Student Name: Archana Saroj
Student ID: C16199009

CPSC-8430 Deep Learning

In this assignment, we covered the following sections:

1. Deep vs. Shallow:
   - Simulate functions: $(\frac{\sin(5\Pi x)}{5\Pi x})$ and $sgn(\sin(5\Pi x))$
   - Train on actual tasks using shallow and deep models: MNIST dataset
2. Optimization:
   - Visualize the optimization process: MNIST dataset
   - Observe gradient norm during training: $(\frac{\sin(5\Pi x)}{5\Pi x})$
   - Investigate what happens when the gradient is almost zero
3. Generalization:
   - Explore if the network can fit random labels
   - Compare the relationship between the number of parameters and generalization
   - Study the relationship between flatness and generalization

1 Deep vs. Shallow

1.1 Simulate a function

In the assignment, three deep neural network (DNN) models were created to simulate functions $(\frac{\sin(5\Pi x)}{5\Pi x})$ and $sgn(\sin(5\Pi x))$. The models were configured differently in terms of the number of dense layers and parameters. In all three deep neural network (DNN) models, a regularization technique of "weight decay" was applied to help prevent overfitting. The maximum number of training epochs was set to 25000. The training process would conclude either when the maximum number of epochs was reached or when the model has converged, which is indicated by a near-zero loss that no longer decreases.The models had the following configurations:

The specific configuration of the third DNN model is as follows::

- Model 1:
  - 7 dense layers
  - Activation function: "leaky relu"
  - Total number of parameters: 571
  - Loss function: "MSELoss"
  - Optimizer function: "Adam"
  - Hyperparameters:
    - Learning rate: 0.0012
    - Weight decay: 1e-4

- Model 2:
  - 4 dense layers
  - Activation function: "leaky relu"
  - Total number of parameters: 572
  - Loss function: "MSELoss"
  - Optimizer function: "Adam"
  - Hyperparameters:
    - Learning rate: 0.0012
    - Weight decay: 1e-4
- Model 3:
  - 1 dense layer
  - Activation function: "leaky relu"
  - Total number of parameters: 571
  - Loss function: "MSELoss"
  - Optimizer function: "Adam"
  - Hyperparameters:
    - Learning rate: 0.0011
    - Weight decay: 10^{-4}

1.1.1 Function 1:($\frac{\sin(5\Pi x)}{5\Pi x}$)

The Fig. 1 is the plot for function 1 In the training process of the three DNN models with the

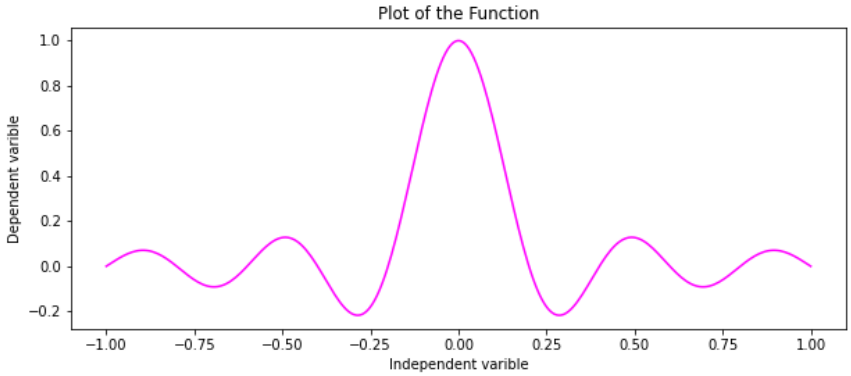

Figure 1: plot for function 1:

specified configurations, they have each converged at the following epoch:

- Model 1: Epoch = 961
- Model 2: Epoch = 857
- Model 3: Epoch = 2186

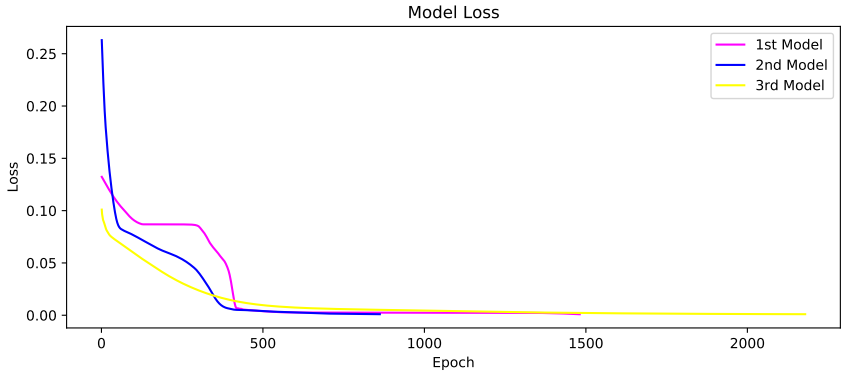

Figure 2: EpochVSLoss

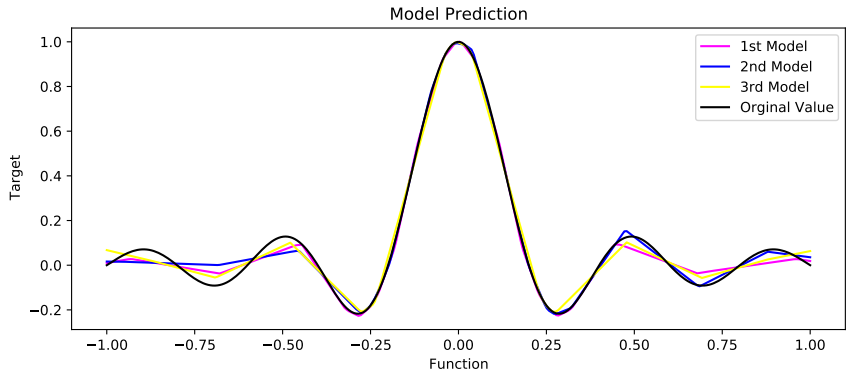

Figure 3: Model Prediction

Fig. 2 is the visualisation of the loss against the no. of epoch for all three models and Fig. 3 is the graph illustrates the model predictions for all three models compared to the ground truth:
Observation : The statement that Models 1 and 2, which have more layers, converge faster than Model 3 with a single layer is generally accurate. As shown in the graph, all the models exhibit similar performance compared to the ground truth. However, the addition of more layers allows the model to reach convergence more quickly. However, it is important to note that having more layers in a model can also result in increased training time, overfitting, and the possibility of getting stuck in suboptimal solutions. The ideal number of layers for a model depends on the problem being solved and the available data.

