Adaptive Hyperparameter Tuning

Determined AI

APRIL 2020

Table of Contents

Introduction

Machine learning model performance hinges on effective hyperparameters that govern everything from feature selection to model architecture to the training process. In the case of deep learning models, it can take days or weeks to train a model under just *one* hyperparameter configuration. Because hyperparameter optimization suffers from the curse of dimensionality,[1] it seems impossible to test enough hyperparameter configurations to ensure that we aren't leaving model performance on the table.

In this white paper, we explore Determined AI's solution to this challenging problem. We reveal the inner workings of our adaptive hyperparameter search capability, a theoretically sound approach to hyperparameter optimization that leverages *partial training* to explore significantly more configurations compared to existing methods. Beyond the algorithms, we explore how Determined *productionizes* hyperparameter tuning, offering the ease-of-use that machine learning engineers crave, and the system performance and reliability that production infrastructure teams demand.

At Determined AI, we build software that helps organizations build better machine learning models faster, and automatic hyperparameter tuning is a foundational component to achieve this goal.

Motivation

Why is automatic hyperparameter tuning a mission-critical capability for organizations developing deep learning models?

Consider a machine learning engineer at a financial services company building a credit card fraud detection engine. Given that the bank houses a vast history of transactions labeled fraudulent or not, machine learning is a natural fit to solve this problem.

Armed with some historical data, they experiment with a couple of models and achieve promising results on held out test data. They've proven the concept, but they wonder if there is room for improvement. The machine learning engineer is squarely in the *hyperparameter tuning* phase of the model development lifecycle, asking themselves: would different features, model architectures, or learning process hyperparameters produce a better model?

[1] The space of possible configurations is exponential in the number of hyperparameters.

They might ask:

- What other features might yield better model performance?

  For example:

  - ✓ Internally available features about customers and merchants.
  - ✓ Derived features like a transaction's standard deviation from the merchant's recent historical average.
  - ✓ For point-of-sale transactions, publicly available economic and demographic features describing the merchant location.

- What model architecture works best?
  - ✓ What kinds of layers should comprise the neural network?
  - ✓ How should they orient and parameterize the layers (e.g., size, activation function)?
- What learning algorithm hyperparameters should we set to yield an effective model within a reasonable amount of time?
  - ✓ What gradient descent mini batch size?
  - ✓ How should the machine learning engineer parameterize the optimizer (e.g., learning rate)?

These are just a handful of the decisions that the machine learning engineer makes during the model development lifecycle. They can manually experiment to navigate these decisions, but hand tuning isn't feasible due to the curse of dimensionality — even if a model has just ten categorical hyperparameters with five possible values per hyperparameter, and each configuration takes one minute to train, it would take nearly twenty compute years to entirely cover the search space. With deep learning, the situation is even more daunting since there are typically tens of hyperparameters and training a single configuration can take hours to weeks.

> "The machine learning engineer is squarely in the hyperparameter tuning phase of the model development lifecycle, asking themselves: would different features, model architectures, or learning process hyperparameters produce a better model?"

Unfortunately, existing hyperparameter tuning techniques fall short in this setting. Random and grid search really only apply when the hyperparameter search space is small and it's inexpensive to evaluate points in the space (think seconds). Recent research has focused on Bayesian methods[2] that optimize hyperparameter configuration selection. The core idea of Bayesian optimization is to model the probability of a hyperparameter configuration's performance on the objective and sample hyperparameter configurations from this distribution. Nevertheless, Bayesian optimization is challenging in deep learning, as it assumes that the initial random exploration of the search space provides adequate information to accurately model performance. In high-dimensional hyperparameter spaces, this is inherently problematic, particularly for models that take days or weeks to train. Thus, like random and grid search, Bayesian methods are still "restricted to problems of moderate dimension"[3] unless there is some attempt to heuristically downsample or reduce the dimensionality of the hyperparameter space.

To help machine learning engineers tune their deep learning models, Determined AI offers adaptive hyperparameter tuning, a practical and broadly applicable technique that adaptively focuses on worthwhile areas of the search space in a resource-aware fashion, yielding significant model performance improvements much more quickly than existing approaches. The algorithm is particularly well-suited to the large search spaces and long training cycles common in deep learning. By leveraging our adaptive approach, machine learning engineers can produce more high performing models in less time.

[2] Practical Bayesian Optimization of Machine Learning Algorithms
[3] Bayesian Optimization in High Dimensions via Random Embeddings

Adaptive Hyperparameter Search

We all know what it's like to watch a game on TV when one side is winning by such a large margin that the game is "over before it's over." While a comeback is possible, it just doesn't feel worth watching for another half hour to the end, so we grab the remote and flip to another game. Just as we might change the channel in this scenario, Determined's adaptive hyperparameter search abandons unpromising hyperparameter configurations that are unlikely to outperform other configurations under consideration.

