CPSC 8430 – Assignment 1

Part 1 Deep vs. Shallow

Task 1: Simulate a Function

- In task 1, two functions were simulated: 1. Y = sin2x, 2. Y = cos1.5x. As per requirements, the functions are single-input and single-output, and non-linear.
- These two functions are trained with three models: Model 1, Model 2, and Model 3, with parameters about 961, 945, and 975, respectively.
- As given, the parameters of these three models are almost equal.
- The learning rates of all three models are set to 0.001.
- Below figure 1 depicts the loss of Model 1, Model 2, and Model 3 for Function 1 and figure 2 for function 2.

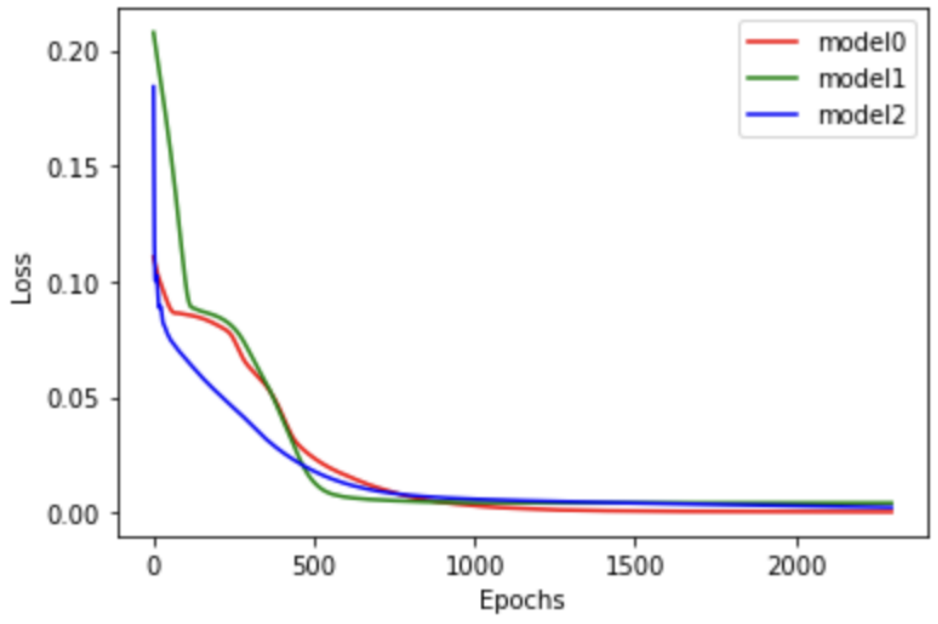

Figure 1

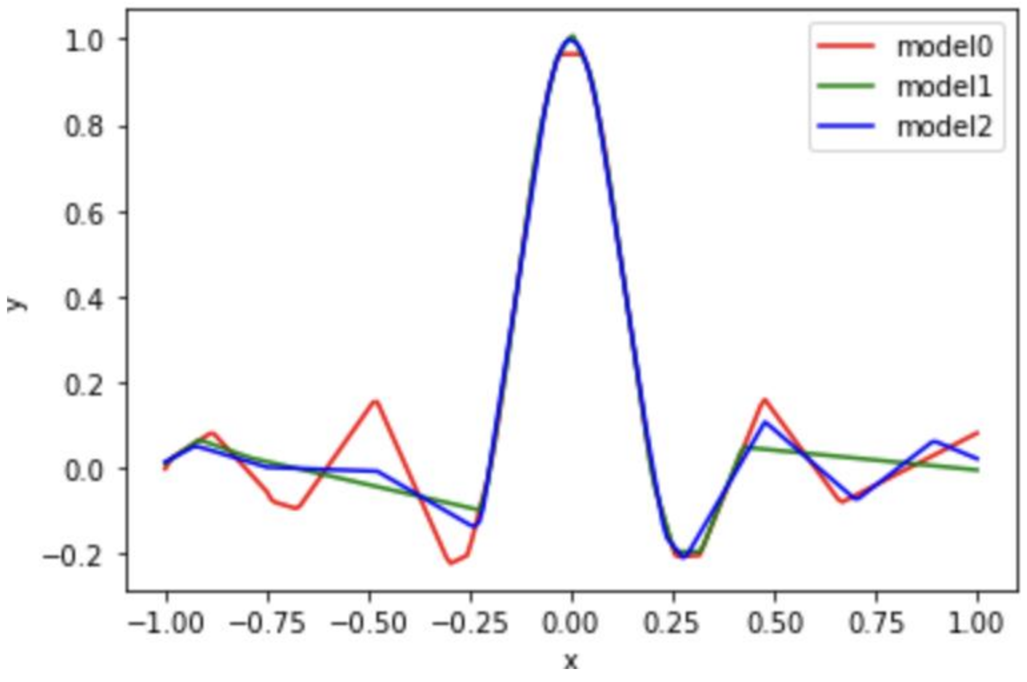

Figure 2

- For the first function, the convergence of all 3 models occurs around 2300 epochs, and for function two, the convergence of the 3 models is around 2300 epochs.
- The figure 3 depicts the ground truth and prediction models for the function y = sin2x. Except for some input values, the models provide almost similar output. Model 2 and Model 3 seem to produce unusual output values for input values ranging from 5.0 to 10.0.

