1. Introduction

In this task, a Fully Connected Neural Network (FCNN) was implemented from scratch using NumPy to classify two synthetic datasets: a linearly separable dataset (Dataset-1) and a non-linearly separable dataset (Dataset-2). The objective was to study the behaviour of neural networks on datasets with different levels of complexity, analyse training and validation performance, visualize decision regions, and understand internal neuron activations.

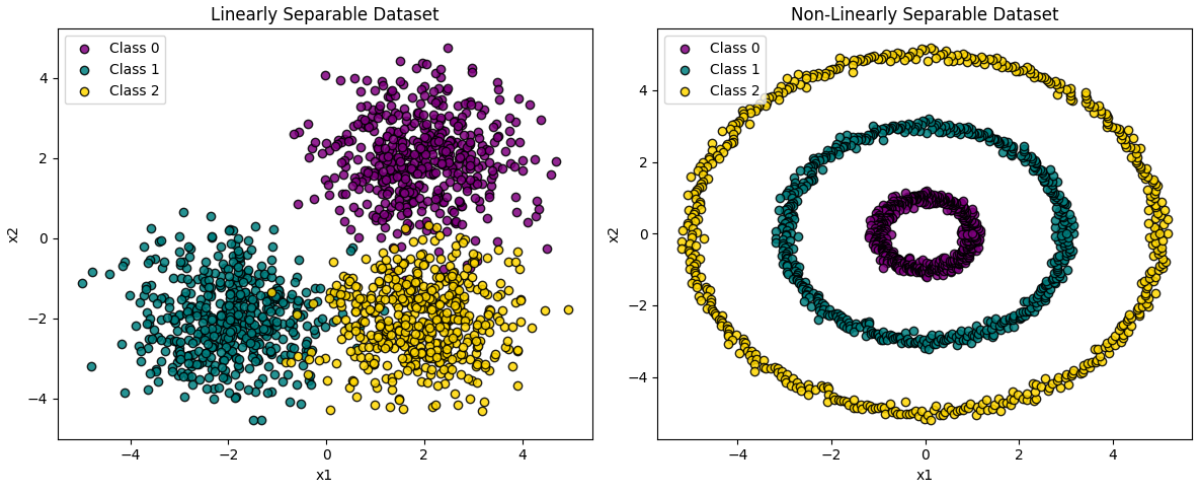
2. Dataset Description

2.1 Dataset-1: Linearly Separable Dataset

Dataset-1 consists of three classes in a two-dimensional input space. Each class forms a cluster that can be separated using linear decision boundaries. This dataset is suitable for evaluating a shallow neural network.

2.2 Dataset-2: Non-Linearly Separable Dataset

Dataset-2 contains three classes arranged in concentric circular patterns, making the data non-linearly separable. This dataset requires a deeper network architecture to correctly model the complex decision boundaries.



3. Data Preprocessing

The dataset was split into:

- **Training set (60%)**
- **Validation set (20%)**
- **Test set (20%)**

Class-wise splitting was performed to preserve class balance.
Class labels were converted into **one-hot encoded vectors** before training to support multi-class classification.

4. Activation Function

The **sigmoid activation function** was used for all hidden and output layers:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The sigmoid function introduces non-linearity and outputs values in the range (0, 1), which is suitable for classification tasks

5. Forward Propagation

For each training example, forward propagation was performed as follows:

Hidden Layer Computation

$$z^{(1)} = XW^{(1)} + b^{(1)}$$
$$a^{(1)} = \sigma(z^{(1)})$$

Output Layer Computation

$$z^{(2)} = a^{(1)}W^{(2)} + b^{(2)}$$
$$a^{(2)} = \sigma(z^{(2)})$$

For Dataset-2, an additional hidden layer was included using the same computations.

6. Loss Function

The **Squared Error Loss** was used:

$$L = \frac{1}{2}\sum(y - \hat{y})^2$$

This loss measures the difference between predicted output and true one-hot encoded labels.

