Parallelizing a Neural Network

Team Name: Parallelogram
Members:

- Yashasvi Sriram Patkuri (150050080)
- Sreekar Garlapati (150050065)
- Sai Teja Gutta (150050072)
- Prakash Reddy (150050075)

Intro

Neural networks can be used to solve problems that are rather difficult to solve using rule-based systems. A neural network typically consists of a layerwise forward pass and a backward pass. These operations can be quite parallelized. For example one of the basic neural nets is a fully connected network wherein forward pass operations are basically matrix multiplications.

What?

We plan to implement a simple fully connected neural network from scratch for the task of handwritten digit recognition. We shall use the MNIST dataset for training and testing. After reaching a satisfactory accuracy we shall try to reproduce the same results by parallelizing forward and backward passes. We plan to support the basic fully connected layers with activation. As we are trying to reproduce the results despite neural nets having intrinsic randomness in them (weights initialization) we shall have to control the randomness by controlling the random seed.

Why?

Neural networks are the basis for recent achievements in deep learning. While there is a lot of material on neural nets and its concepts. Building one from scratch and parallelizing its inner working would give a more complete perspective.

How?

We shall first profile the serial code to see which functions are taking the most amount of execution time. We shall parallelize them in that priority. We plan to use CUDA. Finally, we compare the theoretical speedup with the

practical ones and reason about discrepancies (if any). We shall also try and verify memory correctness using valgrind tool suite.