Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

Lisha Li   LISHAL@CS.UCLA.EDU
UCLA, 4732 Boelter Hall, Los Angeles, CA 90095
Kevin Jamieson   KJAMIESON@EECS.BERKELEY.EDU
UC Berkeley, 465 Soda Hall, Berkeley, CA 94720
Giulia DeSalvo   DESALVO@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012
Afshin Rostamizadeh   ROSTAMI@GOOGLE.COM
Google, 76 9th Avenue, New York, NY 10011
Ameet Talwalkar   AMEET@CS.UCLA.EDU
UCLA, 4732 Boelter Hall, Los Angeles, CA 90095

Abstract

Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While current methods offer efficiencies by adaptively choosing new configurations to train, an alternative strategy is to adaptively allocate resources across the selected configurations. We formulate hyperparameter optimization as a pure-exploration non-stochastic infinitely many armed bandit problem where a predefined resource like iterations, data samples, or features is allocated to randomly sampled configurations. We introduce HYPERBAND for this framework and analyze its theoretical properties, providing several desirable guarantees. Furthermore, we compare HYPERBAND with state-of-the-art methods on a suite of hyperparameter optimization problems. We observe that HYPERBAND provides $5\times$ to $30\times$ speedup over state-of-the-art Bayesian optimization algorithms on a variety of deep-learning and kernel-based learning problems.

Keywords: hyperparameter optimization, model selection, bandits with infinitely many arms, online optimization, deep learning

1. Introduction

In recent years, machine learning models have exploded in complexity and expressibility at the cost of staggering computational costs and a growing number of tuning parameters that are difficult to set by standard optimization techniques. These 'hyperparameters' are inputs to machine learning algorithms 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 quality of the resulting model. Consequently, practitioners often default to brute-force methods like random search and grid search.

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., 2015). However, these methods tackle a fundamentally challenging problem of simultaneously fitting and optimizing a high-dimensional, non-convex function with unknown smoothness, and possibly noisy evaluations. To overcome these difficulties, some Bayesian optimization methods resort to heuristics to model the objective function or speed up resource intensive subroutines. In contrast to naive random search, methods that rely on these heuristics are not endowed with any theoretical consistency guarantees.[1] Moreover, these adaptive configuration selection methods are intrinsically sequential and thus difficult to parallelize.

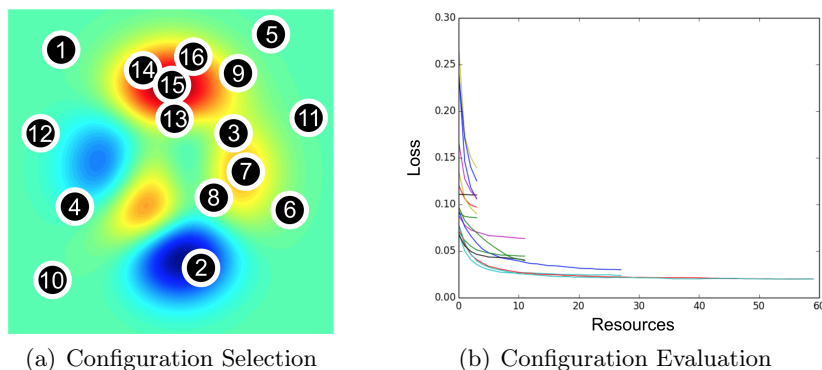(a) Configuration Selection   (b) Configuration Evaluation

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.

Instead, we explore a different direction for hyperparameter optimization that 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 these 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. Configuration evaluation approaches could ostensibly be coupled with either random search or Bayesian optimization approaches. Random search offers a simple, parallelizable, and theoretically principled launching point, while Bayesian optimization may offer improved empirical accuracy.

---

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.

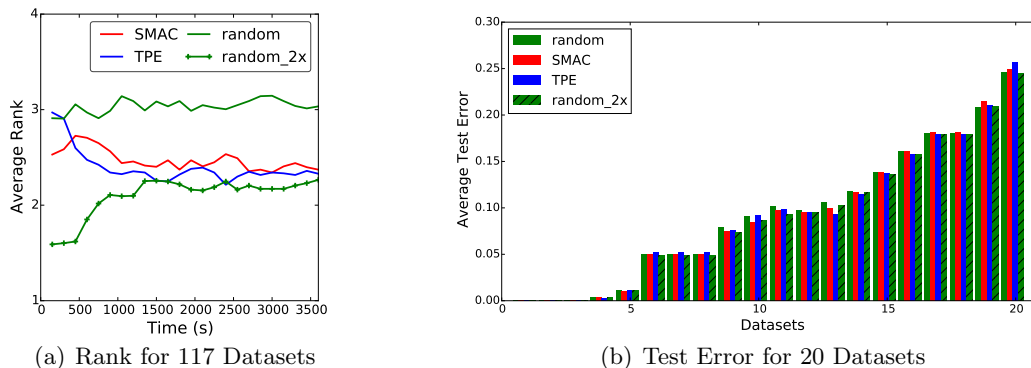(a) Rank for 117 Datasets

(b) Test Error for 20 Datasets

Figure 2: Empirical evaluation of various search methods on 117 datasets. Search methods were executed for a one hour duration for each dataset, continuously reporting their best identified models throughout this time window. Models were evaluated using an unseen test set. Results are reported for random search ('random'), random search run on two machines ('random_2x'), and two Bayesian optimization methods ('SMAC', 'TPE'). (a) Average rank of test error across all datasets, where lower is better. The rank for each dataset is based on the average test error across 20 trials. (b) Average test error for 20 randomly sampled datasets after one hour of execution. See Figure A.3 for corresponding results for all 117 datasets.

To better understand the tradeoff between random and adaptive configuration selection, we revisited a recent empirical study by Feurer et al. (2015a) to extensively compare two state-of-the-art Bayesian optimization methods — SMAC (Hutter et al., 2011) and TPE (Bergstra et al., 2011) — to random search across 117 datasets (see Section 3.2.1 for details of the experimental setup). Figure 2(a) presents a rank plot, which is standard in this field (Feurer et al., 2015a; Dewancker et al., 2016; Feurer et al., 2015b), in which the accuracy of each method is ranked for each dataset, and the average rank across 117 datasets is reported (lower is better). Random search is soundly beaten by SMAC and TPE in these rank plots. However, the test error bar chart in Figure 2(b) of 20 randomly sampled datasets from the 117 tells a different story. While these plots confirm that the Bayesian methods consistently outperform random sampling, they further show that the performance gap is quite small, a subtlety which is lost in rank-based evaluations. In fact, under the same experimental setup, running random search on two machines (denoted as 'random_2x' in the plots) yields superior results to these two Bayesian optimization methods. In light of these results, along with the amenability of random search to theoretical analysis, we focus on speeding up random search using our configuration evaluation approach.

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. 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. 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 $5\times$ to $30\times$ faster than state-of-the-art 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, infinitely-many 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 setting Kaufmann et al. (2015).

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

2. 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 adaptive resource, and present a few guidelines on how to deploy Hyperband in practice.

2.1 SuccessiveHalving

Hyperband extends the SuccessiveHalving algorithm proposed for hyperparameter optimization in 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 configurations 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 time 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 3 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 from the terminal validation loss and will be referred to as "envelope" functions. It is possible to differentiate between the two configurations when the envelopes diverge. 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 terminal losses. There are two
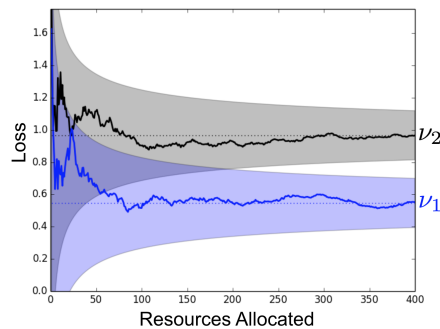
Figure 3: The validation loss as a function of total resources allocated for two configurations. The shaded areas bound the maximum distance from the terminal validation loss and monotonically decreases with the resource.

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 dataset 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 using minimal resources (e.g., if iterative training methods converge very quickly for a given dataset 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, forcing the practitioner to make this tradeoff severely hinders the applicability of existing configuration evaluation methods.

2.2 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 which 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 about B total resources and corresponds to a different tradeoff between

n and B/n. A single execution of HYPERBAND takes a finite number of iterations, 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.

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

> **input** $\qquad$: R, η (default $\eta = 3$)
> **initialization**: $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
>
> **1 for** $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
> **2** $\qquad n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil, \qquad r = R\eta^{-s}$
> $\qquad$ // begin SUCCESSIVEHALVING with (n, r) inner loop
> **3** $\qquad T =$ `get_hyperparameter_configuration`(n)
> **4** $\qquad$ **for** $i \in \{0, \ldots, s\}$ **do**
> **5** $\qquad\qquad n_i = \lfloor n\eta^{-i} \rfloor$
> **6** $\qquad\qquad r_i = r\eta^i$
> **7** $\qquad\qquad L = \{$`run_then_return_val_loss`$(t, r_i) : t \in T\}$
> **8** $\qquad\qquad T =$ `top_k`$(T, L, \lfloor n_i/\eta \rfloor)$
> **9** $\qquad$ **end**
> **10 end**
> **11 return** *Configuration with the smallest intermediate loss seen so far.*

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 with quality hyperparameters (i.e. a useful prior), the better HYPERBAND will perform (see Section 6 for further discussion).

6

- `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.

2.3 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 dataset and optimize hyperparameters for the LeNet convolutional neural network trained using mini-batch SGD.[2] 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 further define the number of iterations as the resource to allocate, with one unit of resource corresponding to one epoch or a full pass over the dataset. 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.

	$s = 4$		$s = 3$		$s = 2$		$s = 1$		$s = 0$	
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 $\eta = 3$.
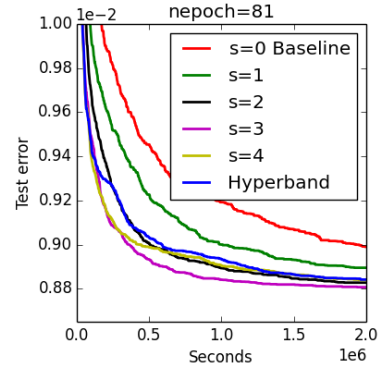

Figure 4: Performance of individual brackets s and HYPERBAND.