1.1.2 Function 2:$sgn(\sin(5\Pi x))$

The learning rate hyperparameter of the model was updated to learning rate=0.009. The models converged at the following epochs: Model 1 at epoch 8146, Model 2 and Model 3 at epoch 25000. Fig. 4 is the plot for function 2. Fig. 5 is the visualisation of the loss against the no. of epoch for all three models and fig. 6 is the graph illustrates the model predictions for all three models compared to the ground truth:
Observation : According to the loss graph, it appears that except for Model 1, neither Model 2 nor Model 3 were able to converge even after reaching the maximum number of epochs. Although their prediction accuracy is comparable, the model with the largest number of layers was capable

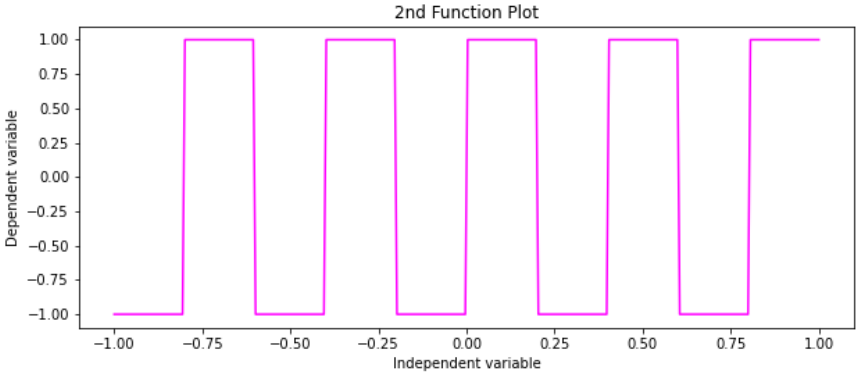

Figure 4: plot for function 2:

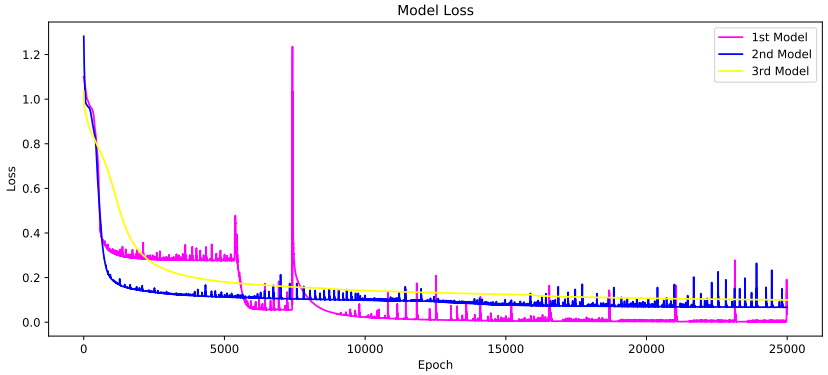

Figure 5: EpochVSLoss for Function 2

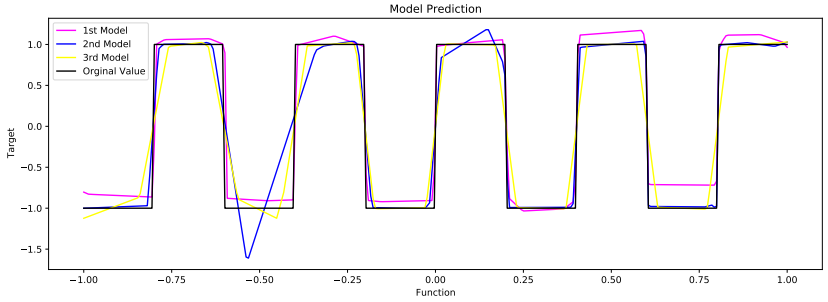

Figure 6: Model Prediction for Function 2

of reaching convergence on a more complicated function.

1.2 Train on Actual Tasks

For this section of the assignment, we have created three Convolutional Neural Network (CNN) models with varying number of layers and similar number of parameters on the MNIST dataset. The details of each model are as follows:

- CNN-1:
  - Number of Convolution Layers: 2 with Kernel size = 4
  - Max pooling with pool size = 2 and strides = 2
  - Number of dense layer = 2
  - Activation Function: "relu"
  - Total number of parameters: 25550
  - Loss Function: "CrossEntropyLoss"
  - Optimizer Function: "Adam"
  - Hyperparameters:
    - Learning Rate: 0.0001
    - Weight Decay: 1e-4
    - Dropout = 0.25
- CNN-2:
  - Number of Convolution Layers: 2 with Kernel size = 4
  - Max pooling with pool size = 2 and strides = 2
  - Number of dense layer = 4
  - Activation Function: "relu"
  - Total number of parameters: 25570
  - Loss Function: "CrossEntropyLoss"
  - Optimizer Function: "Adam"
  - Hyperparameters:
    - Learning Rate: 0.0001
    - Weight Decay: 1e-4
    - Dropout = 0.25
- CNN-3:
  - No. of Convolution Layers: 2 with Kernel size = 4
  - Max pooling with pool size = 2 and Strides = 2
  - No. of dense layer = 1
  - Activation Function: "relu"
  - Total no. of parameters: 25550
  - Loss Function: "CrossEntropyLoss"
  - Optimizer Function: "Adam"
  - Hyperparameters:
    - Learning Rate: 0.0001
    - Weight Decay: 1e-4
    - Dropout = 0.25

