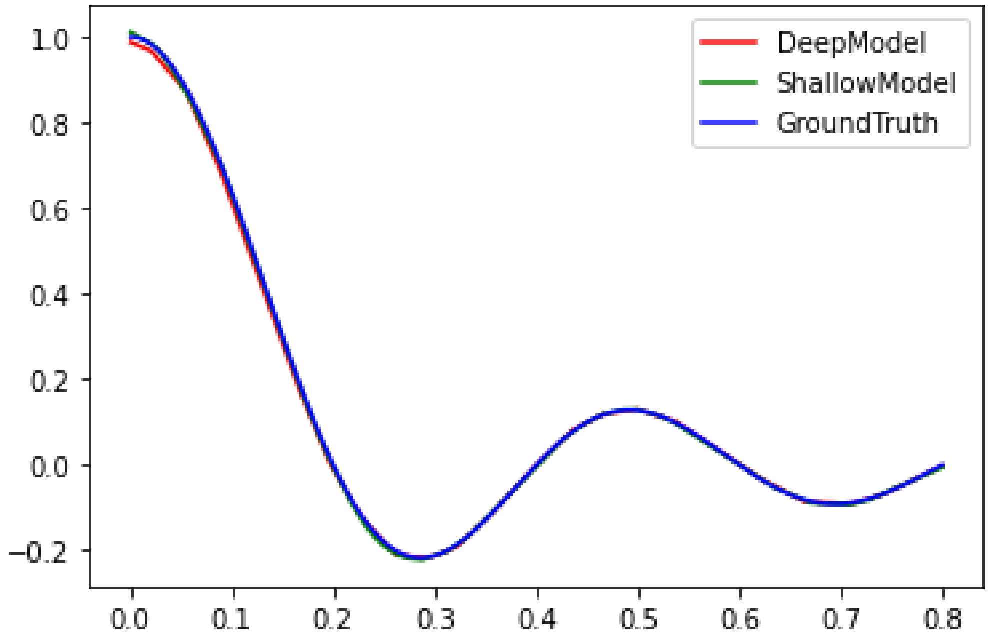
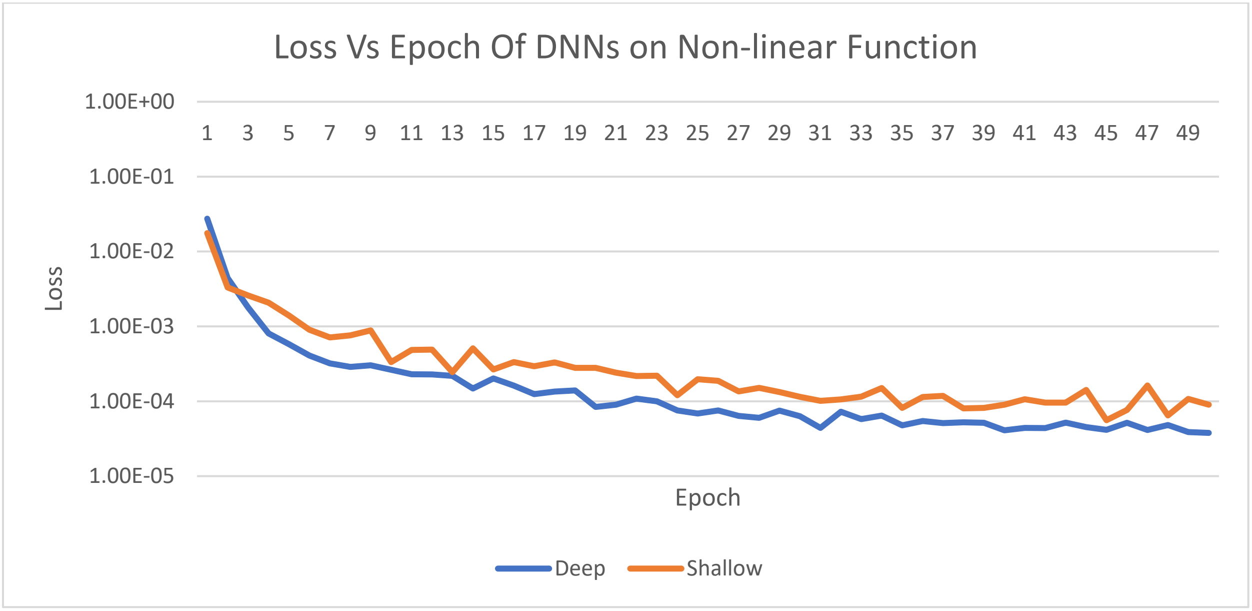
Deep Learning Homework 1 Report