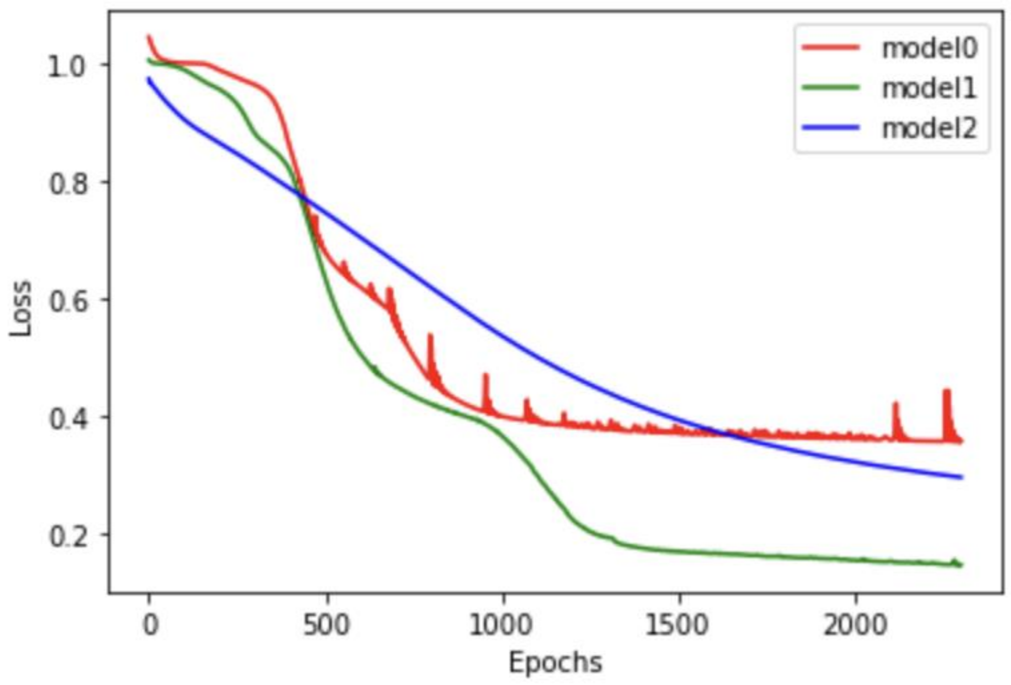

Figure 3

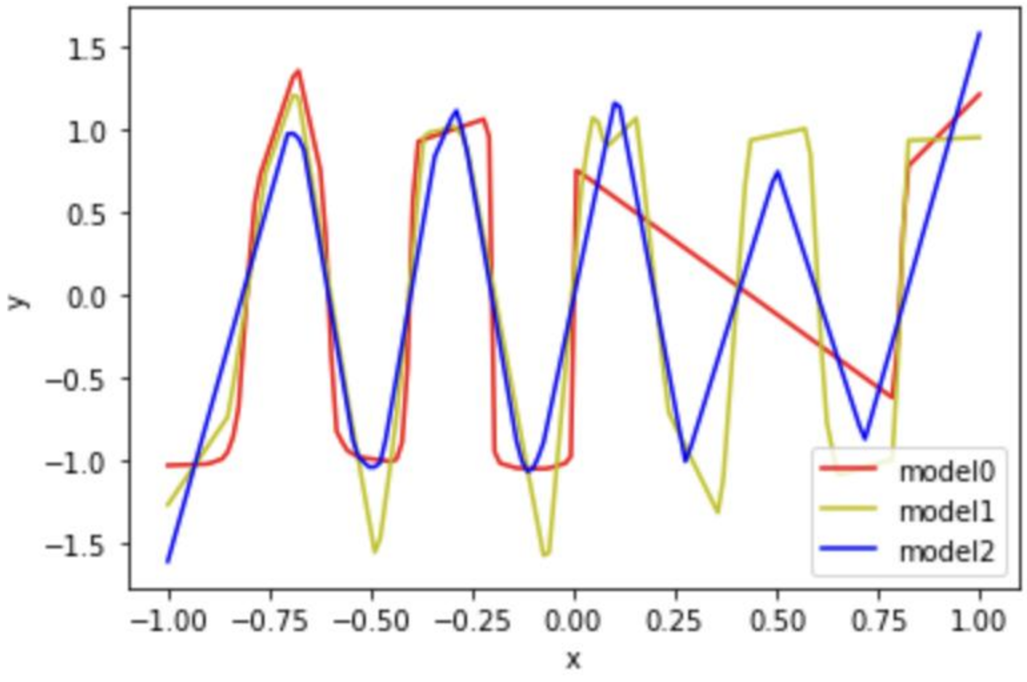

Figure 4

Task 2: Train on actual tasks

- Here I have used the MNIST dataset to train 3 models and visualized the training process by showing both loss and accuracy charts as below.
- Maximum pooling was utilized for pooling, and both models were fully connected. RELU was the activation function used. Each network had a different number of nodes in each layer. The learning rate for each model was 0.001. The loss that was sustained while model 1 and model 2 were being trained is depicted in Figure 5. The two models generally converged within 50 epochs, according to observations.