Figure 4 shows an empirical comparison of the average test error across 70 trials of the different brackets of HYPERBAND if they were used 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. But we note that HYPERBAND does nearly as well as the optimal bracket ($s = 3$) and vastly outperforms the baseline uniform allocation (i.e. random search), which is equivalent to bracket $s = 0$.

2.4 Different types of resources

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

---

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

- **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.

- **Dataset Subsampling** - Here we consider the setting of a black-box batch training algorithm that takes a dataset as input and outputs a model. In this setting, we treat the resource as the size of a random subset of the dataset with R corresponding to the full dataset size. Subsampling dataset 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.

2.5 Setting R

The resource R and η (which we address next) are the only required inputs to HYPERBAND. As mentioned in Section 2.2, 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 dataset 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 4. In this version of the algorithm, we use a budget that doubles over time, $B \in \{2, 4, 8, 16, \ldots\}$, and for each choice of B we consider all possible values of $n \in \{2^k : k \in \{1, \ldots, \log_2(B)\}\}$. For each choice of B and n, we run an instance of the (infinite horizon) SUCCESSIVEHALVING algorithm that grows R with increasing B. The main difference between this algorithm and Algorithm 1 is that the number of unique brackets is growing over time instead of just looped over. We will analyze this version of HYPERBAND in more detail in Section 4 and use it as the launching point for the theoretical analysis of the 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,

8

one unit of resource can be interpreted as the minimum desired resource and R as the ratio between maximum resource and minimum resource.

2.6 Setting η

The value of η can be viewed as a knob that can be tuned based on *practical* user constraints. Larger values of η correspond to a more aggressive elimination schedule and thus fewer rounds 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 4) they suggest choosing $\eta = e \approx 2.718$ but in practice we suggest taking η to be equal to 3 or 4 (if you don't know how to choose η, use $\eta = 3$).

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; specifically, there are $(\lfloor \log_\eta(R) \rfloor + 1) = s_{\max} + 1$ brackets. 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 computational 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.

2.7 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 3) 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 resource required to distinguish ν_2 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 2.1 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 4 and associated theorems/corollaries are provided that take into account the envelope functions and the distribution from which ν_i's are drawn. Of course we do not have knowledge of either function in practice, so we will hedge our aggressiveness with Hyperband. We show in Section 4.3.3 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.

3. Hyperparameter Optimization Experiments

In this section, we evaluate the empirical behavior of Hyperband with three different resource types: iterations, dataset subsamples, and feature samples. For all experiments, we compare Hyperband with three state-of-the-art Bayesian optimization algorithms — SMAC, TPE, and Spearmint. Whereas SMAC and TPE are tree-based Bayesian optimization methods, Spearmint (Snoek et al., 2012) uses Gaussian processes to model the problem. 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). For the deep learning experiments described in the next section, we also compare against a variant of SMAC named SMAC_early that uses the early termination criterion proposed in Domhan et al. (2015) for deep neural networks. Additionally, we show results for SuccessiveHalving corresponding to repeating the most exploration bracket of Hyperband. Finally for all experiments, we benchmark against standard random search and random_2×, which is a variant of random search with twice the budget of other methods (as described in Section 1).

3.1 Early stopping iterative algorithms for Deep Learning

We study a convolutional neural network with the same architecture as that used in Snoek et al. (2012) and Domhan et al. (2015) from cuda-convnet.[3] 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.

Datasets: We considered three image classification datasets: 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 dataset is 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

---

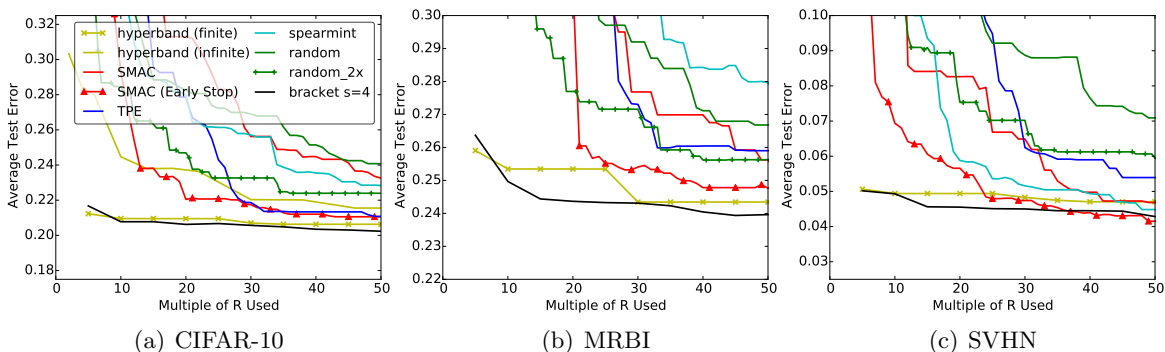3. The model specification is available at `http://code.google.com/p/cuda-convnet/`.

Figure 5: Average test error across 10 trials is shown in all plots. 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.

training, validation, and test respectively. For all datasets, 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 is set to 300 (or 30k total iterations). For SVHN, R is set to 600 (or 60k total iterations) to accommodate the larger training set. η was set to 4 for all experiments, resulting in 5 SUCCESSIVEHALVING brackets for HYPERBAND.

Results: Ten independent trials were performed for each searcher. In each trial, the searcher is given a total budget of $50R$ 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 completion. 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 iterations instead of compute time to make comparisons hardware independent.[4] Comparing progress by iterations instead of time ignores overhead costs not associated with training like 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 impact results significantly.

For CIFAR-10, the results in Figure 5(a) show that HYPERBAND is over an order of magnitude faster than its competitiors. For MRBI, HYPERBAND is over an order of magnitude faster than standard configuration selection approaches and $5\times$ faster than SMAC with early stopping. For SVHN, while HYPERBAND finds a good configuration faster, Bayesian optimization methods are competitive and SMAC with early stopping

---

4. 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.

outperforms Hyperband. We view SMAC with early stopping to be a combination of adaptive configuration selection and configuration evaluation. This result demonstrates that there is merit to incorporating early stopping with configuration selection approaches.

Across the three datasets, Hyperband and SMAC_early are the only two methods that consistently outperform random_2×. On these datasets, 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 2.6, 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 exploration bracket. Indeed, for these experiments, repeating the most exploratory bracket of Hyperband outperforms cycling through all the brackets. In fact, 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 not limit the maximum number of iterations and deploy infinite horizon Hyperband. We evaluate infinite horizon Hyperband for CIFAR-10, using $\eta = 4$ and a starting budget $B = 2R$. Figure 5(a) shows that infinite horizon Hyperband is competitive with other methods but does not perform as well as finite horizon Hyperband within the 50 times max iteration limit on total budget. Hence, we limit our focus on the finite horizon version of Hyperband for the remainder of our empirical studies.

Finally, CIFAR-10 is a very popular dataset and state-of-the-art models achieve much better accuracies than what is shown in Figure 5. 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 dataset 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 dataset is 18% error and hyperparameter optimized result is 15.0% for Snoek et al. (2012)[5] and 17.2% for Domhan et al. (2015). These results are better than our results on CIFAR-10 because they use 25% more data by including the validation set and also train for more epochs. The best model found by Hyperband achieved a test error of 17.0% when trained on the combined training and validation data for 300 epochs.

3.2 Dataset subsampling

We studied two different hyperparameter search optimization problems for which Hyperband uses dataset subsamples as the resource. The first adopts an extensive framework presented in Feurer et al. (2015a) that attempts to automate preprocessing and model selection. Due

---

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

(a) Validation Error on 117 Datasets (b) Test Error on 117 Datasets (c) Test Error on 21 Datasets
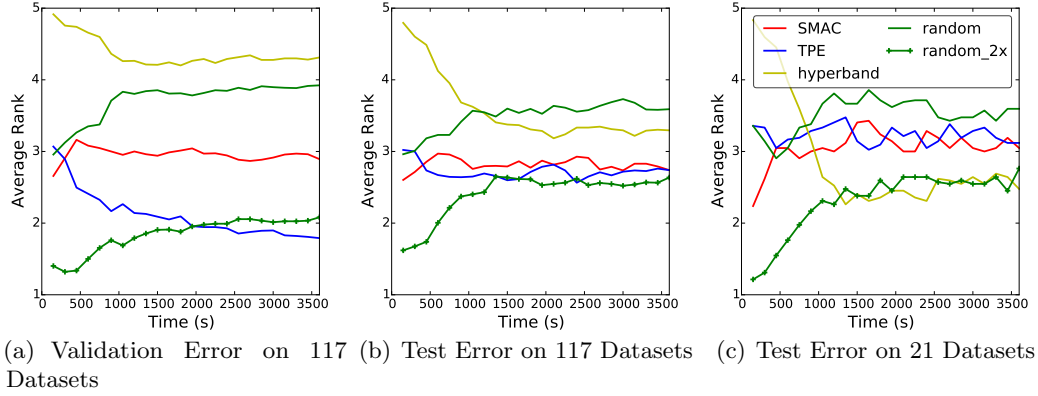
Figure 6: Average rank across all datasets for each searcher. For each dataset, the searchers are ranked according to the average validation/test error across 20 trials.

to certain limitations of the framework that fundamentally limit the impact of dataset downsampling, we focus on tuning a kernel classification task in the second experiment.

3.2.1 117 DATASETS

We use the framework introduced by Feurer et al. (2015a), which explores a structured hyperparameter search space with a total of 110 hyperparameters comprised of 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing methods. Similar to Feurer et al. (2015a), we impose a 3GB memory limit, a 6-minute timeout for each hyperparameter configuration and a one-hour time window to evaluate each searcher on each dataset. Twenty trials of each searcher were performed per dataset 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.

Datasets: Feurer et al. (2015a) used 140 binary and multiclass classification datasets 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 datasets. Due to the limitations of the experimental setup (discussed in Appendix A), we also separately considered 21 of these datasets which, based on preliminary evaluation with subsampled datasets, demonstrate at least modest (though still sublinear) training speedups due to subsampling. Specifically, each of these 21 datasets showed on average at least a $3\times$ speedup due to $8\times$ downsampling on 100 randomly selected hyperparameter configurations.

