Best Arm Identification with Retroactively Increased Sampling Budget for More Resource-Efficient HPO

Jasmin Brandt[1], **Marcel Wever**[2,3], **Viktor Bengs**[2,3] and **Eyke Hüllermeier**[2,3]
[1]Paderborn University, Germany
[2]LMU Munich, Germany
[3]Munich Center for Machine Learning, Germany
jasmin.brandt@upb.de, {viktor.bengs, marcel.wever, eyke}@ifi.lmu.de

Abstract

Hyperparameter optimization (HPO) is indispensable for achieving optimal performance in machine learning tasks. A popular class of methods in this regard is based on Successive Halving (SHA), which casts HPO into a pure-exploration multi-armed bandit setting under finite sampling budget constraints. This is accomplished by considering hyperparameter configurations as arms and rewards as the negative validation losses. While enjoying theoretical guarantees as well as working well in practice, SHA has several hyperparameters itself, one of which is the maximum budget that can be allocated to evaluate a single arm (hyperparameter configuration). Although there are already solutions to this meta hyperparameter optimization problem, such as the doubling trick or asynchronous extensions of SHA, these are either practically inefficient or lack theoretical guarantees. In this paper, we propose incremental SHA (iSHA), a synchronous extension of SHA, allowing to increase the maximum budget a posteriori while still enjoying theoretical guarantees. Our empirical analysis of HPO problems corroborates our theoretical findings and shows that iSHA performs more reliably than existing SHA-based approaches.

1 Introduction

Hyperparameter optimization (HPO) is a crucial step in the process of engineering machine learning (ML) applications, as optimal performance can only be obtained if parameterized ML algorithms are tuned to the task at hand [Feurer and Hutter, 2019; Bischl *et al.*, 2023]. Such a task is specified in the form of a dataset $\mathcal{D}$ and a loss function ℓ. Typically, HPO is carried out in a trial-and-error fashion by evaluating ℓ on the given data $\mathcal{D}$ for various candidate hyperparameter configurations (HPCs).

In the early days of HPO, grid search and random search [Bergstra *et al.*, 2011] have been the main tools. However, they can be criticized for their disability in finding an optimal hyperparameter configuration as well as their computational cost. In the age of deep learning, a highly efficient HPO method is inevitable, as evaluating hundreds or even thousands of configurations is prohibitive. To address this challenge, several HPO methods have been proposed to improve sampling or evaluation efficiency. For the former, the methods mainly focus on Bayesian Optimization [Hutter *et al.*, 2011], whereas for the latter, the HPO problem is extended by a budget parameter. Using this parameter, the optimizer can specify for which budget a hyperparameter configuration should be evaluated. This area of the HPO literature is also referred to as multi-fidelity optimization.

Probably the simplest procedure in this area is the successive halving algorithm (SHA) [Karnin *et al.*, 2013], which is rooted in the multi-armed bandit (MAB) literature [Lattimore and Szepesvári, 2020]. It first evaluates a set of candidate hyperparameter configurations (arms) for a minimum starting budget R_0, discards the worse half, and continues evaluation with the better half for a doubled budget. This procedure is repeated until a maximum budget of R is reached. By concentrating the budget on more promising hyperparameter configurations, the reliability of the evaluations is gradually increased, but also their evaluation costs. In contrast, less promising solutions are discarded early on with little budget.

Following the terminology of the multi-armed bandits, SHA attempts to solve a best arm identification problem for a given fixed sampling budget. [Jamieson and Talwalkar, 2016b] as well as [Li *et al.*, 2018] have derived theoretical bounds on the necessary sampling budget to guarantee to find an optimal or near-optimal arm (hyperparameter configuration) with high probability. However, these theoretical bounds have two problems regarding practical application: Firstly, they depend on problem parameters that are unknown in practice, and secondly, they are derived for worst-case scenarios and are therefore often too conservative. Accordingly, the common approach is to set a budget in an ad-hoc manner and later check whether it was sufficiently large to support the reliability of the returned HPC. If this is not the case, the budget is increased retrospectively, which is problematic as SHA is not incremental by design, so that the entire algorithm must be re-run. This is not only costly but also comes with a loss of valuable knowledge already accumulated. Needless to say, from an ecological perspective, this is undesired either, as the computational resources, as well as the consumed energy for optimizing the hyperparameters for the lower maximum budget, is essentially wasted [Tornede *et al.*, 2023].

Although there are two variants of SHA that do not need to specify the maximum budget R in advance, these have the decisive disadvantage that they come without theoretical guarantees. Asynchronous SHA (ASHA) [Li *et al.*, 2020] is the first variant, in which decisions about candidate evaluations for larger budgets are made asynchronously, allowing for higher parallelization. This variant has recently been further developed into PASHA [Bohdal *et al.*, 2023], which progressively increases the budget if the ranking of the configurations in the top two high-fidelity sets has not stabilized. However, asynchronous decision-making comes at the risk of mistakenly promoting HPCs to the next budget level. While [Li *et al.*, 2020] invoke the law of large numbers to argue that this is not an issue, the problem remains in the case of finite budget constraints, where only a limited number of hyperparameter configurations can be considered.

