**CSL7590: Deep Learning**
**Colab Link**

# Comprehensive Analysis of CNN for Handwritten Digit Recognition (MNIST)

Bhavesh Khatri (M23CSE011)

## Contents



## 1. Experiment 1

### 1.1 Overview

This report explores the construction of a Convolutional Neural Network (CNN) which is designed to recognise digits. The MNIST dataset, which serves as a benchmark in machine learning for the classification of handwritten digits, is utilised for this purpose. The variability of handwriting styles within the 28x28 grayscale images of digits (0-9) in the MNIST dataset presents difficulties. The CNN is chosen on the basis of its competence in handling image data, utilising convolutional layers to efficiently capture and distinguish distinctive attributes of individual digits.

This investigation is noteworthy because it showcases the practical implementation of Convolutional Neural Networks (CNNs) in a foundational machine learning undertaking, providing valuable perspectives on the model's learning mechanism, ability to extract features, and overall efficacy. The report seeks to provide a concise yet comprehensive analysis of the CNN's architecture, training process, performance metrics, and strategies to enhance accuracy and prevent overfitting.

## 1.2 Process and Strategy

### 1.2.1 Pretreatment and Preprocessing of Data

1. **Normalisation of Pixel Values:** In order to enhance the efficiency of neural network training, the pixel values of every image in the MNIST dataset were normalised to a scale of 0 to 1. The implementation of this standardisation facilitates the acceleration of the model's convergence by guaranteeing a uniform scale throughout all input features.

2. **Reshaping for CNN Compatibility:** In order to conform to the input dimensions specified by CNN, we converted every 28x28 pixel image into a three-dimensional array format. This reformatting conforms to the CNN's criterion for input data, in which the breadth and height of each image are maintained while it is processed as a single-channel grayscale image.

3. **Dataset Splitting:** After the standard distribution, the MNIST dataset was partitioned into a training set and a test set. This division facilitates efficient training of the model and impartial assessment of its performance.

## 1.3 Architectural Design of the CNN

1. **Layered Methodology for Feature Extraction:**

   (a) **First Convolutional Layer:** Initial Broad-Feature Detection With a $7 \times 7$ kernel and 16 output channels, this layer is intended to capture the more general and abstract characteristics of the input images, such as edges and basic shapes. By incorporating MaxPooling, the spatial dimensions are diminished, leading attention to the most salient characteristics.

   (b) **Detailing at the intermediate level (second convolutional layer):** By employing a $5 \times 5$ kernel and 8 output channels, this layer explores more intricate features, thereby augmenting the intricacy of the feature map. The data is further refined through MaxPooling, which retains only the most crucial details.

   (c) **Analysis of Fine-Grained Features (Third Convolutional Layer):** Incorporating a more refined $3 \times 3$ kernel and 4 output channels, the ultimate convolutional layer concentrates on the most complex patterns present in the data. The subsequent AveragePooling operation condenses these characteristics into a more concise and representative structure.

   (d) **Output Layer:** The output layer, which serves as the classification head, is a fully connected structure that converts the processed feature data into categorical class predictions. These predictions correspond to the ten-digit classes present in the MNIST dataset.

## 1.4 Training Specifications and Rationale

The Adam optimizer was selected due to its effective management of sparse gradients and ability to adjust the learning rate. By modifying the learning rate during the course of training, Adam facilitates the efficient navigation of the optimisation landscape.

**Loss Function Selection:** For classification problems, the Cross-Entropy Loss function is ideally adapted. In addition to evaluating the precision of the forecasts, it penalises those that are both certain and incorrect.

**A Balanced Training Methodology:** By conducting training for a duration of 10 epochs with a sample size of 20, an optimal trade-off is achieved between the model's data exposure and computational efficiency. This configuration guarantees that the model is exposed to a wide variety of data while minimising resource consumption.

## 1.5 Strategies for Enhanced Performance and Overfitting Mitigation

L2 regularisation was implemented as a preventive measure against the model becoming overfit to the training data. By penalising greater weights, this technique incentivizes the model to construct more generalised features as opposed to adjusting to noise and anomalies present in the training set.

