Handwritten Digit Classification using CNN

Pranav PS – 23BCE7250
Rohiith Krishnan R – 23BCE8372
Ganesh M – 23BCE8887

April 5, 2025

Contents

1 Introduction

1.1 Objective of the Project

The main goal of this project is to create a strong and precise handwritten digit recognition system based on Convolutional Neural Networks (CNNs), a type of deep learning models that are specifically designed for image recognition. The system is trained on the MNIST dataset, which contains grayscale images of handwritten digits (0–9). The aim is to train the model to automatically identify and classify these digits with high precision even when there are variations in handwriting styles.

1.2 Domain Explanation

The project falls under the scope of Optical Character Recognition (OCR), a computer vision and pattern recognition domain concerning the translation of various forms of text images—printed, handwritten, or typed—into machine-readable text. Automated document digitization relies heavily on OCR, allowing computers to capture and read textual information from scanned documents, forms, and photographs. Handwritten digit recognition is a particular sub-domain of OCR, specifically addressed to the problem of recognizing numerals penned by hand, which are liable to be quite different from individual to individual.

1.3 Advantage of Using ML in This Domain

The application of machine learning, particularly deep learning, highly improves the accuracy and efficiency of OCR engines. Conventional image processing methods have limited ability to generalize across various handwriting styles, noises, and image distortions. However, machine learning algorithms, especially CNNs, are capable of learning intricate patterns and features from extensive datasets and thus highly resilient to diverse handwriting styles and resistant to noise. This results in increased accuracy, scalability, and real-time capability. Such systems find applications in diverse areas such as automatic check processing in banking, form digitization in government sectors, automated postal sorting, and assistive devices for the blind. Integration of ML-based OCR systems into these areas eliminates human workload, reduces time, and minimizes errors.

2 Literature Survey

Several techniques have been proposed for digit recognition, including traditional computer vision methods and more recent neural network-based models. The MNIST dataset has become a benchmark for evaluating these models. Studies show that convolutional neural networks (CNNs) outperform other methods in terms of both accuracy and efficiency.

Table 1: Comparison of Related Work on Handwritten Digit Recognition

Sl. No.	Title	Dataset	ML Model	Optimization	Accuracy
1	A Comparison of Three Classification Algorithms for Handwritten Digit Recognition	NIST	MLP, NB, K-Star	-	0.8236
2	Conventional Neural Network for Maths Handwritten Digit Categorization	MNIST	CNN	ADAM	0.9827
3	KNN and the CNN for Handwritten Digit Recognition: A Comparative Study	MNIST	KNN, ANN, CNN	-	-
4	CNN Learning Features for Handwritten Digit Recognition	MNIST	CNN	-	0.9954
5	Handwritten Digit Recognition Based on Deep Learning Algorithms	MNIST	CNN	Over/Undersampling	0.99
6	Handwritten Digit Recognition Using CNN in Python with TensorFlow	MNIST	CNN	-	0.9973
7	Evaluation of Supervised ML Models for Handwritten Digit Recognition	MNIST	CNN, SVM, KNN, RF, GBC, LR, DT, NB	-	96%-83%
8	Evaluation of Classifiers Based on Neural Network for Digit Recognition	MNIST	ANN, CNN	ADAM	0.98
9	Recognition of Handwritten Digit Using Neural Networks	MNIST	ANN	ADAM	0.979
10	An Enhanced Handwritten Digit Recognition Using CNN	MNIST	CNN	-	-

Description: This project develops a handwritten digit recognition model using CNN on MNIST dataset to classify digits (0-9) with high accuracy.

3 Dataset Description with Visualization

We used the MNIST dataset in CSV format, which consists of 70,000 grayscale images of handwritten digits (56,000 for training and 14,000 for testing), each of size 28x28 pixels and flattened to 784 features.

