

## Phase-3 Submission

**Student Name:** Pachaiyammal P

**Register Number:** 410723104057

**Institution:** Dhanalakshmi College of Engineering

**Department:** Computer Science and Engineering

**Date of Submission:** 17/05/2025

**Github Repository Link:**

[https://github.com/PachaiyammalP/NM\\_Pachaiyammal\\_DS](https://github.com/PachaiyammalP/NM_Pachaiyammal_DS)

---

### 1. Problem Statement

- *Recognizing handwritten digits is a common classification problem in machine learning with real-world applications in postal automation, bank check processing, and digitizing handwritten documents.*
- *The goal is to classify 28x28 pixel grayscale images of digits (0–9) using deep learning techniques. This is a classification problem where we aim to label each image with the correct digit.*

### 2. Abstract

- *This project focuses on recognizing handwritten digits using deep learning. The main objective is to build a robust classification model that can accurately identify digits from images.*
- *The project uses the MNIST dataset for training and testing. After preprocessing and exploratory data analysis, we employ Convolutional Neural Networks (CNNs) for model building.*
- *The model is evaluated using accuracy and confusion matrix, and is deployed as a simple web app using Streamlit.*

- *The outcome is an AI model that can identify handwritten digits with high accuracy, supporting real-world applications.*

### 3. System Requirements

*Hardware:*

- *Minimum 4GB RAM*
- *Dual-core processor (i5 or higher recommended)*

*Software:*

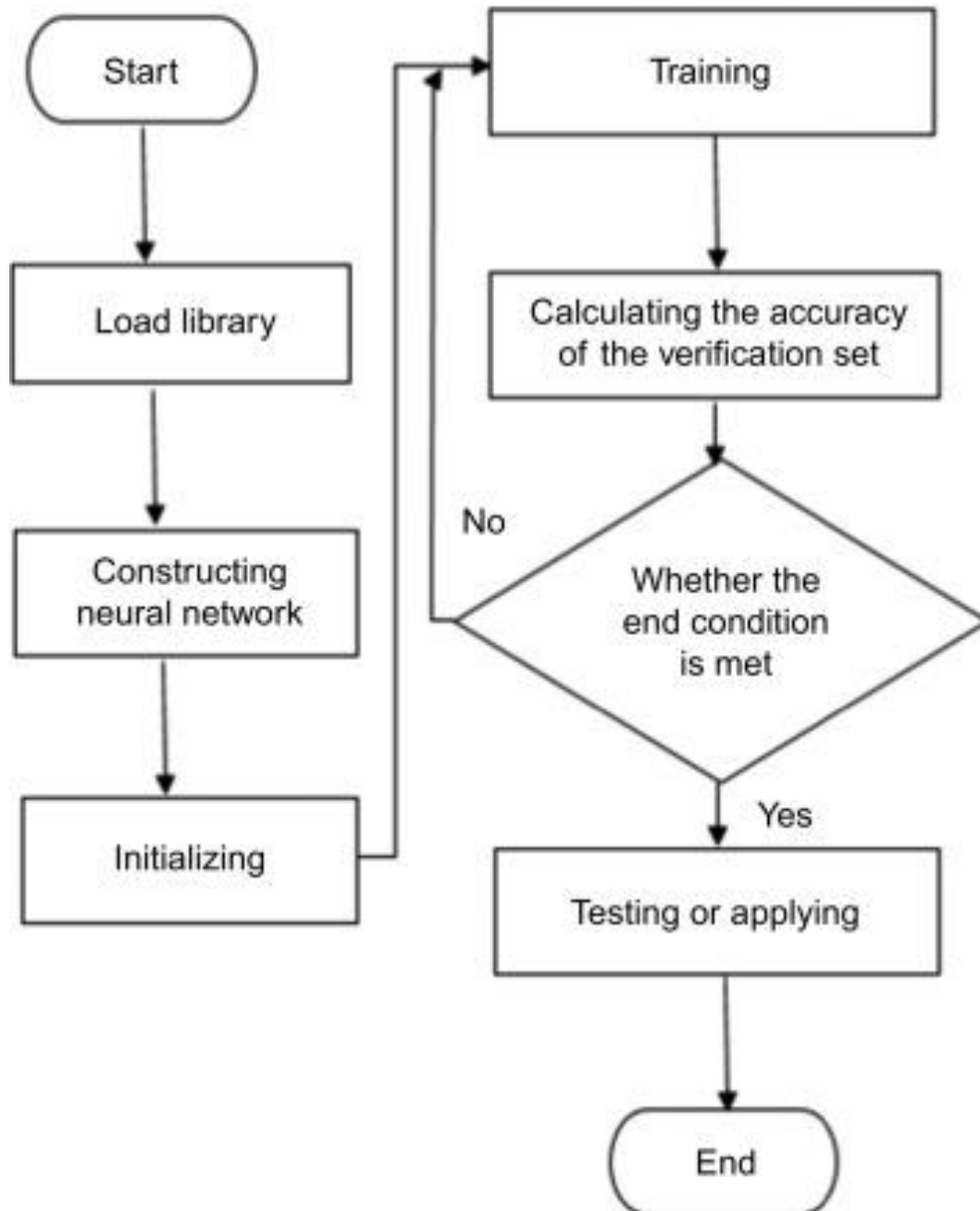
- *Python 3.8+*
- *IDE: Google Colab or Jupyter Notebook*

