CS671: Deep Learning and Applications Programming Assignment-I

Krish
Roll Number: B23143

Aditya Kumar Gupta
Roll Number: B23308

Manish Kumar
Roll Number: B23269

Nimit Garg
Roll Number: B23379

Lakshay Garg
Roll Number: B23145

Arnav Kulkarni
Roll Number: B23067

February 16, 2025

1 Introduction

This report presents the implementation of Perceptron, Multi-Layer Perceptron (MLP), and Convolutional Neural Networks (CNN) for classification tasks. We classify linearly and non-linearly separable datasets using a Perceptron model and train an MLP from scratch for the MNIST dataset. Additionally, a CNN model is implemented using deep learning frameworks for improved performance. The models are evaluated based on accuracy, decision regions, and loss trends.

2 Part-I: Classification Tasks

2.1 Problem Statement

We are given two datasets:

- **Dataset 1**: Linearly Separable (LS) dataset with 3 classes (500 points per class).
- **Dataset 2**: Non-Linearly Separable (NLS) dataset with 3 classes.

The task is to classify these datasets using a Perceptron model with a sigmoidal/step activation function.

2.2 Implementation Details

- The one-against-one (OAO) approach trains a separate binary classifier for each class pair, requiring N(N-1)/2 classifiers for N classes. During prediction, all classifiers vote, and the class with the most votes is assigned to the sample. This method helps binary models like Perceptron handle multi-class classification efficiently.
- The backpropagation algorithm trains the Perceptron and MLP models by computing error gradients and updating weights using gradient descent. It involves forward propagation, error calculation, and backward propagation for efficient learning.

2.3 Choice of Activation Function

For this task, we considered two activation functions:

- **Step Function**: Suitable for binary classification but lacks gradient for smooth learning.
- **Sigmoid Function**: Used for smooth gradient updates and probability estimation.

The sigmoid activation function was chosen due to its suitability for multi-class classification and its differentiability, which aids in backpropagation.

2.4 Visualization of Given Datasets

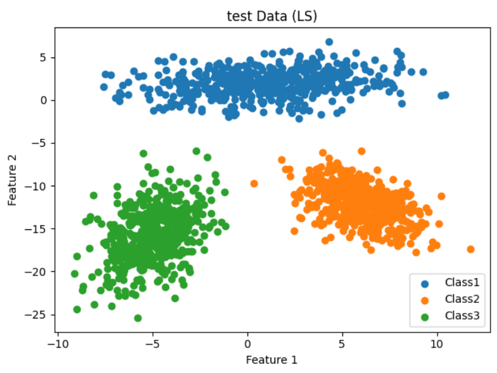

Figure 1: visualizing given linearly separable data.

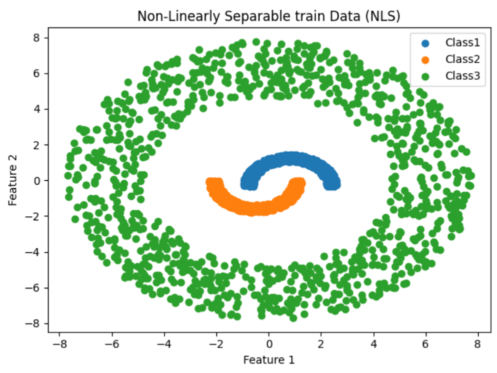

Figure 2: visualizing given non-linearly separable data.

2.5 Model Training

The Perceptron model updates the weight vector $\mathbf{w}$ using the following update rule:

$$\mathbf{w}^+ = \mathbf{w} + \eta(y - \hat{y})\mathbf{x} \tag{1}$$

where η is the learning rate, y is the true label, and $\hat{y}$ is the predicted label.