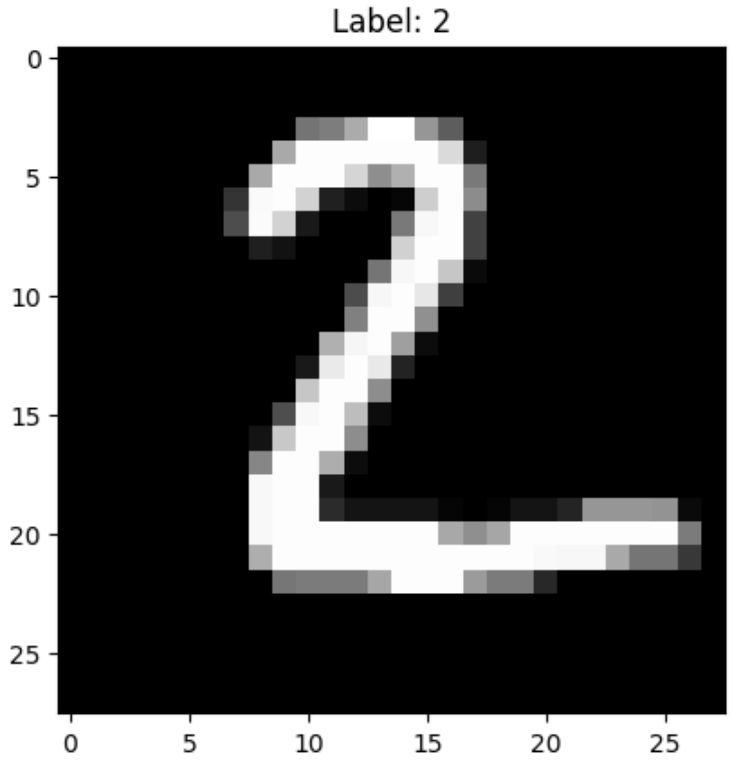

Figure 1: Sample digit images from the MNIST dataset

4 Algorithm

4.1 Mathematical Formulation

Let $x \in R^{784}$ be the input image vector (reshaped as 28×28), and $y \in \{0, 1, ..., 9\}$ be the digit label. A convolutional neural network $f(x; \theta)$, parameterized by weights θ, outputs a vector of raw class scores (logits), one for each class.

The architecture is:

- Convolution Layer 1: $h_1 = \text{ReLU}(\text{Conv2D}(x, W_1) + b_1)$, with 32 filters of size 3×3
- Max Pooling 1: $p_1 = \text{MaxPool}(h_1, 2 \times 2)$
- Convolution Layer 2: $h_2 = \text{ReLU}(\text{Conv2D}(p_1, W_2) + b_2)$, with 64 filters of size 3×3
- Max Pooling 2: $p_2 = \text{MaxPool}(h_2, 2 \times 2)$
- Flatten and Fully Connected: $z = \text{ReLU}(W_3 \cdot \text{flatten}(p_2) + b_3)$ where $W_3 \in R^{128 \times 3136}$
- Output Layer: $\hat{y} = W_4 z + b_4$, where $W_4 \in R^{10 \times 128}$

The loss function used is Cross-Entropy Loss, applied directly to the raw logits:

$$\mathcal{L} = \text{CrossEntropyLoss}(\hat{y}, y)$$

4.2 Pseudo Code

Algorithm 1 Digit Classification using CNN in PyTorch

Define CNN model with 2 Conv2D layers, MaxPooling, ReLU, and 2 Fully Connected layers
Initialize weights θ and optimizer (Adam), set device (CPU/GPU)
Define CrossEntropyLoss as the loss function
for each epoch **do**
    Set model to train mode
    for each batch (x, y) in training set **do**
        $x, y \leftarrow x.to(device), y.to(device)$
        optimizer.zero_grad()
        $\hat{y} \leftarrow f(x; \theta)$ ▷ Forward pass
        $\mathcal{L} \leftarrow \text{CrossEntropyLoss}(\hat{y}, y)$
        $\mathcal{L}$.backward() ▷ Backpropagation
        optimizer.step() ▷ Update weights
        Track loss and accuracy
    end for
end for

4.3 Explanation

The CNN model learns spatial features using convolutional filters and non-linear activations (ReLU). Max pooling layers reduce spatial dimensions and improve generalization. The final fully connected layers map the flattened features to raw class scores. CrossEntropyLoss is applied directly to these logits, which internally handles the softmax and log operations for multi-class classification.

5 Architecture Diagram and Explanation

Figure 2: CNN Architecture for Digit Classification

The model consists of:

- Two convolutional layers with 3x3 kernels and ReLU activation
- Max-pooling layers for spatial downsampling
- A fully connected hidden layer with ReLU activation
- An output layer that produces raw logits for classification