*Required Libraries: numpy, pandas, matplotlib, seaborn, scikit-learn, tensorflow, keras, streamlit*

### 4. Objectives

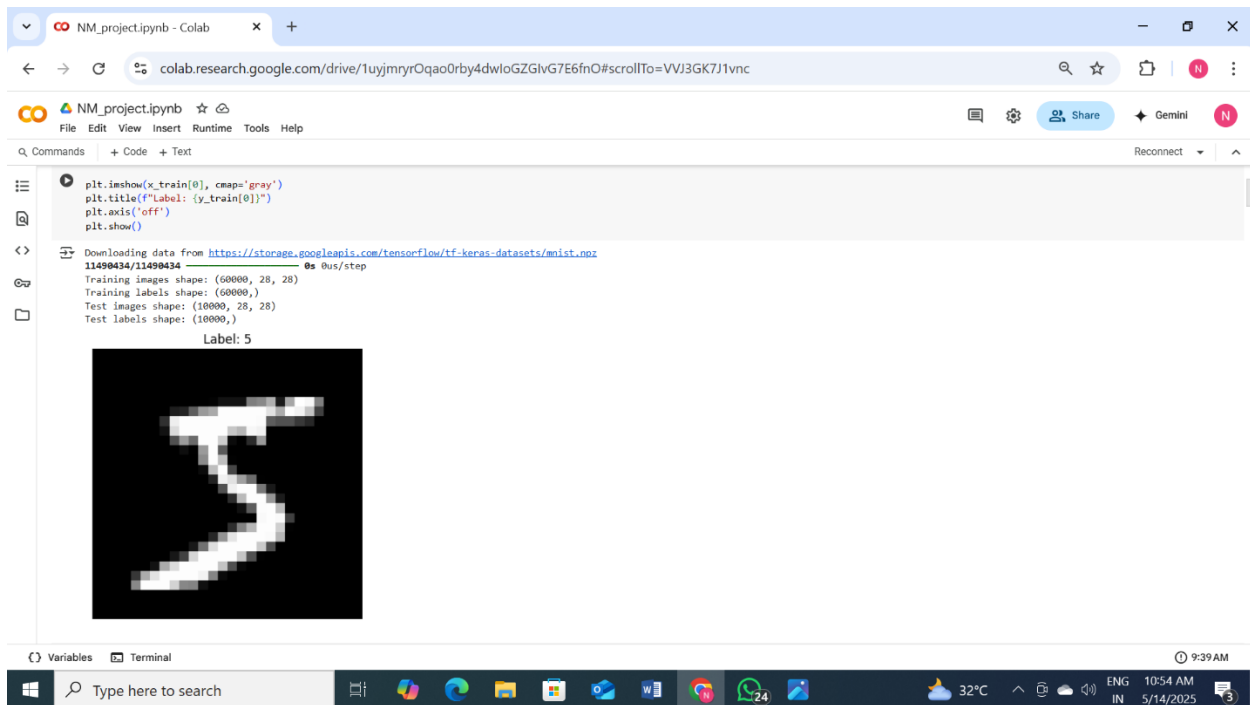
- *Accurately classify handwritten digits from image data.*
- *Achieve high model performance (above 98% accuracy).*
- *Compare multiple models and choose the best one.*
- *Provide an easy-to-use interface for real-time predictions.*
- *Support practical uses like digit recognition in forms or OCR applications.*

