Saroj Paudel
CPSC-8430 Deep Learning HWK1 2/9/2024

1-1 Deep vs Shallow

In this assignment, three DNN models were created to simulate a Sinc function and a sawtooth function. The models were configured differently in terms of depth and number of parameters. Weight decay technique was applied to prevent overfitting. The maximum epoch was set to 15000 and the loss at convergence set to 1e-4.

Model 1:

- 7 dense layers, 571 parameters
- Activation function: ReLU
- Loss function: MSELoss
- Optimizer: Adam
- Hyperparameters: Learning rate: 1e-4, Weight decay: 1e-4

Model 2:

- 4 dense layer, 572 parameters
- Activation function: ReLU
- Loss function: MSELoss
- Optimizer: Adam
- Hyperparameters: Learning rate: 1e-4, Weight decay: 1e-4

Model 3:

- 4 dense layer, 571 parameters
- Activation function: ReLU
- Loss function: MSELoss
- Optimizer: Adam
- Hyperparameters: Learning rate: 1e-4, Weight decay: 1e-4

1-1-1 Function 1 : Sinc Function

The simulated function is shown below.

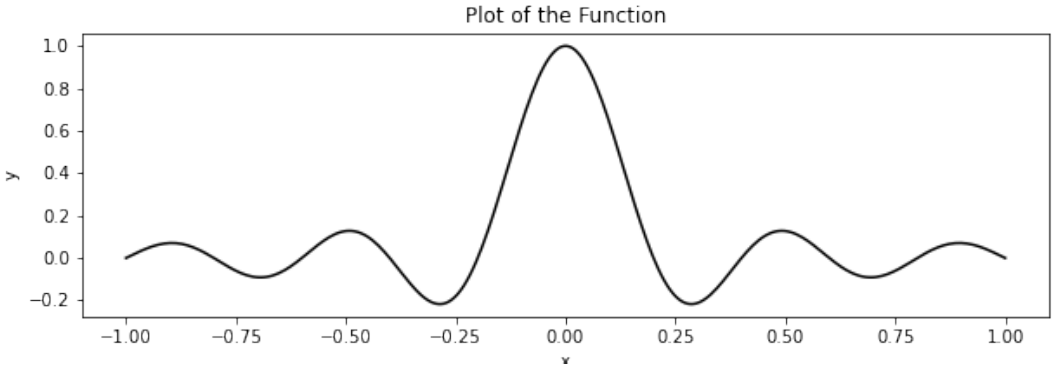

Figure 1: Plot of first function

After training, the result is shown in the figure 2 and the corresponding loss and accuracy in figure 3 and 4 respectively. In Figure 3 we can see that the Model 2 did not converge to the maximum loss specified within the 15000 epochs while Model 1 and Model 3 converged within 8000 epochs and 4000 epochs respectively.

