CS-671 Assignment 1 Report

Group-37

Team Members:

Name	**Roll no.**
Asif Hoda	B23253
Piyush Kumar	B23167
Bhupesh Yadav	B23200
Manjeet Rai	B23152
Yashodeep	B23040
Tarun Singh	B23103

February 16, 2025

1 Classification using Perceptron

The purpose of this report is to implement a classification model using a Perceptron with a one-against-one approach and implement the Backpropagation algorithm. The datasets used include a linearly separable dataset (LS) and a non-linearly separable dataset (NLS). The objective is to train the model using the perceptron learning algorithm with a step/sigmoidal activation function and analyze the classification performance on both the datasets and display the necessary comparisons.

Dataset and Preprocessing

Two datasets were provided:

- LS Dataset: 3 classes, each containing 500 data points. See Fig. 1

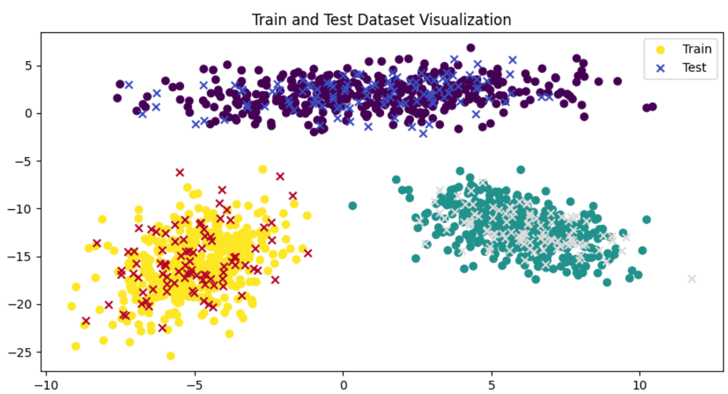


Figure 1: The LS dataset

- NLS Dataset: 3 classes, non-linearly separable. See Fig. 2

Each dataset was split into training (70%) and testing (30%) sets. The data was shuffled and preprocessed accordingly.

Perceptron Model and Training Results

A perceptron model was implemented with the following components:

- One-against-one approach for multi-class classification.
- Backpropagation algorithm implemented from scratch.
- Sigmoidal activation function used for learning.
- Training conducted over multiple epochs with a defined learning rate.