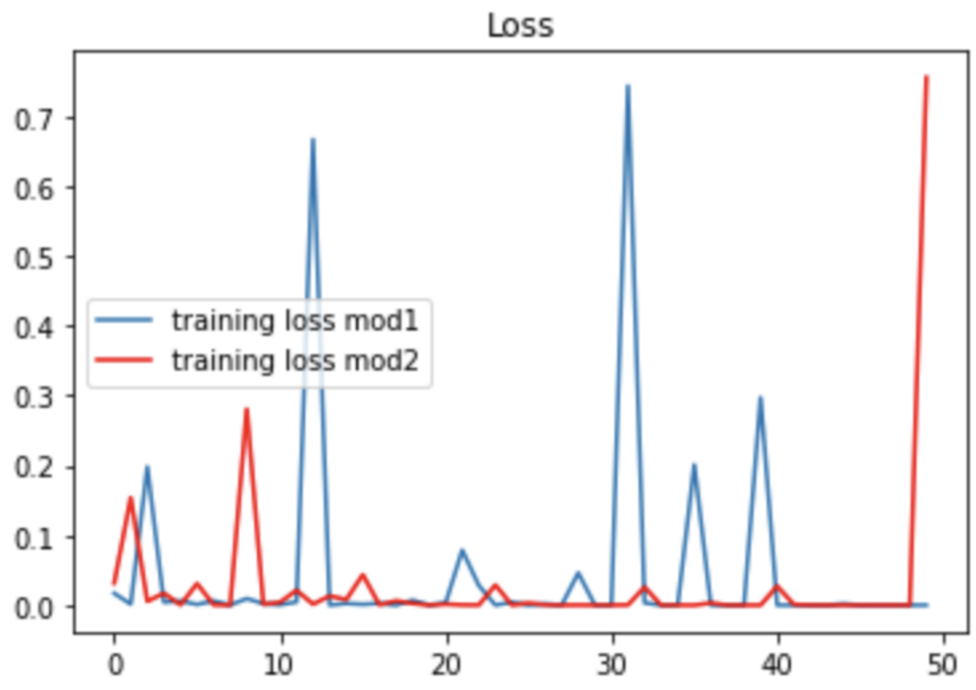

Figure 5

- The accuracy of the two models on training and test sets is shown in Figure 6 during model training. As expected, we can see that the two models regularly outperform each other on training data compared to test data.

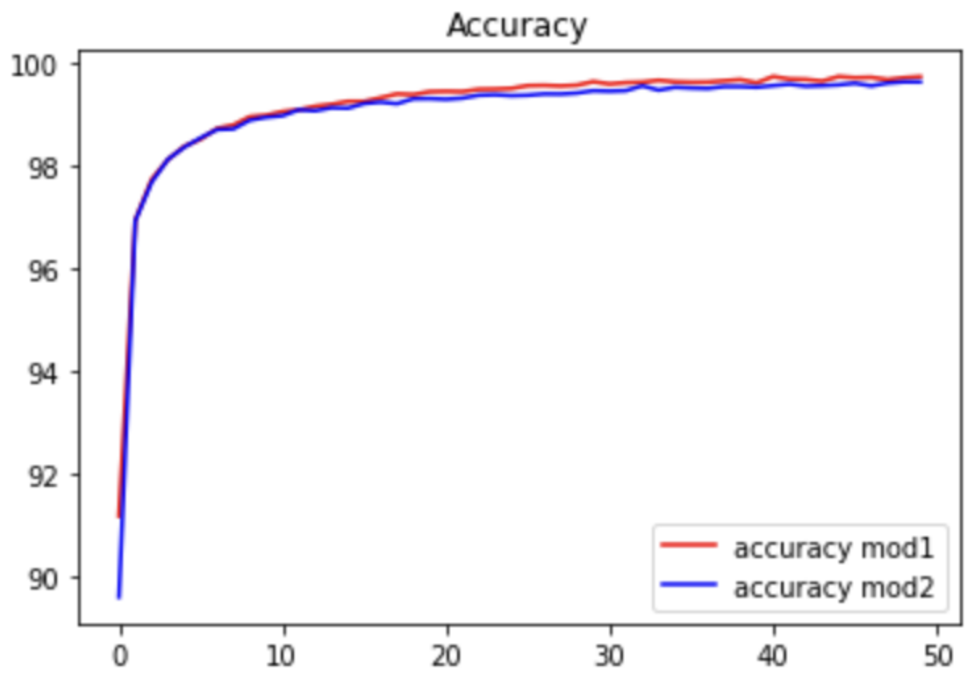

Figure 6

PART 2 – Optimization

Task 1 – Visualizing the Optimization process

- The function simulated here was y=sin(2x)
- The optimization process of the weights inside the network is depicted in Figures 7 and 8.
- Backpropagation optimizes the weights of the network as it learns from the training phase. As the weights are gradually tuned during training, the two figures above indicate this fine-tuning over time.