Hyperband Configuration: We run HYPERBAND with $\eta = 3$, i.e., each run of Successive Halving throws out $2/3$ of the arms and keeps the remaining $1/3$. R is set to equal the full training set size for each dataset and an upper bound on the maximum number of configurations for any round of SUCCESSIVEHALVING is set to $n_{\max} = \max\{9, R/1000\}$. This ensures that the most exploratory bracket of HYPERBAND will downsample at least twice and the minimum sample size allocated will be $\min\{1/9R, 1000\}$. As mentioned in Section 2.6, when $n_{\max}$ is specified, the only difference when running the algorithm is $s_{\max} = \lfloor \log_\eta(n_{\max}) \rfloor$ instead of $\lfloor \log_\eta(R) \rfloor$.

13

Results: The results on all 117 datasets in Figure 6(a,b) show that Hyperband outperforms random search in test error rank despite performing worse in validation error rank. Bayesian methods, which also exhibit overfitting, still outperform both random and Hyperband on test error rank. Notably, random_2× outperforms all other methods. However, for the subset of 21 datasets, Figure 6(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 datasets, the startup overhead was high relatively to total training time, while for larger datasets, only a handful of configurations could be trained within the hour window. Additionally, the results for the most exploratory bracket of Hyperband are not shown due to aforementioned limitations of the framework.

3.2.2 Kernel regularized least squares classification

Hyperband demonstrates modest improvements when using dataset downsampling with sublinear training algorithms, as illustrated in the previous results. We will next show that Hyperband can offer far greater speedups in settings where training time is superlinear in the number of training instances. In this setup, we focus on optimizing hyperparameters for a kernel-based classification task on CIFAR-10. We use 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.[6] 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). Hyperband is run with $\eta = 4$ and $R = 400$, with each unit of resource representing 100 datapoints. Similar to previous experiments, these inputs result in a total of 5 brackets. Each hyperparameter optimization algorithm is run for ten trials on Amazon EC2 `m4.2xlarge` instances; for a given trial, Hyperband is allowed to run for two outer loops, bracket $s = 4$ is repeated 10 times, and all other searchers are run for 12 hours. Figure 7 shows that Hyperband returns a good configuration after just the first SuccessiveHalving bracket in approximately 20 minutes; other searchers fail 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 performance for the two algorithms are the same. Random_2× is competitive with SMAC and TPE.

3.3 Feature subsampling to speed up approximate kernel classification

We next demonstrate the performance of Hyperband when using features as a resource, focusing on random feature approximations for kernel methods. Features are randomly generated using the method described in Rahimi and Recht (2007) to approximate the RBF kernel, and these random features are then used as inputs to a ridge regression classifier.

---

6. The default SVM method in Scikit-learn is single core and takes hours to train on CIFAR-10 whereas a block coordinate least squares solver takes less than 10 minutes on an 8 core machine.
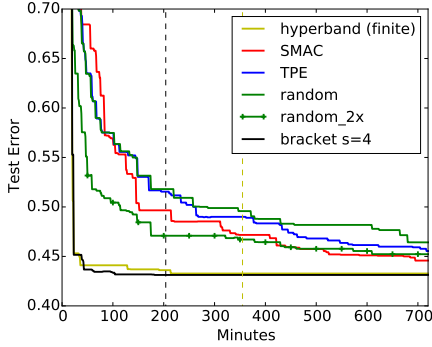
Figure 7: Average test error of the best kernel regularized least square classification model found by each searcher on CIFAR-10. The color coded dashed lines indicate when the last trial of a given searcher finished.
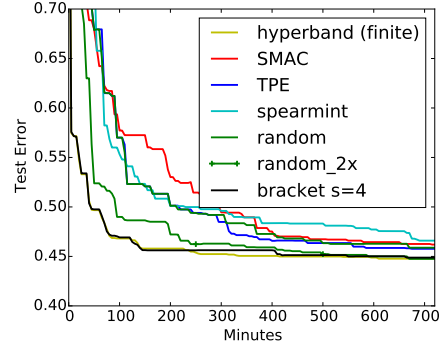
Figure 8: Average test error of the best random features model found by each searcher on CIFAR-10. The test error for Hyperband and bracket $s = 4$ are calculated in every evaluation instead of at the end of a bracket.

We consider hyperparameters of a random feature kernel approximation classifier trained on CIFAR-10, including preprocessing method, kernel length scale, and l_2 penalty. While it may seem natural to use infinite horizon Hyperband since we theoretically improve our approximation as we increase the number of features, in practice the amount of available machine memory imposes a natural upper bound on the number of features. We thus use finite horizion Hyperband with an upper bound of 100k random features, which will comfortably fit into a machine with 60GB of memory. Additionally, we set one unit of resource to be 100 features for an $R = 1000$, which gives 5 different brackets with $\eta = 4$. Each searcher is run for 10 trials, 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 6x faster than Bayesian methods and random search. Hyperband performs similarly to bracket $s = 4$. Random_2× outperforms Bayesian optimization algorithms.

3.4 Experimental discussion

For a given R, the most exploratory SuccessiveHalving round performed by Hyperband evaluates $\eta^{\lfloor \log_\eta(R) \rfloor}$ 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 $\frac{\eta^{\lfloor \log_\eta(R) \rfloor}}{(\lfloor \log_\eta(R) \rfloor + 1)}$. For the values of η and R used in our experiments, the maximum speedup over random search is approximately $50\times$ given linear training time. However, we observe a range of speedups from $6\times$ to $70\times$ faster than random search. The differences in realized speedup can be explained by two factors: (1) the scaling properties of total evaluation time as a function of the allocated resource and (2) the difficulty of finding a good configuration.

Total evaluation time consists of the time to train the model on a given amount of resource as well as overhead costs. If training time is superlinear as a function of the resource, then Hyperband can offer higher speedups. More generally, if training scales like a polynomial of degree $p > 1$, the maximum speedup of Hyperband over random search is approximately $\frac{\eta^{p-1}-1}{\eta^{p-1}}\eta^{\lfloor \log_\eta(R) \rfloor}$. In the kernel least square classifier experiment discussed in Section 3.2.2, the training time scaled quadratically as a function of the resource. This is why the realized speedup of $70\times$ is higher than that expected given linear scaling. However, the maximum speedup of $190\times$ is not realized for two reasons: (1) overhead associated with evaluating many configurations on fewer resources and (2) insufficient difficulty of the search space.

Overhead costs include costs associated with initializing a model, resuming previously trained models, and calculating validation error. If overhead costs are too high, the total evaluation time can be sublinear, hence giving poor speedups over random search. In the case of the downsampling experiments on 117 datasets presented in Section 3.2.1, Hyperband did not provide significant speedup because many datasets could be trained in a matter of a few seconds and the initialization cost was high relative to training time, hence many datasets had sublinear scaling of training time as a function of sample size.

Whether the full speedup is realized is determined by how hard it is to find a good configuration; if 10 randomly sampled configurations is sufficient to find a good hyperparameter setting then the benefit of Hyperband is muted whereas if it takes more than a few hundred configurations then Hyperband can offer significant speedup. Generally the difficulty of the problem scales with the dimension of the search space since coverage diminishes with dimensionality. 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; i.e. if $d = 3$ as in the features subsampling experiment, then $n = O(2^d = 8)$. Hence, Hyperband is only $6\times$ faster than random search on the feature subsampling experiment because 256 configurations are not needed to find a good configuration. For the neural network experiments however, we hypothesize that faster speedups are observed for Hyperband because the dimension of the search space is higher.

For all our experiments, Hyperband outperformed random search by a healthy margin because the validation error was for the most part monotonically decreasing with increasing resource, i.e. overfitting did not pose a problem. However, this convergence property can be violated if regularization hyperparameters are themselves a function of the resource. In the experiments shown in Section 3, the regularization terms were either independent of the number the resource in the case of the neural network experiments or scaled naturally with sample size or number of features as in the kernel experiments. In practice, it is important to choose regularization hyperparameters that are independent of the resource when possible. For example, when training random forests, the optimal maximum tree depth hyperparameter depends on the sample size but the relationship is unknown and can be dataset specific. An alternative regularization hyperparameter is minimum samples per leaf, which would scale better with size of training data as the resource.

16

4. 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 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.

4.1 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 $k = 1, 2, \ldots$ let $\ell_k : \mathcal{X} \to [0,1]$ 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 over $\mathcal{X}$. 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. 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 $k = 1, 2, \ldots$. 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 $\hat{x} \in \mathcal{X}$ that minimizes $\ell_*(\hat{x}) - \nu_*$ by drawing as many random configurations as desired, but using as few total resources as possible.

4.2 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.*[7]

---

7. We can always define $\ell_{i,k}$ so that convergence is guaranteed, i.e. taking the infimum.

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

The objective of the NIAB problem is to identify an arm $\widehat{i}$ with small $\nu_{\widehat{i}}$ using as few total pulls as possible. We are interested in characterizing $\nu_{\widehat{i}}$ 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 order to analyze the behavior of Hyperband in the NIAB setting, we must define a few additional objects. Define F to satisfy

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

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

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

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, γ is the deviation of the validation error of a configuration trained on a subset of resources versus the maximum number of allocatable resources. Define R as the first index such that $\gamma(R) = 0$ if it exists, otherwise set $R = \infty$. For $y \geq 0$ let $\gamma^{-1}(y) = \min\{j \in \mathbb{N} : \gamma(j) \leq y\}$, 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 dataset is used as a resource, then the maximum number of resources cannot exceed the full dataset size, and thus $\gamma(k) = 0$ for all $k \geq R$ where R is the (known) full size of the dataset. 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* 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.

4.3 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 (SHA) for a given set of limits ν_i and then consider the performance of SHA 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. By a simple modification of the proof of Jamieson and Talwalkar (2015) we have the following theorem

---

SUCCESSIVEHALVING (Infinite horizon)

Input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith 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_kth observed loss as S_{k+1}.

Output : $\widehat{i}, \ell_{\widehat{i}, \lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor}$ where $\widehat{i} = S_{\lceil \log_2(n) \rceil}$

HYPERBAND (Infinite horizon)

Input: None

For $k = 1, 2, \ldots$

    For $l \in \mathbb{N}$ s.t. $k - l \geq \log_2(l)$

        $B_{k,l} = 2^k$, $n_{k,l} = 2^l$

        $\widehat{i}_{k,l}, \ell_{\widehat{i}_{k,l}, \lfloor \frac{2^{k-1}}{l} \rfloor} \leftarrow$ SUCCESSIVEHALVING $(B_{k,l}, n_{k,l})$

Figure 9: 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). The HYPERBAND algorithm for the infinite horizon setting. HYPERBAND calls SUCCESSIVEHALVING as a subroutine.