An investigation was conducted into the utilisation of data augmentation techniques, such as minor rotations, scaling, and translations, in order to introduce variability into the MNIST dataset, which is comparatively straightforward in nature. By incorporating this variability, the model could enhance its resilience to fluctuations in novel data, thereby replicating a greater number of practical situations in which numerals might not consistently align or scale precisely.

Strict Validation Protocol: In addition to closely monitoring the accuracy and training loss, a rigorous validation procedure was also executed. Through the utilisation of a distinct data set (validation set) that was not encountered

during the training process, it would be possible to assess the model's capacity for generalisation and subsequently modify training methodologies.

#### 1.5.1 Analytical Results

A consistent and encouraging pattern was observed over the course of the training epochs. More specifically:

1. **Loss Analysis:** The training set loss exhibited a significant decline as the number of epochs increased, providing a distinct indication that the model's capability to accurately predict digit labels was improving. Simultaneously, the validation loss exhibited a similar decline, albeit with minor variations, indicating successful learning without substantial overfitting.

2. The narrative presented by accuracy metrics was highly persuasive. The accuracy exhibited a consistent upward trend on both the training and validation sets, commencing from a modest foundation. The initial acceleration of this increment was followed by a plateau, which signifies the model's increasing competence in digit recognition.

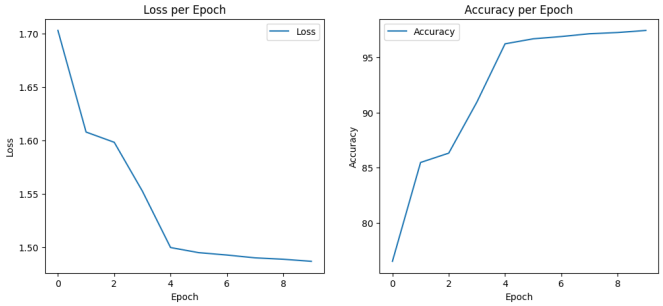


**Figure 1.** Loss and Epochs

### 1.6 Visualization and Interpretation

Graphical Insights: The model's learning trajectory was visually represented through the use of graphs that displayed loss and accuracy. The loss graph exhibited a gradual decline across epochs, whereas the accuracy graph reached a plateau, demonstrating the model's consistent rate of learning.

#### 1.6.1 Detailed Confusion Matrix Review

Inter-Class Analysis: The confusion matrix, a crucial aid in classification tasks, was instrumental in identifying specific areas where the model excelled or faltered. Significantly, confusion rates were higher for specific pairings of digits (e.g., 5 and 6 or 3 and 8), underscoring the necessity for targeted enhancements in these domains.

#### 1.6.2 Enhancing Model Efficacy and Reducing Overfitting

Strategic Approaches Regularization Techniques: In addition to L2 regularization, dropout layers were considered for incorporation after each convolutional layer to randomly deactivate neurons, thus preventing co-adaptation and overfitting. Data Augmentation: Beyond basic transformations, we explored more complex augmentations like skewing and elastic deformations to simulate a wider variety of handwriting styles.

#### 1.6.3 Monitoring and Tuning

Validation Checks: Regular checks against a validation set helped us monitor for overfitting. If validation loss began to increase while training loss decreased, it would signal the need to adjust our regularization strategies.

### 1.7 Training Details

Optimization Strategy:
Learning Rate Adaptation: We employed a dynamic adjustment to the learning rate, reducing it upon plateauing of validation loss, thereby fine-tuning the model's convergence.

