Journal of Machine Learning Research 18 (2018) 1-52 Submitted 11/16; Revised 12/17; Published 4/18

Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

Lisha Li LISHAL@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213
Kevin Jamieson JAMIESON@CS.WASHINGTON.EDU
University of Washington, Seattle, WA 98195
Giulia DeSalvo GIULIAD@GOOGLE.COM
Google Research, New York, NY 10011
Afshin Rostamizadeh ROSTAMI@GOOGLE.COM
Google Research, New York, NY 10011
Ameet Talwalkar TALWALKAR@CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213
Determined AI

Editor: Nando de Freitas

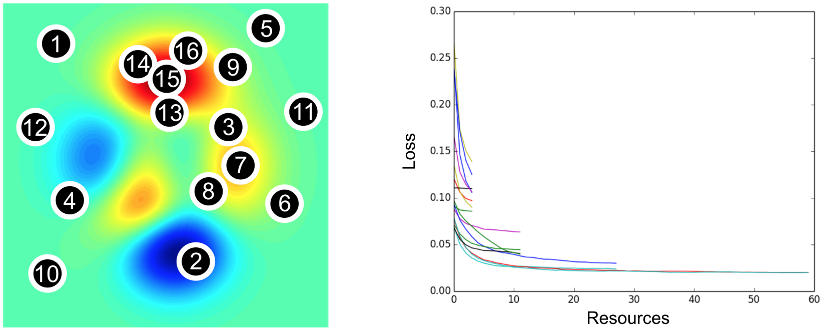
Abstract

Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian optimization to adaptively select configurations, we focus on speeding up random search through adaptive resource allocation and early-stopping. We formulate hyperparameter optimization as a pure-exploration non-stochastic infinite-armed bandit problem where a predefined resource like iterations, data samples, or features is allocated to randomly sampled configurations. We introduce a novel algorithm, Hyperband, for this framework and analyze its theoretical properties, providing several desirable guarantees. Furthermore, we compare Hyperband with popular Bayesian optimization methods on a suite of hyperparameter optimization problems. We observe that Hyperband can provide over an order-of-magnitude speedup over our competitor set on a variety of deep-learning and kernel-based learning problems.

Keywords: hyperparameter optimization, model selection, infinite-armed bandits, online optimization, deep learning

1. Introduction

In recent years, machine learning models have exploded in complexity and expressibility at the price of staggering computational costs. Moreover, the growing number of tuning parameters associated with these models are difficult to set by standard optimization techniques. These "hyperparameters" are inputs to a machine learning algorithm that govern how the algorithm's performance generalizes to new, unseen data; examples of hyperparameters include those that impact model architecture, amount of regularization, and learning rates. The quality of a predictive model critically depends on its hyperparameter configuration, but it is poorly understood how these hyperparameters interact with each other to affect the resulting model.



(a) Configuration Selection

(b) Configuration Evaluation

Figure 1: (a) The heatmap shows the validation error over a two-dimensional search space with red corresponding to areas with lower validation error. Configuration selection methods adaptively choose new configurations to train, proceeding in a sequential manner as indicated by the numbers. (b) The plot shows the validation error as a function of the resources allocated to each configuration (i.e. each line in the plot). Configuration evaluation methods allocate more resources to promising configurations.

Consequently, practitioners often default to brute-force methods like random search and grid search (Bergstra and Bengio, 2012).

In an effort to develop more efficient search methods, the problem of hyperparameter optimization has recently been dominated by *Bayesian optimization* methods (Snoek et al., 2012; Hutter et al., 2011; Bergstra et al., 2011) that focus on optimizing hyperparameter *configuration selection*. These methods aim to identify good configurations more quickly than standard baselines like random search by selecting configurations in an adaptive manner; see Figure 1(a). Existing empirical evidence suggests that these methods outperform random search (Thornton et al., 2013; Eggensperger et al., 2013; Snoek et al., 2015b). However, these methods tackle the fundamentally challenging problem of simultaneously fitting and optimizing a high-dimensional, non-convex function with unknown smoothness, and possibly noisy evaluations.

An orthogonal approach to hyperparameter optimization focuses on speeding up *configuration evaluation*; see Figure 1(b). These approaches are adaptive in computation, allocating more resources to promising hyperparameter configurations while quickly eliminating poor ones. Resources can take various forms, including size of training set, number of features, or number of iterations for iterative algorithms. By adaptively allocating resources, these approaches aim to examine orders-of-magnitude more hyperparameter configurations than approaches that uniformly train all configurations to completion, thereby quickly identifying good hyperparameters. While there are methods that combine Bayesian optimization with adaptive resource allocation (Swersky et al., 2013, 2014; Domhan et al., 2015; Klein et al., 2017a), we focus on speeding up random search as it offers a simple and theoretically principled launching point (Bergstra and Bengio, 2012).[1]

We develop a novel configuration evaluation approach by formulating hyperparameter optimization as a pure-exploration adaptive resource allocation problem addressing how to allocate resources among randomly sampled hyperparameter configurations.[2] Our procedure, Hyperband, relies on a principled early-stopping strategy to allocate resources, allowing it to evaluate orders-of-magnitude more configurations than black-box procedures like Bayesian optimization methods. Hyperband is a general-purpose technique that makes minimal assumptions unlike prior configuration evaluation approaches (Domhan et al., 2015; Swersky et al., 2014; György and Kocsis, 2011; Agarwal et al., 2011; Sparks et al., 2015; Jamieson and Talwalkar, 2015).

Our theoretical analysis demonstrates the ability of Hyperband to adapt to unknown convergence rates and to the behavior of validation losses as a function of the hyperparameters. In addition, Hyperband is 5× to 30× faster than popular Bayesian optimization algorithms on a variety of deep-learning and kernel-based learning problems. A theoretical contribution of this work is the introduction of the pure-exploration, infinite-armed bandit problem in the non-stochastic setting, for which Hyperband is one solution. When Hyperband is applied to the special-case stochastic setting, we show that the algorithm comes within log factors of known lower bounds in both the infinite (Carpentier and Valko, 2015) and finite K-armed bandit settings (Kaufmann et al., 2015).

The paper is organized as follows. Section 2 summarizes related work in two areas: (1) hyperparameter optimization, and (2) pure-exploration bandit problems. Section 3 describes Hyperband and provides intuition for the algorithm through a detailed example. In Section 4, we present a wide range of empirical results comparing Hyperband with state-of-the-art competitors. Section 5 frames the hyperparameter optimization problem as an infinite-armed bandit problem and summarizes the theoretical results for Hyperband. Finally, Section 6 discusses possible extensions of Hyperband.

2. Related Work

In Section 1, we briefly discussed related work in the hyperparameter optimization literature. Here, we provide a more thorough coverage of the prior work, and also summarize significant related work on bandit problems.

2.1 Hyperparameter Optimization

Bayesian optimization techniques model the conditional probability $p(y|\lambda)$ of a configuration's performance on an evaluation metric y (i.e., test accuracy), given a set of hyperparameters λ.

[1] Random search will asymptotically converge to the optimal configuration, regardless of the smoothness or structure of the function being optimized, by a simple covering argument. While the rate of convergence for random search depends on the smoothness and is exponential in the number of dimensions in the search space, the same is true for Bayesian optimization methods without additional structural assumptions (Kandasamy et al., 2015).

[2] A preliminary version of this work appeared in Li et al. (2017). We extend the previous paper with a thorough theoretical analysis of Hyperband; an infinite horizon version of the algorithm with application to stochastic infinite-armed bandits; additional intuition and discussion of Hyperband to facilitate its use in practice; and additional results on a collection of 117 multistage model selection tasks.

Sequential Model-based Algorithm Configuration (SMAC), Tree-structure Parzen Estimator (TPE), and Spearmint are three well-established methods (Feurer et al., 2014). SMAC uses random forests to model $p(y|\lambda)$ as a Gaussian distribution (Hutter et al., 2011). TPE is a non-standard Bayesian optimization algorithm based on tree-structured Parzen density estimators (Bergstra et al., 2011). Lastly, Spearmint uses Gaussian processes (GP) to model $p(y|\lambda)$ and performs slice sampling over the GP's hyperparameters (Snoek et al., 2012).

Previous work compared the relative performance of these Bayesian searchers (Thornton et al., 2013; Eggensperger et al., 2013; Bergstra et al., 2011; Snoek et al., 2012; Feurer et al., 2014, 2015). An extensive survey of these three methods by Eggensperger et al. (2013) introduced a benchmark library for hyperparameter optimization called HPOlib, which we use for our experiments. Bergstra et al. (2011) and Thornton et al. (2013) showed Bayesian optimization methods empirically outperform random search on a few benchmark tasks. However, for high-dimensional problems, standard Bayesian optimization methods perform similarly to random search (Wang et al., 2013). Recent methods specifically designed for high-dimensional problems assume a lower effective dimension for the problem (Wang et al., 2013) or an additive decomposition for the target function (Kandasamy et al., 2015). However, as can be expected, the performance of these methods is sensitive to required inputs; i.e. the effective dimension (Wang et al., 2013) or the number of additive components (Kandasamy et al., 2015).

Gaussian processes have also been studied in the bandit setting using confidence bound acquisition functions (GP-UCB), with associated sublinear regret bounds (Srinivas et al., 2010; Grünewälder et al., 2010). Wang et al. (2016) improved upon GP-UCB by removing the need to tune a parameter that controls exploration and exploitation. Contal et al. (2014) derived a tighter regret bound than that for GP-UCB by using a mutual information acquisition function. However, van der Vaart and van Zanten (2011) showed that the learning rate of GPs are sensitive to the definition of the prior through an example with a poor prior where the learning rate degraded from polynomial to logarithmic in the number of observations n. Additionally, without structural assumptions on the covariance matrix of the GP, fitting the posterior is $O(n^3)$ (Wilson et al., 2015). Hence, Snoek et al. (2015a) and Springenberg et al. (2016) proposed using Bayesian neural networks, which scale linearly with n, to model the posterior.

Adaptive configuration evaluation is not a new idea. Maron and Moore (1997) and Mnih and Audibert (2008) considered a setting where the training time is relatively inexpensive (e.g., k-nearest-neighbor classification) and evaluation on a large validation set is accelerated by evaluating on an increasing subset of the validation set, stopping early configurations that are performing poorly. Since subsets of the validation set provide unbiased estimates of its expected performance, this is an instance of the *stochastic* best-arm identification problem for multi-armed bandits (see the work by Jamieson and Nowak, 2014, for a brief survey).

In contrast, we address a setting where the evaluation time is relatively inexpensive and the goal is to early-stop long-running training procedures by evaluating partially trained models on the full validation set. Previous approaches in this setting either require strong assumptions or use heuristics to perform adaptive resource allocation. György and Kocsis (2011) and Agarwal et al. (2011) made parametric assumptions on the convergence behavior of training algorithms, providing theoretical performance guarantees under these assumptions. Unfortunately, these assumptions are often hard to verify, and empirical performance can drastically suffer when they are violated. Krueger et al. (2015) proposed a heuristic based on sequential analysis to determine stopping times for training configurations on increasing subsets of the data. However, the theoretical correctness and empirical performance of this method are highly dependent on a user-defined "safety zone."

Several hybrid methods combining adaptive configuration selection and evaluation have also been introduced (Swersky et al., 2013, 2014; Domhan et al., 2015; Kandasamy et al., 2016; Klein et al., 2017a; Golovin et al., 2017). The algorithm proposed by Swersky et al. (2013) uses a GP to learn correlation between related tasks and requires the subtasks as input, but efficient subtasks with high informativeness for the target task are unknown without prior knowledge. Similar to the work by Swersky et al. (2013), Klein et al. (2017a) modeled the conditional validation error as a Gaussian process using a kernel that captures the covariance with downsampling rate to allow for adaptive evaluation. Swersky et al. (2014), Domhan et al. (2015), and Klein et al. (2017a) made parametric assumptions on the convergence of learning curves to perform early-stopping. In contrast, Golovin et al. (2017) devised an early-stopping rule based on predicted performance from a nonparametric GP model with a kernel designed to measure the similarity between performance curves. Finally, Kandasamy et al. (2016) extended GP-UCB to allow for adaptive configuration evaluation by defining subtasks that monotonically improve with more resources.

In another line of work, Sparks et al. (2015) proposed a halving style bandit algorithm that did not require explicit convergence behavior, and Jamieson and Talwalkar (2015) analyzed a similar algorithm originally proposed by Karnin et al. (2013) for a different setting, providing theoretical guarantees and encouraging empirical results. Unfortunately, these halving style algorithms suffer from the "n versus B/n" problem, which we will discuss in Section 3.1. Hyperband addresses this issue and provides a robust, theoretically principled early-stopping algorithm for hyperparameter optimization.

We note that Hyperband can be combined with any hyperparameter sampling approach and does not depend on random sampling; the theoretical results only assume the validation losses of sampled hyperparameter configurations are drawn from some stationary distribution. In fact, subsequent to our submission, Klein et al. (2017b) combined adaptive configuration selection with Hyperband by using a Bayesian neural network to model learning curves and only selecting configurations with high predicted performance to input into Hyperband.

2.2 Bandit Problems

Pure exploration bandit problems aim to minimize the simple regret, defined as the distance from the optimal solution, as quickly as possible in any given setting. The pure-exploration multi-armed bandit problem has a long history in the stochastic setting (Even-Dar et al., 2006; Bubeck et al., 2009), and was recently extended to the non-stochastic setting by Jamieson and Talwalkar (2015). Relatedly, the stochastic pure-exploration infinite-armed bandit problem was studied by Carpentier and Valko (2015), where a pull of each arm i yields an i.i.d. sample in $[0, 1]$ with expectation ν_i, where ν_i is a loss drawn from a distribution with cumulative distribution function, F. Of course, the value of ν_i is unknown to the player, so the only way to infer its value is to pull arm i many times. Carpentier and Valko (2015) proposed an anytime algorithm, and derived a tight (up to polylog factors) upper bound on its error assuming what we will refer to as the β-parameterization of F described in

Section 5.3.2. However, their algorithm was derived specifically for the β-parameterization of F, and furthermore, they must estimate β before running the algorithm, limiting the algorithm's practical applicability. Also, the algorithm assumes stochastic losses from the arms and thus the convergence behavior is known; consequently, it does not apply in our hyperparameter optimization setting.[3] Two related lines of work that both make use of an underlying metric space are Gaussian process optimization (Srinivas et al., 2010) and X-armed bandits (Bubeck et al., 2011), or bandits defined over a metric space. However, these works either assume stochastic rewards or need to know something about the underlying function (e.g. an appropriate kernel or level of smoothness).

In contrast, Hyperband is devised for the non-stochastic setting and automatically adapts to unknown F without making any parametric assumptions. Hence, we believe our work to be a generally applicable pure exploration algorithm for infinite-armed bandits. To the best of our knowledge, this is also the first work to test out such an algorithm on a real application.

3. Hyperband Algorithm

In this section, we present the Hyperband algorithm. We provide intuition for the algorithm, highlight the main ideas via a simple example that uses iterations as the adaptively allocated resource, and present a few guidelines on how to deploy Hyperband in practice.

3.1 Successive Halving

Hyperband extends the SuccessiveHalving algorithm proposed for hyperparameter optimization by Jamieson and Talwalkar (2015) and calls it as a subroutine. The idea behind the original SuccessiveHalving algorithm follows directly from its name: uniformly allocate a budget to a set of hyperparameter configurations, evaluate the performance of all configurations, throw out the worst half, and repeat until one configuration remains. The algorithm allocates exponentially more resources to more promising configurations. Unfortunately, SuccessiveHalving requires the number of configurations n as an input to the algorithm. Given some finite budget B (e.g., an hour of training time to choose a hyperparameter configuration), B/n resources are allocated on average across the configurations. However, for a fixed B, it is not clear a priori whether we should (a) consider many configurations (large n) with a small average training time; or (b) consider a small number of configurations (small n) with longer average training times.

We use a simple example to better understand this tradeoff. Figure 2 shows the validation loss as a function of total resources allocated for two configurations with terminal validation losses ν_1 and ν_2. The shaded areas bound the maximum deviation of the intermediate losses from the terminal validation loss and will be referred to as "envelope" functions.[4] It is possible to distinguish between the two configurations when the envelopes no longer overlap. Simple arithmetic shows that this happens when the width of the envelopes is less than $\nu_2 - \nu_1$, i.e., when the intermediate losses are guaranteed to be less than $\frac{\nu_2 - \nu_1}{2}$ away from the

[3] See the work by Jamieson and Talwalkar (2015) for detailed discussion motivating the non-stochastic setting for hyperparameter optimization.

[4] These envelope functions are guaranteed to exist; see discussion in Section 5.2 where we formally define these envelope (or γ) functions.

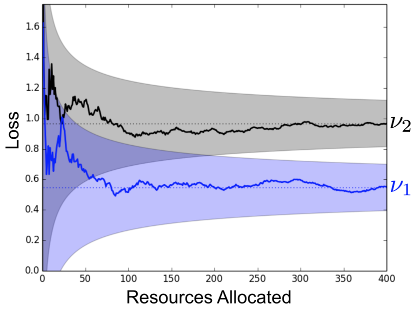


Figure 2: The validation loss as a function of total resources allocated for two configurations is shown. ν_1 and ν_2 represent the terminal validation losses at convergence. The shaded areas bound the maximum distance of the intermediate losses from the terminal validation loss and monotonically decrease with the resource.

terminal losses. There are two takeaways from this observation: more resources are needed to differentiate between the two configurations when either (1) the envelope functions are wider or (2) the terminal losses are closer together.

However, in practice, the optimal allocation strategy is unknown because we do not have knowledge of the envelope functions nor the distribution of terminal losses. Hence, if more resources are required before configurations can differentiate themselves in terms of quality (e.g., if an iterative training method converges very slowly for a given data set or if randomly selected hyperparameter configurations perform similarly well), then it would be reasonable to work with a small number of configurations. In contrast, if the quality of a configuration is typically revealed after a small number of resources (e.g., if iterative training methods converge very quickly for a given data set or if randomly selected hyperparameter configurations are of low-quality with high probability), then n is the bottleneck and we should choose n to be large.

Certainly, if meta-data or previous experience suggests that a certain tradeoff is likely to work well in practice, one should exploit that information and allocate the majority of resources to that tradeoff. However, without this supplementary information, practitioners are forced to make this tradeoff, severely hindering the applicability of existing configuration evaluation methods.

3.2 Hyperband

Hyperband, shown in Algorithm 1, addresses this "n versus B/n" problem by considering several possible values of n for a fixed B, in essence performing a grid search over feasible value of n. Associated with each value of n is a minimum resource r that is allocated to all configurations before some are discarded; a larger value of n corresponds to a smaller r and hence more aggressive early-stopping. There are two components to Hyperband; (1) the inner loop invokes SuccessiveHalving for fixed values of n and r (lines 3–9) and (2) the outer loop iterates over different values of n and r (lines 1–2). We will refer to each such run of SuccessiveHalving within Hyperband as a "bracket." Each bracket is designed to use approximately B total resources and corresponds to a different tradeoff between n

Algorithm 1: Hyperband algorithm for hyperparameter optimization.

```
input         : $R$, $\eta$ (default $\eta = 3$)
initialization: $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
for $s \in \{s_{\max}, s_{\max} - 1, \dots, 0\}$ do
    $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$,    $r = R\eta^{-s}$
    // begin SuccessiveHalving with $(n, r)$ inner loop
    $T =$ get_hyperparameter_configuration($n$)
    for $i \in \{0, \dots, s\}$ do
        $n_i = \lfloor n\eta^{-i} \rfloor$
        $r_i = r\eta^i$
        $L = \{$run_then_return_val_loss$(t, r_i) : t \in T\}$
        $T =$ top_k$(T, L, \lfloor n_i/\eta \rfloor)$
    end
end
return Configuration with the smallest intermediate loss seen so far.
```

and B/n. Hence, a single execution of Hyperband takes a finite budget of $(s_{max} + 1)B$; we recommend repeating it indefinitely.