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*

$$z_{SH} = 2\lceil \log_2(n) \rceil \max_{i=2,\ldots,n} i \left(1 + \gamma^{-1} \left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)\right)$$

$$\leq 2\lceil \log_2(n) \rceil \left(n + \sum_{i=2,\ldots,n} \gamma^{-1} \left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)\right)$$

If the SUCCESSIVEHALVING algorithm of Figure 9 is run with any budget $B > z_{SH}$ then an arm $\widehat{i}$ is returned that satisfies $\nu_{\widehat{i}} - \nu_1 \leq \epsilon$. Moreover, $|\ell_{\widehat{i}, \lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_1| \leq \epsilon/4$.

The next technical lemma will be used to characterize the problem dependent term $\sum_{i=2,\ldots,n} \gamma^{-1} \left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right)$ 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

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

19

If n arms are drawn randomly according to F whose limits correspond to $\nu_1 \leq \cdots \leq \nu_n$, then

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

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 $\widehat{i} \in [n]$ is returned such that with probability at least $1 - \delta$ we have $\nu_{\widehat{i}} - \nu_* \leq \left(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\right) + \epsilon$. In particular, if $B = 4\lceil \log_2(n) \rceil \mathbf{H}(F, \gamma, n, \delta)$ and $\epsilon = 4\left(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\right)$ then $\nu_{\widehat{i}} - \nu_* \leq 5\left(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*\right)$ with probability at least $1 - \delta$.*

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

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

which implies $\mathbb{E}[\min_{i=1,\ldots,n} \nu_i - \nu_*] \approx F^{-1}(\frac{1}{n}) - \nu_*$. 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 SHA to a naive approach that uniformly allocates resources to a fixed set of n arms.

4.3.1 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 $\widehat{i} = \arg\min_{i=1,\ldots,n} \ell_j(X_i)$. 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 $\nu_{\widehat{i}} - \nu_* \leq 2\left(F^{-1}\left(\frac{\log(1/\delta)}{n}\right) - \nu_*\right)$. 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 $\nu_{\widehat{i}} - \nu_* \geq 2(F^{-1}(\frac{\log(c/\delta)}{n + \log(c/\delta)}) - \nu_*)$, 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 $\widehat{i} \in [n]$ that satisfies $\nu_{\widehat{i}} - \nu_* = F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*$ with probability at least $1 - \delta$. This guarantee is similar to the result in Proposition 4. However, SUCCESSIVEHALVING achieves its guarantee as long as[8]

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

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).

4.3.2 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

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

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

$$F(x) \simeq (x - \nu_*)^\beta \qquad (6)$$

Here, a large value of β implies that it is very rare to draw a limit close to the optimal value ν_*. 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 $\nu_{\widehat{i}}$ that satisfies $\nu_{\widehat{i}} - \nu_* \lesssim \left(\frac{\log(1/\delta)}{n}\right)^{1/\beta}$ 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

Uniform allocation $\quad B \simeq \Delta^{-(\alpha+\beta)}\log(1/\delta)$

SUCCESSIVEHALVING $\quad B \simeq \log_2(\Delta^{-\beta}\log(1/\delta))\left[\Delta^{-\alpha}\log(1/\delta) + \frac{\Delta^{-\beta} - \Delta^{-\alpha}}{1 - \alpha/\beta}\log(1/\delta)\right]$

$\qquad \simeq \log(\Delta^{-1}\log(1/\delta))\log(\Delta^{-1})\, \Delta^{-\max\{\beta,\alpha\}}\log(1/\delta)$

---

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

then both also satisfy $\nu_{\hat{i}} - \nu_* \lesssim \Delta$ with probability at least $1 - \delta$.[9] 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 4.3.3 we show how Hyperband addresses this issue.

The second parameterization of F is the following discrete distribution:

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

for some set of unique scalars $\mu_1 < \mu_2 < \cdots < \mu_K$. Note that by letting $K \to \infty$ this discrete CDF can approximate any piecewise-continuous CDF to arbitrary accuracy. In this setting, we have that both uniform allocation and SuccessiveHalving output a $\nu_{\hat{i}}$ 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, if $n \simeq q^{-1} \log(1/\delta)$ and the measurement budgets of the uniform allocation (Equation 3) and SuccessiveHalving (Equation 4) satisfy

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

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

then an arm that is in the best q-fraction of arms is returned, i.e. $\hat{i}/K \approx q$ and $\nu_{\hat{i}} - \nu_* \lesssim \Delta_{\lceil qK \rceil}$, with probability at least $1 - \delta$. 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.

4.3.3 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, l) pairs defined in the Hyperband algorithm of Figure 9, let $\delta_{k,l} = \frac{\delta}{2k^3}$. For all (k, l) define $\mathcal{E}_{k,l}$ to be the event that

---

9. These quantities are intermediate results in the proofs of the theorems of Section 4.3.3.

- $\nu_{\widehat{i}_{k,l}} - \nu_* \le \left(F^{-1}\left(\frac{\log(2/\delta_{k,l})}{n_{k,l}}\right) - \nu_*\right) + \epsilon_{k,l}$ for some $\epsilon_{k,l} \ge 4\left(F^{-1}\left(\frac{\log(2/\delta_{k,l})}{n_{k,l}}\right) - \nu_*\right)$, and

- $B_{k,l} = 2^k > 4l\mathbf{H}(F, \gamma, 2^l, 2k^3/\delta, \epsilon_{k,l}) = 4\lceil\log_2(n_{k,l})\rceil\mathbf{H}(F, \gamma, n_{k,l}, \delta_{k,l}, \epsilon_{k,l})$.

By Corollary 3 we have $\mathbb{P}\left(\bigcup_{k=1}^{\infty}\bigcup_{l=1}^{k}\mathcal{E}_{k,l}^c\right) \le \sum_{k=1}^{\infty}\sum_{l=1}^{k}\delta_{k,l} = \sum_{k=1}^{\infty}\frac{\delta}{2k^2} \le \delta$. For any $B \in \mathbb{N}$, let $\widehat{i}_B$ 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 $l_B \le k_B$ be the largest value that makes $\mathcal{E}_{k_B,l_B}$ hold. Then after plugging in the relevant quantities we obtain

$$\nu_{\widehat{i}_B} - \nu_* \le \left(F^{-1}\left(\frac{\log(4\lfloor\log_2(B)\rfloor^3/\delta)}{2^{l_B}}\right) - \nu_*\right) + \epsilon_{k_B,l_B} + \gamma\left(\lfloor\frac{2^{\lfloor\log_2(B)\rfloor-1}}{\lfloor\log_2(B)\rfloor}\rfloor\right).$$

Also note that on stage k at most $\sum_{i=1}^{k} iB_{i,1} \le k\sum_{i=1}^{k} B_{i,1} \le 2kB_{k,l} = 2\log_2(B_{k,l})B_{k,l}$ total samples have been taken. While this guarantee holds for general F, γ, the value of l_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 4.3.2 for better interpretation.

Theorem 5 *Assume that Assumptions 1 and 2 of Section 4.2 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 $\widehat{i}_T$ be the empirically best-performing arm output from SUCCESSIVEHALVING from the last round k of HYPERBAND of Figure 9 after T total samples had been taken from all rounds, then*

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

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

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 $\nu_{\widehat{i}_T} - \nu_* \le c\left(\frac{\log(T)\overline{\log}(\log(T)/\delta)}{T}\right)^{1/(\alpha+\beta)}$. 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, l in the infinite horizon HYPERBAND algorithm with $n_{k,l}$ arms drawn, consider the setting where the jth pull of the ith arm results in a stochastic loss $Y_{i,j} \in [0, 1]$ such that $\mathbb{E}[Y_{i,j}] = \nu_i$ and $\mathbb{P}(\nu_i - \nu_* \le \epsilon) = c_1^{-1}\epsilon^{\beta}$. If $\ell_j(i) = \frac{1}{j}\sum_{l=1}^{j} Y_{i,l}$ then with probability at least $1 - \delta/2$ we have $\forall k \ge 1, 0 \le l \le k, 1 \le i \le n_{k,l}, 1 \le j \le B_k$,*

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

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 $1 - \delta$

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

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 4.2 hold and that the sampled loss sequences obey the parametric assumptions of Equations 5 and 7. For any $T \in \mathbb{N}$, let $\widehat{i}_T$ be the empirically best-performing arm output from SUCCESSIVEHALVING from the last round k of HYPERBAND of Figure 9 after T total samples had been taken from all rounds. Fix $\delta \in (0,1)$ and $q \in (1/K, 1)$ and let $z_q = \log(q^{-1})(\Delta_{\lfloor qK \rfloor}^{-\alpha} + \frac{1}{qK}\sum_{i=\lfloor qK \rfloor}^K \Delta_i^{-\alpha})$. Once $T = \widetilde{\Omega}\left(z_q \log(z_q) \log(1/\delta)\right)$ total pulls have been made by HYPERBAND we have $\nu_T - \nu_* \leq q\mathbf{1}_{q>1/K}$ 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 $\log(1/\delta)\log(q^{-1})(\Delta_{\lfloor qK \rfloor}^{-\alpha} + \frac{1}{qK}\sum_{i=\lfloor qK \rfloor}^K \Delta_i^{-\alpha})$. 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 $\log(1/\delta)\log(K)\sum_{i=2}^K \Delta_i^{-2}$ which matches known upperbounds 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.

4.4 Finite horizon setting ($R < \infty$)

In this section we analyze the algorithm described in Section 2, i.e., finite horizon HYPERBAND. We present similar theoretical guarantees as in Section 4.3 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*

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

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

SUCCESSIVEHALVING (Finite horizon)

input: Budget B, and n arms where $\ell_{i,k}$ denotes the kth loss from the ith 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_kth observed loss as S_{k+1}.

Output : $\widehat{i}, \ell_{\widehat{i},R}$ where $\widehat{i} = \arg\min_{i \in S_{s+1}} \ell_{i,R}$

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 $s = s_{\max}, s_{\max} - 1, \ldots, 0$

      $B_{k,s} = 2^k$, $n_{k,s} = \lceil \frac{2^k \eta^s}{R(s+1)} \rceil$

      $\widehat{i}_s, \ell_{\widehat{i}_s,R} \leftarrow$ 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.

