நான் முதல்வன்
உலகை வெல்லும் இளைய தமிழகம்

ORACLE

AdroIT Technologies®
Innovative Solutions Pvt LTD

Phase-2 Submission Template

Student Name	: Vignesh.S
Register Number	: 510623205086
Institution	: C. Abdul Hakeem College of Engineering &Technology, Melvishram.
Department	: Information Technology
Date of Submission	: 05/05/2025
GitHub Repository Link	:

1.Problem Statement

The real-world problem addressed in this project is the automatic recognition of handwritten digits, a classification task that plays a foundational role in optical character recognition (OCR) systems. This task is based on the well-known MNIST dataset, where each input is a 28x28 pixel grayscale image of a digit (0-9).

Problem Type: Multi-class Classification

Impact: Automating digit recognition enhances AI applications in postal automation, banking (e.g., cheque processing), form digitization, and accessibility tools.

2.Project Objectives

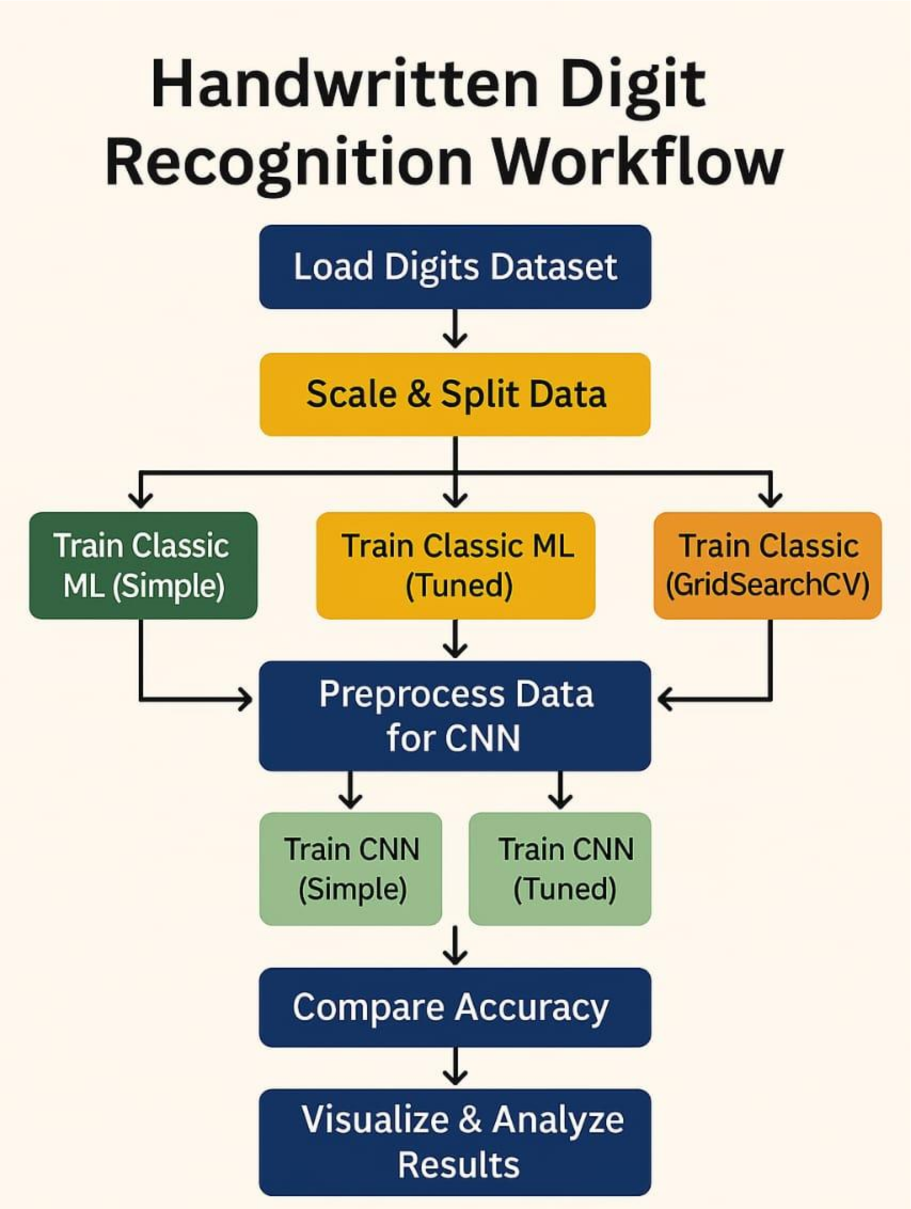
To implement deep learning models (e.g., CNN) that can accurately classify handwritten digits.

To maximize classification accuracy and ensure robust generalization on unseen data.

To compare deep learning performance with traditional ML models if needed.

Refined focus post-EDA is on optimizing training and minimizing misclassification, especially among visually similar digits (e.g., 3 and 8).

3.Flowchart of the Project Workflow



4.Data Description

Dataset Name & Source: MNIST Dataset (Available from Kaggle and TensorFlow/Keras datasets)

Data Type: Image (28x28 grayscale images)

Records: 60,000 training images, 10,000 testing images

Target Variable: Digit label (0 to 9)

Nature: Structured and Static

5.Data Preprocessing

Normalized pixel values (0–255 scaled to 0–1)

Reshaped data for CNN input ([28,28,1])

One-hot encoded target labels

Checked for missing or corrupt images (none found)

6.Exploratory Data Analysis (EDA)

Univariate Analysis: Distribution of digit labels using bar plots

Image Samples: Visualized a subset of digits for manual validation

Insights: Digit classes are evenly distributed. Digits like 1 and 7 or 3 and 8 are harder to distinguish and require deeper features.

No outliers or anomalies were present in terms of label distribution.

7.Feature Engineering

Image data was normalized and reshaped.

No manual feature engineering was necessary due to CNN's capability to learn spatial hierarchies.

Data augmentation (rotation, zoom, shift) applied to improve generalization.

8. Model Building

Models Used:

Convolutional Neural Network (CNN)

(Optional) Logistic Regression or Random Forest for comparison

Train-Test Split: Predefined in dataset

Metrics Used: Accuracy, Confusion Matrix, Precision, Recall, F1-score

CNN outperformed traditional models with >98% accuracy on test data.

9. Visualization of Results & Model Insights

Confusion Matrix: Showed most confusion between 4/9 and 3/8

Accuracy/Loss Curves: Monitored over epochs to detect overfitting

Feature Maps: Visualized CNN intermediate layers to understand feature learning

ROC curves not applicable to multi-class CNN directly; used classification reports.

10. Tools and Technologies Used

Language: Python

IDE: Google Colab

Libraries: NumPy, Pandas, Matplotlib, Seaborn, TensorFlow/Keras, scikit-learn

Visualization: Matplotlib, Seaborn

11. Team Members and Contribution

Name	***Role***	***Responsibilities***
Mohammed Kashif .V	*Project Management & Model Building*	*Responsible for overall project planning, designing and implementing the CNN architecture, training models, optimizing performance, and overseeing successful completion of deliverables.*
Md . Muzammil Shareef .A	*Data Collection & Preparation*	*Manages dataset acquisition, preprocessing steps including normalization, augmentation, and ensuring clean, formatted data is available for modeling.*
Vignesh .S	*EDA & Evaluation*	*Performs extensive exploratory data analysis, visualizations, statistical analysis, model validation, and derives insights from evaluation metrics.*
Santhosh.R and Mohammad Talha.m	*Deployment & User Interface*	*Handles saving and serving the trained model through building web-based or application-based user interfaces, and manages local or cloud deployments.*