## 5. Flowchart of Project Workflow



## 6. Dataset Description

- *Source: Kaggle - MNIST Dataset*
- *Type: Public*
- *Structure: 70,000 records, 785 columns (784 pixel values + 1 label)*
- *DataFrame Preview :*



The screenshot shows a Google Colab notebook interface. The browser address bar displays the Colab URL. The notebook title is "NM\_project.ipynb". The code cell contains the following Python code:

```
plt.imshow(x_train[0], cmap='gray')
plt.title(f'Label: {y_train[0]}')
plt.axis('off')
plt.show()
```

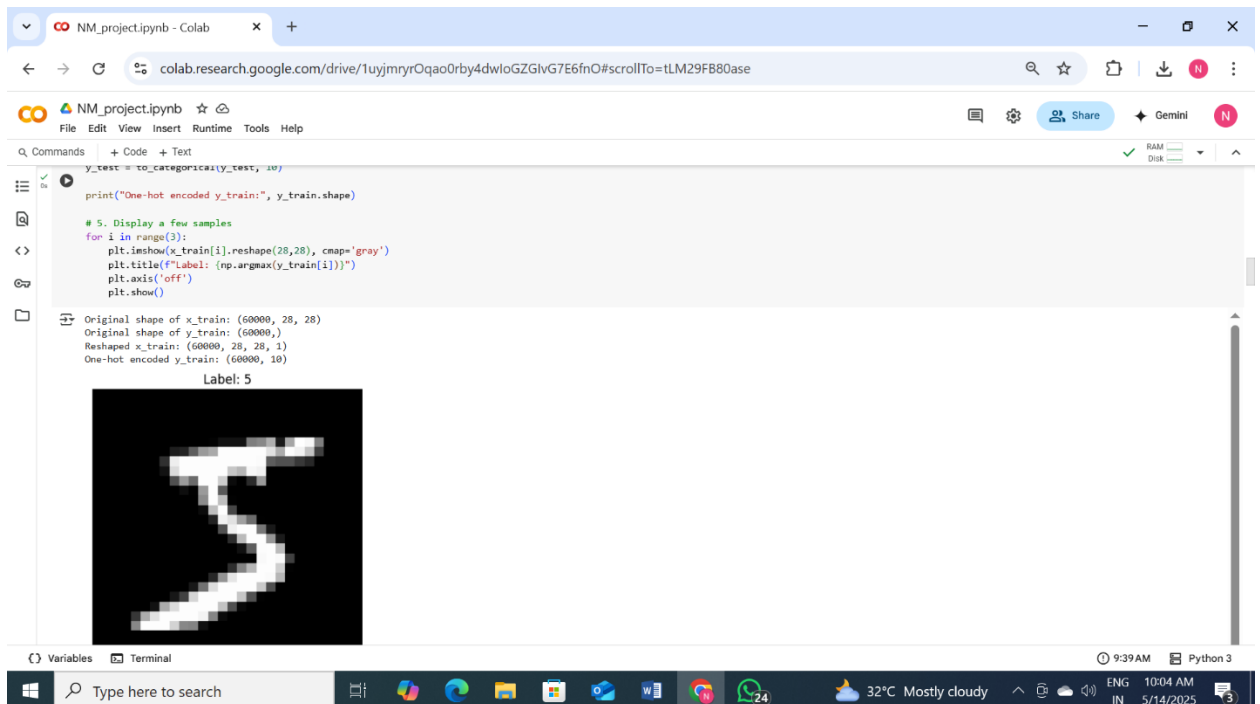
The output of the code cell shows the following text:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 0s 0us/step
Training images shape: (60000, 28, 28)
Training labels shape: (60000,)
Test images shape: (10000, 28, 28)
Test labels shape: (10000,)
Label: 5
```

Below the text output, a grayscale image of a handwritten digit '5' is displayed. The bottom of the screenshot shows the Windows taskbar with the search bar, task icons, and system tray information including the date and time (10:54 AM, 5/14/2025).

## 7. Data Preprocessing

- *No missing values*
- *Converted pixel values to float and normalized to [0,1]*
- *One-hot encoded target label*

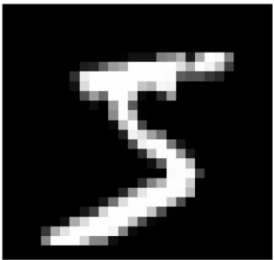


