Handwritten Digit Recognition using Deep Learning in MATLAB

Tanveer Hussain

August 12, 2025

Objective

The objective of this project is to design and implement a **deep learning-based handwritten digit recognition system** in MATLAB using the **MNIST dataset**. The aim is to build, train, and test a **Convolutional Neural Network (CNN)** capable of classifying grayscale images of digits (0–9) with high accuracy.

Project Tasks

In this project, you will:

1. Load the MNIST dataset into MATLAB.
2. Preprocess the data for CNN training.
3. Design the CNN architecture using MATLAB's Deep Learning Toolbox.
4. Train the CNN on the training dataset.
5. Evaluate the trained model on the test dataset.
6. Visualize and interpret the results (accuracy, confusion matrix, sample predictions).
7. Save the trained model for future use.

Total Steps to Complete the Project

Step 1 – Setup Environment

- Install MATLAB with the Deep Learning Toolbox.
- Enable GPU support if available for faster training.

Step 2 – Load Dataset

- Use MATLAB's built-in `digitTrain4DArrayData` or load MNIST manually.
- Split into training set (60,000 images) and test set (10,000 images).

Step 3 – Data Preprocessing

- Normalize pixel values to the range [0, 1].
- Reshape images to $28 \times 28 \times 1$ for CNN input.
- (Optional) Apply data augmentation for better generalization.

Step 4 – Define CNN Architecture

A typical CNN architecture includes:

- `imageInputLayer([28 28 1])`
- Convolutional layers with filters (e.g., `convolution2dLayer(3,8)`)
- ReLU activation layers (`reluLayer`)
- Pooling layers (`maxPooling2dLayer(2)`)
- Fully connected layer (`fullyConnectedLayer(10)`)
- Softmax layer and classification layer

Step 5 – Set Training Options

- Define optimizer (`sgdm`), learning rate, and number of epochs using `trainingOptions`.

Step 6 – Train CNN

- Use `trainNetwork(trainImages, trainLabels, layers, options)` to train the model.

Step 7 – Test and Evaluate

- Classify test images using `classify`.
- Calculate accuracy and generate a confusion matrix using `plotconfusion`.

Step 8 – Save Model

- Save trained model to a `.mat` file for later use.

Expected Results

- A trained CNN model capable of recognizing digits 0–9.
- High classification accuracy (typically >98% on MNIST).
- Confusion matrix showing per-class performance.
- Example predictions with true vs. predicted labels.

Inputs

- **Image data:** 28×28 grayscale handwritten digit images.
- **Labels:** Categorical values $\{0, 1, \ldots, 9\}$.

Outputs

- Numeric prediction for each image (0–9).
- Overall accuracy percentage.
- Visual results including:
  - Confusion matrix plot
  - Example image predictions
- Trained model saved as a `.mat` file.