Mitigation of Scaling Trade-offs in Distributed Deep Learning through Multi-Objective Optimization

Elvis Rojas[1] and Esteban Meneses[2,3]

[1] National University of Costa Rica, San José, Costa Rica
[2] Costa Rica Institute of Technology, Cartago, Costa Rica
[3] National High Technology Center, San José, Costa Rica
erojas@una.ac.cr, emeneses@cenat.ac.cr

Abstract. The potential to solve complex problems along with the performance that deep learning offers has made it gain popularity in the scientific community. Increased performance through scaling creates a challenge related to the trade-off between accuracy and performance. It is mandatory to optimize a set of hyperparameters. In this work, the Multi-Objective Optimization method is presented to find the optimal values of the hyperparameters in a formal way. The expected results is a minimization of the trade-offs.

Keywords: Distributed deep learning · Multi-Objective Optimization · Hyperparameters · Scalability

1 Introduction

Throughout time, high performance computing (HPC) applications have been solving complex problems that require large amounts of hardware resources. However, to take advantage of those resources, it is necessary to scale efficiently. Therefore, it is necessary to solve the challenges that scaling implies. A special type of application that has benefited from the distributed and parallel architecture of HPC systems is deep learning (DL). These applications have evolved creating innovative approaches in many domains of science and engineering [4]. DL frameworks that support the training of neural networks (NN) have evolved to use new acceleration hardware (GPU, TPU) and implement parallel mechanisms for the execution of distributed training (DT). On the other hand, despite improvements in hardware and software to increase performance, it is necessary to take into account other variables that affect DL training. In particular, we must consider the hyperparameters which must vary and be adjusted according to the type of problem. Furthermore, this leads to the generation of trade-offs between performance and accuracy when scaling DL applications. Clearly, the configuration of hyperparameters is not trivial and is key to the correct performance of a distributed DL application. Therefore, to contribute to the study of this problem, we propose the implementation of a hyperparameter optimization approach based on the Multi-Objective Optimization (MOO) method.

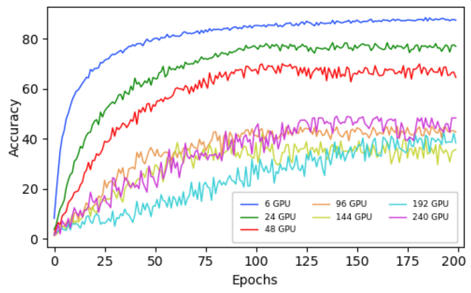

Fig. 1: Accuracy of training scaling up to 240 GPUs with DDP on ResNet50 [8].

2 Hyperparameter optimization with Multi-Objective Optimization

In a previous study [8] several distributed training mechanisms (Distributed Data Parallel, Horovod, DeepSpeed and FairScale) that are used in PyTorch [7] were analyzed to allow distributed training to increase performance when scaling on GPUs. These mechanisms have different approaches to perform parallelization and become an additional element that influences the tradeoffs generated when scaling DL applications. Figure 1 shows the trade off between accuracy and performance when scaling GPUs (240) of several distributed trainings. It is notable that the increase in performance generates a loss of accuracy which can be mitigated with adjustments in the learning rate. However, this may not work, and cause divergence of other model parameters [5]. Additionally, large batch sizes not optimized for scaling can cause tradeoffs [2,3]. Also, sophisticated optimization and autotuning techniques can be used, which vary depending on the configuration of elements involved in the training.

The learning rate and batch size mentioned above are two important hyperparameters, but they are not the only ones. Taking PyTorch as a reference, there are other hyperparameters such as `momentum`, `weight decay`, `betas` that are directly related to the optimizer (ADAM, SGD). In addition, when implementing other optimization mechanisms such as the automatic adjustment of the learning rate (`LambdaLR`, `MultiplicativeLR`, `StepLR`, `MultiStepLR`, `ExponentialLR`, and `ReduceLROnPlateau`) new hyperparameters are added. One example is the `patience` hyperparameter that is part of the `ReduceLROnPlateau` learning rate scheduler. There are many more hyperparameters, which can have a different impact on the behavior of training. The importance of an adequate configuration of the hyperparameters is clear, however, it is also necessary to take into account other elements that add variability in the training, such as the number of GPUs or the type of dataset.

2.1 Multi-Objective Optimization (MOO)

It is an area of multiple-criteria decision-making which is related to the mathematical solution of problems involving more than one objective to be optimized simultaneously. MOO has been applied in many areas of science where decisions are required to be made in the presence of a trade-off between two or more objectives [1]. The general approach is to determine an entire Pareto optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that is not dominant with respect to another. However, identifying a complete Pareto optimal set for many multi-objective problems is practically impossible. So the practical approach is always to investigate a set of solutions that represents the Pareto optimal set as well as possible [6].

2.2 Towards hyperparameter optimization with MOO

DL applications present variable behavior depending on the configuration of their hyperparameters, which are directly affected by other external elements related to the execution environment or the data type. In this context, the K hyperparameters become the K objectives of a multi-objective decision problem. Then, given an n-dimensional array $x = \{x_1, ..., x_n\}$ in a solution space X, an array x^* must be found that minimizes to a set of K objective functions $z(x^*) = \{z_1(x^*), ..., z_k(x^*)\}$. Furthermore, the solution space X will be restricted by a series of constraints $g_j(x^*) = b_j$, for example the number of GPUs.

This leads us to conclude that there is a good opportunity to apply a formal method to determine the most optimal values for the hyperparameters of a DL application. The problem in question not only requires taking into account the most important hyperparameters for optimization, but also the entire context in which the DL application is framed. Analyzing the context of the application, other elements that affect it could be extracted. Of course, these may not be modifiable, but they can be taken into account to generate the set of solutions.

References

1. Chang, K.H.: Chapter 5 - multiobjective optimization and advanced topics. In: Chang, K.H. (ed.) Design Theory and Methods Using CAD/CAE, pp. 325–406. Academic Press, Boston (2015). https://doi.org/10.1016/B978-0-12-398512-5.00005-0
2. Goyal, P., et. al.: Accurate, large minibatch sgd: Training imagenet in 1 hour (2018)
3. Gupta, S., Zhang, W., Wang, F.: Model accuracy and runtime tradeoff in distributed deep learning:a systematic study (2016)
4. Hey, T.: Opportunities and challenges from artificial intelligence and machine learning for the advancement of science, technology, and the office of science missions (9 2020). https://doi.org/10.2172/1734848, https://www.osti.gov/biblio/1734848
5. Koliousis, A., et. al.: Crossbow: Scaling deep learning with small batch sizes on multi-gpu servers. Proc. VLDB Endow. **12**(11), 1399–1412 (Jul 2019)
6. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering System Safety **91**(9), 992–1007 (2006). https://doi.org/10.1016/j.ress.2005.11.018
7. Paszke, A., et. al.: Pytorch: An imperative style, high-performance deep learning library (2019)
8. Rojas, E., Quirós-Corella, F., Jones, T., Meneses, E.: Large-scale distributed deep learning: A study of mechanisms and trade-offs with pytorch. In: High Performance Computing. pp. 177–192. Springer International Publishing, Cham (2022)