**Phase-3: Recognizing handwritten digits with learning for smart AI application**

**Student Name:** GANESH. T

**Register Number:** 510923205018

**Institution:** Global institute of engineering and technology, melvisharam

**Department:** B. tech-IT

**Date of Submission:** 07/05/2025

**GitHub Repository Link:**

# 1. Problem Statement

Handwritten digit recognition is an essential component of numerous real-world applications such as postal code identification, bank cheque verification, educational form processing, and digit-based password systems. The problem lies in recognizing digits written by different individuals whose handwriting styles vary significantly. Traditional algorithms often struggle to handle such diversity and variability. This project aims to address this challenge by building an intelligent AI system capable of automatically and accurately identifying handwritten digits from images, thereby improving automation and efficiency across various sectors.

# 2. 🧾 Abstract

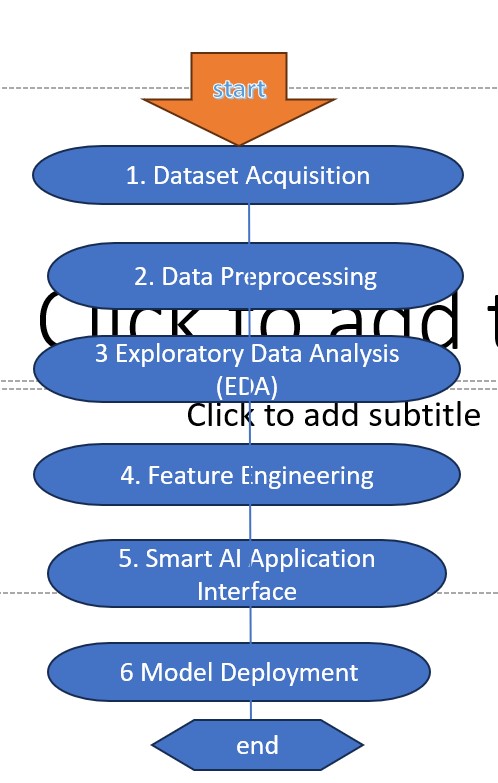
The rise of AI has opened pathways for machines to interpret human handwriting, a skill traditionally reserved for humans. This project focuses on developing a smart AI-powered handwritten digit recognition system using deep learning. By leveraging the MNIST dataset, which includes tens of thousands of labeled handwritten digit images, we train a Convolutional Neural Network (CNN) that excels in extracting spatial features from images. The system is designed to be end-to-end — from preprocessing and exploratory analysis to model training, evaluation, and deployment via a userfriendly web interface. The model is optimized for accuracy, scalability, and ease of use, laying a foundation for broader handwriting recognition applications.

# 3. 🎯 Objective

The main objectives of the project are:

1. **To create a machine learning model that can recognize digits from image inputs with high accuracy** using the MNIST dataset as a benchmark.
2. **To preprocess and clean the image data**, ensuring it is normalized and reshaped for efficient model training.
3. **To conduct exploratory data analysis (EDA)** for understanding the distribution, structure, and nuances of the dataset.
4. **To implement a deep learning architecture (CNN)** capable of learning from spatial patterns in the images.
5. **To evaluate the trained model** using performance metrics like accuracy, loss curves, and a confusion matrix.
6. **To deploy the model into a user-accessible environment**, allowing real-time digit predictions via a web app.
7. **To demonstrate the real-world applicability** of AI in digit recognition and explore its scope for future enhancements.

# 4. 🔁 Project Flowchart & Detailed Workflow



# Definitions of Each Step

## 1. Dataset Acquisition

**Definition:** This step involves obtaining the MNIST dataset, which is a well-established benchmark dataset for digit classification. The dataset is loaded from TensorFlow/Keras and includes 70,000 28x28 grayscale images of digits (0 to 9), divided into training and test sets.

