Deep Learning Homework 1- Report

Name-Anjan Kumar Depuru

Course- CPSC 8430-Deep Learning

Part 1: Deep vs Shallow

1:1 Simulate a Function:

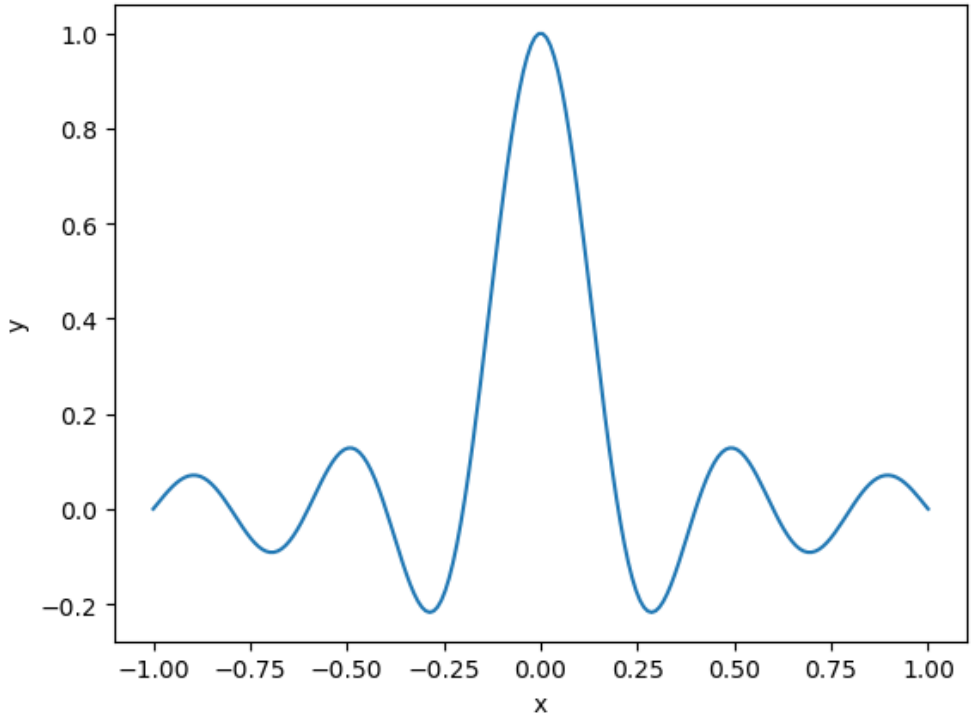
- I had taken (Bonus) which I had attempted and used more than two models.
- I also took (Bonus) which I had attempted and used more than one function.

Each with a unique set of parameters, three distinct models were trained on two distinct functions. Fully connected neural networks with seven, four, and one layers, respectively, are the models designated as model0, model1, and model2 and have 571, 572, and 572 parameters. Each layer has a different average number of nodes. learning rates were 0.001.

The simulated functions included the following: **1) y=(sin(5*(pi*x)))/((5*(pi*x))** 2**) y=sgn(sin(5*pi*X1))**

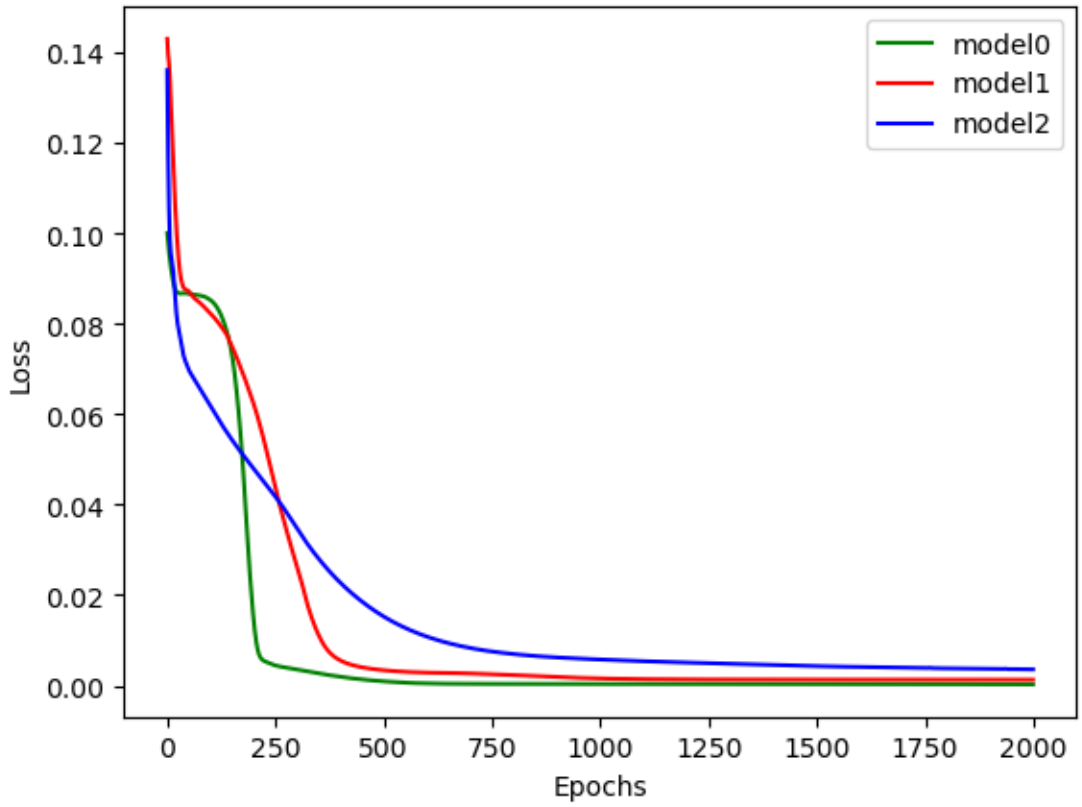
1st FUNCTION:

From the below image, I used the same function: **sin(5*pi*x) / 5*pi*x. (image 1**)