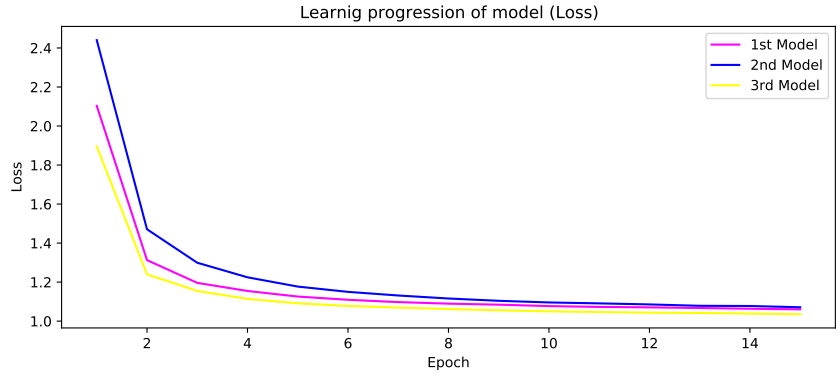

Figure 7: EpochVSLoss

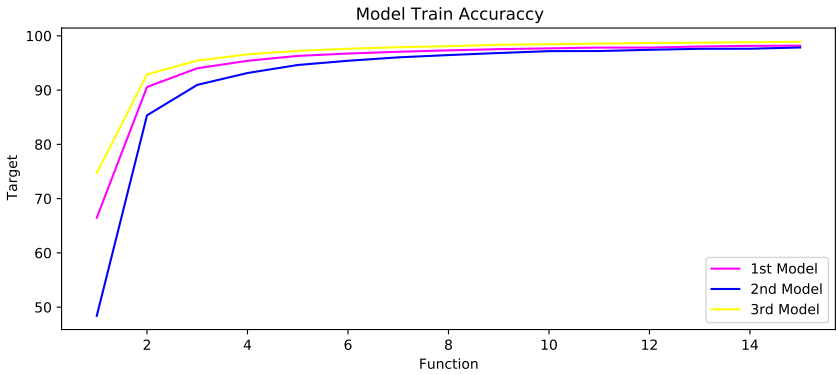

Figure 8: Model Accuracy

Observation : Fig 7 is the visualisation of the loss against the no. of epoch for all three models. Fig 8 illustrates the prediction of all three models against the ground truth. The graphs suggest that Model 3 outperforms the other two models, despite having only one dense layer. Model 3 has the lowest loss value and the highest accuracy in the training dataset, with a train accuracy of 98.898% and a loss of 0.0242. Meanwhile, Model 1 has a train accuracy of 98.045% and a loss of 0.0744, and Model 2 has a train accuracy of 97.825% and a loss of 0.0880. Additionally, Model 3 also achieves the highest accuracy in the test dataset, with a test accuracy of 98.8%, while Model 1 has a test accuracy of 98.66% and Model 2 has a test accuracy of 97.9%.In conclusion, the results from the graphs indicate that Model 3 is the best performing model among the three.

2 Optimization

2.1 Visualize the Optimization Process

The model used for this part of the assignment is a Deep Neural Network (DNN) consisting of a single dense layer and using the rectified linear unit (ReLU) activation function. The model has a total of 397,510 parameters. The loss function applied during training is the Cross-Entropy Loss and the optimization algorithm used is the Adam optimizer with a learning rate of 0.0004 and weight decay of 10^{-4}. The model was trained over 45 epochs and the weights were recorded at each

epoch, with the entire training process being repeated 8 times,

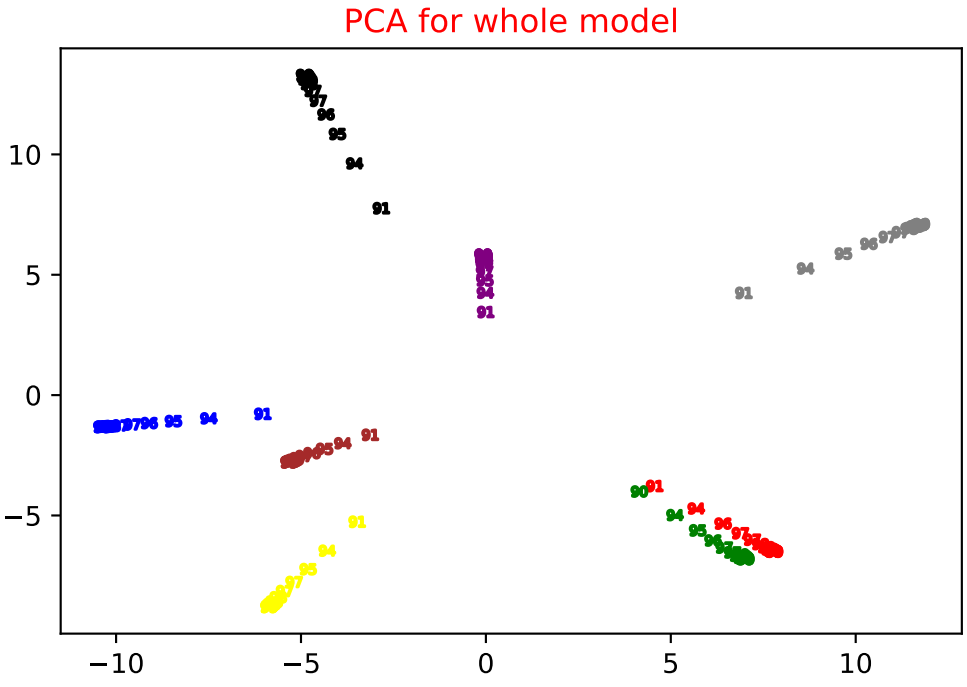

Figure 9: PCA for Model

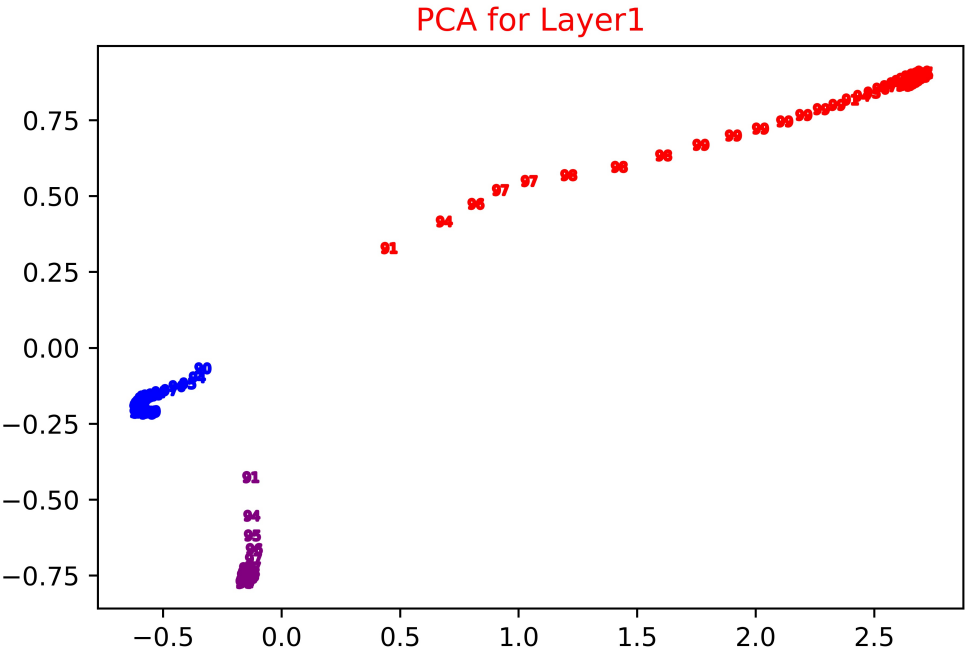

Figure 10: PCA for Layer 1

Observation : Fig 9 illustrates the PCA for the Model whereas the Fig 10 illustrates the PCA

for Layer 1. The Deep Neural Network (DNN) model was trained 8 times for 45 epochs, and the weights obtained from each training run were collected. These weights were then processed through Principal Component Analysis (PCA) in order to reduce their dimensionality. The original weights, which were 417,500 in number for each model, were reduced to only 2 dimensions through this process. The resulting 2-dimensional data was then plotted to generate a set of graphs that provide insight into the optimization process. This visualization of the weight values after PCA reduction helps to understand the training process and any patterns or trends that may be present in the weights.