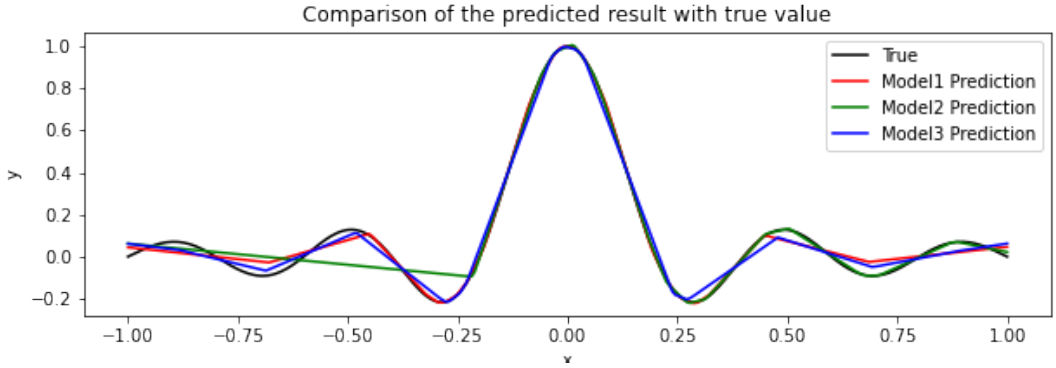

Figure 2: Comparison of the training results with the true value

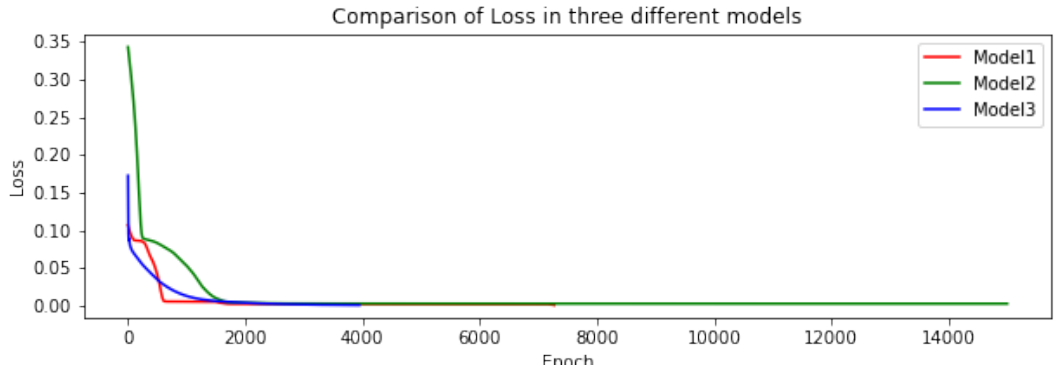

Figure 3: Plot of training Loss vs Epoch

1-1-2 Function 2: Sawtooth function

The simulated sawtooth function is shown in figure 4. And the training results, the loss function and the comparison with the true function are shown in figure 5 and figure 6 respectively.