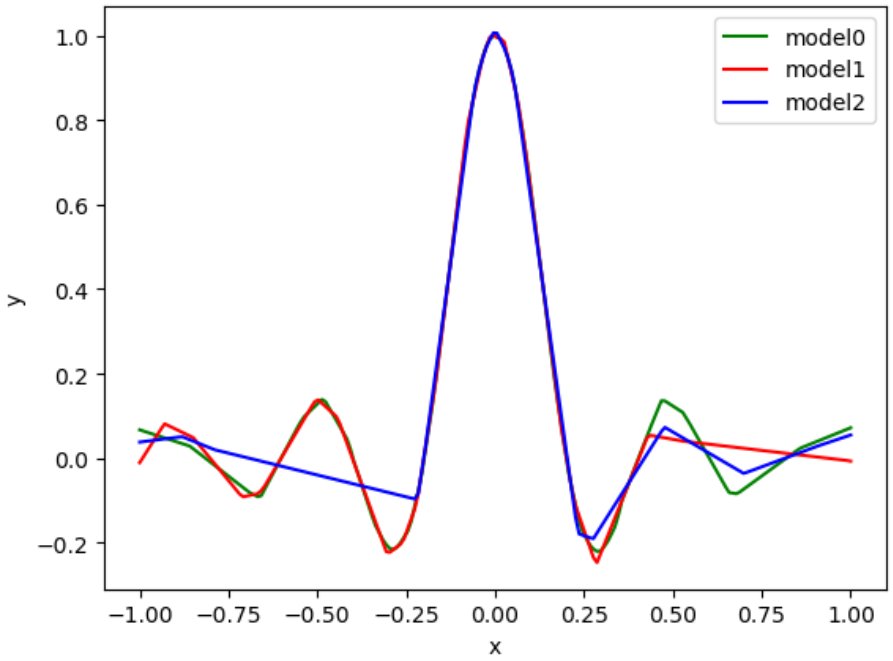


Within a sufficient number of training rounds, all models for both functions experienced loss function convergence. The 2nd image shows the Mean Squared Error (MSE) of the three models (deep, intermediate, and shallow) for each epoch of the training for the simulation (sin(5*(pi*x))/((5*(pi*x)). 2000 iterations are required to reach convergence. The (sin(5*(pi*x))/((5*(pi*x)) function's anticipated and actual values are shown in image 3.

- Now below is a graph showing the loss of each of these models:(**image 2**)

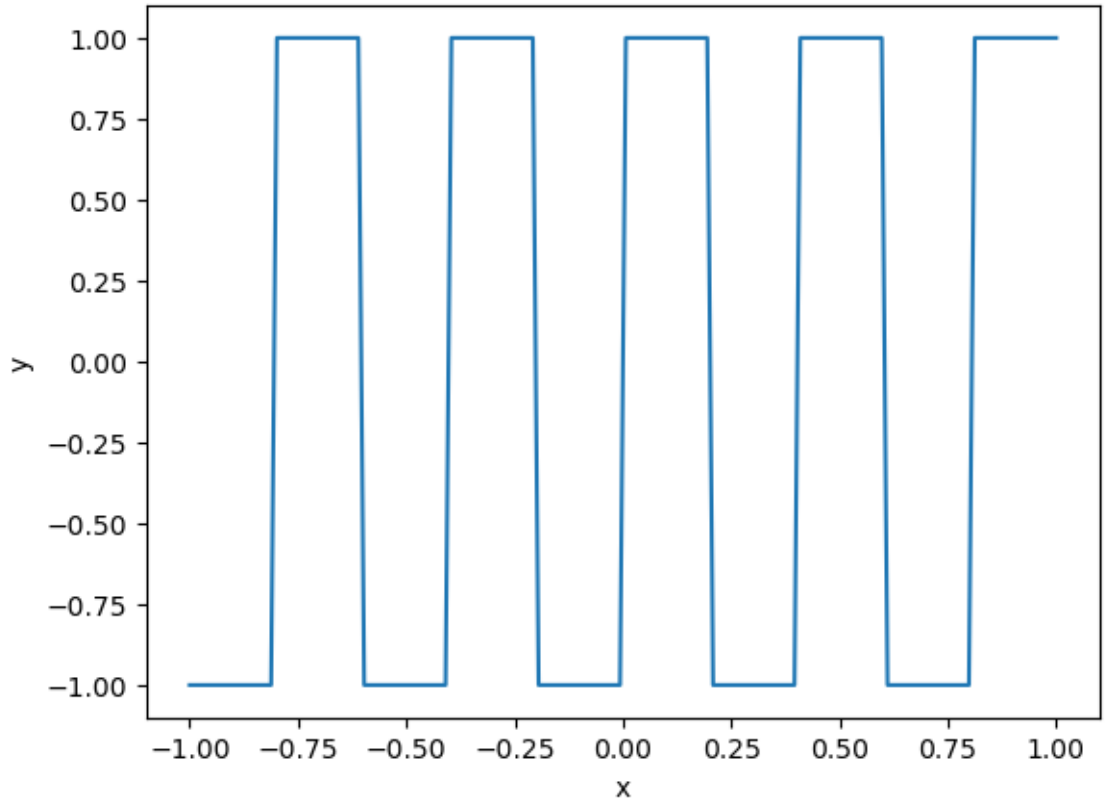


- The following graph contrasts the predictions for the three models. **(image 3)**



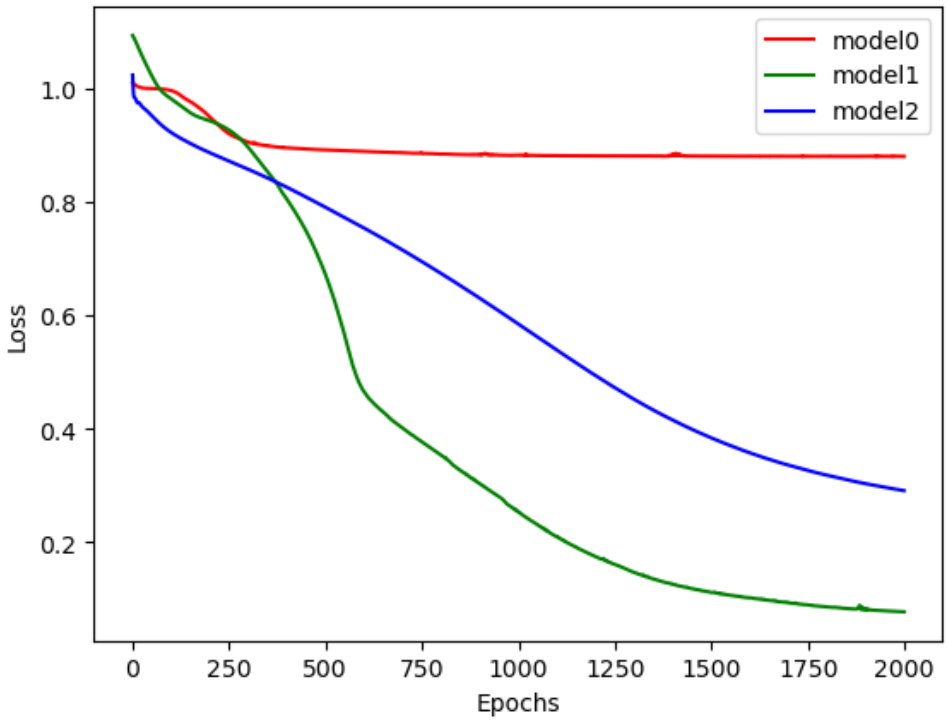
2nd FUNCTION:

Now I am using the 2nd function which is **sgn(sin(5*pi*x) / 5*pi*x)**. Below image gives the function plot of x and y**. (image 4)**

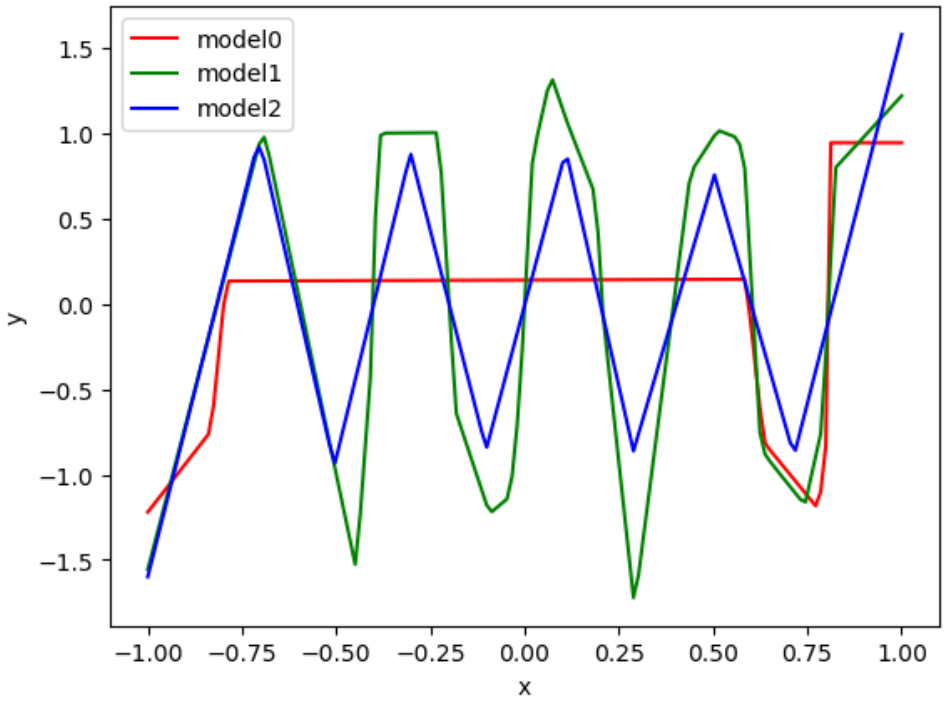


Now the below 5th image shows the Mean Squared Error (MSE) loss that happened during the training of three models. All models converge after a certain number of epochs have been reached for simulation function **sgn(sin(5*pi*x) / 5*pi*x.**

- Below graph showing the loss of each of these models**: (image 5**)



- The 6th image displays the predicted and actual values for the sign(sin(5*pi*X1)) function**. (image 6)**



Result:

All the models reach the maximum number of epochs before convergence as the function is difficult to learn for the 3 models.

If we were given an unknown input, all three (model0,model1,model2) discovered that the accuracy function could produce the desired results. The models in the middle of the graph fared well overall. Less failure at the graph's edges occurs with deeper models.

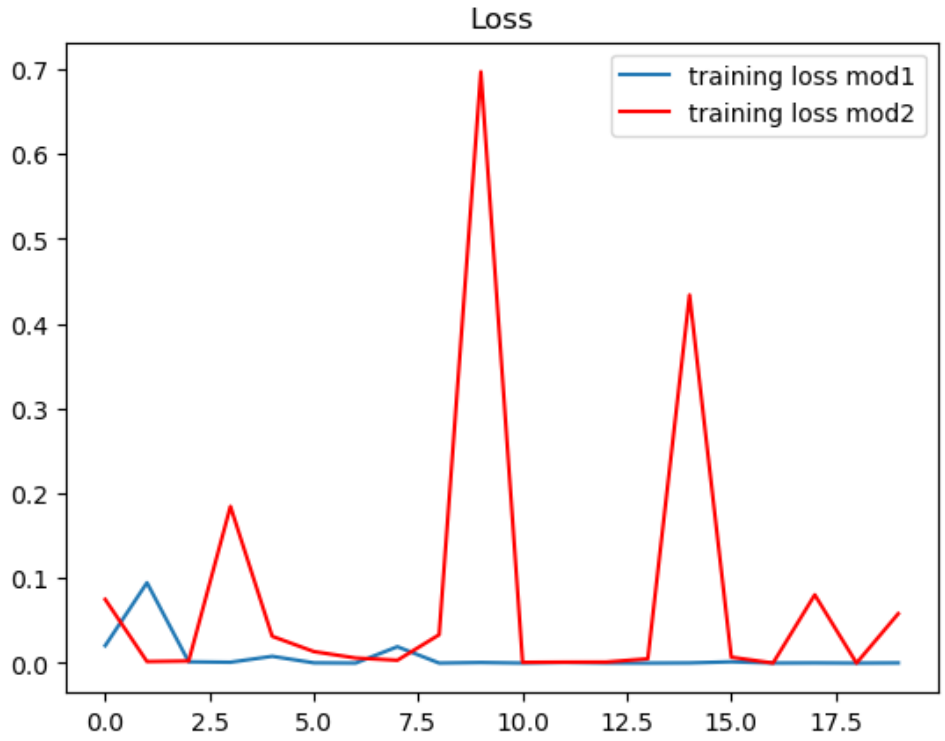
1:1 Train on Actual Tasks

The below models are trained on the MNIST dataset. Training Set: 60,000 Testing set: 10,000. I created two models using Convolutional Neural Networks (CNN) with a batch size of 10.

