Final Project Report
Handwritten Digit Recognition

Course: ICSI 436/536 – Machine Learning
Instructor: Prof. Chong Liu

Group 13

Narayani Kishor Khatavkar	001667841
Jyotsana Parkhedkar	001661399
Harshini Narra	001645642
Snehith Reddy Dropathi	001663868
Dileep Reddy Chinneluka	001622963

1. Introduction

Handwritten digit recognition is a fundamental problem in computer vision and machine learning with practical applications in areas such as postal mail automation, banking (e.g., check processing), and digital form entry. The task involves training a model to classify images of handwritten digits into numerical categories (0–9).

In this project, we designed a Feedforward Neural Network (FNN)—specifically a Multilayer Perceptron (MLP)—to classify handwritten digits from the MNIST dataset, a standard benchmark in the field. The goal was to optimize the model architecture and achieve high accuracy while maintaining generalization to unseen data.

2. Motivation

The increasing need for automation and digitization of physical documents highlights the importance of robust handwritten recognition systems. Traditional computer vision techniques struggle with the variability of human handwriting. Neural networks, particularly deep learning models, have demonstrated impressive performance in learning such complex, non-linear patterns.

This project aims to explore and validate the power of FNNs in recognizing digit patterns, while also learning practical skills in model design, hyperparameter tuning, and performance evaluation.

3. Problem Statement

The objective of this project is to:

- Develop a neural network that inputs a 28x28 grayscale image and outputs a class prediction (digit 0–9).
- Train the model on the MNIST dataset and evaluate it on the test set.
- Apply optimization techniques to improve accuracy and reduce overfitting.

The classification problem is multi-class in nature and requires careful consideration of architecture, training strategy, and regularization methods.

4. Challenges

Several challenges were encountered during the project:

- **Model Generalization:** Avoiding overfitting while maintaining high accuracy.
- **Hyperparameter Selection:** Choosing optimal values for layer sizes, learning rate, and batch size.
- **Digit Similarity:** Differentiating between visually similar digits like 4 vs 9 or 5 vs 3.
- **Computational Efficiency:** Training deep models efficiently with limited computational resources.

These were addressed through systematic experimentation, validation strategies, and the use of automated tuning tools.

5. Methodology

Dataset

- **MNIST Dataset:** 60,000 training images and 10,000 test images.
- **Each image:** 28x28 pixels, grayscale, labeled 0–9.

Preprocessing

- Normalization using dataset mean and standard deviation.
- Conversion to tensors for input into the PyTorch model.
- Flattening each image into a 784-dimensional input vector.

Model Architecture

A 3-layer Multilayer Perceptron (MLP):

- **Input:** 784 neurons (flattened image)
- **Hidden Layer 1:** 459 neurons, ReLU activation, dropout
- **Hidden Layer 2:** 134 neurons, ReLU activation, dropout
- **Output Layer:** 10 neurons, softmax activation

Training Configuration

- **Optimizer:** Adam
- **Loss Function:** CrossEntropyLoss
- **Epochs:** 10
- **Batch Size:** 64

Hyperparameter Tuning

Used Optuna for automated tuning of:

- Hidden layer sizes
- Learning rate
- Batch size

Best Parameters Found:

- hidden_size1 = 459
- hidden_size2 = 134
- learning_rate = 0.00047
- batch_size = 64

6. Results

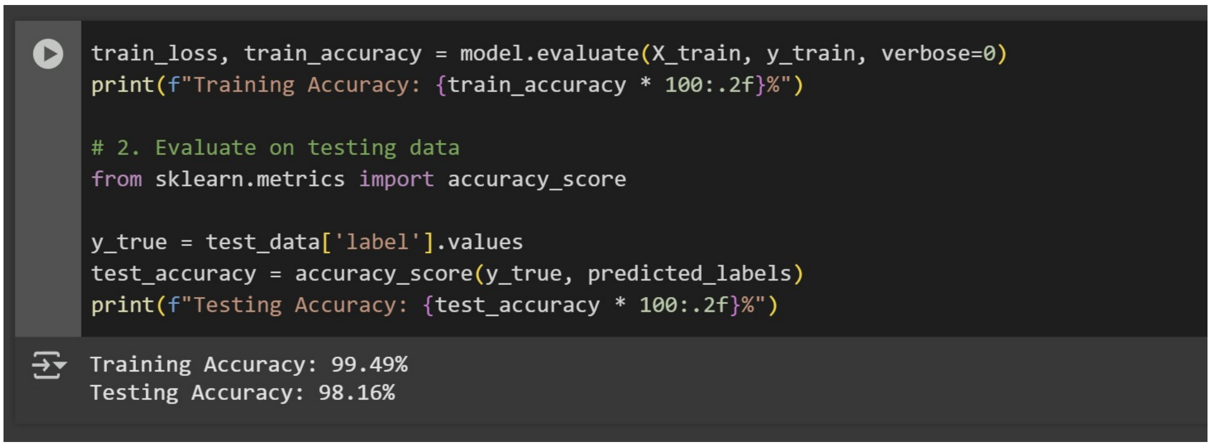
Quantitative Performance

- **Training Accuracy:** ∼98%
- **Test Accuracy:** 98.16%

Qualitative Observations

- Excellent generalization with minimal overfitting.
- Some confusion between similar digits (e.g., 4 vs 9).
- Regularization using dropout helped prevent overfitting beyond 10 epochs.