7. Backpropagation

Backpropagation was used to update weights and biases using the chain rule.

Output Layer Error

$$\delta^{(out)} = (y - \hat{y}) \cdot \sigma'(z^{(out)})$$

Hidden Layer Error

$$\delta^{(hidden)} = \delta^{(out)}W^T \cdot \sigma'(z^{(hidden)})$$

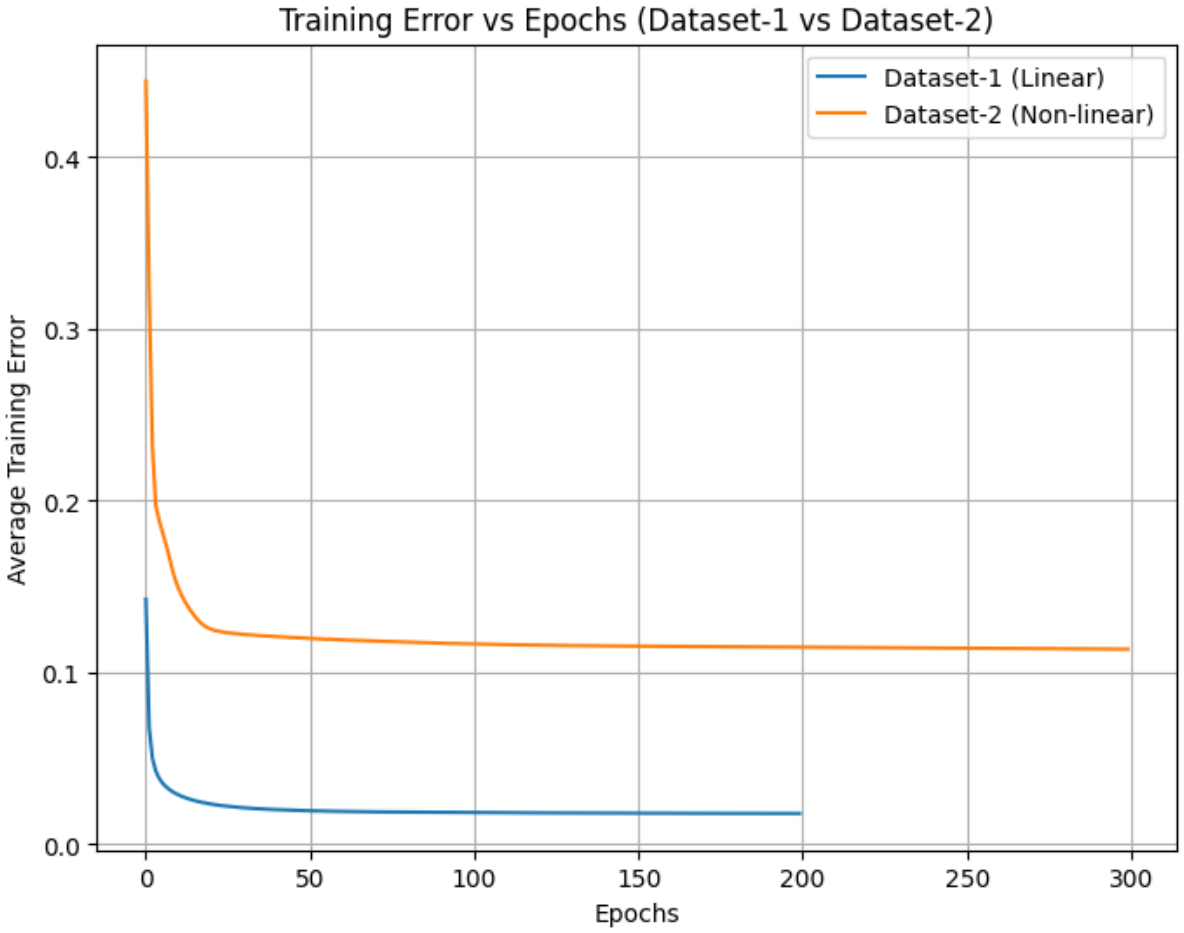
Weight Updates

$$W = W + \eta \cdot (\text{input})^T \cdot \delta$$

where $\eta = 0.01$is the learning rate.