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 according to F then an arm $\widehat{i} \in [n]$ is returned such that with probability at least $1 - \delta$ we have $\nu_{\widehat{i}} - \nu_* \leq (F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*) + \epsilon$. 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 $\nu_{\widehat{i}} - \nu_* \leq 5(F^{-1}(\frac{\log(2/\delta)}{n}) - \nu_*)$ with probability at least $1 - \delta$.*

As in Section 4.3.2 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

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

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

then both also satisfy $\nu_{\widehat{i}} - \nu_* \lesssim \Delta$ 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

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

SUCCESSIVEHALVING:

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

then an arm that is in the best q-fraction of arms is returned, i.e. $\widehat{i}/K \approx q$ and $\nu_{\widehat{i}} - \nu_* \lesssim \Delta_{\lceil qK\rceil}$, 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 4.3.3 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 incites beyond the sample complexities sufficient for SUCCESSIVEHALVING to succeed, given immediately above.

One important thing to note about the budget of SUCCESSIVEHALVING in the finite horizon setting is that for all sufficiently large B (e.g. $B > 3R$) and all distributions F we have that B 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 2.

5. Related Work

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

5.1 Hyperparameter optimization

Bayesian optimization techniques model the conditional probability $p(f|\lambda)$ of a configuration's performance on a metric f given a set of hyperparameters λ. Sequential Model-based Algorithm Configuration (SMAC), Tree-structure Parzen Estimator (TPE), and Spearmint are the three most well known Bayesian searchers. SMAC uses random forests to model $p(f|\lambda)$ 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 utilizes Gaussian processes (GP) to model $p(f|\lambda)$ and performs slice sampling over the GP's hyperparameters (Snoek et al., 2012).

Previous work has 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, 2015a). One of the first extensive surveys of these three methods by Eggensperger et al. (2013) introduces a benchmark library for hyperparameter optimization called HPOlib, which we use in our experiments. With the exception of Bergstra et al. (2011) and Thornton et al. (2013), most of these papers do not extensively compare random search to the Bayesian optimization methods.

As for adaptive configuration evaluation, early stopping poor configurations is not a new idea, however previous approaches either require strong assumptions or use heuristics to perform adaptive resource allocation. Domhan et al. (2015); Swersky et al. (2013, 2014); György and Kocsis (2011); Agarwal et al. (2011) propose methods that make 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) proposes 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."

To overcome these difficulties, Sparks et al. (2015) proposed a halving style algorithm that did not require explicit convergence behavior, and Jamieson and Talwalkar (2015) analyzed a similar algorithm, providing theoretical guarantees and encouraging empirical results. Unfortunately, the halving style algorithms suffer from the n vs B/n issue discussed in Section 2.1.

Finally, Klein et al. (2016) recently introduced Fabolas, a new Bayesian optimization method that combines adaptive selection and evaluation. Similar to Swersky et al. (2013, 2014), it models the conditional validation error as a Gaussian process using a kernel that captures the covariance with downsampling rate to allow for adaptive evaluation. While we intended to compare Hyperband with Fabolas, we encountered some technical difficulties

when using the provided package[10] and are working with the authors of Klein et al. (2016) to resolve the issues.

5.2 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 (Bubeck and Cesa-Bianchi, 2012; Even-Dar et al., 2006), 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 in 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 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 the β parametrization of F used in Section 4.3.2. However, their algorithm was derived specifically for this particular β parameterization, and furthermore must estimate β before running the algorithm, limiting the algorithm's practical applicability. Also, the algorithm assumes stochastic losses from the arms and thus specific convergence behavior, making it inapplicable in our hyperparameter optimization setting; see Jamieson and Talwalkar (2015) for detailed discussion motivating the non-stochastic setting for hyperparameter optimization. Two related lines of work that both make use of an underlying metric space are Gaussian process optimization (Srinivas et al., 2009) and X-armed bandits (Bubeck et al., 2011), or bandits defined over a metric space. However, these works need to know something about the underlying function (e.g. an appropriate kernel or level of smoothness) and assume stochastic rewards.

In contrast, Hyperband is devised for the non-stochastic setting and automatically adapts to unknown F without making any parametric assumptions. One could argue that our work is the first to propose a practical pure exploration algorithm for infinitely many armed bandits and test it on a real application.

6. Extensions and open questions

Aside from other ways to define and allocate resources, other potential extensions to our work including parallelizing Hyperband for distributed computing, adjusting for training methods with different convergence rates, and combining Hyperband with non-random sampling methods.

6.1 Distributed implementations

Hyperband is easy to parallelize 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

---

10. The package provided by Klein et al. (2016) is available at https://github.com/automl/RoBO.

decreases exponentially as arms are whittled down so a more sophisticated job priority queue must be managed. Of course, one could also parallelize each individual execution of R.

6.2 Adjusting for different convergence rates

In order for Hyperband to be generally applicable to all machine learning problems, one key challenge will be adjusting resource allocation for configurations with different convergence rates. Configurations can have different convergence rates if they are in different model classes (i.e. iterative clustering algorithms versus logistic regression using SGD) and even if they are in the same model class but have different hyperparameters (i.e. neural network with differing number of layers or hidden units). The issue arises when configurations with slower convergence rates give a better final model. While if time is a priority, it may make sense to optimize for speed and accuracy, in general, Hyperband should be able to handle differing convergence rates . This was a potential issue for the neural network experiments discussed in Section 3.1 with the learning rate but, the issue was not observed in our experimental results.

6.3 Incorporating non-random sampling

Hyperband could 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 performance the performance of Hyperband by giving better coverage of the search space. Another approach is to use meta-learning to define intelligent priors informed by previous experimentation. Finally, exploring the best way to combine Hyperband with adaptive search strategies like Bayesian Optimization is a very promising future direction.

References

A. Agarwal, J. Duchi, P. L. Bartlett, and C. Levrard. Oracle inequalities for computationally budgeted model selection. In *COLT*, 2011.

Alekh Agarwal, Sham Kakade, Nikos Karampatziakis, Le Song, and Gregory Valiant. Least squares revisited: Scalable approaches for multi-class prediction. In *Proceedings of The 31st International Conference on Machine Learning*, pages 541–549, 2014.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. In *JMLR*, 2012.

J. Bergstra et al. Algorithms for hyper-parameter optimization. In *NIPS*, 2011.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1–122, 2012.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *JMLR*, 12:1655–1695, 2011.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pages 23–37. Springer, 2009.

A. Carpentier and M. Valko. Simple regret for infinitely many armed bandits. In *ICML*, 2015.

Ian Dewancker, Michael McCourt, Scott Clark, Patrick Hayes, Alexandra Johnson, and George Ke. A stratified analysis of bayesian optimization methods. *arXiv preprint arXiv:1603.09441*, 2016.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015.

K. Eggensperger et al. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *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. *JMLR*, 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 et al. Efficient and robust automated machine learning. In *NIPS*, 2015a.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *AAAI*, 2015b.

A. György and L. Kocsis. Efficient multi-start strategies for local search algorithms. *JAIR*, 41, 2011.

F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, 2011.

K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, 2015.

K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014.

Z. Karnin, T. Koren, and O. Somekh. Almost optimal exploration in multi-armed bandits. In *ICML*, 2013.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 2015.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

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. In *JMLR*, 2015.

Hugo Larochelle et al. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007.

Yuval Netzer et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *ICPR*, 2012.

J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

J. Snoek et al. Bayesian optimization using deep neural networks. In *ICML*, 2015.

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

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv:0912.3995*, 2009.

K. Swersky, J. Snoek, and R. Adams. Multi-task bayesian optimization. In *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 *KDD*, 2013.

Appendix A. Additional Experimental Results

Additional details for experiments presented in Section 2 and 3 are provided below.

A.1 LeNet experiment

The search space for the LeNet example discussed in Section 2.3 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 2.3. 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 3.1, the exact architecture used is the 18% model provided on cuda-convnet for CIFAR-10.[11]

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) is 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) is 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: Adaptive allocation for hyperparameter optimization can be thought of as a form of early stopping where less promising configurations are halted before completion. Domhan et al. (2015) propose an early stopping method for neural networks and combine it with SMAC to speed up hyperparameter optimization. Their method stops training a configuration if the probability of the configuration beating the

---

11. 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.

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. 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).[12] Comparing performance by the total multiple of R used 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 Datasets experiment

For the experiments discussed in Section 3.2.1, like Feurer et al. (2015a), we impose a 3GB memory limit, a 6-minute timeout for each hyperparameter configuration and a one-hour time window to evaluate each searcher on each dataset. Moreover, we evaluate the performance of each searcher by aggregating results across all datasets 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 average error across all trials for each (dataset-searcher) pair. Then, the performance of each searcher is ranked by dataset and averaged across all datasets. 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. (2015a) split each dataset 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