Hyperband requires two inputs (1) R, the maximum amount of resource that can be allocated to a single configuration, and (2) η, an input that controls the proportion of configurations discarded in each round of SuccessiveHalving. The two inputs dictate how many different brackets are considered; specifically, $s_{\max} + 1$ different values for n are considered with $s_{\max} = \lfloor \log_\eta(R) \rfloor$. Hyperband begins with the most aggressive bracket $s = s_{\max}$, which sets n to maximize exploration, subject to the constraint that at least one configuration is allocated R resources. Each subsequent bracket reduces n by a factor of approximately η until the final bracket, $s = 0$, in which every configuration is allocated R resources (this bracket simply performs classical random search). Hence, Hyperband performs a geometric search in the average budget per configuration and removes the need to select n for a fixed budget at the cost of approximately $s_{\max} + 1$ times more work than running SuccessiveHalving for a single value of n. By doing so, Hyperband is able to exploit situations in which adaptive allocation works well, while protecting itself in situations where more conservative allocations are required.

Hyperband requires the following methods to be defined for any given learning problem:

- `get_hyperparameter_configuration(`n`)` – a function that returns a set of n i.i.d. samples from some distribution defined over the hyperparameter configuration space. In this work, we assume uniformly sampling of hyperparameters from a predefined space (i.e., hypercube with min and max bounds for each hyperparameter), which immediately yields consistency guarantees. However, the more aligned the distribution is towards high quality hyperparameters (i.e., a useful prior), the better Hyperband will perform (see Section 6 for further discussion).

	$s = 4$		$s = 3$		$s = 2$		$s = 1$		$s = 0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Table 1: The values of n_i and r_i for the brackets of Hyperband corresponding to various values of s, when $R = 81$ and $\eta = 3$.

- `run_then_return_val_loss(t, r)` – a function that takes a hyperparameter configuration t and resource allocation r as input and returns the validation loss after training the configuration for the allocated resources.

- `top_k(configs, losses, k)` – a function that takes a set of configurations as well as their associated losses and returns the top k performing configurations.

3.3 Example Application with Iterations as a Resource: LeNet

We next present a concrete example to provide further intuition about Hyperband. We work with the MNIST data set and optimize hyperparameters for the LeNet convolutional neural network trained using mini-batch stochastic gradient descent (SGD).[5] Our search space includes learning rate, batch size, and number of kernels for the two layers of the network as hyperparameters (details are shown in Table 2 in Appendix A).

We define the resource allocated to each configuration to be number of iterations of SGD, with one unit of resource corresponding to one epoch, i.e., a full pass over the data set. We set R to 81 and use the default value of $\eta = 3$, resulting in $s_{\max} = 4$ and thus 5 brackets of SuccessiveHalving with different tradeoffs between n and B/n. The resources allocated within each bracket are displayed in Table 1.

Figure 3 shows an empirical comparison of the average test error across 70 trials of the individual brackets of Hyperband run separately as well as standard Hyperband. In practice, we do not know a priori which bracket $s \in \{0, \ldots, 4\}$ will be most effective in identifying good hyperparameters, and in this case neither the most ($s = 4$) nor least aggressive ($s = 0$) setting is optimal. However, we note that Hyperband does nearly as well as the optimal bracket ($s = 3$) and outperforms the baseline uniform allocation (i.e., random search), which is equivalent to bracket $s = 0$.

3.4 Different Types of Resources

While the previous example focused on iterations as the resource, Hyperband naturally generalizes to various types of resources:

[5] Code and description of algorithm used is available at `http://deeplearning.net/tutorial/lenet.html`.

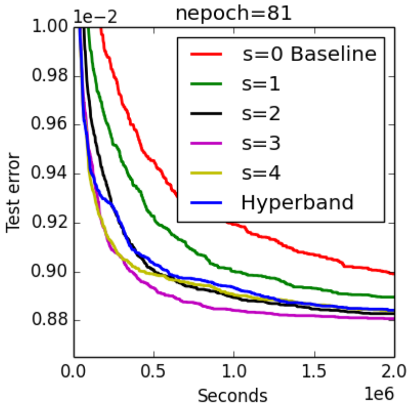


Figure 3: Performance of individual brackets s and Hyperband.

- **Time** – Early-stopping in terms of time can be preferred when various hyperparameter configurations differ in training time and the practitioner's chief goal is to find a good hyperparameter setting in a fixed wall-clock time. For instance, training time could be used as a resource to quickly terminate straggler jobs in distributed computation environments.
- **Data Set Subsampling** – Here we consider the setting of a black-box batch training algorithm that takes a data set as input and outputs a model. In this setting, we treat the resource as the size of a random subset of the data set with R corresponding to the full data set size. Subsampling data set sizes using Hyperband, especially for problems with super-linear training times like kernel methods, can provide substantial speedups.
- **Feature Subsampling** – Random features or Nyström-like methods are popular methods for approximating kernels for machine learning applications (Rahimi and Recht, 2007). In image processing, especially deep-learning applications, filters are usually sampled randomly, with the number of filters having an impact on the performance. Downsampling the number of features is a common tool used when hand-tuning hyperparameters; Hyperband can formalize this heuristic.

3.5 Setting R

The resource R and η (which we address next) are the only required inputs to Hyperband. As mentioned in Section 3.2, R represents the maximum amount of resources that can be allocated to any given configuration. In most cases, there is a natural upper bound on the maximum budget per configuration that is often dictated by the resource type (e.g., training set size for data set downsampling; limitations based on memory constraint for feature downsampling; rule of thumb regarding number of epochs when iteratively training neural networks). If there is a range of possible values for R, a smaller R will give a result faster (since the budget B for each bracket is a multiple of R), but a larger R will give a better guarantee of successfully differentiating between the configurations.

Moreover, for settings in which either R is unknown or not desired, we provide an infinite horizon version of Hyperband in Section 5. This version of the algorithm doubles the budget over time, $B \in \{2, 4, 8, 16, \ldots\}$, and for each B, tries all possible values of $n \in \left\{2^k : k \in \{1, \ldots, \log_2(B)\}\right\}$. For each combination of B and n, the algorithm runs an instance of the (infinite horizon) SuccessiveHalving algorithm, which implicitly sets $R = \frac{B}{2\log_2(n)}$, thereby growing R as B increases. The main difference between the infinite horizon algorithm and Algorithm 1 is that the number of unique brackets grows over time instead of staying constant with each outer loop. We will analyze this version of Hyperband in more detail in Section 5 and use it as the launching point for the theoretical analysis of standard (finite horizon) Hyperband.

Note that R is also the number of configurations evaluated in the bracket that performs the most exploration, i.e $s = s_{\max}$. In practice one may want $n \leq n_{\max}$ to limit overhead associated with training many configurations on a small budget, i.e., costs associated with initialization, loading a model, and validation. In this case, set $s_{\max} = \lfloor \log_\eta(n_{\max}) \rfloor$. Alternatively, one can redefine one unit of resource so that R is artificially smaller (i.e., if the desired maximum iteration is 100k, defining one unit of resource to be 100 iterations will give $R = 1,000$, whereas defining one unit to be 1k iterations will give $R = 100$). Thus, one unit of resource can be interpreted as the minimum desired resource and R as the ratio between maximum resource and minimum resource.

3.6 Setting η

The value of η is a knob that can be tuned based on *practical* user constraints. Larger values of η correspond to more aggressive elimination schedules and thus fewer rounds of elimination; specifically, each round retains $1/\eta$ configurations for a total of $\lfloor \log_\eta(n) \rfloor + 1$ rounds of elimination with n configurations. If one wishes to receive a result faster at the cost of a sub-optimal asymptotic constant, one can increase η to reduce the budget per bracket $B = (\lfloor \log_\eta(R) \rfloor + 1)R$. We stress that results are not very sensitive to the choice of η. If our theoretical bounds are optimized (see Section 5), they suggest choosing $\eta = e \approx 2.718$, but in practice we suggest taking η to be equal to 3 or 4.

Tuning η will also change the number of brackets and consequently the number of different tradeoffs that Hyperband tries. Usually, the possible range of brackets is fairly constrained, since the number of brackets is logarithmic in R; namely, there are $(\lfloor \log_\eta(R) \rfloor + 1) = s_{\max} + 1$ brackets. For our experiments in Section 4, we chose η to provide 5 brackets for the specified R; for most problems, 5 is a reasonable number of n versus B/n tradeoffs to explore. However, for large R, using $\eta = 3$ or 4 can give more brackets than desired. The number of brackets can be controlled in a few ways. First, as mentioned in the previous section, if R is too large and overhead is an issue, then one may want to control the overhead by limiting the maximum number of configurations to $n_{\max}$, thereby also limiting $s_{\max}$. If overhead is not a concern and aggressive exploration is desired, one can (1) increase η to reduce the number of brackets while maintaining R as the maximum number of configurations in the most exploratory bracket, or (2) still use $\eta = 3$ or 4 but only try brackets that do a baseline level of exploration, i.e., set $n_{\min}$ and only try brackets from $s_{\max}$ to $s = \lfloor \log_\eta(n_{\min}) \rfloor$. For computationally intensive problems that have long training times and high-dimensional search spaces, we recommend the latter. Intuitively, if the number of configurations that can be trained to completion (i.e., trained using R resources) in a reasonable amount of time is on the order of the dimension of the search space and not exponential in the dimension, then it will be impossible to find a good configuration without using an aggressive exploratory tradeoff between n and B/n.

3.7 Overview of Theoretical Results

The theoretical properties of Hyperband are best demonstrated through an example. Suppose there are n configurations, each with a given terminal validation error ν_i for $i = 1, \ldots, n$. Without loss of generality, index the configurations by performance so that ν_1 corresponds to the best performing configuration, ν_2 to the second best, and so on. Now consider the task of identifying the best configuration. The optimal strategy would allocate to each configuration i the minimum resource required to distinguish it from ν_1, i.e., enough so that the envelope functions (see Figure 2) bound the intermediate loss to be less than $\frac{\nu_i - \nu_1}{2}$ away from the terminal value. In contrast, the naive uniform allocation strategy, which allocates B/n to each configuration, has to allocate to every configuration the maximum resource required to distinguish any arm ν_i from ν_1. Remarkably, the budget required by SuccessiveHalving is only a small factor of the optimal because it capitalizes on configurations that are easy to distinguish from ν_1.

The relative size of the budget required for uniform allocation and SuccessiveHalving depends on the envelope functions bounding deviation from terminal losses as well as the distribution from which ν_i's are drawn. The budget required for SuccessiveHalving is smaller when the optimal n versus B/n tradeoff discussed in Section 3.1 requires fewer resources per configuration. Hence, if the envelope functions tighten quickly as a function of resource allocated, or the average distances between terminal losses is large, then SuccessiveHalving can be substantially faster than uniform allocation. These intuitions are formalized in Section 5 and associated theorems/corollaries are provided that take into account the envelope functions and the distribution from which ν_i's are drawn.

In practice, we do not have knowledge of either the envelope functions or the distribution of ν_i's, both of which are integral in characterizing SuccessiveHalving's required budget. With Hyperband we address this shortcoming by hedging our aggressiveness. We show in Section 5.3.3 that Hyperband, despite having no knowledge of the envelope functions nor the distribution of ν_i's, requires a budget that is only log factors larger than that of SuccessiveHalving.

4. Hyperparameter Optimization Experiments

In this section, we evaluate the empirical behavior of Hyperband with three different resource types: iterations, data set subsamples, and feature samples. For all experiments, we compare Hyperband with three well known Bayesian optimization algorithms—SMAC, TPE, and Spearmint—using their default settings. We exclude Spearmint from the comparison set when there are conditional hyperparameters in the search space because it does not natively support them (Eggensperger et al., 2013). We also show results for SuccessiveHalving corresponding to repeating the most exploratory bracket of Hyperband to provide a baseline for aggressive early-stopping.[6] Additionally, as standard baselines against which to measure

[6] This is not done for the experiments in Section 4.2.1, since the most aggressive bracket varies from dataset to dataset with the number of training points.

all speedups, we consider random search and "random 2×," a variant of random search with twice the budget of other methods. Of the hybrid methods described in Section 2, we compare to a variant of SMAC using the early termination criterion proposed by Domhan et al. (2015) in the deep learning experiments described in Section 4.1. We think a comparison of Hyperband to more sophisticated hybrid methods introduced recently by Klein et al. (2017a) and Kandasamy et al. (2017) is a fruitful direction for future work.

In the experiments below, we followed these loose guidelines when determining how to configuration Hyperband:

1. The maximum resource R should be reasonable given the problem, but ideally large enough so that early-stopping is beneficial.

2. η should depend on R and be selected to yield ≈ 5 brackets with a minimum of 3 brackets. This is to guarantee that Hyperband will use a baseline degree of early-stopping and prevent too coarse of a grid of n vs B tradeoffs.

4.1 Early-Stopping Iterative Algorithms for Deep Learning

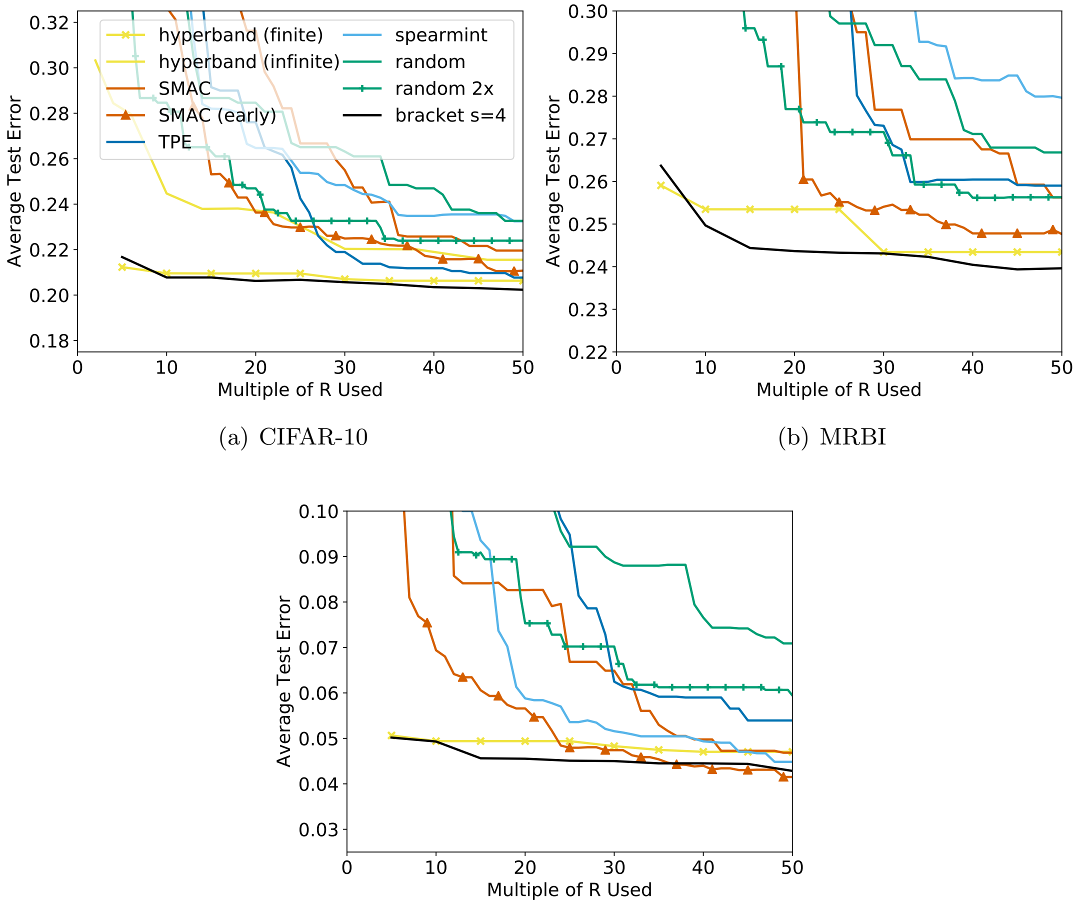
For this benchmark, we tuned a convolutional neural network[7] with the same architecture as that used in Snoek et al. (2012) and Domhan et al. (2015). The search spaces used in the two previous works differ, and we used a search space similar to that of Snoek et al. (2012) with 6 hyperparameters for stochastic gradient decent and 2 hyperparameters for the response normalization layers (see Appendix A for details). In line with the two previous works, we used a batch size of 100 for all experiments.

Data sets: We considered three image classification data sets: CIFAR-10 (Krizhevsky, 2009), rotated MNIST with background images (MRBI) (Larochelle et al., 2007), and Street View House Numbers (SVHN) (Netzer et al., 2011). CIFAR-10 and SVHN contain 32×32 RGB images while MRBI contains 28×28 grayscale images. Each data set was split into a training, validation, and test set: (1) CIFAR-10 has 40k, 10k, and 10k instances; (2) MRBI has 10k, 2k, and 50k instances; and (3) SVHN has close to 600k, 6k, and 26k instances for training, validation, and test respectively. For all data sets, the only preprocessing performed on the raw images was demeaning.

Hyperband Configuration: For these experiments, one unit of resource corresponds to 100 mini-batch iterations (10k examples with a batch size of 100). For CIFAR-10 and MRBI, R was set to 300 (or 30k total iterations). For SVHN, R was set to 600 (or 60k total iterations) to accommodate the larger training set. Given R for these experiments, we set $\eta = 4$ to yield five SuccessiveHalving brackets for Hyperband.

Results: Each searcher was given a total budget of $50R$ per trial to return the best possible hyperparameter configuration. For Hyperband, the budget is sufficient to run the outer loop twice (for a total of 10 SuccessiveHalving brackets). For SMAC, TPE, and random search, the budget corresponds to training 50 different configurations to completion. Ten independent trials were performed for each searcher. The experiments took the equivalent of over 1 year of GPU hours on NVIDIA GRID K520 cards available on Amazon EC2 `g2.8xlarge` instances. We set a total budget constraint in terms of

[7] The model specification is available at `http://code.google.com/p/cuda-convnet/`.



(a) CIFAR-10

(b) MRBI

(c) SVHN

Figure 4: Average test error across 10 trials. Label "SMAC (early)" corresponds to SMAC with the early-stopping criterion proposed in Domhan et al. (2015) and label "bracket $s = 4$" corresponds to repeating the most exploratory bracket of Hyperband.

iterations instead of compute time to make comparisons hardware independent.[8] Comparing progress by iterations instead of time ignores overhead costs, e.g. the cost of configuration selection for Bayesian methods and model initialization and validation costs for Hyperband. While overhead is hardware dependent, the overhead for Hyperband is below 5% on EC2 `g2.8xlarge` machines, so comparing progress by time passed would not change results significantly.

For CIFAR-10, the results in Figure 4(a) show that Hyperband is over an order-of-magnitude faster than its competitiors. For MRBI, Hyperband is over an order-of-

[8] Most trials were run on Amazon EC2 g2.8xlarge instances but a few trials were run on different machines due to the large computational demand of these experiments.

magnitude faster than standard configuration selection approaches and 5× faster than SMAC (early). For SVHN, while Hyperband finds a good configuration faster, Bayesian optimization methods are competitive and SMAC (early) outperforms Hyperband. The performance of SMAC (early) demonstrates there is merit to combining early-stopping and adaptive configuration selection.

Across the three data sets, Hyperband and SMAC (early) are the only two methods that consistently outperform random 2×. On these data sets, Hyperband is over 20× faster than random search while SMAC (early) is $\leq 7\times$ faster than random search within the evaluation window. In fact, the first result returned by Hyperband after using a budget of $5R$ is often competitive with results returned by other searchers after using $50R$. Additionally, Hyperband is less variable than other searchers across trials, which is highly desirable in practice (see Appendix A for plots with error bars).

As discussed in Section 3.6, for computationally expensive problems in high-dimensional search spaces, it may make sense to just repeat the most exploratory brackets. Similarly, if meta-data is available about a problem or it is known that the quality of a configuration is evident after allocating a small amount of resource, then one should just repeat the most exploratory bracket. Indeed, for these experiments, bracket $s = 4$ vastly outperforms all other methods on CIFAR-10 and MRBI and is nearly tied with SMAC (early) for first on SVHN.

