A Study on Parallel Machine Learning, Supervised Learning, and Reinforcement Learning
Christopher Arausa, Joe Anderson

Abstract
The goal of neural networks is typically to model some complex or unknown function. This is done by continuously reducing the error across a dataset, or sometimes by rewarding good behavior. However, this is typically a very timely process when you begin to model very large datasets or complex functions. A common solution is to parallelize the training of the data across a cluster of networks. The big issue with most parallel learning techniques is the large communication bottleneck since most algorithms communicate after every batch. In this project, I applied a naive parameter averaging approach at the end of training to see whether we can avoid communicating completely until the end.

Method
The neural networks were coded using tensorflow’s graph. One single-model trained on all of the training data was maintained, and the rest of the sub-models had their training data partitioned. The partitioning of training data was done randomly in hopes to avoid any sort of over-specialization or falling too deep into a local minima. Validation data remained the same across all models.

Results
When networks of node count 4, 10, and 32 were initialized on the same weights and biases and trained on randomly partitioned training data, the accuracy of the recombination model stayed within 2-3% of the single-model trained on all of the data.

Notice the prediction vectors are near 0 for invalid moves.

Future Works
Apply reinforcement learning to a model already trained with supervised learning to see whether it’s faster or worthwhile to teach a neural network the rules of the game first. Train a network off of solely the best moves, instead of the entire database.