_{k+1} | 1 \in S_k) &= \Pr\Big(\sum_{i \in S_k} \mathbf{1}\{\ell_{i,R_k} > \ell_{1,R_k}\} \geq \lfloor |S_k|/2 \rfloor\Big) \\ &\geq \Pr(\sum_{i=\lfloor |S_k|/2 \rfloor + 1}^{|S_k|} \mathbf{1}\{\ell_{i,R_k} > \ell_{1,R_k}\} \geq \lfloor |S_k|/2 \rfloor) \\ &\geq (1-\delta)^{\lfloor \frac{|S_k|}{2} \rfloor} \end{aligned}$$

where the first line follows by the definition of the early stopping algorithm (Successive Halving), the second by τ_i being non-increasing. Namely, for all $i > \lfloor |S_k|/2 \rfloor + 1$, we have $\tau_i \leq \tau_{\lceil |S_k|/2 \rceil + 1}$ and consequently, $\Pr(\ell_{i,R_k} > \ell_{1,R_k}) \geq \Pr(\ell_{\lfloor |S_k|/2 \rfloor, R_k} > \ell_{1,R_k}) \geq 1 - \delta$.

Consequently, the probability that the UQ-oblivious approach returns the optimal candidate is

$$\begin{aligned} P_{ob} &\geq \Pr\Big(\bigwedge_{i=0,\dots,\lceil \log_2 n \rceil - 1} (1 \in S_{k+1} | 1 \in S_k)\Big) \\ &\geq \prod_{k=0}^{\lceil \log_2 n \rceil - 1} (1-\delta)^{\lfloor \frac{|S_k|}{2} \rfloor} \\ &\geq 1 - n\delta. \end{aligned}$$

We show in the next Corollary that, for $c = \frac{n \cdot \delta}{2}$, the probability P obtained in Theorem 3.6 is no less than $1 - n\delta$. □

Corollary B.2. *For the threshold c in Theorem 3.6 and δ in Theorem 3.7, let $c = \frac{n \cdot \delta}{2}$ and $\beta^{-1}(\epsilon_2, \epsilon_3, \cdots, \epsilon_n, c) = \min\{T : \prod_{i=2}^{n}(1 - (\frac{f(T)}{\epsilon_i})^4) \geq 1 - c\}$. Then by Theorem 3.6 the UQ approach returns the best candidate with probability over $1 - n\delta$ if $B \simeq \gamma^{-1}(\frac{\nu_2 - \nu_1}{2}, \delta) \cdot n$.*

Proof. Let $T > \sqrt[4]{2} \cdot \gamma^{-1}(\frac{\nu_2-\nu_1}{2}, \delta)$, we have

$$\prod_{i=2}^{n}(1-(\frac{4f(T)^2}{(\nu_i-\nu_1)^2})^2) > (1-\frac{\delta}{2})^n \geq 1-\frac{n\delta}{2} = 1-c.$$

This shows us that $T > \beta^{-1}(\frac{\nu_2-\nu_1}{2}, ..., \frac{\nu_n-\nu_1}{2}, c)$. Consequently, according to Theorem 3.6, for $B = R > T \cdot n$, the UQ approach returns the best candidate with probability

$$(1-c)^2 \geq 1-2c = 1-n\delta.$$

The UQ-oblivious approach returns the optimal candidate with probability over $1-n\delta$ if the budget $B_{ob} > z_{ob}$. But the UQ approach achieves the guarantee with budget $B > \sqrt[4]{2} \cdot \gamma^{-1}(\frac{\nu_2-\nu_1}{2}, \delta) \cdot n$, which can be empirically substantially smaller than the budget required in Theorem 3.7.