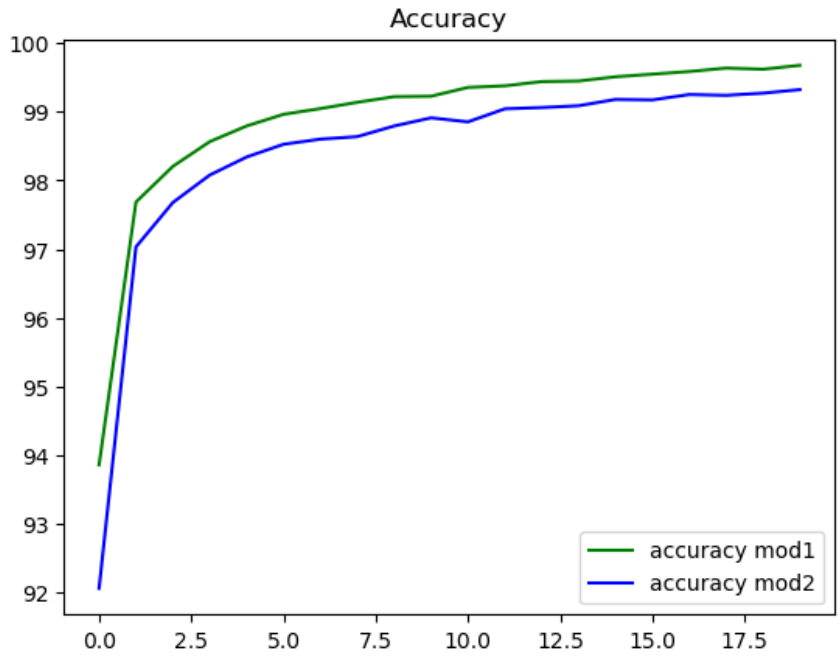
The two models are completely interconnected, and in order to pool the data, we used the max pool function using the activation function. So, I applied a 2D convolution layer and used 2D max pooling and a learning rate of 0.01. Each network's tier has a different number of nodes.

For Two models I have given epochs with a value of 20.

- Below image is the training loss for Model 1 and Model 2. **(image 7)**



- Now the below diagram shows the accuracy of the two models with training and test sets during the training of the models**. (image 8)**



Result:

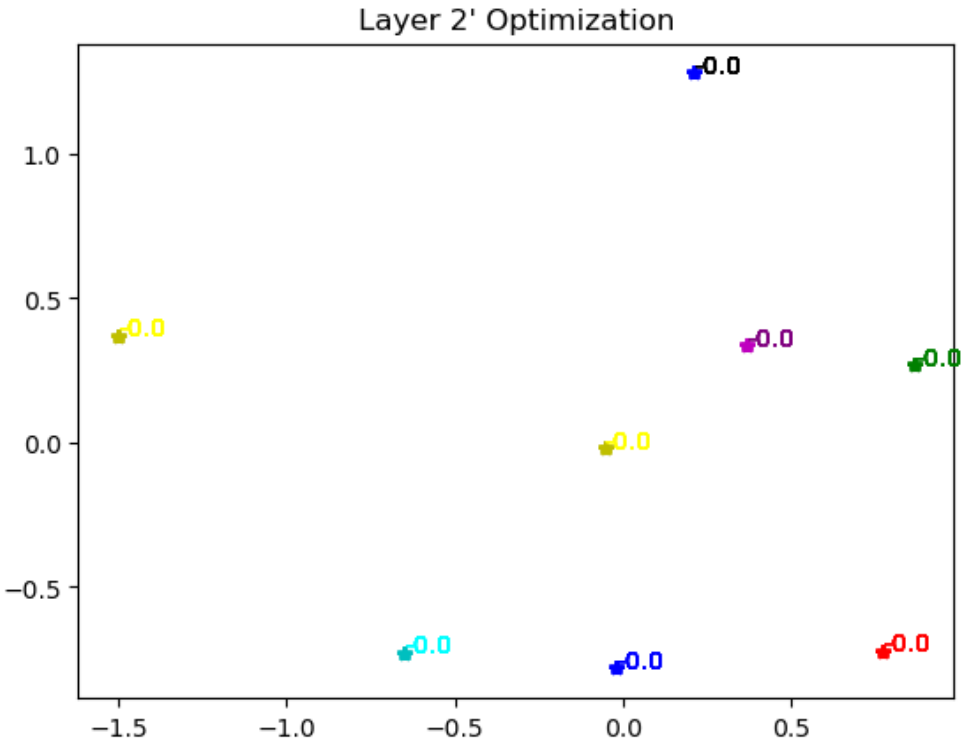
Model 1 is more effective than Model 2 and has a smaller loss; its structure is an optimized CNN model. We can see that, as would be predicted, the two models regularly outperform each other on training data compared to test data.

Part 2: Optimization

1-2 Visualize the optimization process.

The function (sin(5*(pi*x))/((5*(pi*x))) was trained using a Deep Neural Network with three fully connected layers. I used 57 parameters to use to stimulate the function.

Now below image shows one of the layers of the network and it shows the 2nd layer of the network**. (image 9)**

