HyperDrive: Flexible and Efficient Parallel Hyperparameter Exploration *

Jeff Rasley[1] Yuxiong He[2] Feng Yan[3]
Olatunji Ruwase[2] Rodrigo Fonseca[1]

[1]*Brown University* [2]*Microsoft* [3]*University of Nevada, Reno*

Abstract

The quality of machine learning and deep learning models are very sensitive to many different adjustable parameters (hyperparameters) that are set before training even begins. Efficient hyperparameter exploration is of great importance to practitioners in order to find high-quality models at affordable time and cost. This is a challenging process due to a large search space, expensive training runtime, sparsity of good configurations, and scarcity of time/resources. In this work, we design a flexible and efficient parallel hyperparameter exploration framework, HyperDrive, that enables pluggable hyperparameter search algorithms and policies while being agnostic to specific learning frameworks and domains. We also propose a scheduling algorithm POP that infuses probabilistic model-based classification with dynamic scheduling and early termination to jointly optimize quality and cost. We evaluate HyperDrive by implementing several existing scheduling algorithms. We evaluate POP by using workloads from supervised and reinforcement learning domains and show that we speedup the training process by 6.7x compared with basic approaches like random search and up to 2.1x compared with state-of-the-art approaches.

1. Background

Many machine-learning frameworks have been introduced to help developers build and train models for solving different AI tasks across supervised, unsupervised, and reinforcement learning domains. The performance of these models are very sensitive to many adjustable parameters (hyperparameters), which are set prior to training. Hyperparameter examples include learning rate, number or size of hidden layers in a deep neural network, and many more. Hyperparameter exploration searches across different configurations that are derived from a base model, where each *configuration* is a specific set of hyperparameter values or model structure. Typically the goal is to find configurations that optimize model performance (e.g., accuracy, reward). Exploration involves generating configurations from a large space of hyperparameters, and scheduling/training these configurations for some amount of time.

Efficient hyperparameter exploration is of great importance to practitioners in order to improve model performance, reduce training time, and optimize resource usage. It is challenging to design effective scheduling frameworks and algorithms that efficiently explore hyperparameter values while obtaining high model performance and optimizing time/resource costs. The first challenge is due to the size of the hyperparameter search space and the expensive training process for each candidate model. Figure 1 shows model performance (validation accuracy) for 50 configurations on an image classification task (CIFAR-10). To fully explore all 50 configurations it would take 4 days of computing. This problem is made even worse for larger models/datasets that can take days/weeks to train a single model. Second, often only few configurations lead to high performance while a majority perform very poorly. For example, in Figure 1 only 3 configurations exceed 75% accuracy while most don't exceed 20%. Therefore, simple/popular approaches such as random hyperparameter generation, which bet heavily on luck, are very inefficient in finding quality configurations under reasonable time/cost constraints. In addition, improving hyperparameter tuning can involve many other factors, such as incorporating different learning domain goals and different DL/ML framework subtleties. It is challenging to design an effective framework to support such a wide range of usage scenarios.

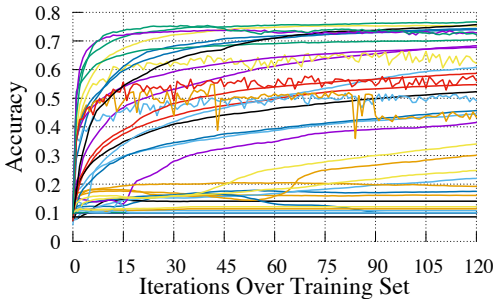

Figure 1. 50 random CIFAR-10 hyperparameter configs.

Adaptive hyperparameter generation techniques like Bayesian optimization can be useful but are difficult to parallelize and typically don't address how to efficiently run configurations. Each configuration is typically executed to the same maximum iteration (which can be large), ignoring the fact that some configurations could show their value early on. For example, in Figure 1, many configurations do not learn, i.e., with accuracy similar to random (e.g., 10%), and can be quickly identified and terminated to save resources.

* An extended version of this work will appear in Middleware 2017 [4].

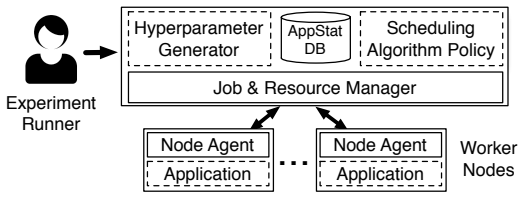

Figure 2. HyperDrive architecture

In addition, allocating the same amount/type of resources for all configurations is not the best method if different configurations demonstrate different learning progress. How to effectively schedule different configurations across both time and space dimensions of resources is largely unattended by prior work.