□

B.3 Proof of Theorem 3.9

Proof. Let $R_k = \sum_{j=0}^{k} r_k$, namely, the total number of epochs allocated for each candidate in S_k. We can guarantee that, for the UQ-oblivious approach, the output candidate $\hat{i}_D$ satisfies

$$\begin{aligned}
\Pr(\nu_{\hat{i}_D} - \nu_1 > \epsilon) &= \Pr\Big(\min_{i \in S_{\lceil \log_2(n) \rceil}} \nu_i - \nu_1 > \epsilon\Big) \\
&= \Pr\Big(\sum_{k=0}^{\lceil \log_2(n) \rceil - 1} \min_{i \in S_{k+1}} \nu_i - \min_{i \in S_k} \nu_i > \epsilon\Big) \\
&\leq \Pr\Big(\sum_{k=0}^{\lceil \log_2(n) \rceil - 1} 2|\nu_i - \ell_{i,R_k}| + \min_{i \in S_{k+1}} \ell_{i,R_k} - \min_{i \in S_k} \ell_{i,R_k} > \epsilon\Big) \\
&= \Pr\Big(\sum_{k=0}^{\lceil \log_2(n) \rceil - 1} 2|\nu_i - \ell_{i,R_k}| > \epsilon\Big) \\
&\leq \frac{2\lceil \log_2(n) \rceil f\Big(\lfloor \frac{B}{n\lceil \log_2(n) \rceil} \rfloor\Big)^2}{\epsilon} = \frac{2n^2 \lceil \log_2(n) \rceil^3}{B^2 \epsilon}.
\end{aligned}$$

by inspecting how the approach eliminates candidates and plugging in an upper bound for $\Pr(2|\nu_i - \ell_{i,R_k}| > \epsilon)$ for all k in the last inequality. We can calculate the bound for the UQ-guided method in

a similar way:

$$
\begin{aligned}
\Pr(\nu_{\hat{i}} - \nu_1 > \epsilon) &= \Pr\Big(\min_{i \in S_{\lfloor \frac{B}{R} \rfloor}} \nu_i - \nu_1 > \epsilon \Big) \\
&= \Pr\Big(\sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} \min_{i \in S_{k+1}} \nu_i - \min_{i \in S_k} \nu_i > \epsilon \Big) \\
&\leq \Pr\Big(\sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} \min_{i \in S_{k+1}} (\nu_i - \ell_{i,R_{k+1}} + \ell_{i,R_{k+1}}) - \min_{i \in S_k} (\nu_i - \ell_{i,R_k} + \ell_{i,R_k}) > \epsilon \Big) \\
&\leq \Pr\Big(\sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} 2|\nu_i - \ell_{i,R_k}| + \min_{i \in S_{k+1}} \ell_{i,R_{k+1}} - \min_{i \in S_k} \ell_{i,R_k} > \epsilon \Big) \\
&\leq \Pr\Big(\sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} 2|\nu_i - \ell_{i,R_k}| + (\min_{i \in S_{k+1}} \ell_{i,R_{k+1}} - \min_{i \in S_{k+1}} \ell_{i,R_k}) + (\min_{i \in S_{k+1}} \ell_{i,R_k} - \min_{i \in S_k} \ell_{i,R_k}) > \epsilon \Big) \\
&\leq \Pr\Big(\sum_{k=0}^{\lfloor \frac{B}{R} \rfloor - 1} 2|\nu_i - \ell_{i,R_k}| > \epsilon \Big) \\
&\leq \frac{2\lfloor \frac{B}{R} \rfloor f(R)^2}{\epsilon} = \frac{2\lfloor \frac{B}{R} \rfloor}{R^2 \epsilon}.
\end{aligned}
$$

□

The smaller upperbound of the UQ-guided approach than that of the UQ-oblivious counterpart in Theorem 3.9.