8. Training Procedure

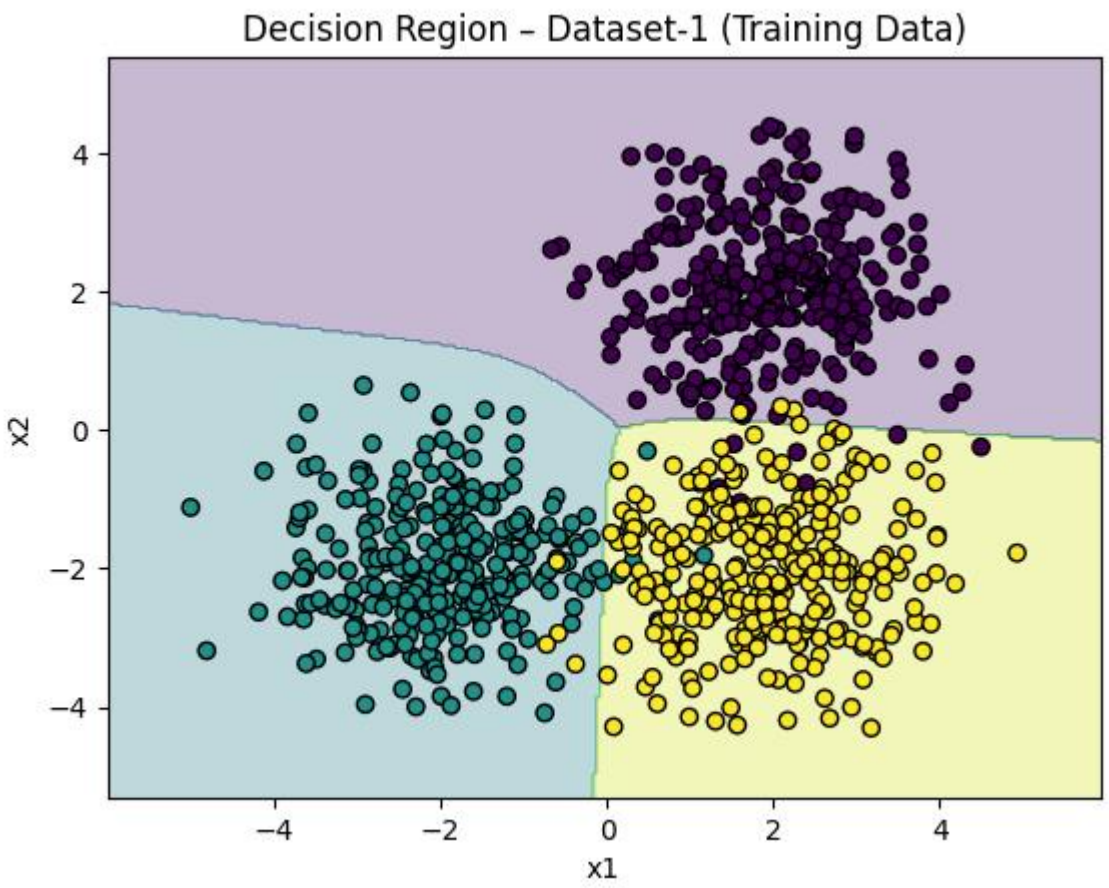
- Training was performed for multiple epochs
- In each epoch:
    - Forward propagation
    - Loss computation
    - Backpropagation
    - Weight updates using gradient descent
- Average training error was recorded per epoch

