Name: Sravani Pati
Email: spati@g.clemson.edu

Course: 8430
Homework 1

GitHub link: https://github.com/sravani919/Deep-Learning-8430-HW1

HW 1-1 Simulate a Function

Models that have been trained, including details on the number of parameters and the precise tasks they are intended to carry out are shown and at least two models are included.

Models: I have produced 3 models that adhere to the given guidelines. I have increased regularization by including a weight decay parameter in the neural network's cost function. As a result, the weights are decreased throughout the backpropagation process.

Model 1:

- Comprising of 7 dense layers with a total of 571 Parameters.
- Utilizing MSELoss as the Loss Function.
- Employing RMSProp as the Optimizer.
- Adopting a Learning Rate of 1e-3.
- Featuring leakyRelu as the Activation Function.
- Incorporating a weight decay of 1e-4.

Model 2:

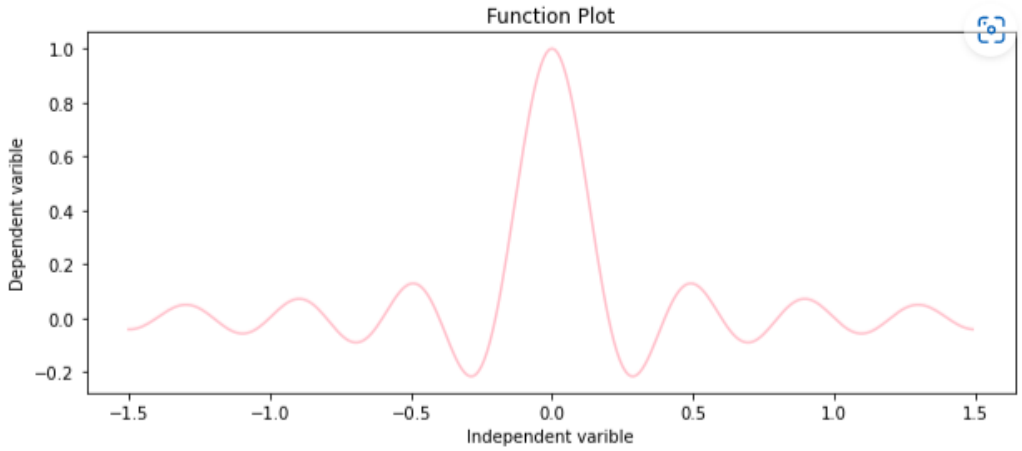
- Comprising of 4 dense layers with a total of 571 Parameters.
- Utilizing MSELoss as the Loss Function.
- Employing RMSProp as the Optimizer.
- Adopting a Learning Rate of 1e-3.
- Featuring leakyRelu as the Activation Function.
- Incorporating a weight decay of 1e-4.

Model 3:

- Comprising of 1 dense layers with a total of 571 Parameters.
- Utilizing MSELoss as the Loss Function.
- Employing RMSProp as the Optimizer.
- Adopting a Learning Rate of 1e-3.
- Featuring leakyRelu as the Activation Function.
- Incorporating a weight decay of 1e-4.

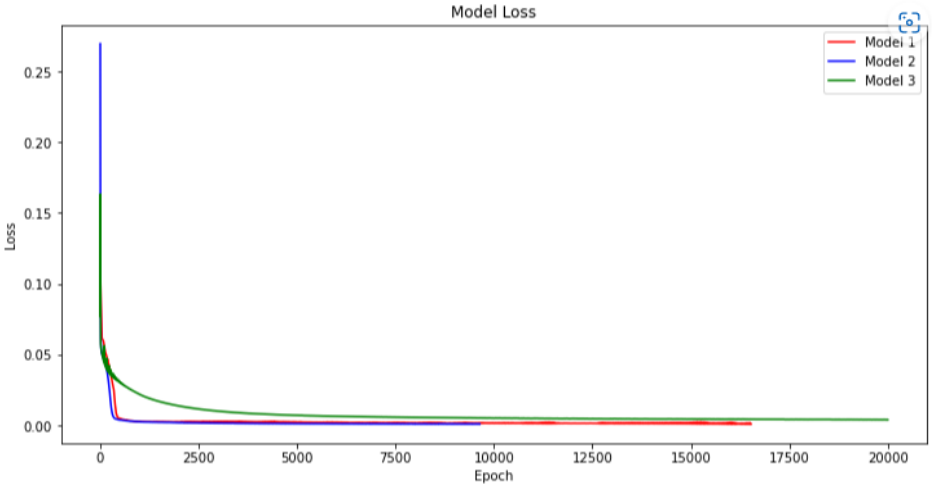
Function 1

The mathematical expression used for this function is sin(5pix)/(5pix).
Graphical representation for this function 1 is as follows:
Here we are plotting function for independent variable and dependent variable for function1.

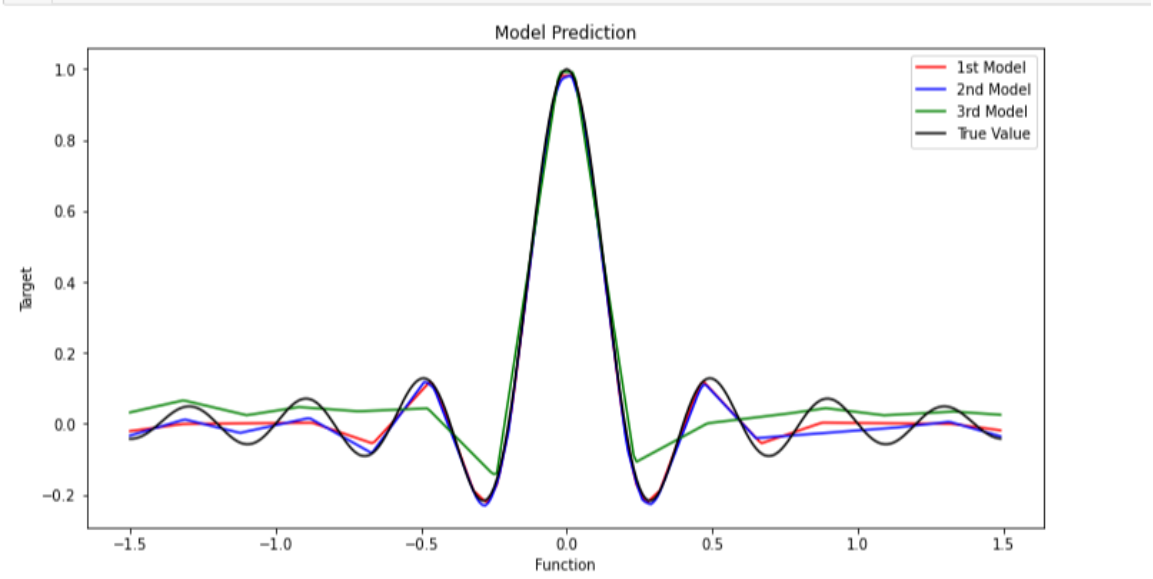


Function Simulation:

All models have a tendency to converge once the maximum number of epochs has been achieved or when the model's learning rate starts to dramatically slow down. The loss for each of the models is shown in the accompanying graph.The below Picture take epoch values
In x-axis and loss is represented in y-axis.Overall, It shows loss function simulation of the three models.



The chart below depicts the comparison between the actual values (Ground Truth) and the values predicted by the three models.