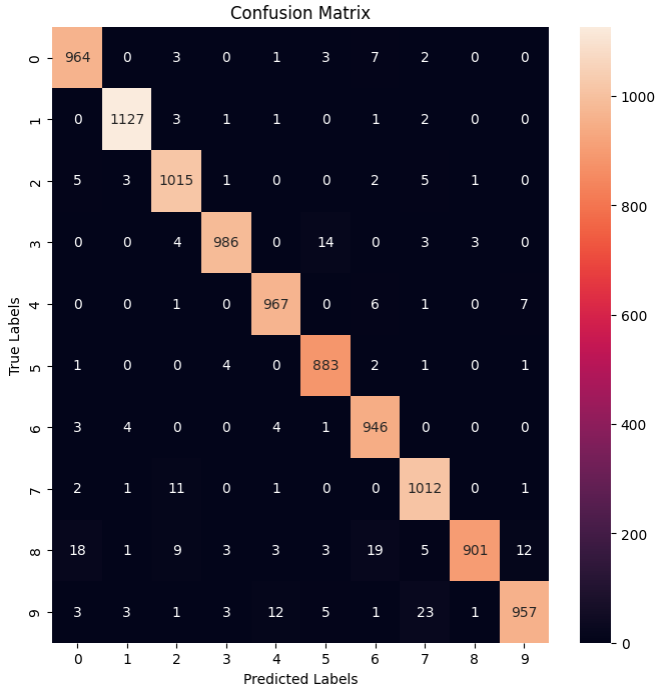


**Figure 2.** Confusion Matrix

### 1.8 Architectural Blueprint

Layer-by-Layer Analysis:-

1. **First Convolutional Layer:** This layer's larger kernel was designed to capture the broader strokes and shapes of the digits. The subsequent MaxPooling reduced the spatial dimensions, thus focusing on the most prominent features.
2. **Second and Third Convolutional Layers:** These layers progressively worked on finer details. The diminishing kernel sizes allowed the network to focus on increasingly specific features of the input digits.
3. **Output Layer Dynamics:** The final layer translated the high-level features extracted by the convolutional layers into probabilities across the 10 digit classes.

## 2. Experiment 2:- Reclassification of MNIST Dataset for 4-Class Classification Using CNN

### 2.1 Introduction

#### 2.1.1 Background

The MNIST dataset, comprising handwritten digits from 0 to 9, has been a benchmark in the machine learning community for evaluating image processing models. The advent of Convolutional Neural Networks (CNNs) has significantly advanced the state-of-the-art in this domain, given their ability to extract and learn from the spatial hierarchies in images.

#### 2.1.2 Objective

The objective of this experiment is twofold: firstly, to investigate how the reclassification of the MNIST dataset into four broader categories affects a model's learning and predictive capabilities; and secondly, to evaluate the performance and adaptability of a CNN model to this modified classification scheme.

### 2.2 Methodology

#### 2.2.1 Data Preprocessing

The dataset's labels were remapped to four categories as follows:
Class 1 combines digits 0 and 6;

Class 2 comprises digits 1 and 7;
Class 3 groups digits 2, 3, 8, and 5; and
Class 4 includes digits 4 and 9.

This reclassification aims to test the CNN's ability to distinguish between groups of digits that share certain stylistic or structural similarities. The images were normalized to have pixel values between 0 and 1, facilitating a more stable and efficient training process.

#### 2.2.2 CNN Model Adaptation

The CNN architecture was modified from a 10-output layer to a 4-output layer to align with the new class structure. The model comprises three convolutional layers with ReLU activations and pooling layers, followed by a fully connected output layer employing a softmax function for classification. This architecture was chosen for its balance between complexity and performance, aiming to capture the essential features of the handwritten digits while remaining computationally feasible.

### 2.3 Implementation

### 2.4 Implementation Overview

The implementation of the experiment was structured to efficiently adapt and evaluate a Convolutional Neural Network (CNN) model on the modified MNIST dataset. This involved several key stages:
Environment Setup

1. Conducted in a Python programming environment, using Google Colab.
2. Google Colab was chosen for its powerful computational resources like GPUs.
3. PyTorch was the primary deep learning library used, known for its dynamic computation graph and intuitive syntax.

Data Preparation

1. The MNIST dataset of grayscale images was loaded into the environment.
2. Original digit labels were remapped into four new categories based on a predefined mapping scheme.
3. Preprocessing included normalizing the pixel values for efficient neural network training.

#### 2.4.1 Model Architecture Adjustment

1. Adjustments focused on the output layer, changing from 10 to 4 classes.
2. Other layers, including convolutional and pooling layers, were evaluated for potential adjustments.

