Deep Learning Higgs

Shubham Gogna and Nicholas Smith

**Overview:**

This project used a neural net implementation in Torch, to distinguish between signal processes that produce a Higgs Boson and background processes that do not. The data set was produced using Monte Carlo simulations. It contains 11,000,000 samples with the last 500,000 used as the test set. It has two class labels, 0 for background, and 1 for the signal. The low level features are kinematic properties measured by the particle detectors in the accelerator. The high level features are derived by physicists to help discriminate between the two classes.

**Experiment:**

This project used a feed forward neural net to train and test the dataset. It was run three times; once for the low level, once for the high level, and once for both features. Each run consisted of 40 epochs in a 5 layer neural net (1 input, 1 output, 3 hidden layers). The input was either a 7 dimensional, 21 dimensional, or 28 dimensional vector and the output was between the two classes. The hidden layers contained 150 units each which was a balance between reasonable run times, accuracy, and options presented in the original paper. The net is trained using a batch size of 64, a negative log likelihood criterion, and a linear architecture with dropout.

**Results:**

When running just the low level, we got 59.9822% global accuracy for training and 60.3696% accuracy for testing. When running just the high level, we got 61.5236% global accuracy for training and 62.7388% accuracy for testing. When we ran both features (21 low level and 7 high level features), we got 66.0157% global accuracy for training and 65.8666% accuracy for testing.

We found that training on low and high level features together achieved the best global accuracy than training on just one or the other. This is to be expected because the neural net may establish relationships not captured in the low or the high level features alone.

We also found that we did not need to run more than 3 epochs before seeing stabilization of the accuracy. This was used to pick the number of units per hidden layer and the number of layers (the maximum tested was 300 units per layer with 3 hidden layers and 200 units with 5 hidden layers). Once we determined a good number of the units and the layers, we ran it for 40 epochs. Below is the global accuracy based on the features selected versus the number of Epochs and the results of doing a tSNE reduction on the first 5000 samples.

**Issues:**

The majority of the issues in this project were memory constraints primarily with LuaJIT. Ideally, we would have liked to shuffle all the data before running the neural net, but LuaJIT’s constraints did not allow that. To address this, we split the complete file into 8 equal sized files. In the program, we shuffle the 8 files as blocks as our way of introducing randomness. Additionally due to the computer’s memory constraints (we allocated up to 6GB RAM in the VM), we were forced to keep neural net sizes smaller than those presented in the paper. In addition to the neural net, when we attempted to run tSNE with a large number of samples, the program would crash due to the absurd amount of RAM requested (upwards of 1TB for 1/8 of the data), so we settled on the first 5000 samples (which ran using about 6GB).



