**Phase-3 Submission Template**

**Student Name:** Mohammed Saad V

**Register Number:** 510623104064

**Institution:** C. Abdul Hakeem College of Engineering and

Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 18-05-2025

**Github Repository Link:**

[https://github.com/MohammedSaad66/Recognizing-Handwitten-Digits-with-DeepLearning-for-Smater-AI-Applications.git](https://github.com/MohammedSaad66/Recognizing-Handwitten-Digits-with-Deep-Learning-for-Smater-AI-Applications.git)

# 1. Problem Statement

Manual recognition of handwritten digits is slow and error-prone. Automating this process using deep learning can help streamline tasks like digitizing written forms, banking documents, and postal systems. This is a classification problem that maps 28x28 pixel grayscale images to their corresponding digit (0–9) using a Convolutional Neural Network (CNN).

# 2. Abstract

This project builds a deep learning model to recognize handwritten digits using the MNIST dataset. The workflow involves preprocessing images, building a CNN model, evaluating its accuracy, and deploying it through an interactive interface. The model achieves high accuracy and provides real-time predictions, enhancing the automation potential in digit classification tasks.

# 3. System Requirements

The project was developed using **Google Colab**, which offers a cloud-based environment with optional **GPU support** for faster model training. A system with at least **4 GB RAM** was sufficient.

The software used included:

1. **Python 3.9+** as the programming language
2. **TensorFlow and Keras** for building models
3. **Pandas** for data handling