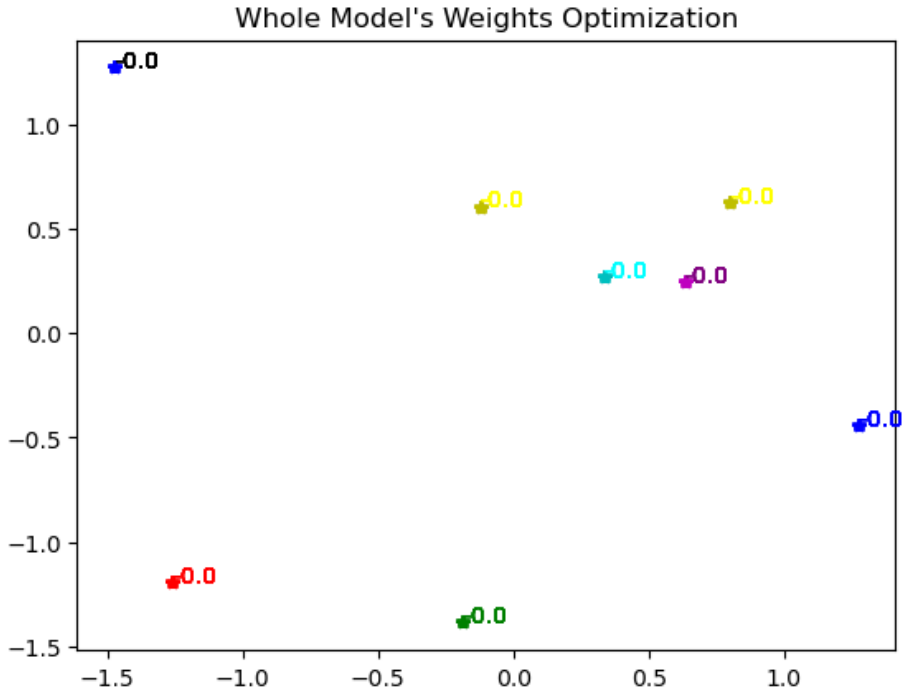


The PCA for the whole model is given in the **image (10).**

For network optimization, the Adam optimizer was utilized. The model's weights were periodically gathered during (8) separate training series, each with 30 training epochs. PCA adoption results in a decrease in the dimensions. The learning rate of all models is 0.001.

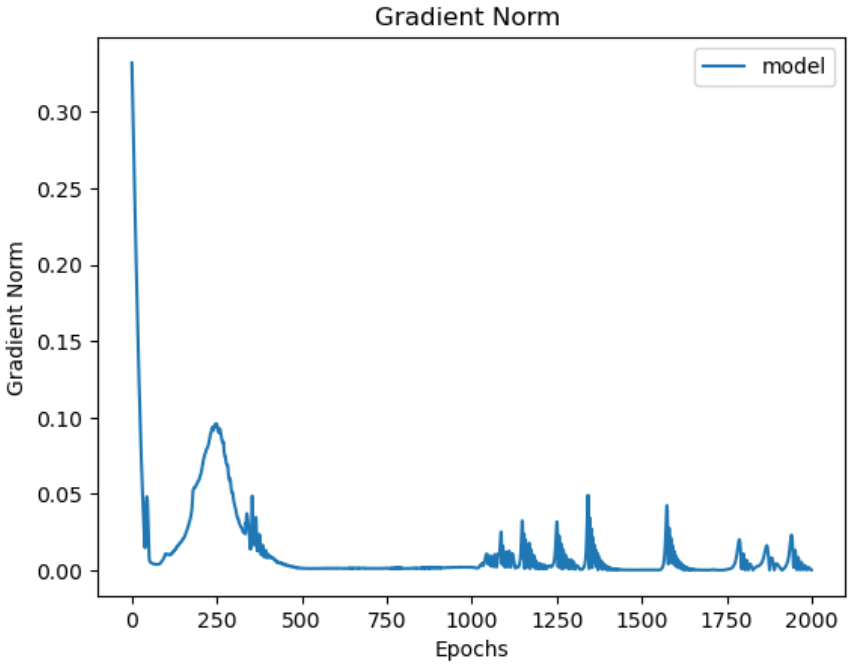
Both images illustrate the backpropagation weight optimization mechanism within the network that learns during the training period. The figures demonstrate the gradual fine-tuning as the weights are gradually optimized.

- Below image shows the whole model optimization. **(image 10)**

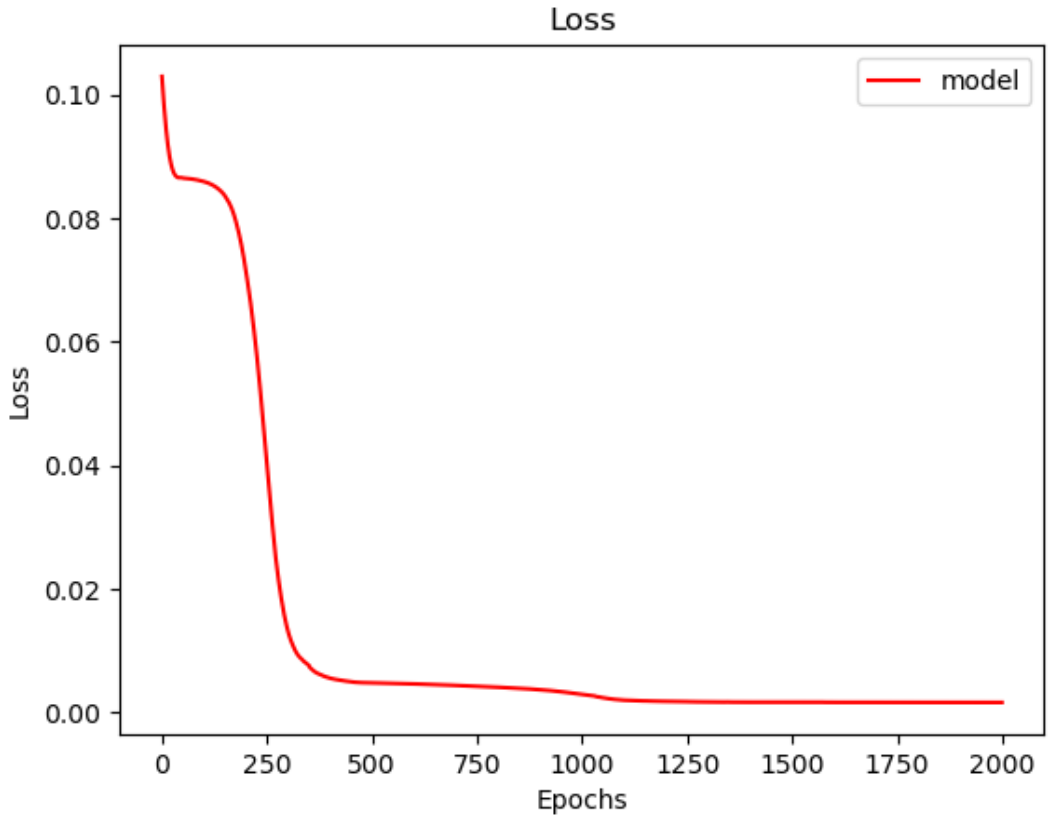


Task 2: Observe Gradient Norm during Training

- I have trained the model on epochs. The below image shows the gradient norm for each epoch**. (Image 11)**



- The model lost data after each epoch period given in the below image**. (image 12)**



Result:

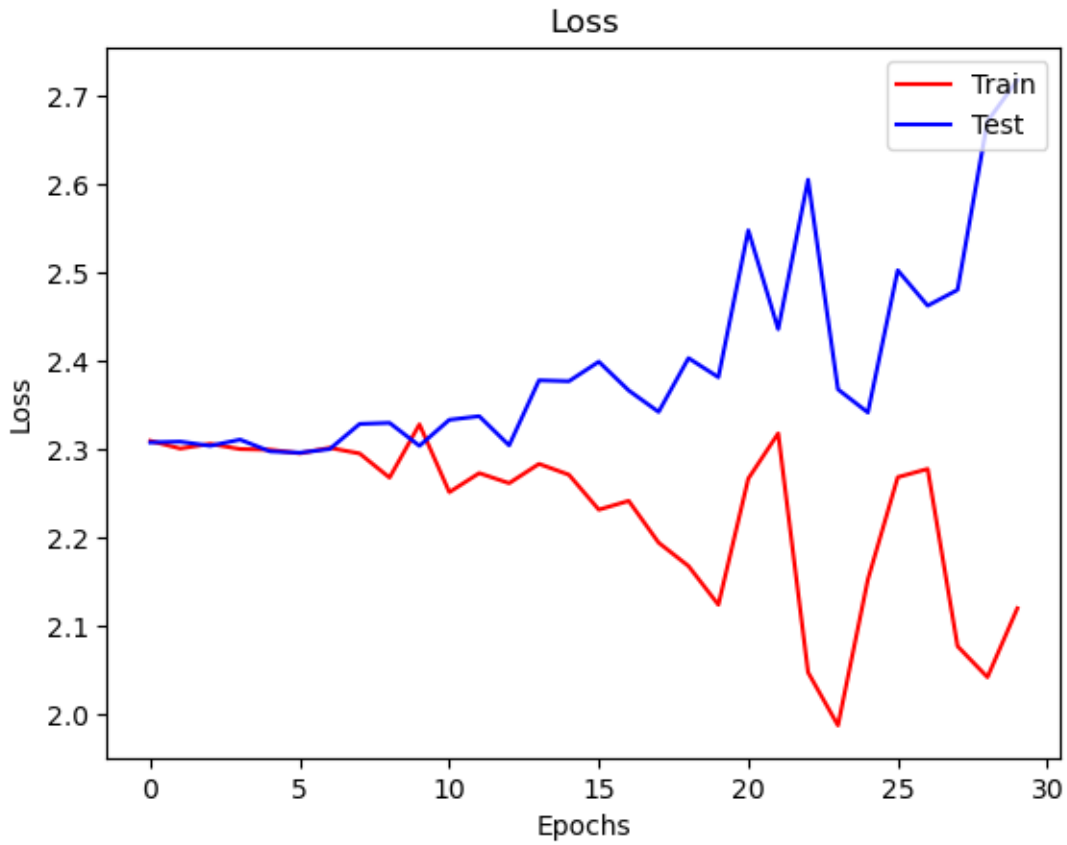
These models are trained and converged. In (image 11)There is a slight increase in gradient when it is at 250 epochs and simultaneously we can observe there is a loss in (image 12) in which the slope changes.

Generalization [Part 1]

Task 1-3 Can the network fit random labels.

The given models are trained on the MNIST dataset. The given Training Set is 60,000 and the Testing set: is 10,000**.** The learning rate given is 0.001. Using feed-forward Utilizing three hidden layers, a deep neural network was created. I used Adam optimizer for the optimization process for the network.

The below image was developed using random models. Since the model must learn from random labels and attempt to memorize them as we advance through the epochs to reduce the loss, the training process is laborious.



(**Image13)**

From the above (image 13), we can say that the implementation during training loss is very low and test data have a high loss when compared. This shows that cannot fit random labels.

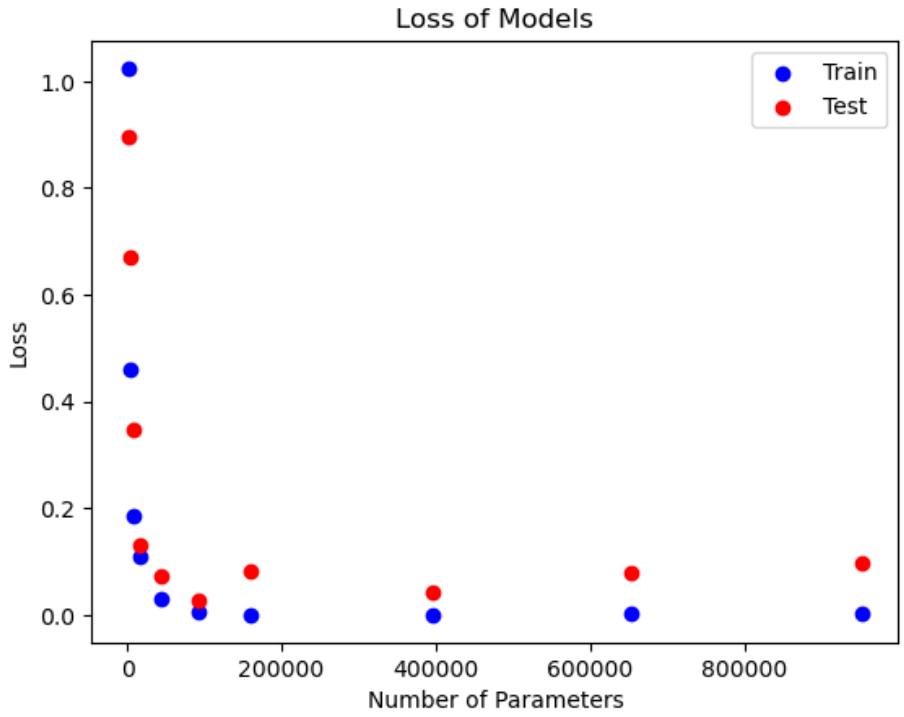
1-3 Parameters Vs Generalization

The given models are trained on the MNIST dataset. The given Training Set is 60,000 and the Testing set: is 10,000**.** The learning rate given is 0.001. I used Adam optimizer for the optimization process for the network. The models' parameter counts range from hundreds to millions.