While we set R for these experiments to facilitate comparison to Bayesian methods and random search, it is also reasonable to use infinite horizon Hyperband to grow the maximum resource until a desired level of performance is reached. We evaluate infinite horizon Hyperband on CIFAR-10 using $\eta = 4$ and a starting budget of $B = 2R$. Figure 4(a) shows that infinite horizon Hyperband is competitive with other methods but does not perform as well as finite horizon Hyperband within the $50R$ budget limit. The infinite horizon algorithm underperforms initially because it has to tune the maximum resource R as well and starts with a less aggressive early-stopping rate. This demonstrates that in scenarios where a max resource is known, it is better to use the finite horizon algorithm. Hence, we focus on the finite horizon version of Hyperband for the remainder of our empirical studies.

Finally, CIFAR-10 is a very popular data set and state-of-the-art models achieve much lower error rates than what is shown in Figure 4. The difference in performance is mainly attributable to higher model complexities and data manipulation (i.e. using reflection or random cropping to artificially increase the data set size). If we limit the comparison to published results that use the same architecture and exclude data manipulation, the best human expert result for the data set is 18% error and the best hyperparameter optimized results are 15.0% for Snoek et al. (2012)[9] and 17.2% for Domhan et al. (2015). These results exceed ours on CIFAR-10 because they train on 25% more data, by including the validation set, and also train for more epochs. When we train the best model found by Hyperband on the combined training and validation data for 300 epochs, the model achieved a test error of 17.0%.

[9] We were unable to reproduce this result even after receiving the optimal hyperparameters from the authors through a personal communication.

4.2 Data Set Subsampling

We studied two different hyperparameter search optimization problems for which Hyperband uses data set subsamples as the resource. The first adopts an extensive framework presented in Feurer et al. (2015) that attempts to automate preprocessing and model selection. Due to certain limitations of the framework that fundamentally limited the impact of data set downsampling, we conducted a second experiment using a kernel classification task.

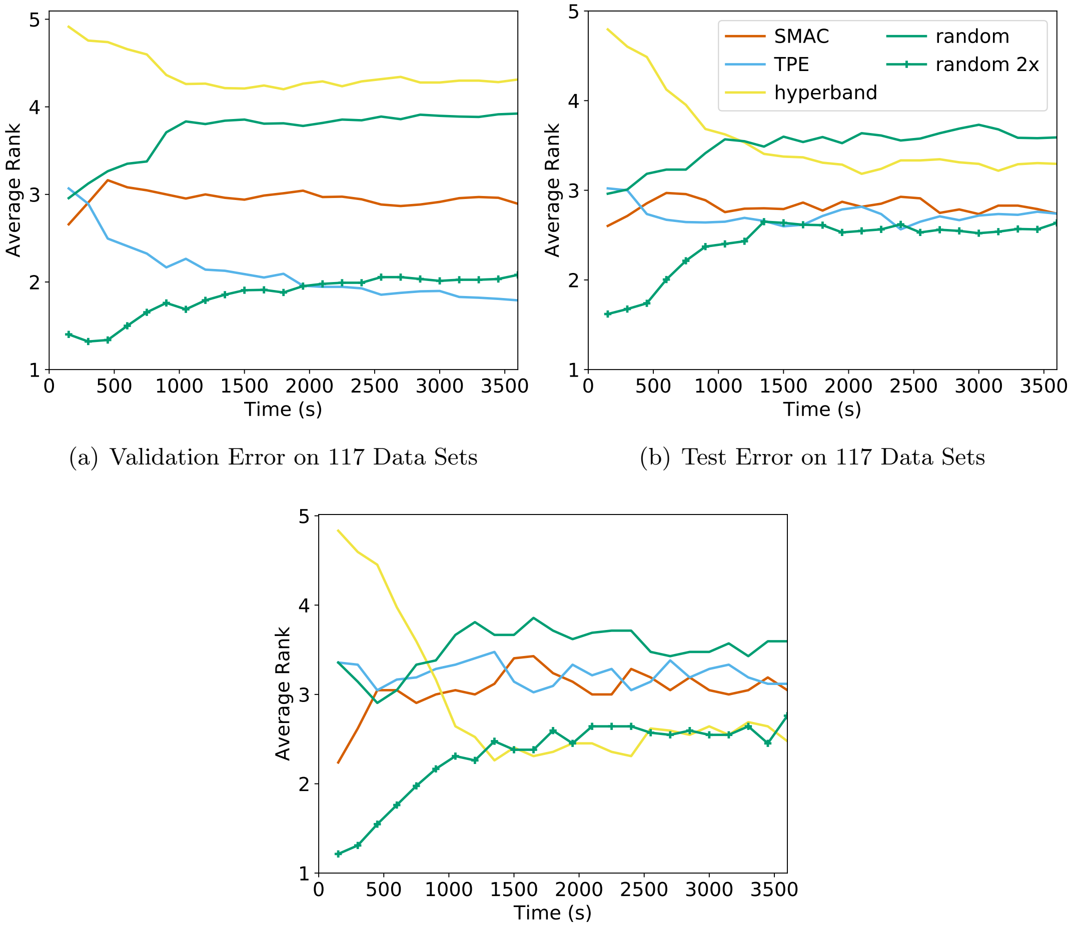
4.2.1 117 Data Sets

We used the framework introduced by Feurer et al. (2015), which explored a structured hyperparameter search space comprised of 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing methods for a total of 110 hyperparameters. We excluded the meta-learning component introduced in Feurer et al. (2015) used to warmstart Bayesian methods with promising configurations, in order to perform a fair comparison with random search and Hyperband. Similar to Feurer et al. (2015), we imposed a 3GB memory limit, a 6-minute timeout for each hyperparameter configuration and a one-hour time window to evaluate each searcher on each data set. Twenty trials of each searcher were performed per data set and all trials in aggregate took over a year of CPU time on `n1-standard-1` instances from Google Cloud Compute. Additional details about our experimental framework are available in Appendix A.

Data sets: Feurer et al. (2015) used 140 binary and multiclass classification data sets from OpenML, but 23 of them are incompatible with the latest version of the OpenML plugin (Feurer, 2015), so we worked with the remaining 117 data sets. Due to the limitations of the experimental setup (discussed in Appendix A), we also separately considered 21 of these data sets, which demonstrated at least modest (though still sublinear) training speedups due to subsampling. Specifically, each of these 21 data sets showed on average at least a 3× speedup due to 8× downsampling on 100 randomly selected hyperparameter configurations.

Hyperband Configuration: Due to the wide range of dataset sizes, with some datasets having fewer than 10k training points, we ran Hyperband with $\eta = 3$ to allow for at least 3 brackets without being overly aggressive in downsampling on small datasets. R was set to the full training set size for each data set and the maximum number of configurations for any bracket of SuccessiveHalving was limited to $n_{\max} = \max\{9, R/1000\}$. This ensured that the most exploratory bracket of Hyperband will downsample at least twice. As mentioned in Section 3.6, when $n_{\max}$ is specified, the only difference when running the algorithm is $s_{\max} = \lfloor \log_\eta(n_{\max}) \rfloor$ instead of $\lfloor \log_\eta(R) \rfloor$.

Results: The results on all 117 data sets in Figure 5(a,b) show that Hyperband outperforms random search in test error rank despite performing worse in validation error rank. Bayesian methods outperform Hyperband and random search in test error performance but also exhibit signs of overfitting to the validation set, as they outperform Hyperband by a larger margin on the validation error rank. Notably, random 2× outperforms all other methods. However, for the subset of 21 data sets, Figure 5(c) shows that Hyperband outperforms all other searchers on test error rank, including random 2× by a very small margin. While these results are more promising, the effectiveness of Hyperband was restricted in this experimental framework; for smaller data sets, the startup overhead was



(a) Validation Error on 117 Data Sets

(b) Test Error on 117 Data Sets

(c) Test Error on 21 Data Sets

Figure 5: Average rank across all data sets for each searcher. For each data set, the searchers are ranked according to the average validation/test error across 20 trials.

high relative to total training time, while for larger data sets, only a handful of configurations could be trained within the hour window.

We note that while average rank plots like those in Figure 5 are an effective way to aggregate information across many searchers and data sets, they provide no indication about the *magnitude* of the differences between the performance of the methods. Figure 6, which charts the difference between the test error for each searcher and that of random search across all 117 datasets, highlights the small difference in the magnitude of the test errors across searchers.

These results are not surprising; as mentioned in Section 2.1, vanilla Bayesian optimization methods perform similarly to random search in high-dimensional search spaces. Feurer et al. (2015) showed that using meta-learning to warmstart Bayesian optimization methods improved performance in this high-dimensional setting. Using meta-learning to identify a

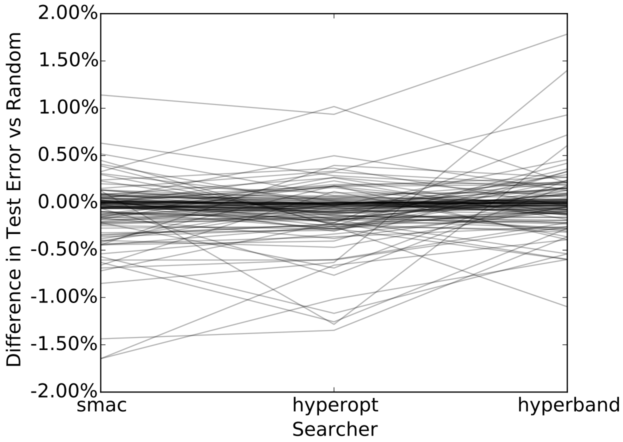


Figure 6: Each line plots, for a single data set, the difference in test error versus random search for each search, where lower is better. Nearly all the lines fall within the -0.5% and 0.5% band and, with the exception of a few outliers, the lines are mostly flat.

promising distribution from which to sample configurations as input into Hyperband is a direction for future work.

4.2.2 Kernel Regularized Least Squares Classification

For this benchmark, we tuned the hyperparameters of a kernel-based classifier on CIFAR-10. We used the multi-class regularized least squares classification model, which is known to have comparable performance to SVMs (Rifkin and Klautau, 2004; Agarwal et al., 2014) but can be trained significantly faster.[10] The hyperparameters considered in the search space include preprocessing method, regularization, kernel type, kernel length scale, and other kernel specific hyperparameters (see Appendix A for more details). For Hyperband, we set $R = 400$, with each unit of resource representing 100 datapoints, and $\eta = 4$ to yield a total of 5 brackets. Each hyperparameter optimization algorithm was run for ten trials on Amazon EC2 `m4.2xlarge` instances; for a given trial, Hyperband was allowed to run for two outer loops, bracket $s = 4$ was repeated 10 times, and all other searchers were run for 12 hours.

Figure 7 shows that Hyperband returned a good configuration after completing the first SuccessiveHalving bracket in approximately 20 minutes; other searchers failed to reach this error rate on average even after the entire 12 hours. Notably, Hyperband was able to evaluate over 250 configurations in this first bracket of SuccessiveHalving, while competitors were able to evaluate only three configurations in the same amount of time. Consequently, Hyperband is over 30× faster than Bayesian optimization methods and 70× faster than random search. Bracket $s = 4$ sightly outperforms Hyperband but the terminal

[10] The default SVM method in Scikit-learn is single core and takes hours to train on CIFAR-10, whereas a block coordinate descent least squares solver takes less than 10 minutes on an 8 core machine.

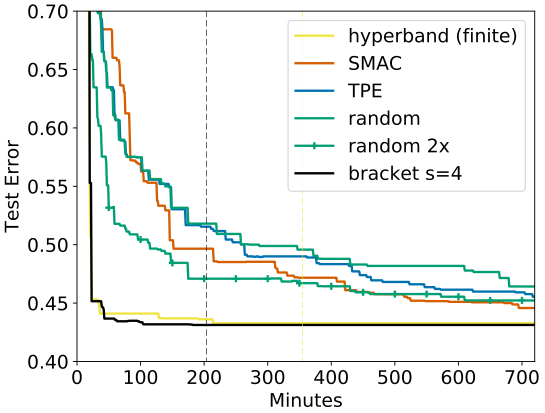


Figure 7: Average test error of the best kernel regularized least square classification model found by each searcher on CIFAR-10. The color coded dashed lines indicate when the last trial of a given searcher finished.

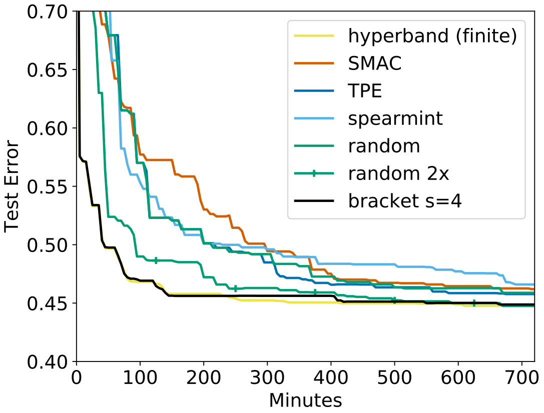


Figure 8: Average test error of the best random features model found by each searcher on CIFAR-10. The test error for Hyperband and bracket $s = 4$ are calculated in every evaluation instead of at the end of a bracket.

performance for the two algorithms are the same. Random 2× is competitive with SMAC and TPE.

4.3 Feature Subsampling to Speed Up Approximate Kernel Classification

Next, we examine the performance of Hyperband when using features as a resource on a random feature kernel approximations task. Features were randomly generated using the method described in Rahimi and Recht (2007) to approximate the RBF kernel, and these random features were then used as inputs to a ridge regression classifier. The hyperparameter search space included the preprocessing method, kernel length scale, and L_2 penalty. While it may seem natural to use infinite horizon Hyperband, since the fidelity of the approximation improves with more random features, in practice, the amount of available machine memory imposes a natural upper bound on the number of features. Thus, we used finite horizion Hyperband with a maximum resource of 100k random features, which comfortably fit into a machine with 60GB of memory. Additionally, we set one unit of resource to be 100 features, so $R = 1000$. Again, we set $\eta = 4$ to yield 5 brackets of SuccessiveHalving. We ran 10 trials of each searcher, with each trial lasting 12 hours on a `n1-standard-16` machine from Google Cloud Compute. The results in Figure 8 show that Hyperband is around 6× faster than Bayesian methods and random search. Hyperband performs similarly to bracket $s = 4$. Random 2× outperforms Bayesian optimization algorithms.

4.4 Experimental Discussion

While our experimental results show Hyperband is a promising algorithm for hyperparameter optimization, a few questions naturally arise:

1. What impacts the speedups provided by Hyperband?
2. Why does SuccessiveHalving seem to outperform Hyperband?
3. What about hyperparameters that should depend on the resource?

We next address each of these questions in turn.

4.4.1 Factors Impacting the Performance of Hyperband

For a given R, the most exploratory SuccessiveHalving round performed by Hyperband evaluates R configurations using a budget of $(\lfloor\log_\eta(R)\rfloor + 1)R$, which gives an upper bound on the potential speedup over random search. If training time scales linearly with the resource, the maximum speedup offered by Hyperband compared to random search is $\frac{R}{(\lfloor\log_\eta(R)\rfloor+1)}$. For the values of η and R used in our experiments, the maximum speedup over random search is approximately 50× given linear training time. However, we observe a range of speedups from 6× to 70× faster than random search. The differences in realized speedup can be explained by three factors:

1. *How training time scales with the given resource.* In cases where training time is superlinear as a function of the resource, Hyperband can offer higher speedups. For instance, if training scales like a polynomial of degree $p > 1$, the maximum speedup for Hyperband over random search is approximately $\frac{\eta^p-1}{\eta^{p-1}}R$. In the kernel least square classifier experiment discussed in Section 4.2.2, the training time scaled quadratically as a function of the resource, which explains why the realized speedup of 70× is higher than the maximum expected speedup given linear scaling.

2. *Overhead costs associated with training.* Total evaluation time also depends on fixed overhead costs associated with evaluating each hyperparameter configuration, e.g., initializing a model, resuming previously trained models, and calculating validation error. For example, in the downsampling experiments on 117 data sets presented in Section 4.2.1, Hyperband did not provide significant speedup because many data sets could be trained in a matter of a few seconds and the initialization cost was high relative to training time.

3. *The difficulty of finding a good configuration.* Hyperparameter optimization problems can vary in difficulty. For instance, an 'easy' problem is one where a randomly sampled configuration is likely to result in a high-quality model, and thus we only need to evaluate a small number of configurations to find a good setting. In contrast, a 'hard' problem is one where an arbitrary configuration is likely to be bad, in which case many configurations must be considered. Hyperband leverages downsampling to boost the number of configurations that are evaluated, and thus is better suited for 'hard' problems where more evaluations are actually necessary to find a good setting. Generally, the difficulty of a problem scales with the dimensionality of the search space. For low-dimensional problems, the number of configurations evaluated by random search and Bayesian methods is exponential in the number of dimensions so good coverage can be achieved. For instance, the low-dimensional ($d = 3$) search space in our feature subsampling experiment in Section 4.3 helps explain why Hyperband is

only 6× faster than random search. In contrast, for the neural network experiments in Section 4.1, we hypothesize that faster speedups are observed for Hyperband because the dimension of the search space is higher.

4.4.2 Comparison to SuccessiveHalving

With the exception of the LeNet experiment (Section 3.3) and the 117 Datasets experiment (Section 4.2.1), the most aggressive bracket of SuccessiveHalving outperformed Hyperband in all of our experiments. In hindsight, we should have just run bracket $s = 4$, since aggressive early-stopping provides massive speedups on many of these benchmarking tasks. However, as previously mentioned, it was unknown a priori that bracket $s = 4$ would perform the best and that is why we have to cycle through all possible brackets with Hyperband. Another question is what happens when one increases s even further, i.e. instead of 4 rounds of elimination, why not 5 or even more with the same maximum resource R? In our case, $s = 4$ was the most aggressive bracket we could run given the minimum resource per configuration limits imposed for the previous experiments. However, for larger data sets, it is possible to extend the range of possible values for s, in which case, Hyperband may either provide even faster speedups if more aggressive early-stopping helps or be slower by a small factor if the most aggressive brackets are essentially throwaways.

We believe prior knowledge about a task can be particularly useful for limiting the range of brackets explored by Hyperband. In our experience, aggressive early-stopping is generally safe for neural network tasks and even more aggressive early-stopping may be reasonable for larger data sets and longer training horizons. However, when pushing the degree of early-stopping by increasing s, one has to consider the additional overhead cost associated with examining more models. Hence, one way to leverage meta-learning would be to use learning curve convergence rate, difficulty of different search spaces, and overhead costs of related tasks to determine the brackets considered by Hyperband.

4.4.3 Resource Dependent Hyperparameters

In certain cases, the setting for a given hyperparameter should depend on the allocated resource. For example, the maximum tree depth regularization hyperparameter for random forests should be higher with more data and more features. However, the optimal tradeoff between maximum tree depth and the resource is unknown and can be data set specific. In these situations, the rate of convergence to the true loss is usually slow because the performance on a smaller resource is not indicative of that on a larger resource. Hence, these problems are particularly difficult for Hyperband, since the benefit of early-stopping can be muted. Again, while Hyperband will only be a small factor slower than that of SuccessiveHalving with the optimal early-stopping rate, we recommend removing the dependence of the hyperparameter on the resource if possible. For the random forest example, an alternative regularization hyperparameter is minimum samples per leaf, which is less dependent on the training set size. Additionally, the dependence can oftentimes be removed with simple normalization. For example, the regularization term for our kernel least squares experiments were normalized by the training set size to maintain a constant tradeoff between the mean-squared error and the regularization term.

5. Theory

In this section, we introduce the pure-exploration non-stochastic infinite-armed bandit (NIAB) problem, a very general setting which encompasses our hyperparameter optimization problem of interest. As we will show, Hyperband is in fact applicable to problems far beyond just hyperparameter optimization. We begin by formalizing the hyperparameter optimization problem and then reducing it to the pure-exploration NIAB problem. We subsequently present a detailed analysis of Hyperband in both the infinite and finite horizon settings.