---

12. We used the code provided at https://github.com/automl/pylearningcurvepredictor.
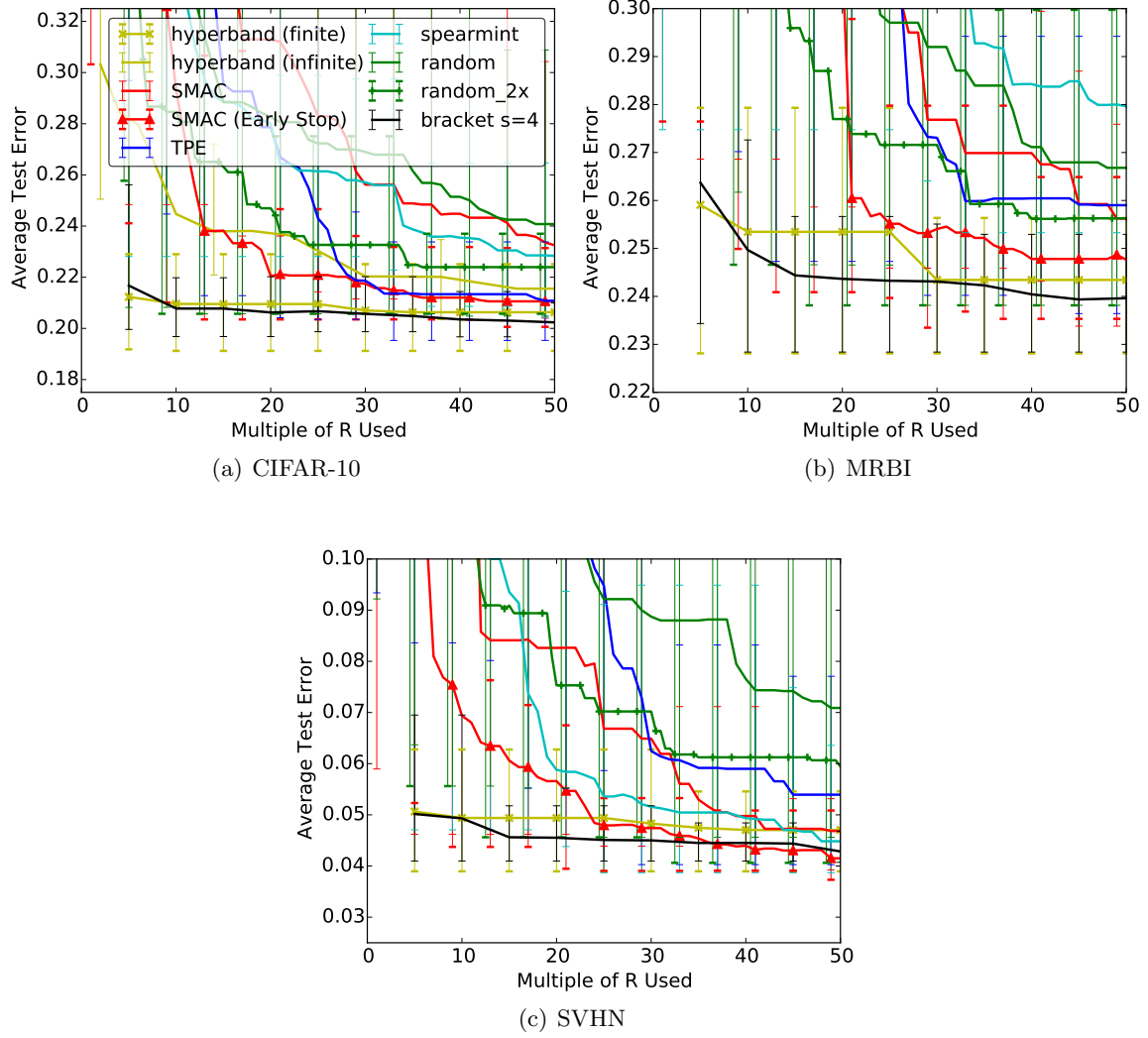
(a) CIFAR-10

(b) MRBI

(c) SVHN

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

of the data for training, but split the rest of the data into two equally sized validation and test sets. We report results on both the validation and test data. Moreover, we perform 20 trials of each (dataset-searcher) pair, and as in Feurer et al. (2015a) we keep 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[13] resulting in random search being able to test 600 configurations on some datasets but just dozens on others. Our Hyperband algorithm was designed under the implicit assumption that computation scaled at least linearly with the dataset size. For very small datasets that are trained in seconds, 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 quadratic storage 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 dataset and not on subsampled datasets. A searcher like Hyperband that uses a subsampled dataset could spend significant time training on a subsample only to error out when attempting to train it on the full dataset.
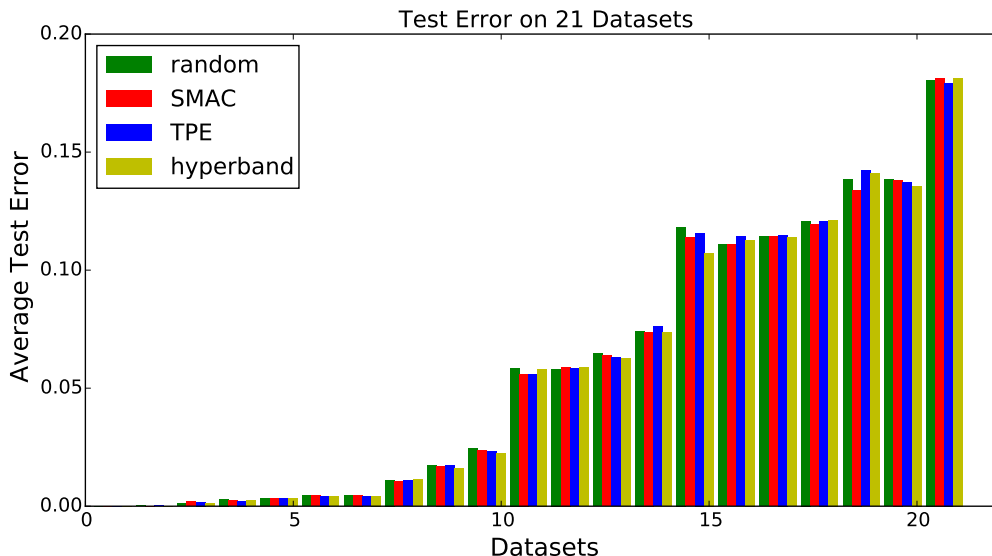

Figure 12: Average test error by dataset after one hour in the subset of 21 datasets benefited from subsampling. The models were able to classify perfectly on some datasets.

We note that while average ranking plots like those of Figure 6 are an effective way to aggregate information across many searchers and datasets, they provide no indication about the *magnitude* of the differences of the methods. Figure 12 and Figure 13, which present the test errors for all searchers on each dataset, corroborates the relative ranking of the

---

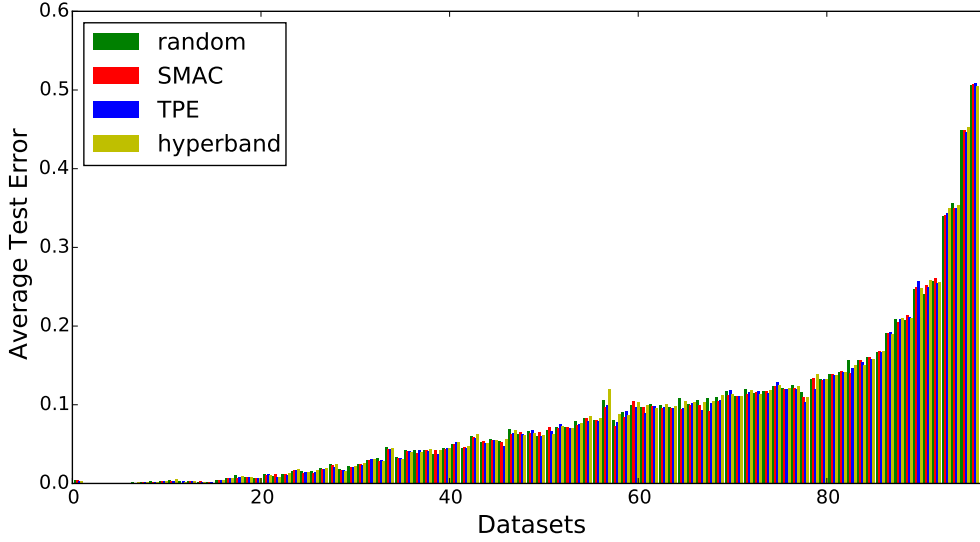13. Training set size ranges from 670 to 73,962 observations, and number of features ranges from 1 to 10,935.

Figure 13: Average test error by dataset after one hour on the remaining 96 datasets not included in the subset of 21 datasets that benefit more from subsampling.

various searchers while also highlighting the fact that the differences in test errors across the searchers is fairly small.

A.4 Kernel classification experiments

Table 4 shows the hyperparameters and associated ranges considered in the kernel least squares classification experiment discussed in Section 3.2.2.

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 3.2.2.

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
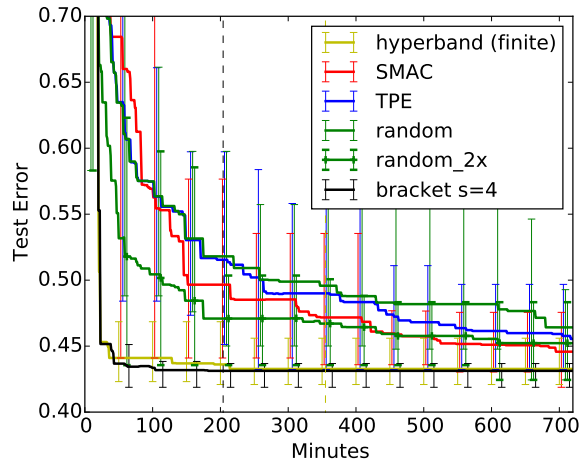
Figure 14: Average test error of the best kernel regularized least square classification model found by each searcher on CIFAR-10. The color coded dashed lines indicate when the last trial of a given searcher finished. Error bars correspond to observed minimum and maximum test error across 10 trials.

to the inverse of the scaled cost term C. Additionally, the average test error with associated minimum and maximum ranges across 10 trials are show in Figure 14.

Table 5 shows the hyperparameters and associated ranges considered in the random features kernel approximation classification experiment discussed in Section 3.3. 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 associated minimum and maximum ranges across 10 trials are show in Figure 15.

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

Table 5: Hyperparameter space for random feature kernel approximation classification problem discussed in Section 3.3.
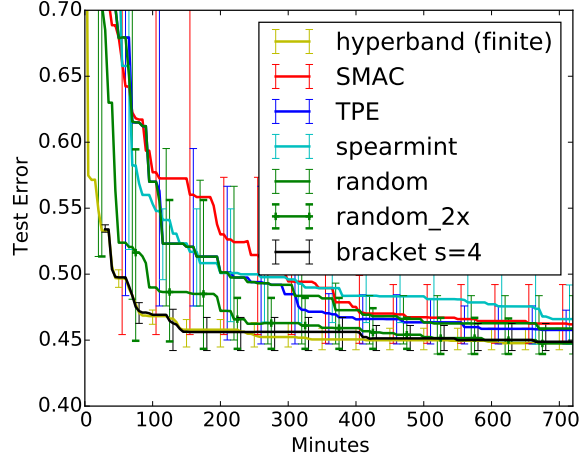
Figure 15: Average test error of the best random features model found by each searcher on CIFAR-10. The test error for HYPERBAND and bracket $s = 4$ are calculated in every evaluation instead of at the end of a bracket. Error bars correspond to observed minimum and maximum test error across 10 trials.

