Towards Characterizing DNNs to Estimate Training Time using HARP (HPC Application Resource (runtime) Predictor

Swathi Vallabhajoyula*
vallabhajosyula.2@buckeyelink.osu.edu
The Ohio State University
Columbus, Ohio, USA

Rajiv Ramnath*
ramnath.6@osu.edu
The Ohio State University
Columbus, Ohio, USA

ABSTRACT

Training DNN models for accuracy is resource intensive and needs high-performance computing resources. These resources come with a cost, and repeatedly training models with default allocations (complete node) for significant periods is expensive. Optimally allocating resources (roughly as needed by the job) allows the user to cut execution costs (sometimes even without compromising execution times). This also enables better utilization of the clusters by making them more available. Finetuning every job is exhaustive in terms of time to learn and understand the application and hardware characteristics. We built a framework called HARP that tries to learn from execution patterns (with some help from the user) and predict resource needs for the required configurations. This study explores the potential scalability of such models across different axis - input, hardware, and application hyperparameters. We also explore the transferability of such models within similar applications/ models (DNN-16 layers and VGG 16, or VGG16 and ResNet50).

CCS CONCEPTS

• **Software and its engineering** → *Designing software*; • **Computing methodologies** → **Cost-sensitive learning**.

KEYWORDS

automated data generation, ML, model scalability, model transferability, walltime estimation,

ACM Reference Format:
Swathi Vallabhajoyula and Rajiv Ramnath. 2023. Towards Characterizing DNNs to Estimate Training Time using HARP (HPC Application Resource (runtime) Predictor. In *Practice and Experience in Advanced Research Computing (PEARC '23), July 23–27, 2023, Portland, OR, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3569951.3597607

1 INTRODUCTION

Deep neural network models are prevalent in every field, and training domain-specific models needs an extensive understanding of several NN models, datasets, and the desired output. Developers try several variations of the model or similar models till it reaches a sure accuracy while trying to train them faster to see the results.

*Both authors contributed equally to this research.

PEARC '23, July 23–27, 2023, Portland, OR, USA

ACM ISBN 978-1-4503-9985-2/23/07.
https://doi.org/10.1145/3569951.3597607

This hyper-parameter tuning is an exhaustive process. Users often allocate the default allocation requests and only finetune them if the job fails or the resource is expensive. We observed a few project spaces in Ohio Supercomputer (OSC) that are used to train custom DNN models in different domains (Figure 1a). By studying the utilization logs, we concluded that by course-tuning the no. of CPU core requests while requesting one GPU core, the project spaces could save up to 40% of the current usage.

2 HIGH-PERFORMANCE DEEP LEARNING AND CHALLENGES TOWARDS ALLOCATIONS

The time and memory requirements for training conventional DNN models depend on a diverse set of features listed below (but not just limited to):

- Training Data: the size of training data, where it is preprocessed (CPU or GPU), how it is loaded into GPU (if GPU training), its sample dimension (if Image) or length of the sequence (min, max with padding, of word-wrapped), and output size.
- Application Hyper-parameters: The batch size, no. of epochs, learning rate, early stopping, optimizer, type of layers, no of layers types of layers (dense, convolutional, recurrent, dropout, normalization, to name a few), and no of trainable parameters,
- The execution endpoint characteristics: type of node (CPU. GPU/TPU), processor type and speed, available Memory and swap Memory, cores per node, and inter-intra-io thread configurations (or no of workers).

Assigning more cores does not necessarily guarantee faster executions (especially for GPU training). CPU is needed to load the dataset while the GPU trains the model. CPU memory should be large enough to hold the dataset, while GPU memory should be large enough to hold the model parameters. The allocations are charged per core hours, and if this job runs for 10 hours, it would cost $1.6 (10*28*$0.003(CPU)+10*$0.09(GPU)) as per the service costs for academic purposes on OSC if the user could tune the allocations as per the memory requirement (memory requirement almost = Memory-per-core*no. of cores), the allocation for this job could be reduced to $0.95 by safely allocating the cores to 5.