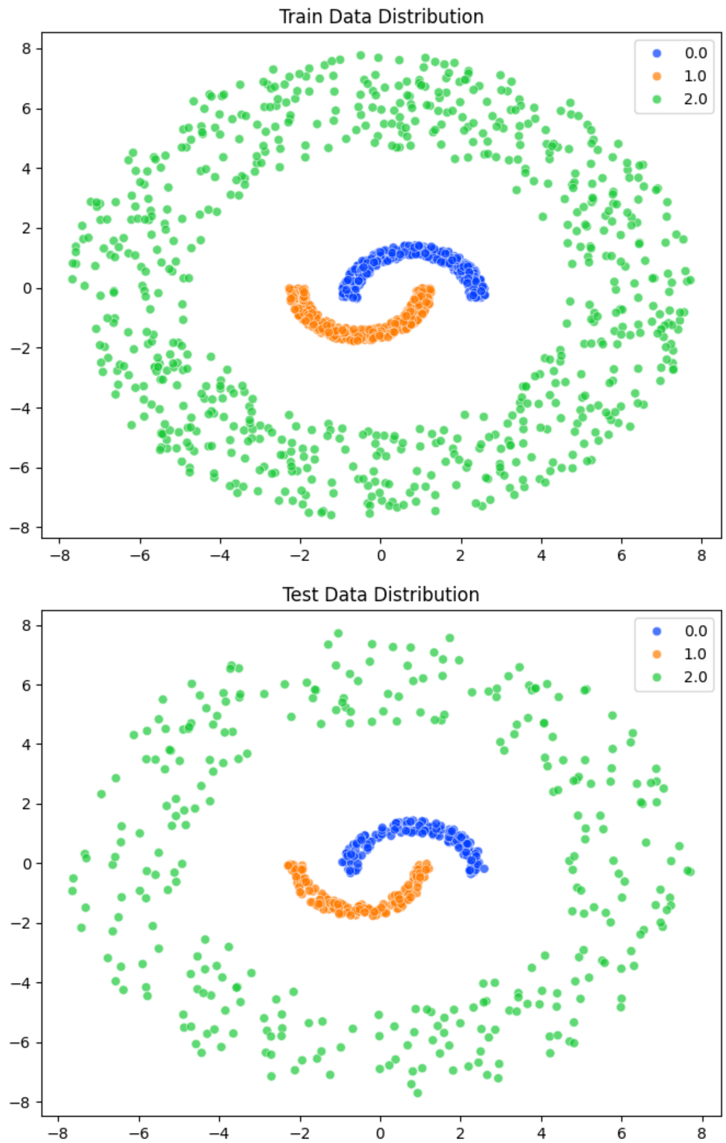


Figure 2: The NLS dataset

Error vs Epochs Plot

Choice of the Activation Function

The sigmoid activation function was chosen over the step activation function for both LS and NLS datasets due to the following reasons:

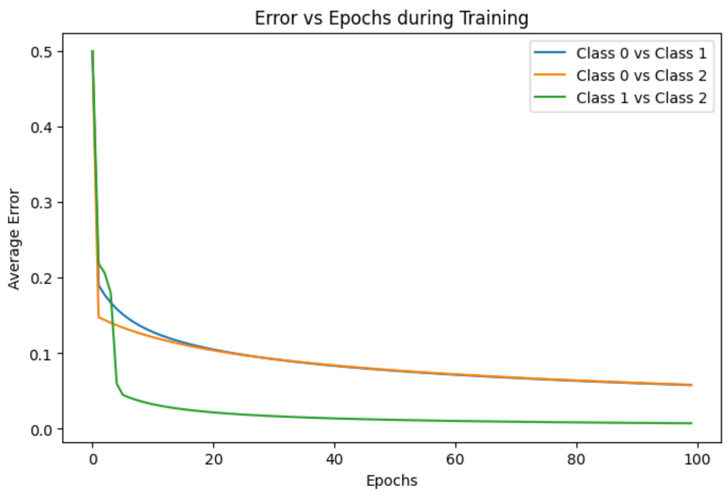


Figure 3: Plot of average error vs epochs on LS training data.

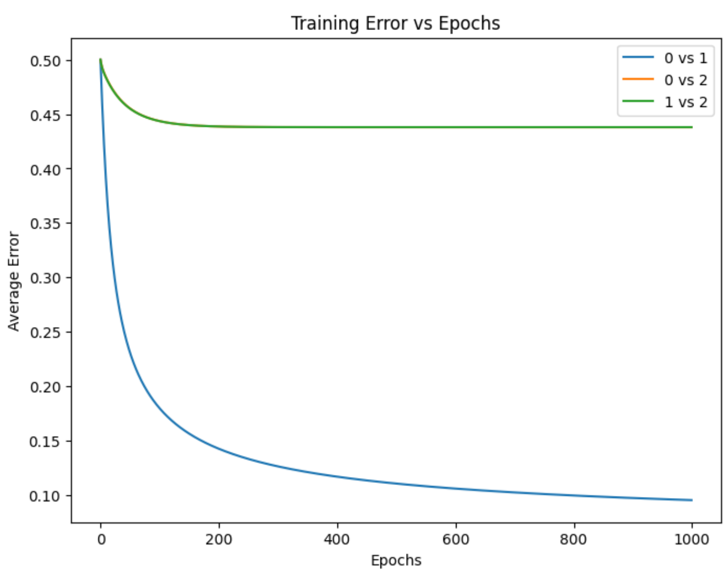


Figure 4: Plot of average error vs epochs on NLS training data.