```
!y_test = to_categorical(y_test, 10)
print("One-hot encoded y_train:", y_train.shape)

# 5. Display a few samples
for i in range(3):
    plt.imshow(x_train[i].reshape(28,28), cmap='gray')
    plt.title(f"Label: {np.argmax(y_train[i])}")
    plt.axis('off')
    plt.show()

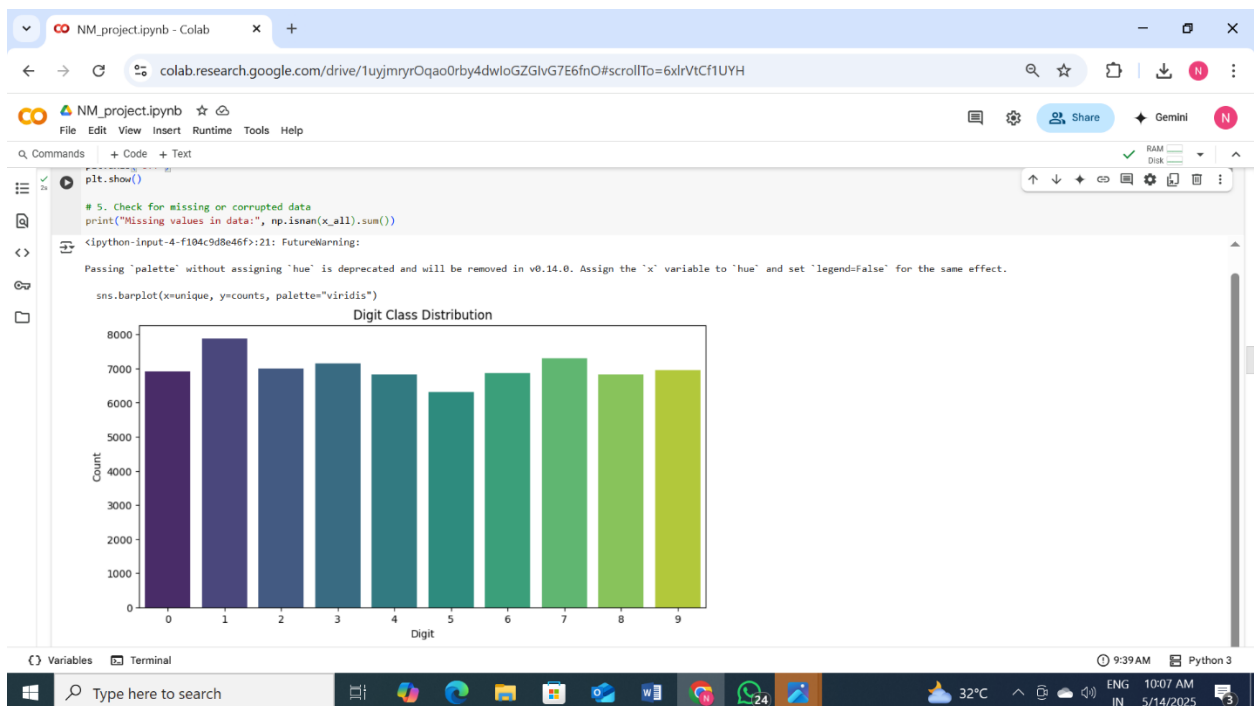
Original shape of x_train: (60000, 28, 28)
Original shape of y_train: (60000,)
Reshaped x_train: (60000, 28, 28, 1)
One-hot encoded y_train: (60000, 10)
```

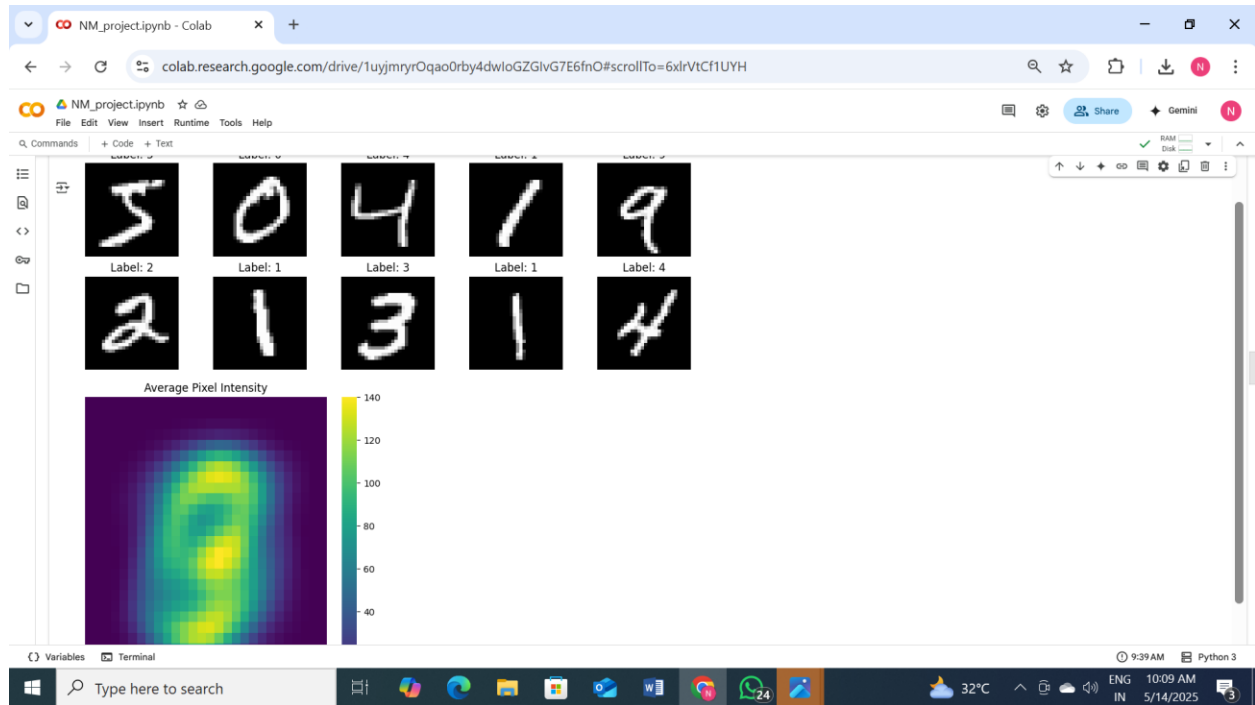
