Best Practices on Deploying Scalable and Effective Deep Learning on HPC

Nikos Nikoloutsakos[1] Christos Tryfonopoulos[1]

[1]Department of Informatics & Telecommunications
University of the Peloponnese, Greece
[dsc18015, trifon]@uop.gr

In recent years Deep Learning (DL) has achieved great success in many fields including computer vision, speech recognition and natural language processing and is able to produce accurate solutions for problems that were previously thought to be difficult to solve. Image classification is a prime example of this, where DL methods such as Convolutional Neural Networks (CNNs) improve prediction performance, by observing a vast amount of training data. These breakthroughs in DL and computer vision in the fields of recognition, classification and detection provide many application possibilities ranging from smart cities to and machine optimized logistics and unsupervised supply chains. Over time DL networks get deeper and more computationally intensive, while at the same time the volume of training data available is increasing rapidly; as a result the training time may become prohibitively long. One way to speed up the training process is to use hardware accelerators such as GPUs or TPUs, while another one is to utilize multiple machines, each equipped with multiple hardware accelerators, to train a given model.

This work aims to explore ways to distribute DL in High Performance Computing (HPC) systems and compare the tools used for deploying models at scale. In this submission, initially we describe the problem from a theoretical perspective by reviewing approaches used in distributed training and perform an empirical analysis for training a CNN by using a popular benchmark dataset (i.e., ImageNet) in a real HPC system. Specifically, we focus on the data parallelism approach, which distributes the data across different nodes and operates on them in parallel, and train a CNN model using different widely used open-source tools/frameworks such as Tensorflow, PyTorch and Horovood. Our goal for this task is threefold: (i) to emphasize efficient resource utilization, (ii) to reduce training times, and (iii) to avoid the quality degradation of the resulting models when scaling out.

In our experimental analysis, we evaluate the performance of the distributed models both in terms of the predictive accuracy of the model and in the computational speed of the process. Moreover we examine the quality implications on

the model itself by assessing different ways of scaling out (i.e., naïve vs linear) and different learning rates.

Our findings allow us to provide guidelines that allow less experienced researchers to select the most suitable framework for each application and scale out their deep learning tasks in HPC infrastructure while avoiding common issues and programming pitfalls that may lead to increased training times or degrading model performance. To further assist the scientific community, we provide code and scripting examples for many widely-used frameworks as a best practice guide. The provided examples show how to efficiently scale out class image classification problems to multiple GPU workers without degrading the resulting model quality; in one of the ready-to-use examples, researchers are shown how to reduce training time from 9h to under 1h in a few easy steps when scaling out from a single to a 16 GPU setup under the GRNET national HPC facility, called ARIS.