HW1-1 Simulate a Function:

Both the deep and shallow neural networks had about 1400 parameters. The deep model used a combination of 5 leaky relu and linear layers. The shallow model only had 3 layers with one leaky relu and 2 linear layers.



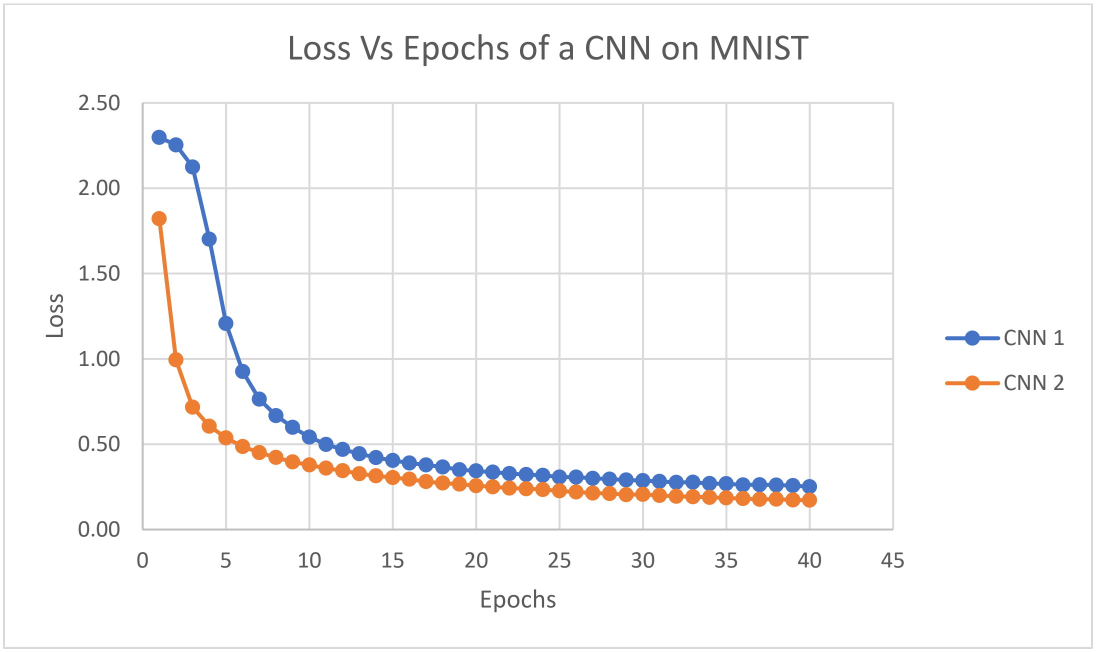


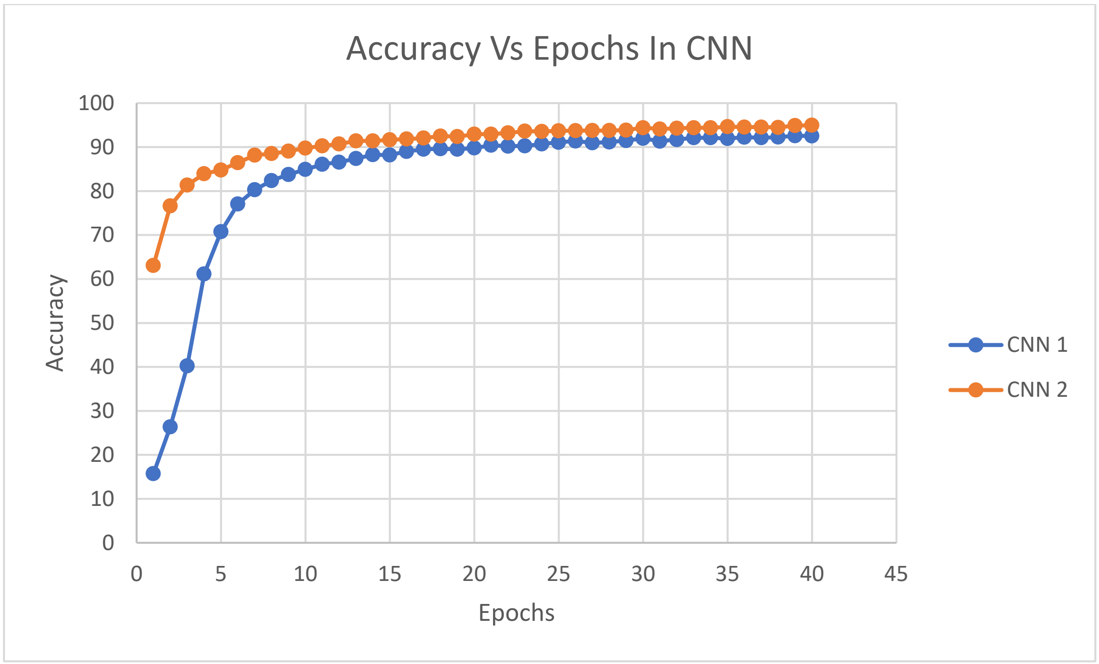
Discussion:

As seen in the graphs above there is not much difference between the model prediction and the ground truth. Also, of note is that the shallow and deep model do not differ by much in the prediction plot. This may be because of the low parameter count and small difference in layer count. The distance might improve with a more complex function and a larger layer count.

HW1-1 Train On Real Task:

To train a model on the MNIST task the CNN was chosen. The CNN included two convolutional layers with two pooling layers directly attached. Behind the convolution and pooling layers a drop out layer followed by an ReLu and another drop out layer fed into two linear layers with a softmax function to decide the prediction of the network.





Deep Learning Homework 1 Report

Discussion:

Why do the accuracy and loss look the way they do? (Mention differences between each model as reasons)

CNN 1 was implemented with more layers and parameters, however, it preformed worse compared to CNN 2 which had fewer layers, only one drop out layer, and less parameters overall. This shows that complexity is not always a good thing in neural networks and can cause slower convergence and bad performance.