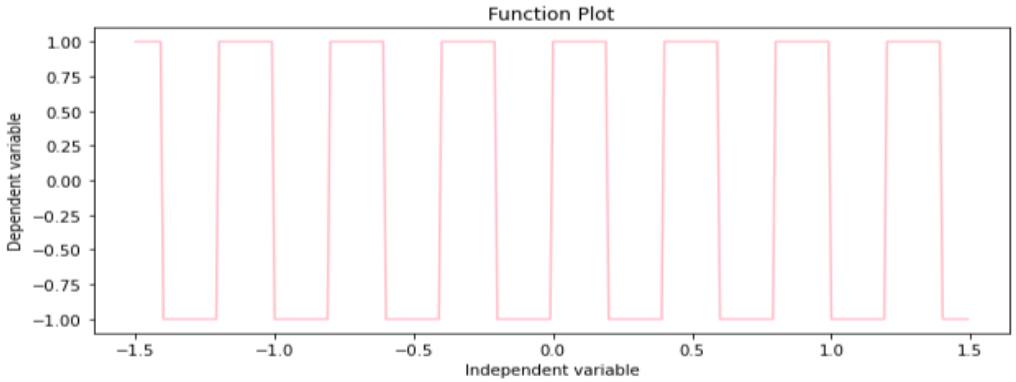


Results:

The findings suggest that Models 1 and 2 converge more quickly than Model 3, which does not converge until it has run its maximum number of epochs. The graph shows that Models 1 and 2 have more layers than Model 3, which results in a lower loss value and higher performance when learning the function. This emphasizes how having more layers is crucial for increasing learning pace and accuracy.

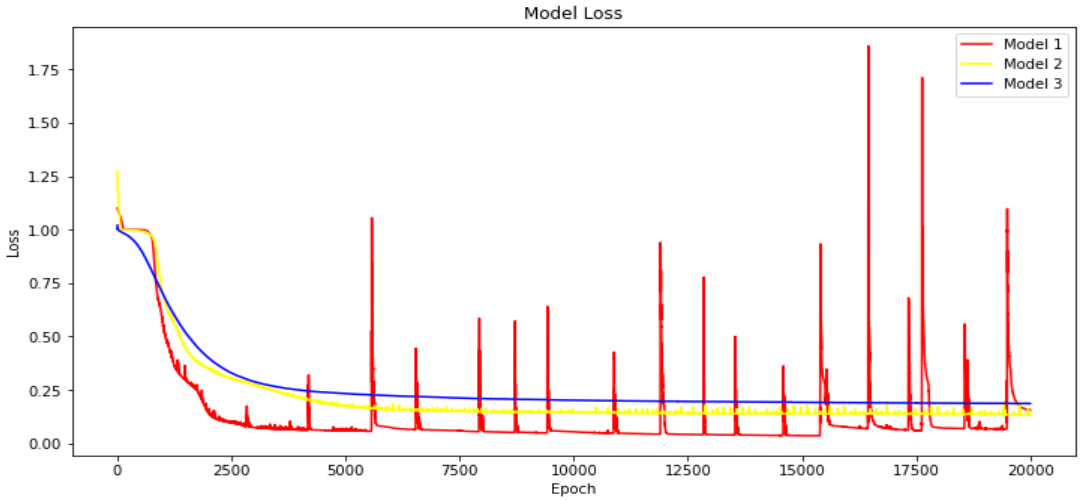
Function 2

The mathematical expression used for this function is sgn(sin(5*pi*x) / 5*pi*x).
Graphical representation for this function 2 is as follows:
Here we are plotting function for independent variable and dependent variable for function2.

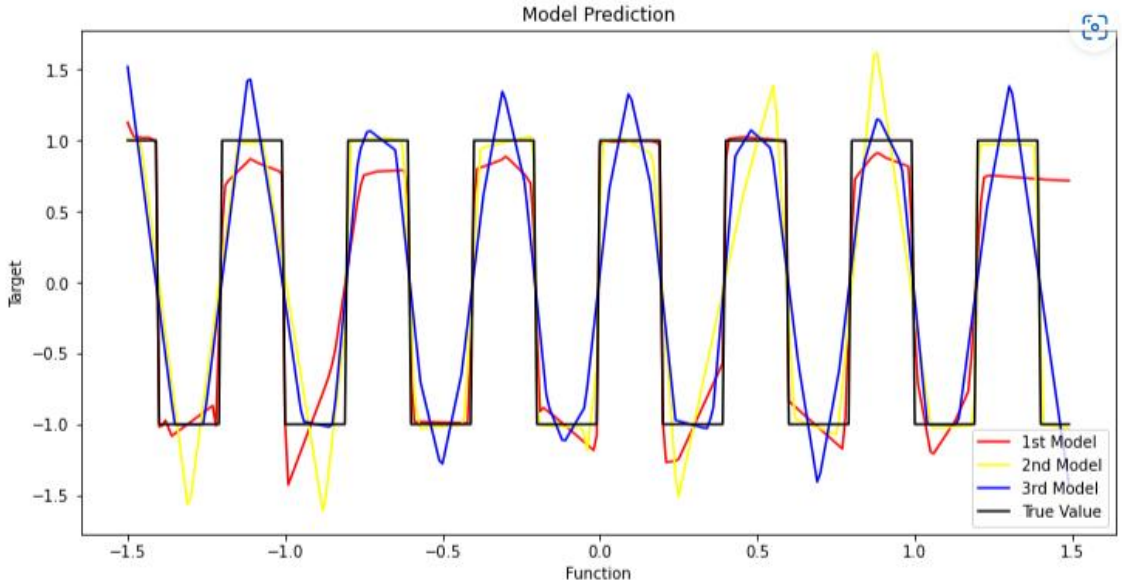


Function Simulation:

All models reach convergence either after reaching the maximum number of epochs or when the learning rate slows down significantly.
The following is a graph that illustrates the loss of each of these models:
The loss for each of the models is shown in the accompanying graph.The below Picture take epoch values in x-axis and loss is represented in y-axis.Overall, It shows loss function simulation of the three models.



The chart below depicts the comparison between the actual values (Ground Truth) and the values predicted by the three models.



Results:

All of these models attain the maximum number of epochs prior to convergence as the function presents a challenging learning task for the three models.Out of the models , Model 1 demonstrates the lowest loss and appears to be the most effective learner, slightly outperforming Model2. However, Model3 fails to achieve a low loss and does not converge within the range of epochs . This serves as further evidence that neural networks with more layers tend to learn better and converge faster.

HW 1-1 Training Actual Tasks

The following models have been trained on the MNIST dataset, containing a training set of 60,000 examples and 10,000 examples in the testing set.

