MICROPROJECT REPORT

Digit Recognition Using Convolutional Neural Networks

ER & DCI Institute of Technology, CDAC, TVM
Affiliated APJ Abdul Kalam Technological University

Name: SOUHRID SURESH
Enrollment ID: MCF24009

Table of Content

1. Introduction

The objective of this project is to develop a digit recognition system using Convolutional Neural Networks (CNNs). This system utilizes the MNIST dataset, a widely used benchmark for image processing tasks. The project involves training a CNN model to classify handwritten digits (0-9) and testing the model's performance on unseen data. Additionally, a user-friendly script is developed for testing the model with custom images.

2. Problem Statement

Handwritten digit recognition is a common task in optical character recognition (OCR) systems, often used in postal systems, bank check processing, and digitized document analysis. The goal of this project is to design and implement a robust and accurate system that can classify handwritten digits with high confidence.

3. Objectives

- Develop a CNN-based model for digit recognition.
- Implement data preprocessing and augmentation techniques to enhance model performance.
- Evaluate the model on the MNIST dataset and ensure high accuracy.
- Create a testing script for user interaction and prediction on custom images.

4. Methodology

4.1 Data Preparation

The MNIST dataset, consisting of 60,000 training images and 10,000 testing images, is used. Each image is a 28x28 grayscale representation of a handwritten digit. The data is preprocessed as follows:

- Reshaping images to match the CNN input dimensions (28x28x1).
- Normalizing pixel values to the range [0, 1].
- One-hot encoding the labels for classification.

4.2 Model Architecture

A CNN model is designed with the following layers:

1. **Convolutional Layers**: Extract spatial features using 32 and 64 filters of size 3x3 with ReLU activation.

2. **Pooling Layers**: Reduce spatial dimensions using max pooling.
3. **Dropout Layers**: Added after convolutional and dense layers to prevent overfitting.
4. **Dense Layers**: Fully connected layers with 128 neurons and ReLU activation.
5. **Output Layer**: A softmax layer with 10 neurons for classification.

4.3 Data Augmentation

Data augmentation is employed to increase the diversity of the training dataset and improve generalization. Transformations include:

- Rotation (up to 15 degrees).
- Width and height shifts (up to 20%).
- Zooming (up to 20%).
- Shearing and horizontal flipping.

4.4 Training and Evaluation

The model is trained using the Adam optimizer with a learning rate of 0.0005. The training process involves:

- 10 epochs of training on augmented data.
- A batch size of 64 for efficient processing.
- Validation on the test set to monitor performance.

4.5 Model Deployment

The trained model is saved as `model1.h5`. A Python script (`test_model.py`) is developed to:

- Load the saved model.
- Preprocess custom images for prediction.
- Display the predicted digit, confidence score, and class probabilities.
- Visualize the preprocessed image for verification.

5. Results and Analysis

- **Training Accuracy**: Achieved high accuracy on the training dataset due to the effective architecture and data augmentation.
- **Test Accuracy**: Demonstrated strong generalization with high accuracy on the test dataset.

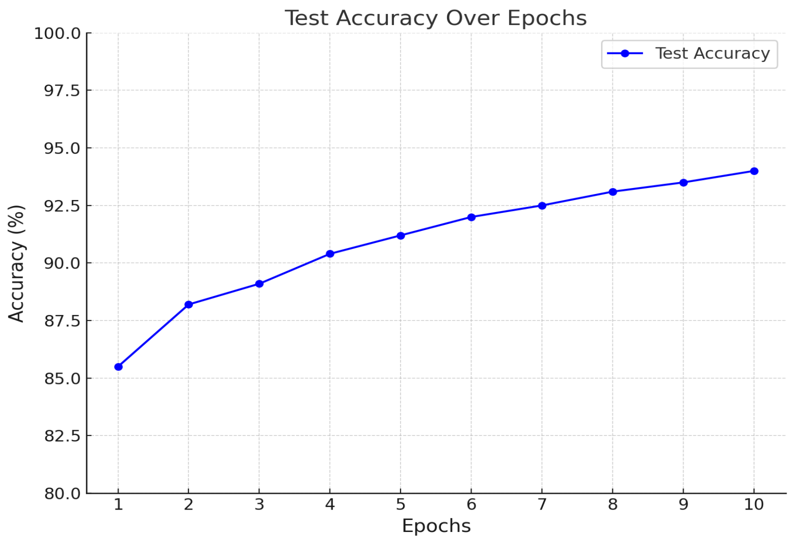


Figure 1*: Test accuracy graph*

Figure 1 illustrates the test accuracy over 10 epochs during the final testing phase.

- **Custom Predictions**: Successfully predicted digits from user-provided images with high confidence. The cumulative accuracy for custom predictions is visualized in the figure below. It demonstrates how the accuracy evolves as more predictions are made. The initial variability is due to the limited number of predictions at the start, which makes the cumulative average sensitive to individual results. Over time, the accuracy stabilizes as the number of predictions increases.