2. HyperDrive Architecture

We design a pluggable hyperparameter exploration framework, HyperDrive, which serves as a generic framework for hyperparameter exploration (Figure 2). HyperDrive largely decouples the hyperparameter generation and scheduling of resources/time for candidate configurations from the specific type of model and/or framework. HyperDrive provides an API that allows users to build Hyperparameter Generators (HG) and Search Algorithm Policies (SAP). We have built several hyperparameter generation techniques as HGs, such as random search and Bayesian optimization. SAPs allow users to centrally allocate resources based on monitored user-defined application-level metrics such as accuracy, loss, reward, model size, etc. We have built SAPs that reproduce parts of prior work [5, 2] and also a new proposed SAP that we introduce next. Through a simple client library HyperDrive supports training models using almost any framework, such as Caffe, TensorFlow, CNTK, and others.

3. POP Scheduling

We propose a new scheduling algorithm POP that dynamically classifies configurations into three categories: Promising, Opportunistic, and Poor, and uses these in two major ways. First, along the time dimension, we quickly identify and terminate poor configurations that are not learning, by incorporating application-level knowledge from model owners. For example, classification tasks have known non-learning random accuracy values which can be used to prune configurations. Second, early classification of promising configurations that are more likely to lead to high task performance through the use of a probabilistic learning curve model [2]. This type of model allows us to predict future model performance (e.g., validation accuracy) by computing the probability $P(m)_i$ of configuration i reaching a model performance y after a future epoch m based on the observed model performance history. Specifically, $P(y(m)_i \geq y | y(1 : m-1)_i)$. We then compute the configuration's expected remaining time to reach a target performance within a user-specified experiment time/resource budget. Finally, we use prioritized execution of promising configurations by devoting more resources to them without starving opportunistic configurations; while balancing exploitation of promising configurations and exploration of opportunistic configurations.

4. Evaluation

In order to evaluate the expressiveness of HyperDrive and to compare the performance of our POP policy with prior work we build two additional policies: Bandit [5] and EarlyTerm [2] based on prior work.

We evaluated the effectiveness of these three policies using HyperDrive with both supervised and reinforcement learning workloads. For supervised learning, we use the popular image classification task CIFAR-10. We use a CNN based on Krizhevsky's cuda-convnet [3] from prior work and explored 14 hyperparameters. For reinforcement learning, we use a model [1] for an OpenAI Gym task called LunarLander, exploring 11 hyperparameters. In this task, an agent is rewarded as it controls a robot to land between two flags. The problem is considered solved if the agent achieves an average reward of 200 over 100 consecutive trials.

In order to evaluate the performance of different policies with the Krizhevsky CIFAR-10 model we compare the training time to reach a given target accuracy using a 4 node GPU (K40m) cluster. We select a target accuracy of 77% based on the domain knowledge of this model. POP reached our target accuracy with a median of only 2.8 hours, whereas Bandit took 4.5 hours and EarlyTerm took 6.1 hours. POP outperforms Bandit by 1.6x and outperforms EarlyTerm by 2.1x. We evaluate the LunarLander model with each policy across a cluster of 15 c4.xlarge instances on AWS. We set a target reward of 200, which is natural since the goal of the task is explicitly defined. We observe that POP achieves a median time to target 2.1x faster than Bandit and 1.3x faster than EarlyTerm. Further analysis of these policies is in [4].

5. Ongoing Work

HyperDrive enables model owners to use application-level metrics to guide hyperparameter exploration. In many cases there is a primary metric that is being optimized like accuracy or reward. However, additional metrics of concern can be impacted by hyperparameter choices as well, such as inference/serving latencies or model sparsity/compressibility. We have been exploring these areas and have seen great initial results with this type of exploration.

We are actively working with engineers at Microsoft to productize HyperDrive internally, which will evolve HyperDrive's usability, scalability, and effectiveness. Hyperparameter exploration will continue to be an active research area but currently there are limited options for researchers to develop/evaluate parallel approaches that incorporate techniques along *both* hyperparameter generation and scheduling (e.g., early termination, suspend/resume). We believe that HyperDrive is a first step in addressing these problems.

References

[1] ASADI, K., AND WILLIAMS, J. D. Sample-efficient Deep Reinforcement Learning for Dialog Control. *arXiv CoRR abs/1612.06000* (2016).

[2] DOMHAN, T., SPRINGENBERG, J. T., AND HUTTER, F. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. IJCAI 2015.

[3] KRIZHEVSKY, A. cuda-convnet. `https://code.google.com/p/cuda-convnet/`, 2017.

[4] RASLEY, J., HE, Y., YAN, F., RUWASE, O., AND FONSECA, R. HyperDrive: Exploring Hyperparameters with POP Scheduling. Middleware 2017.

[5] SPARKS, E. R., TALWALKAR, A., HAAS, D., FRANKLIN, M. J., JORDAN, M. I., AND KRASKA, T. Automating Model Search for Large Scale Machine Learning. SoCC 2015.