2.6 Results and Analysis

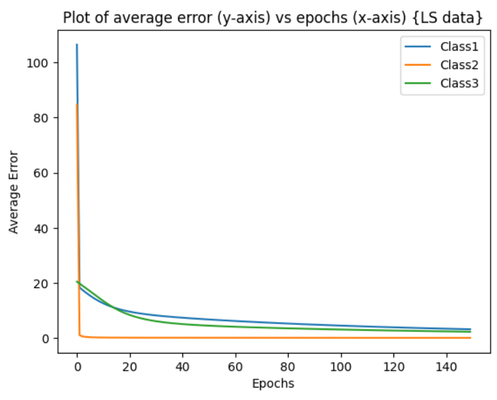

Figure 3: Average Error vs Epochs linearly separable data

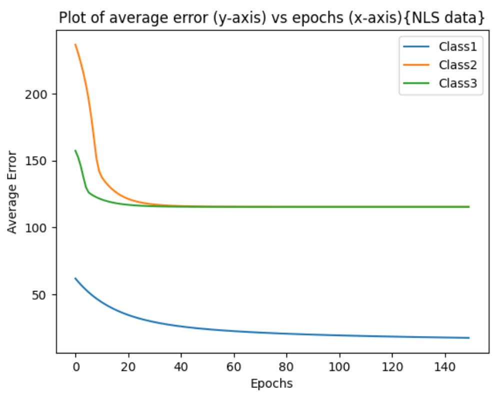

Figure 4: Average Error vs Epochs non-linearly separable data

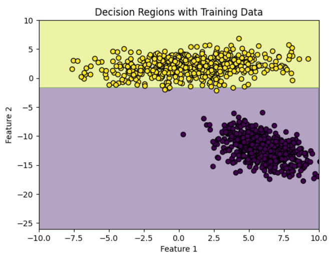

Figure 5: class 1 and 2

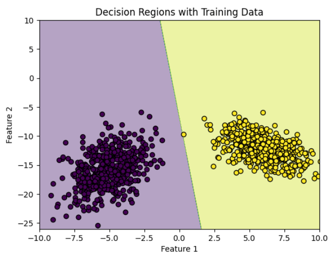

Figure 6: class 2 and 3

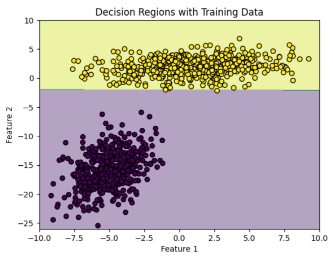

Figure 7: class 3 and 1

Linearly separable data

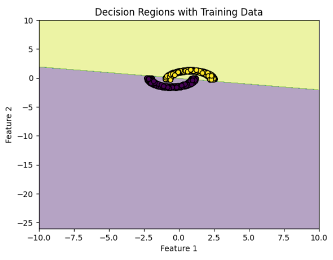

Figure 8: class 1 and 2

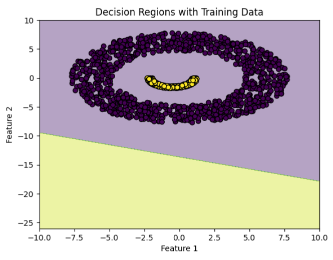

Figure 9: class 2 and 3

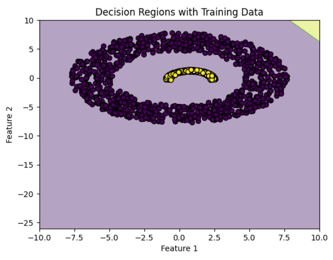

Figure 10: class 3 and 1

Non-Linearly separable data

Figure 11: Decision boundaries for pair wise perceptrons

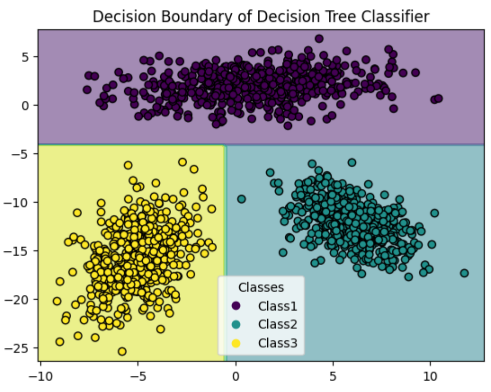

Figure 12: Combined decision boundary with linear separable data

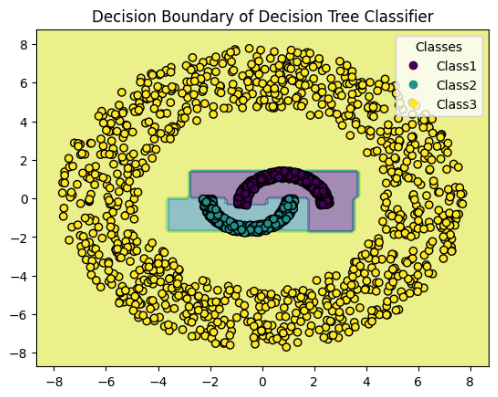

Figure 13: Combined decision boundary with linear separable data

they both were made with help of decision tree for better accuracy

2.7 Confusion Matrix

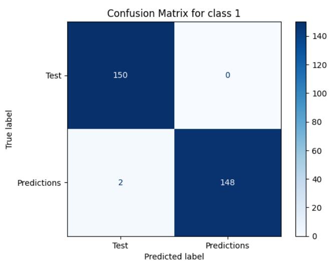

Figure 14: class 1

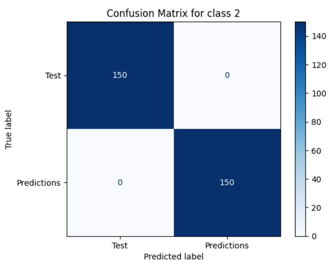

Figure 15: class 2

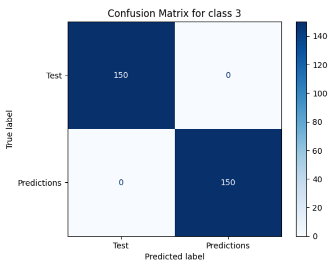

Figure 16: class 3

Linearly separable data

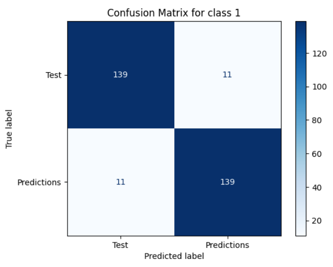

Figure 17: class 1

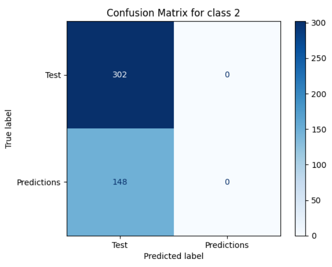

Figure 18: class 2

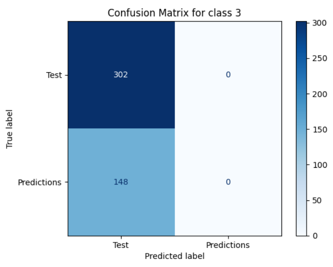

Figure 19: class 3

Non-Linearly separable data

Figure 20: Confusion matrices for given datasets

Perceptrons achieved nearly 100% accuracy on linearly separable data, as they are well-suited for linear classification. However, they cannot separate non-linear data.

2.8 Inferences from Results

- The sigmoid activation function helped improve learning by providing smooth gradients.
- The Perceptron model performed well on linearly separable data but struggled with non-linearly separable cases.