Contributions. We will focus on the HPO application for the most part when presenting the necessary concepts and our results. However, our theoretical results apply to the more general bandit setting and can therefore be carried over to applications other than HPO. Following the terminology of the MAB literature, we are considering the best arm identification (BAI) problem for which the sampling budget is increased retroactively. Our contributions can be summarized as follows:

- We provide the first theoretical results for ASHA, analyzing its capabilities in setups with constraints on the overall budget. These findings are accompanied by empirical evidence for a set of HPO benchmarks.
- We propose an incremental extension of SHA (iSHA) that still allows one to increase the maximum allocatable budget R retrospectively in a synchronous manner.
- A theoretical and empirical analysis of iSHA is provided, finding iSHA to be theoretically sound relative to the original SHA, while being provably more resource-efficient.
- In an extensive empirical study, we compare iSHA to the original SHA, and PASHA embedded into the Hyperband framework. We find iSHA to give more robust results compared to PASHA, often yielding higher quality hyperparameter configurations, while being more resource-efficient than SHA.
- We show a long-missing lower bound on the necessary budget for finding a nearly-optimal arm under common assumptions of the non-stochastic BAI problem.

2 (Near-)Optimal Arm Identification

The best arm identification problem in multi-armed bandits (MABs) is a sequential decision-making problem in which an agent has to choose in each time step $t \in \{1, \dots, B\}$ within a fixed budget $B \in \mathbb{N} \cup \{\infty\}$ one out of $n \in \mathbb{N}$ possible options that we denote by their indices $[n] := \{1, \dots, n\}$ and call arms in the following. After choosing (or pulling) one arm, the agent directly observes a loss $\ell(i)$ for the chosen arm $i \in [n]$ given by a loss function $\ell : [n] \to \mathbb{R}$. The observed loss after r pulls of arm $i \in [n]$ will be denoted $\ell_r(i)$. In the non-stochastic setting, which is the setting we consider, the observed losses are not necessarily governed by an underlying stochastic distribution. Instead, a common assumption is that the sequence of losses of an arm converges asymptotically to a fixed final value [Jamieson and Talwalkar, 2016b; Li *et al.*, 2018; Brandt *et al.*, 2023].

Assumption 2.1. $\forall i \in [n]$ and for $R \in \mathbb{N} \cup \{\infty\}$ the loss function converges against a limit value $\nu_i = \lim_{r \to R} \ell_r(i)$.

Note, that the stochastic scenario in which $(\ell_k(i))_{k \geq 1}$ for each $i \in [n]$ is an i.i.d. sample from a stochastic distribution with $\mathbb{E}[\ell_k(i)] = \nu_i$ can be treated as a special case of our setting [Jamieson and Talwalkar, 2016b].

Goal. Usually, the goal in best arm identification is to find the arm with the smallest loss $i^* \in \arg\min_{i \in [n]} \nu_i$. We will relax this goal to only identify a near-optimal arm where "near-optimal" is defined in the following way.

Definition 2.2. Let $\epsilon > 0$, then we call $\hat{i} \in [n]$ an ϵ-optimal arm if $\nu_{\hat{i}} \leq \nu_{i^*} + \epsilon$.

It is straightforward to extend the scenario to the case in which we deal with a countably infinite set of stochastic bandit arms indexed by $i = 1, 2, \dots$

Hyperparameter Optimization. Hyperparameter optimization (HPO) deals with the problem of finding a suitable hyperparameter configuration $\boldsymbol{\lambda}$ of an ML algorithm $\mathcal{A}$ with a corresponding hyperparameter space Λ for a given learning task (e.g. image classification, regression analysis, etc.). For a suitable loss function ℓ and a finite set of HPC samples, say $\tilde{\Lambda}$, from the possibly uncountable infinite space Λ, we can consider each HPO problem as a MAB problem, by simply considering the HPCs as arms. An example of a loss function is the validation error of a (supervised) learning algorithm $\mathcal{A}$ with parameterization $\boldsymbol{\lambda}$ and resource allocation r, which could be for instance the wall-clock time, number of used data points, etc. Sampling or generating a finite set of HPCs $\tilde{\Lambda}$ can be done in different ways, for example, simple uniform sampling from Λ [Li *et al.*, 2018] or by leveraging a Bayesian search mechanism [Falkner *et al.*, 2018].

3 Successive Halving and Hyperband

The successive halving algorithm (SHA) by [Karnin *et al.*, 2013] is applicable for the non-stochastic best arm identification problem with a finite set of arms under a fixed budget constraint. By sampling n hyperparameter configurations (HPCs) uniformly at random, it has already been applied successfully to HPO by [Jamieson and Talwalkar, 2016a]. Starting from a minimum budget R_0 for which all the n available arms are evaluated, it iteratively discards the worse half and continues to evaluate the remaining arms with double the budget. This procedure is repeated until either only a single arm is left or a maximum allocatable budget R is reached (maximum for a single arm). Typically, the number of arms n is chosen such that at least one candidate reaches the final iteration of the algorithm. A budget level for which arms are evaluated is also referred to as *rung* in the following. Furthermore, we write that an arm is *promoted* to the next rung if it was not discarded and thus considered in the next iteration of SHA. While SHA allows allocating exponentially more budget on the more promising arms, its final performance crucially depends on its parameterization. The parameters n, R

and R_0 need to be chosen with care and depending on the task. With regard to HPO, starting with a too low initial budget R_0, we face the problem of rejecting actually promising arms (HPCs) too early, namely those that require more budget, e.g., more data or more training iterations, to perform well enough to remain in the set of promising candidates.

The Hyperband (HB) algorithm [Li *et al.*, 2018] comes with a heuristic of how to choose different values for n and R_0, and subsequently uses SHA as a subroutine. Even if it can generally be used for a bandit problem with an infinite number of arms, its design is tailored to HPO. HB allows different allocation strategies to be considered for the tradeoff between (i) considering many arms (HPCs) n starting with a rather small R_0, and (ii) giving some arms (HPCs) more budget from the beginning. The latter is motivated by the fact that in machine learning, some HPCs may require a larger amount of resources to show off their better performance. We refer to each call of SHA as a *bracket* [Li *et al.*, 2018], for which the set of arms (HPCs) is sampled uniformly at random and given to SHA as an input.