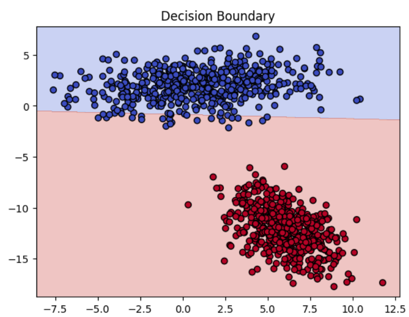
- The step function is a hard threshold function that does not allow smooth gradient-based optimization, making backpropagation ineffective.
- The sigmoid function provides a continuous and differentiable output, which facilitates efficient weight updates using gradient descent.
- For non-linearly separable data (NLS), the sigmoid function allows for

smooth decision boundaries, whereas the step function would lead to abrupt and ineffective classifications.

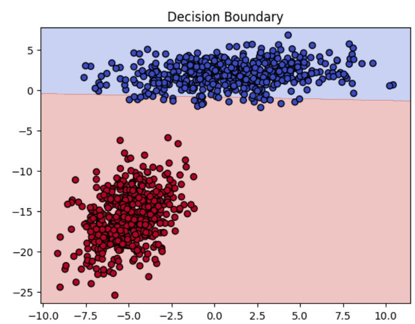
- Even for linearly separable data (LS), using sigmoid ensures stability and prevents oscillations during training.

Decision Region Plots

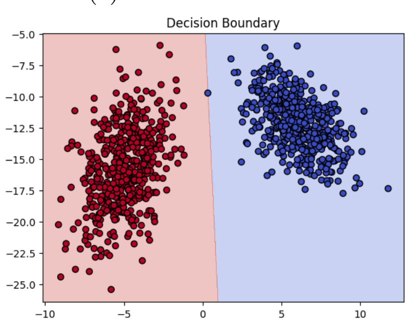
Decision region plots for individual class pairs and overall classification were generated and shown in the end of the document.



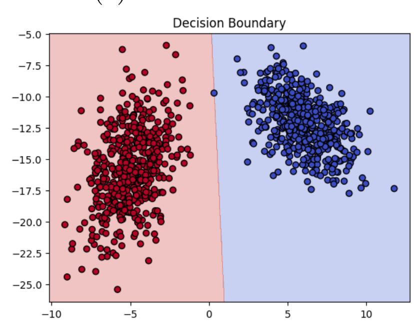
(a) Class 0 vs Class 1



(b) Class 0 vs Class 2

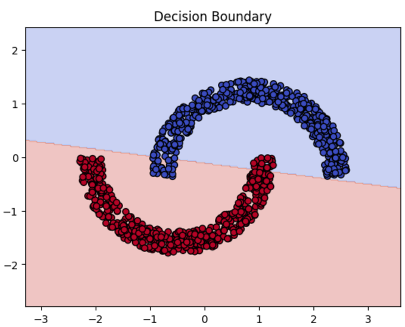


(c) Class 1 vs Class 2

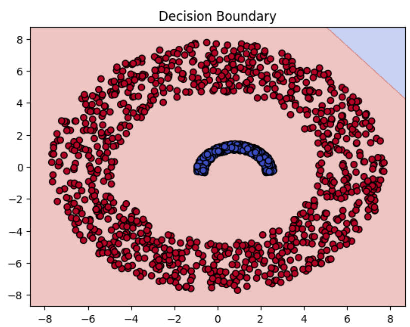


(d) Overall decision boundary

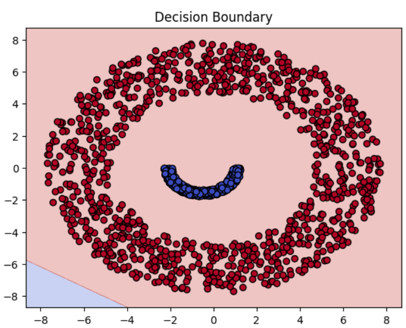
Figure 5: Decision boundary plots with training data for different class pairs and overall classification for LS dataset.