Model 1: A Convolutional Neural Network (LeNet) architecture:

- Two 2D convolution layers, each followed by a 2D max pooling activation function and a ReLU activation function
- Two 2D dense layers, both using ReLU activation

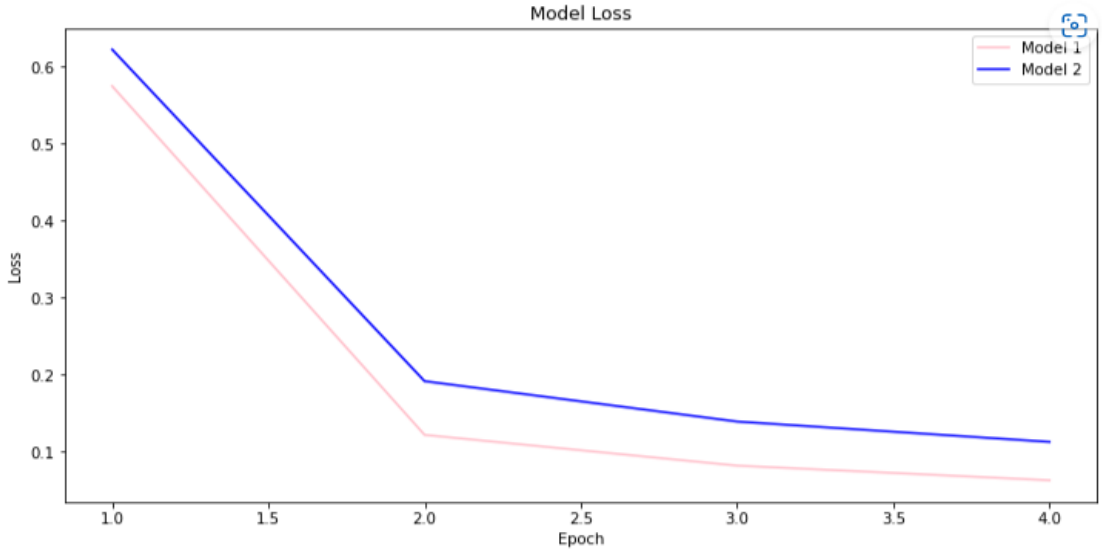
Model 2: A custom Convolutional Neural Network with the following structure:

- Two 2D convolution layers, both using ReLU activation
- Two 2D convolution layers, each applying a 2D max pooling activation followed by a ReLU activation and a dropout
- Two 2D dense layers, the first using ReLU activation followed by dropout, the second using a Log_softmax activation (Output)

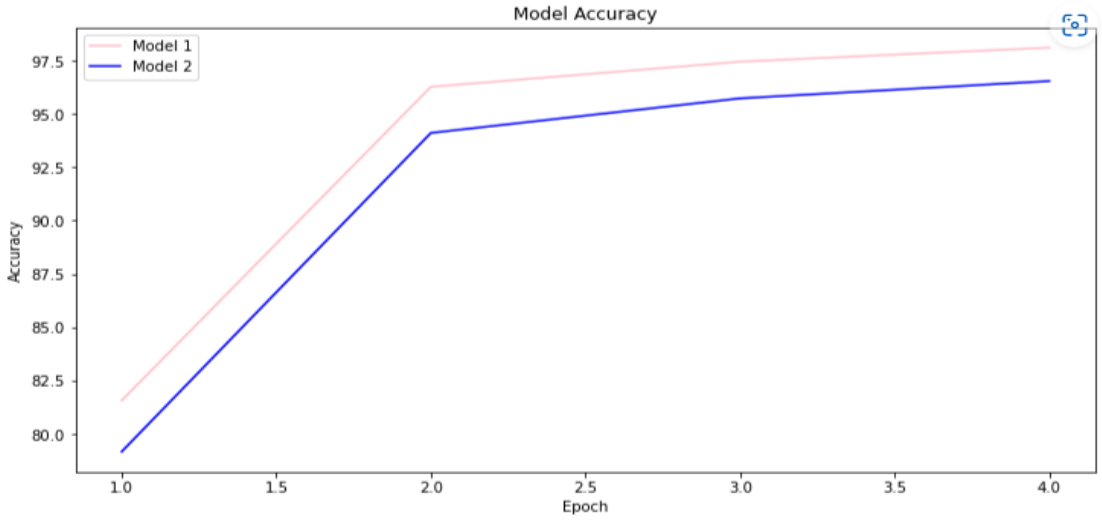
The following hyperparameters were used in the training process:

- Learning rate: 0.01
- Momentum: 0.5
- Optimizer: Stochastic Gradient Descent
- Batch size: 64
- Number of epochs: 10
- Loss function: Cross Entropy

The following is the training loss for Model1 and Model 2.



The following is the training accuracy for Model 1 and Model 2.



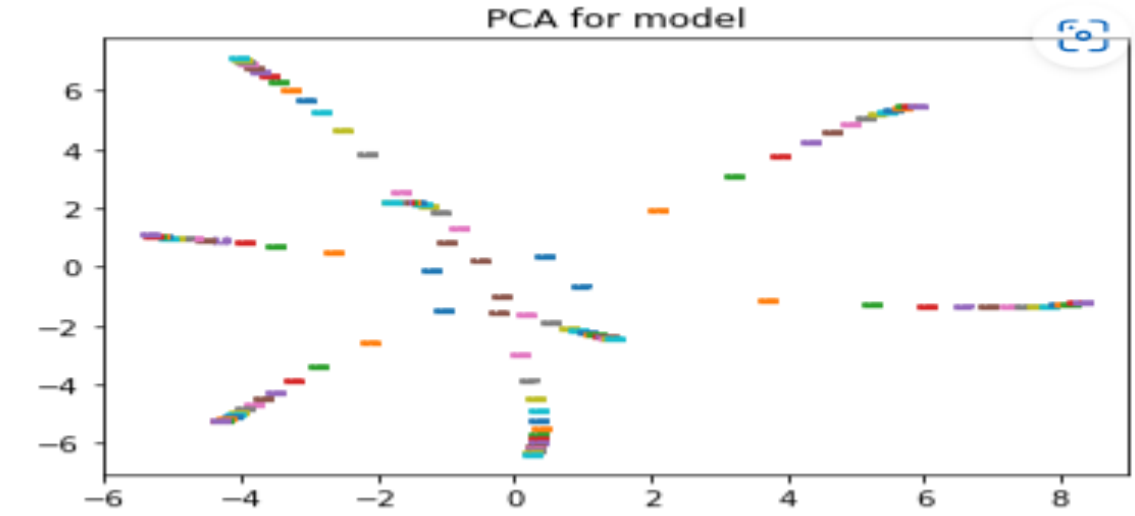
Result:

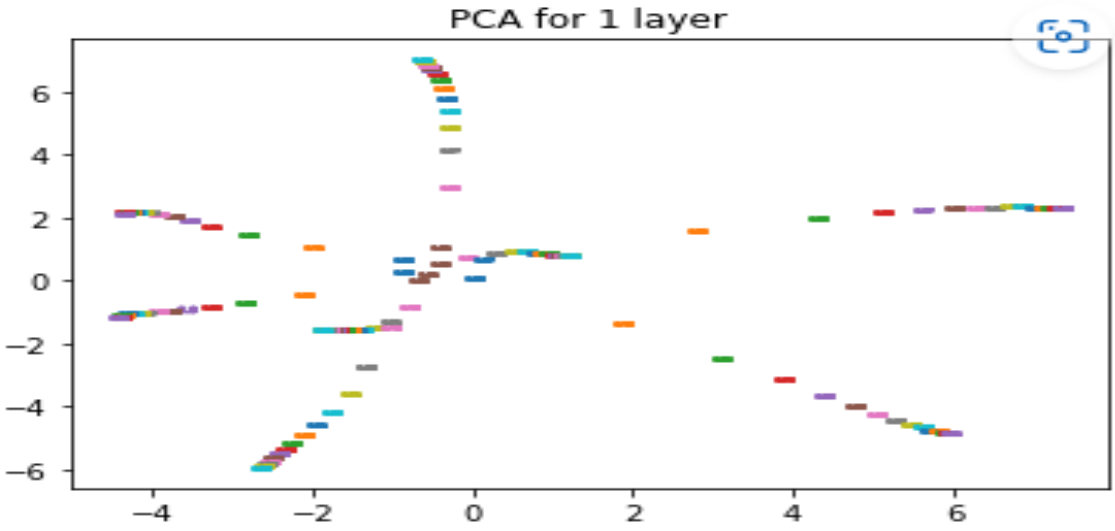
Model 1, with its optimized CNN structure based on the LeNet design, demonstrates superior performance compared to Model 2. THis is evident in its lower loss and higher training accuracy.The improved performance of model 1 can be attributed to its optimzed design in comparison to the custom CNN structure utilized in Model 2.