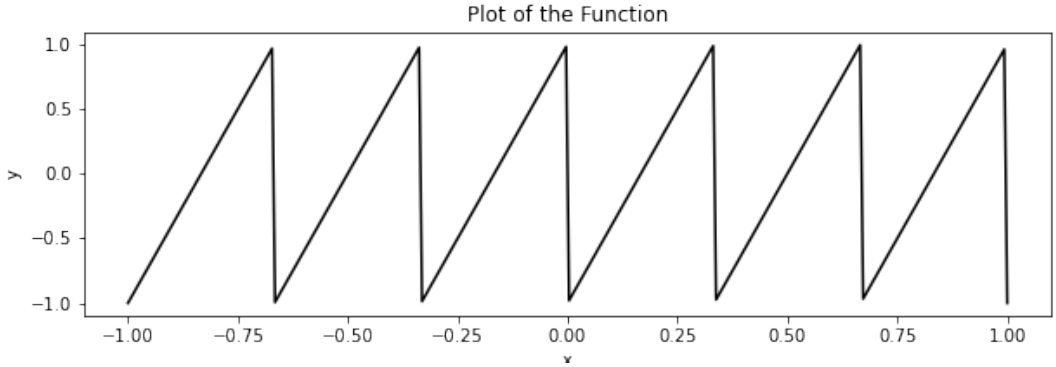

Figure 4: Plot of Second function

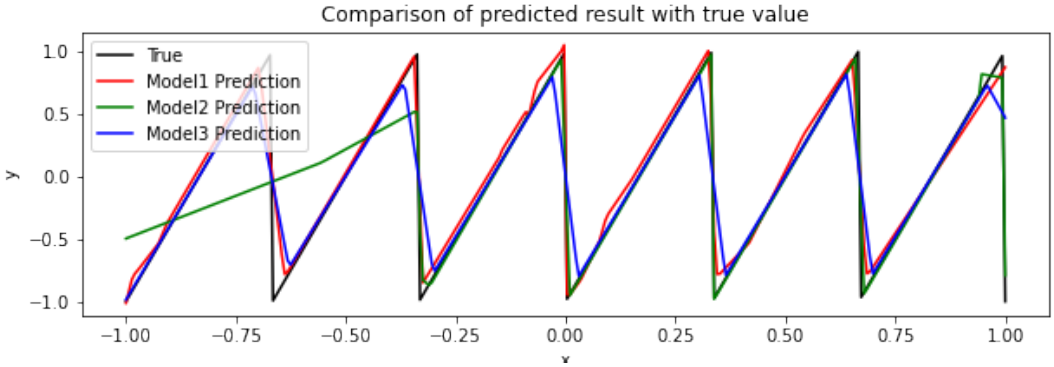

Figure 5: Comparison of training result with the true value for second function

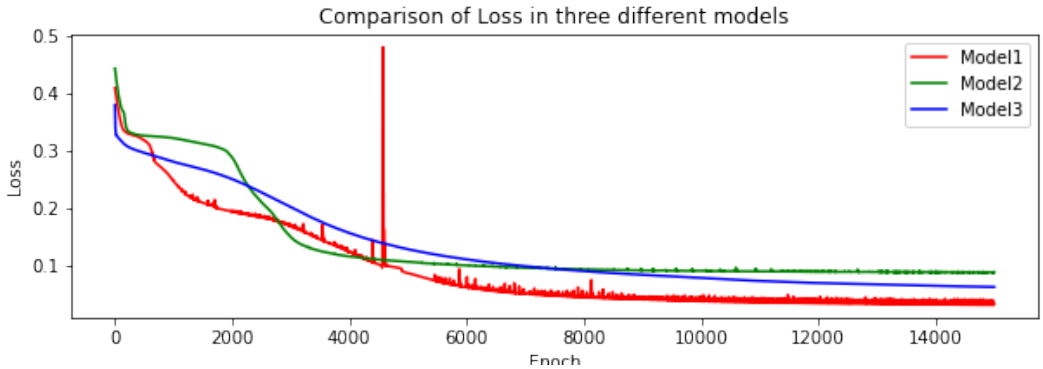

Figure 6: Plot of training Loss vs Epoch for second function

1-1-3 Train on Actual Task

For the actual task, three different CNN models were used with the MNIST dataset.

CNN-1

The parameters as shown below:

(conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
(conv2_drop): Dropout2d(p=0.5, inplace=False)
(fc1): Linear(in_features=320, out_features=50, bias=True)
(fc2): Linear(in_features=50, out_features=10, bias=True)

Total number of parameters: 21840

CNN-2

The parameters are as shown below:

(conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
(conv2_drop): Dropout2d(p=0.5, inplace=False)
(fc1): Linear(in_features=320, out_features=50, bias=True)
(fc2): Linear(in_features=50, out_features=20, bias=True)
(fc3): Linear(in_features=20, out_features=10, bias=True)

Total number of parameters: 22560

CNN-3

The parameters are as shown below:

(conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(20, 20, kernel_size=(5, 5), stride=(1, 1))
(conv2_drop): Dropout2d(p=0.5, inplace=False)
(fc1): Linear(in_features=320, out_features=80, bias=True)
(fc2): Linear(in_features=80, out_features=20, bias=True)
(fc3): Linear(in_features=20, out_features=10, bias=True)

Total number of parameters: 38050

Loss function is negative log likelihood (NLL) and activation function, optimizer, and other hyperparamet ers (learning rate, decay rate and drop out rate) are all identical to the earlier problem. Figure 7 shows t he training loss for three different CNN model for the MNIST dataset while Figure 8 shows the model acc uracy for the corresponding test dataset. CNN-1 resulted in a better result both in terms of minimum los s and higher accuracy while CNN-2 performed the worst.

