Assignment 1 Part B: Multi-Layer Perceptrons and Advanced Techniques

Deep Learning - EE569

Motaz M Alharbi
2190203271

Instuctor: Dr. Nuri Benbarka

Contents

1 Introduction

This report extends the work from Part A, transitioning from simple logistic regression to more complex neural network architectures. The primary objectives are to address non-linearly separable problems, implement a Multi-Layer Perceptron (MLP), automate the network construction process, and apply these advanced techniques to a real-world multi-class classification problem. This investigation covers the implementation of an MLP to solve the XOR problem, refactoring the codebase for scalability, and conducting a systematic analysis of activation functions, weight initialization schemes, and L2 regularization on the MNIST handwritten digit dataset.

2 Task 1: The XOR Problem

2.1 Task Description

The first task was to demonstrate the limitations of a linear model. A dataset was generated where the classes are not linearly separable, mimicking the classic XOR problem. This was achieved by positioning four Gaussian distributions such that a single straight line cannot separate the two classes. The objective was to train the logistic regression model from Part A on this dataset and evaluate its performance.

2.2 Model and Training

The model used is the single-layer logistic regression unit developed in Part A, consisting of an `Input` node, a `Linear` node, and a `Sigmoid` activation, followed by a Binary Cross-Entropy (BCE) loss function. The model was trained for 100 epochs with a learning rate of 0.02.

2.3 Results and Discussion

The results, summarized in Figure 1, confirm the theoretical limitations of the model. The final test accuracy was a mere **31.25%**, which is significantly worse than random guessing. The decision boundary plot clearly shows a single linear plane attempting to separate the four clusters. The confusion matrix reveals a strong bias, with the model incorrectly classifying the vast majority of "True 0" samples as "Predicted 1". This outcome definitively demonstrates that linear models are insufficient for solving non-linearly separable problems.