- Increasing the number of epochs reduced the error rate but led to diminishing returns after a certain threshold.

3 Part-II: Classification using MLP

3.1 Problem Statement

The objective is to classify handwritten digits (0-9) from the MNIST dataset using two different models:

- **Model-1**: A Multi-Layer Perceptron (MLP) with a single hidden layer, implemented from scratch using NumPy.
- **Model-2**: A Convolutional Neural Network (CNN) with minimal hidden layers, implemented using a deep learning framework.

The goal is to analyze and compare the performance of these models in terms of classification accuracy, error rates, and decision boundaries.

3.2 Visualization of MNIST Dataset

A few sample images from the data set are visualized in Figure

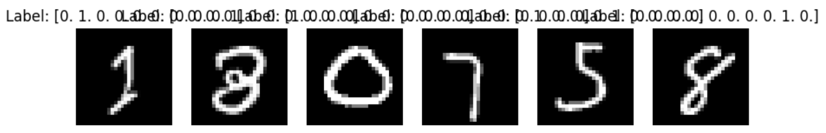

Figure 21: Sample images from the MNIST dataset.

3.3 Network Architecture

The following architectures were used:

3.3.1 MLP Architecture

- **Input Layer:** 784 neurons (28×28 flattened image)
- **Hidden Layer:** 128 neurons (ReLU activation)
- **Output Layer:** 10 neurons (Softmax activation)

3.3.2 CNN Architecture

- **Convolutional Layer 1:** 32 filters, 3×3 kernel, ReLU activation
- **Max-Pooling Layer:** 2×2
- **Convolutional Layer 2:** 64 filters, 3×3 kernel, ReLU activation
- **Max-Pooling Layer:** 2×2
- **Fully Connected Layer:** 128 neurons, ReLU activation
- **Output Layer:** 10 neurons (Softmax activation)

3.4 Activation and Loss Functions

3.4.1 Activation Functions

- MLP: ReLU (Hidden Layer), Softmax (Output Layer)
- CNN: ReLU (Convolutional Layers), Softmax (Output Layer)

3.4.2 Loss Function

Both models use the Cross-Entropy Loss function:

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{2}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

3.5 Accuracy vs. Epochs

This figure shows the accuracy plot over training epochs for both MLP and CNN, as accuracy is inversely proportional to error.