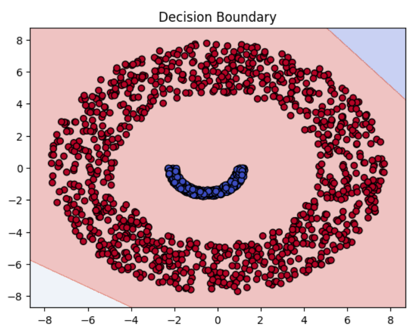
(a) Class 0 vs Class 1



(b) Class 0 vs Class 2



(c) Class 1 vs Class 2

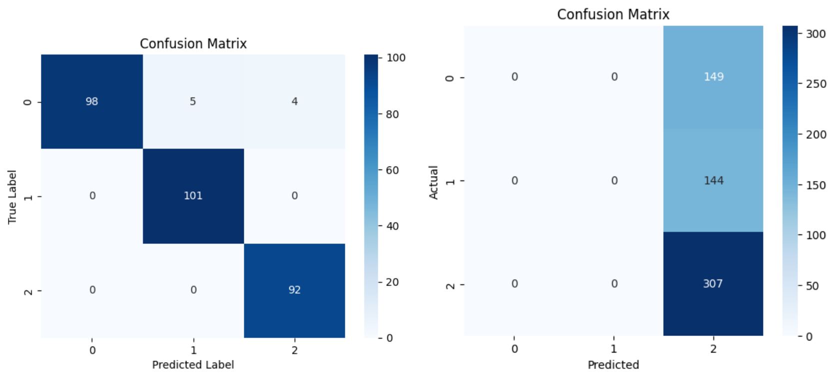


(d) Overall decision boundary

Figure 6: Decision boundary plots with training data for different class pairs and overall classification for NLS dataset.

Confusion Matrix and Accuracy

A confusion matrix was computed in both cases to evaluate the classification performance.



(a) LS dataset (b) NLS dataset

Figure 7: Confusion matrices for classification results.

The overall classification accuracy was computed as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{1}$$

The One-vs-One Perceptron Accuracy achieved was **97%** for the LS dataset and **51%** for the NLS dataset which is significantly lower in comparison.

Inferences and Conclusion

For Linearly Seperable Dataset:

- The perceptron model successfully separates the classes with some misclassifications.
- The training error reduces over epochs, showing convergence.
- The decision regions show distinct class separations, but some overlap exists.
- The model achieved an accuracy of 0.97, which indicates its effectiveness for classification.

For Non-Linearly Seperable Dataset:

- The dataset is not linearly separable, hence perceptron struggles with accuracy.
- The confusion matrix shows misclassification between certain classes.
- To improve accuracy, consider using an MLP or SVM.

2 Classification using MLP & CNN

Implementation and analysis of two classification models on the MNIST dataset is to be done as follows:

1. A Multi-Layer Perceptron (MLP) implemented from scratch using NumPy.
2. A Convolutional Neural Network (CNN) implemented using TensorFlow/Keras.

The MNIST dataset comprises 60,000 training images and 10,000 test images of handwritten digits (0-9). Each image is a 28x28 grayscale image, which is flattened into a 784-dimensional vector when processed by the MLP.

Dataset and Preprocessing

The MNIST dataset is preprocessed as follows:

- Pixel values are normalized to the range [0,1].
- For the MLP, each image is flattened into a 784-dimensional vector.
- For the CNN, images are reshaped to $28 \times 28 \times 1$.
- Labels are one-hot encoded for multi-class classification.

Visualization and Network Overview

Figure 8 displays a few sample images from the dataset along with an overview of the network architectures for both the MLP and CNN, as well as an error plot for the MLP.