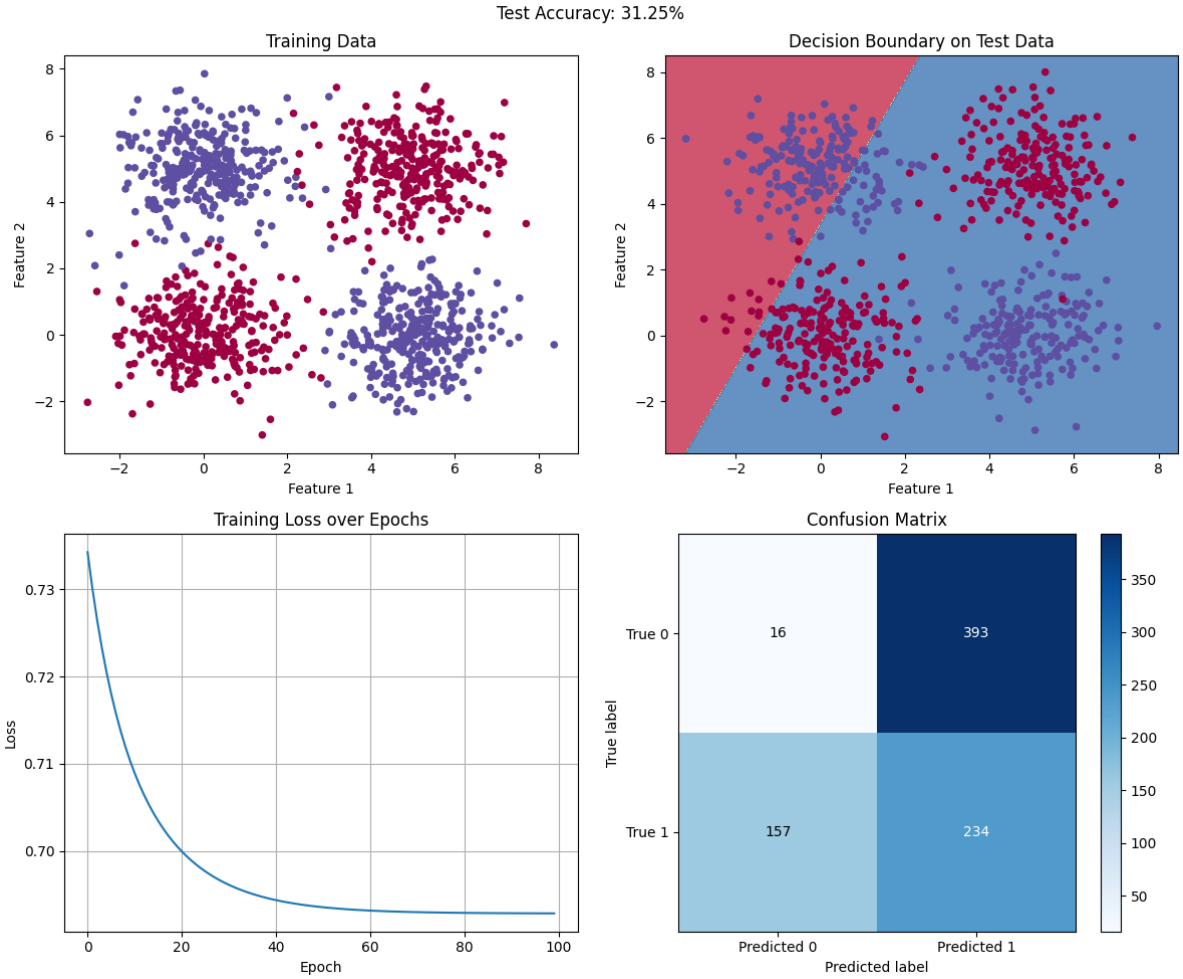


Figure 1: Combined results for Task 1, showing the non-linearly separable training data, the inadequate linear decision boundary, the training loss curve, and the resulting confusion matrix. The low accuracy of 31.25% highlights the model's inability to solve the XOR problem.

3 Task 2: Multi-Layer Perceptrons (MLP)

3.1 Task Description

To overcome the limitations of the linear model, a Multi-Layer Perceptron (MLP) was designed and implemented. The specific architecture required was a network with two hidden layers, each containing 20 neurons, using the `Sigmoid` activation function. This MLP was then trained on the same XOR dataset from Task 1.

3.2 Model Architecture

The MLP was constructed using the existing `Linear` and `Sigmoid` nodes. The architecture is as follows:

$$\text{Input} \to \text{Linear}(2,\ 20) \to \text{Sigmoid} \to \text{Linear}(20,\ 20) \to \text{Sigmoid} \to \text{Linear}(20,\ 1) \to \text{Sigmoid} \to \text{BCE Loss}$$

The model was trained for 1000 epochs with a learning rate of 0.8.

3.3 Results and Discussion

The MLP achieved a high test accuracy of **96.25%**, demonstrating its capacity to solve the XOR problem. As shown in Figure 2, the model learned a complex, non-linear decision boundary that successfully separates the four data clusters. The training loss curve shows a steady and smooth decrease, indicating effective learning. The confusion matrix confirms the high accuracy, with very few misclassifications. This success is attributed to the hidden layers, which transform the input features into a new representation space where the classes become linearly separable.