5.1 Hyperparameter Optimization Problem Statement

Let $\mathcal{X}$ denote the space of valid hyperparameter configurations, which could include continuous, discrete, or categorical variables that can be constrained with respect to each other in arbitrary ways (i.e. $\mathcal{X}$ need not be limited to a subset of $[0,1]^d$). For $k = 1, 2, \dots$ let $\ell_k \colon \mathcal{X} \to [0,1]$ be a sequence of loss functions defined over $\mathcal{X}$. For any hyperparameter configuration $x \in \mathcal{X}$, $\ell_k(x)$ represents the validation error of the model trained using x with k units of resources (e.g. iterations). In addition, for some $R \in \mathbb{N} \cup \{\infty\}$, define $\ell_* = \lim_{k \to R} \ell_k$ and $\nu_* = \inf_{x \in \mathcal{X}} \ell_*(x)$. Note that $\ell_k(\cdot)$ for all $k \in \mathbb{N}$, $\ell_*(\cdot)$, and ν_* are all unknown to the algorithm a priori. In particular, it is uncertain how quickly $\ell_k(x)$ varies as a function of x for any fixed k, and how quickly $\ell_k(x) \to \ell_*(x)$ as a function of k for any fixed $x \in \mathcal{X}$.

We assume hyperparameter configurations are sampled randomly from a known probability distribution $p(x) \colon \mathcal{X} \to [0, \infty)$, with support on $\mathcal{X}$. In our experiments, $p(x)$ is simply the uniform distribution, but the algorithm can be used with any sampling method. If $X \in \mathcal{X}$ is a random sample from this probability distribution, then $\ell_*(X)$ is a random variable whose distribution is unknown since $\ell_*(\cdot)$ is unknown. Additionally, since it is unknown how $\ell_k(x)$ varies as a function of x or k, one cannot necessarily infer anything about $\ell_k(x)$ given knowledge of $\ell_j(y)$ for any $j \in \mathbb{N}$, $y \in \mathcal{X}$. As a consequence, we reduce the hyperparmeter optimization problem down to a much simpler problem that ignores all underlying structure of the hyperparameters: we only interact with some $x \in \mathcal{X}$ through its loss sequence $\ell_k(x)$ for $k = 1, 2, \dots$. With this reduction, the particular value of $x \in \mathcal{X}$ does nothing more than index or uniquely identify the loss sequence.

Without knowledge of how fast $\ell_k(\cdot) \to \ell_*(\cdot)$ or how $\ell_*(X)$ is distributed, the goal of Hyperband is to identify a hyperparameter configuration $x \in \mathcal{X}$ that minimizes $\ell_*(x) - \nu_*$ by drawing as many random configurations as desired while using as few total resources as possible.

5.2 The Pure-Exploration Non-stochastic Infinite-Armed Bandit Problem

We now formally define the bandit problem of interest, and relate it to the problem of hyperparameter optimization. Each "arm" in the NIAB game is associated with a sequence that is drawn randomly from a distribution over sequences. If we "pull" the ith drawn arm exactly k times, we observe a loss $\ell_{i,k}$. At each time, the player can either draw a new arm (sequence) or pull a previously drawn arm an additional time. There is no limit on the number of arms that can be drawn. We assume the arms are identifiable only by their index

i (i.e. we have no side-knowledge or feature representation of an arm), and we also make the following two additional assumptions:

Assumption 1 *For each $i \in \mathbb{N}$ the limit $\lim_{k\to\infty} \ell_{i,k}$ exists and is equal to ν_i.*[11]

Assumption 2 *Each ν_i is a bounded i.i.d. random variable with cumulative distribution function F.*

The objective of the NIAB problem is to identify an arm $\hat{\imath}$ with small $\nu_{\hat{\imath}}$ using as few total pulls as possible. We are interested in characterizing $\nu_{\hat{\imath}}$ as a function of the total number of pulls from all the arms. Clearly, the hyperparameter optimization problem described above is an instance of the NIAB problem. In this case, arm i correspondes to a configuration $x_i \in \mathcal{X}$, with $\ell_{i,k} = \ell_k(x_i)$; Assumption 1 is equivalent to requiring that $\nu_i = \ell_*(x_i)$ exists; and Assumption 2 follows from the fact that the arms are drawn i.i.d. from $\mathcal{X}$ according to distribution function $p(x)$. F is simply the cumulative distribution function of $\ell_*(X)$, where X is a random variable drawn from the distribution $p(x)$ over $\mathcal{X}$. Note that since the arm draws are independent, the ν_i's are also independent. Again, this is not to say that the validation losses do not depend on the settings of the hyperparameters; the validation loss could well be correlated with certain hyperparameters, but this is not used in the algorithm and no assumptions are made regarding the correlation structure.

In order to analyze the behavior of Hyperband in the NIAB setting, we must define a few additional objects. Let $\nu_* = \inf\{m : \mathbb{P}(\nu \le m) > 0\} > -\infty$, since the domain of the distribution F is bounded. Hence, the cumulative distribution function F satisfies

$$\mathbb{P}(\nu_i - \nu_* \le \epsilon) = F(\nu_* + \epsilon) \tag{1}$$

and let $F^{-1}(y) = \inf_x\{x : F(x) \le y\}$. Define $\gamma\colon \mathbb{N} \to \mathbb{R}$ as the pointwise smallest, monotonically decreasing function satisfying

$$\sup_i |\ell_{i,j} - \ell_{i,*}| \le \gamma(j)\ , \quad \forall j \in \mathbb{N}. \tag{2}$$

The function γ is guaranteed to exist by Assumption 1 and bounds the deviation from the limit value as the sequence of iterates j increases. For hyperparameter optimization, this follows from the fact that ℓ_k uniformly converges to ℓ_* for all $x \in \mathcal{X}$. In addition, γ can be interpretted as the deviation of the validation error of a configuration trained on a subset of resources versus the maximum number of allocatable resources. Finally, define R as the first index such that $\gamma(R) = 0$ if it exists, otherwise set $R = \infty$. For $y \ge 0$ let $\gamma^{-1}(y) = \min\{j \in \mathbb{N}\colon \gamma(j) \le y\}$, using the convention that $\gamma^{-1}(0) := R$ which we recall can be infinite.

As previously discussed, there are many real-world scenarios in which R is finite and known. For instance, if increasing subsets of the full data set is used as a resource, then the maximum number of resources cannot exceed the full data set size, and thus $\gamma(k) = 0$ for all $k \ge R$ where R is the (known) full size of the data set. In other cases such as iterative training problems, one might not want to or know how to bound R. We separate these two settings into the *finite horizon* setting where R is finite and known, and the *infinite horizon*

[11] We can always define $\ell_{i,k}$ so that convergence is guaranteed, i.e. taking the infimum of a sequence.

```
SuccessiveHalving (Infinite horizon)
Input: Budget $B$, $n$ arms where $\ell_{i,k}$ denotes the $k$th loss from the $i$th arm
Initialize: $S_0 = [n]$.
For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$
    Pull each arm in $S_k$ for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ times.
    Keep the best $\lfloor |S_k|/2 \rfloor$ arms in terms of the $r_k$th observed loss as $S_{k+1}$.
Output : $\hat{\imath}$, $\ell_{\hat{\imath}, \lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor}$ where $\hat{\imath} = S_{\lceil \log_2(n) \rceil}$
```

```
Hyperband (Infinite horizon)
Input: None
For $k = 1, 2, \ldots$
    For $s \in \mathbb{N}$ s.t. $k - s \geq \log_2(s)$
        $B_{k,s} = 2^k$, $n_{k,s} = 2^s$
        $\hat{\imath}_{k,s}, \ell_{\hat{\imath}_{k,s}, \lfloor \frac{2^{k-1}}{s} \rfloor} \leftarrow$ SuccessiveHalving $(B_{k,s}, n_{k,s})$
```

Figure 9: (Top) The SuccessiveHalving algorithm proposed and analyzed in Jamieson and Talwalkar (2015) for the non-stochastic setting. Note this algorithm was originally proposed for the stochastic setting in Karnin et al. (2013). (Bottom) The Hyperband algorithm for the infinite horizon setting. Hyperband calls SuccessiveHalving as a subroutine.

setting where no bound on R is known and it is assumed to be infinite. While our empirical results suggest that the finite horizon may be more practically relevant for the problem of hyperparameter optimization, the infinite horizon case has natural connections to the literature, and we begin by analyzing this setting.

5.3 Infinite Horizon Setting ($R = \infty$)

Consider the Hyperband algorithm of Figure 9. The algorithm uses SuccessiveHalving (Figure 9) as a subroutine that takes a finite set of arms as input and outputs an estimate of the best performing arm in the set. We first analyze SuccessiveHalving (SH) for a given set of limits ν_i and then consider the performance of SH when ν_i are drawn randomly according to F. We then analyze the Hyperband algorithm. We note that the algorithm of Figure 9 was originally proposed by Karnin et al. (2013) for the stochastic setting. However, Jamieson and Talwalkar (2015) analyzed it in the non-stochastic setting and also found it to work well in practice. Extending the result of Jamieson and Talwalkar (2015) we have the following theorem:

Theorem 1 *Fix n arms. Let $\nu_i = \lim_{\tau\to\infty} \ell_{i,\tau}$ and assume $\nu_1 \leq \cdots \leq \nu_n$. For any $\epsilon > 0$ let*

$$\begin{aligned} z_{SH} &= 2\lceil \log_2(n) \rceil \max_{i=2,\dots,n} i\,(1 + \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)) \\ &\leq 2\lceil \log_2(n) \rceil \big(n + \sum_{i=1,\dots,n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)\big) \end{aligned}$$

If the SuccessiveHalving *algorithm of Figure 9 is run with any budget $B > z_{SH}$ then an arm $\hat{\imath}$ is returned that satisfies $\nu_{\hat{\imath}} - \nu_1 \leq \epsilon/2$. Moreover, $|\ell_{\hat{\imath},\lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_1| \leq \epsilon$.*

The next technical lemma will be used to characterize the problem dependent term $\sum_{i=1,\dots,n} \gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2}\right\}\right)$ when the sequences are drawn from a probability distribution.

Lemma 2 *Fix $\delta \in (0,1)$. Let $p_n = \frac{\log(2/\delta)}{n}$. For any $\epsilon \geq 4(F^{-1}(p_n) - \nu_*)$ define*

$$\mathbf{H}(F,\gamma,n,\delta,\epsilon) := 2n \int_{\nu_*+\epsilon/4}^{\infty} \gamma^{-1}(\tfrac{t-\nu_*}{4}) dF(t) + \left(\tfrac{4}{3}\log(2/\delta) + 2nF(\nu_* + \epsilon/4)\right)\gamma^{-1}\left(\tfrac{\epsilon}{16}\right)$$

and $\mathbf{H}(F,\gamma,n,\delta) := \mathbf{H}(F,\gamma,n,\delta,4(F^{-1}(p_n) - \nu_*))$ so that

$$\mathbf{H}(F,\gamma,n,\delta) = 2n \int_{p_n}^{1} \gamma^{-1}(\tfrac{F^{-1}(t)-\nu_*}{4}) dt + \tfrac{10}{3}\log(2/\delta)\gamma^{-1}\left(\tfrac{F^{-1}(p_n)-\nu_*}{4}\right).$$

For n arms with limits $\nu_1 \leq \cdots \leq \nu_n$ drawn from F, then

$$\nu_1 \leq F^{-1}(p_n) \quad \text{and} \quad \sum_{i=1}^{n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right) \leq \mathbf{H}(F,\gamma,n,\delta,\epsilon)$$

for any $\epsilon \geq 4(F^{-1}(p_n) - \nu_*)$ with probability at least $1 - \delta$.

Setting $\epsilon = 4(F^{-1}(p_n) - \nu_*)$ in Theorem 1 and using the result of Lemma 2 that $\nu_* \leq \nu_1 \leq \nu_* + (F^{-1}(p_n) - \nu_*)$, we immediately obtain the following corollary.

Corollary 3 *Fix $\delta \in (0,1)$ and $\epsilon \geq 4(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*)$. Let $B = 4\lceil \log_2(n) \rceil \mathbf{H}(F,\gamma,n,\delta,\epsilon)$ where $\mathbf{H}(F,\gamma,n,\delta,\epsilon)$ is defined in Lemma 2. If the* SuccessiveHalving *algorithm of Figure 9 is run with the specified B and n arm configurations drawn randomly according to F, then an arm $\hat{\imath} \in [n]$ is returned such that with probability at least $1 - \delta$ we have $\nu_{\hat{\imath}} - \nu_* \leq \left(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\right) + \epsilon/2$. In particular, if $B = 4\lceil \log_2(n) \rceil \mathbf{H}(F,\gamma,n,\delta)$ and $\epsilon = 4(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*)$ then $\nu_{\hat{\imath}} - \nu_* \leq 3\left(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\right)$ with probability at least $1 - \delta$.*

Note that for any fixed $n \in \mathbb{N}$ we have for any $\Delta > 0$

$$\mathbb{P}(\min_{i=1,\dots,n} \nu_i - \nu_* \geq \Delta) = (1 - F(\nu_* + \Delta))^n \approx e^{-nF(\nu_*+\Delta)}$$

which implies $\mathbb{E}[\min_{i=1,\dots,n} \nu_i - \nu_*] \approx F^{-1}(\frac{1}{n}) - \nu_*$. That is, n needs to be sufficiently large so that it is probable that a good limit is sampled. On the other hand, for any fixed n, Corollary 3 suggests that the total resource budget B needs to be large enough in order to overcome the rates of convergence of the sequences described by γ. Next, we relate SH to a naive approach that uniformly allocates resources to a fixed set of n arms.

5.3.1 Non-Adaptive Uniform Allocation

The non-adaptive uniform allocation strategy takes as inputs a budget B and n arms, allocates B/n to each of the arms, and picks the arm with the lowest loss. The following results allow us to compare with SuccessiveHalving.

Proposition 4 *Suppose we draw n random configurations from F, train each with $j = \min\{B/n, R\}$ iterations, and let $\hat{\imath} = \arg\min_{i=1,\dots,n} \ell_j(X_i)$. Without loss of generality assume $\nu_1 \leq \ldots \leq \nu_n$. If*

$$B \geq n\gamma^{-1}\left(\tfrac{1}{2}(F^{-1}(\tfrac{\log(1/\delta)}{n}) - \nu_*)\right) \tag{3}$$

then with probability at least $1-\delta$ we have $\nu_{\hat{\imath}} - \nu_* \leq 2\left(F^{-1}\left(\frac{\log(1/\delta)}{n}\right) - \nu_*\right)$. In contrast, there exists a sequence of functions ℓ_j that satisfy F and γ such that if

$$B \leq n\gamma^{-1}\left(2(F^{-1}(\tfrac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)\right)$$

then with probability at least δ, we have $\nu_{\hat{\imath}} - \nu_* \geq 2(F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)$, where c is a constant that depends on the regularity of F.

For any fixed n and sufficiently large B, Corollary 3 shows that SuccessiveHalving outputs an $\hat{\imath} \in [n]$ that satisfies $\nu_{\hat{\imath}} - \nu_* \lesssim F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*$ with probability at least $1-\delta$. This guarantee is similar to the result in Proposition 4. However, SuccessiveHalving achieves its guarantee as long as[12]

$$B \simeq \log_2(n)\left[\log(1/\delta)\gamma^{-1}\left(F^{-1}(\tfrac{\log(1/\delta)}{n}) - \nu_*\right) + n\int_{\frac{\log(1/\delta)}{n}}^{1} \gamma^{-1}(F^{-1}(t) - \nu_*)dt\right], \tag{4}$$

and this sample complexity may be substantially smaller than the budget required by uniform allocation shown in Eq. (3) of Proposition 4. Essentially, the first term in Eq. (4) represents the budget allocated to the constant number of arms with limits $\nu_i \approx F^{-1}(\frac{\log(1/\delta)}{n})$ while the second term describes the number of times the sub-optimal arms are sampled before discarded. The next section uses a particular parameterization for F and γ to help better illustrate the difference between the sample complexity of uniform allocation (Equation 3) versus that of SuccessiveHalving (Equation 4).

5.3.2 A Parameterization of F and γ for Interpretability

To gain some intuition and relate the results back to the existing literature, we make explicit parametric assumptions on F and γ. We stress that all of our results hold for general F and γ as previously stated, and this parameterization is simply a tool to provide intuition. First assume that there exists a constant $\alpha > 0$ such that

$$\gamma(j) \simeq \left(\frac{1}{j}\right)^{1/\alpha}. \tag{5}$$

[12] We say $f \simeq g$ if there exist constants c, c' such that $cg(x) \leq f(x) \leq c'g(x)$.

Note that a large value of α implies that the convergence of $\ell_{i,k} \to \nu_i$ is very slow.

We will consider two possible parameterizations of F. First, assume there exists positive constants β such that

$$F(x) \simeq \begin{cases} (x - \nu_*)^\beta & \text{if } x \geq \nu_* \\ 0 & \text{if } x < \nu_* \end{cases}. \tag{6}$$

Here, a large value of β implies that it is very rare to draw a limit close to the optimal value ν_*. The same model was studied in Carpentier and Valko (2015). Fix some $\Delta > 0$. As discussed in the preceding section, if $n = \frac{\log(1/\delta)}{F(\nu_* + \Delta)} \simeq \Delta^{-\beta} \log(1/\delta)$ arms are drawn from F then with probability at least $1-\delta$ we have $\min_{i=1,\dots,n} \nu_i \leq \nu_* + \Delta$. Predictably, both uniform allocation and SuccessiveHalving output a $\nu_{\hat{\imath}}$ that satisfies $\nu_{\hat{\imath}} - \nu_* \lesssim \left(\frac{\log(1/\delta)}{n}\right)^{1/\beta}$ with probability at least $1-\delta$ provided their measurement budgets are large enough. Thus, if $n \simeq \Delta^{-\beta}\log(1/\delta)$ and the measurement budgets of the uniform allocation (Equation 3) and SuccessiveHalving (Equation 4) satisfy

$$\begin{aligned} &\text{Uniform allocation} && B \simeq \Delta^{-(\alpha+\beta)} \log(1/\delta) \\ &\textsc{SuccessiveHalving} && B \simeq \log_2(\Delta^{-\beta}\log(1/\delta)) \left[\Delta^{-\alpha}\log(1/\delta) + \frac{\Delta^{-\beta} - \Delta^{-\alpha}}{1 - \alpha/\beta} \log(1/\delta)\right] \\ & && \simeq \log(\Delta^{-1}\log(1/\delta)) \log(\Delta^{-1})\ \Delta^{-\max\{\beta,\alpha\}} \log(1/\delta) \end{aligned}$$

then both also satisfy $\nu_{\hat{\imath}} - \nu_* \lesssim \Delta$ with probability at least $1-\delta$.[13] SuccessiveHalving's budget scales like $\Delta^{-\max\{\alpha,\beta\}}$, which can be significantly smaller than the uniform allocation's budget of $\Delta^{-(\alpha+\beta)}$. However, because α and β are unknown in practice, neither method knows how to choose the optimal n or B to achieve this Δ accuracy. In Section 5.3.3, we show how Hyperband addresses this issue.

The second parameterization of F is the following discrete distribution:

$$F(x) = \frac{1}{K} \sum_{j=1}^{K} \mathbf{1}\{x \leq \mu_j\} \qquad \text{with} \quad \Delta_j := \mu_j - \mu_1 \tag{7}$$

for some set of unique scalars $\mu_1 < \mu_2 < \dots < \mu_K$. Note that by letting $K \to \infty$ this discrete CDF can approximate any piecewise-continuous CDF to arbitrary accuracy. In particular, this model can have multiple means take the same value so that α mass is on μ_1 and $1-\alpha$ mass is on $\mu_2 > \mu_1$, capturing the stochastic infinite-armed bandit model of Jamieson et al. (2016). In this setting, both uniform allocation and SuccessiveHalving output a $\nu_{\hat{\imath}}$ that is within the top $\frac{\log(1/\delta)}{n}$ fraction of the K arms with probability at least $1-\delta$ if their budgets are sufficiently large. Thus, let $q > 0$ be such that $n \simeq q^{-1}\log(1/\delta)$. Then, if the measurement budgets of the uniform allocation (Equation 3) and SuccessiveHalving

[13] These quantities are intermediate results in the proofs of the theorems of Section 5.3.3.