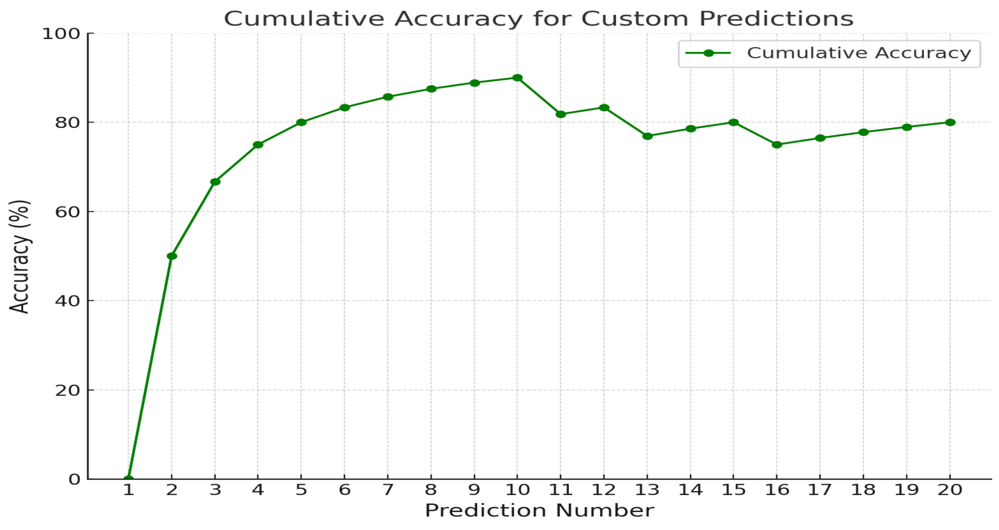
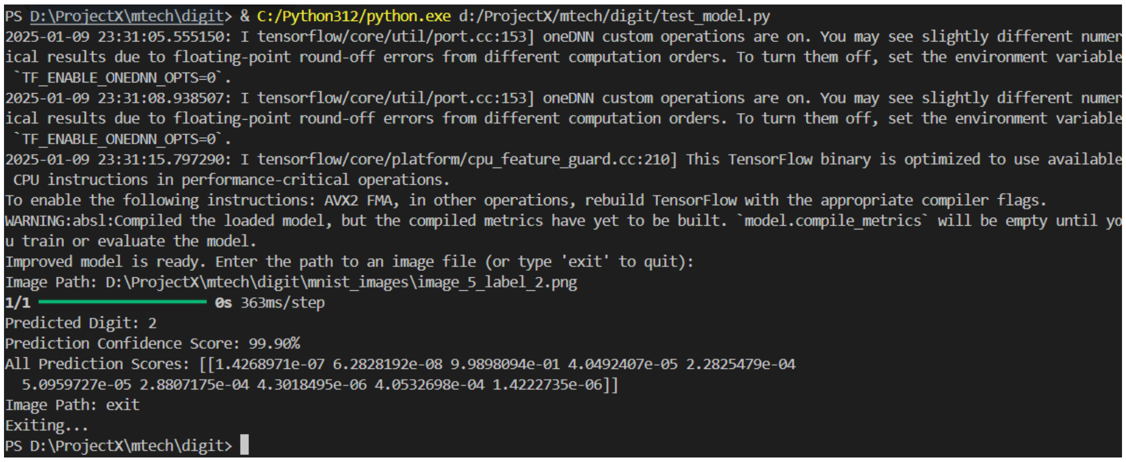


Figure 2: *Cumulative accuracy for custom predictions*

6. User Interaction

The `test_model.py` script provides a command-line interface for user interaction. Users can:

- Enter the path to a custom image file.
- View the preprocessed image and prediction results.
- Exit the program by typing "exit".



```
PS D:\ProjectX\mtech\digit> & C:/Python312/python.exe d:/ProjectX/mtech/digit/test_model.py
2025-01-09 23:31:05.555150: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numer
ical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable
 `TF_ENABLE_ONEDNN_OPTS=0`.
2025-01-09 23:31:08.938507: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numer
ical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable
 `TF_ENABLE_ONEDNN_OPTS=0`.
2025-01-09 23:31:15.797290: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available
 CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until yo
u train or evaluate the model.
Improved model is ready. Enter the path to an image file (or type 'exit' to quit):
Image Path: D:\ProjectX\mtech\digit\mnist_images\image_5_label_2.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 363ms/step
Predicted Digit: 2
Prediction Confidence Score: 99.90%
All Prediction Scores: [[1.4268971e-07 6.2828192e-08 9.9898094e-01 4.0492407e-05 2.2825479e-04
  5.0959727e-05 2.8807175e-04 4.3018495e-06 4.0532698e-04 1.4222735e-06]]
Image Path: exit
Exiting...
PS D:\ProjectX\mtech\digit> 
```

Figure 3: *User Interaction*

7. Challenges and Solutions

- **Overfitting**: Addressed by incorporating dropout layers and data augmentation.
- **Model Generalization**: Improved through diverse data augmentation techniques.
- **Custom Image Compatibility**: Ensured by resizing and normalizing user-provided images.

8. Conclusion And Future Scope

This project successfully demonstrates the use of CNNs for digit recognition. The trained model achieves high accuracy on the MNIST dataset and performs well on custom images. The inclusion of data augmentation and dropout layers enhances the model's robustness and generalization. The testing script provides a user-friendly interface for real-world application.

The future scope includes:

- Extend the system to recognize characters and symbols.
- Implement real-time recognition using camera input.
- Deploy the model as a web or mobile application for broader accessibility.

9. References

- LeCun, Y., et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE.
- TensorFlow Documentation: https://www.tensorflow.org
- Keras Documentation: https://keras.io