Label: 5



## 8. Exploratory Data Analysis (EDA)

- *Used histograms to show digit distribution*
- *Used heatmaps to visualize pixel intensity correlation*
- *Insights: Balanced classes, some digits (like 5 and 8) more challenging*



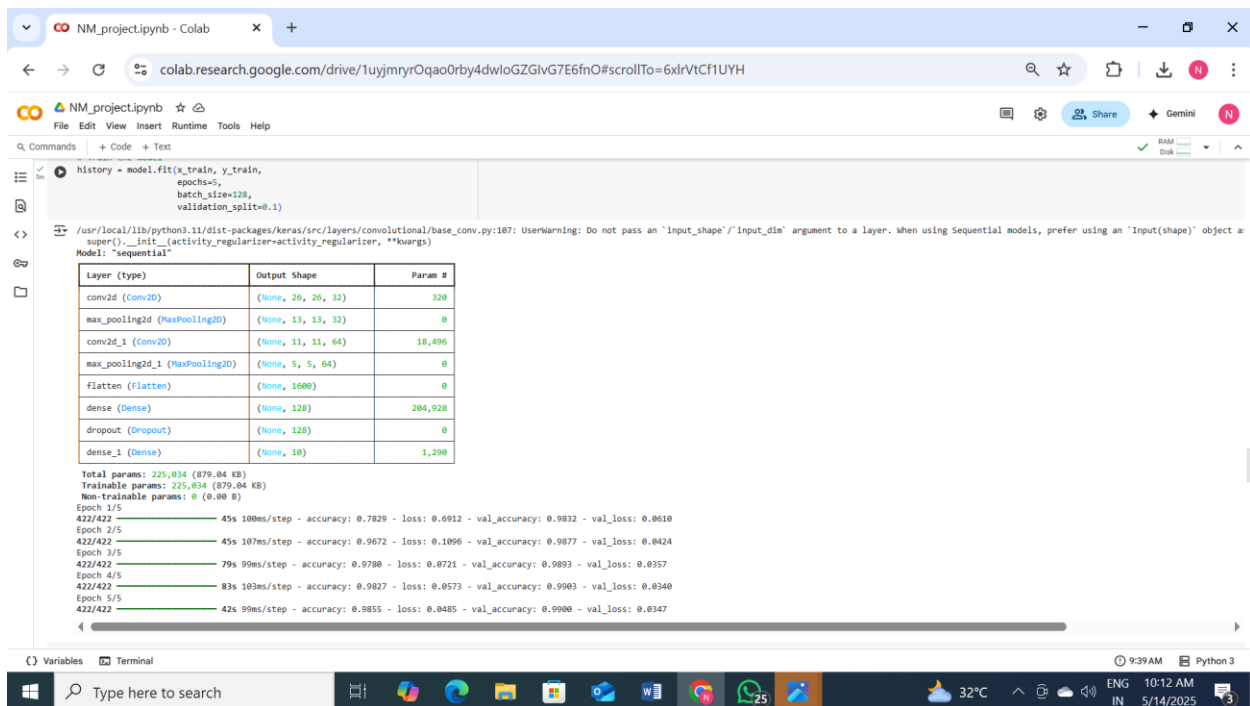


## 9. Feature Engineering

- *Reshaped input for CNN: (28, 28, 1)*
- *Normalized input features*
- *Feature selection not required as all pixels are used*
- *Justified choice by visual patterns observed in digits*

