**Performance Analysis of Distributed Deep Learning**

**Abstract:**

We aim to analyse training and optimization of deep neural networks parameter over distributed framework. This will help us to understand the challenges in deep neural network model learning with large datasets.

**Introduction:**

With the ever increasing abundance of data, there has been incredible growth in the field of machine learning to help make sense of it. Deep Neural Networks have become very popular and have attracted the attention of many researchers. It is very useful and applicable in a wide range of research areas such as self-driving cars, voice recognition in mobile phones etc. Interconnected nature of the human brain inspires the core construct of the Deep Neural Network (DNN). Just by observing large amounts of data and trained properly, the expressiveness of DNNs provides accurate solutions for problems previously thought to be unsolvable. Due to large amounts of data which motivates learning methodology to consider large numbers of parameters. DNNs often have millions of parameters that require to be tuned during the training phase.

However, the large number of parameters and training data pose challenges in terms of scalability and computation time.[4] Computational intensity and memory demands of deep learning have increased significantly over the years. High performance computing clusters are the present day solution for enhancing performance in deep learning. A trade off between time and accuracy needed to be considered.

Convolutional neural network is considered to be the default choice for image classification task.[5] However this requires longer training time for better accuracy. Distributed architecture with parameter servers is considered by many researchers to reduce the training time. Synchronization and communication in distributed architecture are some of the bottle neck in learning the parameter.

Pretrained model such as ResNet-50 which is a CNN trained with imagenet dataset has initialized parameters before training with new data. This type of model reduces the training time. Distributed architecture along with pretrained models reduces the training duration for larger dataset.

In this project, we will try to overcome the disadvantages of training with large data for the convolutional neural network and ResNet-50 by utilizing the distributed deep learning methodologies such as Data parallelism and Model parallelism.[1][3] We will consider some standard datasets such as MNIST, Fashion MNIST etc. to design classification.

**Related Work:**

Other works have attempted to distribute networks like ResNet-50 or network frameworks like spark. These approaches focus on distributing via model parallelism and use batch process to improve learning time generally at the cost to accuracy. These papers generally focus on the training of one network or the creation of tools for a framework. Mikami, H. et. al. discusses how to minimize loss and maximize throughput when moving from a small parrell network to a large scale parallel network for the RestNet-50 architecture.[1][2]

Mikami, H. et. al. also proposed the application of a 2d-torus topology for sharing gradient information. This paper utilized sub 64K batch-sizes.This work aimed at improving the performance of massively distributed implementation of ResNet via a synchronous communication between GPUs. [1] SparkNet is a framework designed around Spark and leverages asynchronous communication to attempt to improve performance.[2] ResNet more generally is a Neural Network architecture that aims to lessen the impact of the vanishing gradient problem through the use of residual blocks. These residual blocks use normal layer feed forwarding common in most neural networks combined with skip connections that forward a layers input to the end of the residual block. [6] This provides ways for the network to continue to learn even if some of the layers fall victim to vanishing gradients . However, this provides an extra consideration should they be adapted to be model parallel as there will be complex layer boundaries. Park, Jay H., et al. considers the improvement of training efficiency, by optimizing how resources are sent to the networks. This resource awareness attempts to address the communication bottlenecks to decrease training time. [5] These approaches generally work on solving one aspect distributed neural network, as well as providing performance tradeoff for that one network or aspect they are tweaking. They do not attempt to cross compare distributed networks performances.

**Proposed Work:**

The training of DNNs will be carried on three kinds of processors, namely CPU, GPU and GPU in different servers. We also try to compare different Neural networks performance in each of the hardware architecture. We will also be using Data parallelism which means that each data set is divided into many subsets, where the number of subsets are equal or greater than the number of systems under each kind of processor. Here, each system will communicate with each other using message passing interface and will share their updated parameters. This update of parameters will be continued until it converges to some level. After the convergence, we will observe the computation time while training and accuracy of the model.We will use two networks to gauge performance changes each network experiences when moving from local to distributed system. This creates something resembling a neural network “shopping guide” to cross compare networks. We propose implementing model parallelism for ResNet-50 and one additional CNN.

First we will implement single core CPU computation will be performed to make it as a baseline. Then we will implement the GPU implementation followed by multi-GPU. Once the local testing baselines are set we will begin implementing the distributed approach of each network. During this stage we will collect both time and accuracy performance to get an understanding on how the networks changed when they are parralized via batches. If time allows, we will try to meet a stretch goal of implementing model parallelism for both networks.

**Project Milestones:**

* Select and preprocess datasets; select models (Week of 02/24)
* Train model locally using CPU or GPU (Week of 03/02)
* Setup distributed server and pipeline needed for distributed model (Week of 03/09)
* Test model locally using CPU or GPU (Week of 03/09)
* Train distributed model and fine tune model (Week of 03/16 and 03/23)
* Assess or start stretch goal if time allows (Week of 03/30)
* Test and evaluate distributed models' implementations (Week of 03/30)
* Start working on final report and address any backlogs (Week of 04/06)
* Work on presentation and collect more data if needed (Week of 04/13)

**Workload Distribution:**

Sushanta: Create CNN local and distributed implementations.

Brendan: Will create ResNet-50 local and distributed implementations.

Rahul: Train, test, and evaluate local exclusively CPU and local exclusively GPU models.

**References:**

1. Mikami, H., Suganuma, H., U.-Chupala, P., Tanaka, Y., & Kageyama, Y. (2018). ImageNet/ResNet-50 Training in 224 Seconds. ArXiv, abs/1811.05233.
2. Moritz, P., Nishihara, R., Stoica, I., & Jordan, M.I. (2015). SparkNet: Training Deep Networks in Spark. CoRR, abs/1511.06051.
3. Ruben Mayer and Hans-Arno Jacobsen. 2020. Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools. *ACM Comput. Surv.* 53, 1, Article 3 (February 2020), 37 pages. DOI: <https://doi.org/10.1145/3363554>
4. Ben-Nun, Tal, and Torsten Hoefler. "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis." *ACM Computing Surveys (CSUR)* 52.4 (2019): 1-43.
5. Park, Jay H., et al. "Accelerated training for cnn distributed deep learning through automatic resource-aware layer placement." *arXiv preprint arXiv:1901.05803* (2019).
6. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.