(Equation 4) satisfy

$$\text{Uniform allocation} \quad B \simeq \log(1/\delta) \begin{cases} K \max_{j=2,\dots,K} \Delta_j^{-\alpha} & \text{if } q = 1/K \\ q^{-1} \Delta_{\lceil qK \rceil}^{-\alpha} & \text{if } q > 1/K \end{cases}$$

$$\textsc{SuccessiveHalving} \quad B \simeq \log(q^{-1}\log(1/\delta))\log(1/\delta) \begin{cases} \Delta_2^{-\alpha} + \sum_{j=2}^{K} \Delta_j^{-\alpha} & \text{if } q = 1/K \\ \Delta_{\lceil qK \rceil}^{-\alpha} + \frac{1}{qK} \sum_{j=\lceil qK \rceil}^{K} \Delta_j^{-\alpha} & \text{if } q > 1/K, \end{cases}$$

an arm that is in the best q-fraction of arms is returned, i.e. $\hat{\imath}/K \approx q$ and $\nu_{\hat{\imath}} - \nu_* \lesssim \Delta_{\lceil \max\{2,qK\} \rceil}$, with probability at least $1-\delta$. This shows that the average resource per arm for uniform allocation is that required to distinguish the top q-fraction from the best, while that for SuccessiveHalving is a small multiple of the average resource required to distinguish an arm from the best; the difference between the max and the average can be very large in practice. We remark that the value of ϵ in Corollary 3 is carefully chosen to make the SuccessiveHalving budget and guarantee work out. Also note that one would never take $q < 1/K$ because $q = 1/K$ is sufficient to return the best arm.

5.3.3 Hyperband Guarantees

The Hyperband algorithm of Figure 9 addresses the tradeoff between the number of arms n versus the average number of times each one is pulled B/n by performing a two-dimensional version of the so-called "doubling trick." For each fixed B, we non-adaptively search a predetermined grid of values of n spaced geometrically apart so that the incurred loss of identifying the "best" setting takes a budget no more than $\log(B)$ times the budget necessary if the best setting of n were known ahead of time. Then, we successively double B so that the cumulative number of measurements needed to arrive at the necessary B is no more than $2B$. The idea is that even though we do not know the optimal setting for B, n to achieve some desired error rate, the hope is that by trying different values in a particular order, we will not waste too much effort.

Fix $\delta \in (0,1)$. For all (k,s) pairs defined in the Hyperband algorithm of Figure 9, let $\delta_{k,s} = \frac{\delta}{2k^3}$. For all (k,s) define

$$\mathcal{E}_{k,s} := \{B_{k,s} > 4\lceil \log_2(n_{k,s}) \rceil \mathbf{H}(F,\gamma,n_{k,s},\delta_{k,s})\} = \{2^k > 4s\mathbf{H}(F,\gamma,2^s,2k^3/\delta)\}$$

Then by Corollary 3 we have

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} \bigcup_{s=1}^{k} \{\nu_{\hat{\imath}_{k,s}} - \nu_* > 3\big(F^{-1}(\tfrac{\log(4k^3/\delta)}{2^s}) - \nu_*\big)\} \cap \mathcal{E}_{k,s} \right) \le \sum_{k=1}^{\infty} \sum_{s=1}^{k} \delta_{k,s} = \sum_{k=1}^{\infty} \frac{\delta}{2k^2} \le \delta.$$

For sufficiently large k we will have $\bigcup_{s=1}^{k} \mathcal{E}_{k,s} \neq \emptyset$, so assume $B = 2^k$ is sufficiently large. Let $\hat{\imath}_B$ be the empirically best-performing arm output from SuccessiveHalving of round $k_B = \lfloor \log_2(B) \rfloor$ of Hyperband of Figure 9 and let $s_B \le k_B$ be the largest value such that

$\mathcal{E}_{k_B,s_B}$ holds. Then

$$\nu_{\hat{i}_B} - \nu_* \leq 3\big(F^{-1}(\frac{\log(4\lfloor\log_2(B)\rfloor^3/\delta)}{2^{s_B}}) - \nu_*\big) + \gamma(\lfloor \tfrac{2^{\lfloor \log_2(B)\rfloor - 1}}{\lfloor \log_2(B)\rfloor} \rfloor).$$

Also note that on stage k at most $\sum_{i=1}^k iB_{i,1} \leq k\sum_{i=1}^k B_{i,1} \leq 2kB_{k,s} = 2\log_2(B_{k,s})B_{k,s}$ total samples have been taken. While this guarantee holds for general F, γ, the value of s_B, and consequently the resulting bound, is difficult to interpret. The following corollary considers the β, α parameterizations of F and γ, respectively, of Section 5.3.2 for better interpretation.

Theorem 5 *Assume that Assumptions 1 and 2 of Section 5.2 hold and that the sampled loss sequences obey the parametric assumptions of Equations 5 and 6. Fix* $\delta \in (0,1)$. *For any* $T \in \mathbb{N}$, *let* $\hat{i}_T$ *be the empirically best-performing arm output from* SuccessiveHalving *from the last round* k *of* Hyperband *of Figure 9 after exhausting a total budget of* T *from all rounds, then*

$$\nu_{\hat{i}_T} - \nu_* \leq c\left(\frac{\overline{\log}(T)^3\overline{\log}(\log(T)/\delta)}{T}\right)^{1/\max\{\alpha,\beta\}}$$

for some constant $c = \exp(O(\max\{\alpha,\beta\}))$ *where* $\overline{\log}(x) = \log(x)\log\log(x)$.

By a straightforward modification of the proof, one can show that if uniform allocation is used in place of SuccessiveHalving in Hyperband, the uniform allocation version achieves $\nu_{\hat{i}_T} - \nu_* \leq c\left(\frac{\log(T)\overline{\log}(\log(T)/\delta)}{T}\right)^{1/(\alpha+\beta)}$. We apply the above theorem to the stochastic infinite-armed bandit setting in the following corollary.

Corollary 6 *[Stochastic Infinite-armed Bandits] For any step* k, s *in the infinite horizon* Hyperband *algorithm with* $n_{k,s}$ *arms drawn, consider the setting where the* j*th pull of the* i*th arm results in a stochastic loss* $Y_{i,j} \in [0,1]$ *such that* $\mathbb{E}[Y_{i,j}] = \nu_i$ *and* $\mathbb{P}(\nu_i - \nu_* \leq \epsilon) = c_1^{-1}\epsilon^\beta$. *If* $\ell_j(i) = \frac{1}{j}\sum_{s=1}^j Y_{i,s}$ *then with probability at least* $1 - \delta/2$ *we have* $\forall k \geq 1, 0 \leq s \leq k, 1 \leq i \leq n_{k,s}, 1 \leq j \leq B_k$,

$$|\nu_i - \ell_{i,j}| \leq \sqrt{\tfrac{\log(B_k n_{k,s}/\delta_{k,s})}{2j}} \leq \sqrt{\log(\tfrac{16B_k}{\delta})}\left(\tfrac{2}{j}\right)^{1/2}.$$

Consequently, if after B *total pulls we define* $\widehat{\nu}_B$ *as the mean of the empirically best arm output from the last fully completed round* k, *then with probability at least* $1 - \delta$

$$\widehat{\nu}_B - \nu_* \leq \operatorname{polylog}(B/\delta)\max\{B^{-1/2}, B^{-1/\beta}\}.$$

The result of this corollary matches the anytime result of Section 4.3 of Carpentier and Valko (2015) whose algorithm was built specifically for the case of stochastic arms and the β parameterization of F defined in Eq. (6). Notably, this result also matches the *lower bounds* shown in that work up to poly-logarithmic factors, revealing that Hyperband is nearly tight for this important special case. However, we note that this earlier work has a more careful analysis for the fixed budget setting.

Theorem 7 *Assume that Assumptions 1 and 2 of Section 5.2 hold and that the sampled loss sequences obey the parametric assumptions of Equations 5 and 7. For any $T \in \mathbb{N}$, let $\hat{\imath}_T$ be the empirically best-performing arm output from* SuccessiveHalving *from the last round k of* Hyperband *of Figure 9 after exhausting a total budget of T from all rounds. Fix $\delta \in (0,1)$ and $q \in (1/K, 1)$ and let $z_q = \log(q^{-1})(\Delta^{-\alpha}_{\lceil \max\{2,qK\}\rceil} + \frac{1}{qK}\sum_{i=\lceil \max\{2,qK\}\rceil}^{K} \Delta_i^{-\alpha})$. Once $T = \widetilde{\Omega}\left(z_q \log(z_q) \log(1/\delta)\right)$ total pulls have been made by* Hyperband *we have $\widehat{\nu}_T - \nu_* \leq \Delta_{\lceil \max\{2,qK\}\rceil}$ with probability at least $1-\delta$ where $\widetilde{\Omega}(\cdot)$ hides $\log\log(\cdot)$ factors.*

Appealing to the stochastic setting of Corollary 6 so that $\alpha = 2$, we conclude that the sample complexity sufficient to identify an arm within the best q proportion with probabiltiy $1-\delta$, up to log factors, scales like $\log(1/\delta)\log(q^{-1})(\Delta^{-\alpha}_{\lceil qK\rceil} + \frac{1}{qK}\sum_{i=\lceil qK\rceil}^{K} \Delta_i^{-\alpha})$. One may interpret this result as an extension of the distribution-dependent pure-exploration results of Bubeck et al. (2009); but in our case, our bounds hold when the number of pulls is potentially much smaller than the number of arms K. When $q = 1/K$ this implies that the best arm is identified with about $\log(1/\delta)\log(K)\{\Delta_2^{-2} + \sum_{i=2}^{K}\Delta_i^{-2}\}$ which matches known upper bounds Karnin et al. (2013); Jamieson et al. (2014) and lower bounds Kaufmann et al. (2015) up to log factors. Thus, for the stochastic K-armed bandit problem Hyperband recovers many of the known sample complexity results up to log factors.

5.4 Finite Horizon Setting ($R < \infty$)

In this section we analyze the algorithm described in Section 3, i.e. finite horizon Hyperband. We present similar theoretical guarantees as in Section 5.3 for infinite horizon Hyperband, and fortunately much of the analysis will be recycled. We state the finite horizon version of the SuccessiveHalving and Hyperband algorithms in Figure 10.

The finite horizon setting differs in two major ways. First, in each bracket at least one arm will be pulled R times, but no arm will be pulled more than R times. Second, the number of brackets, each representing SuccessiveHalving with a different tradeoff between n and B, is fixed at $\log_\eta(R) + 1$. Hence, since we are sampling sequences randomly i.i.d., increasing B over time would just multiply the number of arms in each bracket by a constant, affecting performance only by a small constant.

Theorem 8 *Fix n arms. Let $\nu_i = \ell_{i,R}$ and assume $\nu_1 \leq \cdots \leq \nu_n$. For any $\epsilon > 0$ let*

$$z_{SH} = \eta(\log_\eta(R) + 1)\Big[n + \sum_{i=1}^{n} \min\left\{R, \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)\right\}\Big]$$

If the Successive Halving algorithm of Figure 10 is run with any budget $B \geq z_{SH}$ then an arm $\hat{\imath}$ is returned that satisfies $\nu_{\hat{\imath}} - \nu_1 \leq \epsilon/2$.

Recall that $\gamma(R) = 0$ in this setting and by definition $\sup_{y \geq 0}\gamma^{-1}(y) \leq R$. Note that Lemma 2 still applies in this setting and just like above we obtain the following corollary.

Corollary 9 *Fix $\delta \in (0,1)$ and $\epsilon \geq 4(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*)$. Let $\mathbf{H}(F, \gamma, n, \delta, \epsilon)$ be as defined in Lemma 2 and $B = \eta\log_\eta(R)(n + \max\{R, \mathbf{H}(F, \gamma, n, \delta, \epsilon)\})$. If the* SuccessiveHalving *algorithm of Figure 10 is run with the specified B and n arm configurations drawn randomly*

```
SuccessiveHalving (Finite horizon)
input: Budget $B$, and $n$ arms where $\ell_{i,k}$ denotes the $k$th loss from the $i$th arm,
maximum size $R$, $\eta \geq 2$ ($\eta = 3$ by default).
Initialize: $S_0 = [n]$, $s = \min\{t \in \mathbb{N} : nR(t+1)\eta^{-t} \leq B, t \leq \log_\eta(\min\{R, n\})\}$.
For $k = 0, 1, \ldots, s$
    Set $n_k = \lfloor n\eta^{-k} \rfloor$, $r_k = \lfloor R\eta^{k-s} \rfloor$
    Pull each arm in $S_k$ for $r_k$ times.
    Keep the best $\lfloor n\eta^{-(k+1)} \rfloor$ arms in terms of the $r_k$th observed loss as $S_{k+1}$.
Output : $\hat{\imath}, \ell_{\hat{\imath},R}$ where $\hat{\imath} = \arg\min_{i \in S_{s+1}} \ell_{i,R}$
```

```
Hyperband (Finite horizon)
Input: Budget $B$, maximum size $R$, $\eta \geq 2$ ($\eta = 3$ by default)
Initialize: $s_{\max} = \lfloor \log(R)/\log(\eta) \rfloor$
For $k = 1, 2, \ldots$
    For $s = s_{\max}, s_{\max} - 1, \ldots, 0$
        $B_{k,s} = 2^k$, $n_{k,s} = \lceil \frac{2^k \eta^s}{R(s+1)} \rceil$
        $\hat{\imath}_s, \ell_{\hat{\imath}_s,R} \leftarrow$ SuccessiveHalving $(B_{k,s}, n_{k,s}, R, \eta)$
```

Figure 10: The finite horizon SuccessiveHalving and Hyperband algorithms are inspired by their infinite horizon counterparts of Figure 9 to handle practical constraints. Hyperband calls SuccessiveHalving as a subroutine.

according to F then an arm $\hat{\imath} \in [n]$ is returned such that with probability at least $1 - \delta$ we have $\nu_{\hat{\imath}} - \nu_* \leq \big(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\big) + \epsilon/2$. In particular, if $B = 4\lceil \log_2(n) \rceil \mathbf{H}(F, \gamma, n, \delta)$ and $\epsilon = 4(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*)$ then $\nu_{\hat{\imath}} - \nu_* \leq 3\big(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\big)$ with probability at least $1 - \delta$.

As in Section 5.3.2 we can apply the α, β parameterization for interpretability, with the added constraint that $\sup_{y \geq 0} \gamma^{-1}(y) \leq R$ so that $\gamma(j) \simeq \mathbf{1}_{j<R} \left(\frac{1}{j}\right)^{1/\alpha}$. Note that the approximate sample complexity of SuccessiveHalving given in Eq. (4) is still valid for the finite horizon algorithm.

Fixing some $\Delta > 0$, $\delta \in (0, 1)$, and applying the parameterization of Eq. (6) we recognize that if $n \simeq \Delta^{-\beta} \log(1/\delta)$ and the sufficient sampling budgets (treating η as an absolute constant) of the uniform allocation (Equation 3) and SuccessiveHalving (Eq. (4)) satisfy

$$\text{Uniform allocation} \qquad B \simeq R\Delta^{-\beta} \log(1/\delta)$$

$$\text{SuccessiveHalving} \qquad B \simeq \log(\Delta^{-1} \log(1/\delta)) \log(1/\delta) \left[R + \Delta^{-\beta} \frac{1 - (\alpha/\beta)R^{1-\beta/\alpha}}{1 - \alpha/\beta} \right]$$

then both also satisfy $\nu_{\hat{\imath}} - \nu_* \lesssim \Delta$ with probability at least $1 - \delta$. Recalling that a larger α means slower convergence and that a larger β means a greater difficulty of sampling a good limit, note that when $\alpha/\beta < 1$ the budget of SuccessiveHalving behaves like $R + \Delta^{-\beta} \log(1/\delta)$ but as $\alpha/\beta \to \infty$ the budget asymptotes to $R\Delta^{-\beta} \log(1/\delta)$.

We can also apply the discrete-CDF parameterization of Eq. (7). For any $q \in (0,1)$, if $n \simeq q^{-1}\log(1/\delta)$ and the measurement budgets of the uniform allocation (Equation 3) and SuccessiveHalving (Equation 4) satisfy

$$\text{Uniform allocation:} \qquad B \simeq \log(1/\delta) \begin{cases} K \min\left\{R, \max\limits_{j=2,\dots,K} \Delta_j^{-\alpha}\right\} & \text{if } q = 1/K \\ q^{-1}\min\{R, \Delta_{\lceil qK \rceil}^{-\alpha}\} & \text{if } q > 1/K \end{cases}$$

SuccessiveHalving:

$$B \simeq \log(q^{-1}\log(1/\delta))\log(1/\delta) \begin{cases} \min\{R, \Delta_2^{-\alpha}\} + \sum\limits_{j=2}^{K} \min\{R, \Delta_j^{-\alpha}\} & \text{if } q = 1/K \\ \min\{R, \Delta_{\lceil qK \rceil}^{-\alpha}\} + \frac{1}{qK} \sum\limits_{j=\lceil qK \rceil}^{K} \min\{R, \Delta_j^{-\alpha}\} & \text{if } q > 1/K \end{cases}$$

then an arm that is in the best q-fraction of arms is returned, i.e. $\hat{\imath}/K \approx q$ and $\nu_{\hat{\imath}} - \nu_* \lesssim \Delta_{\lceil \max\{2, qK\} \rceil}$, with probability at least $1 - \delta$. Once again we observe a stark difference between uniform allocation and SuccessiveHalving, particularly when $\Delta_j^{-\alpha} \ll R$ for many values of $j \in \{1, \dots, n\}$.

Armed with Corollary 9, all of the discussion of Section 5.3.3 preceding Theorem 5 holds for the finite case ($R < \infty$) as well. Predictably analogous theorems also hold for the finite horizon setting, but their specific forms (with the polylog factors) provide no additional insights beyond the sample complexities sufficient for SuccessiveHalving to succeed, given immediately above.

It is important to note that in the finite horizon setting, for all sufficiently large B (e.g. $B > 3R$) and all distributions F, the budget B of SuccessiveHalving should scale *linearly* with $n \simeq \Delta^{-\beta}\log(1/\delta)$ as $\Delta \to 0$. Contrast this with the infinite horizon setting in which the ratio of B to n can become unbounded based on the values of α, β as $\Delta \to 0$. One consequence of this observation is that in the finite horizon setting it suffices to set B large enough to identify an Δ-good arm with just constant probability, say $1/10$, and then repeat SuccessiveHalving m times to boost this constant probability to probability $1 - (\frac{9}{10})^m$. While in this theoretical treatment of Hyperband we grow B over time, in practice we recommend fixing B as a multiple of R as we have done in Section 3. The fixed budget version of finite horizon Hyperband is more suitable for practical application due to the constant time, instead of exponential time, between configurations trained to completion in each outer loop.

6. Conclusion

We conclude by discussing three potential extensions related to parallelizing Hyperband for distributed computing, adjusting for training methods with different convergence rates, and combining Hyperband with non-random sampling methods.

Distributed implementations. Hyperband has the potential to be parallelized since arms are independent and sampled randomly. The most straightforward parallelization scheme is to distribute individual brackets of SuccessiveHalving to different machines. This can be done asynchronously and as machines free up, new brackets can be launched with a different set of arms. One can also parallelize a single bracket so that each round of SuccessiveHalving runs faster. One drawback of this method is that if R can be computed on one machine, the number of tasks decreases exponentially as arms are whittled down so a more sophisticated job priority queue must be managed. Devising parallel generalizations of Hyperband that efficiently leverage massive distributed clusters while minimizing overhead costs is an interesting avenue for future work.