Appendix B. Proofs

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:

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

For notational ease, let $\ell_{i,j} := \ell_j(X_i)$. Moreover, define $[\cdot] = \{\{\cdot\}_{t=1}\}_{i=1}^{n}$ so that $[\ell_{i,t}] = \{\{\ell_{i,t}\}_{t=1}^{\infty}\}_{i=1}^{n}$. Without loss of generality, we may assume that the n infinitely long loss sequences $[\ell_{i,t}]$ with limits $\{\nu_i\}_{i=1}^{n}$ were fixed prior to the start of the game so that the $\gamma(t)$ envelope is also defined for all time and are fixed. Let Ω be the set that contains all possible sets of n infinitely long sequences of real numbers with limits $\{\nu_i\}_{i=1}^{n}$ and envelopes $[\gamma(t)]$, that is,

$$\Omega = \left\{ [\ell'_{i,t}] : [\, |\ell'_{i,t} - \nu_i| \leq \gamma(t) \,] \wedge \lim_{\tau \to \infty} \ell'_{i,\tau} = \nu_i \quad \forall i \right\}$$

where we recall that $\wedge$ is read as "and" and $\vee$ is read as "or." Clearly, $[\ell_{i,t}]$ is a single element of Ω.

We present a proof by contradiction. We begin by considering the singleton set containing $[\ell_{i,t}]$ under the assumption that the SUCCESSIVEHALVING algorithm fails to identify the best arm, i.e., $S_{\lceil \log_2(n) \rceil} \neq 1$. We then consider a sequence of subsets of Ω, with each one contained in the next. The proof is completed by showing that the final subset in our sequence (and thus our original singleton set of interest) is empty when $B > z_{SH}$, which contradicts our assumption and proves the statement of our theorem.

Let $T = \{i \in [n] : \nu_i \leq \nu_1 + \epsilon/2\}$ denote the "good" set of arms of which any of them returned would be acceptable.

To reduce clutter in the following arguments, it is understood that S_k' for all k in the following sets is a function of $[\ell_{i,t}']$ in the sense that it is the state of S_k in the algorithm when it is run with losses $[\ell_{i,t}']$. We now present our argument in detail, starting with the singleton set of interest, and using the definition of S_k in Figure 9.

$$\left\{ [\ell_{i,t}'] \in \Omega : [\ell_{i,t}' = \ell_{i,t}] \wedge S_{\lceil \log_2(n) \rceil}' \cap T = \emptyset \right\}$$

$$= \left\{ [\ell_{i,t}'] \in \Omega : [\ell_{i,t}' = \ell_{i,t}] \wedge \bigvee_{k=1}^{\lceil \log_2(n) \rceil} \{S_k' \cap T = \emptyset, \ S_{k-1}' \cap T \neq \emptyset\} \right\}$$

$$= \left\{ [\ell_{i,t}'] \in \Omega : [\ell_{i,t}' = \ell_{i,t}] \wedge \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \bigwedge_{j \in S_k' \cap T} \left\{ \sum_{i \in S_k'} \mathbf{1}\{\ell_{i,r_k}' < \ell_{j,r_k}'\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$= \left\{ [\ell_{i,t}'] \in \Omega : [\ell_{i,t}' = \ell_{i,t}] \wedge \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \bigwedge_{j \in S_k' \cap T} \left\{ \sum_{i \in S_k'} \mathbf{1}\{\nu_i - \nu_j < \ell_{j,r_k}' - \nu_j - \ell_{i,r_k}' + \nu_i\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$\subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \bigwedge_{j \in S_k' \cap T} \left\{ \sum_{i \in S_k'} \mathbf{1}\{\nu_i - \nu_j < |\ell_{j,r_k}' - \nu_j| + |\ell_{i,r_k}' - \nu_i|\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$\subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \bigwedge_{j \in S_k' \cap T} \left\{ \sum_{i \in S_k'} \mathbf{1}\{2\gamma(r_k) > \nu_i - \nu_j\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}, \tag{8}$$

where the last set relaxes the original equality condition to just considering the maximum envelope γ that is encoded in Ω. The summation in Eq. 8 only involves the ν_i, and this summand is maximized if each S_k' contains the first $|S_k'|$ arms. Hence we have,

$$Eq. (8) \subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \bigwedge_{j \in S_{k-1}' \cap T} \left\{ \sum_{i=1}^{|S_k'|} \mathbf{1}\{2\gamma(r_k) > \nu_i - \nu_j\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$\subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i=1}^{|S_k'|} \mathbf{1}\{2\gamma(r_k) > \max\{(\nu_1 + \epsilon) - (\nu_1 + \epsilon/2), \nu_i - \nu_1\}\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$\subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i=1}^{|S_k'|} \mathbf{1}\{2\gamma(r_k) > \max\{\epsilon/2, \nu_i - \nu_1\}\} > \lfloor |S_k'|/2 \rfloor \right\} \right\}$$

$$= \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ 2\gamma(r_k) > \max\{\epsilon/2, \nu_{\lfloor |S_k'|/2 \rfloor + 1} - \nu_1\} \right\} \right\}$$

$$\subseteq \left\{ [\ell_{i,t}'] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ r_k < \gamma^{-1}\left(\max\left\{ \tfrac{\epsilon}{4}, \tfrac{\nu_{\lfloor |S_k'|/2 \rfloor + 1} - \nu_1}{2} \right\} \right) \right\} \right\}, \tag{9}$$

40

where we use the definition of γ^{-1} in Eq. 9. Next, we recall that $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$. By plugging in this value for r_k and rearranging we have that

$$Eq. \ (9) \subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \frac{B/2}{\lceil \log_2(n) \rceil} < (\lfloor |S'_k|/2 \rfloor + 1)(1 + \gamma^{-1} \left(\max \left\{ \frac{\epsilon}{4}, \frac{\nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1}{2} \right\} \right)) \right\} \right\}$$

$$= \left\{ [\ell'_{i,t}] \in \Omega : \frac{B/2}{\lceil \log_2(n) \rceil} < \max_{k=0,\dots,\lceil \log_2(n) \rceil - 1} (\lfloor |S'_k|/2 \rfloor + 1)(1 + \gamma^{-1} \left(\max \left\{ \frac{\epsilon}{4}, \frac{\nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1}{2} \right\} \right)) \right\}$$

$$\subseteq \left\{ [\ell'_{i,t}] \in \Omega : B < 2 \lceil \log_2(n) \rceil \max_{i=2,\dots,n} i \left(\gamma^{-1} \left(\max \left\{ \frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2} \right\} \right) + 1 \right) \right\} = \emptyset$$

where the last equality holds if $B > z_{SH}$.

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

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

∎

B.2 Proof of Lemma 2

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

$$\xi_1 = \{\nu_1 \leq F^{-1}(p_n)\}$$

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

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

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

which just equals μM. 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 $\epsilon \geq 4(F^{-1}(p_n) - \nu_*)$.

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

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

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

which shows the desired result.

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

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

$$\leq \sum_{i=1}^{n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{16}, \tfrac{\nu_i - \nu_*}{4}\right\}\right)$$

$$= \sum_{i=1}^{n} \min\{M, \gamma^{-1}\left(\tfrac{\nu_i - \nu_*}{4}\right)\}$$

$$\leq n\mu + \sqrt{2n\mu M \log(1/\delta)} + \tfrac{2}{3} M \log(1/\delta)$$

$$\leq \left(\sqrt{n\mu} + \sqrt{\tfrac{2}{3} M \log(2/\delta)}\right)^2 \leq 2n\mu + \tfrac{4}{3} M \log(2/\delta).$$

A direct computation yields

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

$$= \mathbb{E}[\gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{16}, \tfrac{\nu_i - \nu_*}{4}\right\}\right)]$$

$$= \gamma^{-1}\left(\tfrac{\epsilon}{16}\right) F(\nu_* + \epsilon/4) + \int_{\nu_* + \epsilon/4}^{\infty} \gamma^{-1}(\tfrac{t - \nu_*}{4}) dF(t)$$

so that

$$\sum_{i=2}^{n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i - \nu_1}{2}\right\}\right) \leq 2n\mu + \tfrac{4}{3} M \log(2/\delta)$$

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

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[14], and let $\hat{i} = \arg\min_{i=1,\dots,n} \ell_j(X_i)$. Let $\nu_i = \ell_*(X_i)$ and without loss of generality assume $\nu_1 \leq \dots \leq \nu_n$. If*

$$B \geq n\gamma^{-1}\left(\tfrac{1}{2}(F^{-1}(\tfrac{\log(1/\delta)}{n}) - \nu_*)\right) \tag{10}$$

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

---

14. Here j can be bounded (finite horizon) or unbounded (infinite horizon).

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

$$\mathbb{P}\left(\ell_*(X_{i_*}) - \nu_* \le \epsilon\right) = \mathbb{P}\left(\bigcup_{i=1}^{n} \{\ell_*(X_i) - \nu_* \le \epsilon\}\right)$$

$$= 1 - (1 - F(\nu_* + \epsilon))^n \ge 1 - e^{-nF(\nu_* + \epsilon)},$$

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

$$\ell_*(X_{\widehat{i}}) - \nu_* \le \ell_j(X_{\widehat{i}}) - \nu_* + \gamma(j) \le \ell_j(X_{i_*}) - \nu_* + \gamma(j)$$

$$\le \ell_*(X_{i_*}) - \nu_* + 2\gamma(j) \le F^{-1}\left(\frac{\log(1/\delta)}{n}\right) - \nu_* + 2\gamma(j).$$

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[15] $\inf_{x \in [\nu_*, 1-\nu_*]} \inf_{\Delta \in [0, 1-x]} \frac{F(x+\Delta) - F(x+\Delta/2)}{F(x+\Delta) - F(x)} \ge c$. 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 $\ell_j : \mathcal{X} \to \mathbb{R}$ $\forall j \in \mathbb{N}$ with $\ell_* := \lim_{j\to\infty} \ell_j$, $\nu_* = \inf_{x \in \mathcal{X}} \ell_*(x)$ such that $\sup_{x \in \mathcal{X}} |\ell_j(x) - \ell_*(x)| \le \gamma(j)$ and $\mathbb{P}_\mu(\ell_*(X) - \nu_* \le \epsilon) = F(\epsilon)$. Moreover, if n configurations $X_1, \ldots, X_n$ are drawn from μ and $\widehat{i} = \arg\min_{i \in 1, \ldots, n} \ell_{B/n}(X_i)$ then with probability at least δ*

$$\ell_*(X_{\widehat{i}}) - \nu_* \ge 2\left(F^{-1}\left(\frac{\log(c/\delta)}{n + \log(c/\delta)}\right) - \nu_*\right)$$

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

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

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

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, $|\ell_j(x) - \ell_*(x)| \le \gamma(j)$ for all $x \in \mathcal{X}$.