HW 1-2 Visualize the optimization process

The training procedure involves collecting the weights every three epochs and repeating he process eight times. The dimensionality of the weights is then reduced to two dimensions using PCA(Principal Component Analysis).This model has been trained on the MNIST dataset with

60,000 samples in the training set and 10,000 in the testing set.It consists of three dense layers and utilizes the cross entropy loss as the loss function,the Adam optimiser for optimization , and a learning rate of 0.0004.The batch size used for training is 1000 and the activation function used is ReLu.



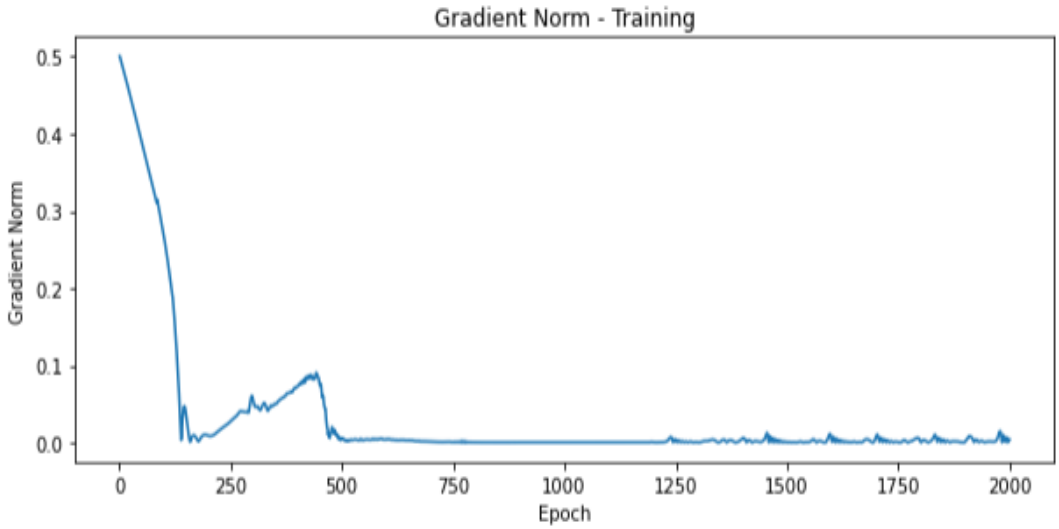


The graph represents the reduction of dimensions in the collected weights of the models after undergoing the PCA algorithm.Initially, Each model contained 8,240 parameters or weights,but after 8 training cycle , each for 45 epochs, and applying PCA, the dimensions of the weights were reduced and are displayed in the graph above.

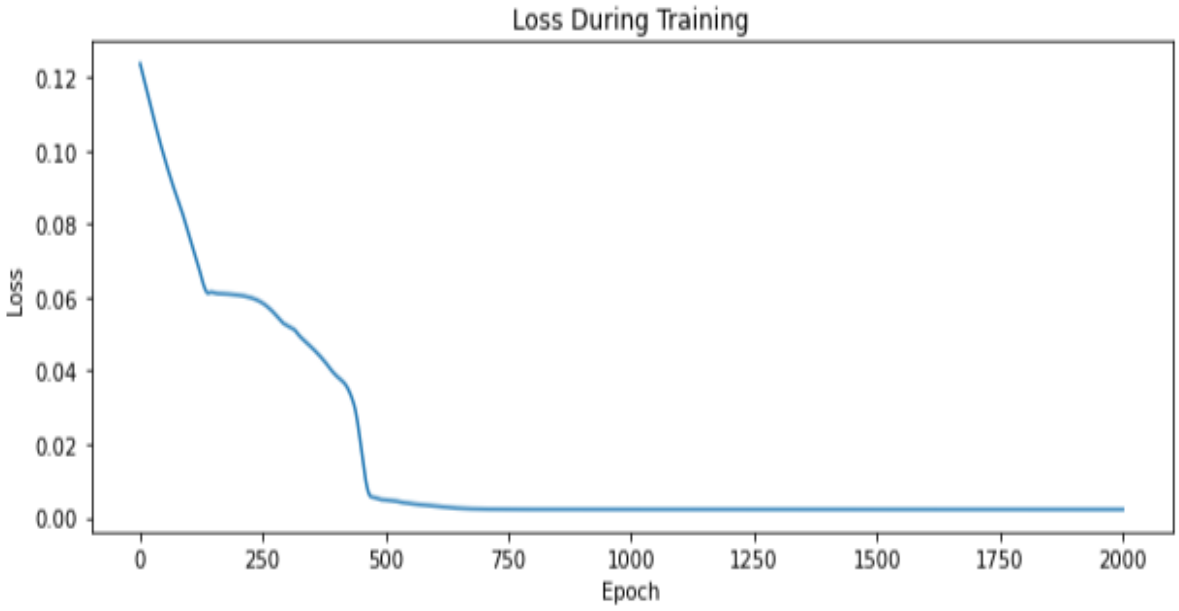
HW 1-2 Observe gradient norm during training

The graph represents the gradient norm computed across the epochs, using the function sin(5pix)/5pix. Instead of training the model based on iterations, the model was trained on epochs as the input size was already small. The use of this function for calculating both the gradient norm and the loss allowed for the reuse of the same calculation.

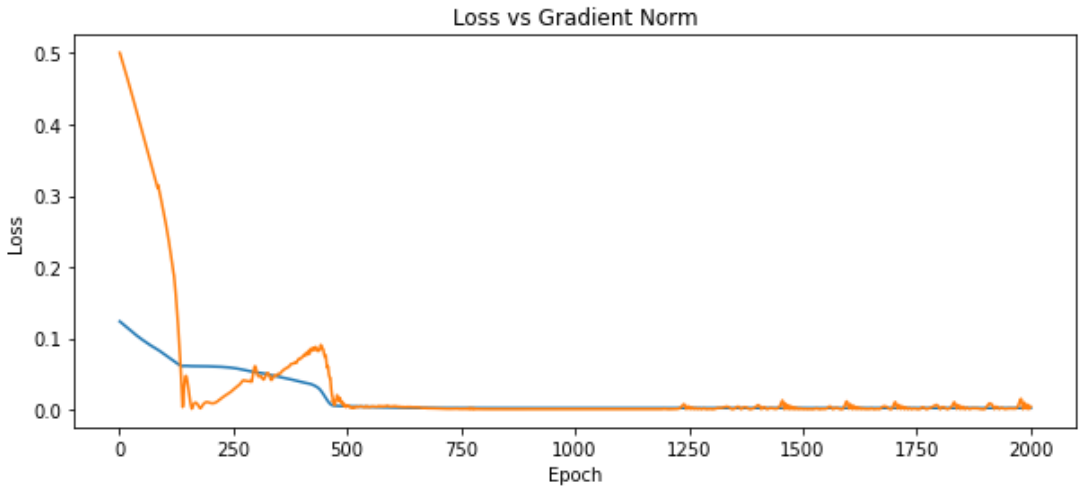
Following is a graph for gradient norm across the epochs.



