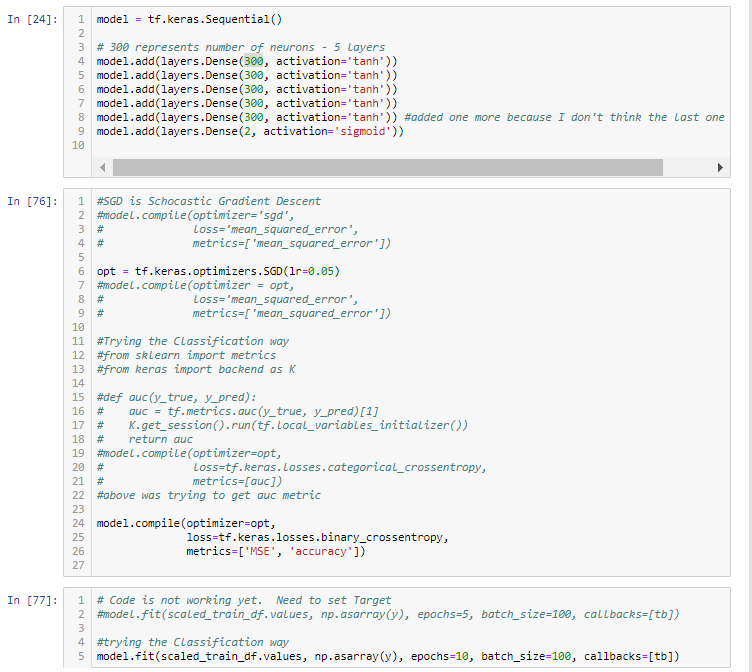
In this assignment, we attempted a few techniques to improve the MSE for our Neural Net.

1. Data: The HIGGS data set had 11 million records, the first 10.5 million training and the last 500k, testing. The first column, ‘Target,’ was used as the Y value and the remaining features the X values. The following is data augmentation approaches:
   1. Full Training Data: Using 10.5 Million records for the NN
   2. Scaled Training Data: Using 50k records for the NN.
2. Physics with Deep Learning Paper on Higgs dataset: Attempt to Replicate
   1. Neurons = 300 (ref: pg 6)
   2. Layers = 5 (ref: pg 6)
   3. activation = ’tanh’ (last layer, activation = ‘sigmoid’) (ref: pg 9)
   4. Optimizer = Stochastic Gradient Descent (ref: pg 8)
   5. Learning Rate = 0.05 (ref: 6)
   6. Loss = Binary Crossentropy (ref: lecture and documentation – based on the binary prediction)
   7. Metrics = auc (ref: pg 6) - (not able to duplicate auc, but page 14 also used MSE, so we will use that)
   8. Epochs = 10 (ref: pg 9)
   9. Batch size = 100 (ref: pg 9)

\*\*Insert Code for the three items that include the above information. (Example image below. Replace with final code)



1. Improvements to the procedure:
2. Standard practices now vs 2014:
   1. Optimizers: Several new optimizers have been created since this paper was written. rsmProp and the Adams optimizer are two such ones. In practice today, SGD is not used as often as the new ones.
   2. Activation: Since 2014, there are new activation techniques including ReLu, eLu, and GeLu.
3. Versioning the Data: Depending on the activation function used, we versioned the data:
   1. Sigmoid: Normalized between 0 to 1
   2. Tanh: Rescaled to -1 to 1
4. NN Knobs adjusted:
   1. Learning Rate: The amount that the weights are updated during training which controls how much to change the model in response to the estimated error each time the model weights update. This is a hyperparameter in NN with values between 0 and 1. This is probably the most important hyperparameter to tune. A LR that is too large can cause the model to converge too quickly and too small can cause the model to stall. We ran our NN using LR = 0.05, 0.01, and 0.001.
   2. Layers: A NN has 3 types of layers; Input, Hidden, and Output. Each hidden layer’s task is to approximate the output function. The more hidden layers are added, the longer it’ll take to learn (i.e. vanishing gradient problem). We used between 1-5 hidden layers.
   3. Activation Function: Used to determine the output of NN. The two used in this assignment is Sigmoid (0 to 1) and Tanh (-1 to 1). Sigmoid and Tanh are ideal for probability prediction or logistic prediction.
   4. Epochs: When an entire dataset is passed forward and backward through the NN once.
   5. Batch Size: The number of records passed through the NN forward and backward each pass until the whole dataset is done. For example, batch size of 500 for a dataset of 5000 means one epoch will have 10 sets of records sent through the NN.
   6. Optimizer: Algorithms that minimizes (or maximizes) a loss function using gradient values. Examples are Gradient Descent, Stochastic Gradient Descent (SGD), Mini batch Gradient Descent, Adaptive Moment Estimation (ADAM). For this assignment, we used SGD on the full training data set due to its large size and GD for the scaled training data set.

In running several NN models, we concluded that adjusting the LR, activation function, number of neurons and versioning the data is vital for function optimization. The research paper mentioned the current approach back in June of 2014 as a ‘feed-forward neural networks with a single layer and boosted-decision trees.’ They improved on that approach by using a ‘five-layer neural network with 300 hidden units in each layer, a learning rate of 0.05, and a weight decay coefficient of 1 x 10^-5.’

Here was our most optimal model returning a MSE of 0.2281:

1. Rescaled data -1 to 1
2. 3 Layers with 300 neurons
3. Learning Rate = 0.01
4. Activation Function: Tanh
5. Batch Size: 200
6. Epochs: 5

We tried several configurations by adjusting the learning rate, the number of layers, the number of neurons, scaling the data to fit the activation functions, changing the activation function, number of epochs and batch size, and using the full data set or subset. The MSE mainly staying between 0.22 to 0.28. The main trade-off was the amount of time required to complete each model configuration.