Parallelization Using Separation of Neural Networks

James Brandewie; Dr. Joseph Anderson
Department of Mathematics and Computer Science, Salisbury University

Abstract

Neural networks (NN) are one of the most critically important up and coming methods for interpreting and predicting data. However, arguably the greatest weakness of these models is the time it takes to train them. At times, it can take immense computers full weeks to train one industry-ready neural network. Parallelization is made more difficult by sequential aspects such as back propagation, epochs, and batching. This project deals with taking a new approach to parallelization techniques for neural networks in order to decrease this training time.

Previous Research

NN parallelism is largely divided into two categories: model parallelism & data parallelism. Data parallelism works by having the same NN on all sub-computers and division of the training data. Model parallelism on the other hand, involves a sort of splitting up of the NN structurally. It usually involves a pre-classification of the data, then feeding the data into the respective "model." This project involves a different method of model parallelism: splitting the neural net structurally based on weights.

Theory of Independence

The parallelization involves the following procedure: train entire NN on part of the data, split up into sub neural nets based on current weights, train each sub neural nets on remaining data, recombine at the end. The reasoning behind the anticipated success is as follows, observing the neural net below, the trend of the current training shows the left two nodes as being reasonably independent from each other.

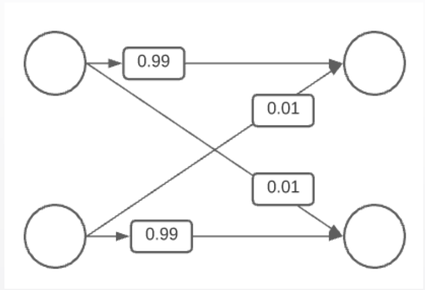

Fig.1 Independence of Nodes

Procedure

Numpy & Pytorch libraries were used for all procedures including training, separation, and recombination. SU's HPC lab was used for computationally intense data. One difficulty is that in splitting by node, you initially lose ½ of the weights if separating by node. This was absolved by alternating which group you take from for two additional NNs. An example of this is shown below.

.

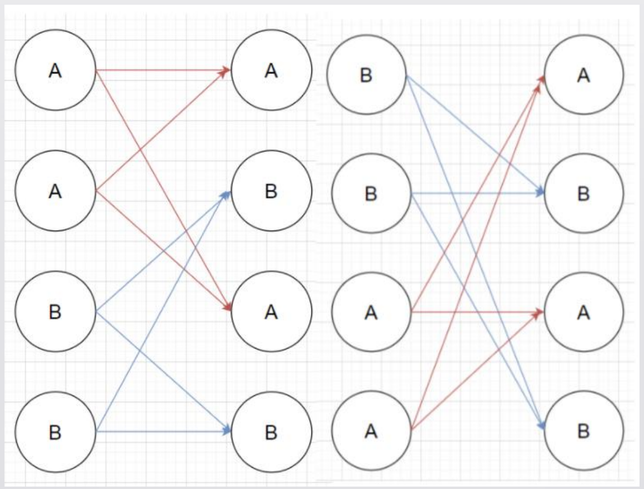

Fig. 2 Alternating Group Choice for Generating Neural Nets

Results

Splitting while prioritizing this independence never functioned as well as baseline, always decreasing in test accuracy after the recombination of split trained NNs. As such, a different approach was taken. Using the HPC lab, all combinations of splitting the neural network were iterated through. The results of this iteration can be seen below.

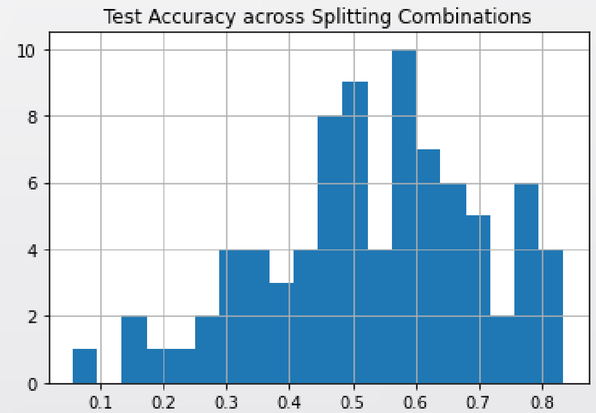

Fig. 3 Accuracy vs Combination

Conclusion and Future Work

Variation was immense in test accuracy across combinations of splitting, suggesting the structure plays an important role. However, no split resulted in significantly increased test accuracy, even for higher layer networks. This suggests that the approach of a structural split parallelization based on weights fails.

Acknowledgements

"The work is funded by NSF CCF-1757017 under Research Experiences for Undergraduates Program."

.