A typical procedure to manually estimate the resources has the following steps:

(1) Training the model for a short time with some or complete training data

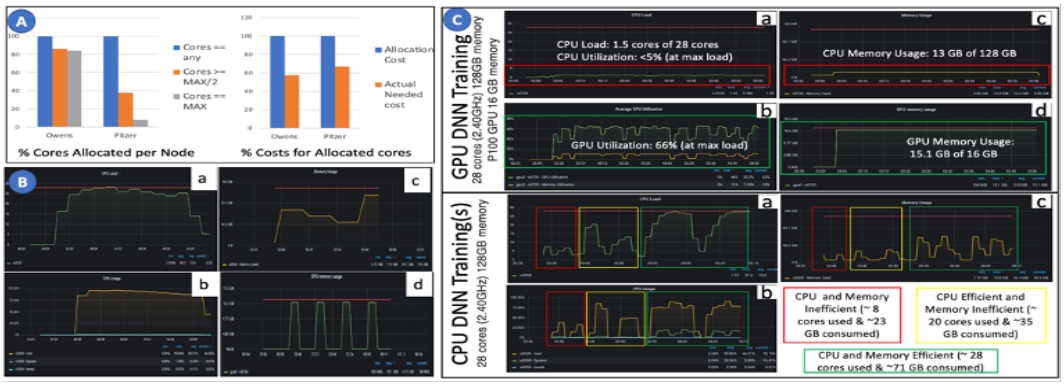

Figure 1: (a) Stats from different user groups and their allocation requests (1 node-GPU) (b) Near ideal resource utilizations (from profile logs of different jobs from grafana). (c)(Top) The resource utilization of one GPU allocation on Owens for training a fully connected DNN model. (c) (Bottom) Showing the influence of different hyperparameters on resource utilizations for the same model.

(2) Observe the utilization traces from Grafana[1] by using any profiling or command line tools like Tau and note the CPU/GPU utilization and memory utilization.
(3) Repeat steps 2 and 3 by changing the parameters (batch size, input size) and observing the differences.
(4) Creates a simple equation to calculate the needs by considering the changes and using it to repeat steps 1, 2, and 3).

The users who extensively use CIs to train and hyperparameter tune large models use a similar strategy and often follow the customized (back of the envelop calculations). However, with the rate at which DNN models evolve, the user upgrades them to build comparable models. Moreover, hence have to re-do the calculations every time the model changes, which is unresourceful.

3 CURRENT RESEARCH DIRECTION

Having a model like HARP[2] (HPC Application Resource (runtime) Predictor) enables configuring the application (a DNN training code) with the framework along with the list of known potential features (mentioned above). It generates the scaled-down (SD) training data (like in step 1 of manual estimations) and runs the full-scale (FS) application (like step 3) with minimal configurations. It used the SD executions to learn about the application's depends on hardware against different configurations and used FS executions to scale the predictions to actual runs allocation. The feasibility of such a framework and the formulation is presented in papers [1] and [2].

We released version 1.0 of HARP software as part of the ICICLE[3] release as an AI4CI solution. The code is available to download and install on Linux-based systems. It is tested on all clusters at OSC.

Figure 2 shows the potential feature space and their overlaps and the different components of the proposed HARP framework. A human in a loop decides the possible configurations to profile the application against. Then the framework automatically executed the three phases – "generate" to generate the scaled-down and "full-scale runs," and "build" to pipeline the generated data through a standard machine learning pipeline. An optimal predictive model is selected per application based on the policy metrics and time-cot limitation. We compare our results to the default settings – allocating a large walltime with all resources on the node.

Table 1 shows the results for estimating the executing time for a target application (training VGG16 model on Wiki Images for estimating a person's age). For prediction Model VGG16, we run HARP from end-to-end, emulate the scaled-down(SD) and full-scale (FS) runs, train models on emulated data, and build the estimators. For the prediction Model Fully-DNN, we run HARP from end-to-end, emulate only full-scale (FS) runs, train models on this newly emulated data and already existing data that was previously generated by emulating a fully connected DNN model, and build the estimators on this augmented dataset. By using an existing dataset, we were able to learn the execution behaviors from existing similar datasets and reduce the cost of data generation (from $5.15 to $1.35). The availability of many training emulations for similar models helps gain better prediction accuracies (MAE, MAPE, UPP) without generating many additional application-specific emulations. We draw a default baseline by calculating the cost of executing the jobs with average walltime as estimated walltime. Every time a job fails due to under allocation of time, we double the previously estimated time and rerun the job; this cycle is repeated till the job runs complete, and the cost of execution is the cumulative cost of all runs per job. The actual execution cost is estimated based on the actual allocation of CPUs and GPUs per job. The Precited estimation chooses the low-cost execution allocation (CPU cores and GPU) and estimates the cost of executing the application against that configuration. By fine-tuning the CPU cores (especially GPU jobs), we reduce the execution costs, even with a predictive model, by 40% (as observed initially in Introduction).

