Project Title:

Handwritten Digit Recognition Using Machine Learning Algorithms and Comparison Using R Software

Supervisor:

Dariush Najarzadeh

Prepared by:

Paniz Tarhib

University:

University of Tabriz

February 2025

Introduction

Handwritten digit recognition is one of the classic problems in image processing and machine learning, with wide applications in various domains such as license plate recognition, digit extraction from administrative forms, and analysis of historical manuscripts. With advancements in machine learning algorithms, the accuracy of recognizing handwritten digits has significantly improved.

In this project, the aim is to design and implement models for recognizing handwritten digits (0–9). Unlike well-known datasets like MNIST, the dataset used here is custom-made by the author — digits were handwritten on paper, photographed using a mobile phone camera, and then used for model training. This introduces more realistic challenges, including variations in handwriting style, lighting conditions, image quality, and size differences.

Dataset Construction • Image Collection Digits were written on paper and photographed with a mobile phone camera. Approximately 20 samples were collected for each digit (0 to 9).

• Image Storage Images are saved in a specific folder (e.g., D://nums). Each file is named in a format that allows easy extraction of the label (digit); for example, the first image of digit zero is named 0_1.jpg in JPG format. The first number in the filename represents the label.

Sample Images from the Dataset Below are samples of the handwritten digits:

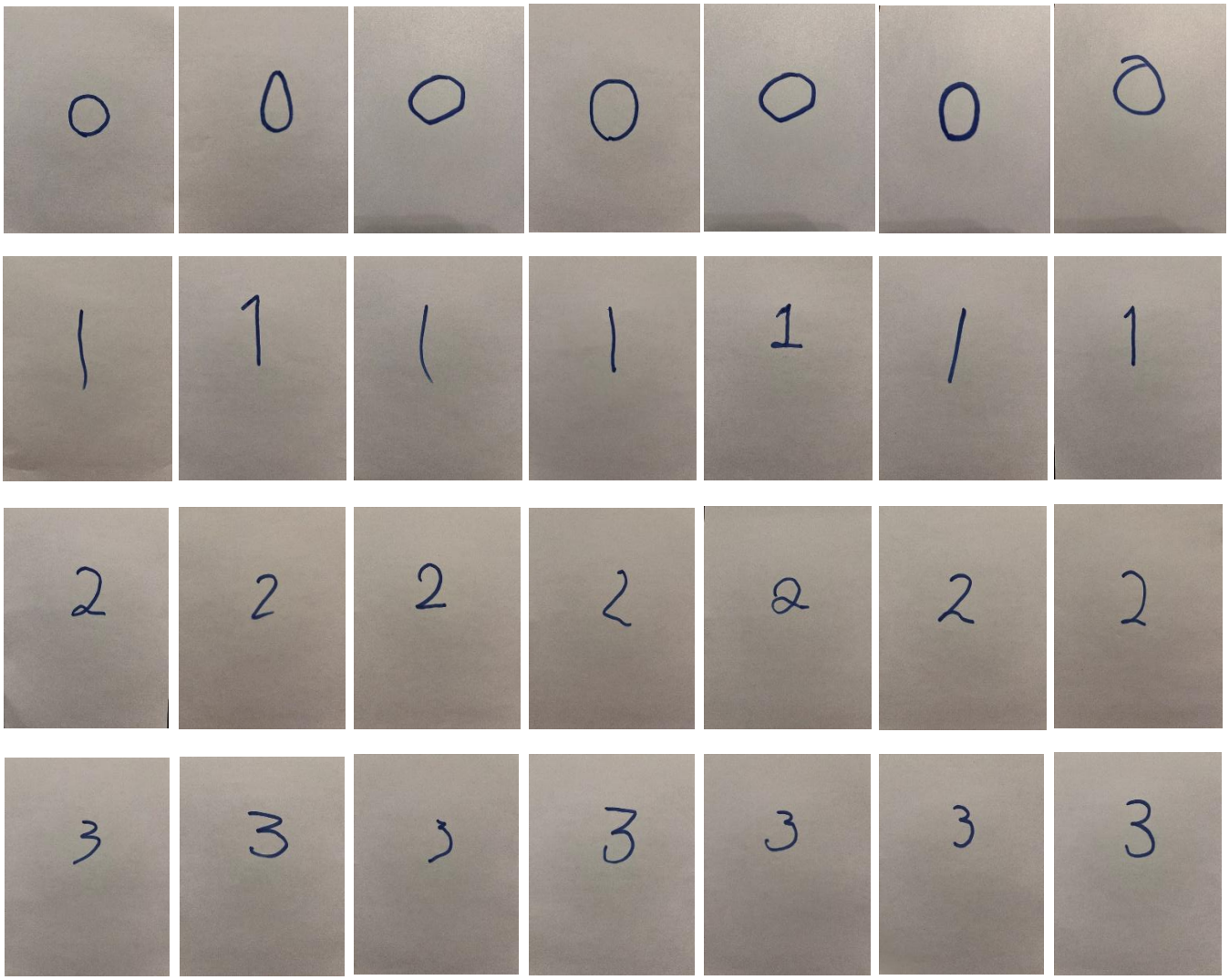

Image Preprocessing

- Loading and Grayscale Conversion: Using the imager package, images are loaded and converted to grayscale. Color images have three channels (Red, Green, Blue), but color information is unnecessary for digit recognition. Grayscale conversion reduces data size and accelerates processing, as only one channel is retained.
- Resizing: Images are resized to the standard 28×28 pixels. Machine learning models require uniform input dimensions. The 28×28 size is a common standard for digit recognition tasks.
- Conversion to Numerical Vector: The resized grayscale image (28×28 matrix) is flattened into a 784-dimensional numerical vector containing pixel intensity values (typically 0–255).

This entire preprocessing pipeline is implemented in the function process_image().

- Data Frame Construction: The resulting vectors, along with labels extracted from filenames, are stored in a data frame. Each row contains 784 features (pixels) and one label (digit 0–9).

Data Splitting and Normalization

- Data Splitting: Using the initial_split function from the tidymodels package, the dataset is divided into training (80%) and testing (20%) sets, stratified by label to maintain class distribution.
- Normalization: Pixel values (0–255) are normalized (typically to [0,1]) to improve model performance, especially for distance-based algorithms like KNN that rely on Euclidean distance. This step, along with handling missing values, is performed within a processing recipe.

Models and Results

❖ **KNN Model**

- Implementation: K-Nearest Neighbors algorithm with K=1 and rectangular weighting. Trained using the kknn engine.
- Result: Accuracy: 41.67%
- Explanation: This simple baseline model provides a reference for comparing more complex approaches.

❖ **MLP Model (Multi-Layer Perceptron)**

- Implementation: A single-hidden-layer neural network using the nnet package. PCA was applied beforehand to reduce dimensionality while preserving 95% of variance (to mitigate overfitting). The model has 6 hidden units.
- Result: Accuracy: 51.67%
- Explanation: The improvement over KNN demonstrates that even a basic neural network can capture more complex patterns.

❖ **Multi-Layer Neural Network (Keras – Dense Layers)**

- Implementation: Since nnet in R supports only single hidden layers with limited capacity, the keras library was used to build a deeper network. Images were converted to matrices and labels to one-hot encoding format. Architecture:
- Dense layer: 200 units + ReLU activation
- Dense layer: 155 units + ReLU activation
- Output Dense layer: 10 units + Softmax activation
- Result: Accuracy: 55%
- Explanation: Increasing model depth allowed extraction of more sophisticated features and improved performance compared to single-layer networks, though accuracy remains moderate.

❖ **CNN Model (Convolutional Neural Network)**

- Implementation: Fully connected (Dense) layers have limitations in image processing as they ignore spatial relationships between pixels. Therefore, a Convolutional Neural Network was implemented using keras. Architecture:
- Conv2D: 64 filters, 3×3 kernel, ReLU
- MaxPooling2D: 2×2 pool size
- Conv2D: 64 filters, 3×3 kernel, ReLU
- MaxPooling2D: 2×2 pool size

- Flatten
- Dense: 128 units, ReLU
- Output Dense: 10 units, Softmax
- Result: Accuracy: 76.67% (highest)
- Explanation: CNN's ability to extract local and hierarchical features results in significantly better performance, especially on images with variations and complexity.

Comparison of Results

The comparison clearly shows that more complex models — particularly CNN — deliver superior performance in handwritten digit recognition, especially when dealing with noisy, varied, and complex images.

Suggestions for Improvement

✓ Apply thresholding to reduce noise and unwanted shadows

✓ Increase dataset diversity through data augmentation (rotation, scaling, translation)

✓ Perform hyperparameter tuning (e.g., grid search)

✓ Experiment with deeper or more advanced model architectures

These improvements are expected to further enhance accuracy and robustness in real-world handwritten digit recognition scenarios.