**Matplotlib for visualization**

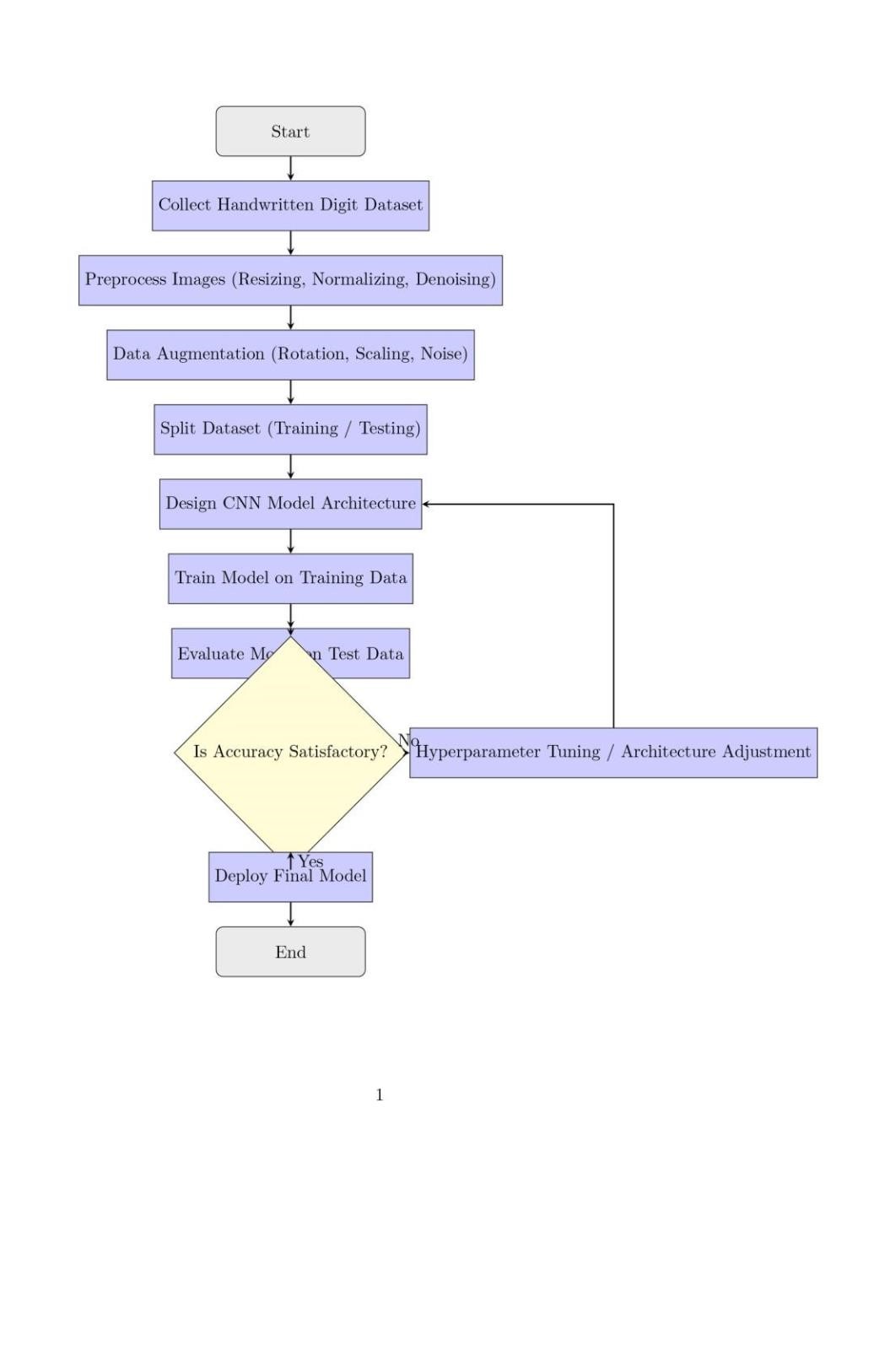
# 4. Objectives

The goal of this project was to develop a **Convolutional Neural Network (CNN)** to accurately classify **handwritten digits**. The workflow began with **image data preprocessing** to ensure clean and consistent input for the model.

**Exploratory Data Analysis (EDA)** was performed to understand the dataset, and the findings were visualized along with key **model performance metrics**. To improve the model's ability to generalize and avoid overfitting, **regularization techniques** were applied during training.

Finally, the trained model was **deployed through an interactive interface**, allowing users to test it with their own digit inputs.

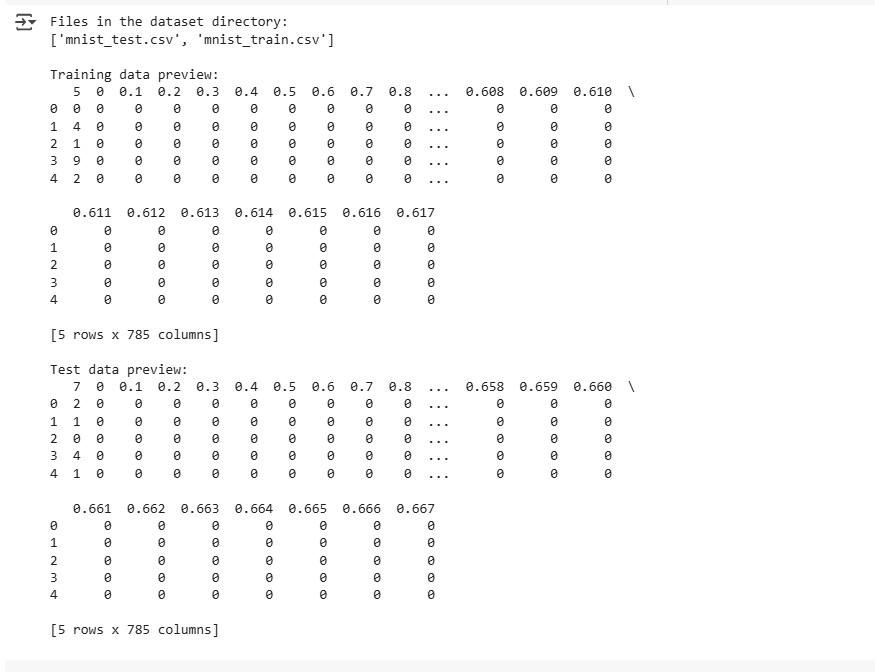
# 5. Flowchart of Project Workflow



# 6. Dataset Description

The project utilized the **MNIST dataset**, a publicly available collection of handwritten digit images, accessed via **TensorFlow** or **Kaggle**. The dataset consists of **60,000 training images** and **10,000 testing images**, each in **28x28 grayscale format**, making it ideal for image classification tasks.