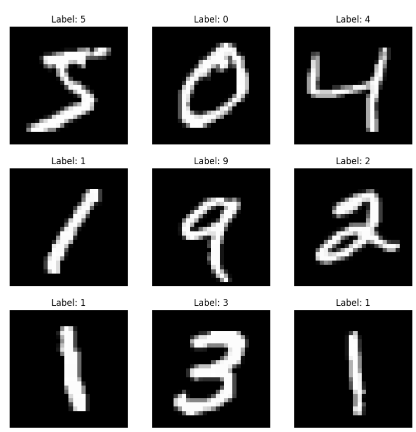


Figure 8: Sample Images

Model-1: Multi-Layer Perceptron (MLP)

Network Architecture

The MLP model consists of:

- An input layer with 784 neurons.
- One hidden layer with 64 neurons using the sigmoid activation function.
- An output layer with 10 neurons using the softmax activation function.

The sigmoid activation is used in the hidden layer to introduce non-linearity, and the softmax function in the output layer produces a probability distribution over the 10 digit classes.

Training and Loss Function

The network is trained using backpropagation with the categorical cross-entropy loss function. The training loss per epoch is plotted in Figure 9.

Results

The MLP model achieved a test accuracy of **86.45%**. The confusion matrix for the MLP is shown in Figure 10.

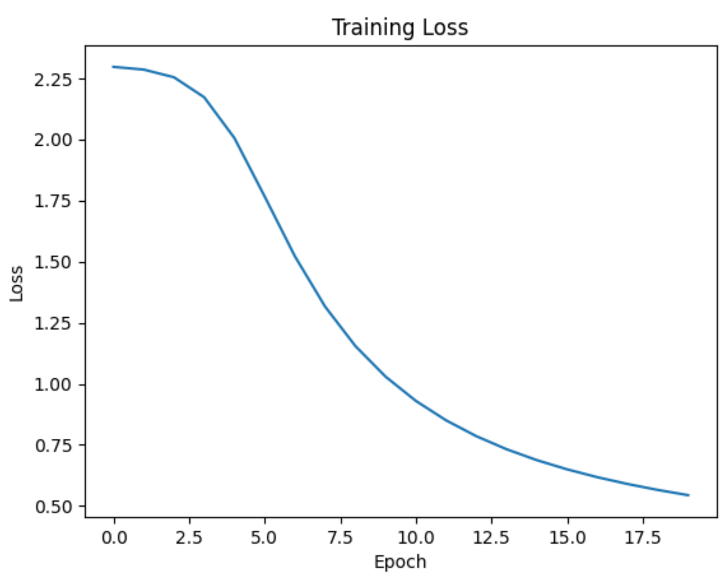


Figure 9: MLP Training Loss vs. Epochs

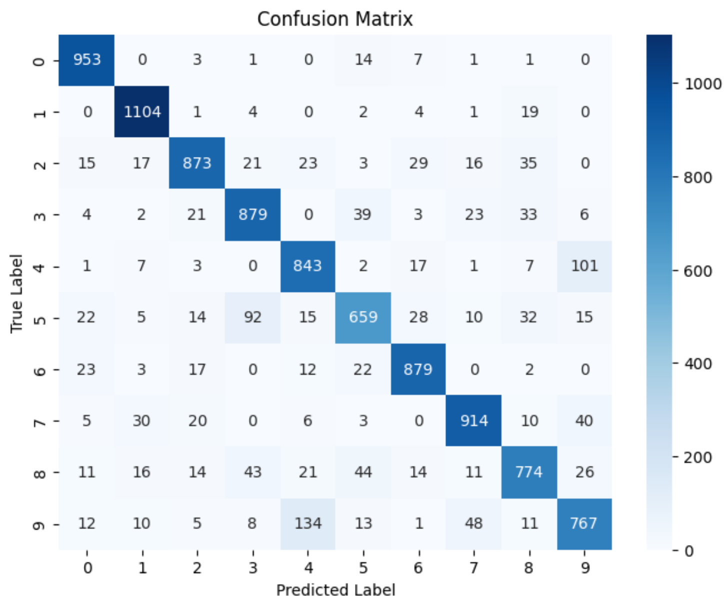


Figure 10: Confusion Matrix for MLP Model

Model-2: Convolutional Neural Network (CNN)

Network Architecture

The CNN model comprises:

- A convolutional layer with 32 filters (size 3×3) using the ReLU activation.
- A max-pooling layer to reduce spatial dimensions.
- A flattening layer.
- Two dense layers: one with 128 neurons (ReLU activation) and an output layer with 10 neurons (softmax activation).