4 Related Work

The pure-exploration and best arm identification problem in MABs has been studied intensively with a stochastic feedback mechanism (see [Gong and Sellke, 2023] for a more recent overview). Especially in the case of a fixed budget, there is some work in this direction [Carpentier and Valko, 2015; Abbasi-Yadkori *et al.*, 2018; Shen, 2019; Azizi *et al.*, 2022; Kato *et al.*, 2022]. However, the non-stochastic setting, as considered in our work, has so far only been investigated in [Jamieson and Talwalkar, 2016a; Li *et al.*, 2018] and [Brandt *et al.*, 2022]. The first two with a special focus on HPO similar to our work and the latter with a focus on algorithm selection/configuration [Rice, 1976].

Considering HPO as a black-box optimization problem, various methods can be used to tackle this problem [Feurer and Hutter, 2019; Bischl *et al.*, 2023]. Grid search and random search are rather straightforward solutions. However, both are rather expensive, and thus, methods emerged to improve sample efficiency and evaluation efficiency. While the former methods are mostly centered around Bayesian optimization [Frazier, 2018; Hutter *et al.*, 2011], the latter emerged in the branch of multi-fidelity optimization.

In multi-fidelity optimization, the goal is to distribute the budget for evaluating HPCs in a way that more budget is concentrated on the more promising HPCs and less so on inferior candidates. The successive halving algorithm (SHA), initially proposed by [Karnin *et al.*, 2013] and later used by [Jamieson and Talwalkar, 2016b; Jamieson and Talwalkar, 2016a] for HPO, devises a powerful HPO method, which has been incorporated as a subroutine in the well-known HPO method Hyperband [Li *et al.*, 2018]. Hyperband has been extended in various directions such as improving its sampling efficiency [Falkner *et al.*, 2018; Awad *et al.*, 2021; Mallik *et al.*, 2023] and introducing shortcuts in the evaluation process [Mendes *et al.*, 2021].

But also SHA has been subject to improvements. [Li *et al.*, 2020] extend SHA to asynchronous SHA (ASHA), which helps to better leverage parallel computing resources by promoting candidates asynchronously to the next rung. Simultaneously, the maximum budget R can be adapted on-the-fly. Progressive ASHA (PASHA) proposed by [Bohdal *et al.*, 2023] builds on ASHA and incorporates a mechanism to only introduce higher rungs where necessary. While both ASHA and PASHA have been extensively studied empirically, a thorough (theoretical) analysis of the costs of the asynchronous promotion scheme is still lacking. Also, these empirical studies have considered comparably large setups with vast amounts of resources. In our study, we consider small-scale setups and analyze the behavior of ASHA and PASHA in that scope.

Algorithm 1 Incremental Successive-Halving Algorithm (iSHA)

Input: S initial set of HPCs, r, maximum resource R, reduction factor η, $(C_k)_k$ sequence of HPCs promoted in previous run, $(L_k)_k$ old sequence of losses
Initialize: $S_0 \leftarrow S$, $\tilde{n} = |C_0|$, $n = |S_0| + |C_0|$, $s = \log_\eta(R)$

for $k \in \{0, 1, \ldots, s\}$ **do**
  $n_k = \lfloor n/\eta^k \rfloor - \lfloor \tilde{n}/\eta^k \rfloor$, $r_k = r\eta^k$
  pull each arm in S_k for r_k times
  if $k \leq s - 1$ **then**
    $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor - \lfloor \tilde{n}/\eta^{k+1} \rfloor$ arms from $S_k \cup C_k \backslash C_{k+1}$
  else
    $S_{k+1} \leftarrow$ keep the best $\lfloor n/\eta^{k+1} \rfloor$ arms from $S_k \cup C_k$
  end if
end for
Output: Remaining configuration

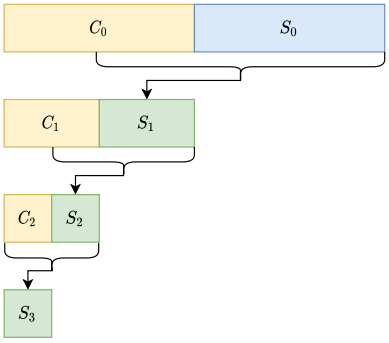

Figure 1: Illustration of how iSHA continues a previously conducted SHA run.

5 Incremental Successive Halving

Due to the static budget setting in SHA, the execution of SHA cannot simply be continued for an adapted parameterization, e.g., a higher maximum allocatable budget R. By rerunning SHA from scratch, however, knowledge about previously evaluated hyperparameter configurations (HPCs) is discarded and resources already allocated are wasted. As another extreme, ASHA and PASHA allow to dynamically increase the maximum allocatable budget R, devising a scheme

for asynchronous promotions to higher rungs. However, as we show in Sections 6 and 7.2, the asynchronous promotions in ASHA and PASHA can be erroneous and thus impede the identification of the optimal hyperparameter configurations.

With incremental successive halving (iSHA), we propose a middle ground for budget-constrained scenarios, i.e., scenarios in which we cannot rely on the law of large numbers as required by [Li *et al.*, 2020]. Similar to ASHA and PASHA, we allow the maximum allocatable budget to be increased after an SHA run, making SHA in principle stateful. Algorithm 1 translates this into pseudocode. Differences from the original SHA are highlighted in blue. While Algorithm 1 also covers the case of ASHA, adding a single configuration at a time, we assume $|S| + |C_0| = |C_0| \cdot \eta$ for our theoretical and empirical analysis, where C_0 are the configurations in the beginning of the previous run, S are the new configurations and $\eta \geq 1$ is a reduction factor.