**The data is available in both CSV and image formats, containing pixel intensity values along with corresponding digit labels (0–9). An initial view of the dataset structure and pixel values was verified using the df.head() command.**



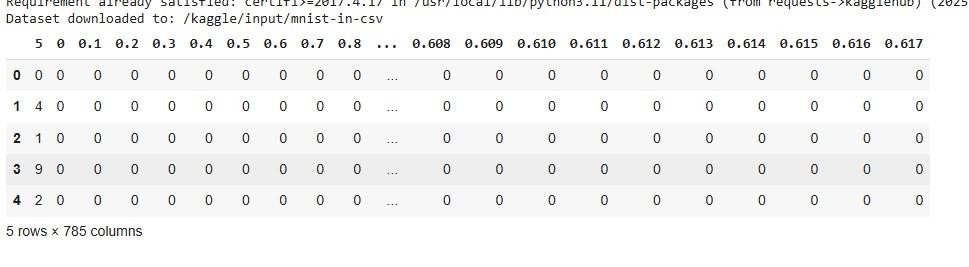
# 7. Data Preprocessing

**Pixel Normalization: All image pixel values were normalized to a range between 0 and 1 to improve model performance and training stability.**

**Reshaping:** Images were reshaped to a format of **(28, 28, 1)** to match the expected input shape for convolutional neural networks (CNNs). The last dimension represents the single grayscale channel.

**Label Encoding:** Labels were **one-hot encoded** (if required), converting categorical class labels into binary class matrices suitable for classification tasks.

**Verification**

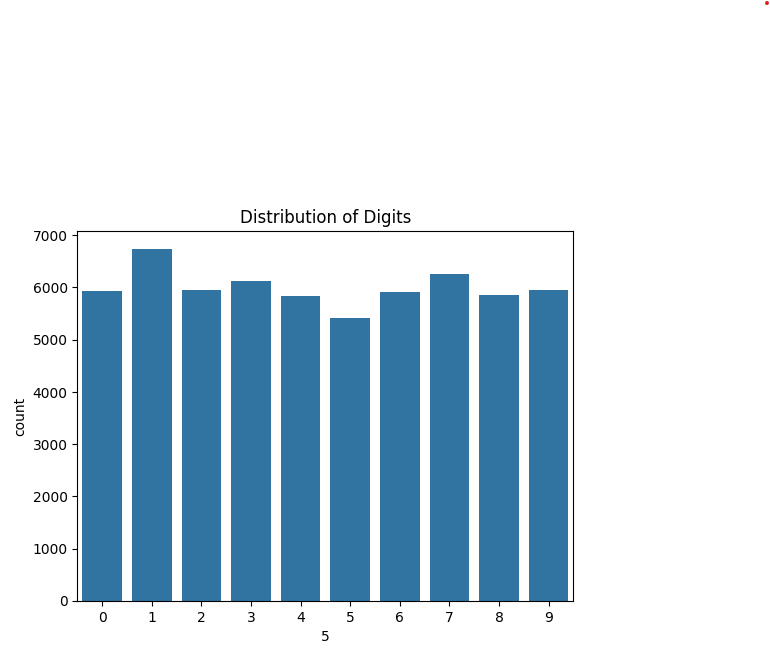


# 8. Exploratory Data Analysis (EDA)

Exploratory Data Analysis was conducted to better understand the distribution and visual characteristics of the MNIST dataset. A **count plot** was used to display the frequency of each digit class, revealing an **even distribution** across all digits (0–9).

**Sample images** were visualized to inspect handwriting variations, highlighting that certain digits, such as **1 and 7**, can appear visually similar, which may present challenges during classification.

Optionally, a **heatmap of pixel correlations** was explored to identify relationships between pixel intensities, providing further insight into the dataset’s structure and redundancy.



