Machine Learning Job Length Prediction

Michael David
EECS 4080
York University
Project on GitHub.

August 4, 2025

1 Introduction

This report explores the task of predicting machine learning job length. This is important for implementation in reinforcement-learning-based cluster schedulers to make better decisions. Previously, this has been done only for job speed, not duration itself [4]. Using training and inference benchmarks with NVIDIA hardware from MLCommons (v4.0, v4.1, and v5.0), we construct a dataset of 207 entries and train different prediction models from traditional regression methods to advanced neural networks on key features like model architecture, dataset characteristics, and system specifications. By analyzing performance across various configurations and job types, we aim to identify the most effective model for accurate job prediction in large-scale machine learning workloads.

2 Related Work

Related work on ML job length prediction has focused mainly on training jobs. Marzi et al. evaluate job speed using the Full Parameter Time Complexity (FPTC) framework, which models training time as a function of dataset and model parameters. While effective in some cases where all the hyperparameters are easily available, they show it is highly context-dependent and not easily generalizable, especially for more complex models [4]. In contrast, Gu et al. analyze GPU cluster management for distributed deep learning and propose Tiresias, their own GPU cluster scheduler, to addresses unpredictable completion times through two-dimensional prioritization and placement strategies [3]. My project extends these directions by predicting total job duration (in minutes), both for training and inference jobs, using single job characteristics like model, dataset, and hardware.

3 Dataset Construction

The construction of the dataset involved some careful parsing from the result repositories of the MLCommons benchmarks under NVIDIA hardware, which in the end resulted in 207 rows from what was available. [2], [1]. From these 207 entries, 110 are training jobs and 97 are inference jobs, with 14 corresponding unique models and datasets covered by the benchmarks. For the training results, each entry can be built from three main sets of information: the model and dataset coming from the folder name in the results and the benchmark info site, the system specification file that reports all of the hardware configurations and available resources, and several JSON result files over multiple trials that feature the batch size and time where the time can be aggregated as an average and converted to minutes. There were also some result files that mentioned epoch count, but sometimes it was not present at all or wildly large, deeming it unreliable to use.

For the inference results, specifically looking at the offline setting which is available for all records, each entry can be built from similar sets of information: the model and dataset come from folder names and the site itself, the system specification file is the same as for training jobs and provides the same information, and now there is a pre-made summary of several results where the mean time per sample can be extracted, scaled, and converted to minutes. The scaling for the offline setting of inference jobs comes from the fact that there are minimum 24,576 total samples for the single query that is evaluated and this factor needs to be included to find total time.

Additionally, there are derived attributes that were included after and mapped to the entries, ranking the speed of GPUs, size of dataset, and model complexity to help reduce error, which were compiled by checking all distinct names of GPUs, dataset, and models, asking an LLM to produce the ranking from these and then correcting the results from a few quick searches if there were inaccuracies.

Note that the dataset name in conjunction with a size rank are used instead of a specific size in gigabytes since these are all standard datasets and there is a relationship that can be made from the name itself to the target job length. Furthermore, the exact sizes of these are not easily available online or in original

papers. What is more commonly found is number of images or examples and this was used to revise the dataset size rank instead.

4 Input Features

The input attributes we consider for the this regression task are listed below. We are focusing on resources and specifications of singular jobs on a server or cluster. Each numerical attribute is scaled using the min-max scaling technique and each categorical attribute is one-hot encoded, with the target job length time in minutes logarithmically scaled down due to variance. This can be undone to recover the original time after.

- model_name: name of the ML model
- data_name: name of the dataset
- type: type of job (train or inference)
- gpu_name: name of GPU(s)
- gpu_count: number of GPU(s)
- cpu_count: number of CPU(s)
- cpu_core_count: number of CPU cores
- cpu_mem_gb: memory capacity for CPUs in GB (gigabytes)
- gpu_mem_gb: memory capacity for GPUs in GB (gigabytes)
- batch_size: number of samples processed before updating model parameters
- gpu_speed_rank: rank for GPU speed (1 is the fastest and greater rank means slower GPU)
- data_size_rank: rank for dataset size (1 is the smallest and greater rank means larger size)
- model_complexity: rank for complexity of the ML model (1 is the simplest and greater rank means more complex)

5 Model Architecture

The models considered for this regression task are Random Forest regression trees, XGBoost regression trees, and MLP neural network. These are passed through a pipelines of Grid Search over a selection of hyperparameters to find the best combination with the lowest error, and then each optimal model is trained and evaluated using 10-fold cross-validation.