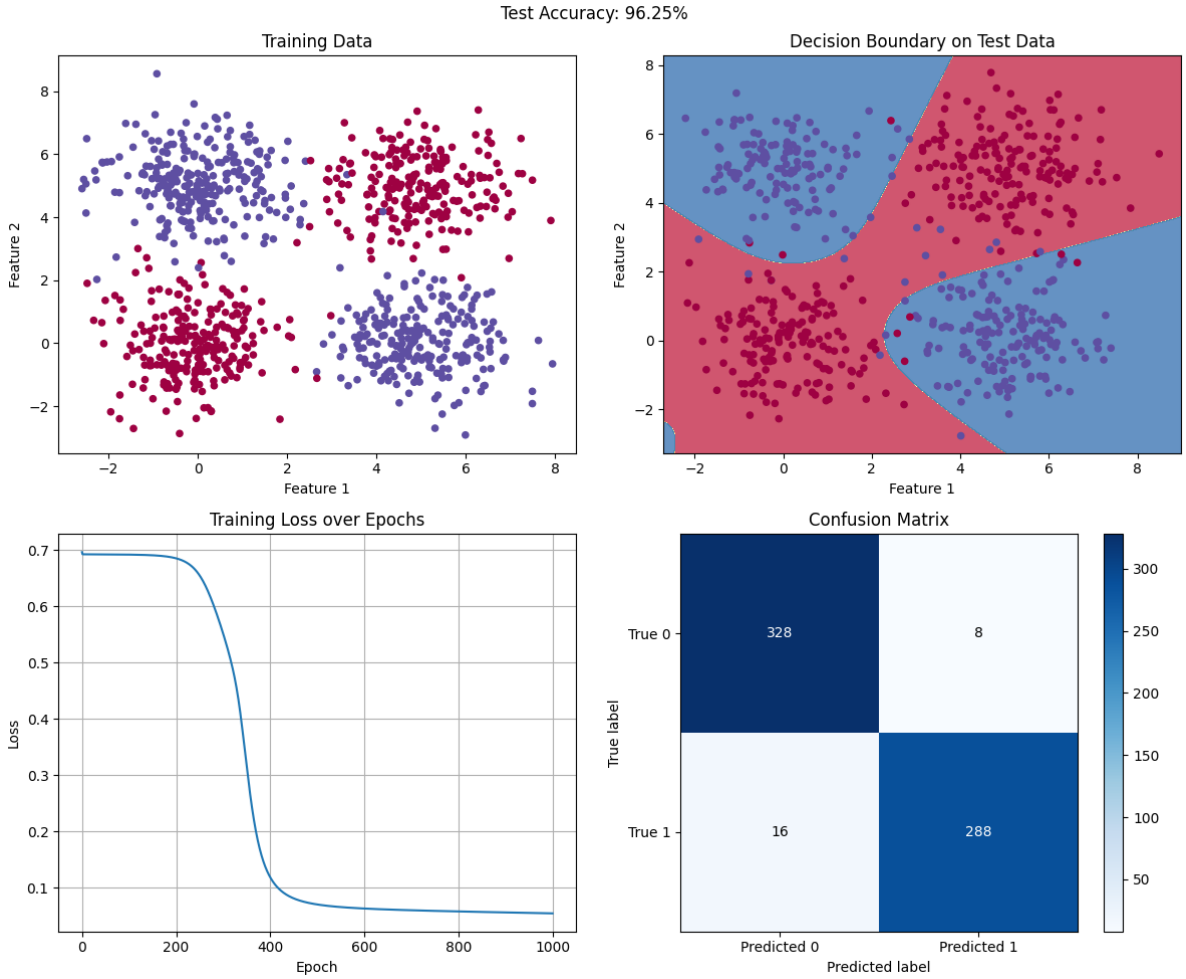


Figure 2: Combined results for the MLP in Task 2. The learned decision boundary is non-linear and effectively separates the XOR classes, resulting in a high test accuracy of 96.25%.

4 Task 3: Code Refactoring and Automation

4.1 Task Description

This task focused on improving the code's quality and maintainability by automating the creation of the neural network architecture. The manual definition of parameter nodes and the graph list was replaced with a more scalable, automated approach.

4.2 Implementation Details

An `AutomatedLinear` class was created to encapsulate parameter initialization, and a `topological_sort` function was implemented to automatically build the execution graph. These changes allow for the dynamic creation of networks of arbitrary depth and width with minimal manual setup.

4.3 Results

A deeper network with three hidden layers of 100 neurons each was built using the new automated framework and trained for 4000 epochs. The results, shown in Figure 3, demonstrate that the automated system functions correctly, producing an excellent model with **98.28%** test accuracy. Interestingly, the training loss curve shows several large spikes, which may indicate some training instability, possibly due to the high learning rate (0.9) for a deeper network. However, the model consistently recovered and converged to a highly effective solution, validating the refactored code.