The following is a graph for loss across the epochs



Loss Vs Gradient Norm:



Result:

The training of the model has produced successful convergence. The gradient shows a slight increase after 100 epochs, which is reflected in the loss plateauing initially and then decreasing at a slower rate before finally plateauing after approximately 500 epochs, as demonstrated in the other graph.

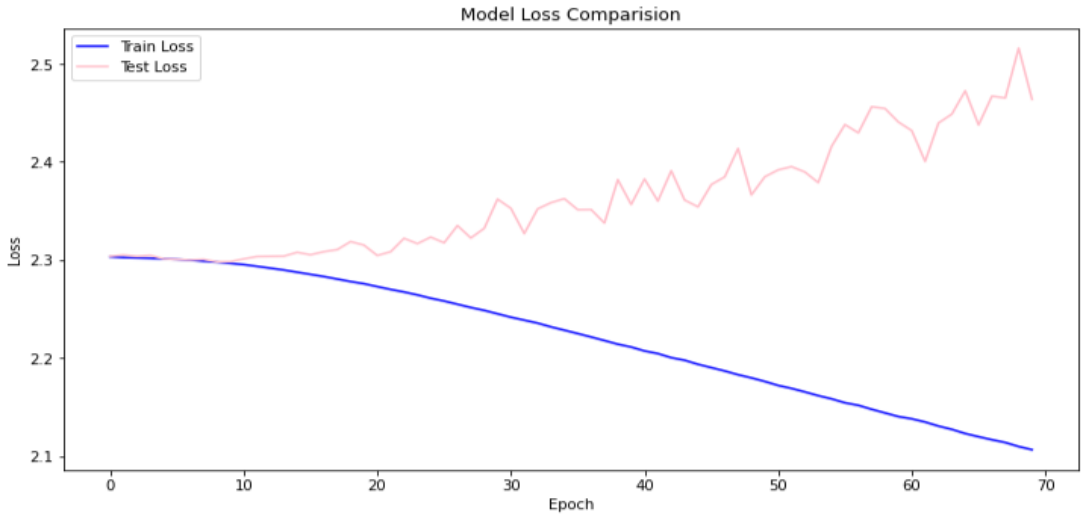
HW 1-3 Can network fit random labels?

The model was trained using a dataset of 60,000 samples from the MNIST dataset, with a testing subset of 10,000 samples.

Model 1 is a CNN (LeNet) model with the following structure:

1. 2D convolution layer followed by a 2D max pooling operation and ReLU activation function.
2. 2D convolution layer followed by a 2D max pooling operation and ReLU activation function.
3. 2D dense layer with a ReLU activation function.
4. 2D dense layer with a ReLU activation function.

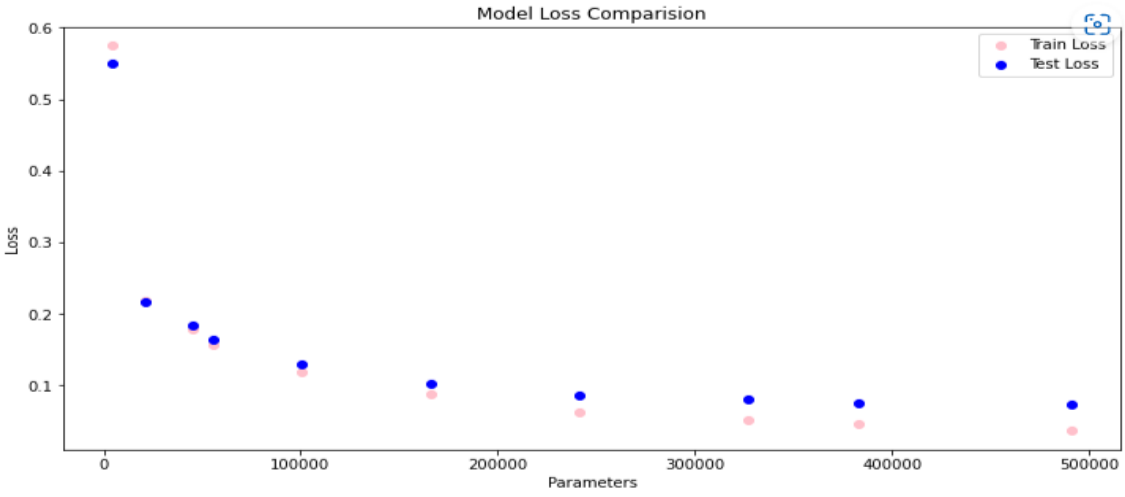
The hyperparamters for this model include a learning rate of 0.0001, Adam as the optimizer, a , train batch size of 100, a test batch size of 100, 100 epochs, and a cross entropy loss function.



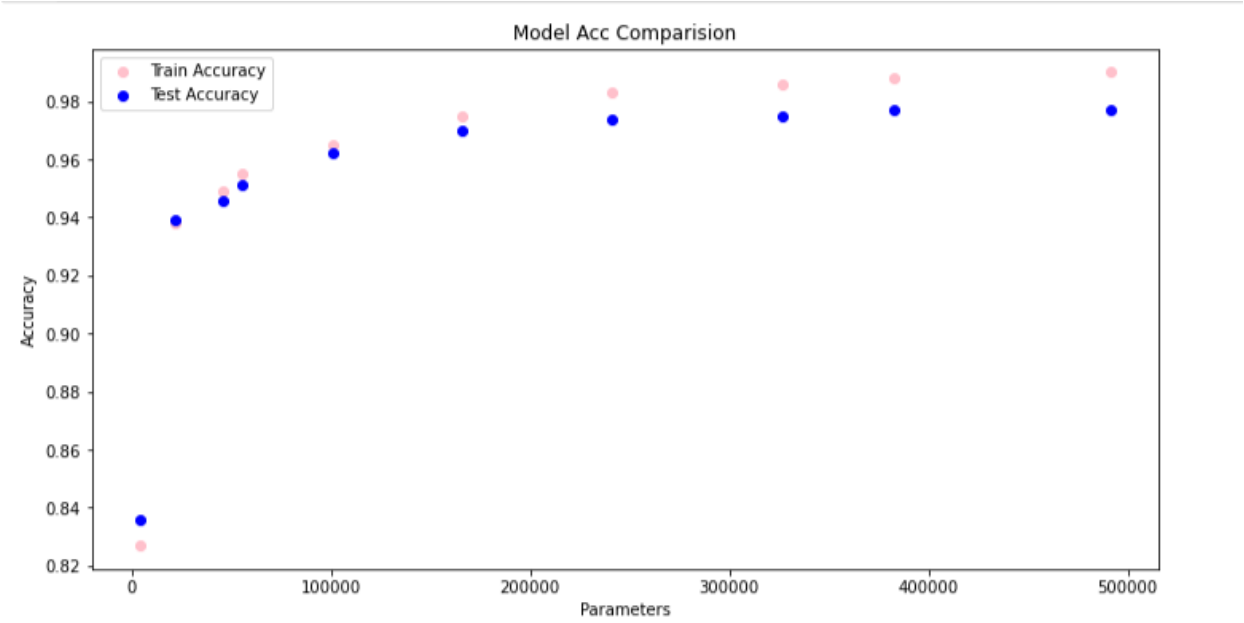
The model described above was trained on random data, which resulted in a slow training process as the model had to learn from random labels. As the training progressed, the model attempted to memorize the labels, leading to a reduction in loss over time. However, as the number of epochs increased, the test loss continued to rise while the train loss gradually decreased, as depicted in the graph. The gap between the train and test loss also grew larger as the number of epochs increased, indicating overfitting of the model on the training data.