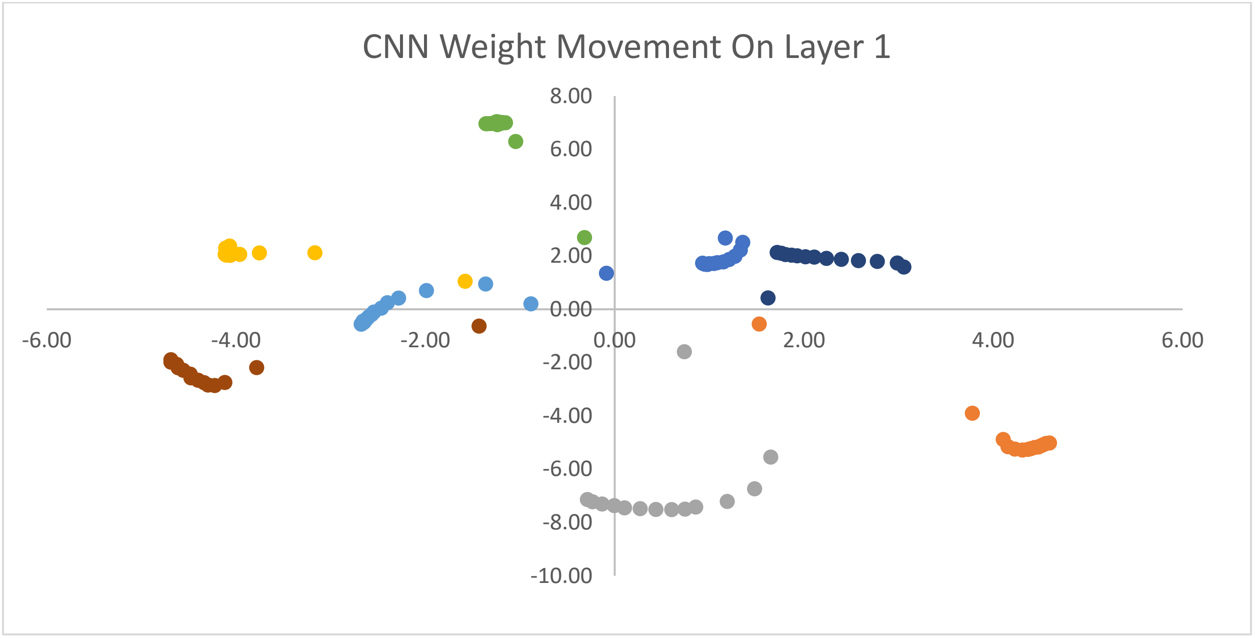
If I had more time, I would have liked to create more models with more strict variables to control the variability of each model.

HW1-2 Visualize The Optimization Process:

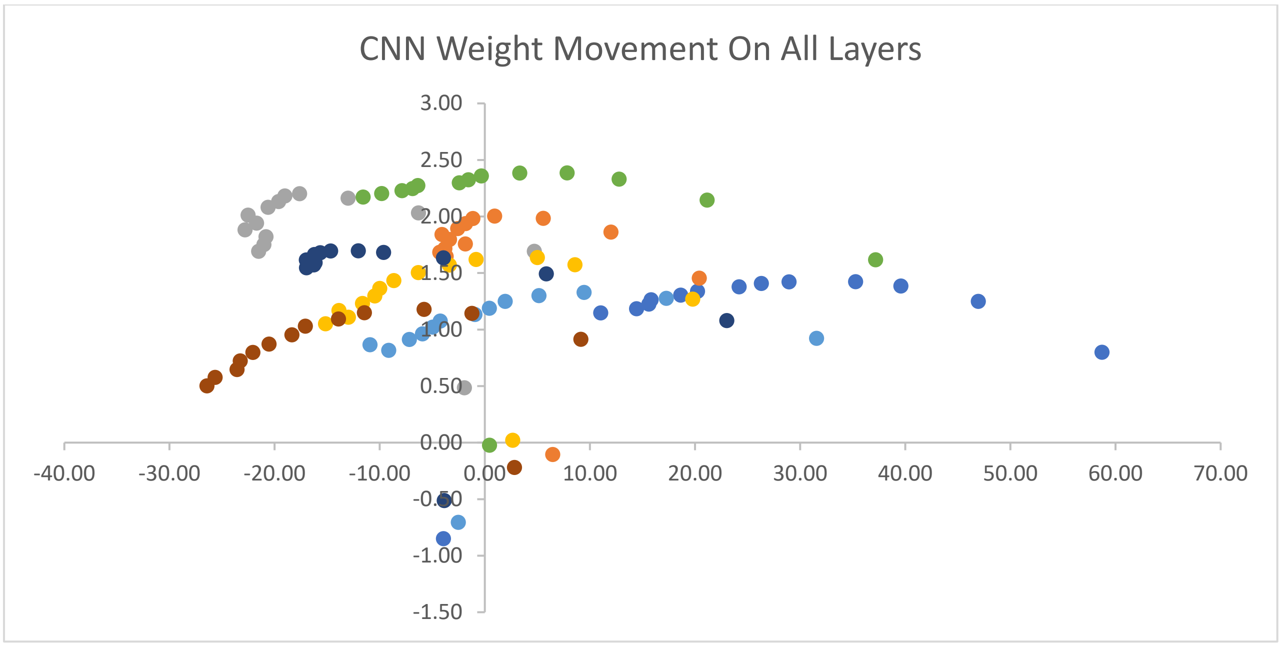
Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc)