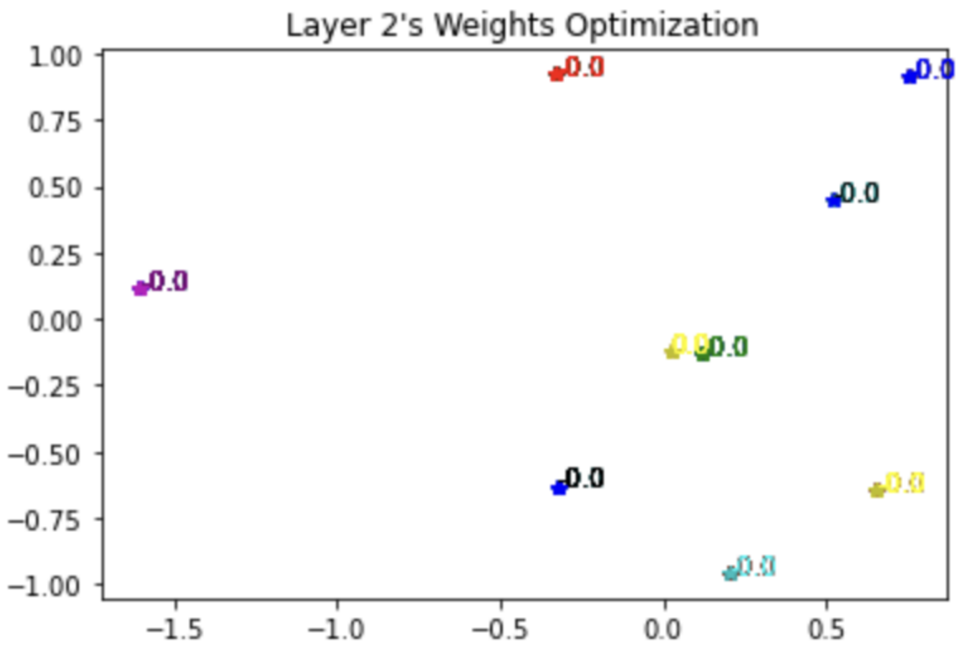

Figure 7

- Figure 7 is Layer 2’s weights optimization, and figure 8 is the whole model’s weight optimization.

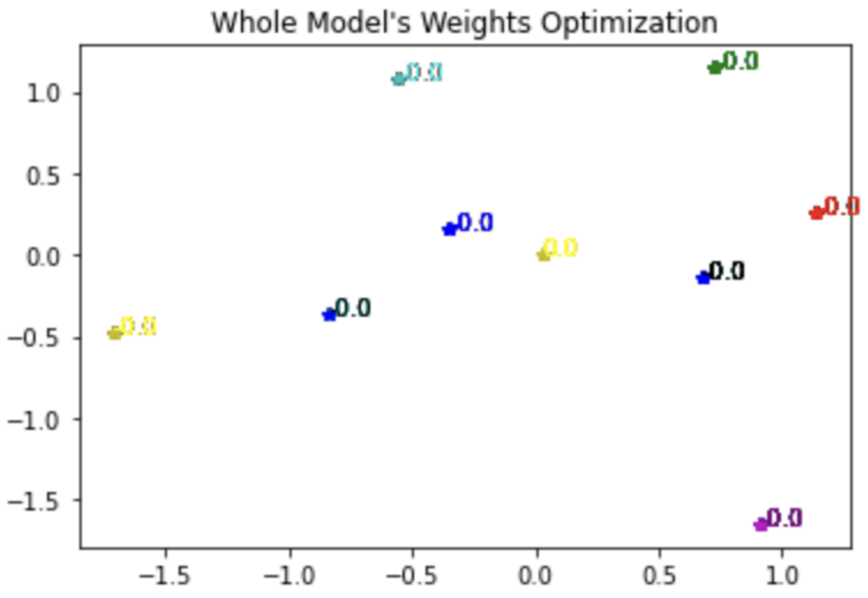
Whole Model's Weights Optimization
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

Figure 8

Task 2 – Observation of Gradient norm during training

- The model's loss is shown in Figure 9 for each iteration of training, while the gradient norm is shown in Figure 10. Each training epoch's loss for the model is depicted in Figure 9.
- The gradient norm for each epoch is shown in figure 10. Each spike in Figure 10 corresponds to a change in slope in Figure 9. The graph 10 differs from the ordinary.