HW 1-3 Number of parameters vs Generalization

The model described below was trained on the MNIST dataset, with a training set of 60,000 samples and a testing set of 10,000 samples. The model consists of three dense layers, with a cross entropy loss function, Adam optimizer, learning rate of 1e-3, batch size of 50, and ReLU activation function. The size of the models was varied by approximately doubling the inputs and outputs in each dense layer, resulting in an increase in the number of parameters for training. The graph compares the loss for various models.



The Following shows Graph for Accuracy comparison



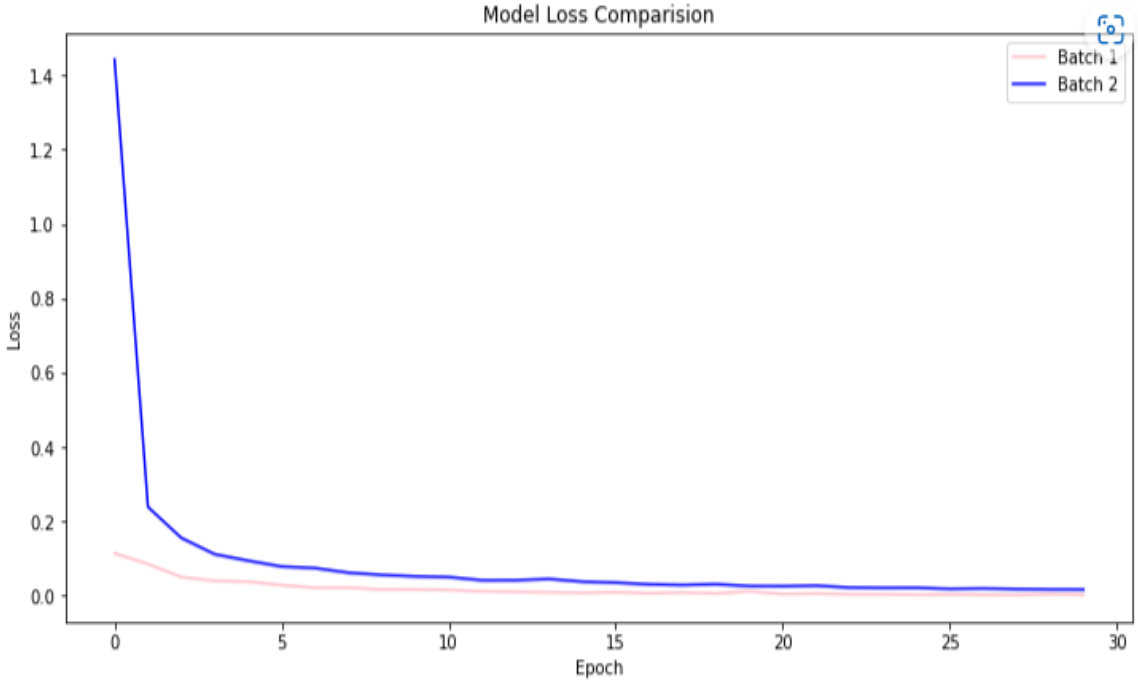
From the graphs, it is evident that as the number of parameters increases, the difference between the train and test loss/accuracy also increases. The test loss reaches a plateau earlier than the training loss or accuracy, indicating overfitting. This is because the larger number of parameters for the model to train leads to overfitting. Although the increased parameters result in higher training accuracy and lower training loss, it is important to reduce the difference between the train and test loss/accuracy in order to prevent overfitting. The trend of overfitting can be inferred from the graphs, but due to limitations in computing power, further increasing the number of parameters was not possible.