4 FUTURE DIRECTION

The current data and the models are configured to estimate resource needs for single-node training. We must study what features influence distributed training (no. of nodes, type of destitution, bandwidths). For the single node application, we captured around 40 features we perceived to influence the execution time (Applying principle component analysis with 0.99 variances, reduced those

[1]https://grafana.osc.edu/
[2]https://github.com/ICICLE-ai/harp
[3]https://icicle.osu.edu/cyberinfrastructure/software

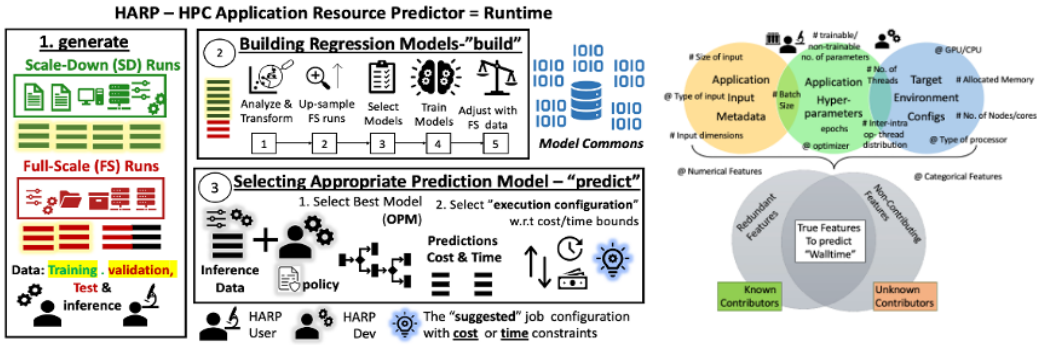

Figure 2: (Left) Visualizing the HARP framework; (Right) The distribution of features space their overlap with an example (for profiling a sample DNN model)

Prediction Model	Walltime Error (Actual, Predicted)			Training Samples	New Data Generation Cost (in $)	Cost of Execution (in $)		
	MAE	MAPE	UPP			Actual	Default	Pred.
VGG16	1043	701	11	511	5.15	**0.15**	0.31	0.23
Fully-DNN	116	56	0	775	1.35			**0.09**

Table 1: MAE: Mean Absolute Error, MAPE: Mean Absolute Percentage Error, UPP: Under Prediction Error. Table showing the estimation errors and the actual, default, and predicted execution costs for estimating execution time for training a VGG16 model on Wiki Image Dataset for age classification. Two different models were used to make the predictions - 1. model trained on VGG16 emulations, 2, Model training of an existing emulation for training a fully connected DNN and a small sample of VGG16 emulations

to 10). Including distributed training will increase the feature dimensionality. To do such feature engineering efficiently, we want to perform the ablation analysis on the current single node data along the three directions – input (without changing hardware or application features) and changing hardware keeping the other two constants, and training fine-tuning model parameters on a given node and input.

Funding and Collaboration: *This work was partially supported by the National Science Foundation and the NSF AI Institute for Intelligent Cyberinfrastructure with Computational Learning in the Environment(ICICLE) under grant agreements OAC-1945347 and OAC-2112606.The ideation behind the HARP development - data, AI and automation-based challenges are presented, and HARP's solutions ia accepted as a short paper Insights from the HARP Framework: Using an AI-Driven Approach for Efficient Resource Allocation in HPC Scientific Workflows to PEARC'23*

REFERENCES

[1] Manikya Swathi Vallabhajosyula and Rajiv Ramnath. 2022. Towards Practical, Generalizable Machine-Learning Training Pipelines to Build Regression Models for Predicting Application Resource Needs on HPC Systems. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 43, 5 pages. https://doi.org/10.1145/3491418.3535172

[2] Swathi Vallabhajosyula and Rajiv Ramnath. 2022. Establishing a Generalizable Framework for Generating Cost-Aware Training Data and Building Unique Context-Aware Walltime Prediction Regression Models. In *2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. 497–506. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom57177.2022.00070