Adjusting for different convergence rates. A second open challenge involves generalizing the ideas behind Hyperband to settings where configurations have drastically differing convergence rates. Configurations can have different convergence rates if they have hyperparameters that impact convergence (e.g., learning rate decay for SGD or neural networks with differing numbers of layers or hidden units), and/or if they correspond to different model families (e.g., deep networks versus decision trees). The core issue arises when configurations with drastically slower convergence rates ultimately result in better models. To address these issues, we should be able to adjust the resources allocated to each configuration so that a fair comparison can be made at the time of elimination.

Incorporating non-random sampling. Finally, Hyperband can benefit from different sampling schemes aside from simple random search. Quasi-random methods like Sobol or latin hypercube which were studied in Bergstra and Bengio (2012) may improve the performance of Hyperband by giving better coverage of the search space. Alternatively, meta-learning can be used to define intelligent priors informed by previous experimentation (Feurer et al., 2015). Finally, as mentioned in Section 2, exploring ways to combine Hyperband with adaptive configuration selection strategies is a very promising future direction.

Acknowledgments

KJ is supported by ONR awards N00014-15-1-2620 and N00014-13-1-0129. AT is supported in part by a Google Faculty Award and an AWS in Education Research Grant award.

Appendix A. Additional Experimental Results

Additional details for experiments presented in Section 3 and 4 are provided below.

A.1 LeNet Experiment

The search space for the LeNet example discussed in Section 3.3 is shown in Table 2.

Hyperparameter	Scale	Min	Max
Learning Rate	log	1e-3	1e-1
Batch size	log	1e1	1e3
Layer-2 Num Kernels (k2)	linear	10	60
Layer-1 Num Kernels (k1)	linear	5	k2

Table 2: Hyperparameter space for the LeNet application of Section 3.3. Note that the number of kernels in Layer-1 is upper bounded by the number of kernels in Layer-2.

A.2 Experiments Using Alex Krizhevsky's CNN Architecture

For the experiments discussed in Section 4.1, the exact architecture used is the 18% model provided on cuda-convnet for CIFAR-10.[14]

Search Space: The search space used for the experiments is shown in Table 3. The learning rate reductions hyperparameter indicates how many times the learning rate was reduced by a factor of 10 over the maximum iteration window. For example, on CIFAR-10, which has a maximum iteration of 30,000, a learning rate reduction of 2 corresponds to reducing the learning every 10,000 iterations, for a total of 2 reductions over the 30,000 iteration window. All hyperparameters, with the exception of the learning rate decay reduction, overlap with those in Snoek et al. (2012). Two hyperparameters in Snoek et al. (2012) were excluded from our experiments: (1) the width of the response normalization layer was excluded due to limitations of the Caffe framework and (2) the number of epochs was excluded because it is incompatible with dynamic resource allocation.

Data Splits: For CIFAR-10, the training (40,000 instances) and validation (10,000 instances) sets were sampled from data batches 1-5 with balanced classes. The original test set (10,000 instances) was used for testing. For MRBI, the training (10,000 instances) and validation (2,000 instances) sets were sampled from the original training set with balanced classes. The original test set (50,000 instances) was used for testing. Lastly, for SVHN, the train, validation, and test splits were created using the same procedure as that in Sermanet et al. (2012).

Comparison with Early-Stopping: Domhan et al. (2015) proposed an early-stopping method for neural networks and combined it with SMAC to speed up hyperparameter optimization. Their method stops training a configuration if the probability of the configuration beating the current best is below a specified threshold. This probability is estimated by extrapolating learning curves fit to the intermediate validation error losses of a configuration.

[14] The model specification is available at `http://code.google.com/p/cuda-convnet/`.

Hyperparameter	Scale	Min	Max
Learning Parameters			
Initial Learning Rate	log	$5 * 10^{-5}$	5
Conv1 L_2 Penalty	log	$5 * 10^{-5}$	5
Conv2 L_2 Penalty	log	$5 * 10^{-5}$	5
Conv3 L_2 Penalty	log	$5 * 10^{-5}$	5
FC4 L_2 Penalty	log	$5 * 10^{-3}$	500
Learning Rate Reductions	integer	0	3
Local Response Normalization			
Scale	log	$5 * 10^{-6}$	5
Power	linear	0.01	3

Table 3: Hyperparameters and associated ranges for the three-layer convolutional network.

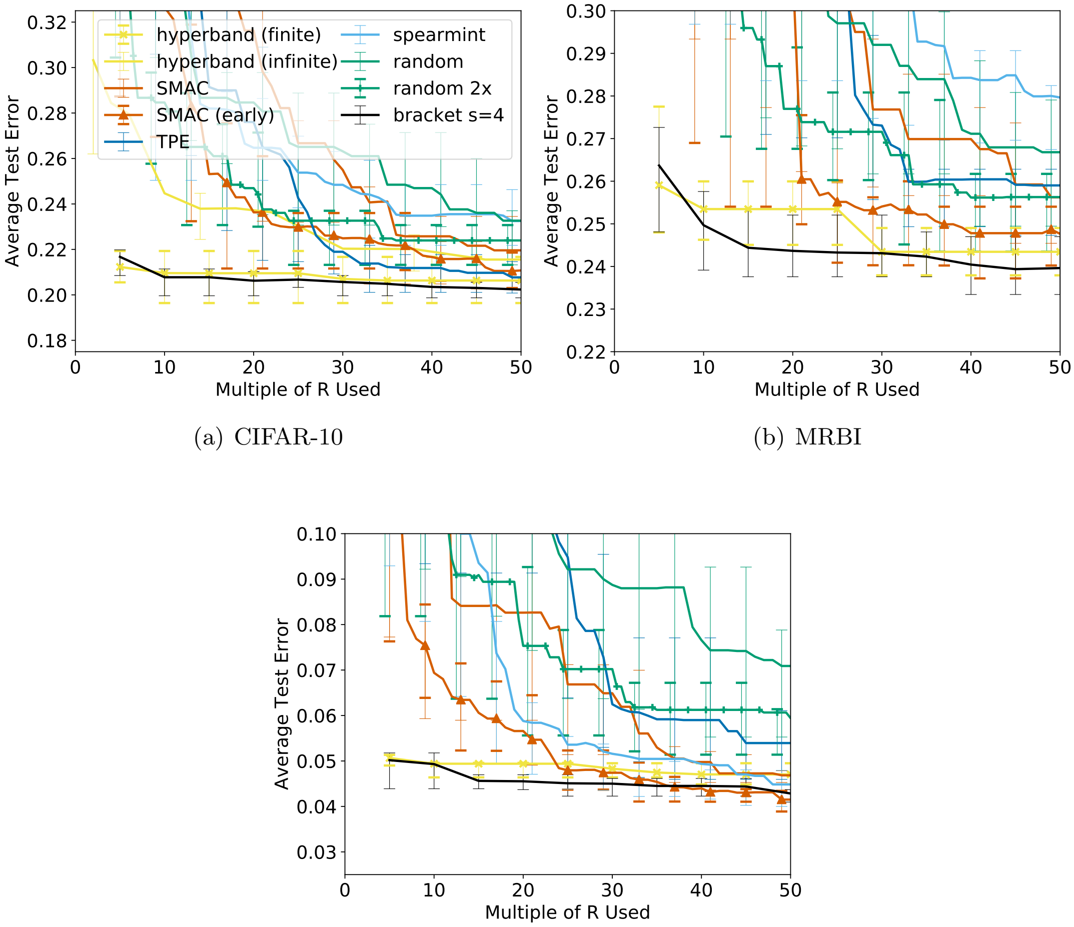
If a configuration is terminated early, the predicted terminal value from the estimated learning curves is used as the validation error passed to the hyperparameter optimization algorithm. Hence, if the learning curve fit is poor, it could impact the performance of the configuration selection algorithm. While this approach is heuristic in nature, it could work well in practice so we compare Hyperband to SMAC with early termination (labeled SMAC (early) in Figure 11). We used the conservative termination criterion with default parameters and recorded the validation loss every 400 iterations and evaluated the termination criterion 3 times within the training period (every 8k iterations for CIFAR-10 and MRBI and every 16k iterations for SVHN).[15] Comparing the performance by the number of total iterations as mulitple of R is conservative because it does not account for the time spent fitting the learning curve in order to check the termination criterion.

A.3 117 Data Sets Experiment

For the experiments discussed in Section 4.2.1, we followed Feurer et al. (2015) and imposed a 3GB memory limit, a 6-minute timeout for each hyperparameter configuration and a one-hour time window to evaluate each searcher on each data set. Moreover, we evaluated the performance of each searcher by aggregating results across all data sets and reporting the average rank of each method. Specifically, the hour training window is divided up into 30 second intervals and, at each time point, the model with the best validation error at that time is used in the calculation of the average error across all trials for each (data set-searcher) pair. Then, the performance of each searcher is ranked by data set and averaged across all data sets. All experiments were performed on Google Cloud Compute n1-standard-1 instances in us-central1-f region with 1 CPU and 3.75GB of memory.

Data Splits: Feurer et al. (2015) split each data set into 2/3 training and 1/3 test set, whereas we introduce a validation set to avoid overfitting to the test data. We also used 2/3 of the data for training, but split the rest of the data into two equally sized validation and test sets. We reported results on both the validation and test data. Moreover, we performed

[15] We used the code provided at https://github.com/automl/pylearningcurvepredictor.



(a) CIFAR-10

(b) MRBI

(c) SVHN

Figure 11: Average test error across 10 trials is shown in all plots. Error bars indicate the top and bottom quartiles of the test error corresponding to the model with the best validation error

20 trials of each (data set-searcher) pair, and as in Feurer et al. (2015) we kept the same data splits across trials, while using a different random seed for each searcher in each trial.

Shortcomings of the Experimental Setup: The benchmark contains a large variety of training set sizes and feature dimensions[16] resulting in random search being able to test 600 configurations on some data sets but just dozens on others. Hyperband was designed under the implicit assumption that computation scaled at least linearly with the data set size. For very small data sets that are trained in seconds, the initialization overheads dominate the computation and subsampling provides no computational benefit. In addition, many of the classifiers and preprocessing methods under consideration return memory errors as they require storage quadratic in the number of features (e.g., covariance matrix) or the number of observations (e.g., kernel methods). These errors usually happen immediately (thus wasting little time); however, they often occur on the full data set and not on subsampled data sets. A searcher like Hyperband that uses a subsampled data set could spend significant time training on a subsample only to error out when attempting to train it on the full data set.

A.4 Kernel Classification Experiments

Table 4 shows the hyperparameters and associated ranges considered in the kernel least squares classification experiment discussed in Section 4.2.2.

Hyperparameter	Type	Values
preprocessor	Categorical	min/max, standardize, normalize
kernel	Categorical	rbf, polynomial, sigmoid
C	Continuous	log $[10^{-3}, 10^5]$
gamma	Continuous	log $[10^{-5}, 10]$
degree	if kernel=poly	integer [2, 5]
coef0	if kernel=poly, sigmoid	uniform [-1.0, 1.0]

Table 4: Hyperparameter space for kernel regularized least squares classification problem discussed in Section 4.2.2.

The cost term C is divided by the number of samples so that the tradeoff between the squared error and the L_2 penalty would remain constant as the resource increased (squared error is summed across observations and not averaged). The regularization term λ is equal to the inverse of the scaled cost term C. Additionally, the average test error with the top and bottom quartiles across 10 trials are show in Figure 12.

Table 5 shows the hyperparameters and associated ranges considered in the random features kernel approximation classification experiment discussed in Section 4.3. The regularization term λ is divided by the number of features so that the tradeoff between the squared error and the L_2 penalty would remain constant as the resource increased. Additionally, the average test error with the top and bottom quartiles across 10 trials are show in Figure 13.

[16] Training set size ranges from 670 to 73,962 observations, and number of features ranges from 1 to 10,935.

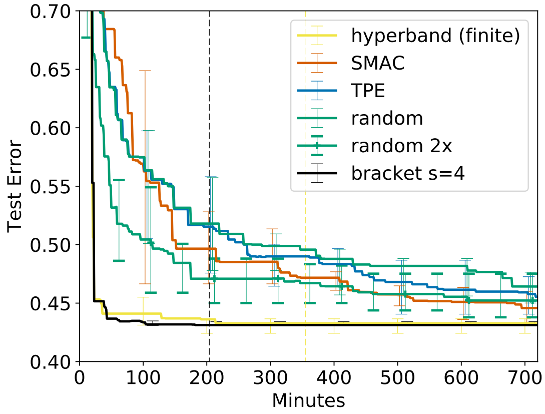


Figure 12: Average test error of the best kernel regularized least square classification model found by each searcher on CIFAR-10. The color coded dashed lines indicate when the last trial of a given searcher finished. Error bars correspond to the top and bottom quartiles of the test error across 10 trials.

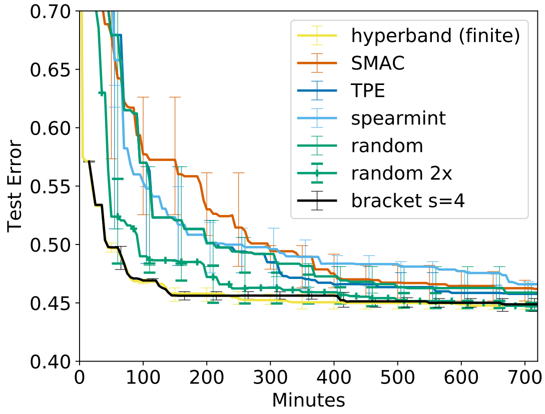


Figure 13: Average test error of the best random features model found by each searcher on CIFAR-10. The test error for Hyperband and bracket $s = 4$ are calculated in every evaluation instead of at the end of a bracket. Error bars correspond to the top and bottom quartiles of the test error across 10 trials.

Hyperparameter	Type	Values
preprocessor	Categorical	none, min/max, standardize, normalize
λ	Continuous	log $[10^{-3}, 10^5]$
gamma	Continuous	log $[10^{-5}, 10]$

Table 5: Hyperparameter space for random feature kernel approximation classification problem discussed in Section 4.3.

Appendix B. Proofs

In this section, we provide proofs for the theorems presented in Section 5.

B.1 Proof of Theorem 1

Proof First, we verify that the algorithm never takes a total number of samples that exceeds the budget B:

$$\sum_{k=0}^{\lceil \log_2(n) \rceil - 1} |S_k| \left\lfloor \tfrac{B}{|S_k| \lceil \log(n) \rceil} \right\rfloor \leq \sum_{k=0}^{\lceil \log_2(n) \rceil - 1} \tfrac{B}{\lceil \log(n) \rceil} \leq B \,.$$

For notational ease, let $\ell_{i,j} := \ell_j(X_i)$. Again, for each $i \in [n] := \{1, \dots, n\}$ we assume the limit $\lim_{k \to \infty} \ell_{i,k}$ exists and is equal to ν_i. As a reminder, $\gamma : \mathbb{N} \to \mathbb{R}$ is defined as the pointwise smallest, monotonically decreasing function satisfying

$$\max_i |\ell_{i,j} - \nu_i| \leq \gamma(j) \ , \quad \forall j \in \mathbb{N}. \tag{8}$$

Note γ is guaranteed to exist by the existence of ν_i and bounds the deviation from the limit value as the sequence of iterates j increases.

Without loss of generality, order the terminal losses so that $\nu_1 \leq \nu_2 \leq \dots \leq \nu_n$. Assume that $B \geq z_{SH}$. Then we have for each round k

$$\begin{aligned} r_k &\geq \frac{B}{|S_k| \lceil \log_2(n) \rceil} - 1 \\ &\geq \frac{2}{|S_k|} \max_{i=2,\dots,n} i \Big(1 + \gamma^{-1}\big(\max\big\{ \frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2} \big\}\big) \Big) - 1 \\ &\geq \frac{2}{|S_k|} \ (\lfloor |S_k|/2 \rfloor + 1) \Big(1 + \gamma^{-1}\big(\max\big\{ \frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2} \big\}\big) \Big) - 1 \\ &\geq \Big(1 + \gamma^{-1}\big(\max\big\{ \frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2} \big\}\big) \Big) - 1 \\ &= \gamma^{-1}\big(\max\big\{ \frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2} \big\}\big), \end{aligned}$$

where the fourth line follows from $\lfloor |S_k|/2 \rfloor \geq |S_k|/2 - 1$.

First we show that $\ell_{i,t} - \ell_{1,t} > 0$ for all $t \geq \tau_i := \gamma^{-1}\big(\frac{\nu_i - \nu_1}{2}\big)$. Given the definition of γ, we have for all $i \in [n]$ that $|\ell_{i,t} - \nu_i| \leq \gamma(t) \leq \frac{\nu_i - \nu_1}{2}$ where the last inequality holds for $t \geq \tau_i$. Thus, for $t \geq \tau_i$ we have

$$\begin{aligned} \ell_{i,t} - \ell_{1,t} &= \ell_{i,t} - \nu_i + \nu_i - \nu_1 + \nu_1 - \ell_{1,t} \\ &= \ell_{i,t} - \nu_i - (\ell_{1,t} - \nu_1) + \nu_i - \nu_1 \\ &\geq -2\gamma(t) + \nu_i - \nu_1 \\ &\geq -2\frac{\nu_i - \nu_1}{2} + \nu_i - \nu_1 \\ &= 0. \end{aligned}$$

Under this scenario, we will eliminate arm i before arm 1 since on each round the arms are sorted by their empirical losses and the top half are discarded. Note that by the assumption the ν_i limits are non-decreasing in i so that the τ_i values are non-increasing in i.

Fix a round k and assume $1 \in S_k$ (note, $1 \in S_0$). The above calculation shows that

$$t \geq \tau_i \implies \ell_{i,t} \geq \ell_{1,t}. \tag{9}$$

Consequently,

$$\begin{aligned} \{1 \in S_k, \;\; 1 \notin S_{k+1}\} &\iff \left\{ \sum_{i \in S_k} \mathbf{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor |S_k|/2 \rfloor \right\} \\ &\implies \left\{ \sum_{i \in S_k} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor |S_k|/2 \rfloor \right\} \\ &\implies \left\{ \sum_{i=2}^{\lfloor |S_k|/2 \rfloor + 1} \mathbf{1}\{r_k < \tau_i\} \geq \lfloor |S_k|/2 \rfloor \right\} \\ &\iff \left\{ r_k < \tau_{\lfloor |S_k|/2 \rfloor + 1} \right\}. \end{aligned}$$

where the first line follows by the definition of the algorithm, the second by Equation 9, and the third by τ_i being non-increasing (for all $i < j$ we have $\tau_i \geq \tau_j$ and consequently, $\mathbf{1}\{r_k < \tau_i\} \geq \mathbf{1}\{r_k < \tau_j\}$ so the *first* indicators of the sum not including 1 would be on before any other i's in $S_k \subset [n]$ sprinkled throughout $[n]$). This implies

$$\{1 \in S_k, \;\; r_k \geq \tau_{\lfloor |S_k|/2 \rfloor + 1}\} \implies \{1 \in S_{k+1}\}. \tag{10}$$

Recalling that $r_k \geq \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big\}\big)$ and $\tau_{\lfloor |S_k|/2 \rfloor + 1} = \gamma^{-1}\big(\frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big)$, we examine the following three exhaustive cases:

- **Case 1:** $\frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2} \geq \frac{\epsilon}{4}$ and $1 \in S_k$

  In this case, $r_k \geq \gamma^{-1}\big(\frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big) = \tau_{\lfloor |S_k|/2 \rfloor + 1}$. By Equation 10 we have that $1 \in S_{k+1}$ since $1 \in S_k$.