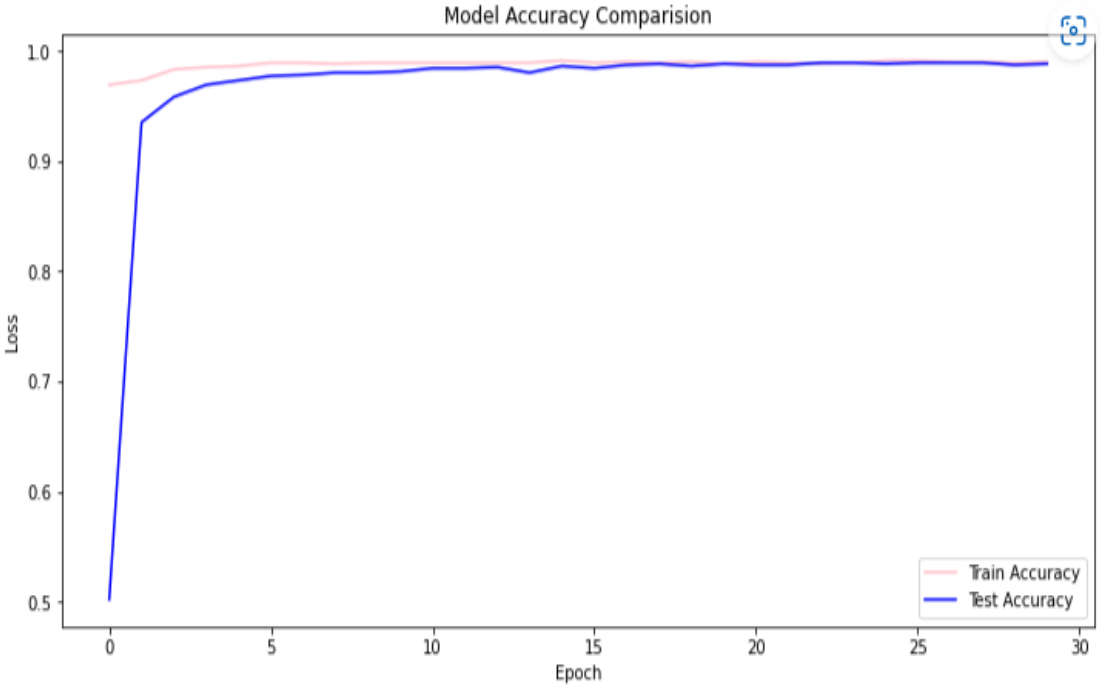
HW 1-3 Flatness vs Generalization

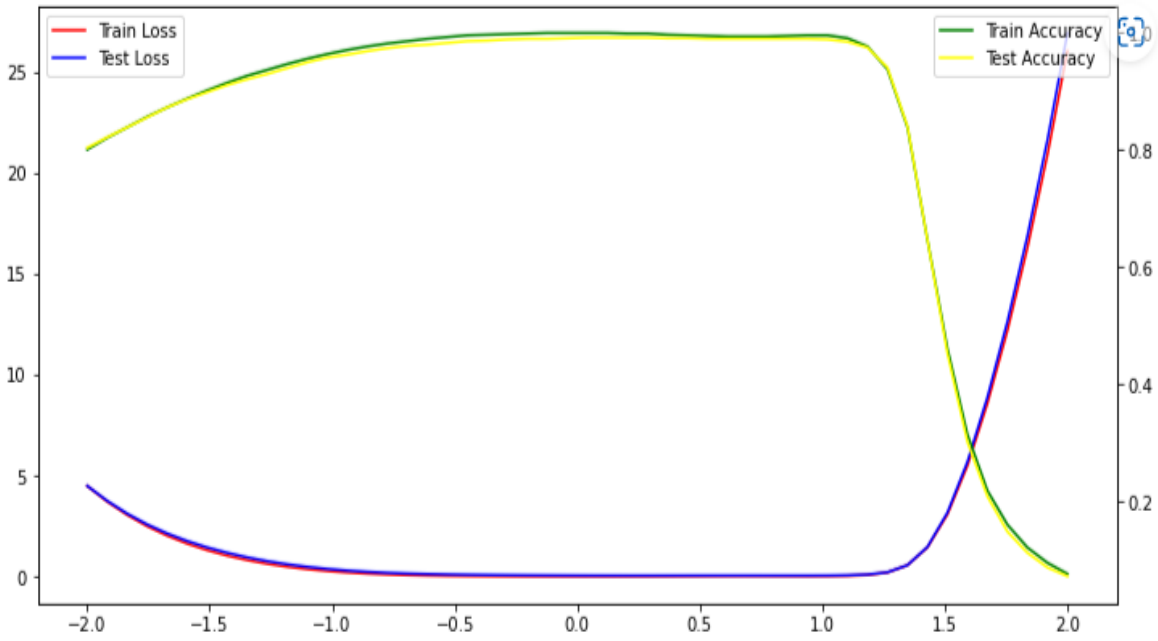
Part 1

This model is trained on a dataset consisting of 60,000 training samples and 10,000 testing samples from the MNIST dataset. The architecture of the model includes 3 dense layers and 2 convolution layers, and the loss function used is Cross Entropy Loss. The optimizer used is SGD and the learning rate is set to 1e-3 and 1e-2. The batch size is varied between 100 and 500, and the activation function used is ReLU. The training process involves collecting the weights from models trained with different batch sizes and using them to calculate an interpolation ratio. 50 models are then created with the new weight values, and their loss and accuracy are captured and plotted in a single graph for comparison.

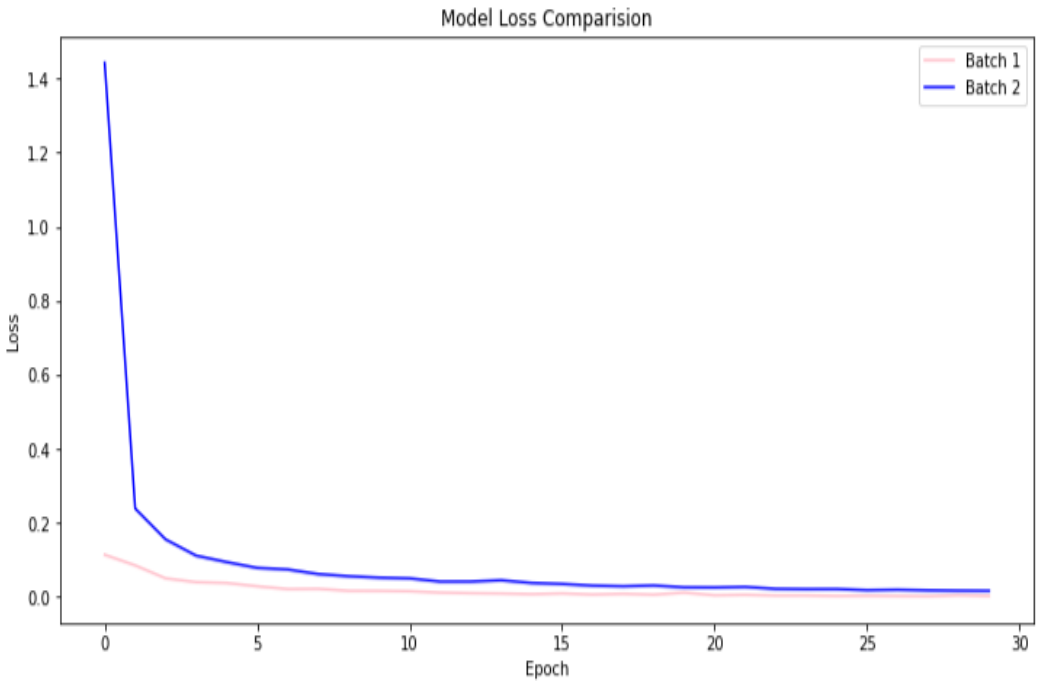
Learning Rate 1e-2

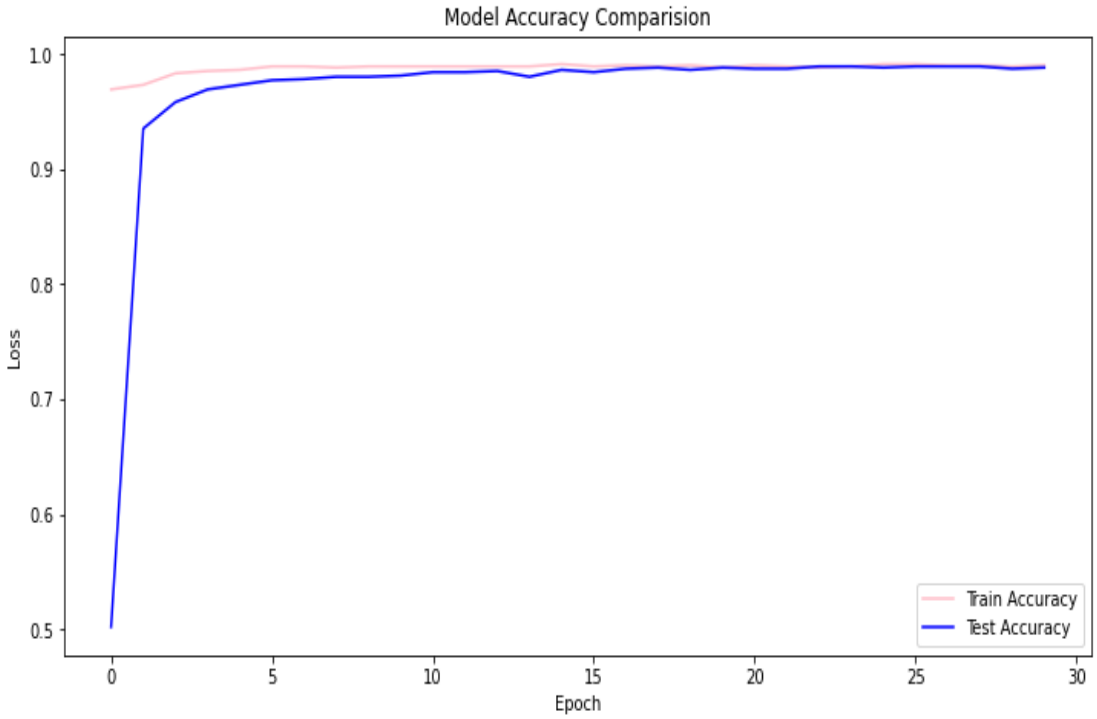

Model Loss Comparision
Batch 1
Batch 2
Loss
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
5
10
15
20
25
30