In Figure 1 we see the mechanism underlying iSHA to continue a previously conducted run of SHA that resulted in the rungs C_0, C_1 and C_2. The initially sampled set of HPCs C_0 is padded with newly sampled HPCs S_0 to initially achieve the same number of HPCs as if SHA had been restarted. However, only the new configurations are executed (following the typical SHA budget allocation) and finally compared with the previous configurations from C_0. The already promoted configurations in C_1 from the previous SHA run will remain and only the required number of configurations will be promoted, i.e., S_1, such that the size of the union of C_1 and S_1 matches the size of the second rung if SHA had been restarted. This mechanism is then iteratively continued for subsequent rungs.

Intuitively speaking, the strategy of iSHA is to continue a previous SHA run in the most efficient, and thus, resource-saving way. However, similarly to ASHA and PASHA, this efficiency may come at the cost of a potential drop in performance, as previously made decisions cannot be revoked. More specifically, in the worst case, all promotions of the previous run would not have occurred if we had known the complete set of candidate HPCs from the start. Only filling up the rungs leaves less space for the desired candidates to be promoted to the highest rung.

Nevertheless, we prove in the next section that we are still able to identify near-optimal solutions with a high probability, which will be confirmed by empirical results in Section 7.2 later on. Furthermore, we demonstrate that this robustness gives iSHA an edge over ASHA and PASHA when it comes to the quality of returned hyperparameter configurations in settings with limited budget.

6 Theoretical Results

We split the theoretical results into four parts. First, we present a lower bound for the necessary budget for the ϵ-optimal arm identification problem. Second, we provide a theoretical analysis of ASHA. Third, we give some theoretical guarantees for iSHA, our extension of SHA, and fourth, we extend these guarantees to an incremental extension of Hyperband.

As a prerequisite for the theoretical results, we ease the notation by simply writing $\ell_{i,t}$ instead of $\ell_t(i)$. Since by assumption 2.1 there exists a limit value of the loss function for every arm, we denote the corresponding convergence speed by $\gamma(t) \geq \sup_i |\ell_{i,t} - \nu_i|$, $\forall t \in \mathbb{N}$.

6.1 Lower Bound

Although there is already literature tackling the problem of ϵ-optimal arm identification in MABs like [Li *et al.*, 2018], a lower bound for the necessary budget was missing until now.

Theorem 6.1 (Lower bound for $\epsilon/2$-optimal arm identification)**.** *If an algorithm Alg correctly identifies an $\epsilon/2$-optimal arm for any loss function ℓ, then there exist slightly modified limits $\{\tilde{\nu}_i\}_{i\in[n]}$ with $|\nu_i - \tilde{\nu}_i| \leq \epsilon/2$ for each $i \in \{1, \dots, n\}$ such that Alg needs at least $n \cdot \gamma^{-1}\left(\max\left\{\frac{\nu_n - \nu_1}{2}, \frac{\epsilon}{4}\right\}\right)$ total pulls of arms in expectation.*

The proof is deferred to Section D.

6.2 Theoretical Analysis of ASHA

We now analyze ASHA [Li *et al.*, 2020], which, to the best of our knowledge, is the only algorithm with a similar goal of more efficient resource use as our proposed iSHA.

Theorem 6.2 (Necessary Budget for ASHA)**.** *Fix n arms and assume $\nu_1 \leq \ldots \leq \nu_n$. Let*

$$z_{\mathrm{ASHA}} = n(\lfloor \log_\eta(n) \rfloor + 1) \cdot \max\Big\{ \max_{k\in[K]} \eta^{-k} \cdot \gamma^{-1}\Big(\tfrac{\nu_{\lfloor rung_{k-1}|/\eta\rfloor+1} - \nu_1}{2}\Big), \eta^{-K} \max_{i\in rung_K\setminus\{1\}} \gamma^{-1}\Big(\tfrac{\nu_i - \nu_1}{2}\Big)\Big\},$$

where $K \leq \lfloor \log_\eta(n) \rfloor$ is the top rung of ASHA. If ASHA's total number of pulls exceeds z_{ASHA}, then the best arm is returned.

The dependence is linear-logarithmic in n, and the limit gap from the best arm to the other arms occurs in the inverted convergence rate γ^{-1}. The first term in the maximum makes sure that the best arm reaches the top rung K, while the second term makes sure that the best arm will eventually be returned. In view of the lower bound in Theorem 6.1, ASHA is nearly optimal.

As a corollary of Theorem 6.2 (see the proof in Section C), we obtain the following result regarding the mechanism of the sampling process pursued by ASHA.

Corollary 6.3 (Worst Case Promotion Costs)**.** *Assume all rungs to be full, i.e., no promotion is possible, and the top rung K only contains the current incumbent arm. If at that time a new best arm (HPC) $\hat{i}$ is sampled, then promoting $\hat{i}$ to the sole solution of the new top rung $K+1$ requires the sampling of $\eta^K - 1$ additional arms (HPCs) and a total of η^{K+1} many jobs.*

From these results, we can draw two major conclusions. The more arms (HPCs) have already been considered in ASHA when $\hat{i}$ enters the pool of considered hyperparameter configurations, i.e., the later in the process, the more budget needs to be spent to promote $\hat{i}$ to the top rung. Particularly, in a scenario with a limited budget, e.g., limited by the overall budget (total number of pulls) or by the number of arms

to be sampled, ASHA fails to return $\hat{i}$, if the required budget for promoting the best configuration exceeds the remaining budget. A similar result can be shown for PASHA, since its sampling mechanism is similar to ASHA.

6.3 Theoretical Analysis of iSHA

For iSHA (Algorithm 1), we first prove a lower bound on the necessary budget to return a nearly optimal arm (configuration), when $n/\tilde{n} = \eta$ which corresponds to $|S| + |C_0| = |C_0| \cdot \eta$. The proof is given in Appendix B.1.