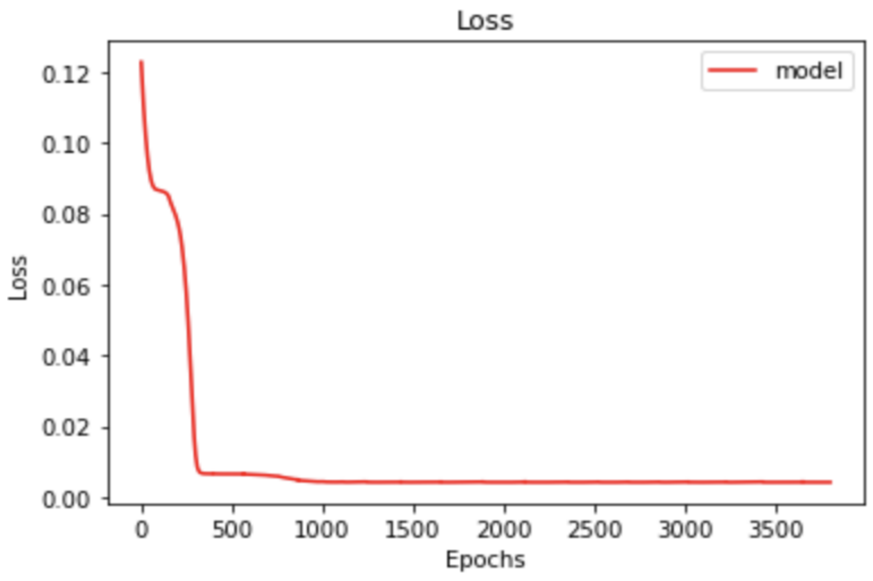

Figure 9

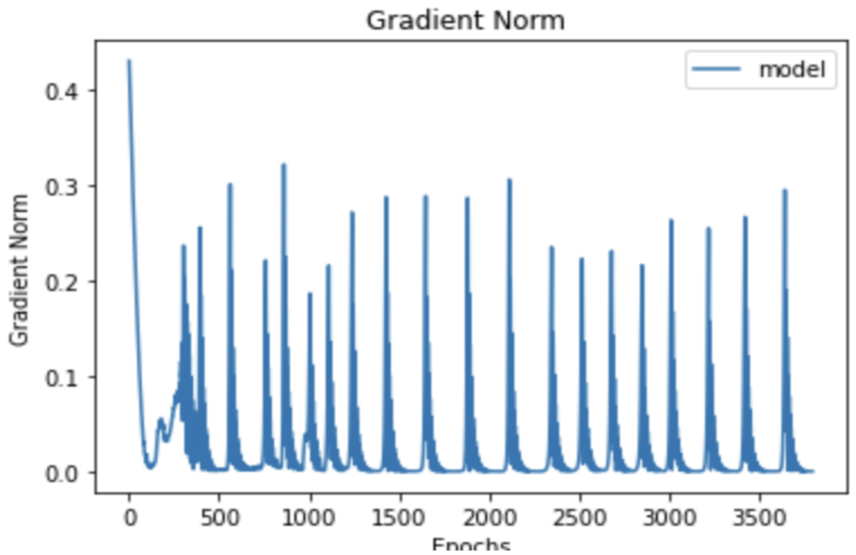

Figure 10

Part 3: Generalization

Task 1 – Can Network fit random labels?

- In this task, I have used the MNIST dataset to check if Network can fit random labels.
- This is a Deep Neural Network model with 3 hidden layers. It was trained for 3800 times on the MNIST dataset.
- The learning rate is 0.001. The optimizer chosen is the Adam optimizer.

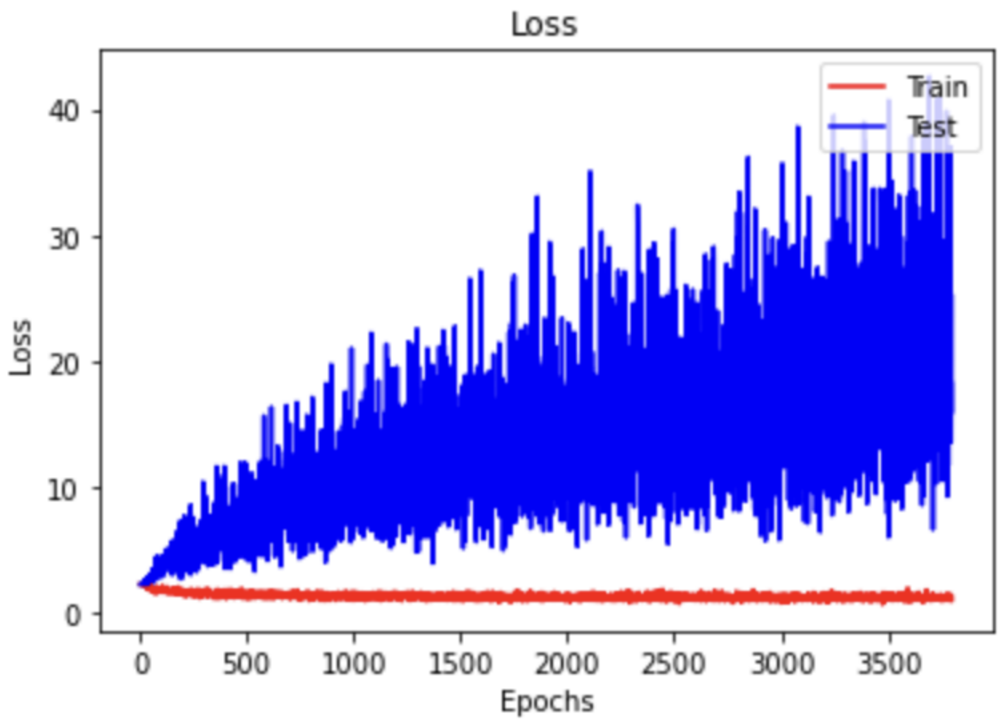

Figure 11

Task 2 – Number of Parameters vs. Generalization

- I have used MNIST dataset to train a DNN model same as above but with different parameters for each of the 10 models ranging between given values.
- As expected, the train data has lesser loss and more accuracy than that of the test data and there always seems to be difference between these two even if the parameters are closing to 1,000,000.
- if we notice Figure 12 and 13, expanding the quantity of boundaries
- in the model abatements its misfortune and furthermore increment its precision. Be that as it may, after specific number of boundaries in a model, the model improvement from one cycle to another is unimportant adding extra boundaries further develops the model scarcely.