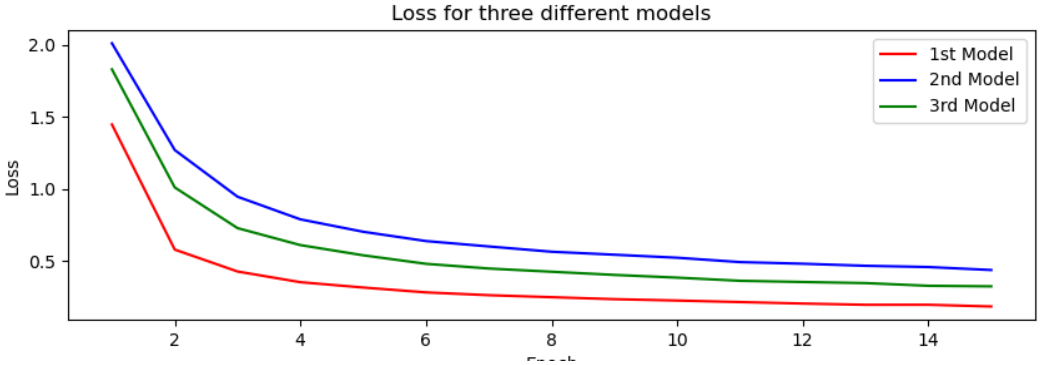

Figure 6: Plot of training Loss vs Epoch for MNIST dataset

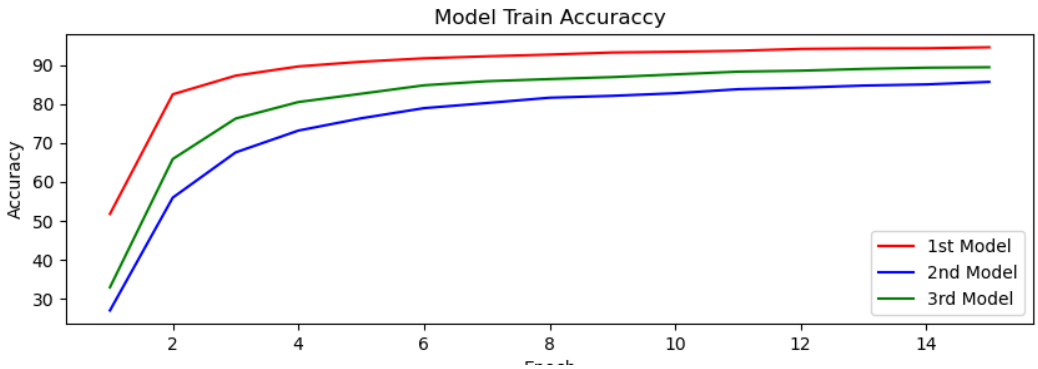

Figure 7: Accuracy vs Epoch

1-2 Optimization

1-2-1 Visualization of the optimization

A DNN model with the following parameters was used for this exercise.

(fc1): Linear(in_features=784, out_features=500, bias=True)
(fc2): Linear(in_features=500, out_features=50, bias=True)
(fc3): Linear(in_features=50, out_features=10, bias=True)

No. of parameters: 418,060
Activation function: ReLU
Loss Function: Cross-Entropy Loss
Optimizer: Adam
Learning rate: 1e-4
Decay rate: 1e-4

The model was trained eight times, each with 45 epochs each. Figure 8 illustrates the PCA result for the whole model while Figure 9 shows the PCA results for a single layer (Layer 1)