Algorithm Overview

Our adaptive hyperparameter search algorithm is based on the Successive Halving (SHA) and Hyperband algorithms presented by our cofounder Ameet Talwalkar et al.[4,5] The idea behind SHA follows directly from its name: allocate a resource budget to a set of hyperparameter configurations, evaluate the performance of all configurations, throw out the worst half[6], and repeat until one configuration remains. Hyperband extends SHA to address a key drawback: since we don't know how quickly or smoothly the optimization objective converges, we can't say a priori how aggressively to prune underperforming configurations. Should we train many configurations for a small amount of time on average, or fewer configurations longer? Hyperband simply runs SHA at different points on this tradeoff spectrum.

> "Hyperband *partially* evaluates hyperparameter configurations, adaptively dedicating resources to more promising configurations."

While the intuition of adaptive downsampling is clear — don't waste resources evaluating losing hyperparameter configurations — we are playing with fire if we tune our models strictly based on gut feeling or heuristics. In the next section, we describe the strong theoretical guarantees backing this simple algorithm.

[4] Non-stochastic best arm identification and hyperparameter optimization

[5] Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

[6] Despite the name "Successive Halving," we can prune by factors other than 2. In Determined AI's platform, this configurable factor defaults to 3.

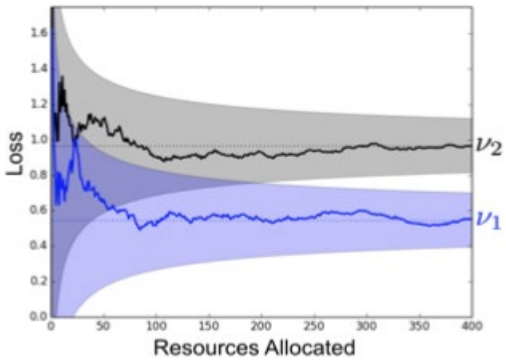
Early Stopping Safety

How do we know that it's safe for Hyperband to abandon hyperparameter configurations based on partial evaluation? What if we abandon a late-blooming configuration too soon? While a detailed theoretical analysis is beyond the scope of this white paper, we provide an intuitive, high-level description of theoretical properties of Determined's adaptive hyperparameter tuning.

> "While the intuition of Hyperband is clear, we are playing with fire if we tune our models based on gut feeling or heuristics."

The theory behind Hyperband relies on "envelope functions," which plot over time (or iterations) the maximum distance of intermediate observed losses from the true *terminal validation loss* observed if we train fully.

Suppose there are n configurations and consider the task of identifying the winning configuration whose terminal validation loss is v_w. The optimal strategy would allocate to each configuration the minimum resource required to distinguish its terminal validation loss v_i from v_w. Mathematically, this means training just long enough so that the envelope functions depicted above bound the intermediate loss to be less than $(v_i - v_w) / 2$ away from the terminal value. The resource budget required by SHA when given access to these envelope functions is only a constant factor away from this optimal approach because it capitalizes on



configurations that are easy to distinguish from v_w. Of course, it is unrealistic to assume knowledge of these envelope functions. Remarkably, because Hyperband calls SHA for geometrically increasing values of n, Hyperband's required resource budget is only a log factor larger than SHA. We refer the reader to the original research paper[7] for more details about the theoretical underpinnings of SHA and Hyperband.

[7] Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

Productionization

Adaptive downsampling algorithms like SHA and Hyperband have achieved impressive results on many hyperparameter tuning benchmarks, spawning a wave of related research.[8] Beyond research, these downsampling methods are available in many open source and commercial automated ML offerings.[9] This widespread adoption validates adaptive downsampling as the preferred technique for tuning deep learning models. *What, then, differentiates Determined's offering?* Determined's Hyperband implementation is production-grade, offering both first-rate **ease-of-use** and superior system **performance** to delight machine learning and infrastructure teams alike.

Ease-of-use is one of the most important considerations in production; if an advanced method is too cumbersome to use, the machine learning engineer may never realize its benefits. In a recent survey on hyperparameter tuning methods in machine learning research, over 40% of respondents resort to manual tuning, while the vast majority of automatic tuning leverages grid or random search.[10] If the research community itself struggles in applying state-of-the-art hyperparameter tuning methods, how can we expect machine learning engineers in industry to employ these more advanced methods? At Determined, we obsess over ease-of-use; to that end, we deliver a user-friendly, configuration-based API for hyperparameter tuning.

"Determined's Hyperband implementation is production-grade, offering both first-rate ease-of-use and superior system performance to delight machine learning and infrastructure teams alike."