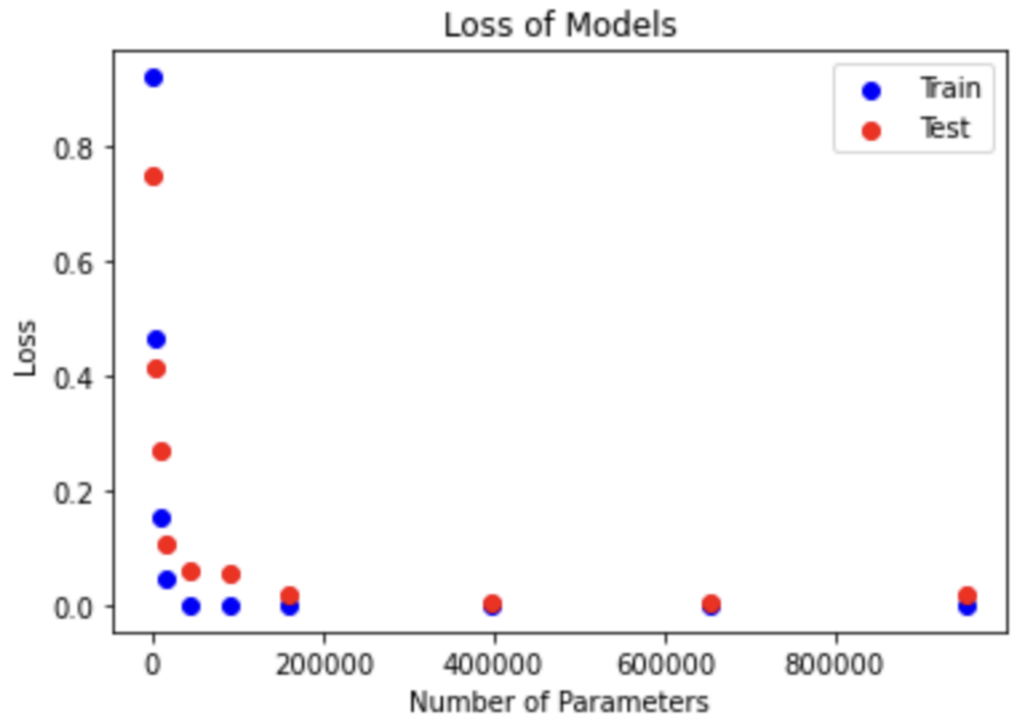

Figure 12

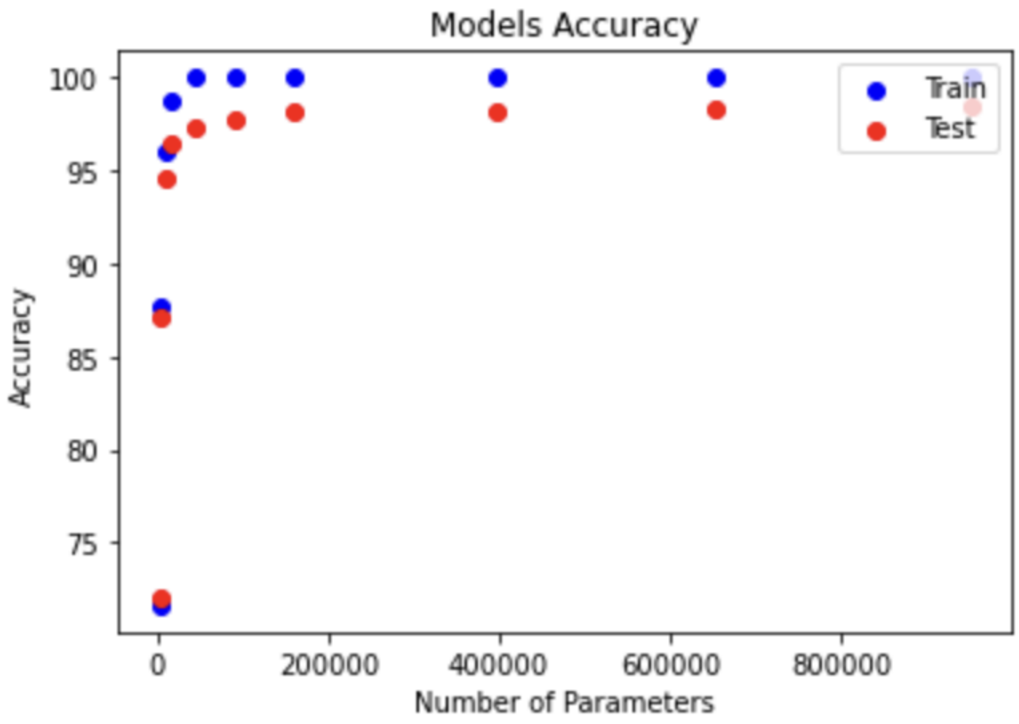
Models Accuracy
Train
Test
Accuracy
Number of Parameters
100
95
90
85
80
75
0
200000
400000
600000
800000

Figure 13

Task 3 – Flatness vs. Generalization

Part 1: -

- For the purpose of training and testing, the MNIST dataset was selected. Two separate batch sizes, 64 and 1024, were utilized to construct the two Deep Neural Networks. A learning rate of 0.001 was established for the entire model. The Adam Optimizer was utilized to optimize the neural network. Alpha represents the linear combination between theta1 and theta2, with theta1 representing the parameters of the first model and theta2 representing the parameters of the second model.ion