A simple calculation reveals that

$$\frac{2\lfloor \frac{B}{R} \rfloor}{R^2 \epsilon} < \frac{2n^2 \lceil \log_2(n) \rceil^3}{B^2 \epsilon}$$

by diving the first term by the second:

$$\frac{2\lfloor \frac{B}{R} \rfloor}{R^2 \epsilon} \Big/ \frac{2n^2 \lceil \log_2(n) \rceil^3}{B^2 \epsilon} = \lfloor \frac{B}{R} \rfloor \frac{B^2}{R^2} \cdot \frac{1}{n \lceil \log_2(n) \rceil^3} < 1.$$

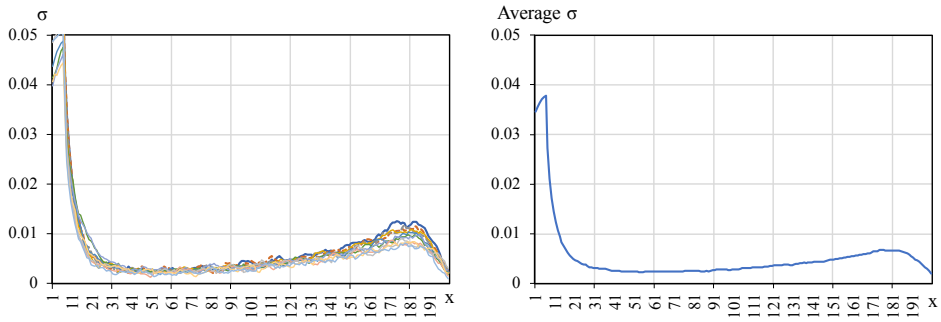
The last inequality holds because both $\frac{B}{R}$ and $n\lceil \log_2(n) \rceil$ are the number of rounds and are considered the same.

C Computational Details

Approximating $\Delta\sigma$ for Candidates. In our probabilistic model, $\ell(\mathbf{y}, M^*_{\gamma_c}(\mathbf{X}))$ is approximated by a Gaussian distribution parameterized by μ_c and σ_c. To compute the uncertainty reduction that would result from training each candidate for one additional epoch, we consider the uncertainty as a time series in the form of $(\sigma_c(t))_{t=1}^T$. Following Equation 1, we examine the uncertainty trendings in NAS-BENCH-201 [9]. Figure 5 shows a certain pattern of uncertainty behavior as t increases, both individually and aggregately, for different candidates.

We then model $(\sigma_c(t))_{t=1}^T$ according to different phases in the following way:

1. For $t \in (0, 6)$, $\sigma(t)$ increases. We use linear regression to fit the $\sigma(t)$, namely, $\sigma(t) = a_1 t + b_1$.



(a) Uncertainty scope for different candidates.(b) Average uncertainty for different candidates.

Figure 5: Landscape of the uncertainty scope for different epochs.

2. For $t \in (6, 50)$, $\sigma(t)$ drops quickly. We use the exponential model to fit $\sigma(t)$, namely, $\sigma(t) = a_1 e^{-b_1 x}$.

3. For $t \in (50, 180)$, $\sigma(t)$ increases steadily. We use another linear regression to fit $\sigma(t)$.

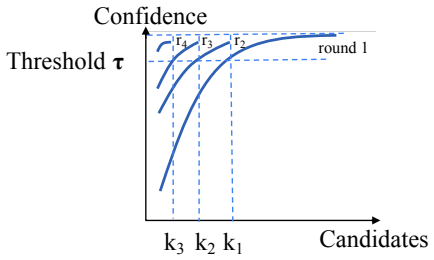
4. For $t \in (180, 200)$, $\sigma(t)$ reverses the trend and decreases again. We use linear regression.

For each launch, we sample a few candidates and train each one fully till convergence. We then use the abovementioned way to model the uncertainty behavior for the whole dataset. This makes the approximation for $\Delta_t\sigma = \sigma(t) - \sigma(t+1)$ effective and efficient.

