Phase-2

Student Name: DHASHVANDHAN.R

Register Number: 422723104024

Institution:V.R.S College of engineering and technology

Department:Computer science engneering Date of

Submission:10.05.2025

Github Repository Link:

https://github.com/Dhashvandhan399/Handwriting-digits-with-smart-ai-application-

---

Recognizing Handwritten Digits Using Smart AI Application

1. Problem Statement

Recognizing handwritten digits is a classic and foundational problem in computer vision and machine learning. The objective is to develop an AI-based model that can accurately identify digits (0–9) from images of handwritten text. This project uses deep learning techniques to process image data and predict the correct numeric label.

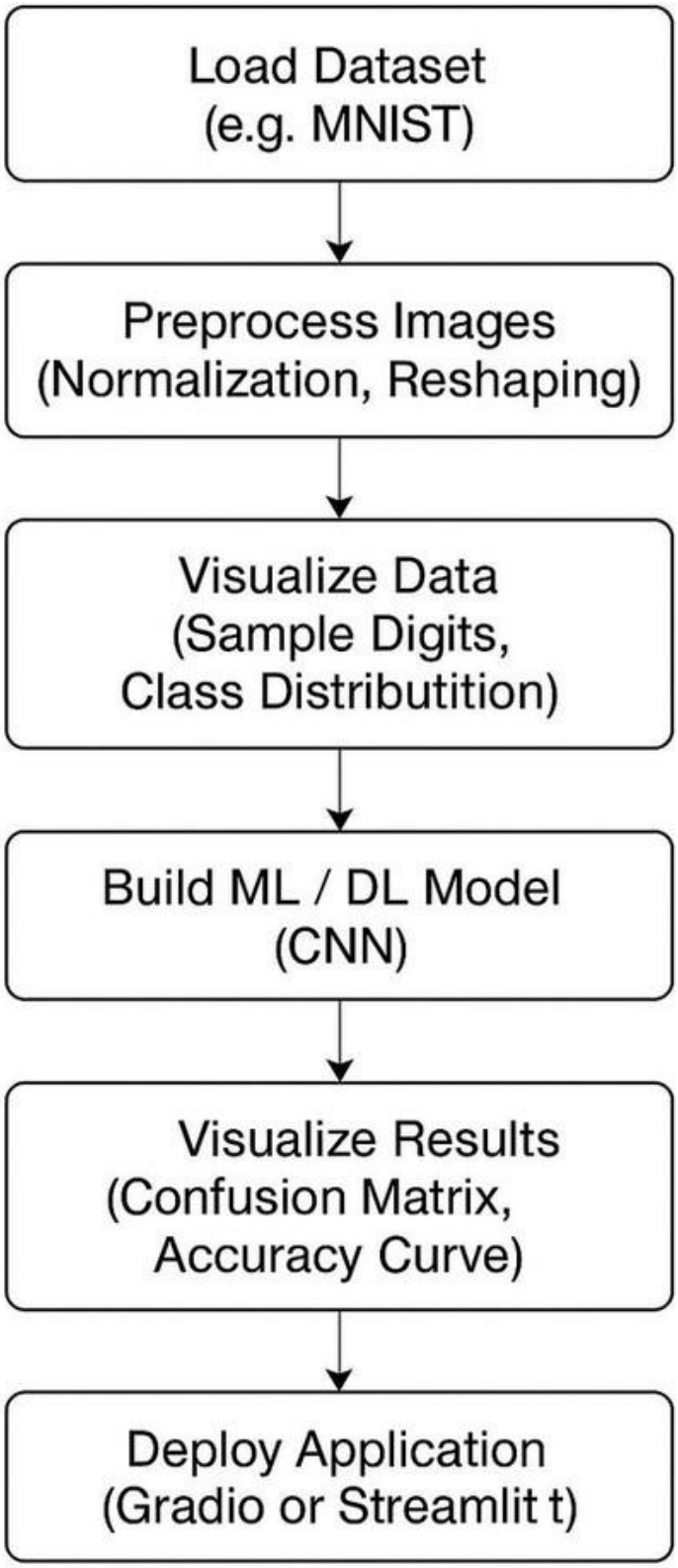
This problem falls under multiclass classification, where the model must choose one out of ten possible classes (0–9) for each image.