[8] The original Hyperband paper has been cited 350+ times as of the writing of this white paper
[9] E.g., KubeFlow, tune, Keras Tuner, HpBandSter, scikit-hyperband, and hypersearch
[10] Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020

While users enjoy Determined's intuitive hyperparameter tuning API, the platform automatically handles system performance concerns: cluster resource sharing, scheduling, fault tolerance, parallelization, and autoscaling.

In this section, we discuss the three design pillars that together make hyperparameter tuning in Determined a delight rather than a systems engineering chore: a user-friendly API, workload orchestration, and experiment parallelization.

User-Friendly API

Despite their drawbacks in deep learning, the classical hyperparameter tuning methods have clean and intuitive APIs: random search only takes as input the number of configurations to evaluate and a training resource budget, and grid search is comparably simple, only requiring a sampling interval for numeric hyperparameters. In contrast, SHA and its variants have internal settings like the percentage of configurations to prune after evaluation. To facilitate use and increase adoption, Determined offers a simplified variant of adaptive downsampling with an intuitive interface akin to that of random search. In the experiment configuration, we only need to specify the maximum number of hyperparameter configurations to explore, and the maximum number of training steps that any trial can take (i.e., the per-configuration resource budget constraint).

Advanced users can leverage Determined's `adaptive` searcher should they desire more control over the internal settings. For practical reasons, this searcher's configuration options don't mirror Hyperband's exactly. For instance, Determined accepts a `mode` parameter that lets users adaptively downsample more aggressively than Hyperband. Hyperband is conservative by design because, in general, we don't know how quickly the validation metric will converge (i.e., we don't know the behavior of the envelope functions). In practice, if a user believes that the validation metric converges quickly, then they can adaptively downsample more like vanilla SHA in order to speed up the optimization process or to explore more points in the hyperparameter space. We refer the reader to our product documentation for further details on the `adaptive` searcher's configuration options.

Orchestration

The applications we use every day — email, business productivity applications, e-commerce sites, navigation apps that get us from A to B — are built to withstand the inconvenient forces lurking in production. Systems must perform in the face of bursty and often unpredictable resource needs, high concurrency, and hardware failures. *Orchestration* — the coordination and management of the systems powering these critical applications — is a tall order given everything that can go wrong. Orchestrating resources for large-scale hyperparameter tuning workloads is similarly rife with these challenges, and even faces additional headwinds specific to deep learning:

1. Hardware unexpectedly fails all the time. Hyperparameter tuning must self-heal in such a scenario. Better yet, a trial should pick up where it left off when the hardware failure occurred. This necessitates automatic checkpointing during trial execution.
2. Hardware expectedly goes away, too, e.g., when using lower cost preemptible cloud instances that cost up to 70% less than regular instances.
3. When running on the cloud, hardware ought to autoscale up to meet the demand of a large hyperparameter tuning workload, and autoscale down on completion.
4. Whether on prem or in the cloud, a machine learning team might need to pause a long-running hyperparameter tuning experiment in order to manually prioritize a different workload, and then resume the job where it left off.
5. Hyperparameter tuning experiments must be reproducible under any of the above circumstances.

As a holistic training platform designed from the ground up for deep learning, Determined handles *all* of the above, allowing the machine learning engineer to focus on models rather than systems.

Parallelization

Given the (often headline-worthy) resource needs for training deep learning models, parallelization is virtually a hard requirement for hyperparameter tuning methods. Fortunately, SHA and Hyperband are parallelizable in myriad ways. In Determined, we parallelize Hyperband by running each SHA trial independently and synchronously evaluating trials of a given SHA run when it's time to abandon the low performing configurations.

It's a bit tricky to get parallelization right — adaptive downsampling algorithms are certainly not as straightforward to implement as, say, random search — and many of the previously mentioned orchestration challenges prove more difficult in our setting. If a trial fails, adaptive downsampling requires more careful workload coordination and state management than, say, random search would need in the same scenario.

At Determined, system performance guides our product design alongside ease-of-use, so we continue to push the envelope on the parallelization front. We are actively investigating removing the synchronous evaluation barrier of SHA, specifically by early-promoting promising configurations in the asynchronous variant of SHA (aptly named ASHA). By combining early promotion with early stopping, a given hyperparameter tuning workload is less susceptible to resource underutilization (either due to stragglers, or during the final stages of SHA when just a handful of configurations remain). For further detail on ASHA, we refer the reader to the ASHA research paper our co-founder Ameet Talwalkar et al. recently published.[11]

[11] A System for Massively Parallel Hyperparameter Tuning

About Determined

In the machine learning model development lifecycle, hyperparameter tuning can make or break the ultimate usefulness of the model. Along with resource sharing, experiment tracking capabilities, and distributed training, Determined's adaptive hyperparameter tuning gives machine learning teams a suite of model development and training tools to deliver *better models, faster*. Learn more on our website, or drop us an email.

Determined AI