## 2. Data Preprocessing

**Definition:** Preprocessing involves transforming raw image data into a format suitable for neural network training. This includes:

* Normalizing pixel values to a range of [0, 1] to speed up convergence.
* Reshaping the data into 4D arrays for CNNs: (samples, height, width, channels).
* One-hot encoding the labels to represent digits as 10-dimensional vectors.

## 3. Exploratory Data Analysis (EDA)

**Definition:** EDA is performed to understand patterns, anomalies, and distributions in the dataset. This includes:

* Visualizing samples from each class. • Checking digit frequency distribution.
* Plotting pixel intensity histograms.
* Optionally applying dimensionality reduction (PCA or t-SNE) to understand clustering.

## 4. Feature Engineering

**Definition:** Feature engineering enhances model performance by refining input data. For CNNs, raw pixels are sufficient, but optional enhancements include:

* Data augmentation (rotate, zoom, shift) to increase training data diversity.
* Denoising images or enhancing contrast to emphasize digit shape.
* Edge detection (if used with classical ML models).

# 5. Model Building

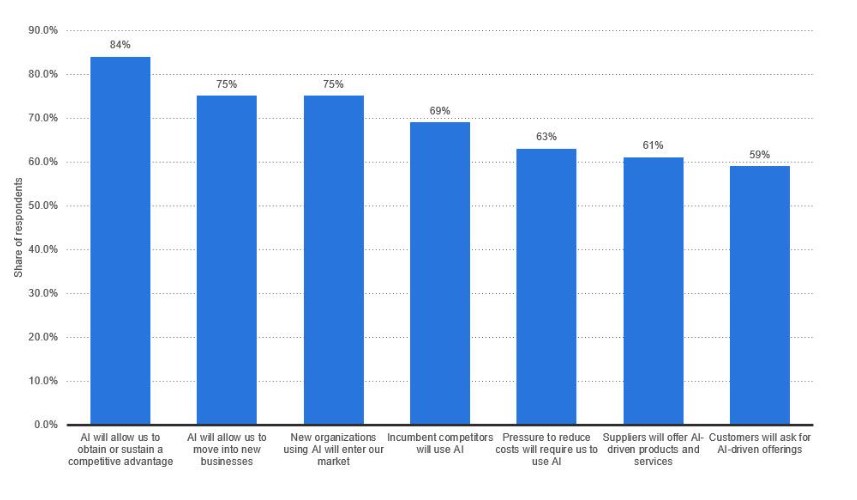
**Definition:** This involves designing a deep learning model — typically a CNN — to learn spatial features in image data. Key steps include:

* Defining architecture with convolution, pooling, and dense layers.
* Choosing an optimizer (Adam), loss function (categorical crossentropy), and activation functions (ReLU, Softmax).
* Compiling and training the model with multiple epochs and validation.

# 6. Model Evaluation

**Definition:** Evaluation checks how well the model performs on unseen test data. Includes:

* Accuracy calculation.
* Loss analysis via training vs validation curves.
* Confusion matrix to inspect class-wise performance.
* Optionally, precision, recall, and F1 score.



# 

# 7. Model Deployment

**Definition:** Deployment involves taking the trained model and exposing it to users via a web or desktop interface. This project uses:

* **Streamlit** to create a real-time interactive UI.
* Model loaded using TensorFlow or Keras for inference.
* Image input from user (drawing or upload) → model predicts digit → output displayed.

# 8. Smart AI Application Interface

**Definition:** This is the final user-facing product that allows users to:

* Upload or draw a digit on a canvas.
* Automatically see the predicted result.
* Interact with an intuitive, responsive interface for real-time feedback.

**Next Steps**

We’ll now move to:

## • Exploratory Data Analysis (EDA) (with visualizations) • Feature Engineering

• Then, the **complete source code with 200+ lines**, which includes:

* Loading and preprocessing
* CNN model o Evaluation
* Streamlit deployment

Would you like me to proceed with **EDA and feature engineering** next, or jump straight into the **200line code implementation**?