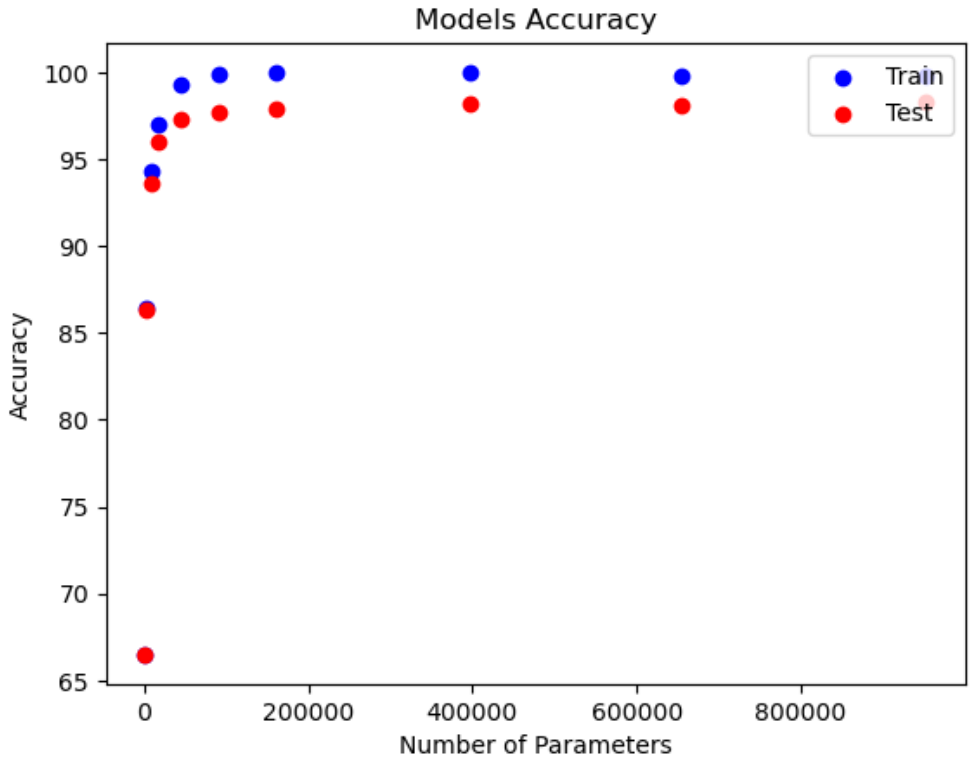
From the below images, it shows the disparity between the train and test loss and accuracy grows, as seen by the graphs above, as the number of parameters increases.

The model improvement from iteration to iteration is minimal after a certain number of parameters. The model scarcely benefits from the addition of new parameters.

[Image 14]



[Image 15]



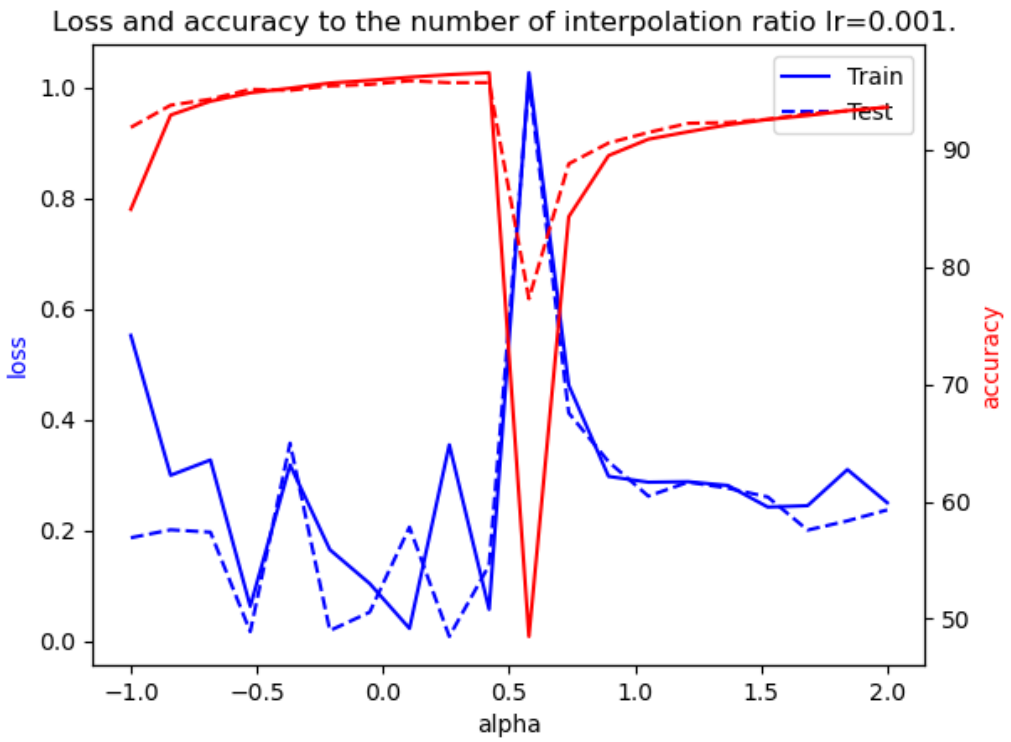
So in **[image 15**], we can say Running the models on the training dataset rather than the testing dataset results in improved accuracy and lower loss values for the model.

HW 1-3 Flatness vs Generalization [Part 1]

The given models are trained on the MNIST dataset. The given Training Set is 60,000 and the Testing set: is 10,000**.** The learning rate given is 0.001. I used Adam optimizer for the optimization process for the network.

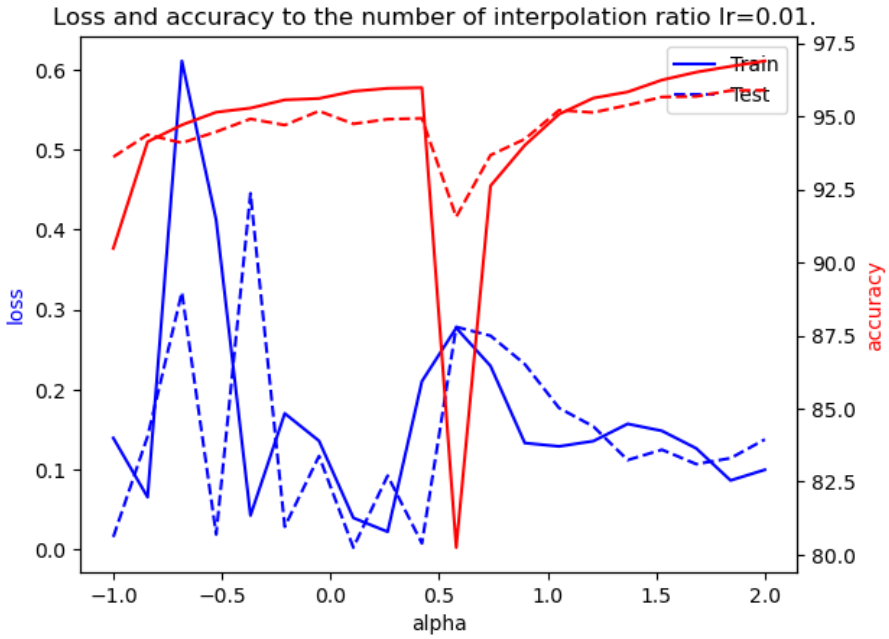
- I had taken the two different batch sizes of [64,1024]
- Alpha is the linear interpolation between theta1 and theta 2, theta1 is the model 1 parameters and theta 2 is the model2 parameters.
- The formula theta=(1-alpa[i])*model1params+alpa[i]*model2params
- **Image 16** depicts the loss and accuracy and also the linear interpolation ratio with a learning rate of 0.001