To visualize the optimization process weights were recorded every 3 epochs and stored in an list for later processing. The model used was the exact same as the deep model from HW1-1 used to simulate a function. The Adam optimizer with the MSELoss function were used when training the model.

For reduction I could not get PCA to work so I tried my own visualization idea instead and did a simple tensor resize of each layer and (for whole model weights) stacked the weights into a 2 row array. The two rows were then summed to produce two values which became the weight's x and y value.



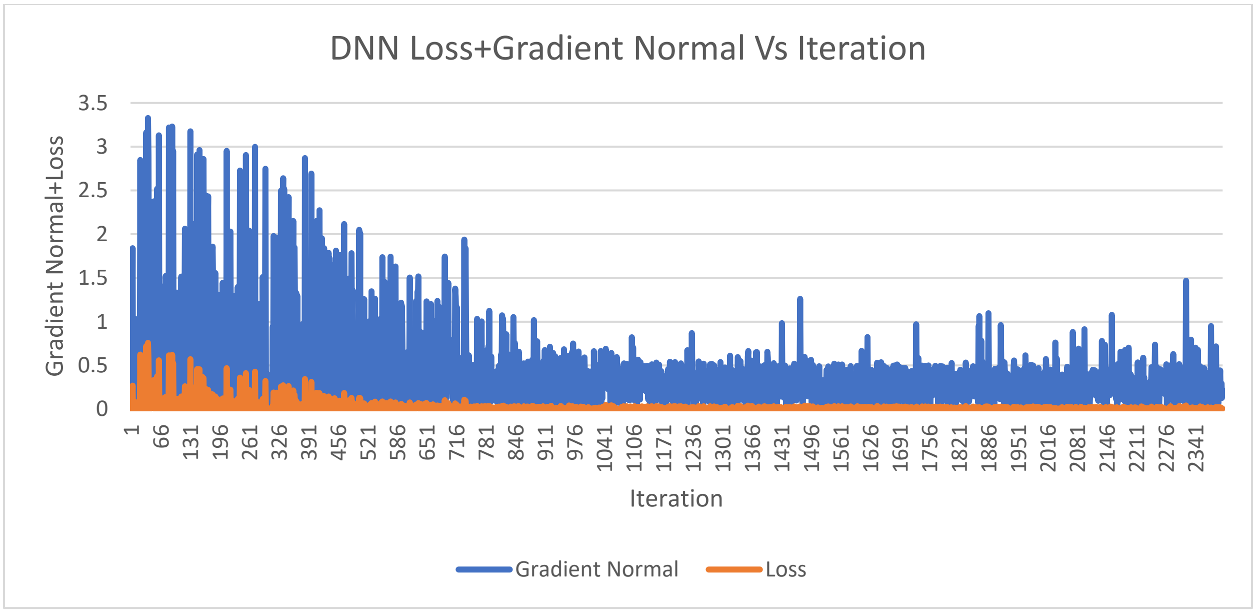
Deep Learning Homework 1 Report

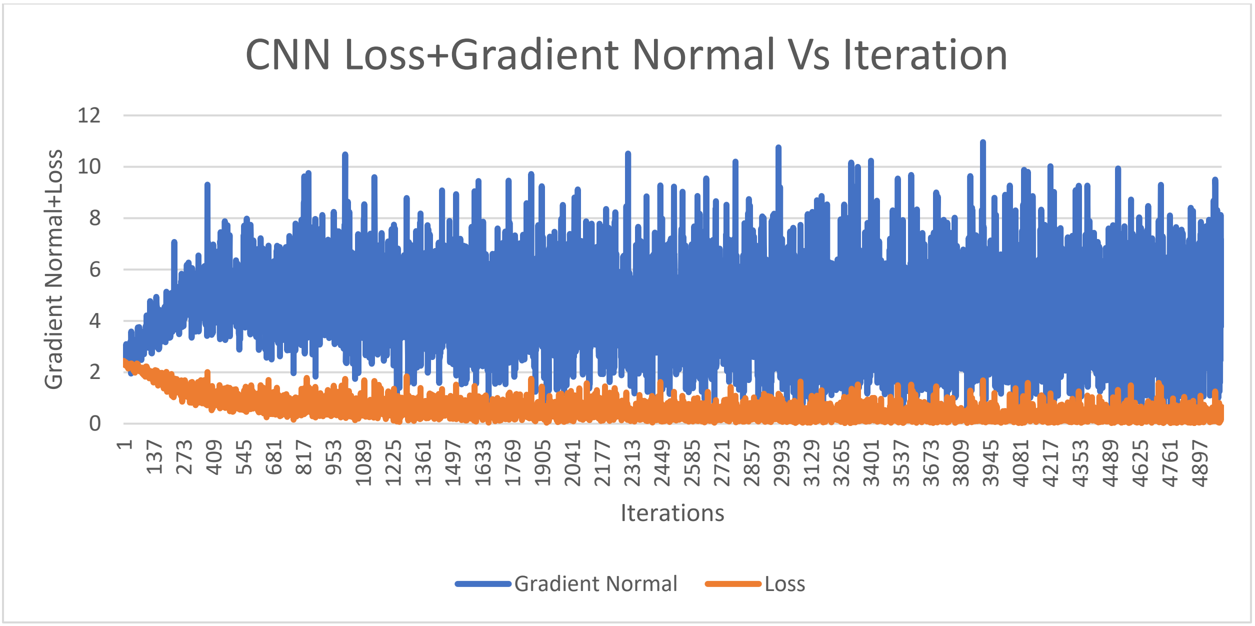


Discussion:

As shown the density and “drift” of the weights in both models, weights tend to change drastically after initialization but slow down and become centered/denser around a given optima. This follows the idea of loss and that lower values of loss will cause lower magnitudes of weight movement.

HW1-2 Observe Gradient Norm During Training:





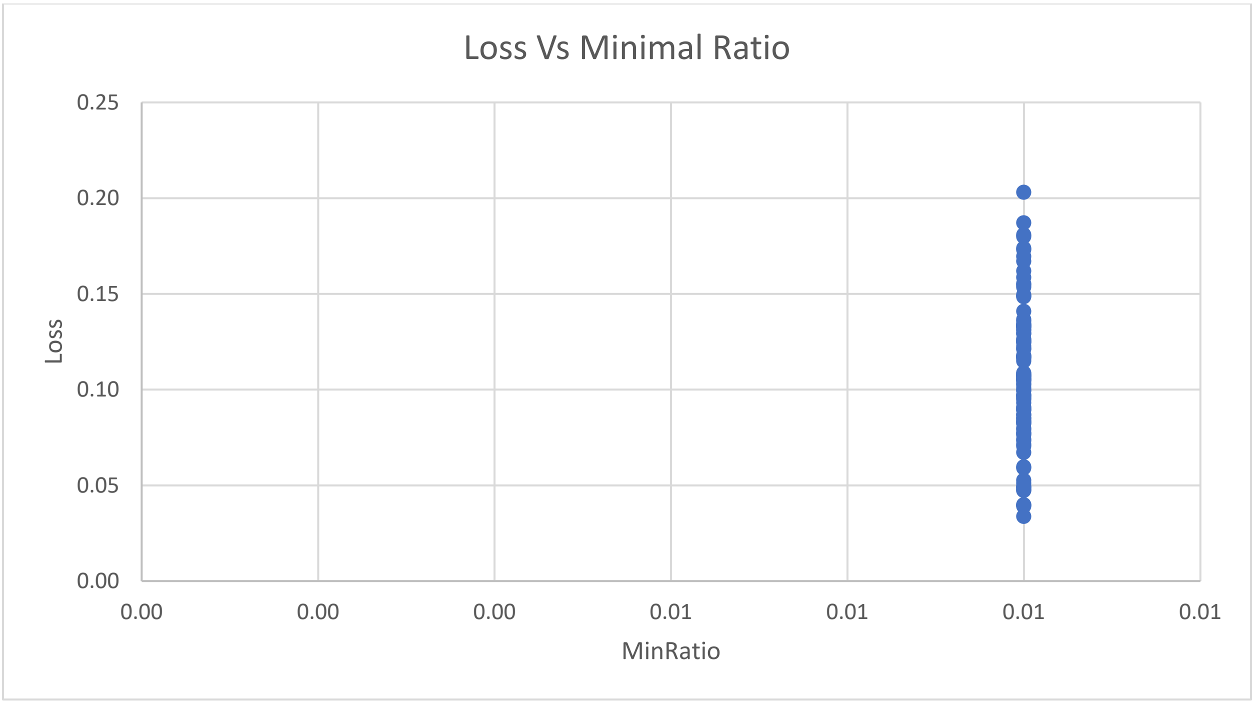
Discussion:

The gradient normal and loss for the DNN shows that as the loss decreased and the weights began to settle in their optima the gradient of the weights started to flatten to zero. This shows that an optimum was close to being reached and can be used as a metric for when enough training has been done. On the other hand, the gradient norm for the CNN was very chaotic and match that of the loss which bounced up and down rather sharply. This may have been due to the inability of the network to reach lower stages of loss. The training needed to be done on a smaller network for performance reasons and that may have been one cause for the loss to be stuck in an area with many local minima which caused the gradient normal to be so chaotic.

HW1-2 What happens when gradient is almost zero:

State how you get the weight which gradient norm is zero and how you define the minimal ratio.

To get the weights and produce the gradient normal all the weight gradients of the network are collected squared and summed. The resulting value is then raised to the 0.5 to get the gradient normal. This function is used as a loss function to update the model weights. The optimizer should then be able to reduce the gradient to near zero. With a list of gradient normals a hessian matrix can be applied to the list of values to produce a matrix. Then this matrix is solved for its eigen values. The minimal ratio is defined by the ratio of positive eigen values produce from the hessian matrix.