Training and Evaluation

The CNN model is trained using the Adam optimizer and categorical cross-entropy loss. Training and validation loss and accuracy curves are plotted in Figure 11.

Results

The CNN model achieved a test accuracy of **98.58%** which is a big improvement over MLP. The corresponding confusion matrix is presented in Figure 12.

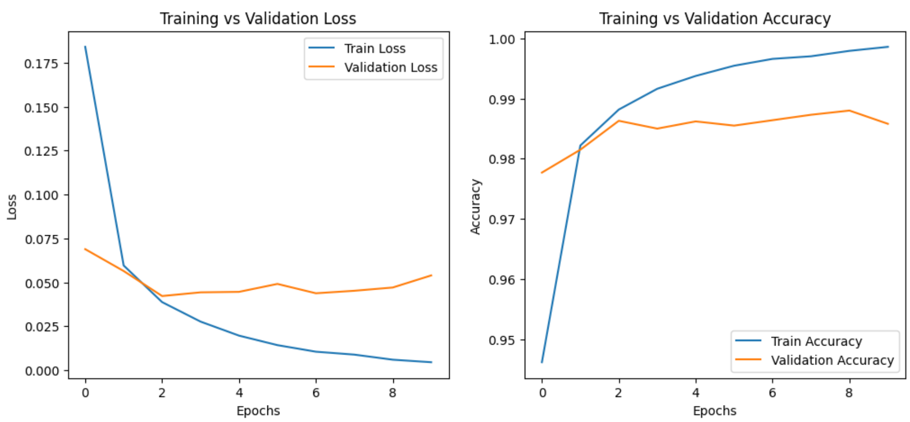


Figure 11: Training and Validation Loss and Accuracy for CNN

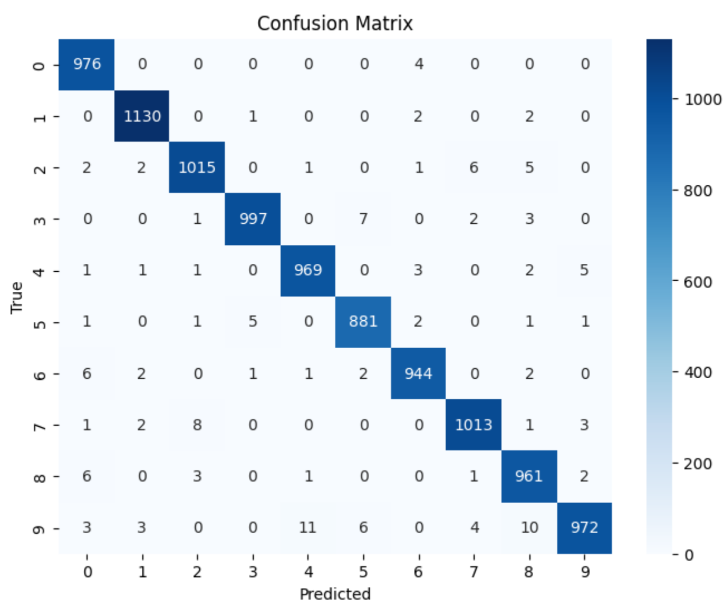


Figure 12: Confusion Matrix for CNN Model

Inferences and Conclusion

- The MLP model, implemented from scratch, validates the fundamental principles of neural network training but exhibits moderate classification accuracy.
- The CNN model, by leveraging convolutional layers to capture spatial features, significantly improves the recognition accuracy.
- Training curves indicate convergence for both models, with the CNN showing more robust performance on the validation set.
- Confusion matrices reveal that misclassifications predominantly occur among similar digit classes, suggesting potential areas for further improvement.