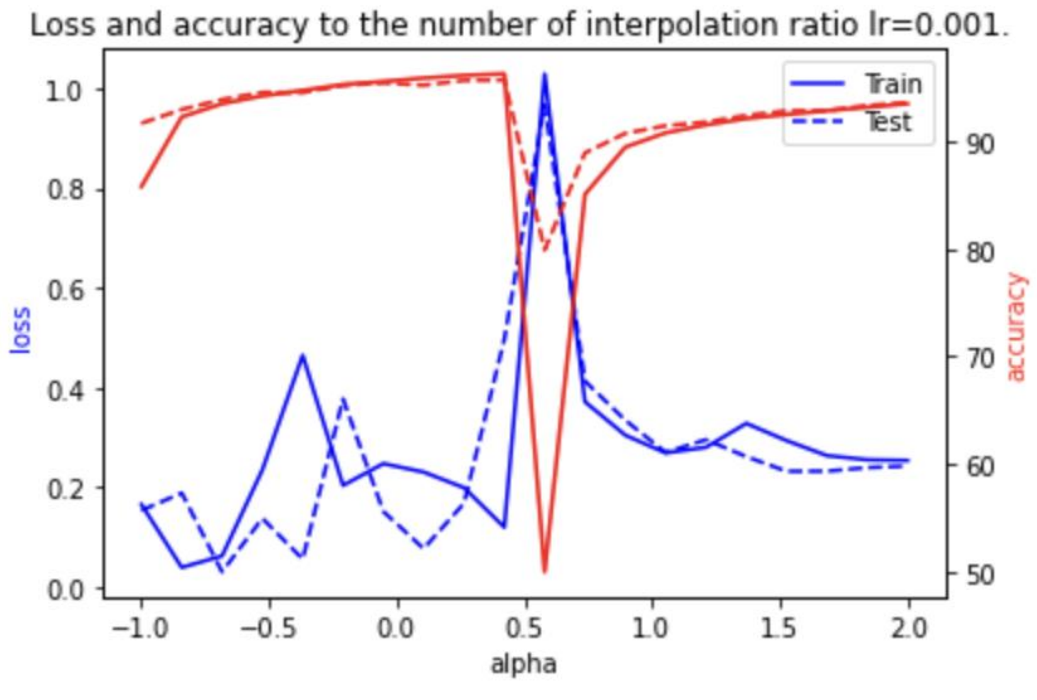

Figure 14

- With a learning rate of 0.001, the loss, accuracy, and linear interpolation alpha of the two models during training are shown in Figure 14. Figure 15, on the other hand, shows the same metrics but with a learning rate of 0.01.

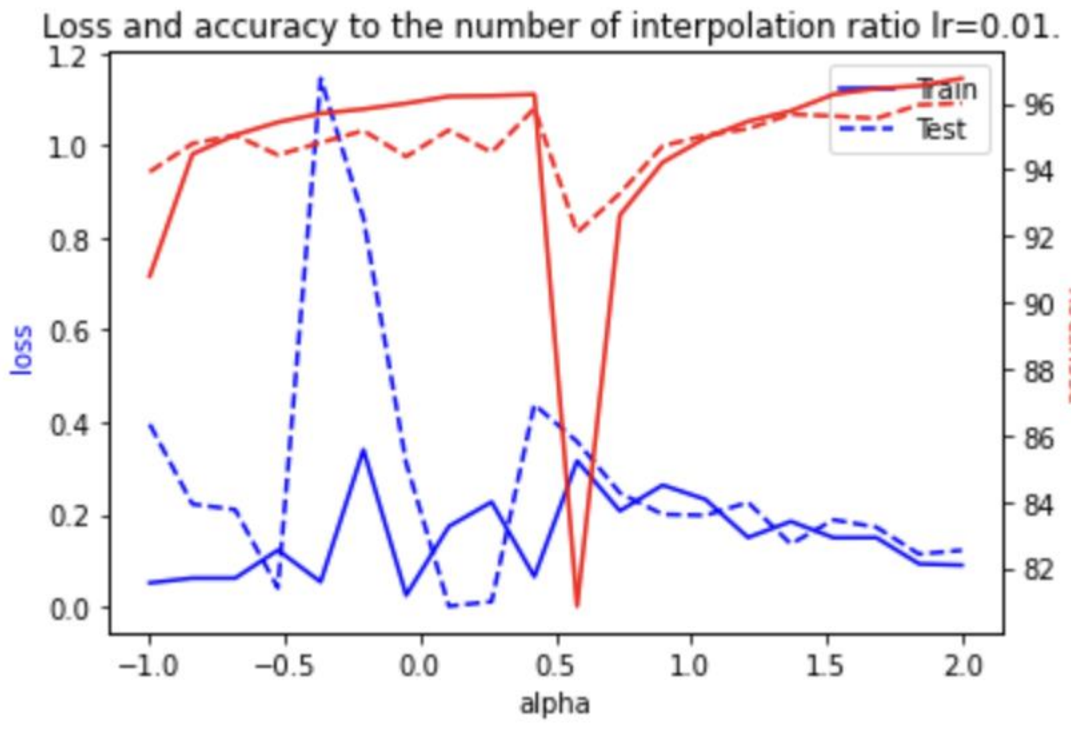
Loss and accuracy to the number of interpolation ratio lr=0.01.
loss
accuracy
alpha
Train
Test
1.2
1.0
0.8
0.6
0.4
0.2
0.0
96
94
92
90
88
86
84
82
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Figure 15

Part 2: Flatness Vs Generalization:

- The training and evaluation of the module were based on the MNIST data set. Each of the five similar Deep Neural Networks had two hidden layers and a total of 16630 parameters. Batches of 5 to 1000 were used to train these networks. In order to improve the optimization process, the Adam Optimizer was used, and each model had a learning rate of 0.001.

