Deep Learning Homework-1

Vinay Kumar Reddy Vallapu Reddy

github

Part-1:

Task1(Simulate a Function):

Models: I have used three models that are same as the one in the requirements.

Model-1: The model in question has 7 dense layers and employs the activation function 'Leaky_Relu'. It has 571 parameters and is optimized using the loss function "MSELoss" and the optimization algorithm "Adam". The hyperparameters of the model include a learning rate of 0.001 and weight decay of 0.0001.

Model-2:

The model in question has 4 dense layers and employs the activation function 'Leaky_Relu'. It has 572 parameters and is optimized using the loss function "MSELoss" and the optimization algorithm "Adam". The hyperparameters of the model include a learning rate of 0.001 and weight decay of 0.0001.

Model-3

The model in question has 1 dense layer and employs the activation function 'Leaky_Relu'. It has 571 parameters and is optimized using the loss function "MSELoss" and the optimization algorithm "Adam". The hyperparameters of the model include a learning rate of 0.001 and weight decay of 0.0001.

In all the above-mentioned models a regularization technique is used which reduce the complexity of model and prevent from overfitting.

All the above models are trained with maximum epochs 20000, where in train function will complete its execution if one of the conditions is satisfied.

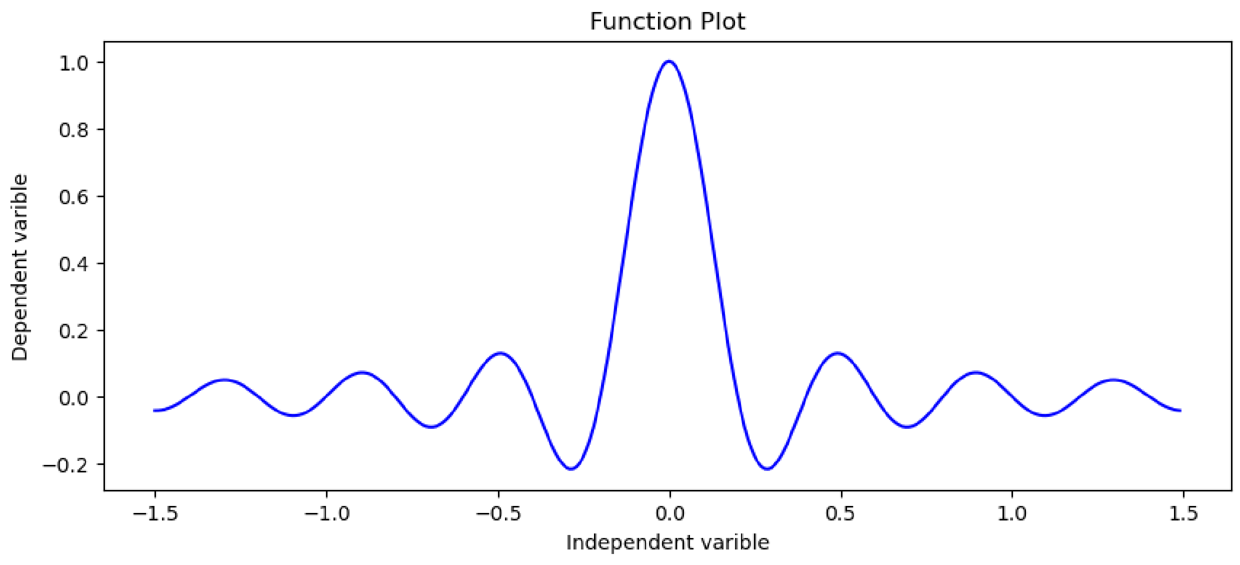
1.Maximum epochs reached

2.The model has reached the convergence(i.e., learning stopped)

Functions:

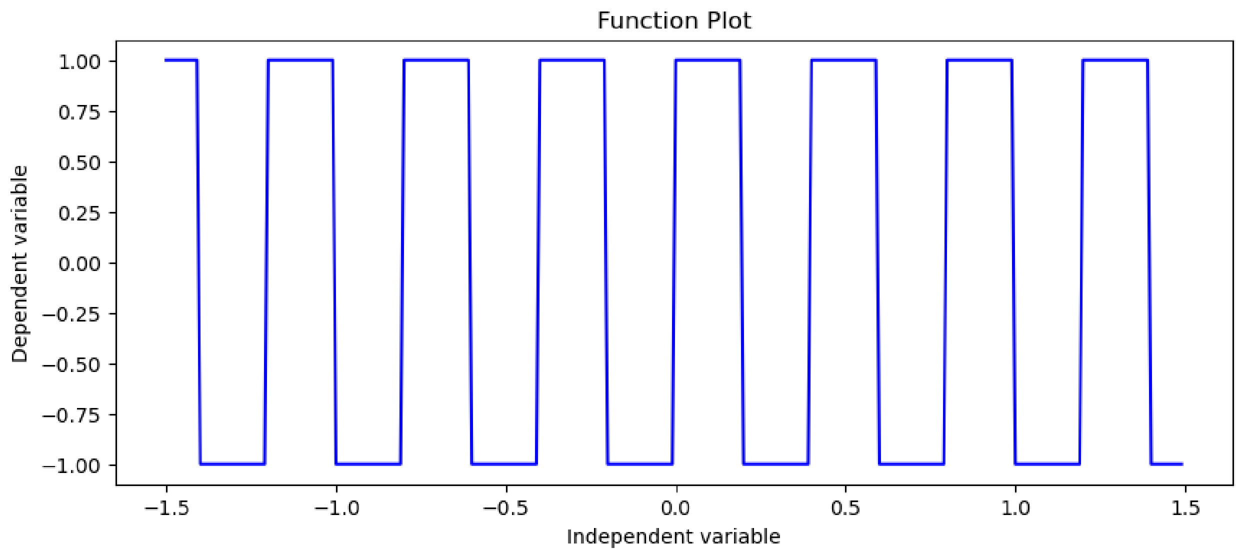
Function-1- sin(5*pi*x)/(5*pi*x)

Following is the plot for function 1

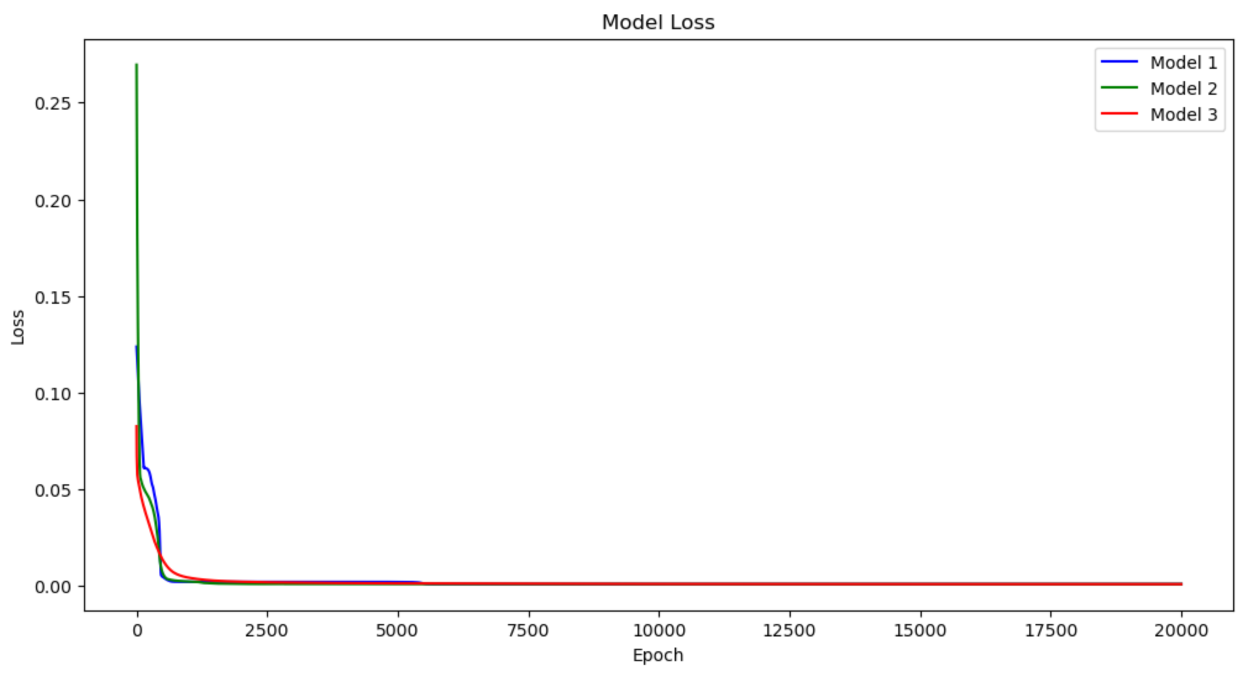


Function-2 – sgn(sin(5*pi*x)/5*pi*x)