-[Image 16]



- **Now [Image 17]** shows the same loss and accuracy and also the linear interpolation ratio with a learning rate of 0.001.

[Image 17]

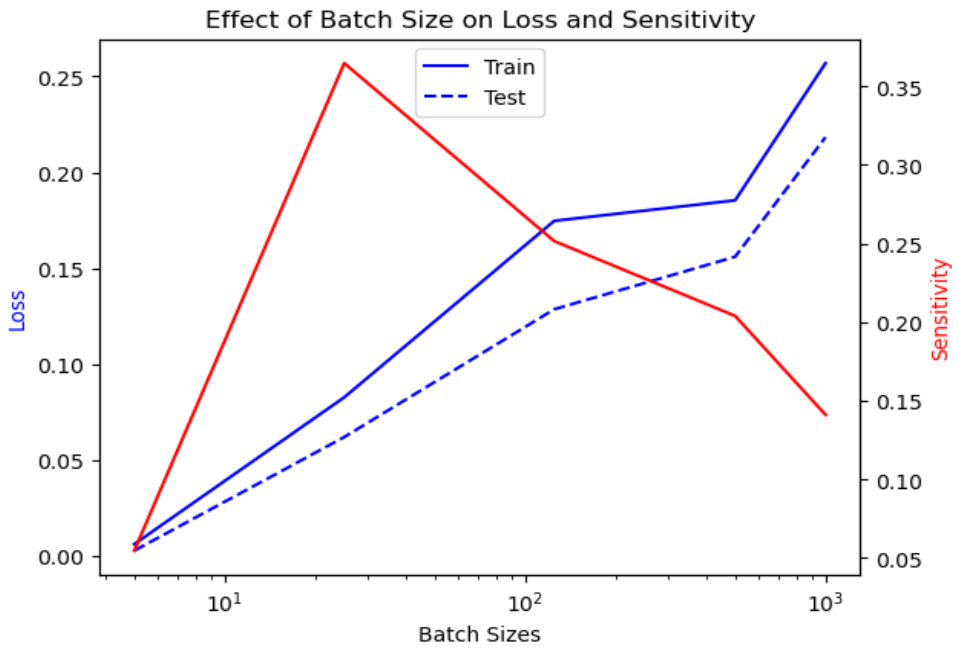


Flatness Vs Generalization – [Part 2]

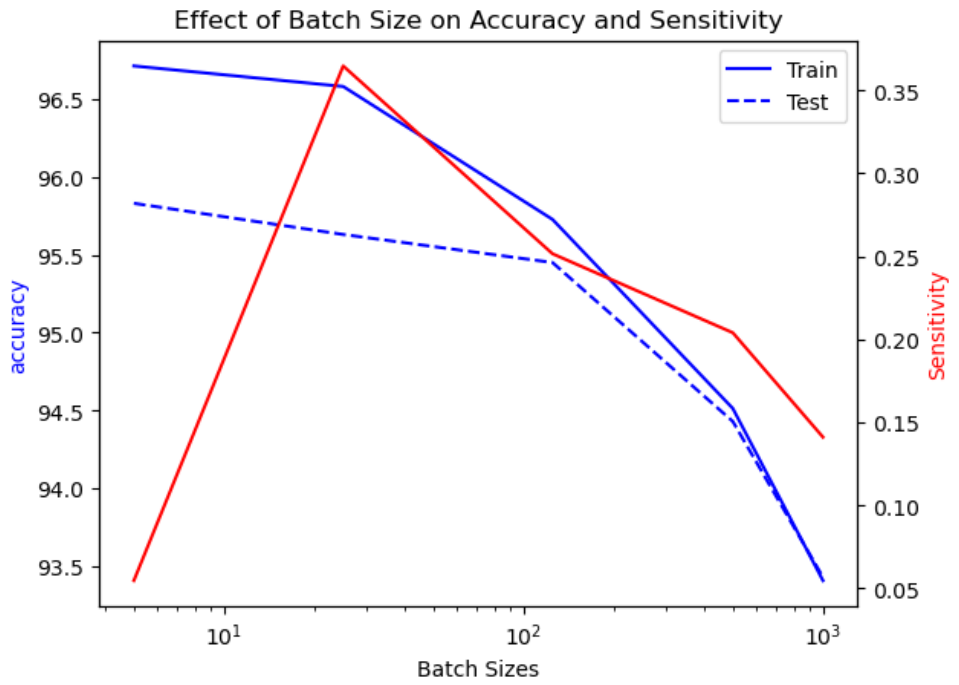
The given models are trained on the MNIST dataset. The given Training Set is 60,000 and the Testing set: is 10,000**.** The learning rate given is 0.001. I used Adam optimizer for the optimization process for the network.

- Different batches ranging in size from 5 to 1000 were trained using five identical Deep Neural Networks, each of which has two hidden layers.
- The accuracy and loss for the training dataset and test dataset for each of the five models were calculated after training. Then, using the Forbenius norm of gradient approach, the sensitivity of the models was determined.
- Below in [image 18] we can see the size of loss and sensitivity of batch sizes.
- Below in [image 19] we can see the accuracy and sensitivity of batch sizes.

[Image 18]



[Image 19]



Result-The above images tell that the Training and testing, accuracy and loss, and sensitivity can be seen in the best results between 10^ 1 and 10^3 of batch sizes.