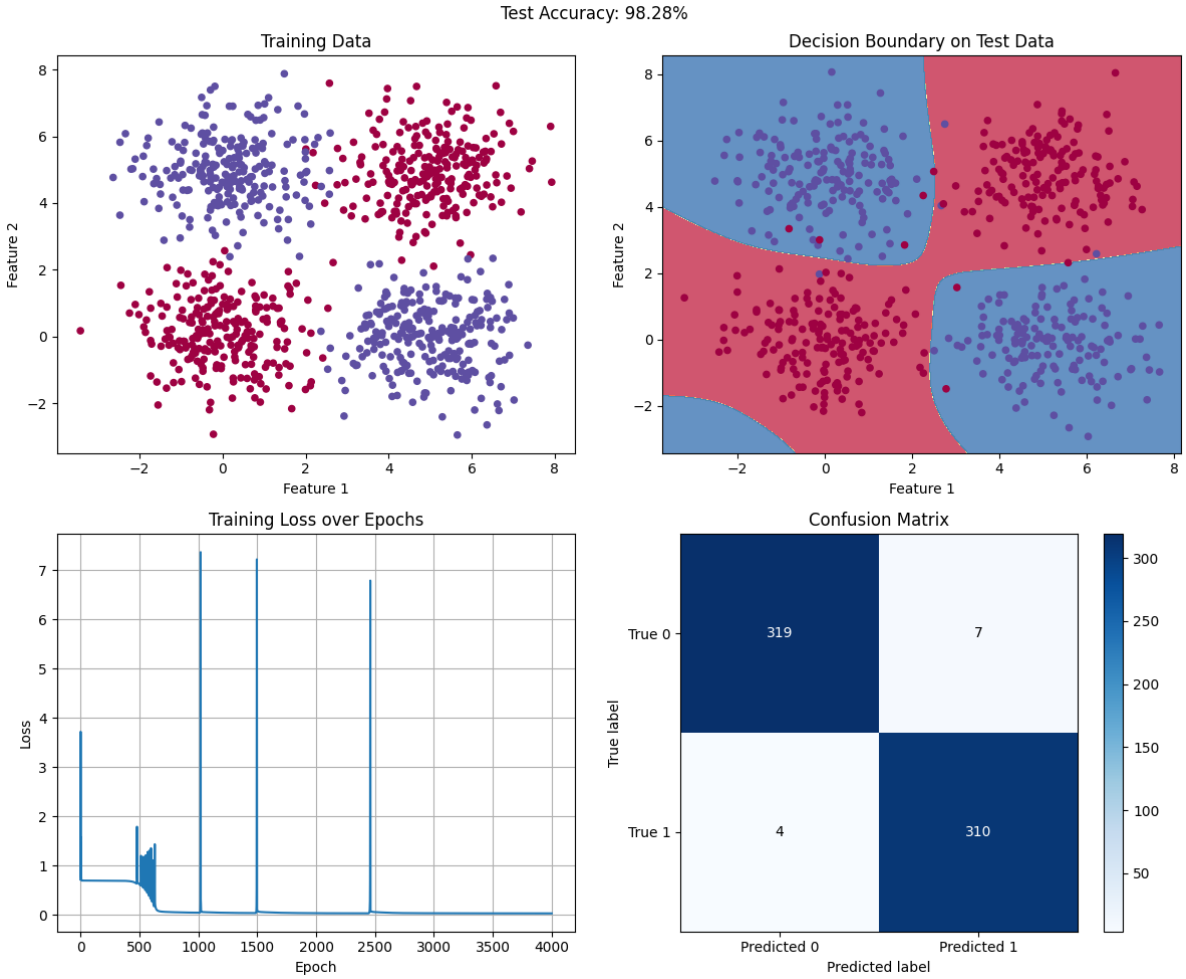


Figure 3: Results from the deeper network built using the automated framework. The model learns a robust non-linear boundary, achieving 98.28% accuracy and validating the refactored code.