Theorem 6.4 (Necessary Budget for iSHA)**.** *Fix n arms from which $\tilde{n}$ arms were already considered in a previous run, and assume $\nu_1 \leq \cdots \leq \nu_n$ as well as $r \in (R/\eta^s)_{s=0,\ldots,\log_\eta(R)}$. For any $\epsilon > 0$ let*

$$z_{\text{iSHA}} = \eta \lceil \log_\eta(n) \rceil \cdot \max_{i=2,\ldots,n} i\Big(1 + \min\Big\{R, \gamma^{-1}\Big(\max\Big\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\Big\}\Big)\Big\}\Big).$$

If iSHA's total number of pulls exceeds z_{iSHA}, then an arm $\hat{\imath}$ is returned that satisfies $\nu_{\hat{\imath}} - \nu_1 \leq \epsilon/2$.

As the dependency on n and the gap is similar as for ASHA, we conclude from Theorem 6.1 that iSHA is nearly optimal as well. Further, we can specify the improvement of iSHA over the costly re-run of SHA.

Theorem 6.5 (Improvement of number of pulls of iSHA in comparison to SHA)**.** *Fix a maximal budget per arm of R, r and η. Assume that we have already run SHA on $\tilde{n}$ arms with R, r, and η. Now sample $n - \tilde{n}$ new arms with $n = \tilde{n}\eta$, and (re-)run SHA and iSHA over s rounds with the above variables. Then, for $\eta_- = \eta - 1$ and $s^+ = s + 1$ we have*

$$\frac{\#\{\textit{total pulls of iSHA}\}}{\#\{\textit{total pulls of SH}\}} \leq 1 - \frac{(s^+)(\tilde{n}R + \eta^s)(\eta_-) - (\eta^{s^+} - 1)(2R + n)}{(s^+)(nR + \eta^s)(\eta_-) - (\eta^{s^+} - 1)(R + n)}.$$

Again, as a corollary of Theorem 6.5 (see Section Appendix B.2), we obtain the following result regarding the "limit case", i.e., if we would increase the maximum size R infinitely often, or, equivalently, the number of possible rungs s infinitely often.

Corollary 6.6. *If we run iSHA and SHA infinitely often with*

(i) an ever-increasing maximum size R, and

(ii) such that the newly sampled number of configurations in each new run of iSHA fulfills $|S| + |C_0| = |C_0| \cdot x$, where C_0 is the number of configurations in the previous run and $x > 1$,

then the ratio of total pulls of iSHA and total pulls of SHA converges to $1 - x^{-1}$.

In our setting, we use $x = \eta$, so that the improvement ratio is $1 - \eta^{-1}$. Note that a comparison similar to Theorem 6.5 or Corollary 6.6 is difficult to make for ASHA or PASHA, since both do not include the parameter R. Finally it is worth mentioning that no similar statement as Corollary 6.3 holds for iSHA, since $\hat{i}$ can be still (and very likely will be) returned as an output for the available budget.

6.4 Incremental Hyperband

Like the original version of SHA and its extensions ASHA and PASHA, we can also employ iSHA as a subroutine in Hyperband. To this end, Hyperband needs to be made incremental itself, as done in Algorithm 4 in the appendix, which we call incremental Hyperband (iHB). In the following, we provide a theoretical analysis of this incremental version of Hyperband with iSHA as a subroutine. Figure 3 in the appendix illustrates how every Hyperband bracket is updated after increasing the maximum budget R.

Recall, that for $\epsilon > 0$, we aim to find an *ϵ-optimal configuration* $\hat{\boldsymbol{\lambda}}$ which was defined as $\nu_{\hat{\boldsymbol{\lambda}}} - \nu_{\boldsymbol{\lambda}^*} \leq \epsilon$. for $\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \Lambda} \nu_{\boldsymbol{\lambda}}$. To ensure that the search is possible by sampling merely a finite subset of HPCs, we make the following assumption similar to [Brandt *et al.*, 2023]:

Assumption 6.7. The proportion of ϵ-optimal configurations in Λ is $\alpha \in (0, 1)$.

Note that we now have at least one ϵ-optimal configuration in a sampled set of configurations with probability at least $1 - \delta$, if the sample size is at least $\lceil \log_{1-\alpha}(\delta) \rceil$ for a fixed failure probability $\delta \in (0, 1)$. With this, we can state the following theorem, the proof of which is given in Appendix B.3.

Theorem 6.8. *Let η, R, α and δ be fixed such that*

$$R \geq \max\Big\{ \lceil \log_{1-\alpha}(\delta) \rceil (\eta_-) + 1, \eta \bar{\gamma}^{-1}\Big(L_{\eta, L_{\eta,R}} + 4 + \tfrac{\lfloor L_{\eta,R} \rfloor}{2} - \tfrac{\sum_{k=1}^{\lfloor L_{\eta,R} \rfloor + 1} \log_\eta(k)}{\lfloor L_{\eta,R} \rfloor + 1}\Big)\Big\}$$

for $\bar{\gamma}^{-1} := \max_{s=0,\ldots,\lfloor L_{\eta,R} \rfloor} \max_{i=2,\ldots,n_s} i\Big(1 + \min\Big\{R, \gamma^{-1}\Big(\max\Big\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\Big\}\Big)\Big\}\Big)$

and $L_{\eta,R} = \log_\eta(R)$, then iHB finds an ϵ-optimal configuration with probability at least $1 - \delta$.

To conclude, despite the incremental extension of Hyperband, we can maintain the theoretical guarantees of the original Hyperband. Although promotions in iSHA are also to some extent performed asynchronously, we can still identify a nearly best arm when doing promotions in a batch, provided a sufficiently large batch size.