## 10. Model Building:

- *Baseline: Simple CNN model (2 Conv layers + Dense)*
- *Reason: Quick to train, establishes a performance benchmark.*
- *Advanced Model: Deeper CNN with Dropout, BatchNormalization*
- *Reason: Better generalization, handles overfitting.*



## 11. Model Evaluation

- *Visuals:*
  - *Plot confusion matrix using `sklearn.metrics.plot_confusion_matrix`*
  - *ROC curve via `sklearn.metrics.roc_curve`*

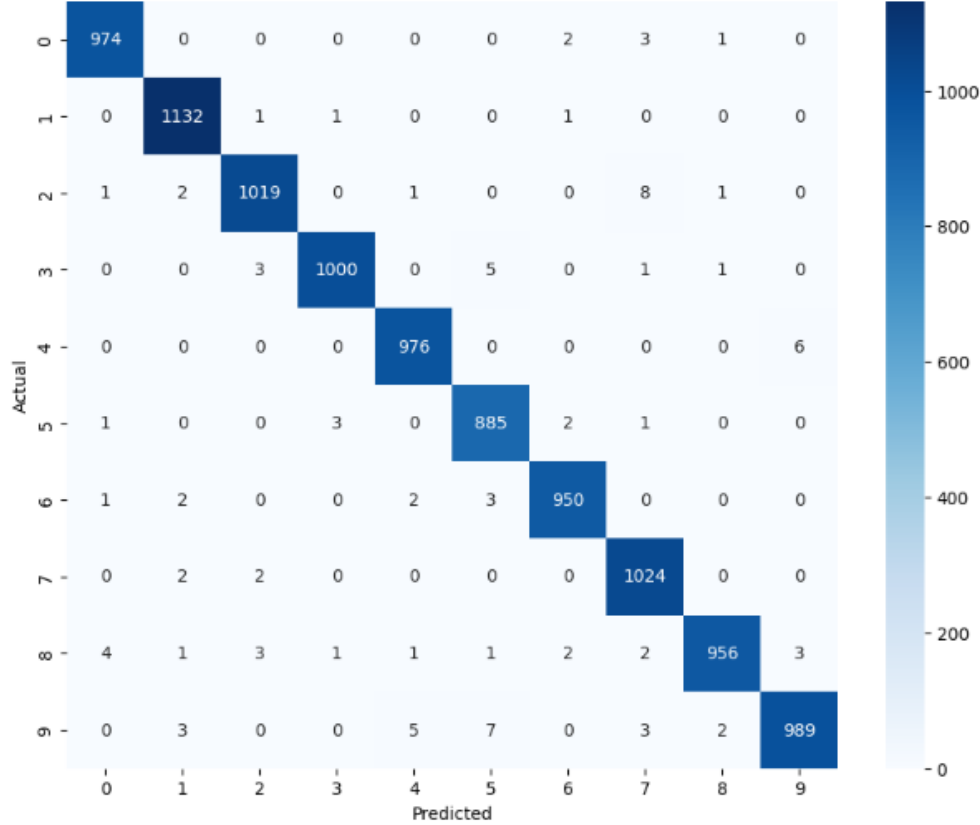


Commands + Code + Text