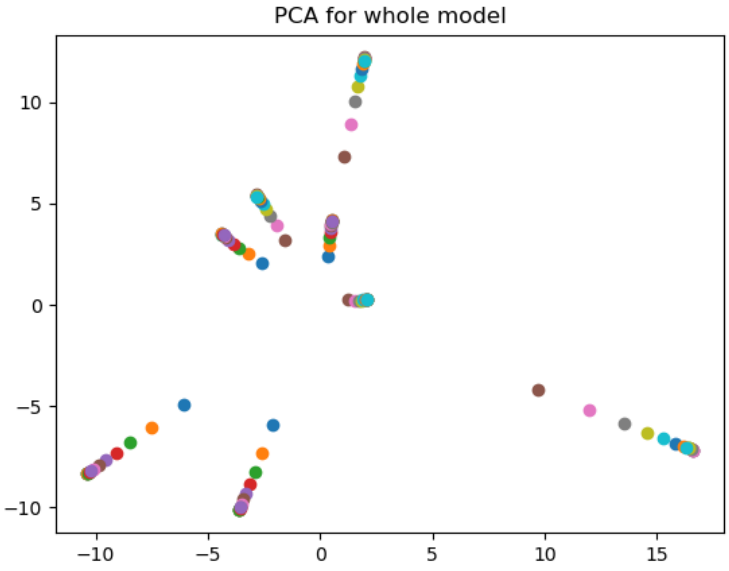

Figure 8: PCA of Whole DNN model

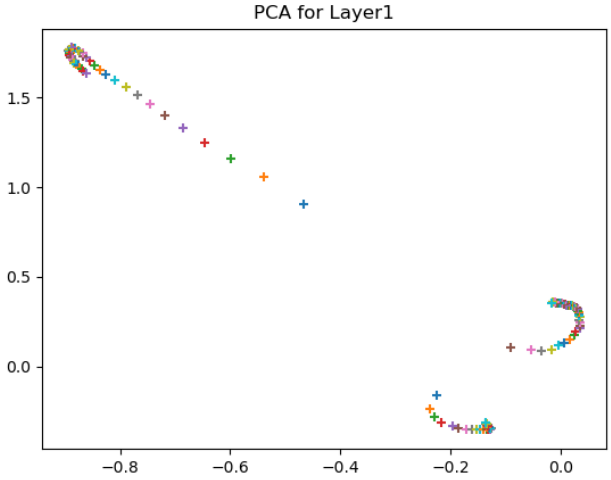

Figure 9: PCA of Layer 1

1-2-2 Gradient Norm During Training

For this exercise we resort back to the non-linear Sinc function from the first exercise. A DNN with single dense layer as shown below is used.

DNN Model

(fc1): Linear(in_features=1, out_features=500, bias=True)
(fc2): Linear(in_features=500, out_features=1, bias=True)
Total number of parameters: 1501
Activation Function: ReLU
Loss function: MSE

Optimizer: Adam
Learning rate: 1e-4
Decay rate: 1e-4

The model was trained for 5000 epochs. The gradient norm that measures the rate of change of loss wit h respect to the model parameters was seen to converge. Figure 10 shows the gradient norm and avera ge loss during training.

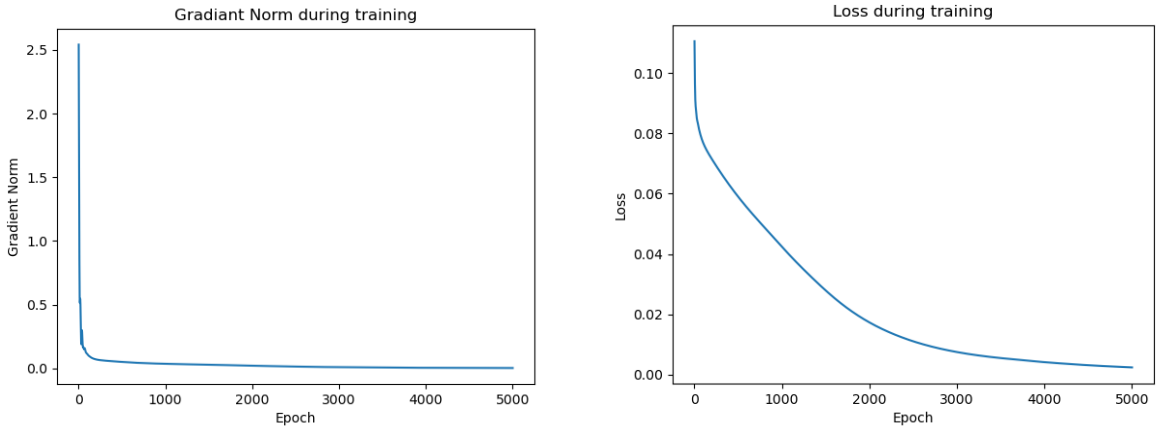

Figure 10. Gradient loss (left) and Average Loss (right) during training.