7 Empirical Evaluation

In addition to the theoretical results of the previous section, we evaluate iSHA empirically and compare it to PASHA [Bohdal *et al.*, 2023] and SHA [Jamieson and Talwalkar, 2016a]. We are especially interested in the following two research questions:

RQ1 Is iSHA able to retain the quality of returned HPCs as compared to applying SHA from scratch?

RQ2 How does the proposed iSHA compare to the state-of-the-art algorithms ASHA and PASHA?

7.1 Experiment Setup

In our experimental evaluation, we compare iSHA to two asynchronous extensions of SHA, namely ASHA and PASHA. For the comparison, we integrate all SHA variants as subroutines in Hyperband to answer the research questions **RQ1** and **RQ2**. To this end, we conduct extensive experiments tackling numerous HPO tasks, considering various

Benchmark	Model	# Inst.	Objective	Fidelity
lcbench	neural network	34	val_accuracy	epochs
rbv2_svm	SVM	106	acc	fraction
rbv2_ranger	random forest	119	acc	fraction
rbv2_xgboost	XGBoost	119	acc	fraction
nb301	neural network	1	val_accuracy	epochs

Table 1: List of considered benchmarks from YAHPO-Gym with the type of learner, number of considered datasets, objective function, and the type of budget that can be used as a fidelity parameter.

Benchmark / ϵ	0.001	0.005	0.01	0.05
lcbench	0.0004±0.0011	0.0042±0.0147	0.0095±0.0284	0.0919±0.1599
nb301	0.0001±0.0000	0.0001±0.0000	0.0078±0.0000	0.8962±0.0000
rbv2_svm	0.0092±0.0397	0.0554±0.1296	0.126±0.1995	0.4927±0.2995
rbv2_ranger	0.0026±0.0095	0.0365±0.1323	0.0732±0.1967	0.5336±0.3778
rbv2_xgboost	0.0123±0.0308	0.0213±0.0560	0.0316±0.0845	0.1541±0.2306

Table 2: Mean (± standard deviation) proportion of 10,000 randomly sampled hyperparameter configurations that are within an ϵ distance of the best hyperparameter configuration's performance.

types of learners and two different fidelity parameters: the number of *epochs* and the *fraction* of the training data used for fitting a model.

As a benchmark library, we use YAHPO Gym [Pfisterer *et al.*, 2022], which provides fast-to-evaluate surrogate benchmarks for HPO with particular support for multi-fidelity optimization, rendering it a perfect fit for our study. From YAHPO Gym, we select the benchmarks listed in Table 1. All the benchmarks consist of several datasets, which are referred to as benchmark instances, allowing for a broad comparison. Due to space limitations, we only present a summary of the results here, whereas detailed results can be found in Appendix E.

In Table 2, we show the mean fraction of 10,000 randomly sampled hyperparameter configurations to be at most ϵ worse than the best hyperparameter configuration. As can be seen, the considered benchmarks are of varying difficulty and the size of the ϵ-optimal fraction also varies substantially in size across the datasets contained in the corresponding benchmark suites as indicated by the standard deviation.

Furthermore, we set the initial max size $R_{t-1} = 16$ and increase it after the first run by a factor of η to $R_t = \eta R_{t-1}$, as this is a budget that is supported by all benchmark scenarios. Since ASHA and PASHA automatically increase the maximum budget depending on the observed performances, we only ensure an upper limit of R_t for both to ensure a fair comparison. As a termination criterion, we use that the number of HPCs would exceed the pool size of the Hyperband bracket. For benchmarks considering a fraction of the training dataset as a fidelity parameter, we translate a budget r by r/R_t into a fraction between 0 and 1.

Furthermore, we repeat each combination of algorithm, η, and benchmark instance for 30 seeds, resulting in a total amount of $30 \times 3 \times 2 \times 379 = 68,220$ hyperparameter optimization runs. We run all experiments on a single workstation equipped with 2xIntel Xeon Gold 5122 and 256GB RAM. The code is publicly available via GitHub[1].

[1]https://github.com/mwever/incremental-successive-halving

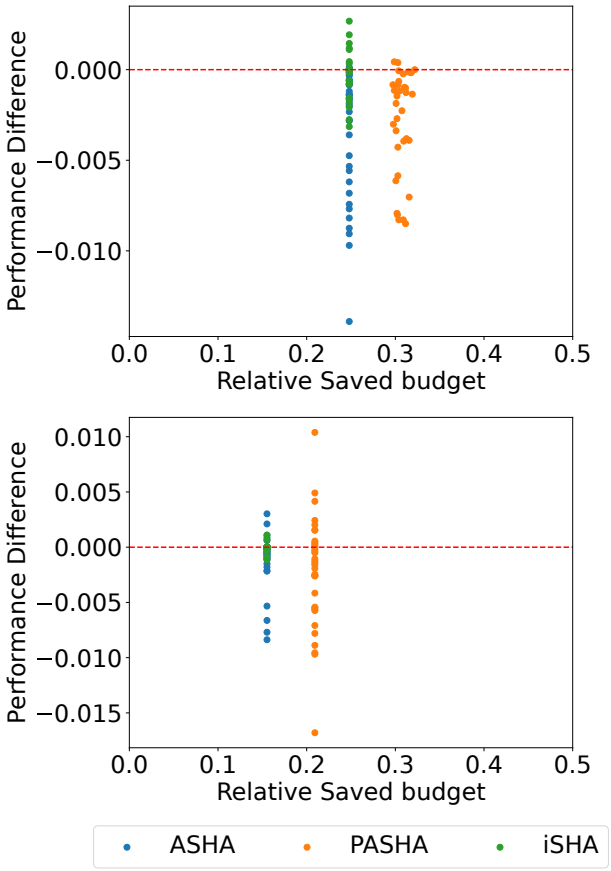

Figure 2: Scatter plots relating the performance on the y-axis and the consumed budget on the x-axis to the performance achieved and budget consumed by SHA. Note that the ranges for the performance and budget vary from $\eta = 2$ (top) to $\eta = 3$ (bottom). Higher values are better for both relative saved budget and relative performance.

