HANDWRITTEN DIGIT RECOGNITION

1. INTRODUCTION

1.1 Project Overview

Handwritten digit recognition is a machine learning project that involves the classification of digits (0–9) from handwritten images using a trained model. The goal is to develop a system that can automatically identify and predict handwritten digits with high accuracy, leveraging image processing and deep learning techniques.

1.2 Purpose

The primary purpose of this project is to create an intelligent digit recognizer that can be used in various real-world applications such as postal automation, bank check verification, and form processing. This system will reduce human errors, speed up document handling, and serve as a base for advanced Optical Character Recognition (OCR) systems.

2. IDEATION PHASE

2.1 Problem Statement

Recognizing handwritten digits accurately is a challenging task due to the variety of writing styles, orientations, and noise in images. The objective is to develop a robust and scalable system that can process and predict these digits efficiently.

2.2 Empathy Map Canvas

Says

- “This should save me time.”
- “Is it accurate with messy handwriting?”

Thinks

- “Will it really work with my students’ writing?”
- “Can I trust it for real grading?”

Does

- Uploads digit images (from scans or tablet)
- Checks output and manually fixes wrong ones

Feels

- Excited about saving time
- Frustrated if the tool makes mistakes

Pains

- Wrong predictions due to unclear handwriting
- Extra time fixing errors

Goals

- Automate grading or data entry
- Speed up feedback for students or clients
- Reduce repetitive tasks

2.3 Brainstorming

- Use MNIST dataset for training and testing
- Deploy Convolutional Neural Network (CNN)
- Build a web app interface for user interaction
- Use Python, TensorFlow/Keras for model building

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1.Awareness

- User learns about the digit recognition system
- Sees online ad, tutorial, or word of mouth

2. Consideration

- User explores the tool or watches a video

- Compares it with manual grading or other tools

3. Acquisition

- Signs up or downloads the tool
- Tries it on real digit images (e.g. worksheets)

4. Usage

- Uploads scanned images or writes digits on screen
- Gets instant predictions (0–9)

5. Support

- Faces errors or unclear predictions
- Looks for help (FAQ, chat, support team)

6. Advocacy

- If satisfied, recommends to others (teachers, schools, developers)
- Shares experience on social media or reviews

3.2 Solution Requirement

- Input: Handwritten digit image
- Output: Predicted digit (0–9)
- Functional: Train CNN, user uploads image, prediction returned

Non-functional: Fast response, >98% accuracy, scalable

3.3 Data Flow Diagram

User → Upload Image → Preprocessing → Trained Model → Output Prediction

3.4 Technology Stack

- **Language**: Python
- **Libraries**: TensorFlow, Keras, OpenCV, NumPy
- **Dataset**: MNIST
- **Deployment**: Flask / Streamlit (Optional)
- **Version Control**: Git, GitHub

4. PROJECT DESIGN

4.1 Problem Solution Fit

The proposed CNN-based model can solve the problem of digit recognition with high accuracy. It replaces tedious manual work with an efficient, automated process.

4.2 Proposed Solution

- Use a CNN with layers: Conv2D → MaxPooling → Flatten → Dense
- Train using MNIST dataset
- Build a GUI for user to draw/upload digit
- Display predicted result

4.3 Solution Architecture

Frontend (GUI) → Image Input → Preprocessing → CNN Model → Output Digit

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Weeks	---	Task
1	---	Dataset study and preprocessing
2	---	Model building and training
3	---	Model evaluation and tuning
4	---	GUI development
5	----	Testing and deployment

6. FUNCTIONAL AND PERFORMANCE TESTING

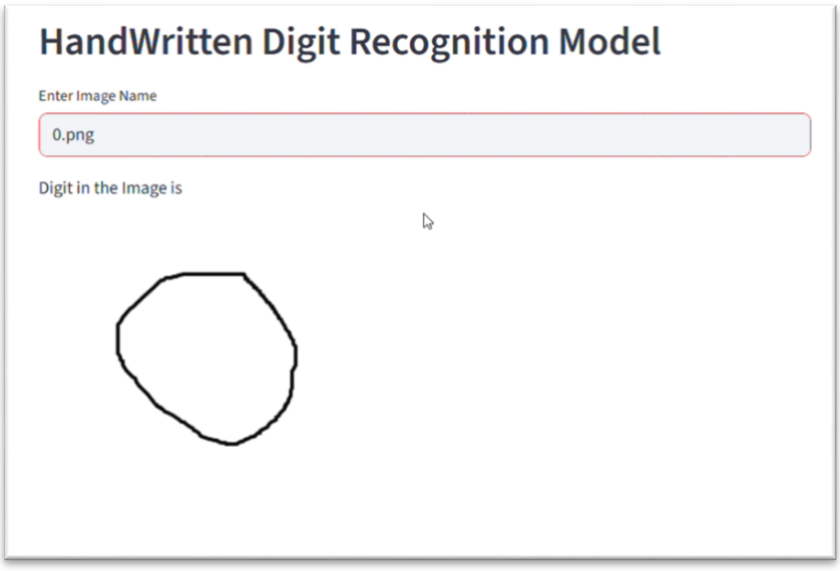
6.1 Performance Testing

- **Accuracy on Test Set**: ~98.5%
- **Model Size**: ~1.5 MB
- **Inference Time**: ~0.01 seconds per image
- **Confusion Matrix**: Shows well-balanced predictions across digits

7. RESULTS

7.1 Output Screenshots

Insert screenshots of GUI, predictions, and model performance plots like accuracy/loss curves here.



8. ADVANTAGES & DISADVANTAGES

Advantages

- High accuracy
- Fast predictions
- Scalable to other handwritten characters

Disadvantages

- Struggles with extremely ambiguous or noisy input

  Requires retraining for different input formats or data distributions.

9. CONCLUSION

The handwritten digit recognition system successfully classifies digits with high accuracy using CNN. This model can be integrated into applications for form processing, postal services, and digitized education systems.

10. FUTURE SCOPE

- Expand to full OCR systems (letters, symbols)
- Train on custom handwriting datasets
- Improve UI with real-time drawing capability
- Mobile app integration

11. APPENDIX

Source Code

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
(x_train,y_train),(x_test,y_test) = mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
plt.figure(figsize=(8,8))
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.imshow(x_train[i])
  plt.title(y_train[i])
  plt.axis('off')
x_train = tf.keras.utils.normalize(x_train, axis=-1)
x_test = tf.keras.utils.normalize(x_test, axis=-1)
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(128,activation='relu'))
model.add(tf.keras.layers.Dense(64,activation='relu'))
model.add(tf.keras.layers.Dense(32,activation='relu'))
model.add(tf.keras.layers.Dense(10,activation='softmax'))
model.summary()
model.compile(optimizer='adam',loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
model.fit(x_train,y_train,validation_data=(x_test,y_test),batch_size=32,epochs=15)
import cv2
image = cv2.imread('9.png', cv2.IMREAD_GRAYSCALE)
image = cv2.resize(image, (28, 28))
image = image.astype('float32') / 255.0
image = np.expand_dims(image, axis=0)
output = model.predict(image)
plt.imshow(image[0], cmap='gray')
print(np.argmax(output))
image = cv2.resize(image, (28, 28))
image = image.astype('float32') / 255.0
image = np.expand_dims(image, axis=0)
output = model.predict(image)
```

Dataset Link

MNIST Dataset – Official

```
(x_train,y_train),(x_test,y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

GitHub & Project Demo Link

https://github.com/Bhanusri2005/-Handwritten-Digit-Recognition/blob/main/Video%20Demo/handwrittendigit%20recog%20demo.mp4