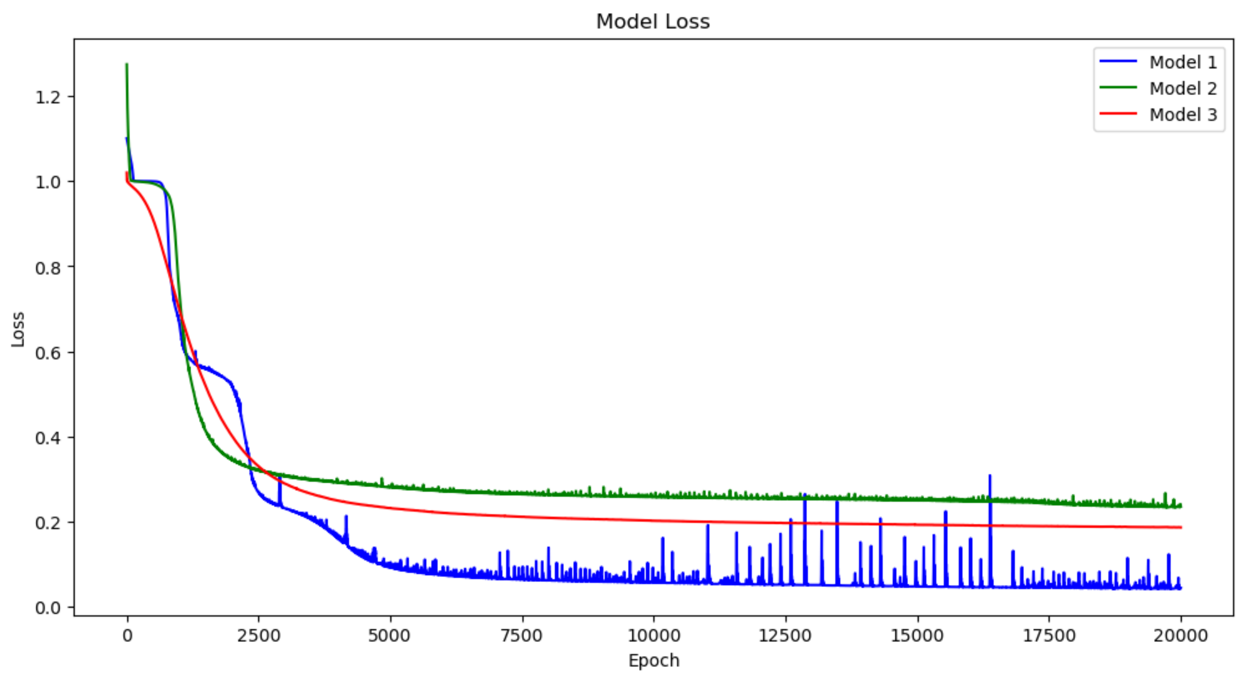
Following is the plot for the function 2.



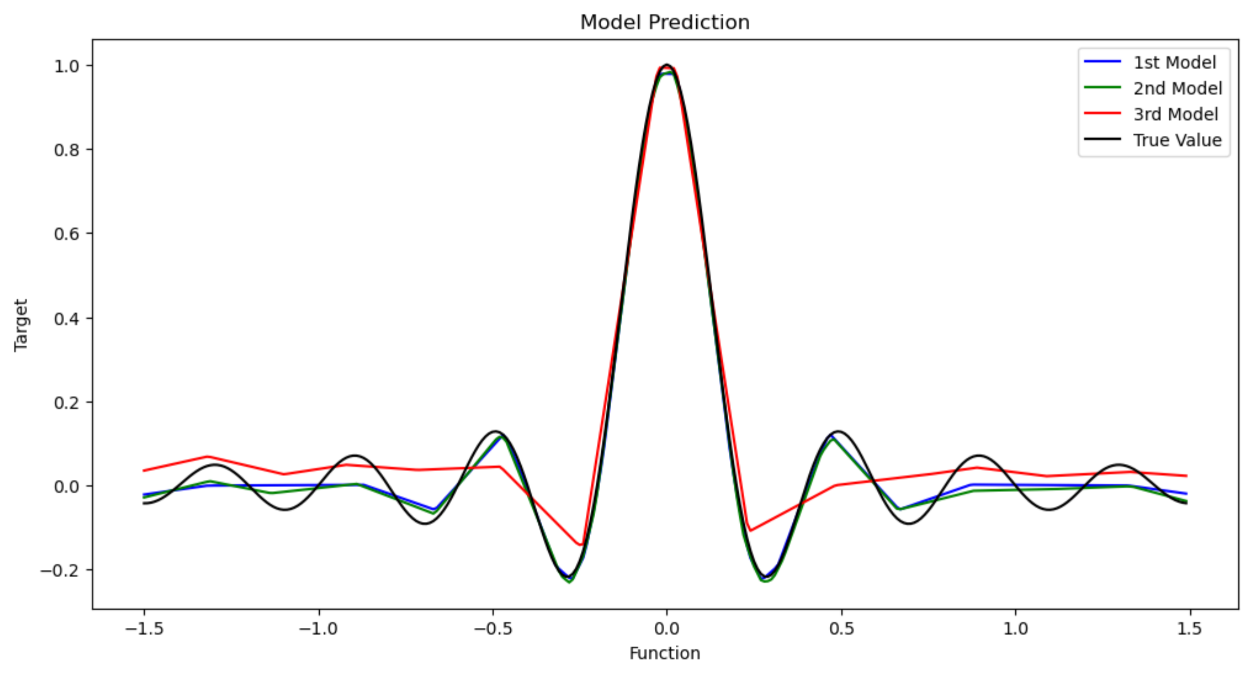
The following visualizes the training loss against the number of epochs for all the 3 models for the function 1:



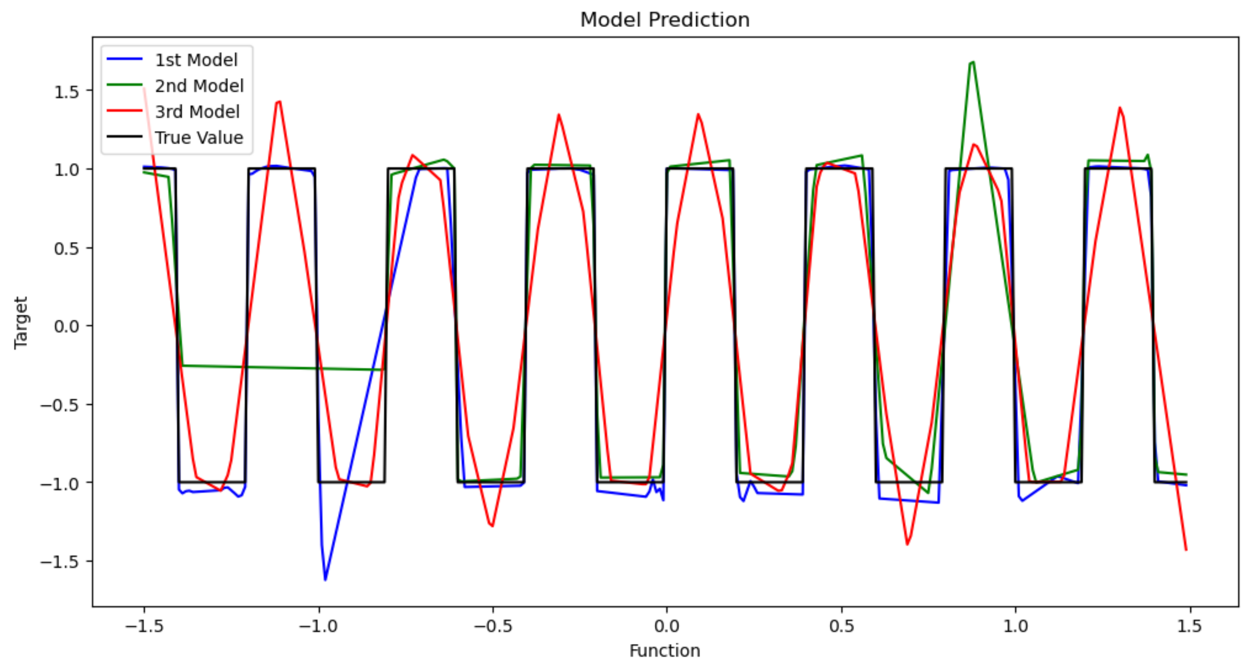
The following visualizes the training loss against the number of epochs for all the 3 models for the function 2:



The following visualizes the model prediction for all the 3 models compared to the ground truth for function 1.



The following visualizes the model prediction for all the 3 models compared to the ground truth for function 2.



Comments on results.

for the function 1:

All the three models reached convergence after maximum number of epochs (20000). Model 1 and Model 2 has low loss value and learned the function better than model 3. To learn faster and better models should have a greater number of layers. At the center of the graph models performed well, extreme ends of the x axis have some misclassifications. The deeper the model the lesser the misclassifications at the end of the axis.

For the function 2:

All the three models reached convergence after maximum number of epochs reached(20000). Model 1 has the lowest loss , slightly outperforming the model 2. Model 3 fails to reach low loss and to converge over the range of epochs. This is testing statement to the fact that a neural network with the more layers tends to learn better and converge quicker.

Task 2 (Train on Actual Tasks):

The below models are trained on MNIST datasets.

Training set:60000 and test set : 10000

Training data is shuffled whereas test data is not shuffled .

Convolution Neural Network -1:

This model consists of a 2D convolution layer with a kernel size of 4, followed by a max pooling layer with a size of 2. The model has 2 dense layers and uses the "Relu" activation function. The loss function used is "CrossEntropyLoss", and the optimization algorithm is "Adam". The learning rate is set to 0.001 and the weight decay to 0.0001, and the model also includes a dropout of 0.25.

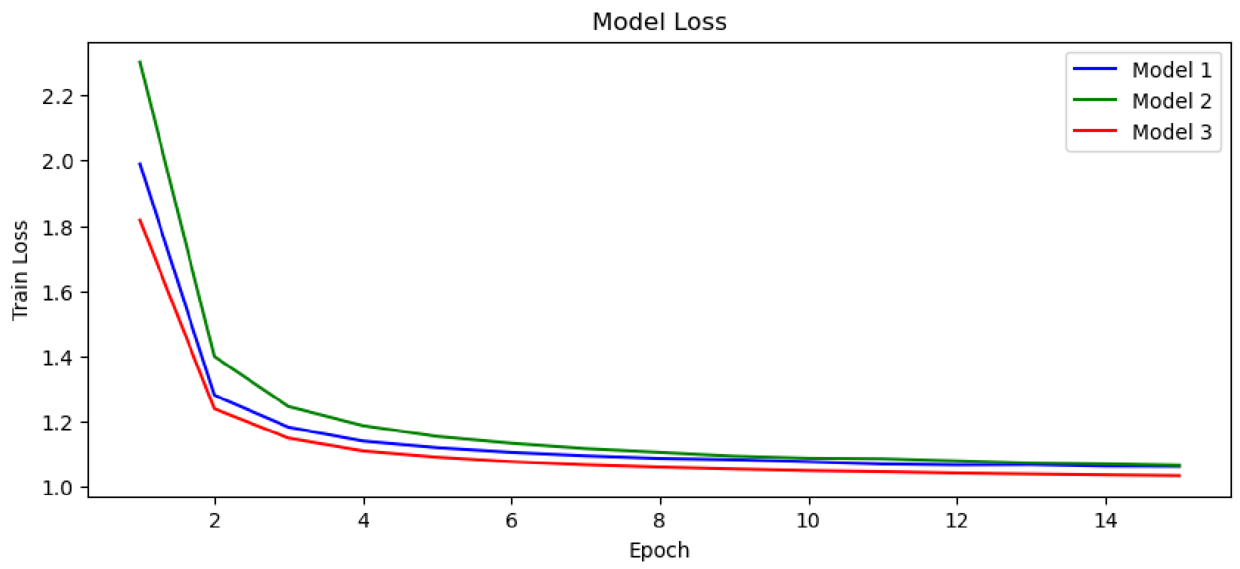
Convolution Neural Network-2:

This model consists of a 2D convolution layer with a kernel size of 4, followed by a max pooling layer with a size of 2. The model has 4 dense layers and uses the "Relu" activation function. The loss function used is "CrossEntropyLoss", and the optimization algorithm is "Adam". The learning rate is set to 0.001 and the weight decay to 0.0001, and the model also includes a dropout of 0.25.

Convolution Neural Network -3:

This model consists of a 2D convolution layer with a kernel size of 4, followed by a max pooling layer with a size of 2. The model has 1 dense layer and uses the "Relu" activation function. The loss function used is "CrossEntropyLoss", and the optimization algorithm is "Adam". The learning rate is set to 0.001 and the weight decay to 0.0001, and the model also includes a dropout of 0.25.

The following visualizes the training loss for all the three models against epochs:



The following visualizes the accuracy for all the three models against epochs:

Observations:

From the above graphs it can be observed that model 3 even it has only one layer performed well for both training and testing data and had lowest loss value as compared to the other 2.

For training data:

Model-1 Training Accuracy=98.13% and loss during =0.0677

Model-2: Training Accuracy=97.9% and loss during= 0.0845

Model-3: Training Accuracy=98.97% and loss during= 0,0553

For testing data:

Model -1 Test Accuracy=98.79%

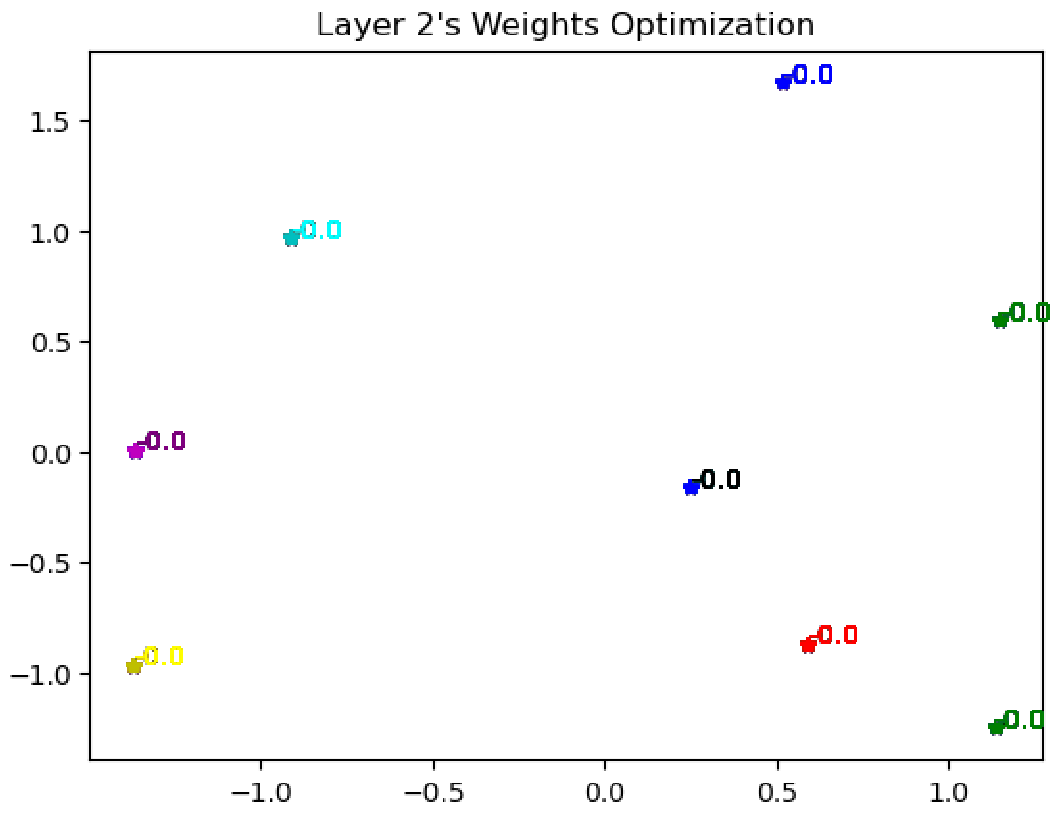
Model-2 Test Accuracy=98.19%

Model-3 Test Accuracy=99%

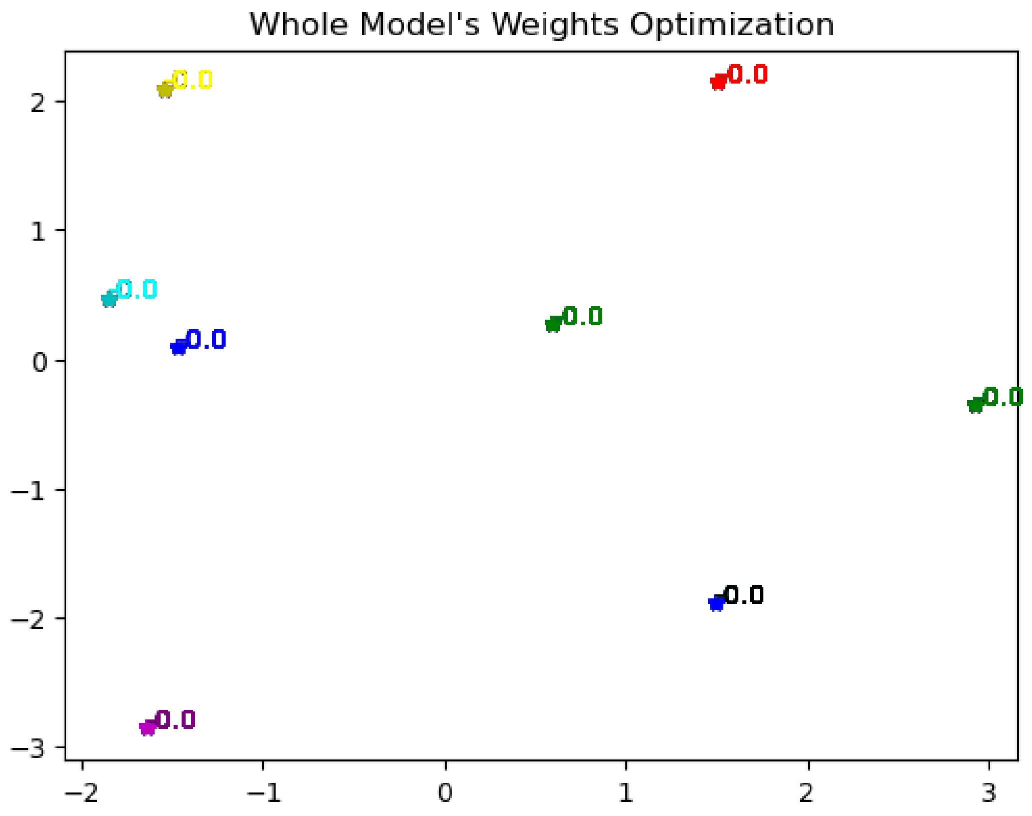
- Have used more than two models in the previous question
- Have used more than one function in the previous question

Part-2-1: Visualize the optimization process.

A deep neural network with three fully connected layers and 701 parameters was trained to replicate the function sgn(sin(5*pi*x)/5*pi*x) using cross-entropy loss as the loss function and Adam optimization algorithm with a learning rate of 0.001. Eight training series, each lasting 300 epochs, were carried out and the collected weights were subjected to dimension reduction through PCA. The aim was to accurately replicate the function and the dimension reduction helped in achieving this goal.



Layer 2's Weights Optimization
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

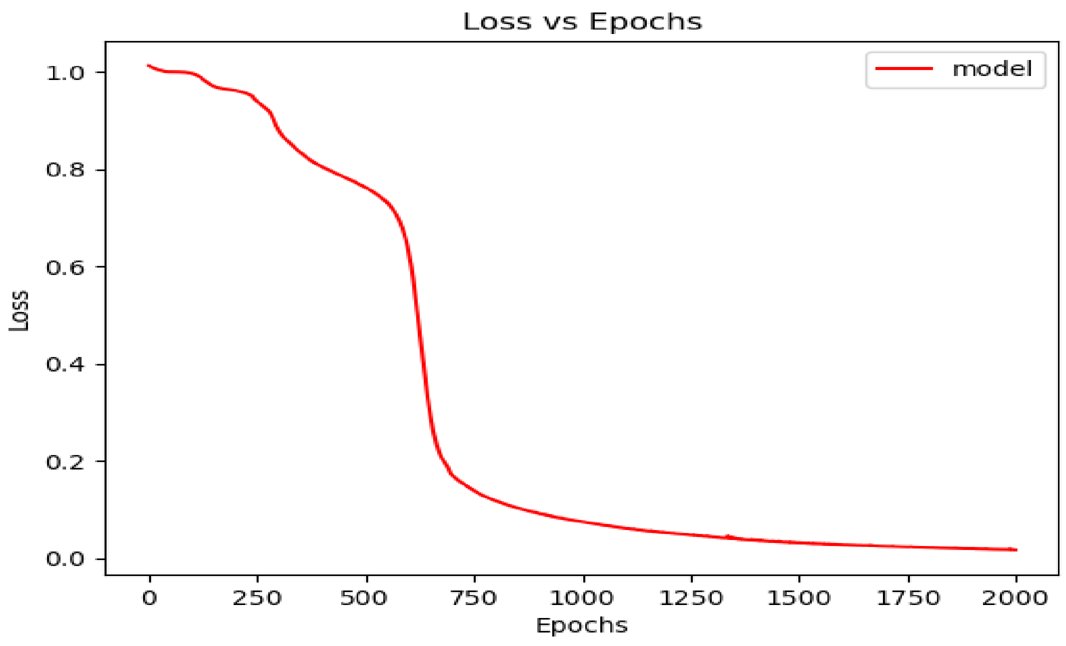


Whole Model's Weights Optimization
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