7.2 Empirical Results

In Figure 2 we present the performance of the finally chosen hyperparameter configuration and the budget saved by ASHA, PASHA, and iSHA relative to the performance of the solution returned and the budget consumed by re-running SHA from scratch for the higher maximum budget R_t. Hence, a relative performance of 0.0 means that the solution quality matches the one returned by SHA, which is also indicated by the red dashed line, a larger (smaller) value means a performance improvement (degradation) w.r.t. SHA. The relative saved budget denotes the percentage of the budget that any of the approaches saves in contrast to the re-run of SHA. Therefore, a relative saved budget of 0 means that the consumed budget is on par with the budget of SHA. A higher relative saved budget correspondingly means that the approach was more efficient than SHA.

As can be seen, iSHA robustly yields competitive performance to re-running SHA from scratch for a larger maximum assignable budget R, while substantially reducing the consumed budget to roughly 75% for $\eta = 2$ and 84.5% for $\eta = 85\%$. Regarding **RQ1**, we can confirm that iSHA retains the quality of returned HPCs.

		Performance			Budget	
	Approach	Impr	Degr	Tie	Mean	Std
$\eta = 2$	PASHA	12	72	295	0.6926	0.007
	ASHA	9	81	289	0.7520	0.0
	iSHA	4	5	370	0.7520	0.0
$\eta = 3$	PASHA	49	114	216	0.7908	0.0
	ASHA	11	75	293	0.8448	0.0
	iSHA	4	14	361	0.8448	0.0

Table 3: Aggregated statistics across benchmark instances comparing the performance and budget to natively applying SHA. Differences in accuracy larger than 0.001 are considered for improvements or degradations.

On the contrary, the performances of ASHA and PASHA show way more variance, including variations. Since higher rungs are only introduced in PASHA whenever necessary, i.e., if the soft ranking over the configurations of the last two rungs changes, PASHA has the potential to reduce the consumed budget even more than iSHA or ASHA do. However, there is no guarantee that this will maintain performance. As can be seen for $\eta = 3$, PASHA is clearly less robust than ASHA suggesting that the progressive nature of PASHA is introducing even more variance.

This is again confirmed by the results in Table 3, where we simply count the number of benchmark instances for which an improvement, degradation, or tie w.r.t. the performance of re-running SHA is obtained. While PASHA gives the most improvements in terms of performance for both values of η, it also comes with the most performance degradations for $\eta = 3$ which outnumber the improvements by a factor of 2 to 3, whereas for $\eta = 2$ degradations occur more frequently than improvements by a factor of 4 to 5. Furthermore, we provide the average and the standard deviations for the relative budget consumed across the benchmark instances. On average, for both values of η, iSHA and ASHA reduce the budget consistently but PASHA can reduce the budget even more.

While, of course, performance improvements are desirable, the selection of the returned solution is made on the same set of candidates in all approaches, including SHA. Therefore, improvements cannot be interpreted as an advantage of one method over SHA but as random noise effects as these decisions highly depend on the order of hyperparameter configurations being considered in the successive halving variant. Assuming the original behavior of SHA to be the ground truth behavior, we can thereby define a consistency metric between ASHA, PASHA, iSHA and the ground truth behavior of SHA. In Table 4, we present consistencies of ASHA, PASHA, and iSHA to SHA. Fixing a benchmark, the consistency is calculated as follows:

$$(|M|)^{-1} \cdot \sum_i [\![|\mu_{SHA}(i) - \mu_{xSHA}(i)| \leq 0.001]\!],$$

where $[\![\cdot]\!]$ is the indicator function, $\mu_{SHA}(i)$ is the full budget performance for the returned hyperparameter configurations by SHA on instance i and $\mu_{xSHA}(i)$ the performance of another considered approach xSHA and M is the number of instances in the respective benchmark. In this comparison, iSHA stands out to be by far the most consistent approach.

η	ASHA	PASHA	iSHA
2	0.7625	0.7783	**0.9763**
3	0.5699	0.7731	**0.9525**

Table 4: Consistency of the performance of hyperparameter configurations returned by ASHA, PASHA, and iSHA with the performance of those returned by SHA.

From these results, we can conclude that iSHA is a robust and more resource-efficient incremental version of SHA, and the theoretical guarantees given in the previous section can be validated in practice as well. PASHA is able to reduce the consumed budget drastically. However, the reduced budget comes at the risk of losing consistency and lack of performance guarantees. In turn, ASHA reduces the consumed budget in the same way as iSHA does but also performs poorly in terms of consistent behavior with the original SHA version.

8 Conclusion and Future Work

In this paper, we proposed an extension to the well-known HPO method Successive Halving (SHA), called Incremental Successive Halving (iSHA), aiming to improve its efficiency when the max size hyperparameter R of SHA needs to be increased post-hoc. We derived theoretical guarantees on the quality of the final choice, as well as on the saved budget, when a previous SHA run is continued. Furthermore, we provide the first theoretical analysis of asynchronous SHA, emphasizing the price that needs to be paid for the asynchronous promotions. In an empirical study, we also find that iSHA yields results similar to the much more expensive baseline variant of SHA and often better results than the current state-of-art among the asynchronous variants of SHA. In fact, our approach only requires the budget of the sole run with the increased max size.