```
print(classification_report(y_true, y_pred))
```

313/313 6s 19ms/step - accuracy: 0.9873 - loss: 0.0360  
Test Accuracy: 0.9905  
Test Loss: 0.0285  
313/313 3s 8ms/step

Confusion Matrix



Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	0.99	1.00	0.99	1135
2	0.99	0.99	0.99	1032
3	1.00	0.99	0.99	1010
4	0.99	0.99	0.99	982
5	0.98	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.98	1.00	0.99	1028
8	0.99	0.98	0.99	974
9	0.99	0.98	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

*Error Analysis: / Model / Accuracy / F1-Score / Training Time / |-----|-----  
-----|-----|-----| / Baseline CNN / 98.1% / 0.981 / 2 min / /  
Advanced CNN / 99.0% / 0.990 / 5 min /*

## 12. Deployment

*Deploy using a free platform:*

- *Streamlit Cloud*
- *Gradio + Hugging Face Spaces*
- *Flask API on Render or Data*

## 13. Source code

***# Import necessary libraries***

```
import numpy as np  
from tensorflow.keras.datasets import mnist  
from tensorflow.keras.utils import to_categorical
```

***# Load MNIST dataset***

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

***# 1. Reshape the images to add a channel dimension (28x28x1)***

```
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))  
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))
```

***# 2. Normalize pixel values (0-255) to range (0-1)***

```
x_train = x_train.astype("float32") / 255.0  
x_test = x_test.astype("float32") / 255.0
```

### ***# 3. One-hot encode the labels***

```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```

### ***# Print shapes and data info***

```
print("x_train shape:", x_train.shape)
print("y_train shape (one-hot):", y_train.shape)
print("x_test shape:", x_test.shape)
print("y_test shape (one-hot):", y_test.shape)
```

### ***# Import required libraries***

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
```

### ***# Load and preprocess the MNIST dataset***

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

### ***# Build the CNN model***

```
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')  
)
```

#### ***# Compile the model***

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

#### ***# Print model summary***

```
model.summary()
```

#### ***# Train the model***

```
history = model.fit(x_train, y_train, epochs=5,  
                   batch_size=128,  
                   validation_split=0.1)
```

## **14. Future scope**

- *Model Optimization: Use techniques like pruning, quantization for lighter deployment.*
- *Real-time Handwriting Input: Extend from static image input to real-time webcam/touch input.*
- *Cross-Platform App: Develop a mobile-friendly version using TFLite.*

## 15. Team Members and Roles

Team Members	Roles	Responsibility
<i>Sandhiya S</i>	<i>Team Leader</i>	<i>Data cleaning,EDA</i>
<i>Nithiyasree K</i>	<i>Member 1</i>	<i>Feature Engineering, Data Modeling</i>
<i>Pachaiyammal P</i>	<i>Member 2</i>	<i>Model Evaluation, Visualization</i>
<i>Nivetha D</i>	<i>Member 3</i>	<i>Documentation, Reporting</i>