LITERATURE REVIEW: Evaluating Distributed methods for training Generative Adversarial Networks

Omid Davoudi
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
omiddavodui@cmail.carleton.ca

September 27, 2019

1 Introduction

Deep Learning has opened many possibilities in the world of machine learning. This comes at the cost of requiring extremely large datasets and high computational power. As datasets grow in size, it is getting harder and harder to fit them in the memory of a single machine. The slowdown in hardware development can also be a major concern for the future of machine learning.

One of the most realistic ways to combat those problems is to distribute the training of Deep Learning models among a number of distinct computers. The massively parallel nature of artificial neural networks, which form the backbone of deep learning, helps in this regard.

Generative Adversarial Networks (GANs)[1] are a deep learning model that learns the distribution of the input data and outputs data points from that distribution. One of the domains where GANs are popular is the domain of image generation. A properly trained GAN can learn from a dataset of images and then, output new images that, while not found in the dataset, follow the same general rule. For example, a GAN properly trained on dataset containing images of cats will be able to output new cat images.

GANs are generally hard to train.[2] Their structure and competing nature means that convergence is not guaranteed. Even when they do converge, there could be many problems with the output, such as different distributions of data or in severe cases, mode collapse. The additional challenge of having to train these models in a distributed manner could further complicate the situation and possibly prevent convergence.

In this section, a number of approaches for distributing the training of neural networks are introduced. Furthermore, an approach specifically designed for distributed training of GANs is reviewed.

2 Literature Review

Related literature to this field can be classified in three major subgroups:

- Algorithms that try to distribute any machine learning algorithm, including neural networks, over a set of machines.

- Algorithms that try to distribute neural network training itself
- Algorithms and architectures that try to distribute training of a specific class of neural networks.

This section presents the publications based on the above groupings.

2.1 General Machine Learning Distribution

Previous work on methods to distribute machine learning algorithms in general mostly tries to find efficient ways for each machine to access the elements of the dataset over a network. One of the main characteristics of these approaches is that there are very few assumptions about the underlying machine learning algorithm.

As mentioned above, these approaches usually try to help the machine learning algorithm access data over a network. An approach by Li et. al[3] defines a parameter server where each node can access the parameters of the model as well as the training data without worrying about where the data is actually located. The approach groups the machines into two categories: Server nodes and workers. Server nodes each contain a portion of the model data and they communicate with each other to keep the model consistent across the whole server group. Data is replicated across servers to increase reliability. As server nodes are not required to contain all of the data at the same time, the system can be scalable even when the parameters exceed the storage capability of any single server node. There is also a server manager node which is designed to organize the servers themselves by checking for node removals, failures or additions. Worker nodes are where the actual computation takes place. worker nodes are separated into worker groups and each node can access or update the parameters of the model by issuing push and pull commands to the server group. A scheduler for each worker group assigns tasks to each worker and monitors the progress, rescheduling the tasks in case of node failure or addition.

References

[1] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems.* 2014, pp. 2672–2680.

[2] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. “Which training methods for GANs do actually converge?” In: *arXiv preprint arXiv:1801.04406* (2018).

[3] Mu Li et al. “Scaling Distributed Machine Learning with the Parameter Server”. In: *Proceedings of OSDI* (2014), pp. 583–598.