It is worth noting that we considered a synchronous scenario to have a fair comparison with our proposed method. When a (GPU) cluster is available, the asynchronous nature of both ASHA and PASHA might lead to a parallelization speedup for HPO. However, for many practitioners who do not have such a cluster available, but for instance only one GPU, the speedup of an asynchronous approach is lost. In such a case, a more reliable, incremental and synchronous method such as iSHA is exactly the desired approach.

In future work, we plan to combine our SHA extensions with more sophisticated strategies for sampling hyperparameter configurations, as for example done by [Awad *et al.*, 2021] or [Falkner *et al.*, 2018] and HyperJump, to improve iHB's efficacy and efficiency even further. Another interesting avenue of future research is outlined by PriorBand, where a prior distribution is incorporated for sampling new hyperparameter configurations [Mallik *et al.*, 2023].

Ethical Statement

There are no ethical issues.

Acknowledgments

This work was partially supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

Contribution Statement

Jasmin Brandt and Marcel Wever had an equal contribution on this paper. While Jasmin Brandt was responsable for the theoretical parts, Marcel Wever did the empirical analysis. All authors were involved in devoloping the ideas and writing the paper draft in multiple iterations.

References

[Abbasi-Yadkori *et al.*, 2018] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, Alan Malek, and Michal Valko. Best of Both Worlds: Stochastic & Adversarial Best-arm Identification. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 918–949. PMLR, 06–09 Jul 2018.

[Awad *et al.*, 2021] Noor H. Awad, Neeratyoy Mallik, and Frank Hutter. DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 2147–2153, 2021.

[Azizi *et al.*, 2022] Mohammad Javad Azizi, Branislav Kveton, and Mohammad Ghavamzadeh. Fixed-Budget Best-Arm Identification in Structured Bandits. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 2798–2804. ijcai.org, 2022.

[Bergstra *et al.*, 2011] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[Bischl *et al.*, 2023] 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, 2023.

[Bohdal *et al.*, 2023] Ondrej Bohdal, Lukas Balles, Martin Wistuba, Beyza Ermis, Cédric Archambeau, and Giovanni Zappella. PASHA: Efficient HPO and NAS with Progressive Resource Allocation. In *The 11th International Conference on Learning Representations*, 2023.

[Brandt *et al.*, 2022] Jasmin Brandt, Viktor Bengs, Björn Haddenhorst, and Eyke Hüllermeier. Finding Optimal Arms in Non-stochastic Combinatorial Bandits with Semi-bandit Feedback and Finite Budget. In *Advances in Neural Information Processing Systems*, 2022.

[Brandt *et al.*, 2023] Jasmin Brandt, Elias Schede, Björn Haddenhorst, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. AC-Band: A Combinatorial Bandit-Based Approach to Algorithm Configuration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):12355–12363, 2023.

[Carpentier and Valko, 2015] Alexandra Carpentier and Michal Valko. Simple regret for infinitely many armed bandits. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1133–1141. JMLR.org, 2015.

[Falkner *et al.*, 2018] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1436–1445. PMLR, 2018.

[Feurer and Hutter, 2019] Matthias Feurer and Frank Hutter. Hyperparameter Optimization. In *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 3–33. Springer, 2019.

[Frazier, 2018] Peter I. Frazier. A Tutorial on Bayesian Optimization. *CoRR*, abs/1807.02811, 2018.

[Gong and Sellke, 2023] Xiao-Yue Gong and Mark Sellke. Asymptotically Optimal Pure Exploration for Infinite-Armed Bandits. *CoRR*, abs/2306.01995, 2023.

[Hutter *et al.*, 2011] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization - 5th International Conference*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2011.

[Jamieson and Talwalkar, 2016a] Kevin Jamieson and Ameet Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248. PMLR, 2016.

[Jamieson and Talwalkar, 2016b] Kevin G. Jamieson and Ameet Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 240–248, 2016.

[Karnin *et al.*, 2013] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost Optimal Exploration in Multi-Armed Bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR, 2013.

[Kato *et al.*, 2022] Masahiro Kato, Kaito Ariu, Masaaki Imaizumi, Masatoshi Uehara, Masahiro Nomura, and Chao Qin. Optimal Fixed-Budget Best Arm Identification using the Augmented Inverse Probability Weighting Estimator in Two-Armed Gaussian Bandits with Unknown Variances. *CoRR*, abs/2201.04469, 2022.

[Lattimore and Szepesvári, 2020] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[Li *et al.*, 2018] 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(185):1–52, 2018.

[Li *et al.*, 2020] 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.

[Mallik *et al.*, 2023] Neeratyoy Mallik, Edward Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning. *arXiv preprint arXiv:2306.12370*, 2023.

[Mendes *et al.*, 2021] Pedro Mendes, Maria Casimiro, and Paolo Romano. HyperJump: Accelerating HyperBand via Risk Modelling. *arXiv preprint arXiv:2108.02479*, 2021.

[Pfisterer *et al.*, 2022] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. YAHPO Gym - An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization. In *International Conference on Automated Machine Learning, AutoML 2022*, volume 188 of *Proceedings of Machine Learning Research*, pages 3/1–39. PMLR, 2022.

[Rice, 1976] John R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65–118, 1976.

[Shen, 2019] Cong Shen. Universal Best Arm Identification. *IEEE Trans. Signal Process.*, 67(17):4464–4478, 2019.

[Tornede *et al.*, 2023] Tanja Tornede, Alexander Tornede, Jonas Hanselle, Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Towards green automated machine learning: Status quo and future directions. *Journal of Artificial Intelligence Research*, 77:427–457, 2023.