---

15. 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 $c = 1 - 2^{-\beta} \ge \frac{1}{2}\min\{1, \beta\}$.

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 $\ell_*(X_{\widehat{i}}) - \nu_* \geq F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) + \frac{1}{2}\gamma(B/n)$ 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

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

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 $\widehat{i} = \arg\min_i \ell_{B/n}(X_i)$ then $\ell_{B/n}(X_{\widehat{i}}) \in I_1$ and $\ell_*(X_{\widehat{i}}) \in I_2$. That is, $\nu_{\widehat{i}} - \nu_* \geq \widehat{\nu} - \nu_* + \frac{1}{2}\gamma(j) = F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_* + \frac{1}{2}\gamma(j)$. So if we wish $\nu_{\widehat{i}} - \nu_* \leq 2(F^{-1}(\frac{\log(c/\delta)}{n+\log(c/\delta)}) - \nu_*)$ 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

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

and

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

We conclude that

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

$$\le c\, p_n^{-\alpha/\beta} \log(1/\delta) + cn \begin{cases} \log(1/p_n) & \text{if } \alpha = \beta \\ \frac{1 - p_n^{1-\alpha/\beta}}{1 - \alpha/\beta} & \text{if } \alpha \ne \beta. \end{cases}$$

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

$$\mathbf{H}(F, \gamma, n_{k,l}, \delta_{k,l}) \le c\, p_{n_{k,l}}^{-\alpha/\beta} \log(1/\delta_{k,l}) + cn_{k,l} \begin{cases} \log(1/p_{n_{k,l}}) & \text{if } \alpha = \beta \\ \frac{1 - p_{n_{k,l}}^{1-\alpha/\beta}}{1 - \alpha/\beta} & \text{if } \alpha \ne \beta. \end{cases}$$

$$\le c\log(k/\delta)\left[\Delta^{-\alpha} + \begin{cases} \Delta^{-\beta} \log(\Delta^{-1}) & \text{if } \alpha = \beta \\ \frac{\Delta^{-\beta} - \Delta^{-\alpha}}{1 - \alpha/\beta} & \text{if } \alpha \ne \beta \end{cases}\right]$$

$$\le c\log(k/\delta) \min\left\{\tfrac{1}{|1-\alpha/\beta|}, \log(\Delta^{-1})\right\} \Delta^{-\max\{\beta,\alpha\}}$$

where the last line follows from

$$\Delta^{\max\{\beta,\alpha\}} \frac{\Delta^{-\beta} - \Delta^{-\alpha}}{1 - \alpha/\beta} = \beta \frac{\Delta^{\max\{0,\alpha-\beta\}} - \Delta^{\max\{0,\beta-\alpha\}}}{\beta - \alpha}$$

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

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

$$B_{k,l} < \min\{2^k : 2^k > 4\lceil \log(n_{k,l}) \rceil \mathbf{H}(F, \gamma, n_{k,l}, \delta_{k,l})\}$$

$$< \min\{2^k : 2^k > c\log(k/\delta)\log(\log(k/\delta))z_\Delta\}$$

$$\le c z_\Delta \log(\log(z_\Delta)/\delta) \log(\log(\log(z_\Delta)/\delta))$$

$$= c z_\Delta \overline{\log}(\log(z_\Delta)/\delta).$$

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

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

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

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

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 $\nu_{\widehat{i}} - \nu_* < \Delta_2$. Thus, if $\nu_{\widehat{i}} - \nu_* \leq \left(F^{-1}(p_n) - \nu_* \right) + \epsilon$ then we set

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

so that

$$\nu_{\widehat{i}} - \nu_* \leq \max \left\{ 5 \left(F^{-1}(p_n) - \nu_* \right), \Delta_2 \right\} = \Delta_{\lfloor \max\{2, cKp_n\} \rfloor}.$$

We treat the case when $5 \left(F^{-1}(p_n) - \nu_* \right) \leq \Delta_2$ and the alternative separately.

First assume $5 \left(F^{-1}(p_n) - \nu_* \right) > \Delta_2$ so that $\epsilon = 4 \left(F^{-1}(p_n) - \nu_* \right)$ and $\mathbf{H}(F, \gamma, n, \delta, \epsilon) = \mathbf{H}(F, \gamma, n, \delta)$. We also have

$$\gamma^{-1} \left(\frac{F^{-1}(p_n) - \nu_*}{4} \right) \leq c \left(F^{-1}(p_n) - \nu_* \right)^{-\alpha} \leq c \Delta_{\lfloor p_n K \rfloor}^{-\alpha}$$

and

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

46

so that

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

$$\leq c \Delta_{\lfloor p_n K \rfloor}^{-\alpha} \log(1/\delta) + \frac{cn}{K} \sum_{i=\lfloor p_n K \rfloor}^K \Delta_i^{-\alpha}.$$

Now consider the case when $5\big(F^{-1}(p_n) - \nu_*\big) \leq \Delta_2$ so that $\epsilon \geq \tfrac{4}{5}\Delta_2$. In this case $F(\nu_* + \epsilon/4) = 1/K$, $\gamma^{-1}\left(\tfrac{\epsilon}{16}\right) \leq c\Delta_2^{-\alpha}$, and $\int_{\nu_* + \epsilon/4}^\infty \gamma^{-1}(\tfrac{t - \nu_*}{4}) dF(t) \leq c \sum_{i=2}^K \Delta_i^{-\alpha}$ so that

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

$$\leq c(\log(1/\delta) + n/K)\Delta_2^{-\alpha} + \frac{cn}{K} \sum_{i=2}^K \Delta_i^{-\alpha}.$$

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 $5\big(F^{-1}(1/K) - \nu_*\big) \leq \Delta_2$. That is, when $p_{n_{k,l}} \leq 1/K$ the algorithm has found the best arm but will continue to take samples indefinitely. 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

$$\mathbf{H}(F, \gamma, n_{k,l}, \delta_{k,l}, \epsilon_{k,l}) \leq c \begin{cases} (\log(1/\delta_{k,l}) + \tfrac{n_{k,l}}{K})\Delta_2^{-\alpha} + \tfrac{n_{k,l}}{K} \sum_{i=2}^K \Delta_i^{-\alpha} & \text{if } 5\big(F^{-1}(p_{n_{k,l}}) - \nu_*\big) \leq \Delta_2 \\ \Delta_{\lfloor p_{n_{k,l}} K \rfloor}^{-\alpha} \log(1/\delta_{k,l}) + \tfrac{n_{k,l}}{K} \sum_{i=\lfloor p_{n_{k,l}} K \rfloor}^K \Delta_i^{-\alpha} & \text{if otherwise} \end{cases}$$

$$\leq c\log(k/\delta) \begin{cases} \sum_{i=2}^K \Delta_i^{-\alpha} & \text{if } q = 1/K \\ \Delta_{\lfloor qK \rfloor}^{-\alpha} + \tfrac{1}{qK} \sum_{i=\lfloor qK \rfloor}^K \Delta_i^{-\alpha} & \text{if otherwise.} \end{cases}$$

$$\leq c\log(k/\delta)\Delta_{\lfloor qK \rfloor}^{-\alpha} + \frac{1}{qK} \sum_{i=\lfloor qK \rfloor + 1}^K \Delta_i^{-\alpha}$$

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 $z_q = \log(q^{-1})(\Delta_{\lfloor qK \rfloor}^{-\alpha} + \tfrac{1}{qK} \sum_{i=\lfloor qK \rfloor}^K \Delta_i^{-\alpha})$, we apply the exact sequence of steps as in the proof of Theorem 5 to obtain

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

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 $\lfloor \tfrac{2^{k-1}}{l} \rfloor \geq cB_{k,l}/\log(B_{k,l})$ times so the possible error is bounded by $\gamma(B_{k,l}/\log(B_{k,l})) \leq c \left(\tfrac{\log(B_{k,l})}{B_{k,l}}\right)^{1/\alpha} \leq c \left(\tfrac{\log(T)^2 \log(\log(T))}{T}\right)^{1/\alpha}$. This is dominated by $\Delta_{\lfloor qK \rfloor}$ 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, $s = \min\{t \in \mathbb{N} : nR(t+1)\eta^{-t} \leq B\}$. It is straightforward to verify that $B \geq z_{SH}$ ensures that $r_0 \geq 1$ and $n_s \geq 1$.

The remainder of the proof goes nearly identically up to Eq. (9), so that's where we will begin. Namely, if the claim is *not* true, then we must have

$$r_k < \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_{n_{k+1}+1} - \nu_1}{2}\right\}\right) \quad \text{for some } k \in \{0, \ldots, s-1\}. \tag{11}$$

Combining Eq. (11) with

$$r_k + 1 \geq \widetilde{r}_k = R\eta^{k-s} = \frac{nR}{\widetilde{n}_k}\eta^{-s} = \frac{nR\eta}{\widetilde{n}_{k+1}}\eta^{-s} > \frac{B}{(n_{k+1}+1)\eta\log_\eta(\frac{nR}{B})}. \tag{12}$$

and rearranging, we observe that

$$Eq.\ (11) \implies B < \eta\log_\eta(\tfrac{nR}{B}) \max_{k=0,\ldots,s-1}(n_{k+1}+1)\left[1 + \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_{n_{k+1}+1}-\nu_1}{2}\right\}\right)\right]$$

$$\leq \eta\log_\eta(\tfrac{R}{r}) \max_{i=n_s+1,\ldots,n} i\left[1 + \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right)\right]$$

$$\leq \eta\log_\eta(\tfrac{R}{r})\left[n + \max_{i=n_s+1,\ldots,n} i\gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right)\right]$$

$$\implies \text{contradiction for the specified value of } B.$$

In addition, we note that

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

∎