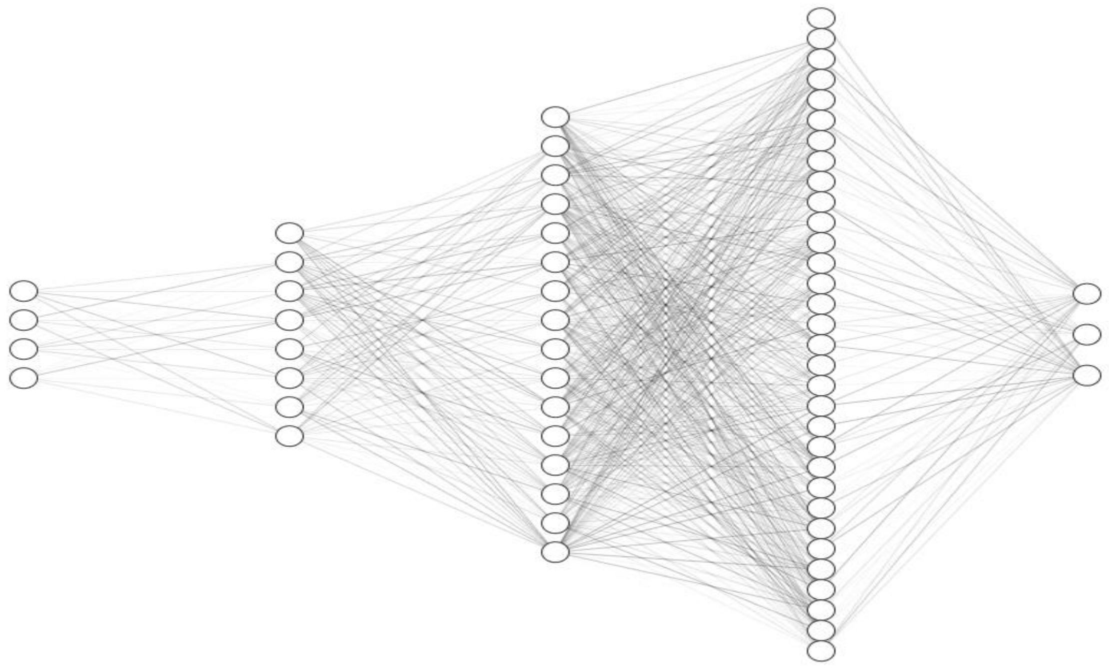
Multi-Layer Perceptron Test Score Regression w/ Google TensorFlow

Cs 461 Program 3 Writeup

Ryan Phillips

Network Architecture:



The network used for the regression task is a basic Multi-Layer Perceptron (MLP) with one hot encoded features as input. The network up samples input features steadily from 64->128->256 using fully connected layers, then, outputs 3 predictions (test scores). All weights are initialized as a gaussian distribution with a mean of 0.0 and a standard deviation of 0.2. Bias is initialized with a constant 0 across all layers. For nonlinear activations, the Rectified Linear Unit (ReLU) is used on all layers besides the output/predictions. Dropout and batch normalization are applied on every layer besides the input and output layers. During training dropout occurs with a 62% chance for each neuron in its respective layer, and batch size is set to 16. Mean Squared Error is the loss function used to compute gradients which are then back propagated using an Adaptive Moment Estimation (ADAM) optimizer. The optimizers hyperparameters consist of learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07.

Results Of Best Model:

Validation Acc: 70.0021%

Validation MAE (Mean Absolute Error): 9.7

Final Validation Loss: 148.6

Observations/Conclusions

Increasing depth and width of the network proved to increase accuracy up to a certain point. However, to see accuracy ready for a production system, only using an MLP, I believe a substantial amount of initial feature engineering would have to be done. Future work could use unsupervised or generative methods to add more beneficial features and potentially increase the size of the latent space learned. Theoretically, there could be potential in using k-means clustering to from a graph of rich feature associations that could then be passed to a graph neural network (GNN) to have message passing performed. Even using an adversarial loss could potentially widen the latent space allowing the model to learn unseen features.