- **Case 2:** $\frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2} < \frac{\epsilon}{4}$ and $1 \in S_k$

  In this case $r_k \geq \gamma^{-1}\big(\frac{\epsilon}{4}\big)$ but $\gamma^{-1}\big(\frac{\epsilon}{4}\big) < \tau_{\lfloor |S_k|/2 \rfloor + 1}$. Equation 10 suggests that it may be possible for $1 \in S_k$ but $1 \notin S_{k+1}$. On the good event that $1 \in S_{k+1}$, the algorithm continues and on the next round either case 1 or case 2 could be true. So assume $1 \notin S_{k+1}$. Here we show that $\{1 \in S_k, \ \ 1 \notin S_{k+1}\} \implies \max_{i \in S_{k+1}} \nu_i \leq \nu_1 + \epsilon/2$. Because $1 \in S_0$, this guarantees that SuccessiveHalving either exits with arm $\widehat{i} = 1$ or some arm $\widehat{i}$ satisfying $\nu_{\widehat{i}} \leq \nu_1 + \epsilon/2$.

  Let $p = \min\{i \in [n] : \frac{\nu_i - \nu_1}{2} \geq \frac{\epsilon}{4}\}$. Note that $p > \lfloor |S_k|/2 \rfloor + 1$ by the criterion of the case and
  $$r_k \geq \gamma^{-1}\left(\frac{\epsilon}{4}\right) \geq \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right) = \tau_i, \quad \forall i \geq p.$$

  Thus, by Equation 9 ($t \geq \tau_i \implies \ell_{i,t} \geq \ell_{1,t}$) we have that arms $i \geq p$ would always have $\ell_{i,r_k} \geq \ell_{1,r_k}$ and be eliminated before or at the same time as arm 1, presuming $1 \in S_k$. In conclusion, if arm 1 is eliminated so that $1 \in S_k$ but $1 \notin S_{k+1}$ then $\max_{i \in S_{k+1}} \nu_i \leq \max_{i<p} \nu_i < \nu_1 + \epsilon/2$ by the definition of p.

- **Case 3:** $1 \notin S_k$

  Since $1 \in S_0$, there exists some $r < k$ such that $1 \in S_r$ and $1 \notin S_{r+1}$. For this r, only case 2 is possible since case 1 would proliferate $1 \in S_{r+1}$. However, under case 2, if $1 \notin S_{r+1}$ then $\max_{i \in S_{r+1}} \nu_i \leq \nu_1 + \epsilon/2$.

Because $1 \in S_0$, we either have that 1 remains in S_k (possibly alternating between cases 1 and 2) for all k until the algorithm exits with the best arm 1, or there exists some k such that case 3 is true and the algorithm exits with an arm $\widehat{i}$ such that $\nu_{\widehat{i}} \leq \nu_1 + \epsilon/2$. The proof is complete by noting that

$$|\ell_{\widehat{i}, \lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_1| \leq |\ell_{\widehat{i}, \lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_{\widehat{i}}| + |\nu_{\widehat{i}} - \nu_1| \leq \epsilon/4 + \epsilon/2 \leq \epsilon$$

by the triangle inequality and because $B \geq 2\lceil \log_2(n) \rceil \gamma^{-1}(\epsilon/4)$ by assumption.

The second, looser, but perhaps more interpretable form of z_{SH} follows from the fact that $\gamma^{-1}(x)$ is non-increasing in x so that

$$\max_{i=2,\ldots,n} i \, \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right) \leq \sum_{i=1,\ldots,n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right).$$

■

B.2 Proof of Lemma 2

Proof Let $p_n = \frac{\log(2/\delta)}{n}$, $M = \gamma^{-1}\left(\frac{\epsilon}{16}\right)$, and $\mu = \mathbb{E}[\min\{M, \gamma^{-1}\left(\frac{\nu_i - \nu_*}{4}\right)\}]$. Define the events

$$\xi_1 = \{\nu_1 \leq F^{-1}(p_n)\}$$
$$\xi_2 = \left\{ \sum_{i=1}^{n} \min\{M, \gamma^{-1}\left(\tfrac{\nu_i - \nu_*}{4}\right)\} \leq n\mu + \sqrt{2n\mu M \log(2/\delta)} + \tfrac{2}{3} M \log(2/\delta) \right\}$$

Note that $\mathbb{P}(\xi_1^c) = \mathbb{P}(\min_{i=1,\dots,n} \nu_i > F^{-1}(p_n)) = (1-p_n)^n \leq \exp(-np_n) \leq \frac{\delta}{2}$. Moreover, $\mathbb{P}(\xi_2^c) \leq \frac{\delta}{2}$ by Bernstein's inequality since

$$\mathbb{E}\left[\min\{M, \gamma^{-1}\left(\tfrac{\nu_i-\nu_*}{4}\right)\}^2\right] \leq \mathbb{E}\left[M\min\{M, \gamma^{-1}\left(\tfrac{\nu_i-\nu_*}{4}\right)\}\right] = M\mu.$$

Thus, $\mathbb{P}(\xi_1 \cap \xi_2) \geq 1-\delta$ so in what follows assume these events hold.

First we show that if $\nu_* \leq \nu_1 \leq F^{-1}(p_n)$, which we will refer to as equation $(*)$, then $\max\left\{\frac{\epsilon}{4}, \frac{\nu_i-\nu_1}{2}\right\} \geq \max\left\{\frac{\epsilon}{4}, \frac{\nu_i-\nu_*}{4}\right\}$.
Case 1: $\frac{\epsilon}{4} \leq \frac{\nu_i-\nu_1}{2}$ and $\epsilon \geq 4(F^{-1}(p_n)-\nu_*)$.

$$\tfrac{\nu_i-\nu_1}{2} \overset{(*)}{\geq} \tfrac{\nu_i-\nu_*+\nu_*-F^{-1}(p_n)}{2} = \tfrac{\nu_i-\nu_*}{4} + \tfrac{\nu_i-\nu_*}{4} - \tfrac{F^{-1}(p_n)-\nu_*}{2} \overset{(*)}{\geq} \tfrac{\nu_i-\nu_*}{4} + \tfrac{\nu_i-\nu_1}{4} - \tfrac{F^{-1}(p_n)-\nu_*}{2} \overset{\text{Case 1}}{\geq} \tfrac{\nu_i-\nu_*}{4}.$$

Case 2: $\frac{\epsilon}{4} > \frac{\nu_i-\nu_1}{2}$ and $\epsilon \geq 4(F^{-1}(p_n)-\nu_*)$.

$$\tfrac{\nu_i-\nu_*}{4} = \tfrac{\nu_i-\nu_1}{4} + \tfrac{\nu_1-\nu_*}{4} \overset{\text{Case 2}}{<} \tfrac{\epsilon}{8} + \tfrac{\nu_1-\nu_*}{4} \overset{(*)}{\leq} \tfrac{\epsilon}{8} + \tfrac{F^{-1}(p_n)-\nu_*}{4} \overset{\text{Case 2}}{<} \tfrac{\epsilon}{4}$$

which shows the desired result.

Consequently, for any $\epsilon \geq 4(F^{-1}(p_n)-\nu_*)$ we have

$$\begin{aligned}
\sum_{i=1}^n \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right) &\leq \sum_{i=1}^n \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_*}{4}\right\}\right)\\
&\leq \sum_{i=1}^n \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{16}, \tfrac{\nu_i-\nu_*}{4}\right\}\right)\\
&= \sum_{i=1}^n \min\{M, \gamma^{-1}\left(\tfrac{\nu_i-\nu_*}{4}\right)\}\\
&\leq n\mu + \sqrt{2n\mu M\log(1/\delta)} + \tfrac{2}{3}M\log(1/\delta)\\
&\leq \left(\sqrt{n\mu} + \sqrt{\tfrac{2}{3}M\log(2/\delta)}\right)^2 \leq 2n\mu + \tfrac{4}{3}M\log(2/\delta).
\end{aligned}$$

A direct computation yields

$$\begin{aligned}
\mu &= \mathbb{E}[\min\{M, \gamma^{-1}\left(\tfrac{\nu_i-\nu_*}{4}\right)\}]\\
&= \mathbb{E}[\gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{16}, \tfrac{\nu_i-\nu_*}{4}\right\}\right)]\\
&= \gamma^{-1}\left(\tfrac{\epsilon}{16}\right)F(\nu_*+\epsilon/4) + \int_{\nu_*+\epsilon/4}^{\infty} \gamma^{-1}(\tfrac{t-\nu_*}{4})dF(t)
\end{aligned}$$

so that

$$\begin{aligned}
\sum_{i=1}^n \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right) &\leq 2n\mu + \tfrac{4}{3}M\log(2/\delta)\\
&= 2n\int_{\nu_*+\epsilon/4}^{\infty} \gamma^{-1}(\tfrac{t-\nu_*}{4})dF(t) + \left(\tfrac{4}{3}\log(2/\delta) + 2nF(\nu_*+\epsilon/4)\right)\gamma^{-1}\left(\tfrac{\epsilon}{16}\right)
\end{aligned}$$

which completes the proof. ■

B.3 Proof of Proposition 4

We break the proposition up into upper and lower bounds and prove them seperately.

B.4 Uniform Allocation

Proposition 10 *Suppose we draw n random configurations from F, train each with a budget of j,*[17] *and let $\hat{\imath} = \arg\min_{i=1,\dots,n} \ell_j(X_i)$. Let $\nu_i = \ell_*(X_i)$ and without loss of generality assume $\nu_1 \leq \ldots \leq \nu_n$. If*

$$B \geq n\gamma^{-1}\left(\tfrac{1}{2}(F^{-1}(\tfrac{\log(1/\delta)}{n}) - \nu_*)\right) \tag{11}$$

then with probability at least $1-\delta$ we have $\nu_{\hat{\imath}} - \nu_* \leq 2\left(F^{-1}\left(\frac{\log(1/\delta)}{n}\right) - \nu_*\right)$.

Proof Note that if we draw n random configurations from F and $i_* = \arg\min_{i=1,\dots,n} \ell_*(X_i)$ then

$$\begin{aligned}\mathbb{P}\left(\ell_*(X_{i_*}) - \nu_* \leq \epsilon\right) &= \mathbb{P}\Big(\bigcup_{i=1}^{n}\{\ell_*(X_i) - \nu_* \leq \epsilon\}\Big)\\ &= 1 - (1 - F(\nu_* + \epsilon))^n \geq 1 - e^{-nF(\nu_*+\epsilon)}\,,\end{aligned}$$

which is equivalent to saying that with probability at least $1-\delta$, $\ell_*(X_{i_*}) - \nu_* \leq F^{-1}(\log(1/\delta)/n) - \nu_*$. Furthermore, if each configuration is trained for j iterations then with probability at least $1-\delta$

$$\begin{aligned}\ell_*(X_{\hat{\imath}}) - \nu_* &\leq \ell_j(X_{\hat{\imath}}) - \nu_* + \gamma(j) \leq \ell_j(X_{i_*}) - \nu_* + \gamma(j)\\ &\leq \ell_*(X_{i_*}) - \nu_* + 2\gamma(j) \leq F^{-1}\left(\tfrac{\log(1/\delta)}{n}\right) - \nu_* + 2\gamma(j).\end{aligned}$$

If our measurement budget B is constrained so that $B = nj$ then solving for j in terms of B and n yields the result. ∎

The following proposition demonstrates that the upper bound on the error of the uniform allocation strategy in Proposition 4 is in fact tight. That is, for any distribution F and function γ there exists a loss sequence that requires the budget described in Eq. (3) in order to avoid a loss of more than ϵ with high probability.

Proposition 11 *Fix any $\delta \in (0,1)$ and $n \in \mathbb{N}$. For any $c \in (0,1]$, let $\mathcal{F}_c$ denote the space of continuous cumulative distribution functions F satisfying*[18] $\inf_{x\in[\nu_*,1-\nu_*]}\inf_{\Delta\in[0,1-x]}\frac{F(x+\Delta)-F(x+\Delta/2)}{F(x+\Delta)-F(x)} \geq c$. *And let Γ denote the space of monotonically decreasing functions over $\mathbb{N}$. For any $F \in \mathcal{F}_c$ and $\gamma \in \Gamma$ there exists a probability distribution μ over $\mathcal{X}$ and a sequence of functions $\ell_j : \mathcal{X} \to \mathbb{R} \quad \forall j \in \mathbb{N}$ with $\ell_* := \lim_{j\to\infty} \ell_j$, $\nu_* = \inf_{x\in\mathcal{X}} \ell_*(x)$ such that*

[17] Here j can be bounded (finite horizon) or unbounded (infinite horizon).

[18] Note that this condition is met whenever F is convex. Moreover, if $F(\nu_* + \epsilon) = c_1^{-1}\epsilon^{\beta}$ then it is easy to verify that $c = 1 - 2^{-\beta} \geq \frac{1}{2}\min\{1, \beta\}$.

$\sup_{x\in\mathcal{X}} |\ell_j(x) - \ell_*(x)| \leq \gamma(j)$ and $\mathbb{P}_\mu(\ell_*(X) - \nu_* \leq \epsilon) = F(\epsilon)$. Moreover, if n configurations $X_1, \dots, X_n$ are drawn from μ and $\hat{\imath} = \arg\min_{i\in 1,\dots,n} \ell_{B/n}(X_i)$ then with probability at least δ

$$\ell_*(X_{\hat{\imath}}) - \nu_* \geq 2(F^{-1}(\tfrac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)$$

whenever $B \leq n\gamma^{-1}\left(2(F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)\right)$.

Proof Let $\mathcal{X} = [0,1]$, $\ell_*(x) = F^{-1}(x)$, and μ be the uniform distribution over $[0,1]$. Define $\widehat{\nu} = F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)})$ and set

$$\ell_j(x) = \begin{cases} \widehat{\nu} + \frac{1}{2}\gamma(j) + (\widehat{\nu} + \frac{1}{2}\gamma(j) - \ell_*(x)) & \text{if } |\widehat{\nu} + \frac{1}{2}\gamma(j) - \ell_*(x)| \leq \frac{1}{2}\gamma(j) \\ \ell_*(x) & \text{otherwise.} \end{cases}$$

Essentially, if $\ell_*(x)$ is within $\frac{1}{2}\gamma(j)$ of $\widehat{\nu} + \frac{1}{2}\gamma(j)$ then we set $\ell_j(x)$ equal to $\ell_*(x)$ reflected across the value $2\widehat{\nu} + \gamma(j)$. Clearly, $|\ell_j(x) - \ell_*(x)| \leq \gamma(j)$ for all $x \in \mathcal{X}$.

Since each $\ell_*(X_i)$ is distributed according to F, we have

$$\mathbb{P}\Big(\bigcap_{i=1}^n \{\ell_*(X_i) - \nu_* \geq \epsilon\}\Big) = (1 - F(\nu_* + \epsilon))^n \geq e^{-nF(\nu_*+\epsilon)/(1-F(\nu_*+\epsilon))}\,.$$

Setting the right-hand-side greater than or equal to δ/c and solving for ϵ, we find $\nu_* + \epsilon \geq F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) = \widehat{\nu}$.

Define $I_0 = [\nu_*, \widehat{\nu})$, $I_1 = [\widehat{\nu}, \widehat{\nu} + \frac{1}{2}\gamma(B/n))$ and $I_2 = [\widehat{\nu} + \frac{1}{2}\gamma(B/n), \widehat{\nu} + \gamma(B/n)]$. Furthermore, for $j \in \{0,1,2\}$ define $N_j = \sum_{i=1}^n \mathbf{1}_{\ell_*(X_i)\in I_j}$. Given $N_0 = 0$ (which occurs with probability at least δ/c), if $N_1 = 0$ then $\ell_*(X_{\hat{\imath}}) - \nu_* \geq F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) + \frac{1}{2}\gamma(B/n)$ and the claim is true.

Below we will show that if $N_2 > 0$ whenever $N_1 > 0$ then the claim is also true. We now show that this happens with at least probability c whenever $N_1 + N_2 = m$ for any $m > 0$. Observe that

$$\begin{aligned} \mathbb{P}(N_2 > 0 | N_1 + N_2 = m) &= 1 - \mathbb{P}(N_2 = 0 | N_1 + N_2 = m) \\ &= 1 - (1 - \mathbb{P}(\nu_i \in I_2 | \nu_i \in I_1 \cup I_2))^m \geq 1 - (1-c)^m \geq c \end{aligned}$$

since

$$\mathbb{P}(\nu_i \in I_2 | \nu_i \in I_1 \cup I_2) = \frac{\mathbb{P}(\nu_i \in I_2)}{\mathbb{P}(\nu_i \in I_1 \cup I_2)} = \frac{\mathbb{P}(\nu_i \in [\widehat{\nu} + \frac{1}{2}\gamma, \widehat{\nu} + \gamma])}{\mathbb{P}(\nu_i \in [\widehat{\nu}, \widehat{\nu} + \gamma])} = \frac{F(\widehat{\nu} + \gamma) - F(\widehat{\nu} + \frac{1}{2}\gamma)}{F(\widehat{\nu} + \gamma) - F(\widehat{\nu})} \geq c\,.$$

Thus, the probability of the event that $N_0 = 0$ and $N_2 > 0$ whenever $N_1 > 0$ occurs with probability at least $\delta/c \cdot c = \delta$, so assume this is the case in what follows.

Since $N_0 = 0$, for all $j \in \mathbb{N}$, each X_i must fall into one of three cases:

1. $\ell_*(X_i) > \widehat{\nu} + \gamma(j) \iff \ell_j(X_i) > \widehat{\nu} + \gamma(j)$
2. $\widehat{\nu} \leq \ell_*(X_i) < \widehat{\nu} + \frac{1}{2}\gamma(j) \iff \widehat{\nu} + \frac{1}{2}\gamma(j) < \ell_j(X_i) \leq \widehat{\nu} + \gamma(j)$

3. $\widehat{\nu} + \frac{1}{2}\gamma(j) \leq \ell_*(X_i) \leq \widehat{\nu} + \gamma(j) \iff \widehat{\nu} \leq \ell_j(X_i) \leq \widehat{\nu} + \frac{1}{2}\gamma(j)$

The first case holds since within that regime we have $\ell_j(x) = \ell_*(x)$, while the last two cases hold since they consider the regime where $\ell_j(x) = 2\widehat{\nu} + \gamma(j) - \ell_*(x)$. Thus, for any i such that $\ell_*(X_i) \in I_2$ it must be the case that $\ell_j(X_i) \in I_1$ and vice versa. Because $N_2 \geq N_1 > 0$, we conclude that if $\hat{\imath} = \arg\min_i \ell_{B/n}(X_i)$ then $\ell_{B/n}(X_{\hat{\imath}}) \in I_1$ and $\ell_*(X_{\hat{\imath}}) \in I_2$. That is, $\nu_{\hat{\imath}} - \nu_* \geq \widehat{\nu} - \nu_* + \frac{1}{2}\gamma(j) = F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_* + \frac{1}{2}\gamma(j)$. So if we wish $\nu_{\hat{\imath}} - \nu_* \leq 2(F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)$ with probability at least δ then we require $B/n = j \geq \gamma^{-1}\left(2(F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)\right)$. ■

B.5 Proof of Theorem 5

Proof Step 1: Simplify $\mathbf{H}(F,\gamma,n,\delta)$. We begin by simplifying $\mathbf{H}(F,\gamma,n,\delta)$ in terms of just n,δ,α,β. In what follows, we use a constant c that may differ from one inequality to the next but remains an absolute constant that depends on α,β only. Let $p_n = \frac{\log(2/\delta)}{n}$ so that

$$\gamma^{-1}\left(\tfrac{F^{-1}(p_n)-\nu_*}{4}\right) \leq c\left(F^{-1}\left(p_n\right) - \nu_*\right)^{-\alpha} \leq c\, p_n^{-\alpha/\beta}$$

and

$$\int_{p_n}^{1} \gamma^{-1}(\tfrac{F^{-1}(t)-\nu_*}{4})dt \leq c\int_{p_n}^{1} t^{-\alpha/\beta}dt \leq \begin{cases} c\log(1/p_n) & \text{if } \alpha=\beta \\ c\frac{1-p_n^{1-\alpha/\beta}}{1-\alpha/\beta} & \text{if } \alpha\neq\beta. \end{cases}$$