1-2-3 What happens when gradient is almost zero?

Continued from the earlier exercise, the minimum ratio which signifies the proportion of Hessian eigenvalues greater than zero was calculated

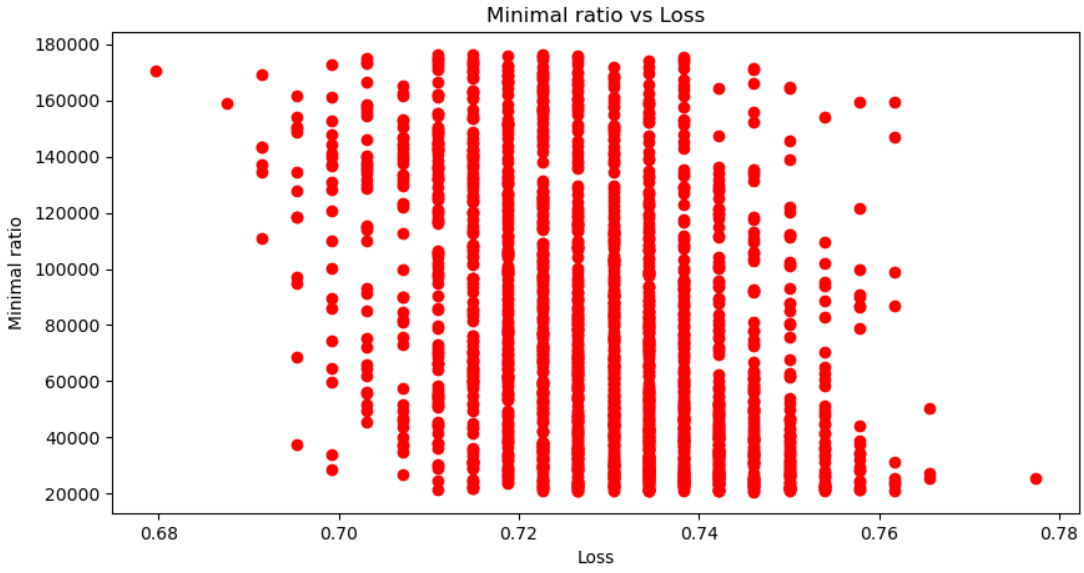

Figure 11. Minimal ratio vs Loss during training for 2000 epochs

1-3 Generalization

1-3-1 Can Network fit random labels?

In this exercise, a DNN model is used on randomized labels on the MNIST dataset. The DNN model parameters are as shown below

(fc1): Linear(in_features=784, out_features=256, bias=True)
(fc2): Linear(in_features=256, out_features=10, bias=True)
Total number of parameters: 203,530
Activation Function: ReLU
Loss function: CrossEntropyLoss
Optimizer: Adam
Learning rate: 1e-4
Decay rate: 1e-4

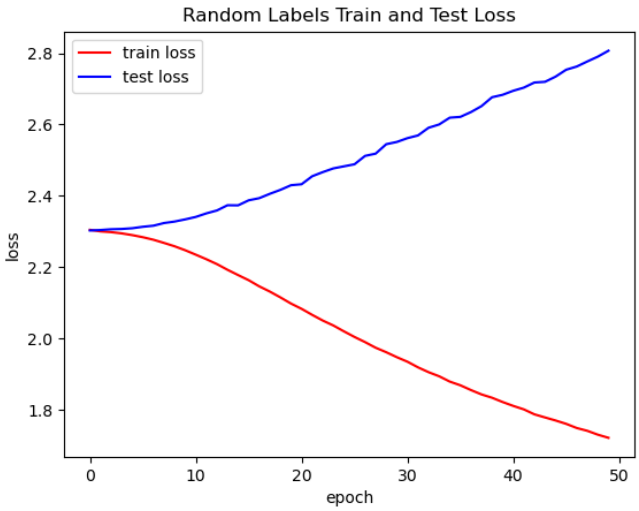

Figure 12: Random Labels train vs Test Loss

As can be seen in the Figure 12, a model with randomized labels is meaningless as the model cannon lea rn to associate the input and corresponding labels.