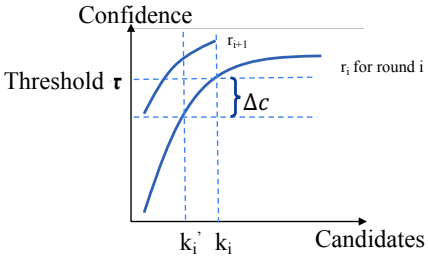
An alternative way to approximate $\Delta_t\sigma$ is to use

$$\frac{1}{\delta}\sum_{b\in D_{t-1}}(\ell(\mathbf{y}, M^b_{\gamma_c}(\mathbf{X})) - \mathbb{E}_{D_{t-1}}[\ell(\mathbf{y}, M^b_{\gamma_c}(\mathbf{X}))])^2 - \frac{1}{\delta}\sum_{b\in D_t}(\ell(\mathbf{y}, M^b_{\gamma_c}(\mathbf{X})) - \mathbb{E}_{D_t}[\ell(\mathbf{y}, M^b_{\gamma_c}(\mathbf{X}))])^2.$$

Building Probabilistic Model. At any given time t, the approximation of converged validation loss follows the Gaussian distribution: $\ell(\mathbf{y}, M^*_{\gamma_c}(\mathbf{X})) \sim \mathcal{N}(\mu_c, \sigma_c^2)$. In our experiments, $\delta = 10$, μ_c is the current accuracy $\ell(\mathbf{y}, M^t_{\gamma_c}(\mathbf{X}))$, and σ_c is the unbiased estimation of the standard deviation of $\ell(\mathbf{y}, M^i_{\gamma_c}(\mathbf{X}))^t_{i=t-10}$.



(a) Confidence curves in different rounds.



(b) Analysis of the choice of τ.

Figure 6: Illustration for *confidence curve* and *discarding mechanisms*. After obtaining the *confidence curve*, a threshold τ determines the number of candidates we will keep (k_1 for round 1, k_2 for round 2, and k_3 for round 3). We choose the smallest k such that $P_k \geq \tau$ for each round, proceed training with the best performed k candidates, and discard the rest configurations.

D Benchmark and Dataset Information

Table 2 consolidates information on the datasets, hyperparameters, fidelity, and dataset sizes for Nas-Bench-201 and LCBench. The datasets for LCBench are drawn from various sources openml, open.

Table 2: Benchmark and Dataset information.

Tasks	Datasets	Hyperparameters	Fidelity	# Training set	# Validation set	# Test set
Nas-Bench-201	CIFAR-10 CIFAR-100 ImageNet-16-120	$1 \leftarrow 0$ $2 \leftarrow \{0, 1\}^*$ $3 \leftarrow \{0, 1, 2\}^*$ Range: {none, skip_connect, nor_conv_1x1, nor_conv_3x3, avg_pool_3x3}	1-200	25K images 50K images 151.7K images	25K images 5K images 3K images	10K images 5K images 3K images
LCBench	Fashion-MNIST adult higgs jasmine vehicle volkert	Batch size: [16, 512], log-scale Learning rate: [$1e^{-4}$, $1e^{-1}$], log-scale Momentum: [0.1, 0.99] Weight decay: [$1e^{-5}$, $1e^{-1}$] Number of layers: [1, 5] Maximum number of units per layer: [64, 1024], log-scale Dropout: [0.0, 1.0]	1-50	"Whenever possible, we use the given test split with a 33% test split and additionally use fixed 33% of the training data as validation split. In case there is no such OpenML task with a 33% split available for a dataset, we create a 33% test split and fix it across the configurations." zimmer2021auto		

E More Results on Experiments

We include more results on NAS-Bench-201 and LCBench. For example, Figure 11 and 12 show the results on LCBench, where we proved the consistently better performance of the UQ-guided approaches than the UQ-oblivious methods on Fashion-MNIST. Figure 11 shows the results of the validation loss while Figure 12 demonstrates the results of regret. UQ-guided approaches obtained an average of over 50% improvement over the UQ-oblivious counterparts.