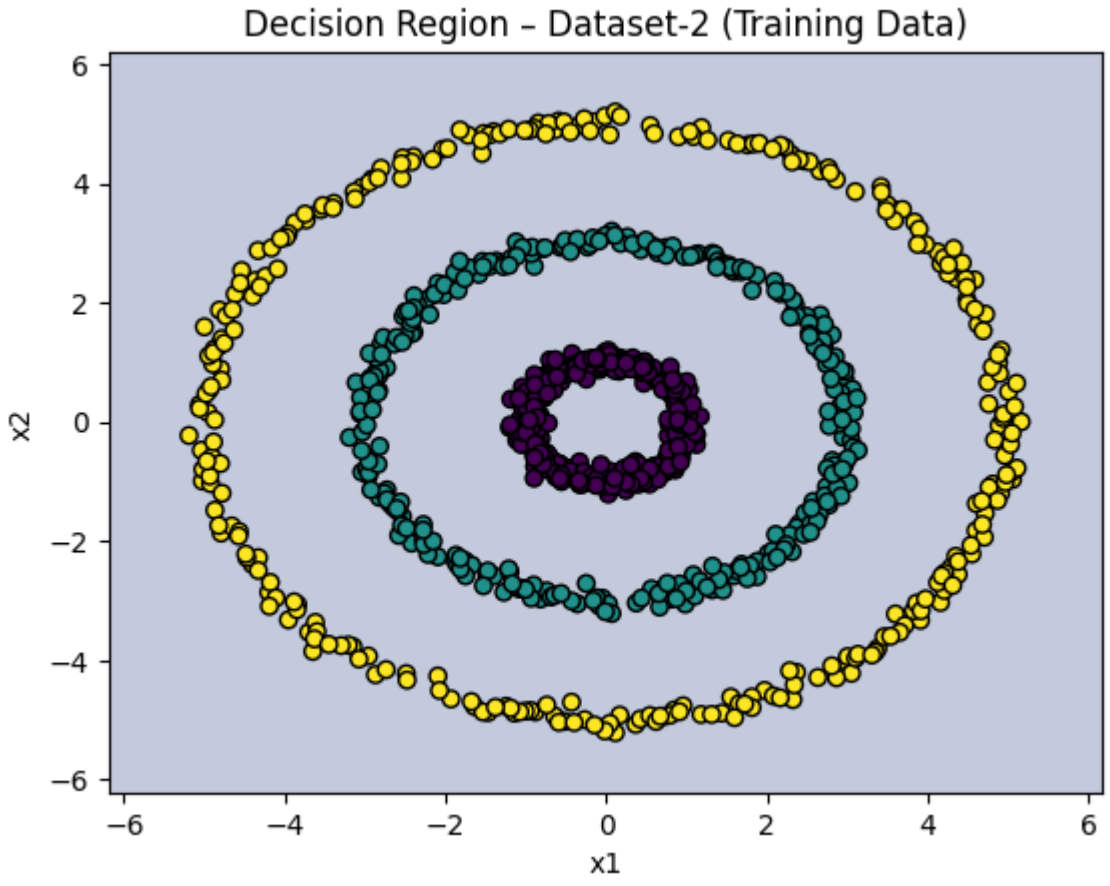


9. Decision Region Visualization

Decision regions were plotted to visualize how the trained model separates different classes in input space.

- Dataset-1 shows clear linear decision boundaries
- Dataset-2 shows complex non-linear boundaries learned by deeper network


Decision Region – Dataset-1 (Training Data)
x2
x1
4
2
0
−2
−4
−4
−2
0
2
4


Decision Region – Dataset-2 (Training Data)
x2
x1
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

10. Observations

- Shallow network performed well for linearly separable data
- Deeper network was required for non-linear data
- Validation and test accuracy confirmed good generalization
- Decision regions and neuron activations clearly reflect dataset complexity

11. Conclusion

This task demonstrates that neural network architecture must be chosen based on data complexity. A simple FCNN is sufficient for linear problems, whereas non-linear datasets require deeper architectures. Visualization of decision regions and neuron outputs provides valuable insights into how neural networks learn internal representations.