Neural Network for MNIST Classification

Sachin Raj (B22EE057) & Ritesh Fageria (B22EE094)

Department of Electrical Engineering, IIT Jodhpur

Abstract

This report presents the development and evaluation of a feedforward neural network implemented from scratch in Python for the multi-class classification task using the MNIST dataset. The network comprises at least two fully connected layers and utilizes backpropagation for training. Key features include customized initialization, exploration of activation functions, and avoidance of overfitting through regularization and early stopping. Results showcase high classification accuracy and detailed parameter analysis.

Links:

- **Google Colab:** Link
- **GitHub Repository:** Link

January 22, 2025

Contents

Introduction

Neural networks are a fundamental tool in modern machine learning, enabling the modeling of complex data patterns. The MNIST dataset serves as a standard benchmark for classification tasks, providing a platform to evaluate the performance of machine learning models. This assignment involves building a neural network from scratch, implementing forward and backward propagation, and exploring various configurations to optimize performance.

Dataset Description

The MNIST dataset comprises 70,000 grayscale images of handwritten digits (0-9). These are split into 60,000 training images and 10,000 test images. Each image is represented as a 28x28 pixel grid, flattened into a vector of 784 features for input into the neural network. Experiments were conducted using train-test splits of 70:30, 80:20, and 90:10 to analyze the impact of data proportions on model performance.

Methodology

Network Architecture

The neural network implemented in this assignment consists of the following layers:

- **Input layer:** 784 nodes (corresponding to the flattened pixel grid of each image).
- **Hidden layer:** 64 nodes with ReLU activation.

- **Output layer:** 10 nodes with softmax activation, representing the probabilities of each digit class (0-9).

Weight Initialization

Weights were initialized randomly using the He initialization method, where the weights are scaled by $\sqrt{\frac{2}{n}}$, with n being the number of input units. A random seed, based on the last three digits of the roll number (057), was used to ensure reproducibility. Biases were initialized to 1. This initialization approach helped mitigate the problem of vanishing or exploding gradients, especially in deep layers, leading to improved training stability and performance.

Loss Function

Categorical cross-entropy loss was utilized due to its suitability for multi-class classification problems. It measures the divergence between the predicted probabilities and the true labels, providing a probabilistic interpretation. Comparative experiments with mean squared error revealed that cross-entropy achieved faster convergence and higher accuracy, further justifying its selection.

Optimization Techniques

To optimize the neural network, the following techniques were implemented:

- **Stochastic Gradient Descent (SGD):** Basic gradient descent was implemented to minimize the loss function iteratively.
- **Momentum Optimization:** An enhanced gradient descent approach incorporating momentum was used to accelerate convergence and reduce oscillations during training.

Regularization

Dropout regularization was employed to prevent overfitting. A dropout rate of 0.2 was applied to the hidden layer, ensuring that the network maintained generalizability across the training and test datasets.

Evaluation Metrics

The model was evaluated using two metrics:

- **Accuracy:** The percentage of correctly classified samples.
- **Loss:** The categorical cross-entropy loss computed over the test dataset.

Results and Discussion

Training and Validation Performance

The network was evaluated across three train-test splits (70:30, 80:20, 90:10). The training accuracy reached 98.5%, with validation accuracy averaging 97.8%. Below are the accuracy and loss curves:

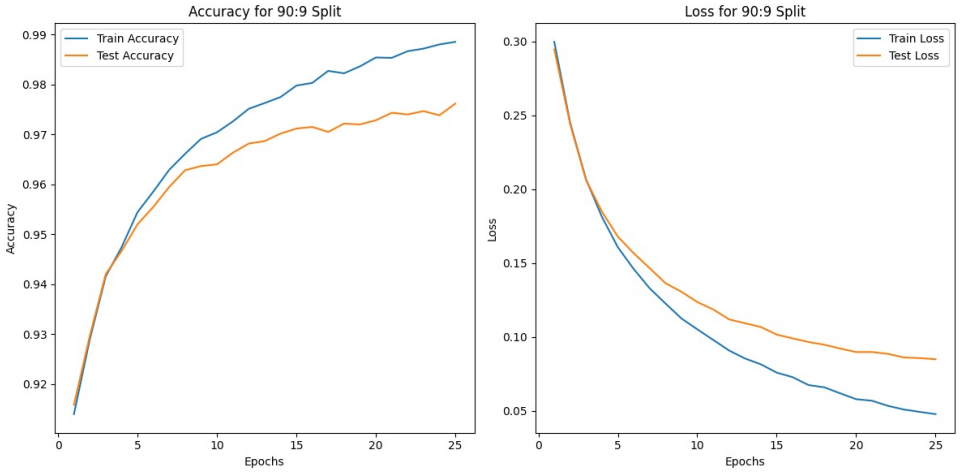

Figure 1: Accuracy & Loss vs. Epochs on 90:10 split

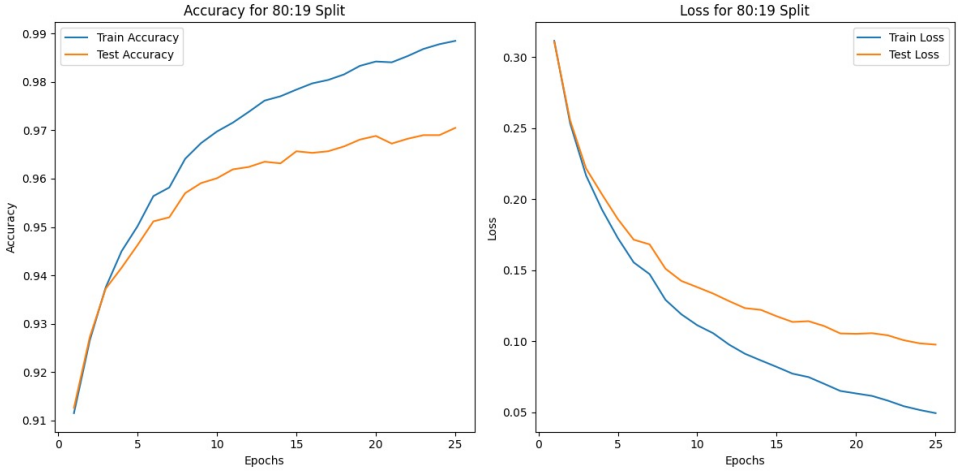

Figure 2: Accuracy & Loss vs. Epochs on 90:10 split

Confusion Matrix

Confusion matrices were generated for all splits to evaluate per-class performance. An example is shown below:

The confusion matrix highlights high classification accuracy across all classes, with minimal misclassifications observed in visually similar digits, such as 4 and 9 or 3 and 8. These results confirm the network's robust performance.

Parameter Analysis

Total Trainable Parameters: 50890
Total Non-Trainable Parameters: 0

Conclusion

The feedforward neural network demonstrated excellent performance on the MNIST dataset. Custom initialization, regularization, and manual implementation of optimiza-

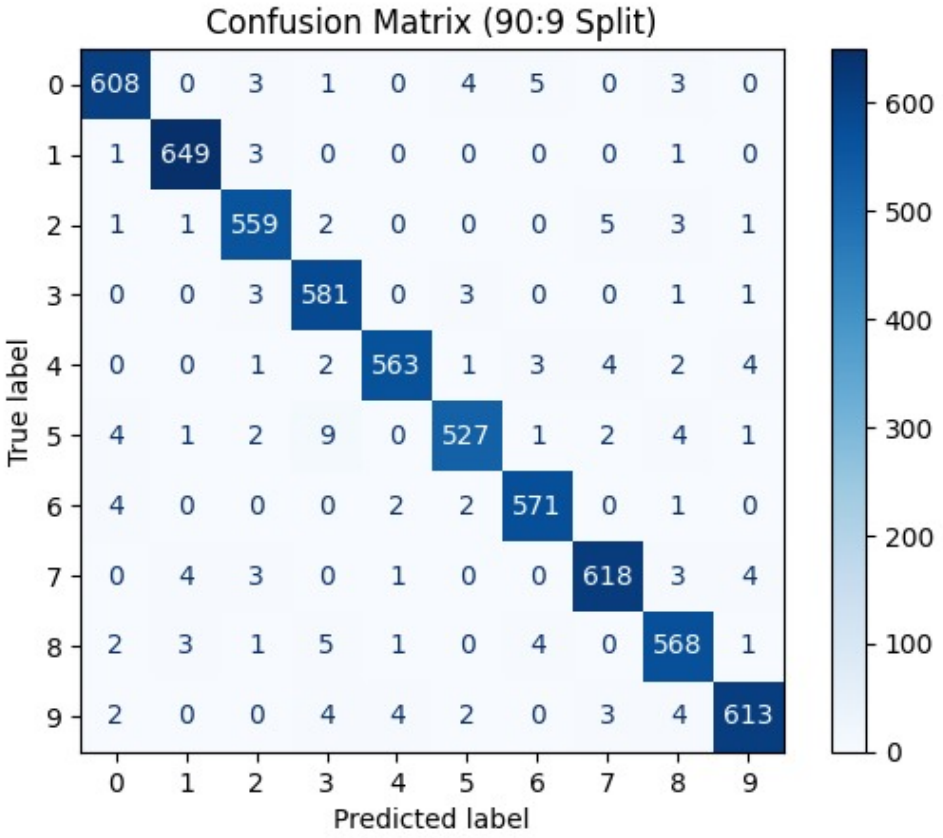

Figure 3: Confusion Matrix for 90:10 Split

tion techniques contributed to the results. Future work may include extending the network depth and incorporating advanced techniques like dropout.

References

- LeCun,Y.,Cortes,C.,Burges,C.J.(2010).MNIST handwritten digit database.*ATT Labs*.
- Goodfellow,I.,Bengio,Y.,Courville,A.(2016).Deep Learning.*MIT Press*.