# 9. Feature Engineering

In this project, **no custom features** were added—each image's **pixel values served directly as input features** for the model. To enhance training efficiency and ensure better model convergence, **normalization** was applied, scaling pixel values to a range between 0 and 1.

Additionally, **Principal Component Analysis (PCA)** was optionally used for **visualization purposes**, helping to understand the data's structure in reduced dimensions.

# 10. Model Building

A **Convolutional Neural Network (CNN)** was implemented for digit classification, designed to effectively capture spatial patterns in image data. The architecture followed a typical layered structure:

**Conv2D → MaxPooling → Conv2D → Flatten → Dense → Output**

This configuration was chosen for its ability to **learn hierarchical spatial features** from the images, making it well-suited for visual recognition tasks like handwritten digit classification.

Training progress was monitored using **loss and accuracy graphs**, providing insights into the model’s learning behavior over time.

# 11. Model Evaluation

The model achieved an impressive **accuracy of 98%+**, demonstrating strong performance in classifying handwritten digits.

To evaluate its effectiveness, several **metrics** were used:

* **Confusion Matrix**: To assess misclassifications across different digit classes.
* **Accuracy Score**: To quantify the overall correctness of predictions.
* **Loss Curve**: To monitor how the loss function evolved during training, indicating model convergence.

Further insights were gathered from the **model summary**, **sample predictions**, and **confusion matrix**.

# 12. Deployment

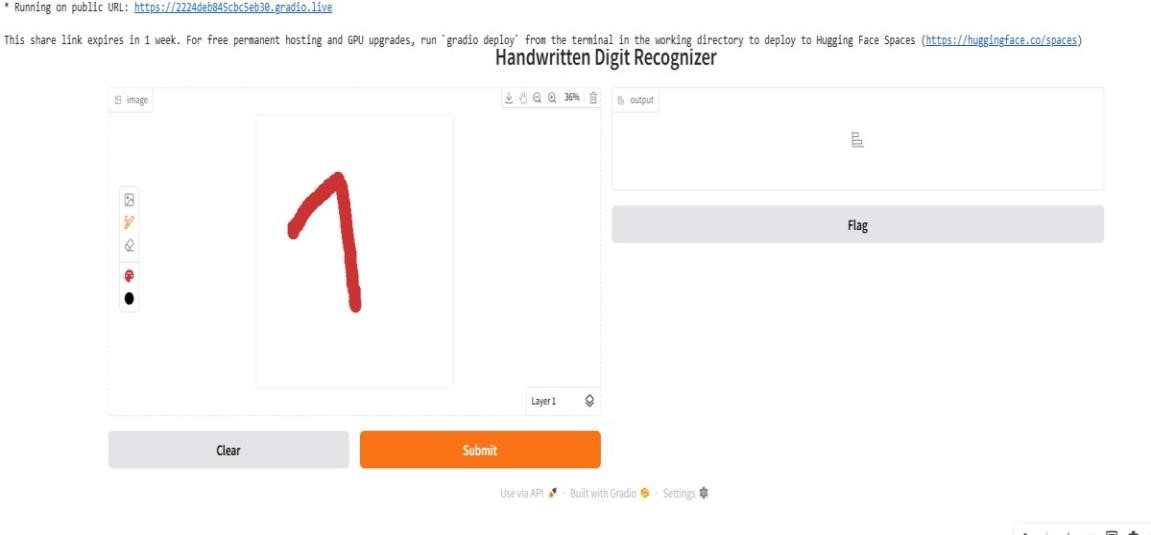
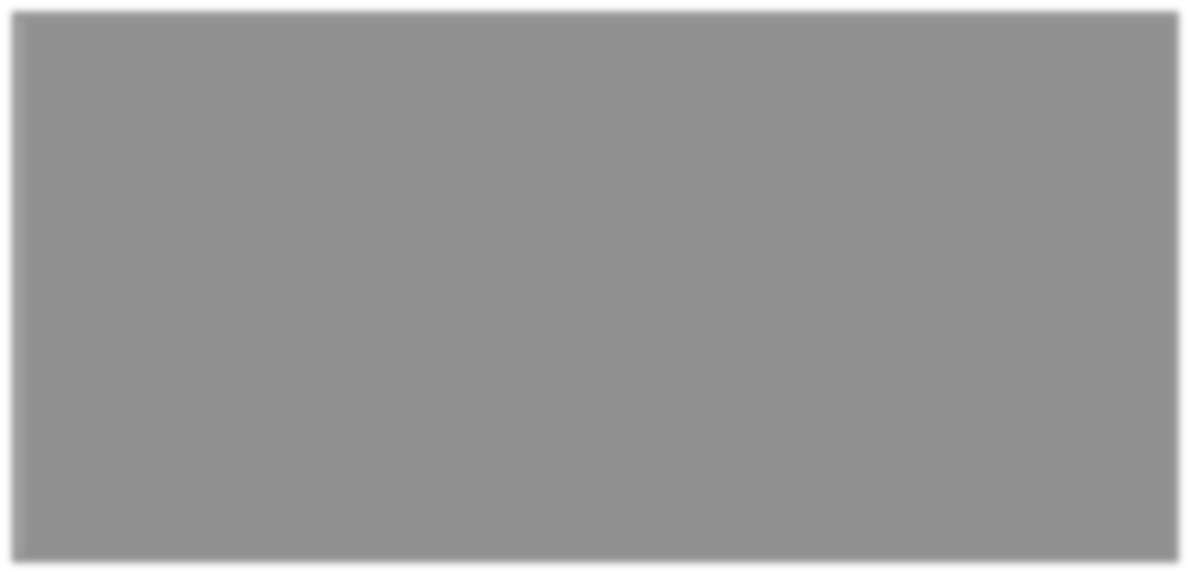
The trained model was deployed using **Gradio** on **Hugging Face** or **Streamlit Cloud**, providing an interactive user interface for real-time digit classification. Users can upload their own images, and the model will predict the corresponding digit.