Grid Search Parameter List

- Random Forest
  - max_depth (maximum depth of trees): [3, 5, 7, 10]
  - min_samples_split (minimum number of samples to split a node): [2, 3, 4]
- XGBoost
  - max_depth (maximum depth of trees): [3, 5, 7, 10]
  - learning_rate (contribution of each tree to overall model): [0.01, 0.05, 0.1]
  - min_samples_split (minimum number of samples to split a node): [2, 3, 4]
- Neural Network (using adaptive learning rate and ReLU activation)

- hidden_layer_sizes (number of neurons at each hidden layer): [(128, 64), (32, 128, 64), (64, 128, 32), (32, 64, 128, 64), (32, 64, 128, 32), (32, 64, 128, 256, 64)]
- alpha (regularization contribution): [0.0001, 0.001, 0.01, 0.1]
- learning_rate_init (initial learning rate): [0.001, 0.01, 0.1]

6 Challenges and Improvements

Originally, this regression task was tackled with a smaller benchmark, DAWNBench, which was the precursor to MLCommons, covering only 3 datasets and much older models. The quality of the raw data was not robust and parsing info was difficult with less strict submission guidelines for entries, leading to very high errors. To find where problems in the data were arising, each model was trained on each job type separately, revealing that training jobs were contributing to the high errors in particular.

This led to include more preprocessing and scaling of several features related to training jobs and the target, as well as adding in the derived attributes helping get a better sense of power with specific model and hardware combinations, and then switching over to MLCommons. These changes had resulted in much lower errors. There were still some issues with inference jobs, though after fixing some malformed entries, getting rid of duplicates, and trying more varied parameter combinations, these were also able to get to an acceptable R^2 value above 0.6. It was important to make sure each individual subset of data had enough predictive power first, and then they could be recombined.

7 Model Evaluation

Each regression model is passed through a pipeline of Grid Search hyperparameter tuning and 10-fold cross-validation. Grid Search score is given by root-mean-squared error and cross-validation scores are given by each of R^2 correlation, root-mean-squared error, and mean-absolute error.

Grid Search Results

- Random Forest
  - Best parameters: max_depth: 10, min_samples_split: 2
  - Best score: 0.61
- XGBoost
  - Best parameters: learning_rate: 0.1, max_depth: 3, min_samples_split: 4
  - Best score: 0.53
- Neural Network
  - Best parameters: alpha: 0.1, hidden_layer_sizes: (64, 128, 32), learning_rate_init: 0.01
  - Best score: 0.6

Cross-Validation Mean Scores

Model	R^2	RMSE	MAE
Random Forest	0.76	0.57	0.41
XGBoost	0.81	0.52	0.39
Neural Network	0.63	0.69	0.5

8 Conclusion and Future Work

The best performing regression model for job length prediction overall is XGBoost, likely from its ability to handle nonlinear feature interactions effectively and enhanced predictive power using gradient boosting. There is still future work that can be done such as further improving the predictive accuracy for inference jobs, expanding the dataset to include a more diverse range of GPUs like Intel or AMD, and ultimately integrate the prediction model directly into an RL-based scheduler. As server hardware and clusters keep getting more advanced, this is a crucial piece of support to ensure efficient running of jobs as well as lower power consumption from better scheduling decisions that lead to less bottlenecks.

References

[1] Mlperf inference benchmark. `https://mlcommons.org/benchmarks/inference/`, 2025. Accessed: 2025-08-17.

[2] Mlperf training benchmark. `https://mlcommons.org/benchmarks/training/`, 2025. Accessed: 2025-08-17.

[3] Jinliang Gu, Mosharaf Chowdhury, Kang G. Shin, Yibo Zhu, Myeongjae Jeon, Jing Qian, Haoyu Liu, and Chuanxiong Guo. Tiresias: A gpu cluster manager for distributed deep learning. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 485–500, 2019.

[4] Francesco Marzi, Gabriele d'Aloisio, Alessio Di Marco, and Giovanni Stilo. Towards a prediction of machine learning training time to support continuous learning systems development. *arXiv preprint arXiv:2309.11226*, September 2023.