2.2 Observe Gradient Norm During Training (using functions: $(\frac{\sin(5\Pi x)}{5\Pi x})$)

The model used for this section of the assignment is a Deep Neural Network (DNN) consisting of one dense layer and utilizing the rectified linear unit (ReLU) activation function. The total number of parameters in the model is 1,501. The loss function used during training is Mean Squared Error (MSE) Loss, and the optimization algorithm employed is the Adam optimizer. The hyperparameters for the optimization process include a learning rate of 10^{-3} and weight decay of 10^{-4}.After training the model for 1800 epochs, convergence was reached with a final MSE loss value of 0.0009995186. The gradient norm, which is a measure of the rate of change of the loss with respect to the model parameters, was also observed after convergence. Fig. 11 illustrates the gradient norm that was observed. Fig. 12 illustrates the loss observed across the epochs.

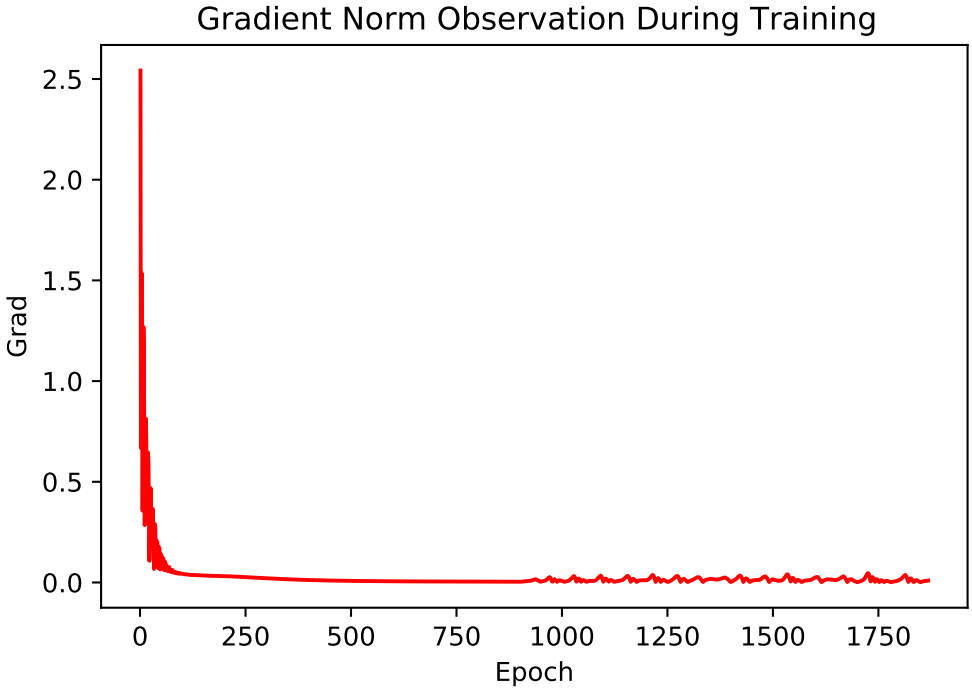

Figure 11: Gradient Norm Observed During Training

Observation : The training process of a deep learning model involves updating its parameters over a series of iterations or epochs until convergence is reached. In this specific case, the training lasted for 1800 epochs and during this time, the model's gradient values and loss underwent changes that provide insight into its learning progress.Starting with the gradient values, it is observed that there was a drastic decrease until around the 200th epoch. This drop in gradient values could indicate

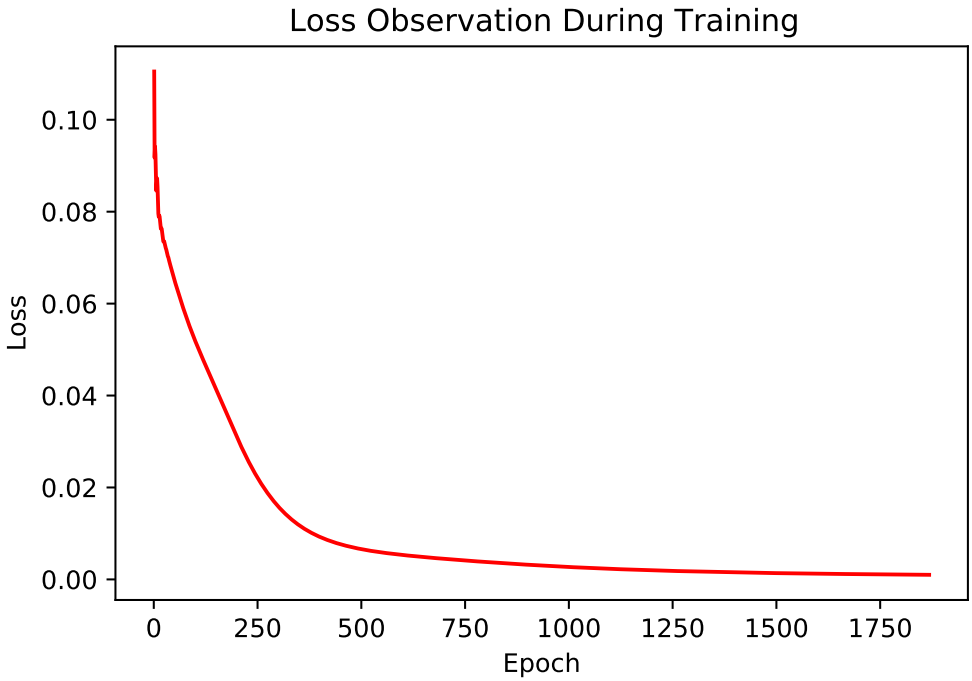

Figure 12: Loss Observed During Training

that the model was successfully updating its parameters to minimize the loss during this time. After the 200th epoch, there was a slight increase in the gradient values at around the 900th epoch, followed by another dip at the 1000th epoch. Despite these fluctuations, the final gradient value at the end of the training was 0.005702.Moving on to the loss, the model experienced a substantial drop in loss until around the 400th epoch. However, after this point, the rate of decrease in loss slowed down and stabilized into a plateau. This could mean that the model has reached a certain level of optimization for the training data and further training may not result in significant improvement. Overall, the changes in the gradient values and loss over time are typical of deep learning training processes and provide valuable information for determining the optimal number of epochs to train for, or when to stop training if the model has already converged.