6 Evaluation Metrics with Graphs and Explanation

6.1 Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100$$

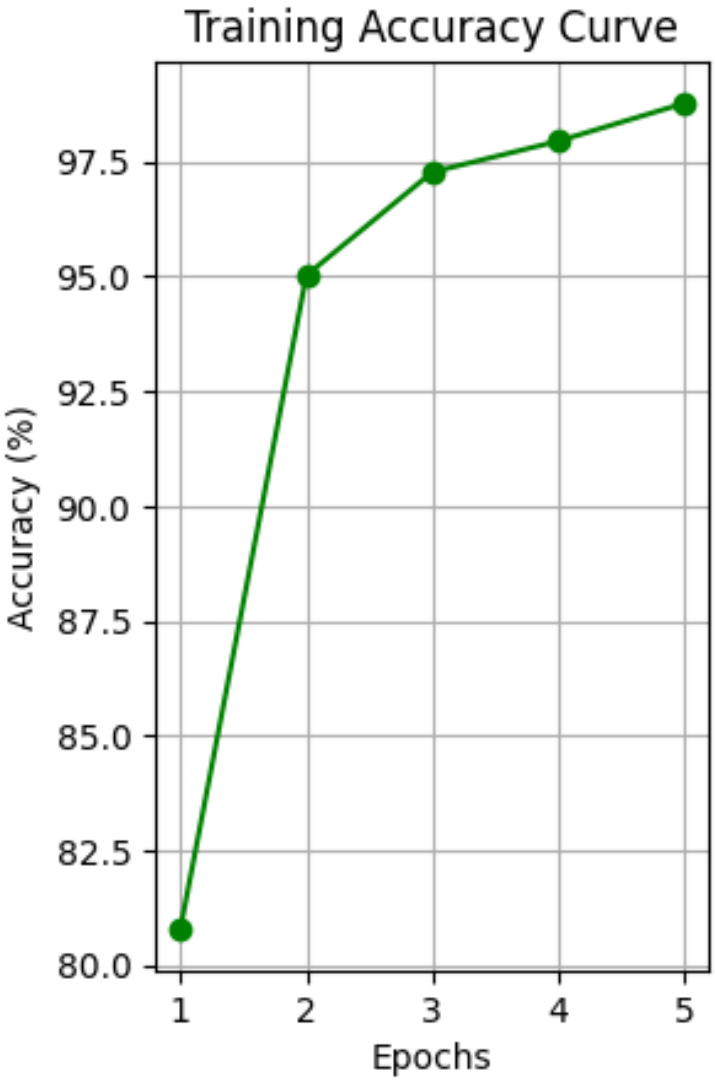

Figure 3: Training vs Validation Accuracy

6.2 Loss

$$\text{Loss} = -\sum y \log(\hat{y})$$

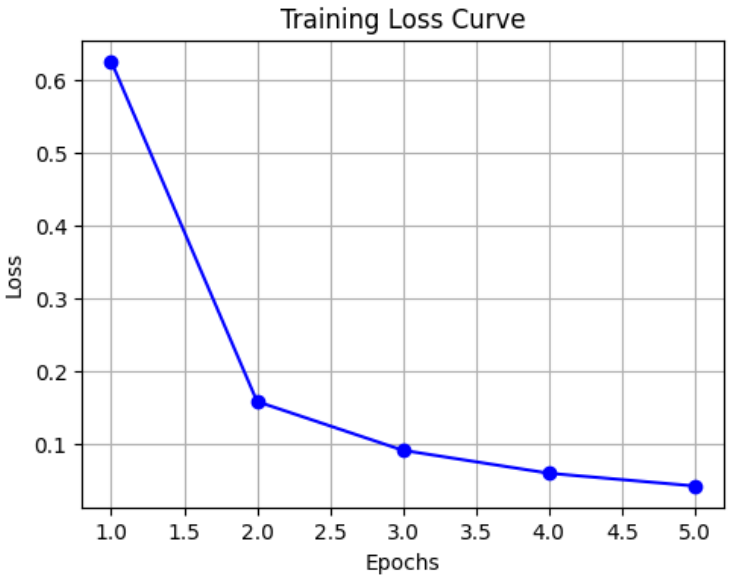

Figure 4: Training vs Validation Loss

6.3 Confusion Matrix

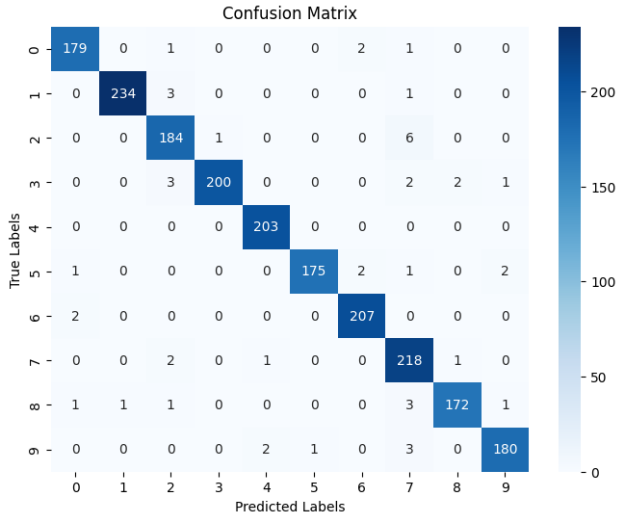

Figure 5: Confusion Matrix for Test Data

6.4 Explanation

The model achieved over 98% accuracy on the MNIST test set. The confusion matrix shows excellent performance, with few misclassifications mostly occurring between similar digits.

7 Conclusion

This project was able to effectively apply Convolutional Neural Networks (CNNs) to the problem of handwritten digit classification. By using the MNIST dataset, a common benchmark for machine learning, we trained a deep network model able to well classify digits from raw images. The design involved several convolutional and pooling layers followed by fully connected layers, which allowed the model to learn intricate spatial hierarchies of features directly from the input images. Through the training and evaluation pipeline, we obtained high classification accuracy, aided by metrics like loss curves and confusion matrices. The PyTorch usage for model development provided flexibility, modularity, and efficiency in training and exper- imentation. The results emphasize the power of CNNs in image classification tasks, especially in domains in which features are not strictly defined, and data is highly variable—like handwritten inputs. In addition, the performance of the model supports its feasibility for inclusion in broader Optical Character Recognition (OCR) systems. This might involve real-time applications like automated form data entry, postal address recognition, and digit interpretation in financial documents. In subsequent work, the model might be augmented by trying data augmentation, regularization methods, or transfer learning to further improve generalization and robust- ness. In general, this project highlights the power of deep learning in addressing real-world visual recognition issues with little human feature engineering.