1-3-2 Number of parameters vs Generalization

For this exercise, 10 different DNNs are generated with different number of parameters as below.

Model 0 , number of parameters: 1600
Model 1 , number of parameters: 3190
Model 2 , number of parameters: 4780
Model 3 , number of parameters: 6370
Model 4 , number of parameters: 7960
Model 5 , number of parameters: 9550
Model 6 , number of parameters: 11140
Model 7 , number of parameters: 12730
Model 8 , number of parameters: 14320
Model 9 , number of parameters: 15910

Other details remain the same except for the number of parameters. After training all the models to the same MNIST dataset, the comparison of the train and test loss with the number of parameters is obtained as shown in Figure 13 below. The accuracy of corresponding models for test and train data is shown in Figure 14. These plots show for the selected DNN models the training loss decreases and training accuracy increases with increase in the number of parameters. However, the same cannot be said for the test data loss and test accuracy. This discrepancy can be attributed to overfitting of the models and can be rectified by using a better decay factor.

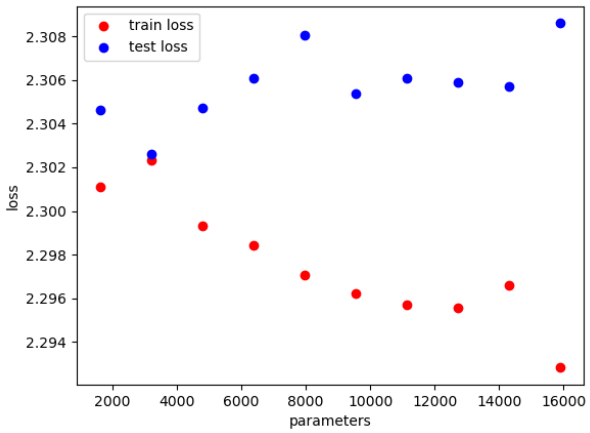

Figure 13: Loss vs number of Parameters

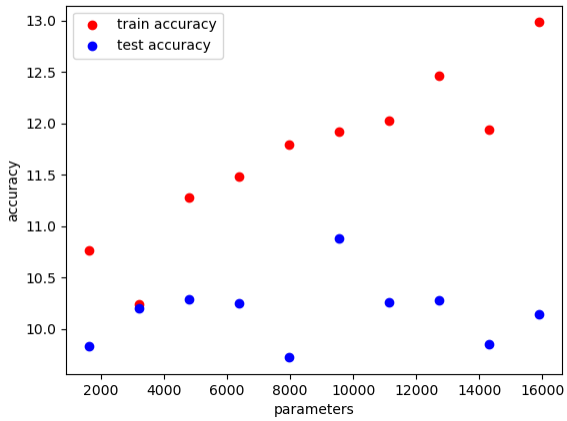

Figure 14: Accuracy vs number of Parameters

1-3-3 Flatness vs Generalization

For this exercise two DNN models (Model A and Model B) with the following parameters were considered and trained separately to same MNIST dataset. Each DNN was trained to 10 epochs.