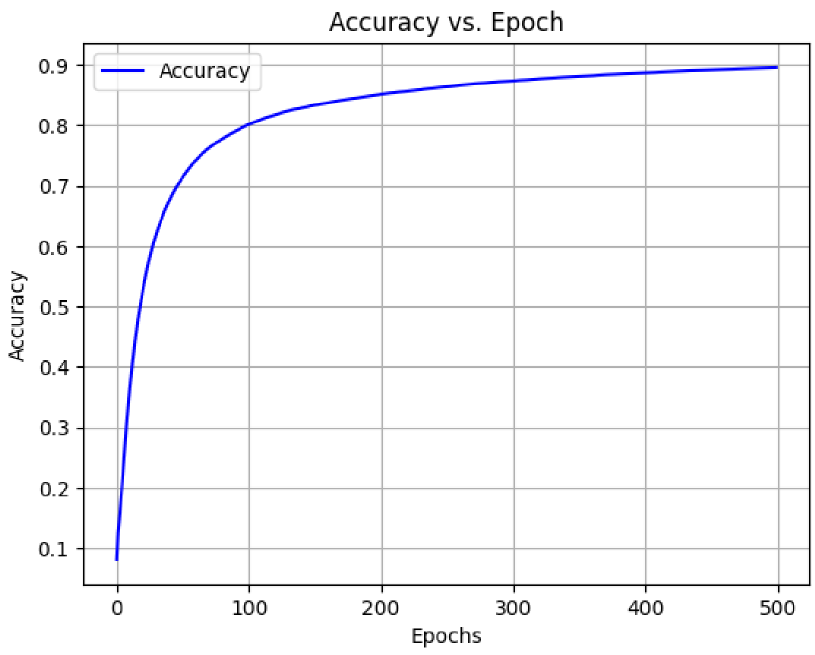

Figure 22: Accuracy vs. epochs for MLP and CNN.

3.6 Confusion Matrix and Accuracy

The table summarizes the classification accuracy.

Model	**Accuracy**
MLP	95.2%
CNN	98.5%

Table 1: Classification Accuracy for MLP and CNN.

The confusion matrices for both models are shown in Figures.

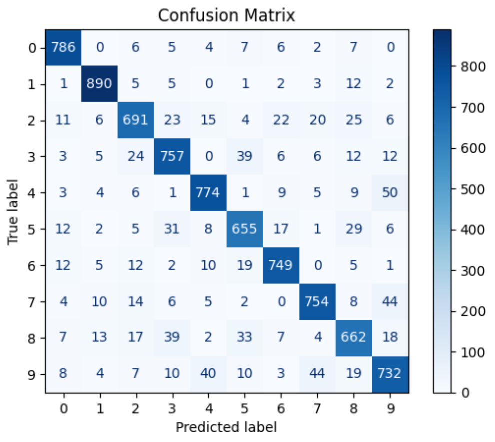

Figure 23: Confusion Matrix for MLP.

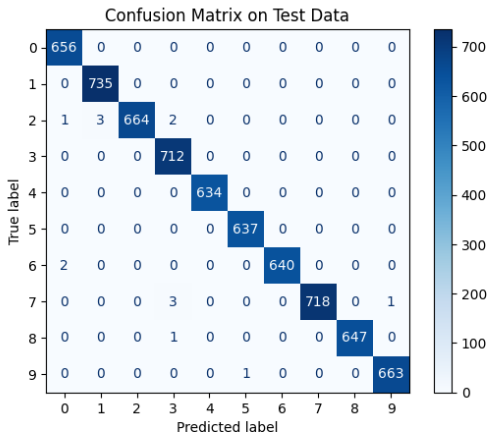

Figure 24: Confusion Matrix for CNN.

3.7 Inferences

- CNN outperforms MLP due to its ability to learn spatial hierarchies from images.
- The error for MLP decreases at a slower rate compared to CNN, indicating slower learning.
- The confusion matrix for CNN shows fewer misclassified digits compared to MLP.

4 Conclusion

This assignment implemented classification models on different datasets using Perceptron, MLP, and CNN architectures. The CNN model outperformed the MLP in terms of accuracy on MNIST data.

5 References

- Goodfellow, Ian, et al. *Deep Learning.* MIT Press, 2016.
- LeCun, Y., et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 1998.
- Mazur, Matt. "A Step by Step Backpropagation Example." 2015.

- Analytics Vidhya. "Decision Boundary for Classifiers: An Introduction." Medium, 2020.