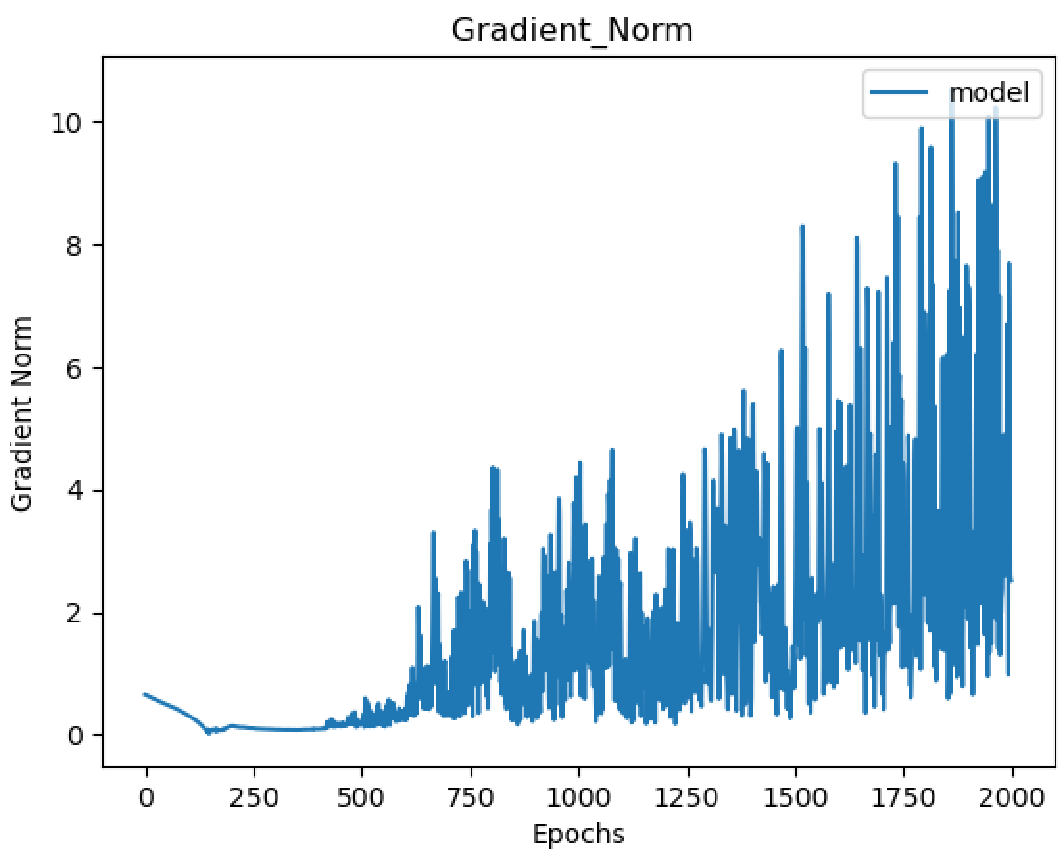
Training algorithm is used to depict the optimization process of a deep neural network and the incremental fine-tuning of the weights over time. The numbers underscore the necessity of being patient during the training process and giving adequate time for the network to learn. This emphasizes the need of enabling the network to constantly change its settings in order to attain maximum performance and accuracy.

Part-2-task-2 : Gradient Norm during training .

The following graph illustrates the loss during training against epochs.



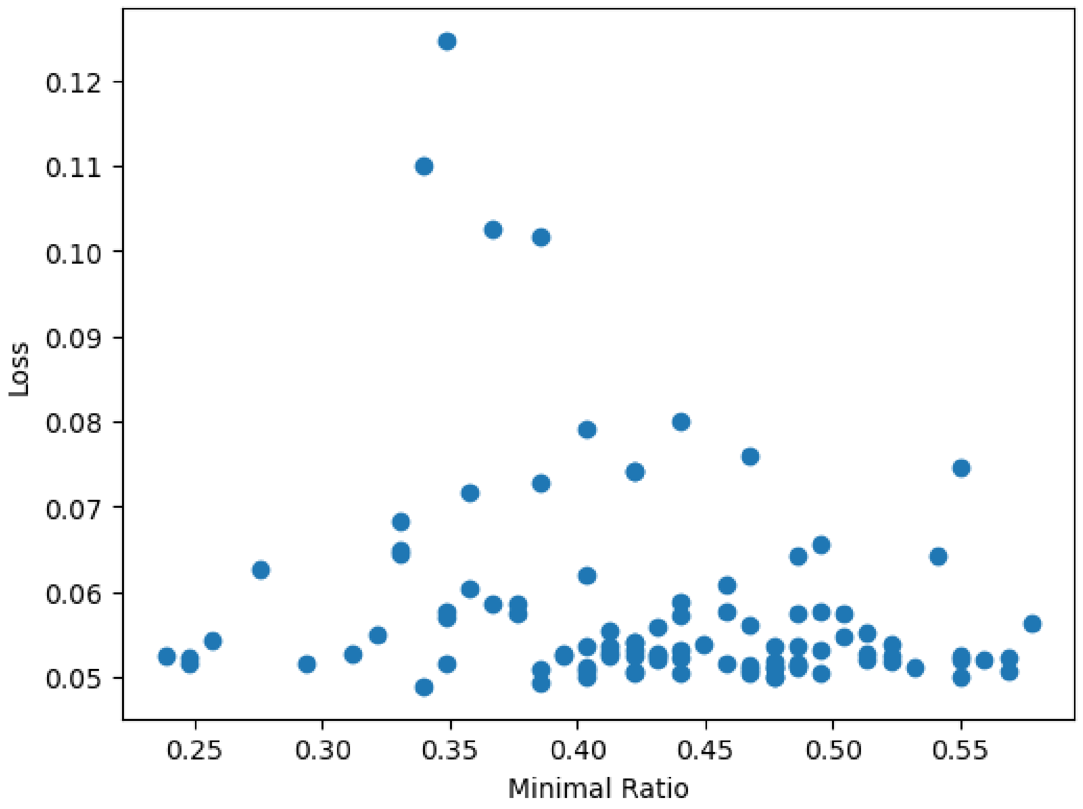
The following visualizes gradient norm against epochs or iterations



The function I used is sgn(sin(5*pi*x)/5*pi*x) .

The model is trained until it reached convergence at 2000 . initially the loss were very high and decreasing after reaching around 550 to 600 the loss is almost became minimum. There gradient norm is increasing after reaching 500 epochs and maximum when it is model is converged

What happens when gradient norm is zero.



A function's gradient approaching zero indicates that its rate of change in that direction is minimal, resulting in a flat slope at that point. This is crucial in optimization methods such as gradient descent, as a gradient close to zero signals that the algorithm is near a minimum or maximum of the function.

Part -3 :

Task-1 : Can Network fits Random Variables:

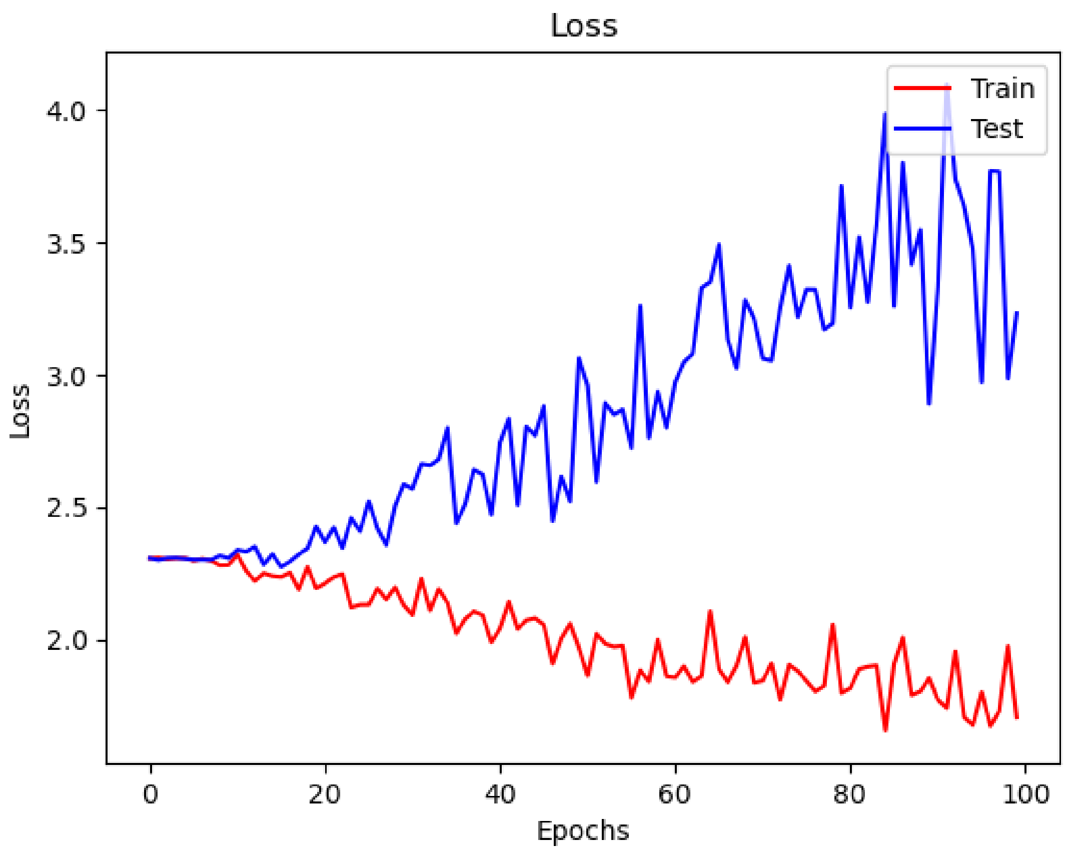
In these tasks I have trained feed forward DNN against MINST data

Training set: 60000 and test data set: 10000

Details of model I have used :

- Dense layers : 3
- Activation function: “Relu”
- Loss function: “Cross Entropy Loss”
- Optimizer function: “Adam”
- Hyperparameters.
  Learning rate =0.001

The model mentioned above is based on random models. The model must learn from random labels, and as we progress through the epochs and try to lower the loss, it tries to memorize the labels, which slows down the training process. The train loss gradually declines as the epochs rise while the test loss keeps rising. So we cant fit the random variables.



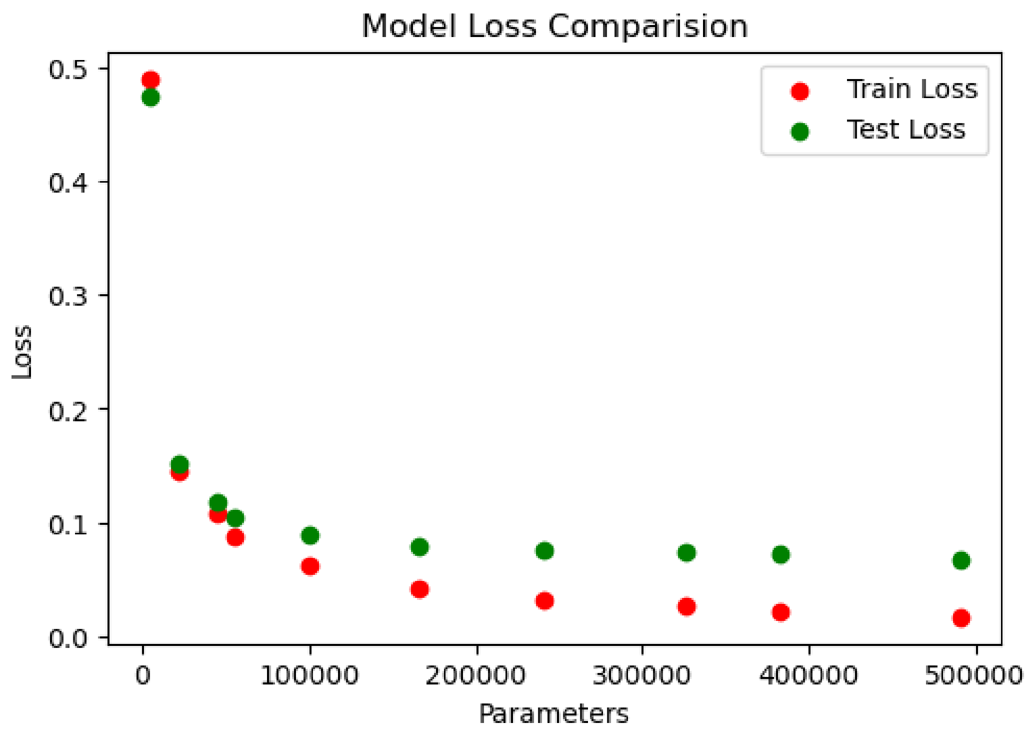
Task 2: Number of parameters vs. Generalization

I have created 10 feed forward deep neural network models having the same structure but having different dense layers such that the parameters numbers of the models have the following range : 4079,21435,45360,55630,100710,166060,241410,326760,382770,491360.

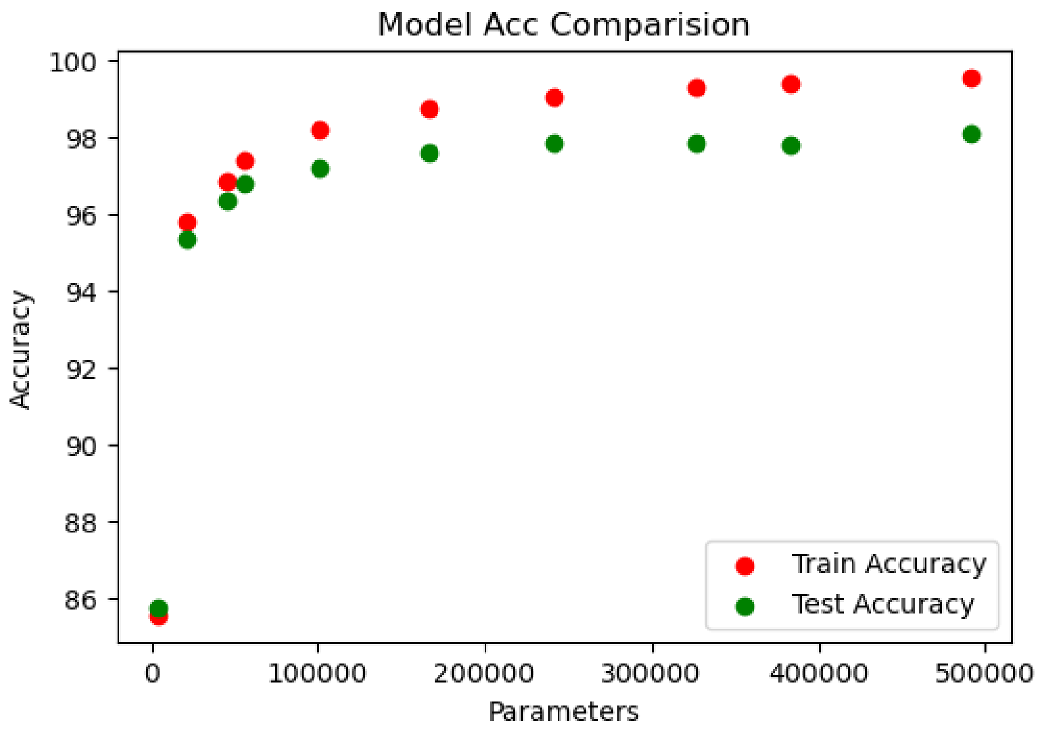
Details of the model I have used :

- Number of dense layers = 1.
- Activation function : "Relu"
- Loss Function: "CrossEntropyLoss"
- Optimizer : "Adam"
- Hyper-parameters:
  Learning rate : 0.001
  Dropout=0.25

The following visualizes the Loss comparisons of both test and training against parameters.