(fc1): Linear(in_features=784, out_features=256, bias=True)
(fc2): Linear(in_features=256, out_features=10, bias=True)
Total number of parameters: 7960
Activation Function: ReLU
Loss function: CrossEntropyLoss

Optimizer: Adam
Learning rate: 1e-4
Decay rate: 1e-4

Different linear combination of these two DNN models as shown in the following equation was used to evaluate the accuracy and loss of the resulting model. The value of α was varied from -1 to 1.

$$\theta = (1 - \alpha)DNN_1 + \alpha DNN_2$$

The normalized accuracy and normalized loss results are shown in Figure 15. The result shows that for this particular example case, DNN1 performs much better alone as compared to when linearly combined with DNN2 as evident from higher accuracy when $\alpha = 0$. However, when it comes to the training accuracy, a combination for α around 0.4 – 0.5 shows better accuracy.

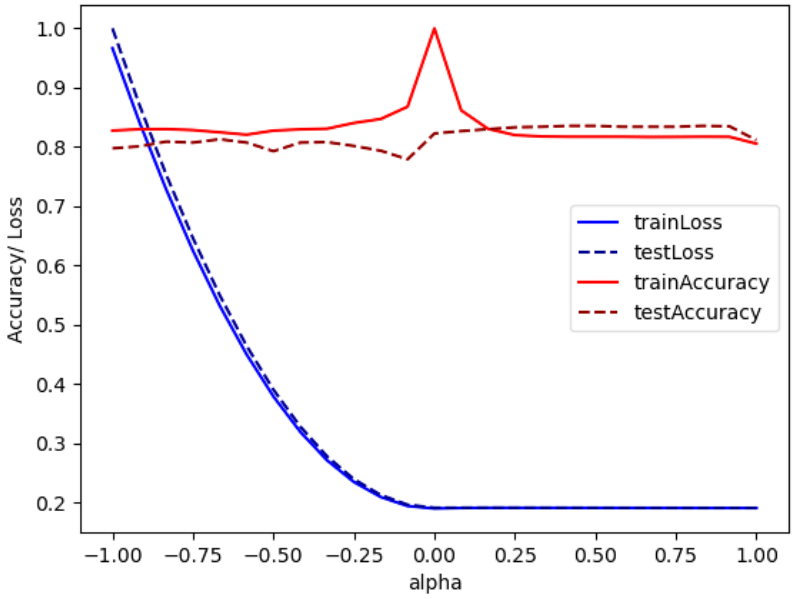

Figure 15: Accuracy and Loss for the interpolated model

1-3-4 Sensitivity to Learning rate.

The same DNN model was used for this exercise. However, the learning rate was changed to study the sensitivity of the model when the learning rate was varied as below:

Five Learning rates: 5e-2, 1e-2, 5e-3, 1e-3, 1e-4

Figure 16 shows the training and test accuracy data as well as the sensitivity data when the learning rate was varied. For the DNN model considered, the accuracy was highest for 10^{-2} learning rate. Contrary to belief, smaller learning rate did not increase the accuracy of the model.

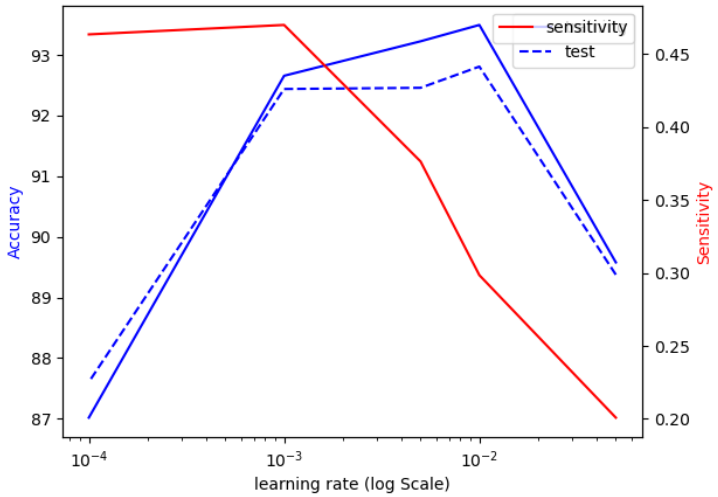

Figure 16: Accuracy and sensitivity against the learning rate