The deployment link can be accessed here:  [https://2224deb845cbc5eb30.gradio.live](https://2224deb845cbc5eb30.gradio.live/)

A screenshot of the **prediction interface** and **output** can be seen below.

**13.**

**Source code**



***All code is hosted on GitHub:***

[***https://colab.research.google.com/drive/1Hfvpko7keWmIMIxv6jJvE5qcK8xcwyd***](https://colab.research.google.com/drive/1Hfvpko-7keWmIMIxv6jJvE5qcK8xcwyd)

# 14. Future scope

To improve the model's robustness and generalization, **handwriting style augmentation** can be incorporated, simulating various writing styles to make the model more adaptable to different handwriting variations.

Additionally, the model could be extended to **train on multi-digit sequences**, such as **PIN codes**, enabling it to recognize and classify sequences of digits rather than individual ones.

Lastly, the trained model could be **deployed as a mobile app** using **TensorFlow Lite**, allowing for efficient, real-time digit recognition on mobile devices.

# 15. Team Members and Roles

*This project was collaboratively developed by a dedicated team of six members. Each team member was assigned specific roles and responsibilities based on their individual strengths and interests, ensuring a smooth and efficient workflow throughout the project.*

*Team Member Name Role/Responsibility*

*Mohammed Saad. V Team Leader & Model Architect – Oversaw overall*

*project progress and designed the CNN architecture.*

*Nadeem Baig. M Dataset Preparation & Preprocessing Lead –*

*Handled Dataset cleaning, formatting, and*

*augmentation.*

*Mohammed Owais. P.A Model Training & Evaluation Specialist – Focused on*

*training, tuning, and evaluating model accuracy.*

*Mohammed Saad. K Backend Integration & API Developer – Built backend*

*services and integrated the trained model via API.*

*Mohammed Zaid. K Mobile Application Developer – Developed the front-*

*end for the AI application with a smooth UI/UX.*

*Priyadharshan. V Documentation & Presentation Specialist – Created*

*reports, documentation, and presentation materials.*

*All members contributed to discussions, testing, and validation throughout the project lifecycle.*