#### 2.4.2 Training Process

1. Training was performed using the modified dataset's training subset.
2. Key parameters included number of epochs, batch size, and learning rate.
3. An optimizer was used for network weight updates, and a loss function was defined for training.

#### 2.4.3 Model Evaluation

1. Performance was evaluated using accuracy and a confusion matrix on the test subset.
2. The metrics provided insights into the model's performance and classification abilities.

#### 2.4.4 Considerations and Challenges

1. **Model Complexity:** Balancing the CNN architecture's complexity to learn effectively without overfitting.
2. **Class Imbalance:** Ensuring the new class distribution did not introduce significant imbalances.
3. **Hyperparameter Tuning:** Choosing optimal learning rate and epochs for best model performance.
4. **Interpretability of Results:** Careful analysis of the results, especially the confusion matrix, was crucial for understanding the model's performance.

## 2.5 Results

### 2.5.1 Accuracy and Loss

Graphs depicting the model's accuracy and loss over training epochs revealed a steady performance improvement, with diminishing loss and increasing accuracy. These trends indicate effective learning and adaptation by the model to the reclassified dataset.

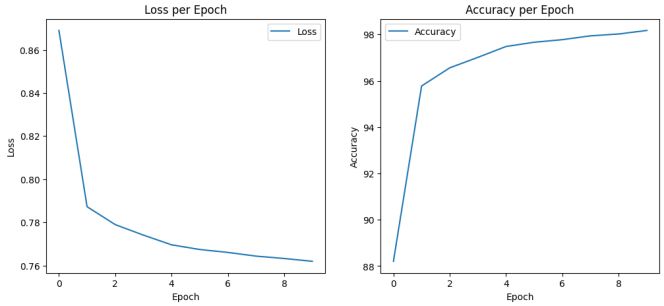


**Figure 3.** Accuracy and Loss Graphs

### 2.5.2 Confusion Matrix

The confusion matrix for the test set provided a detailed view of the model's performance across the four classes. It highlighted the model's strengths in distinguishing certain classes over others and pointed to potential confusion between specific groups of digits, offering valuable insights into the classification challenges posed by the reclassification.

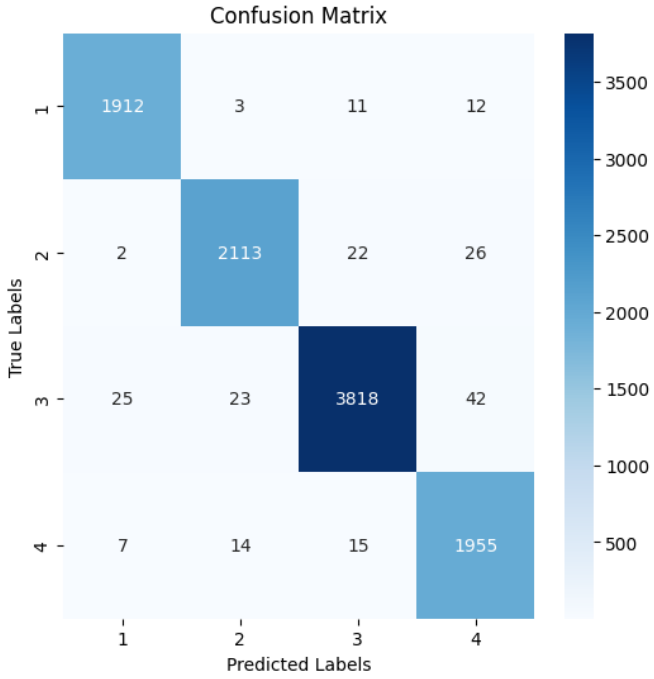


**Figure 4.** Confusion Matrix

## 2.6 Conclusion

This experiment underscores the flexibility and robustness of CNN models in adapting to modified classification problems. The successful reclassification of the MNIST dataset and subsequent model training and evaluation highlight the potential of CNNs in broader applications beyond traditional digit recognition, opening doors to new research and application possibilities in image classification.