2.3 What Happened When Gradient is Almost Zero (using functions: $(\frac{\sin(5\Pi x)}{5\Pi x})$)

For this section of the assignment to observe the when gradient is almost zero and compute minimum ratio, a DNN model and function were used. The following are details on the same:

- 1 Dense layer
- Activation Function: "relu"
- Total number of parameters: 1501
- Loss Function: "MSELoss"
- Optimizer Function: "Adam"

Post-training the above model for 100 epochs, the loss did not reach zero.Fig 13 illustrates the graph for the loss vs minimal ratio:

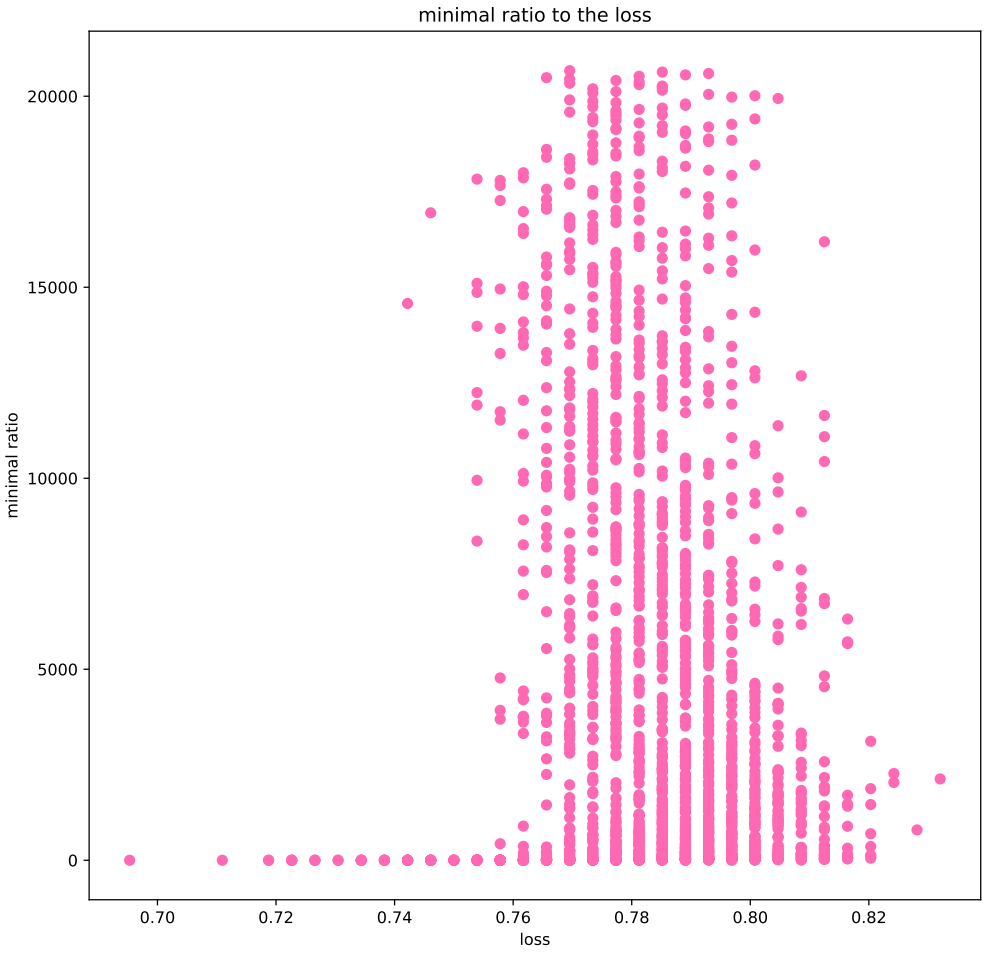

Figure 13: loss vs minimal ratio

3 Generalization

3.1 Can network fit random labels?

In this experiment, we train our Deep Neural Network (DNN) model on the MNIST dataset with randomized labels in the training set. The objective is to observe the model's behavior during the learning process and compare its performance against the test data, which has proper labels configured with the images.The model consists of a single dense layer with 397510 parameters and ReLU as the activation function. The loss function used is Cross Entropy Loss, and the optimizer used is Adam. The hyperparameter learning rate is set to 0.0001.

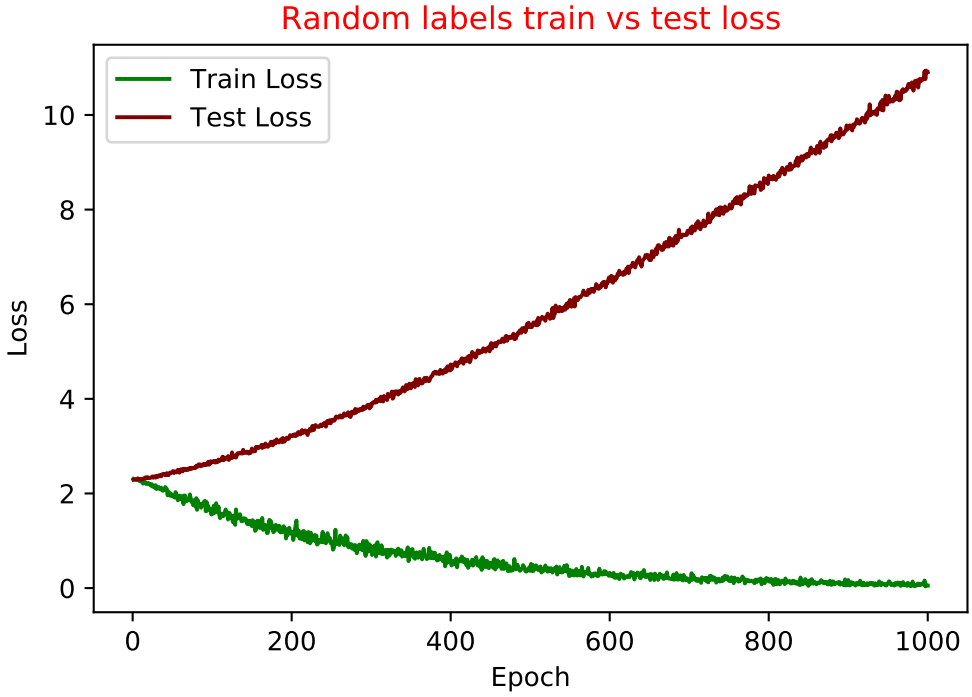

Figure 14: Random Labels train vs Test loss

Observation : Fig 14 shows the graph achieved on training the above model with the given train and test conditions. The results show the performance of the model on the training and test data through a graph. It is important to note that training a model with randomized labels is not a meaningful or useful exercise as the model will not learn to associate images with their correct labels. As a result, the performance on the test data is poor since the model has not learned the correct associations during training.