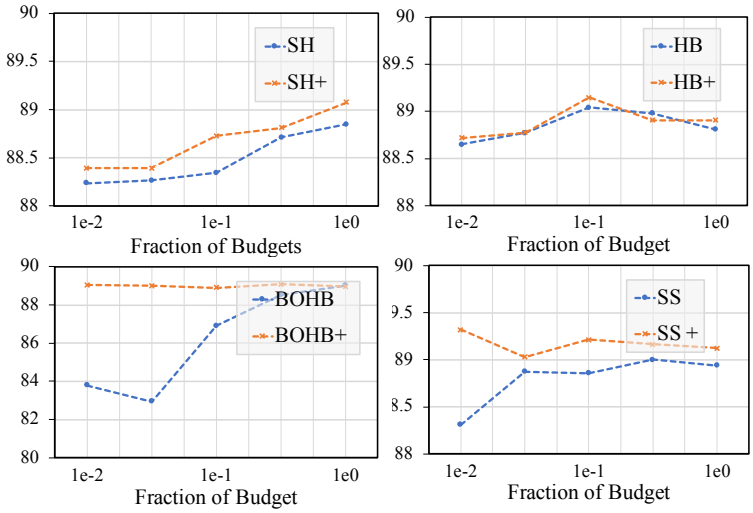


Figure 7: Results of test accuracy when optimizing on CIFAR-10.

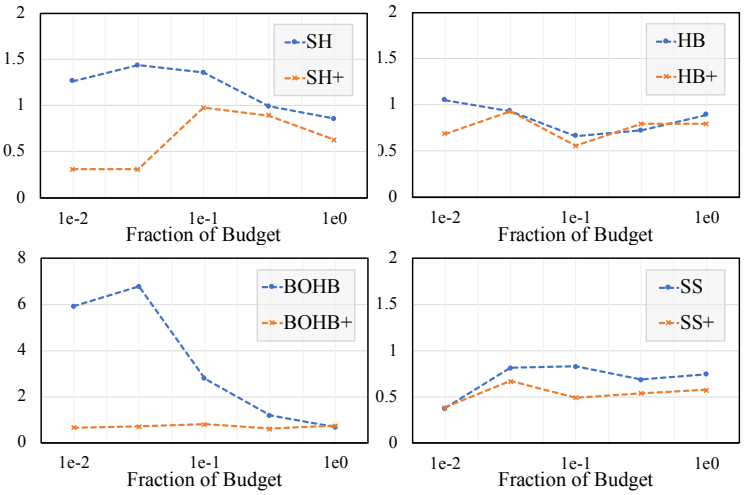


Figure 8: Results of regret (%) when optimizing on CIFAR-10.

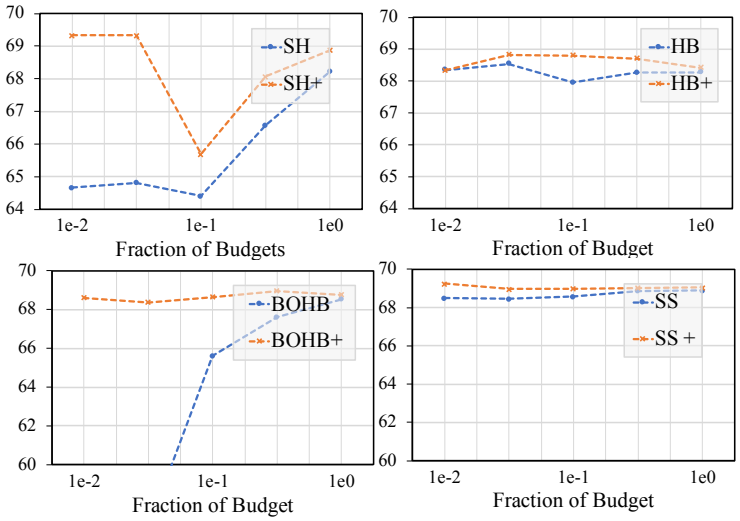


Figure 9: Results of test accuracy when optimizing on CIFAR-100.

