NAAN MUDHALVAN PROJECT

TITLE:

RECOGNIZING HANDWRITTEN DIGITS WITH DEEP LEARNING FOR SMARTER AI APPLICATION

SUBMITTED BY

STUDENT NAME	ABINISHA A
REGISTER NO	510223205001
INSTITUITION	ANNAMALAIAR COLLEGE OF ENGINEERING
DEPARTMENT	INFORMATION TECHNOLOGY
DATE OF SUBMISSION	03-05-2025

GITHUB REPOSITORY LINK:https://github.com/munni437/phase2/blob/main/PROJECT%20PHASE%20II%20(1)

1. Problem Statement

Recognizing handwritten digits is a classic problem in computer vision that has practical applications in postal automation, bank check processing, and digitized note-taking. Manual digit recognition is time-consuming and error-prone. By using deep learning techniques, we aim to develop an automated, accurate, and scalable solution to recognize handwritten digits, contributing to smarter AI applications.

2. Objectives of the Project

- To develop a deep learning model capable of accurately classifying handwritten digits from images.
- To achieve high accuracy and low loss using convolutional neural networks (CNNs).
- To gain insights into model behavior through visualization techniques like confusion matrices and activation maps.
- To explore real-time deployment options for the model.

3.Flowchart of the project workflow

Step 1: Data Collection
→ Download the MNIST handwritten digits dataset.

Step 2: Data Preprocessing
→ Normalize pixel values, reshape images, and encode labels.

Step 3: Exploratory Data Analysis (EDA)
→ Visualize data distributions and identify useful patterns.

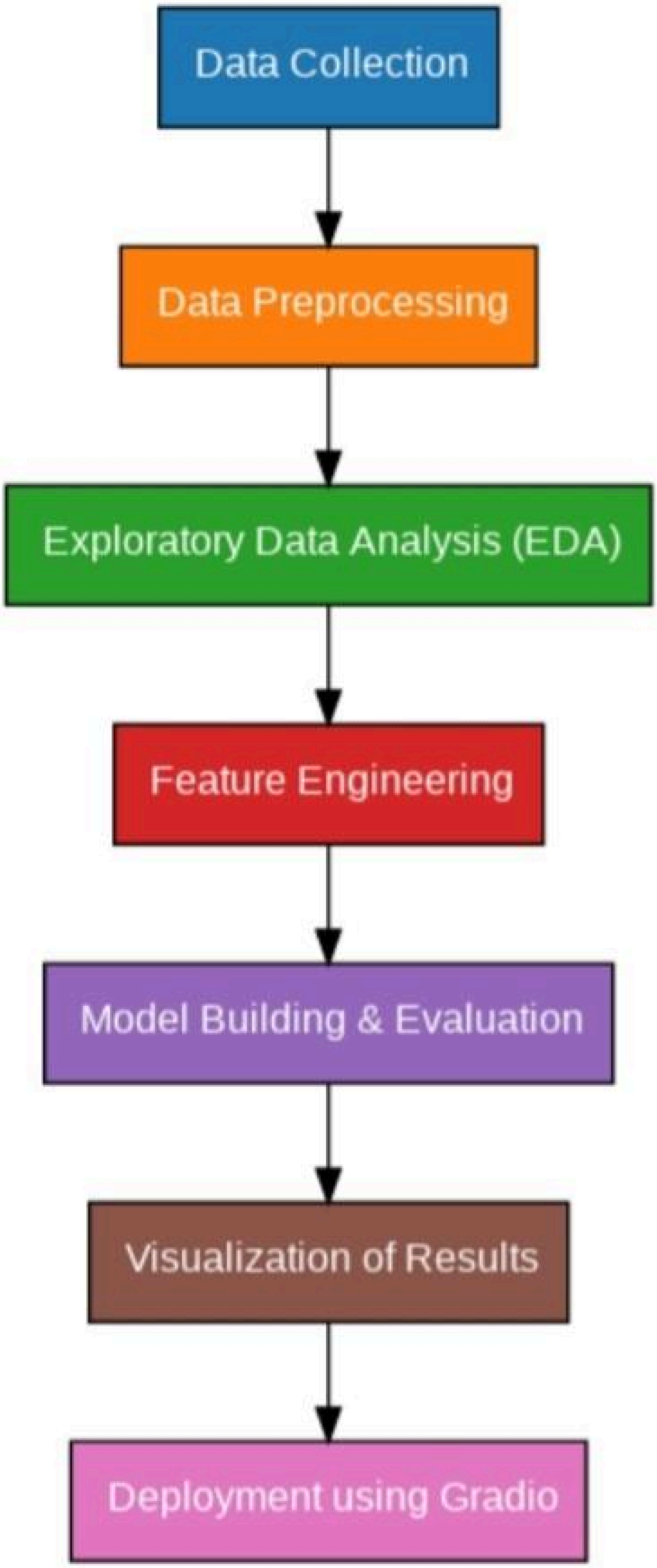
Step 4: Feature Engineering
→ Prepare data for CNN input (e.g., reshaping, augmentation).