3.2 Number of parameters v.s. Generalization

In this section, we create ten Convolutional Neural Network (CNN) models with a similar architecture, but varying the values of the dense layer. The number of parameters for each model ranges from $39,760$ to $15,900,10$. The models have the following details:

- 2 convolution layers with a kernel size of 4
- Max pooling with a pool size of 2 and strides of 2
- 2 dense layers
- ReLU activation function
- Total number of parameters: 25550
- Cross Entropy Loss as the loss function
- Adam as the optimizer
- Learning rate of 0.0001

- Dropout of 0.25

After training all the models, a plot of the loss comparison between the models is observed. Fig 15 shows the Loss comparison between models. Fig 16 shows the Accuracy comparison between models. The plot provides insights into the performance of the models and helps determine the best model based on the lowest loss.

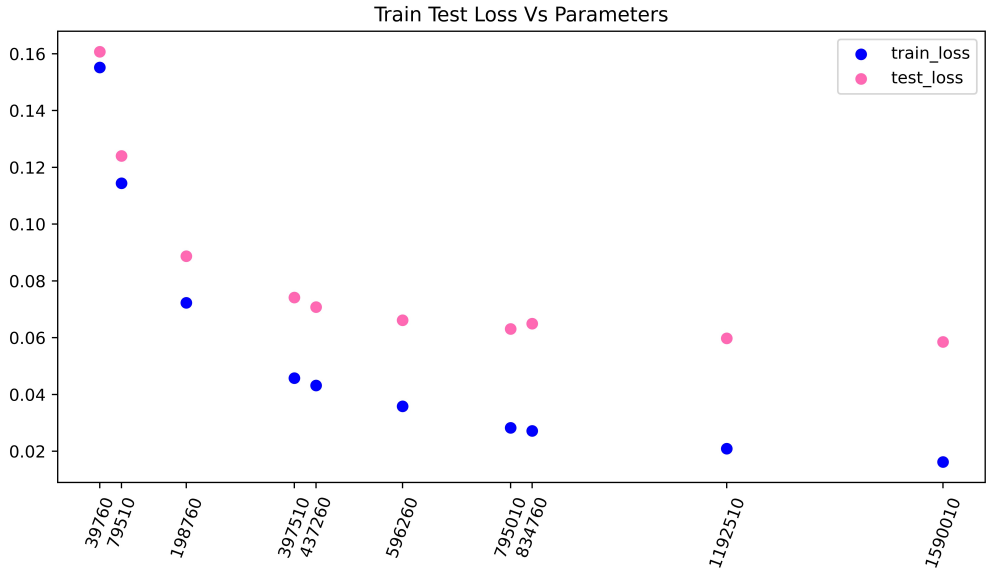

Figure 15: Loss comparison between models

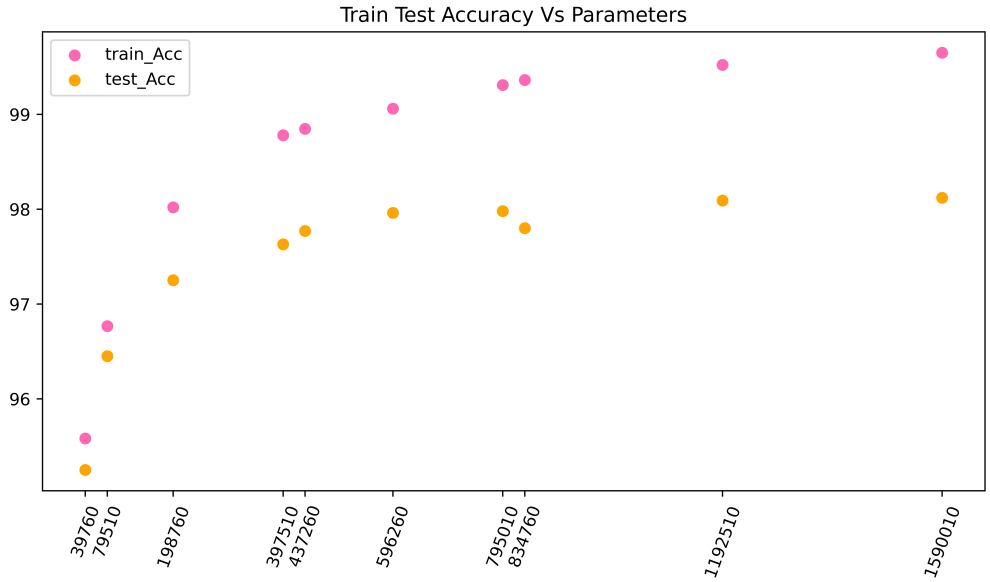

Figure 16: Accuracy comparison between models

Observation : It is commonly observed that as the number of parameters in a model increases, the model tends to have better performance as indicated by lower loss and higher accuracy. However,

the test loss and accuracy tend to reach a plateau much earlier than the training loss or accuracy. This can be attributed to the occurrence of overfitting in the model, where there are too many parameters for the model to train. Overfitting occurs when a model becomes too complex and starts to memorize the training data rather than generalizing to new, unseen data. As a result, the model performs well on the train data but poorly on the test data, leading to a gap in performance between the train and test data.To prevent overfitting, it is crucial to carefully consider the number of parameters in the model and to use regularization techniques like dropout or weight decay. The goal is to achieve the best accuracy and lowest loss while also avoiding overfitting and maintaining a narrow gap in the relative performance of both train and test data. This requires striking a balance between model complexity and model generalization, so that the model is capable of generalizing to new data.

3.3 Flatness v.s. Generalization

3.3.1 Part 1

In this section of the assignment, two deep neural network (DNN) models are to be created with the same architecture, but with different values of training batch size. The first model has a training batch size of 64, while the second model has a training batch size of 1000. The MNIST dataset will be used for training. The details of the model are as follows:

- One dense layer
- Activation Function: `relu`
- Total number of parameters: 397510
- Loss Function: `CrossEntropyLoss`
- Optimizer Function: `Adam`
- Maximum number of epochs: 15

Hyperparameters:

- Learning Rate: 0.0015
- Weight decay: 1e-4

The training of the model depicted in Fig. 17 results in different loss trends depending on the batch size used. After training the model with different batch sizes, we extracted the parameters of the models in vector format and assigned them to the values of $batch1_{param}$ and $batch2_{param}$. Next, we created a set of 31 numbers ranging from -2.0 to 2.0 as the values of α.Using α, $batch1_{param}$, and $batch2_{param}$, we generated the values of θ using the formula:

$$\theta = (1 - \alpha) * batch1_{param} + \alpha * batch2_{param}$$

for all the different values of α and used the resultant vector to be converted into parameters to be inserted in the model to generate 31 new models.All of these models were trained once, and their loss and accuracy were captured. Similarly, we performed loss and accuracy on the test data set and collected these data. The batch size used for the new models based on the interpolation formula and test operations were:i)Test Batch: 64 ii)Train Batch: 100.The Fig 18 shows the loss and accuracy were observed per α values:

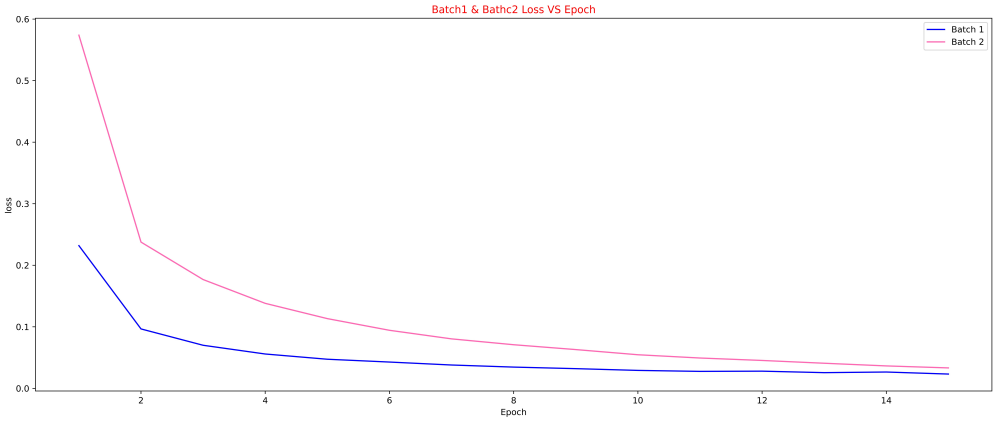

Figure 17: Model with different batch sizes

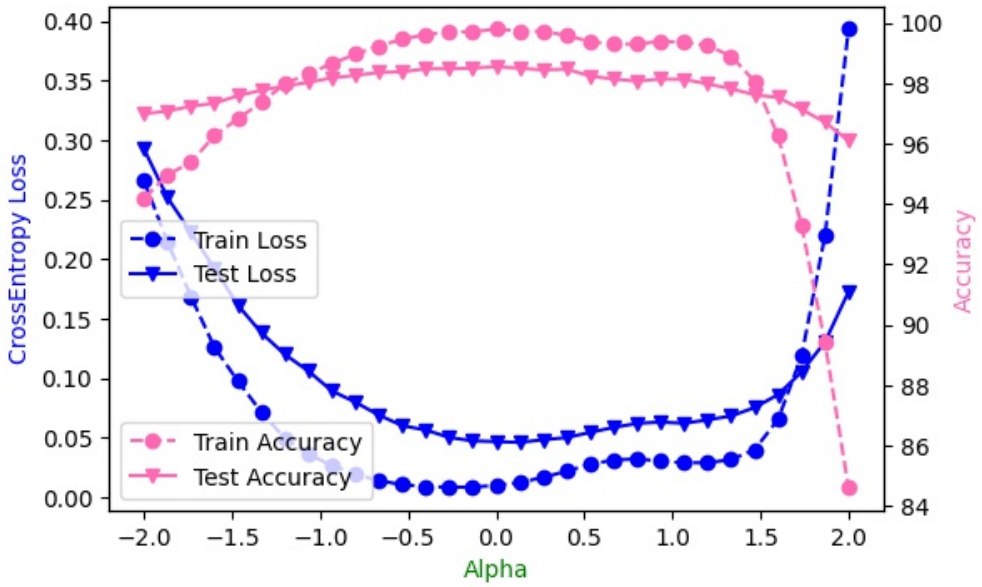

Figure 18: loss and accuracy were observed per alpha

We also trained the MNIST dataset using the same model, but this time, instead of varying the batch sizes, we varied the learning rates as follows: i)$LR_1 = 1e^{-3}$ ii)$LR_2 = 1e^{-2}$ The training of the model with these different learning rates yields the following loss trends as shown in Fig 19: To further clarify and improve the description, we repeat the procedure of obtaining feature vectors from the two models trained with different learning rates. We use 31 different values of alpha to generate 31 new models by inserting these values into the model as parameters. These 31 models are then trained once and their loss and accuracy are recorded. We evaluate the performance of these models by calculating the loss and accuracy on a test dataset using a batch size of 100 for both training and testing, and a learning rate of 10^{-3}. The results of the loss and accuracy for each alpha value are summarized in Fig 20

Observation : The models trained with different batch sizes show improved performance with lower loss and higher accuracy as alpha approaches 0.0, but a decrease in performance as alpha increases from 0.0 to 0.5, followed by improvement and then a steep decline for alpha values between 1.5 to