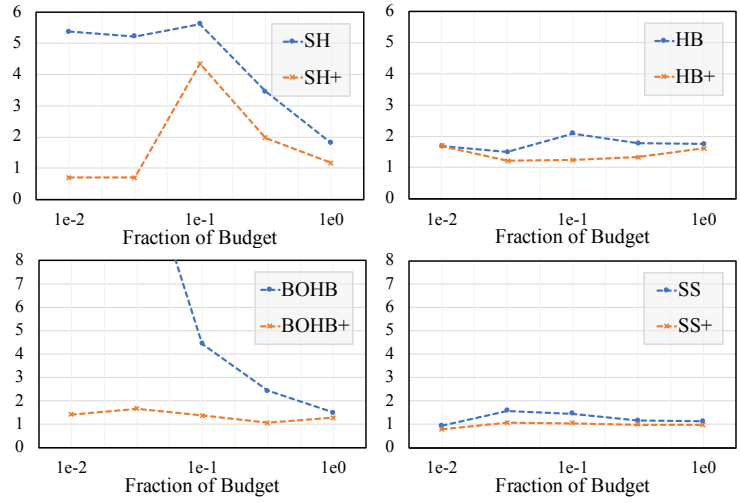


Figure 10: Results of test accuracy when optimizing on CIFAR-100.

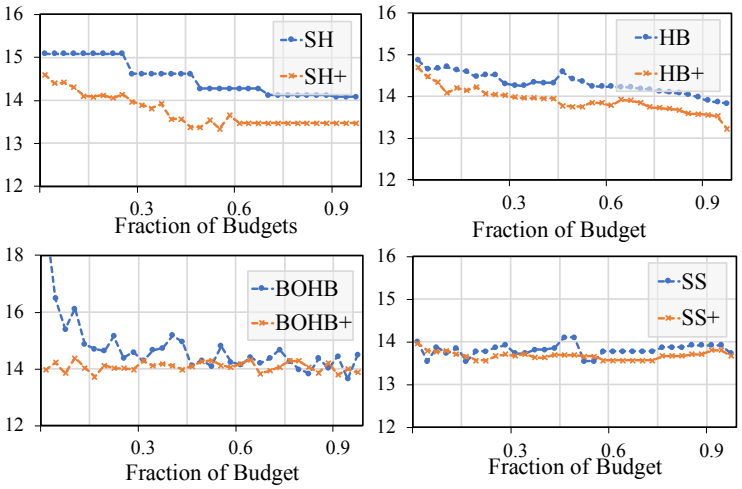


Figure 11: Results of validation error for optimizing on Fashion-MNIST.

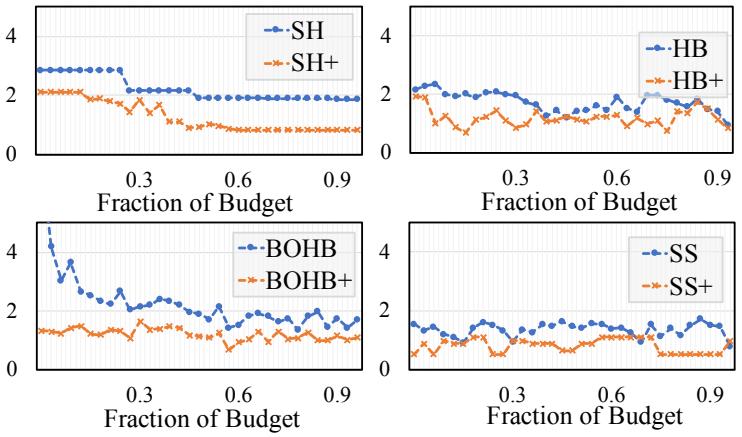


Figure 12: Results of regret (%) on test accuracy for optimizing on Fashion-MNIST.