Step 5: Model Building and Training
→ Train a Convolutional Neural Network to classify digits.

Step 6: Model Evaluation
→ Measure accuracy, precision, recall, and F1-score.

Step 7: Deployment
→ Build a web interface for real-time digit prediction

3. Flowchart of the Project Workflow



4. Scope of the Project

- **Features**: Pixel intensity values of images (28x28 grayscale).
- **Modeling**: Use CNN architectures (e.g., LeNet, CNN with BatchNorm, Dropout).
- **Limitations**: The project will be limited to the MNIST dataset and similar datasets due to computational constraints.
- **Tools**: Restricted to Python-based open-source libraries and deployment on web-based platforms (optional).

5. Data Sources

- **Dataset**: MNIST handwritten digits dataset. https://www.tensorflow.org/datasets/catalog/mnist
- **SourceType**: Public and static

6. High-Level Methodology

- **Data Collection**: Dataset will be downloaded using Keras/TensorFlow dataset utilities or from Kaggle.
- **Data Cleaning**: Check for anomalies like corrupt or mislabeled images; ensure correct formatting and normalization (pixel values scaled 0-1).

```
from tensorflow.keras.datasets import mnist


# Load dataset

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

print(f"Training samples: {x_train.shape}, Test samples: {x_test.shape}")
```

Exploratory Data Analysis (EDA):

- Display sample digits per class.
- Use heatmaps to understand Visualize digit distribution.
- average patterns.

```
# Save this as app.py
import streamlit as st
from tensorflow.keras.models import load_model
import numpy as np
from PIL import Image

model = load_model('your_model.h5')  # Save your model first

st.title("Digit Recognizer")

uploaded_file = st.file_uploader("Choose an image...", type=["png",
"jpg", "jpeg"])

if uploaded_file is not None:
    image = Image.open(uploaded_file).convert('L').resize((28, 28))
    img_array = np.array(image) / 255.0
    img_array = img_array.reshape(1, 28, 28, 1)
    prediction = model.predict(img_array)
```

Feature Engineering:

- Normalize pixel values.

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout


model = Sequential([

    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),

    MaxPooling2D((2,2)),

    Conv2D(64, (3,3), activation='relu'),

    MaxPooling2D((2,2)),

    Flatten(),

    Dense(128, activation='relu'),

    Dropout(0.5),

    Dense(10, activation='softmax')

])


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

🎬 Model Evaluation:

- Metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix.
- Use train/test split or K-fold cross-validation.

```
# Train the model

model.fit(x_train, y_train_cat, epochs=5, validation_split=0.2, batch_size=64)


# Evaluate on test data

test_loss, test_acc = model.evaluate(x_test, y_test_cat)
```

```
print(f"Test Accuracy: {test_acc:.4f}")
```

🎬 Visualization & Interpretation:

```
# Plot training history

import matplotlib.pyplot as plt


history = model.history.history

plt.plot(history['accuracy'], label='Train Accuracy')

plt.plot(history['val_accuracy'], label='Validation Accuracy')

plt.legend()

plt.title('Model Accuracy')

plt.show()
```

- **Deployment** (optional):
  - Streamlit or Gradio app to take an image input and display the predicted digit.

7. Tools and Technologies

- **Programming Language**: Python
- **Notebook/IDE**: Google Colab / Jupyter Notebook
- **Libraries**:
  - o Data Handling: `numpy`, `pandas`
  - o Visualization: `matplotlib`, `seaborn`
  - o Modeling: `tensorflow`, `keras`, `scikit-learn`
- **Optional Deployment Tools**: `Streamlit`, `Gradio`, `Flask`

8. Team Members and Roles

1.MUNNI A	**Data preprocessing,model development,documentation and presentation**
2.ABINISHA A	**Visulaization and model performance evaluation**
3.HANI HAIRUS A	**Interface design and optional development**