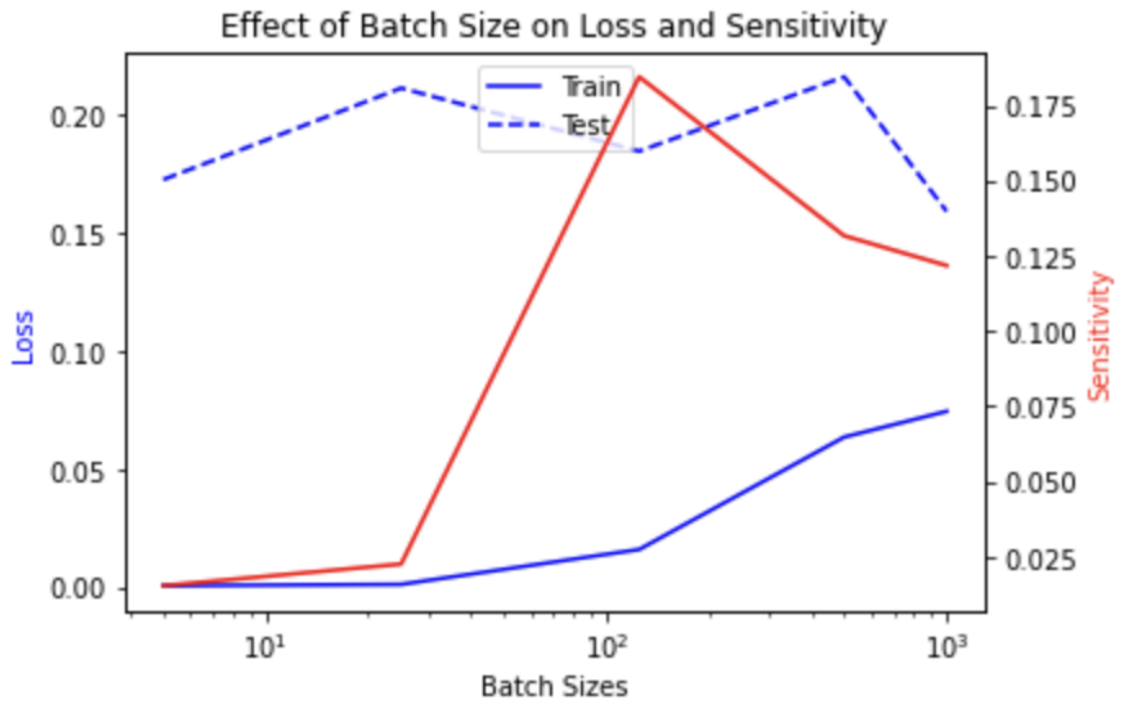

Figure 16

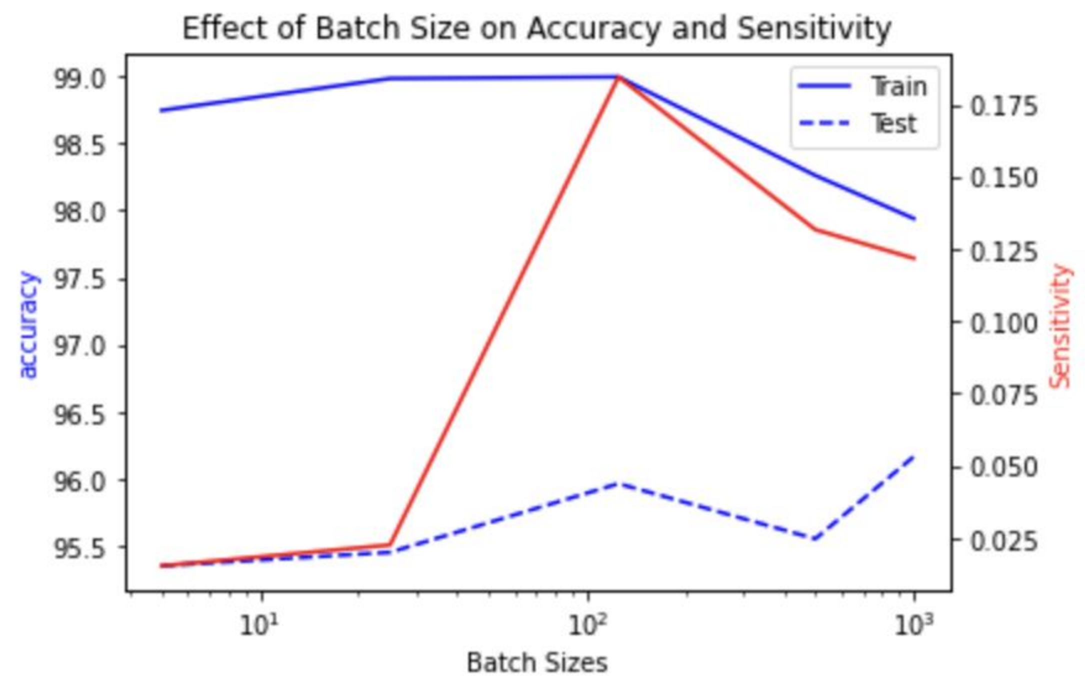

Figure 17

- The accuracy and loss of each of the five models were determined for the training dataset and the test dataset after the training process was completed. The Frobenius norm of the gradient method was then used to determine the models' sensitivity.
- Figures 16 and 17 show how batch size affects sensitivity and accuracy. Sensitivity decreases with increasing batch size. The network is most likely to produce the best results when the batch size is between 100 and 1000, as evidenced by this observation.

Github link- https://github.com/Sujith-sai/DeeplearningHW1