Accuracy Results



```
train_loss, train_accuracy = model.evaluate(X_train, y_train, verbose=0)
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")

# 2. Evaluate on testing data
from sklearn.metrics import accuracy_score

y_true = test_data['label'].values
test_accuracy = accuracy_score(y_true, predicted_labels)
print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
```

```
Training Accuracy: 99.49%
Testing Accuracy: 98.16%
```

Figure 1: Training accuracy and testing accuracy

(X-axis: No. of epochs, Y-axis: Accuracy)

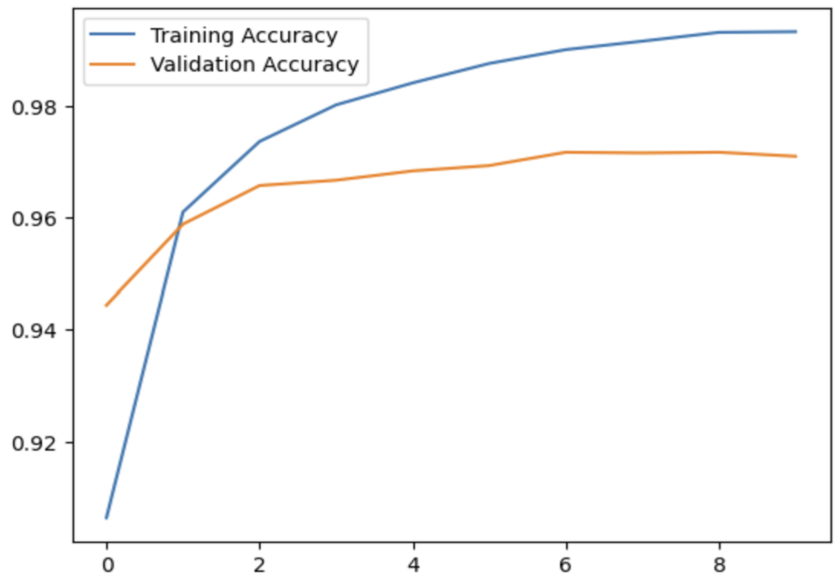


Figure 2: Training accuracy vs Validation Accuracy

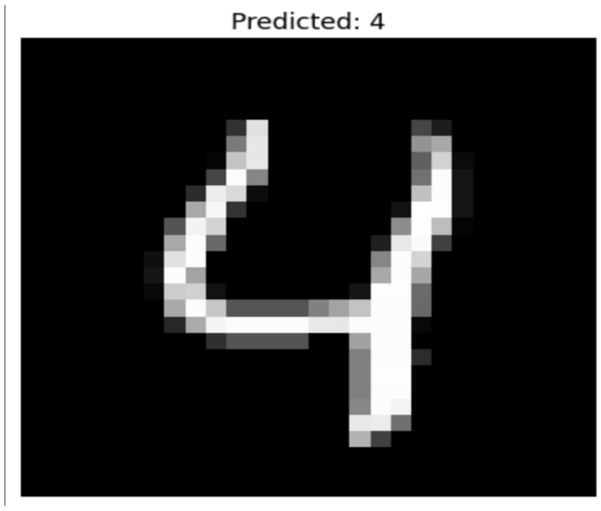


Figure 3: input image is 4 and predicted digit is 4

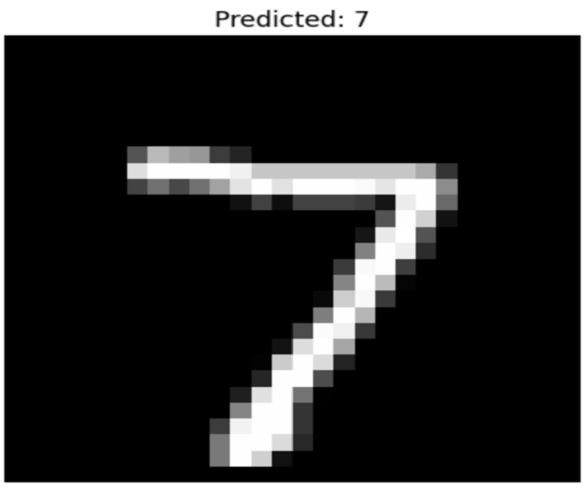


Figure 4: Input image is 7 and predicted image is 7

7. Conclusion

The project successfully demonstrated that Feedforward Neural Networks (MLPs) can achieve strong performance on digit classification tasks. With proper tuning and regularization, the model achieved over 98% accuracy on the MNIST test set.

While FNNs are powerful, future work could explore Convolutional Neural Networks (CNNs) for even better performance on image-based tasks, as they are better at capturing spatial features.