The following visualizes the accuracy of test and training of all models against parameters.



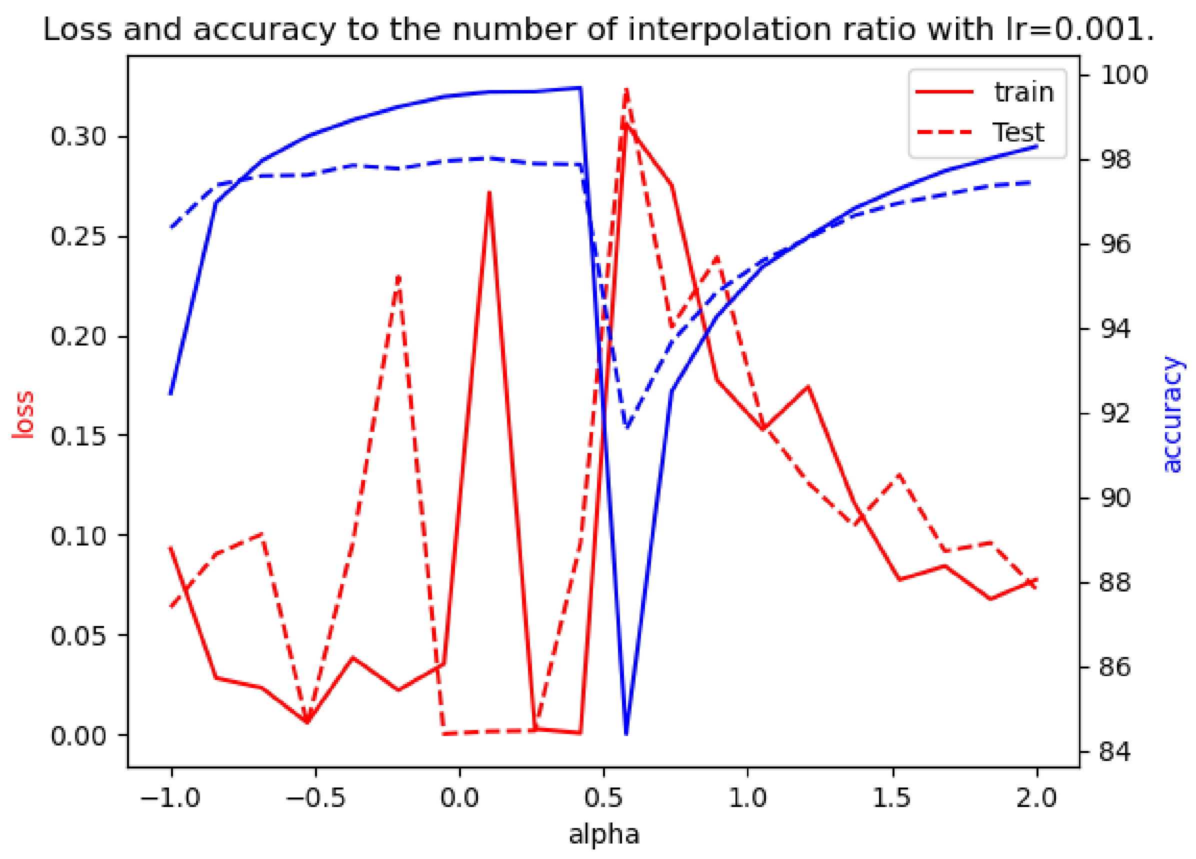
Results : if we observe both figures we can clearly see that as the number of parameters increases decreases the model's loss and increases the accuracy. However after increasing particular number of parameters the model improvement from iteration to iteration is almost negligible further more increasing in the number of parameters improves the model performance barely. Also, higher accuracy and low loss are obtained when models are executed against training data compared to testing data . This is due to overfitting as there are a greater number of parameters for the model to train.

Task-3: Flatness vs. generalization – part-1

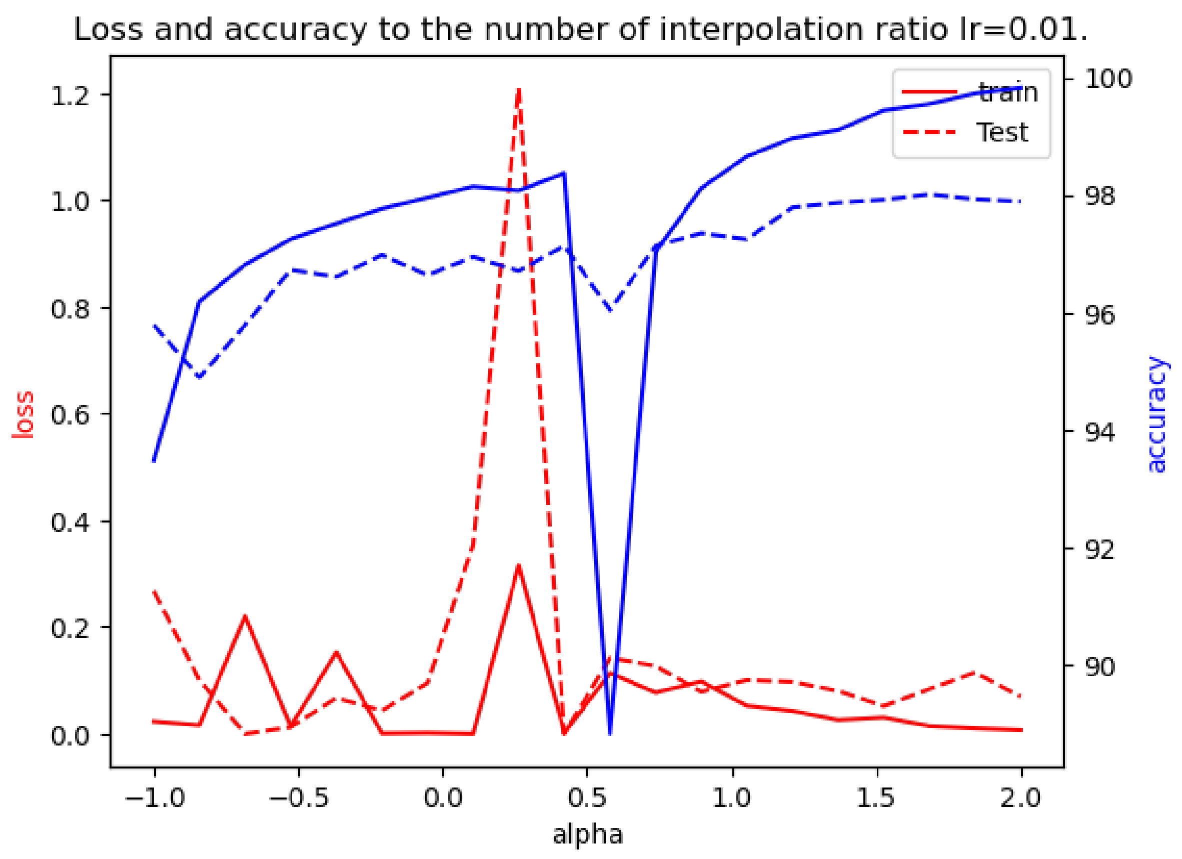
The MINST data was chosen as the training and testing data. Two feed forward deep neural networks were implemented with two different batch sizes (64 and 1024). Adam optimizer is used for optimization and “CrossEntropyLoss” is used as to calculate loss for the neural network. Activation function “Relu” is used . Alpha is the interpolation ratio and theta is the parameter of the model(i.e., theta1 is the parameter of the model1 and theta2 is the parameter of the model 2)

In this I have used “(1-aplha)*batch1_para+alpha*batch2_para” to calculate the interpolation ratio.