We conclude that

$$\begin{aligned} \mathbf{H}(F,\gamma,n,\delta) &= 2n\int_{p_n}^{1}\gamma^{-1}(\tfrac{F^{-1}(t)-\nu_*}{4})dt + \tfrac{10}{3}\log(2/\delta)\gamma^{-1}\left(\tfrac{F^{-1}(p_n)-\nu_*}{4}\right) \\ &\leq cp_n^{-\alpha/\beta}\log(1/\delta) + cn\begin{cases} \log(1/p_n) & \text{if } \alpha=\beta \\ \frac{1-p_n^{1-\alpha/\beta}}{1-\alpha/\beta} & \text{if } \alpha\neq\beta. \end{cases} \end{aligned}$$

Step 2: Solve for $(B_{k,l}, n_{k,l})$ in terms of Δ. Fix $\Delta > 0$. Our strategy is to describe $n_{k,l}$ in terms of Δ. In particular, parameterize $n_{k,l}$ such that $p_{n_{k,l}} = c\frac{\log(4k^3/\delta)}{n_{k,l}} = \Delta^{\beta}$ so that $n_{k,l} = c\Delta^{-\beta}\log(4k^3/\delta)$ so

$$\begin{aligned} \mathbf{H}(F,\gamma,n_{k,l},\delta_{k,l}) &\leq cp_{n_{k,l}}^{-\alpha/\beta}\log(1/\delta_{k,l}) + cn_{k,l}\begin{cases} \log(1/p_{n_{k,l}}) & \text{if } \alpha=\beta \\ \frac{1-p_{n_{k,l}}^{1-\alpha/\beta}}{1-\alpha/\beta} & \text{if } \alpha\neq\beta. \end{cases} \\ &\leq c\log(k/\delta)\left[\Delta^{-\alpha} + \begin{cases} \Delta^{-\beta}\log(\Delta^{-1}) & \text{if } \alpha=\beta \\ \frac{\Delta^{-\beta}-\Delta^{-\alpha}}{1-\alpha/\beta} & \text{if } \alpha\neq\beta \end{cases}\right] \\ &\leq c\log(k/\delta)\min\{\tfrac{1}{|1-\alpha/\beta|},\log(\Delta^{-1})\}\Delta^{-\max\{\beta,\alpha\}} \end{aligned}$$

where the last line follows from

$$\Delta^{\max\{\beta,\alpha\}}\frac{\Delta^{-\beta}-\Delta^{-\alpha}}{1-\alpha/\beta} = \beta\frac{\Delta^{\max\{0,\alpha-\beta\}}-\Delta^{\max\{0,\beta-\alpha\}}}{\beta-\alpha}$$
$$= \beta\begin{cases}\frac{1-\Delta^{\beta-\alpha}}{\beta-\alpha} & \text{if } \beta>\alpha\\ \frac{1-\Delta^{\alpha-\beta}}{\alpha-\beta} & \text{if } \beta<\alpha\end{cases} \leq c\min\{\tfrac{1}{|1-\alpha/\beta|},\log(\Delta^{-1})\}.$$

Using the upperbound $\lceil\log(n_{k,l})\rceil \leq c\log(\log(k/\delta)\Delta^{-1}) \leq c\log(\log(k/\delta))\log(\Delta^{-1})$ and letting $z_\Delta = \log(\Delta^{-1})^2\Delta^{-\max\{\beta,\alpha\}}$, we conclude that

$$\begin{aligned}B_{k,l} &< \min\{2^k : 2^k > 4\lceil\log(n_{k,l})\rceil\mathbf{H}(F,\gamma,n_{k,l},\delta_{k,l})\}\\ &< \min\{2^k : 2^k > c\log(k/\delta)\log(\log(k/\delta))z_\Delta\}\\ &\leq cz_\Delta\log(\log(z_\Delta)/\delta)\log(\log(\log(z_\Delta)/\delta))\\ &= cz_\Delta\overline{\log}(\log(z_\Delta)/\delta).\end{aligned}$$

Step 3: Count the total number of measurements. Moreover, the total number of measurements before $\hat{\imath}_{k,l}$ is output is upperbounded by

$$T = \sum_{i=1}^{k}\sum_{j=l}^{i}B_{i,j} \leq k\sum_{i=1}^{k}B_{i,1} \leq 2kB_{k,1} = 2B_{k,1}\log_2(B_{k,1})$$

where we have employed the so-called "doubling trick": $\sum_{i=1}^{k}B_{i,1} = \sum_{i=1}^{k}2^i \leq 2^{k+1} = 2B_{k,i}$. Simplifying,

$$T \leq cz_\Delta\overline{\log}(\log(z_\Delta)/\delta)\overline{\log}(z_\Delta\log(\log(z_\Delta)/\delta)) \leq c\Delta^{-\max\{\beta,\alpha\}}\overline{\log}(\Delta^{-1})^3\overline{\log}(\log(\Delta^{-1})/\delta)$$

Solving for Δ in terms of T obtains

$$\Delta = c\left(\frac{\overline{\log}(T)^3\overline{\log}(\log(T)/\delta)}{T}\right)^{1/\max\{\alpha,\beta\}}.$$

Because the output arm is just the empirical best, there is some error associated with using the empirical estimate. The arm returned returned on round (k,l) is pulled $\lfloor\frac{2^{k-1}}{l}\rfloor \gtrsim B_{k,l}/\log(B_{k,l})$ times so the possible error is bounded by $\gamma(B_{k,l}/\log(B_{k,l})) \leq c\left(\frac{\log(B_{k,l})}{B_{k,l}}\right)^{1/\alpha} \leq c\left(\frac{\log(B)^2\log(\log(B))}{B}\right)^{1/\alpha}$ which is dominated by the value of Δ solved for above. ■

B.6 Proof of Theorem 7

Proof Step 1: Simplify $\mathbf{H}(F,\gamma,n,\delta,\epsilon)$. We begin by simplifying $\mathbf{H}(F,\gamma,n,\delta,\epsilon)$ in terms of just n,δ,α,β. As before, we use a constant c that may differ from one inequality to the next but remains an absolute constant. Let $p_n = \frac{\log(2/\delta)}{n}$ First we solve for ϵ by noting that we identify the best arm if $\nu_{\hat{\imath}} - \nu_* < \Delta_2$. Thus, if $\nu_{\hat{\imath}} - \nu_* \leq \big(F^{-1}(p_n) - \nu_*\big) + \epsilon/2$ then we set

$$\epsilon = \max\big\{2\big(\Delta_2 - \big(F^{-1}(p_n) - \nu_*\big)\big), 4\big(F^{-1}(p_n) - \nu_*\big)\big\}$$

so that

$$\nu_{\hat{i}} - \nu_* \leq \max\left\{3\big(F^{-1}(p_n) - \nu_*\big), \Delta_2\right\} = \Delta_{\lceil \max\{2, cKp_n\}\rceil}.$$

We treat the case when $3\big(F^{-1}(p_n) - \nu_*\big) \leq \Delta_2$ and the alternative separately.

First assume $3\big(F^{-1}(p_n) - \nu_*\big) > \Delta_2$ so that $\epsilon = 4\big(F^{-1}(p_n) - \nu_*\big)$ and $\mathbf{H}(F, \gamma, n, \delta, \epsilon) = \mathbf{H}(F, \gamma, n, \delta)$. We also have

$$\gamma^{-1}\left(\tfrac{F^{-1}(p_n) - \nu_*}{4}\right) \leq c\left(F^{-1}(p_n) - \nu_*\right)^{-\alpha} \leq c\,\Delta_{\lceil p_n K\rceil}^{-\alpha}$$

and

$$\int_{p_n}^{1} \gamma^{-1}(\tfrac{F^{-1}(t) - \nu_*}{4})dt = \int_{F^{-1}(p_n)}^{1} \gamma^{-1}(\tfrac{x - \nu_*}{4})dF(x) \leq \frac{c}{K} \sum_{i=\lceil p_n K\rceil}^{K} \Delta_i^{-\alpha}$$

so that

$$\begin{aligned}\mathbf{H}(F, \gamma, n, \delta) &= 2n\int_{p_n}^{1} \gamma^{-1}(\tfrac{F^{-1}(t) - \nu_*}{4})dt + \tfrac{10}{3}\log(2/\delta)\gamma^{-1}\left(\tfrac{F^{-1}(p_n) - \nu_*}{4}\right) \\ &\leq c\Delta_{\lceil p_n K\rceil}^{-\alpha}\log(1/\delta) + \frac{cn}{K}\sum_{i=\lceil p_n K\rceil}^{K} \Delta_i^{-\alpha}.\end{aligned}$$

Now consider the case when $3\big(F^{-1}(p_n) - \nu_*\big) \leq \Delta_2$. In this case $F(\nu_* + \epsilon/4) = 1/K$, $\gamma^{-1}\left(\frac{\epsilon}{16}\right) \leq c\Delta_2^{-\alpha}$, and $\int_{\nu_* + \epsilon/4}^{\infty} \gamma^{-1}(\frac{t - \nu_*}{4})dF(t) \leq c\sum_{i=2}^{K} \Delta_i^{-\alpha}$ so that

$$\begin{aligned}\mathbf{H}(F, \gamma, n, \delta, \epsilon) &= 2n\int_{\nu_* + \epsilon/4}^{\infty} \gamma^{-1}(\tfrac{t - \nu_*}{4})dF(t) + \left(\tfrac{4}{3}\log(2/\delta) + 2nF(\nu_* + \epsilon/4)\right)\gamma^{-1}\left(\tfrac{\epsilon}{16}\right) \\ &\leq c(\log(1/\delta) + n/K)\Delta_2^{-\alpha} + \frac{cn}{K}\sum_{i=2}^{K} \Delta_i^{-\alpha}.\end{aligned}$$

Step 2: Solve for $(B_{k,l}, n_{k,l})$ in terms of Δ. Note there is no improvement possible once $p_{n_{k,l}} \leq 1/K$ since $3\big(F^{-1}(1/K) - \nu_*\big) \leq \Delta_2$. That is, when $p_{n_{k,l}} \leq 1/K$ the algorithm has found the best arm but will continue to take samples indefinetely. Thus, we only consider the case when $q = 1/K$ and $q > 1/K$. Fix $\Delta > 0$. Our strategy is to describe $n_{k,l}$ in terms of q. In particular, parameterize $n_{k,l}$ such that $p_{n_{k,l}} = c\frac{\log(4k^3/\delta)}{n_{k,l}} = q$ so that $n_{k,l} = cq^{-1}\log(4k^3/\delta)$ so

$$\begin{aligned}\mathbf{H}(F, \gamma, n_{k,l}, \delta_{k,l}, \epsilon_{k,l}) &\leq c\begin{cases}(\log(1/\delta_{k,l}) + \frac{n_{k,l}}{K})\Delta_2^{-\alpha} + \frac{n_{k,l}}{K}\sum_{i=2}^{K}\Delta_i^{-\alpha} & \text{if } 5\big(F^{-1}(p_{n_{k,l}}) - \nu_*\big) \leq \Delta_2 \\ \Delta_{\lceil p_{n_{k,l}} K\rceil}^{-\alpha}\log(1/\delta_{k,l}) + \frac{n_{k,l}}{K}\sum_{i=\lceil p_{n_{k,l}} K\rceil}^{K}\Delta_i^{-\alpha} & \text{if otherwise}\end{cases} \\ &\leq c\log(k/\delta)\begin{cases}\Delta_2^{-\alpha} + \sum_{i=2}^{K}\Delta_i^{-\alpha} & \text{if } q = 1/K \\ \Delta_{\lceil qK\rceil}^{-\alpha} + \frac{1}{qK}\sum_{i=\lceil qK\rceil}^{K}\Delta_i^{-\alpha} & \text{if } q > 1/K.\end{cases} \\ &\leq c\log(k/\delta)\Delta_{\lceil\max\{2, qK\}\rceil}^{-\alpha} + \frac{1}{qK}\sum_{i=\lceil\max\{2, qK\}\rceil}^{K}\Delta_i^{-\alpha}\end{aligned}$$

Using the upperbound $\lceil \log(n_{k,l}) \rceil \leq c \log(\log(k/\delta) q^{-1}) \leq c \log(\log(k/\delta)) \log(q^{-1})$ and letting $z_q = \log(q^{-1})(\Delta_{\lceil \max\{2,qK\}\rceil}^{-\alpha} + \frac{1}{qK} \sum_{i=\lceil \max\{2,qK\}\rceil}^{K} \Delta_i^{-\alpha})$, we apply the exact sequence of steps as in the proof of Theorem 5 to obtain

$$T \leq c z_q \overline{\log}(\log(z_q)/\delta) \overline{\log}(z_q \log(\log(z_q)/\delta))$$

Because the output arm is just the empirical best, there is some error associated with using the empirical estimate. The arm returned on round (k,l) is pulled $\lceil \frac{2^{k-1}}{l} \rceil \geq c B_{k,l}/\log(B_{k,l})$ times so the possible error is bounded by $\gamma(B_{k,l}/\log(B_{k,l})) \leq c \left(\frac{\log(B_{k,l})}{B_{k,l}}\right)^{1/\alpha} \leq c \left(\frac{\log(T)^2 \log(\log(T))}{T}\right)^{1/\alpha}$. This is dominated by $\Delta_{\lceil \max\{2,qK\}\rceil}$ for the value of T prescribed by the above calculation, completing the proof. ■

B.7 Proof of Theorem 8

Proof Let s denote the index of the last stage, to be determined later. If $\widetilde{r}_k = R\eta^{k-s}$ and $\widetilde{n}_k = n\eta^{-k}$ so that $r_k = \lfloor \widetilde{r}_k \rfloor$ and $n_k = \lfloor \widetilde{n}_k \rfloor$ then

$$\sum_{k=0}^{s} n_k r_k \leq \sum_{k=0}^{s} \widetilde{n}_k \widetilde{r}_k = nR(s+1)\eta^{-s} \leq B$$

since, by definition, $s = \min\{t \in \mathbb{N} : nR(t+1)\eta^{-t} \leq B, t \leq \log_\eta(\min\{R,n\})\}$. It is straightforward to verify that $B \geq z_{SH}$ ensures that $r_0 \geq 1$ and $n_s \geq 1$.

The proof for Theorem 1 holds here with a few modifications. First, we derive a lower bound on the resource per arm r_k per round if $B \geq z_{SH}$ with generalized elimination rate η:

$$\begin{aligned} r_k &\geq \frac{B}{|S_k|(\lfloor \log_\eta(n) \rfloor + 1)} - 1 \\ &\geq \frac{\eta}{|S_k|} \max_{i=2,\ldots,n} i \bigg(1 + \min\big\{R, \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2}\big\}\big)\big\}\bigg) - 1 \\ &\geq \frac{\eta}{|S_k|} \; (\lfloor |S_k|/\eta \rfloor + 1) \bigg(1 + \min\big\{R, \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big\}\big)\big\}\bigg) - 1 \\ &\geq \bigg(1 + \min\big\{R, \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big\}\big)\big\}\bigg) - 1 \\ &= \min\big\{R, \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor + 1} - \nu_1}{2}\big\}\big)\big\}. \end{aligned}$$

Also, note that $\gamma(R) = 0$, hence if the minimum is ever active, $\ell_{i,R} = \nu_i$ and we know the true loss. The rest of the proof is same as that for Theorem 1 for η in place of 2.

In addition, we note that

$$\max_{i=n_s+1,\ldots,n} i \; \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right) \leq n_s \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_{n_s+1} - \nu_1}{2}\right\}\right) + \sum_{i > n_s} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right).$$

■

References

A. Agarwal, J. Duchi, P. L. Bartlett, and C. Levrard. Oracle inequalities for computationally budgeted model selection. In *Conference On Learning Theory (COLT)*, 2011.

A. Agarwal, S. Kakade, N. Karampatziakis, L. Song, and G. Valiant. Least squares revisited: Scalable approaches for multi-class prediction. In *International Conference on Machine Learning (ICML)*, pages 541–549, 2014.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper-parameter optimization. In *Neural Information Processing Systems (NIPS)*, 2011.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory (ALT)*, 2009.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

A. Carpentier and M. Valko. Simple regret for infinitely many armed bandits. In *International Conference on Machine Learning (ICML)*, 2015.

E. Contal, V. Perchet, and N. Vayatis. Gaussian process optimization with mutual information. In *International Conference on Machine Learning (ICML)*, 2014.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

K. Eggensperger et al. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *Neural Information Processing Systems (NIPS) Bayesian Optimization Workshop*, 2013.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

M. Feurer. Personal communication, 2015.

M. Feurer, J. Springenberg, and F. Hutter. Using meta-learning to initialize Bayesian optimization of hyperparameters. In *ECAI Workshop on Meta-Learning and Algorithm Selection*, 2014.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Neural Information Processing Systems (NIPS)*, 2015.

D. Golovin, B. Sonik, S. Moitra, G. Kochanski, J. Karro, and D.Sculley. Google Vizier: A service for black-box optimization. In *Knowledge Discovery and Data Mining (KDD)*, 2017.

S. Grünewälder, J. Audibert, M. Opper, and J. Shawe–Taylor. Regret bounds for Gaussian process bandit problems. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

A. György and L. Kocsis. Efficient multi-start strategies for local search algorithms. *Journal of Artificial Intelligence Research*, 41:407–444, 2011.

F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization (LION)*, 2011.

K. Jamieson and R. Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2014.

K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil'UCB: An optimal exploration algorithm for multi-armed bandits. In *Conference On Learning Theory (COLT)*, pages 423–439, 2014.

K. G. Jamieson, D. Haas, and B. Recht. The power of adaptivity in identifying statistical alternatives. In *Neural Information Processing Systems (NIPS)*, pages 775–783, 2016.

K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian optimization and bandits via additive models. In *International Conference on Machine Learning (ICML)*, 2015.

K. Kandasamy, G. Dasarathy, J. B. Oliva, J. G. Schneider, and B. Póczos. Gaussian process bandit optimization with multi-fidelity evaluations. In *Neural Information Processing Systems (NIPS)*, 2016.

K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian optimization with continuous approximation. In *International Conference on Machine Learning (ICML)*, 2017.

Z. Karnin, T. Koren, and O. Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning (ICML)*, 2013.

E. Kaufmann, O. Cappé, and A. Garivier. On the complexity of best arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 16:1–42, 2015.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017a.

A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In *International Conference On Learning Representation (ICLR)*, 2017b.

A. Krizhevsky. Learning multiple layers of features from tiny images. In *Technical report, Department of Computer Science, Univsersity of Toronto*, 2009.

T. Krueger, D. Panknin, and M. Braun. Fast cross-validation via sequential testing. *Journal of Machine Learning Research*, 16:1103–1155, 2015.

H. Larochelle et al. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning (ICML)*, 2007.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *International Conference On Learning Representation (ICLR)*, 2017.

O. Maron and A. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11:193–225, 1997.

V. Mnih and J.-Y. Audibert. Empirical Bernstein stopping. In *International Conference on Machine Learning (ICML)*, 2008.

Y. Netzer et al. Reading digits in natural images with unsupervised feature learning. In *Neural Information Processing Systems (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems (NIPS)*, 2007.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR)*, 2012.

J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems (NIPS)*, 2012.

J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Prabhat, and R. Adamst. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning (ICML)*, 2015a.

J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Bayesian optimization using deep neural networks. In *International Conference on Machine Learning (ICML)*, 2015b.

E. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska. Automating model search for large scale machine learning,. In *ACM Symposium on Cloud Computing (SOCC)*, 2015.

J. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Neural Information Processing Systems (NIPS)*, 2016.

N. Srinivas, A. Krause, M. Seeger, and S. M. Kakade. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

K. Swersky, J. Snoek, and R. Adams. Multi-task Bayesian optimization. In *Neural Information Processing Systems (NIPS)*, 2013.

K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

C. Thornton et al. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Knowledge Discovery and Data Mining (KDD)*, 2013.

A. van der Vaart and H. van Zanten. Information rates of nonparametric Gaussian process methods. *Journal of Machine Learning Research*, 12:2095–2119, 2011.

Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. de Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

Z. Wang, B. Zhou, and S. Jegelka. Optimization as estimation with Gaussian processes in bandit settings. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on massively scalable Gaussian processes. *arXiv:1511.01870*, 2015.