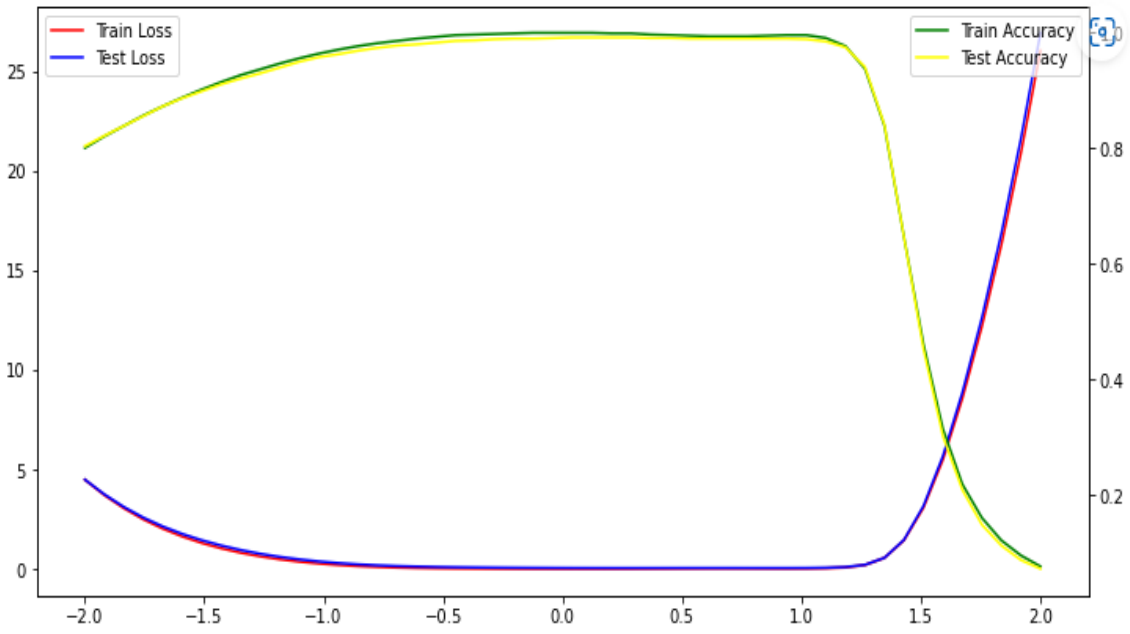

Model Accuracy Comparision
Train Accuracy
Test Accuracy
Loss
Epoch
0.5
0.6
0.7
0.8
0.9
1.0
0
5
10
15
20
25
30



Learning rate 1e-3:







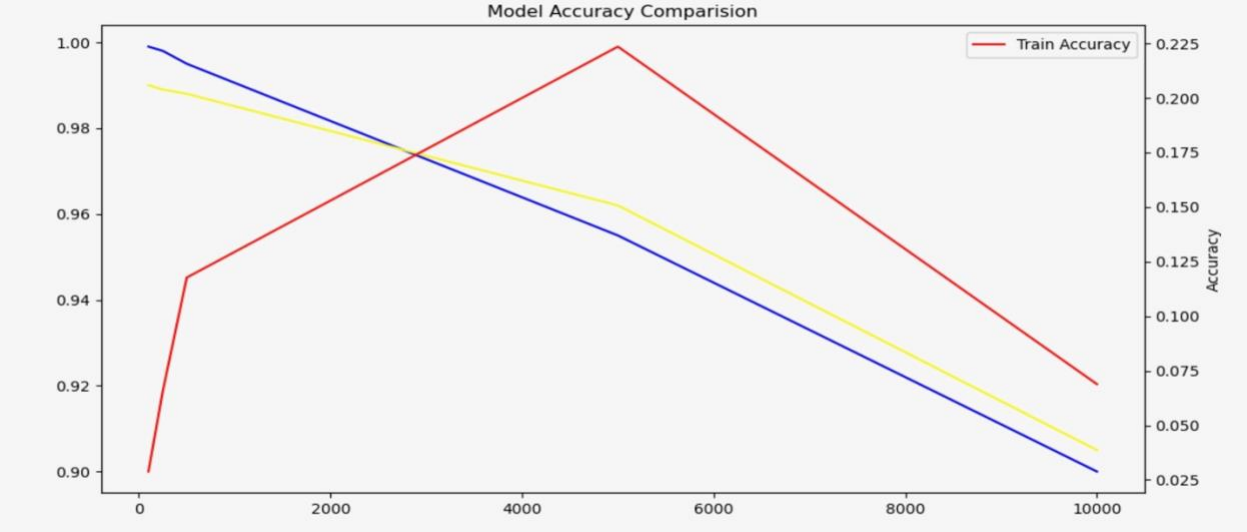
Result:

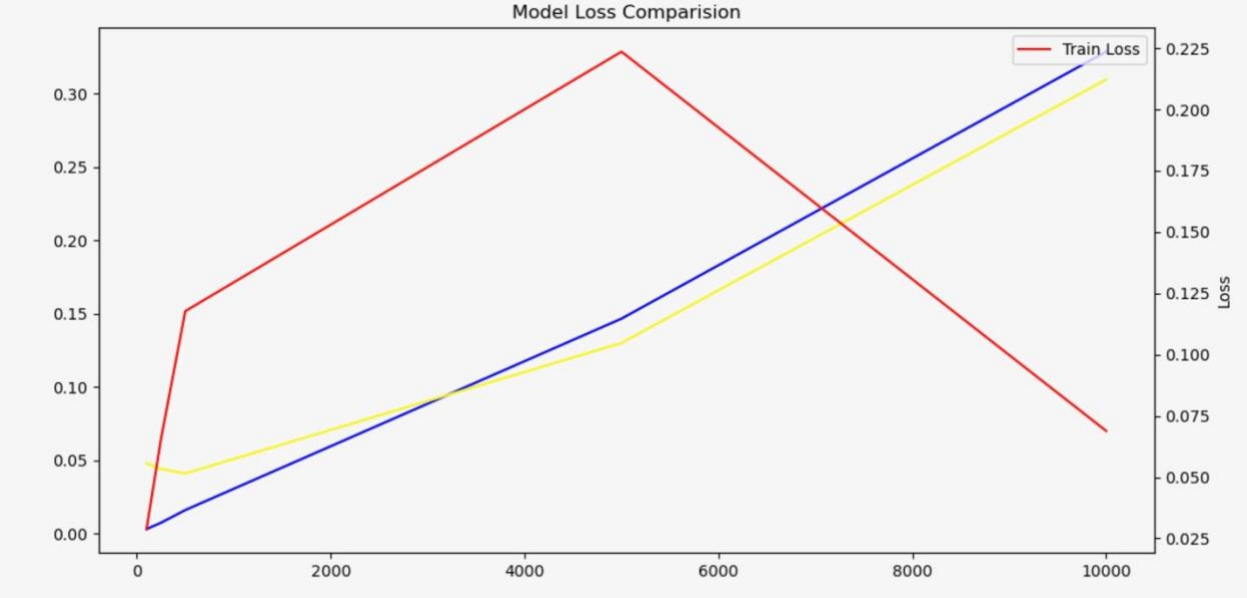
The above experiment showed that the accuracy of the test and train models decreases around 1.5 for both models with varying learning rates. However, the accuracy quickly increases at an alpha value of 1.5, which is calculated using the interpolation ratio formula " (1-alpha) *

batch1_param + alpha * batch2_param." This result indicates that while all machine learning algorithms interpolate between datapoints, having more parameters than data can cause the model to memorize the data and interpolate between them.

Flatness vs Generalization – Part 2

The model described below has been trained on 60,000 samples from the MNIST dataset with a testing set of 10,000 samples. It consists of two convolution layers and three dense layers, with a ReLU activation function. The optimization algorithm used is SGD and the learning rate is set to 1e-3, with a batch size of 50. The loss function applied is Cross Entropy Loss.





As the batch size increases, the sensitivity of the network decreases, as seen in the above graphs. The sensitivity reaches its highest point at around 4000 batches and then starts to decline, resulting in a decrease in accuracy and an increase in loss. Hence, we can conclude that when the batch size exceeds 5000 epochs, the network becomes less sensitive.