The following visualizes the loss and accuracy against the alpha during the training of two models with learning rate = 0.001



The following visualizes the loss and accuracy against the alpha during the training of two models with learning rate = 0.001



Observations:

We can see that for both models with varying learning rates, accuracy in both the test and the train drops off before 0.5. After 0.5, the accuracy starts to rise sharply, and at this point, both the train and test lose less. Thus, interpolation between datapoints is a term used to describe machine learning techniques. They will start to memorize the data and interpolate between it, though, if there are more parameters than data.

Task-4: Flatness vs Generalization part-2:

In these tasks I have created feed forward deep neural network with 5 different batch sizes

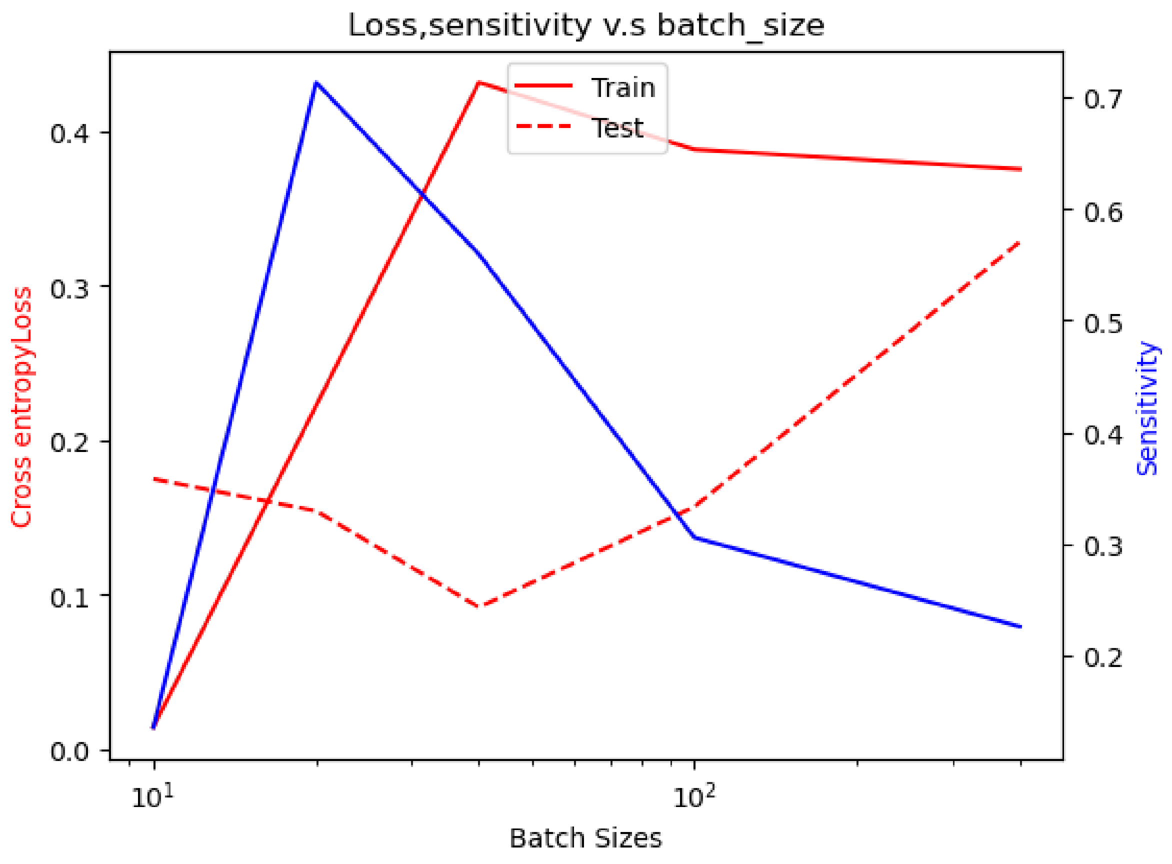
[10,20,40,100,400]. Following are the model details.

- 1 dense layer
- Activation function : “Relu”
- Loss function : “CrossEntropyLoss”
- Optimizer: “Adam”
- Max Epoch : 10

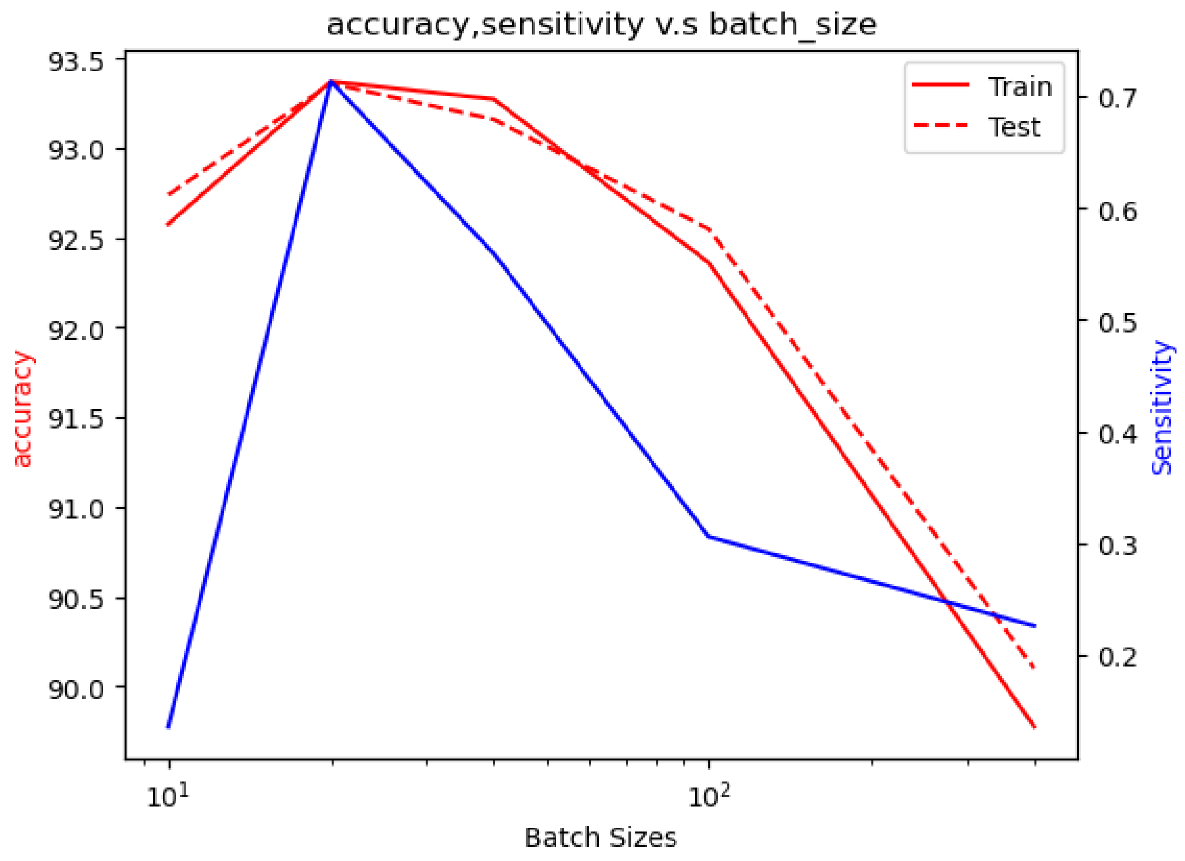
And also, I have used different learning rate for the five models.

To assess the effects of batch size on the learning of the models, we will calculate each model's sensitivity along with its loss and accuracy values.

The following visualizes the loss, sensitivity against the batch size.



The following visualizes the accuracy, sensitivity vs. batch size.



The graphs above show that the sensitivity decreases as batch size increases. The optimum batch size is between 10 and 100, after which sensitivity decreases as accuracy decreases and loss value increases. We can conclude that as the batch size exceeds 100, the network becomes less responsive.