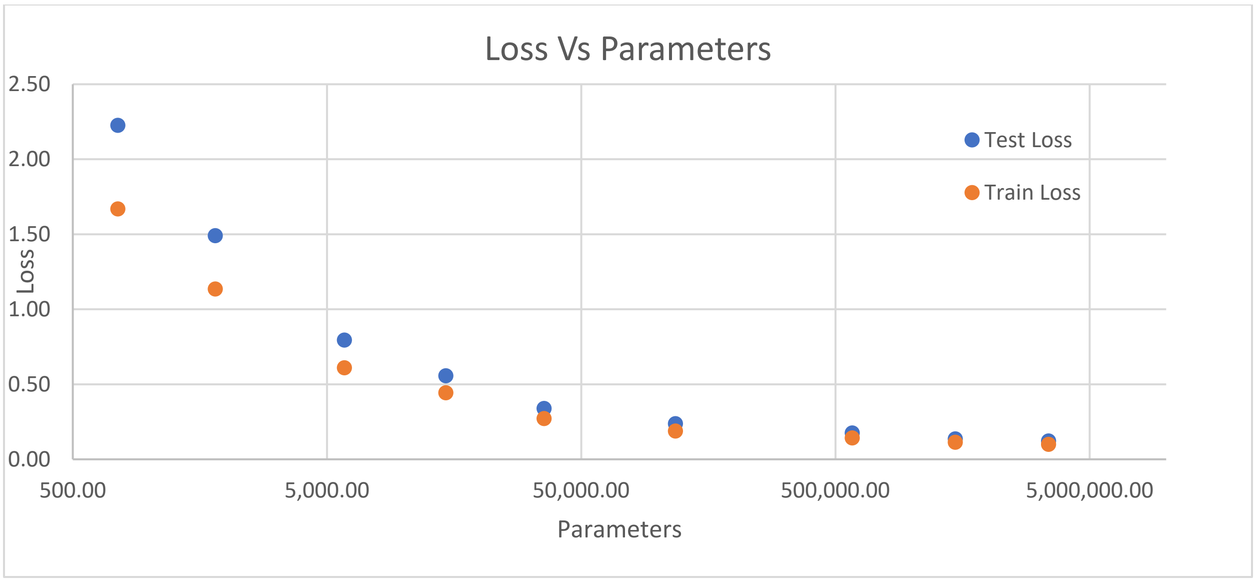
Discussion:

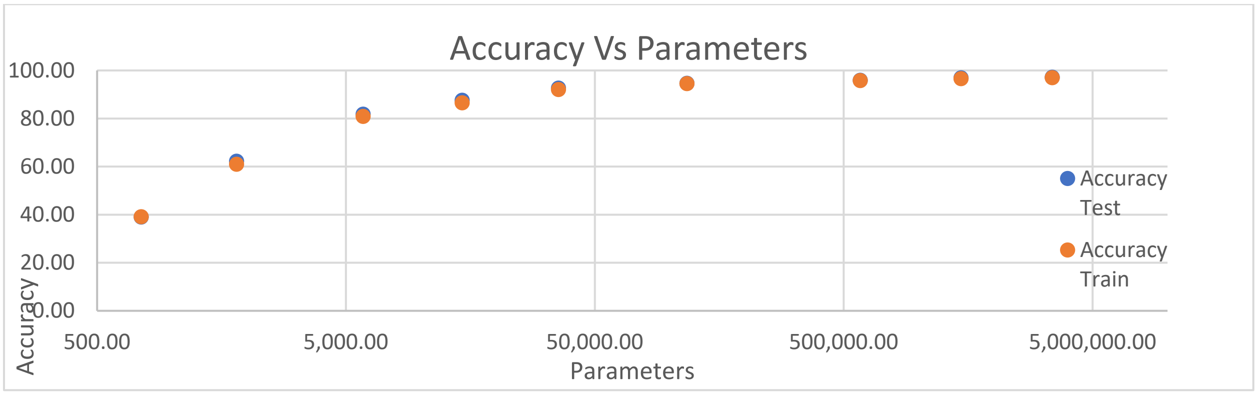
As seen by the graph the minimal ratio is not correct. There could be a few reasons for this, but the best explanation is that the gradient normal was not introduced correctly into the optimizer and therefore a gradient near zero was not possible. Had this been implemented correctly the minimal ratio values that were higher would suggest that the loss was at a minimum. Conversely a low minimal ratio with a high loss value would suggest the model had been stuck in a local minimum.

HW1-3 Number of parameters v.s. Generalization:

For this experiment, the 10 CNNs were trained on the MNIST set and their layers were kept the same. The only variable that was change was input and output features of applicable layers. These features were scaled to increase and decrease the parameter count of each network.

Deep Learning Homework 1 Report





Discussion:

There was not any significant difference between the training and testing loss or accuracy of each model, but this may be due to the structure of the CNN. The drop out layers may have prevented the parameters from overfitting and allowed the model to retain its explainability.

Code Available At:

https://github.com/beamreaching1/Deep-Learning