5 Task 4: Handwritten Digit Classification on MNIST

5.1 Task Description

The final task was to apply the custom framework to classify handwritten digits from the MNIST dataset. This involved implementing a multi-class classification model and systematically testing the effects of different activation functions, weight initialization methods, and L2 regularization.

5.2 Experimental Setup

The MNIST dataset was split into 60% for training and 40% for testing. The base model architecture consisted of one hidden layer with 64 neurons, a `Softmax` output layer, and a Cross-Entropy (CE) loss function.

5.3 Results and Discussion

The final accuracies for all experiments are summarized in Table 1.

Table 1: Test Accuracy for Different MNIST Experiments

Experiment Category	**Configuration**	**Test Accuracy (%)**
Activation Function	Sigmoid	97.63
	Tanh	97.77
	ReLU	96.66
Weight Initialization	Random	97.21
	Xavier	97.77
	He	96.94
L2 Regularization	False (Baseline)	97.77
	True	**98.47**

5.3.1 Activation Functions

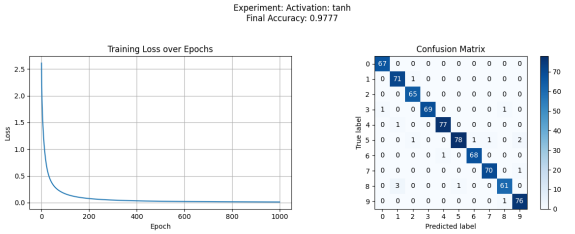
All three activation functions performed exceptionally well. Contrary to common expectations for deeper networks, for this shallow architecture, **Tanh (97.77%)** and **Sigmoid (97.63%)** slightly outperformed ReLU (96.66%). This highlights that the optimal choice of activation can be problem-dependent.

5.3.2 Weight Initialization

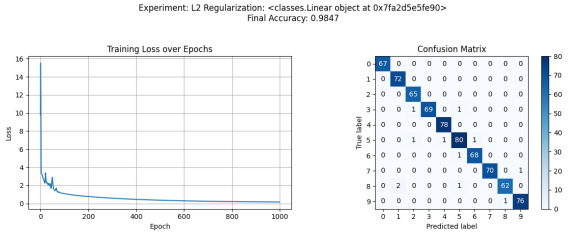
The choice of weight initialization had a noticeable impact, with **Xavier (97.77%)** providing the best performance, aligning perfectly with its design for saturating activation functions like Tanh and Sigmoid. While all methods led to good results, this demonstrates the benefit of using a theoretically-grounded initialization scheme.

5.3.3 L2 Regularization

Adding L2 regularization provided the most significant performance boost, increasing the accuracy from the best baseline of 97.77% to a new high of **98.47%**. This result clearly shows that regularization was effective at preventing overfitting on the training data, allowing the model to generalize better to the unseen test set.



(a) Best Activation: Tanh (97.77%)



(b) Best Overall: With L2 Regularization (98.47%)

Figure 4: Selected results from the MNIST experiments. (a) The performance using the Tanh activation function with Xavier initialization. (b) The best-performing model, which included L2 regularization.

6 Conclusion

This assignment successfully demonstrated the progression from basic linear models to robust, automated Multi-Layer Perceptrons. It was empirically shown that linear models fail on non-linear problems like XOR, while MLPs can learn complex decision boundaries to solve them with high accuracy (over 96%). Automating the network construction process proved essential for scalability and maintainability, enabling the creation of deeper models that achieved even higher performance (98.28%). Finally, the application to the MNIST dataset highlighted the practical importance of modern neural network components; a combination of Tanh activation, Xavier initialization, and particularly L2 regularization, provided the best performance, achieving a final test accuracy of **98.47%** and showcasing the effectiveness of the custom-built deep learning framework.