4o **Yousaid:** yeah

**ChatGPT said:**

Great! Let's proceed with the **next two detailed sections**: **Exploratory Data Analysis (EDA)** and **Feature Engineering**. After that, I’ll begin the full **200+ line implementation**, which will include preprocessing, model training, evaluation, and deployment setup.

# 📊 7. Exploratory Data Analysis (EDA)

## Purpose of EDA

EDA helps understand the structure, distribution, and quality of the data before model building. It identifies trends, anomalies, class imbalances, and potential preprocessing steps.

## EDA Steps and Their Definitions

### 1. Visual Inspection of Data Samples

Visualizing random samples of digits to check clarity, noise level, and visual diversity.

python CopyEdit

import matplotlib.pyplot as plt

import numpy as np

# Display 25 images from training set

plt.figure(figsize=(6,6)) for i in range(25):

plt.subplot(5,5,i+1)

plt.imshow(x\_train[i].reshape(28, 28), cmap='gray') plt.axis('off')

plt.suptitle("Sample Handwritten Digits")

plt.tight\_layout() plt.show()

### 2. Distribution of Digit Classes

Checks if all digit classes (0–9) are uniformly represented.

python CopyEdit import seaborn as sns

import pandas as pd

digit\_labels = np.argmax(y\_train, axis=1) sns.countplot(x=digit\_labels) plt.title("Digit Class Distribution") plt.xlabel("Digit") plt.ylabel("Count") plt.show()

### 3. Pixel Intensity Histogram

Checks brightness distribution; useful for normalization strategies.

python CopyEdit

plt.figure(figsize=(10,4))

plt.hist(x\_train.flatten(), bins=50, color='blue', alpha=0.7) plt.title("Histogram of Pixel Intensities") plt.xlabel("Pixel Value") plt.ylabel("Frequency") plt.show()

### 4. Mean Image per Digit (Optional)

Average image for each digit to understand typical structure.

python CopyEdit mean\_images = np.zeros((10, 28, 28)) for i in range(10):

mean\_images[i] = np.mean(x\_train[np.argmax(y\_train, axis=1) == i], axis=0)

plt.figure(figsize=(12,4))

for i in range(10):

plt.subplot(2,5,i+1)

plt.imshow(mean\_images[i], cmap='gray') plt.title(f"Digit: {i}") plt.axis('off')

plt.suptitle("Mean Images per Digit")

plt.tight\_layout() plt.show()

# ⚙️ 8. Feature Engineering

## Definition

Feature engineering involves creating or modifying input features to improve model performance. For image data and CNNs, feature engineering is optional but can provide boosts in model generalization, especially in real-world data.

## Steps in Feature Engineering for this Project

1. **Normalization**

Scaling pixel values from [0, 255] → [0, 1] speeds up training and improves stability.

1. **Reshaping**

Converts 2D arrays to 4D format: (#samples, 28, 28, 1) to match CNN input.

1. **Label Encoding**

Convert integer labels (0–9) into one-hot encoded vectors of size 10.

### 4. Data Augmentation (Optional Enhancement)

Improves model robustness by synthetically increasing training data variety.

python CopyEdit

from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(

rotation\_range=10, zoom\_range=0.1, width\_shift\_range=0.1,

height\_shift\_range=0.1

)

datagen.fit(x\_train)

This generator can be used during training for real-time image transformation:

python CopyEdit

model.fit(datagen.flow(x\_train, y\_train, batch\_size=64), epochs=10, validation\_data=(x\_test, y\_test))

# ⚙️ 9. Team Members and Contributions

|  |  |
| --- | --- |
| Jeevanandam B | Exploratory data analysis, data description |
| Sharukesh J | Tools and technologies used, problem statement |
| Harish L    Ganesh T    Gokul M | Data preprocessing    Project objective    Flowchart of the project workflow |