Solving this problem contributes to advancements in OCR (Optical Character Recognition), digital document processing, and intelligent form reading

2. Project Objectives

- *Build a robust deep learning model capable of classifying handwrittendigits.*
- *Train and evaluate the model using the MNIST dataset.*
- *Achieve high prediction accuracy and minimize classification errors.*
- *Deploy a user-friendly interface using Gradio for real-time testing.*
- *Enable digit recognition through image uploads or real-time drawing.*

3. Flowchart of the Project Workflow


Load Dataset
(e.g. MNIST)
Preprocess Images
(Normalization, Reshaping)
Visualize Data
(Sample Digits,
Class Distributition)
Build ML / DL Model
(CNN)
Visualize Results
(Confusion Matrix,
Accuracy Curve)
Deploy Application
(Gradio or Streamlit t)

4. Data Description

- *Dataset Name: MNIST Handwritten Digits Dataset● Source: https://www.kaggle.com/c/digit-recognizer , http://yann.lecun.com/exdb/mnist/*
- *Type of Data: Image data (28x28 grayscale pixels)*
- *Records and Features: 70,000 images (60,000 training, 10,000testing)*
- *Target Variable: Digit label (0–9)*
- *Static or Dynamic: Static dataset*

5. Data Preprocessing

1. Normalization
   - Pixel values in the MNIST images range from 0 to 255.
   - All pixel values were scaled to the range [0, 1] by dividing by 255.
   - This helps speed up training and improves convergence.
2. Reshaping
   - Original images are 28x28 grayscale (2D).
   - They were reshaped into 4D tensors with shape (num_samples, 28, 28, 1) to match the input requirements of Convolutional Neural Networks (CNNs).
3. Label Encoding
   - Digit labels (0–9) were one-hot encoded to represent them as categorical vectors.
   - Example: digit '3' becomes [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].
4. Train-Test Split o The MNIST dataset already provides a standard split:

- 60,000 training images 10,000 test images
-

5. Data Augmentation (Optional for robustness)

   o To prevent overfitting and improve generalization, image augmentation techniques like rotation, shifting, and zooming were applied using Keras' ImageDataGenerator.

   o This creates new variations of images on the fly during training.

6. Final Output

   o After preprocessing, the data was ready for feeding into the CNN model with normalized, reshaped images and encoded labels.

6. Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps to understand the structure, distribution, and patterns in the data before modeling. For the MNIST digit recognition task, the following EDA steps were conducted:

1. *Class Distribution Check* o *Verified the number of samples per class (digits 0–9).*
   - o *Found uniform distribution: ~6,000 samples for each digit in the training set.*
2. *Sample Visualization* o *Displayed a grid of sample images using matplotlib to visually inspect digit quality.*
   - o *Confirmed variations in handwriting styles, thickness, and orientation.*
3. *Pixel Intensity Analysis* o *Calculated average pixel intensity per image and per digit class.*
   - o *Revealed that some digits (like 1 and 7) tend to have fewer active pixels compared to others (like 8 or 0).*
4. *Image Dimension Consistency* ° *All images confirmed to be 28x28 pixels in size.*
   - o *No resizing was necessary.*

5. *Heatmaps of Average Digits* ○ *Created mean images for each digit by averaging pixel values across all samples of that digit.*
    - *Helped visualize the typical shape and pixel distribution per digit.*
6. *Insights Gathered* ○ *The dataset is clean, balanced, and well-suited for training a deep learning model.*
    - *Despite being grayscale, sufficient variation exists to test model robustness*

7. Feature Engineering

No manual features—relied on automatic feature extraction via Convolutional Neural Networks (CNNs)

Data augmentation techniques used to enhance training diversity

Used dropout layers in the model for regularization

8. Model Building

- *Algorithms Used:*

○ *Convolutional Neural Network (CNN): For spatial feature extraction and classification*

- *Model Architecture:*

○ *Input Layer → Conv2D → MaxPooling → Conv2D → Flatten → Dense → Output (Softmax)* ●

Train-Test Split:

○ *Predefined MNIST split (60,000 training / 10,000 testing)* ●

Evaluation Metrics:

○ *Accuracy: Percentage of correct predictions*

○ *Confusion Matrix: Visual analysis of misclassifications*

○ *Loss: Cross-entropy loss tracked over epochs*

9. Visualization of Results & Model Insights ●

Training Progress:

○ Plotted training and validation accuracy/loss over epochs ○ Monitored overfitting signs through validation curves

- Confusion Matrix:

○ Showed which digits were commonly misclassified

○ Most errors occurred between similar-looking digits (e.g., 4 and 9)

● User Testing:
○ Integrated model with Gradio to allow users to draw or upload digit images ○ Real-time prediction results with confidence scores

10. Tools and Technologies Used

- *Programming Language: Python 3*
- *Notebook Environment: Google Colab / Jupyter Notebook* • *Key Libraries:*

○ *tensorflow, keras for deep learning*
○ *numpy, pandas for data manipulation*
○ *matplotlib, seaborn for visualization*
○ *Gradio for creating an interactive web interface*

11. Team Members and Contributions

- R.DHASHVANDHAN: DataPreprocessing,Model
- M.DHRANIDHARAN:Deployment
- V.DHANUSH KUMAR: EDA
- S.DHEENADHAYALAN:Report Preparation