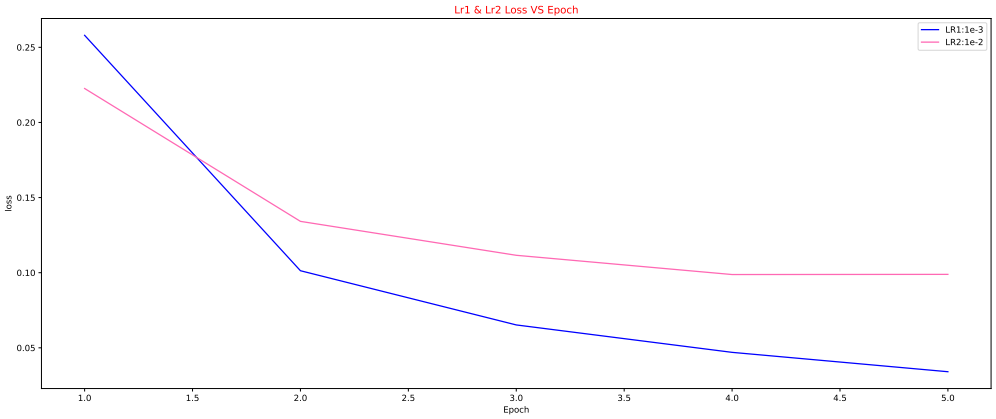

Figure 19: Model with different learning rates

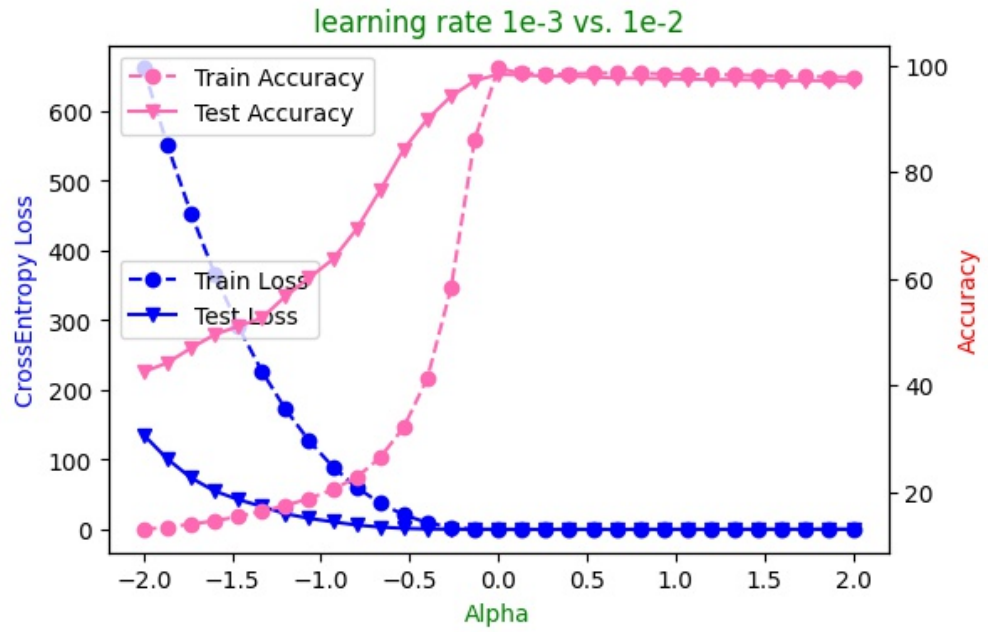

Figure 20: Loss and accuracy were observed per alpha values

2.0. The models trained with different learning rates, however, show stable good performance as alpha approaches 0.0. Machine learning algorithms interpolate between datapoints, but may start memorizing the data if the number of parameters exceeds the amount of data

3.3.2 Part 2

In this section, we create 5 DNN models with different batch sizes (10, 385, 760, 1135, 1510) to train on the MNIST dataset. The models have the same architecture, including 1 dense layer, ReLU activation, 397510 parameters, cross entropy loss, Adam optimizer, and maximum 5 epochs. The hyperparameters used are learning rate 10^{-3} and weight decay 10^{-4}. The sensitivities, loss, and accuracy are recorded during training to analyze the impact of batch size on model learning. The graph of train and test loss versus batch size is shown in Fig. 21. The graph of train and test

accuracy vs. batch size is shown in Fig. 22.

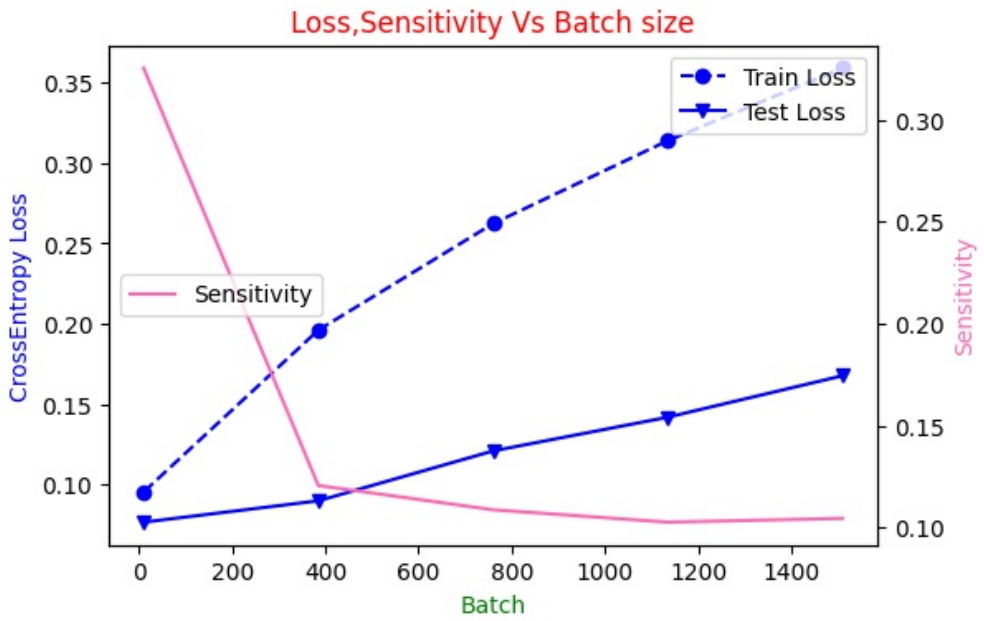

Figure 21: Train and test loss vs. batch size

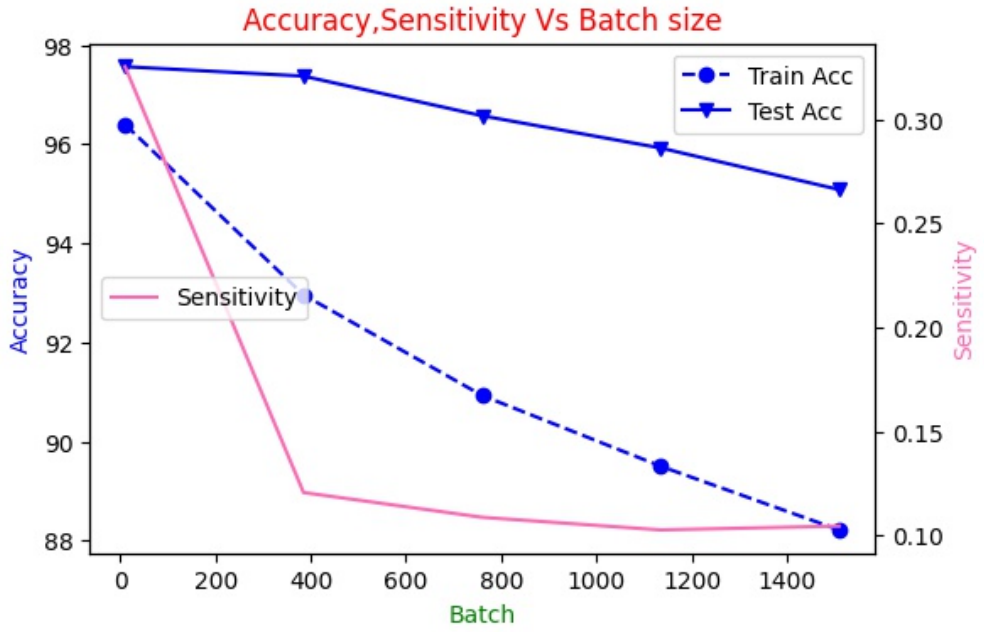

Figure 22: Train and test accuracy vs. batch size

Observation : From the graph presented above, it can be observed that as the batch size increases, the performance of the model deteriorates over the same number of epochs. This means that a model with a lower batch size has the potential to learn faster and more efficiently. On the contrary, as the batch size increases, the sensitivity of the model decreases. This implies that the

model may become less sensitive to changes in the input data as the batch size grows larger. In order to achieve optimal performance, it is important to carefully balance the batch size with other hyperparameters, as well as to monitor the model's sensitivity during training.