🌟 Tamizhan Skills SE RISE Internship – Machine Learning & AI

📝 Project 2: Handwritten Digit Recognition using CNN

👤 Name: Prachee
🏫 College: CRSSIET, Jhajjar
🎓 Course: B.Tech CSE (AI & ML), 2023–2027

📄 Project Summary:

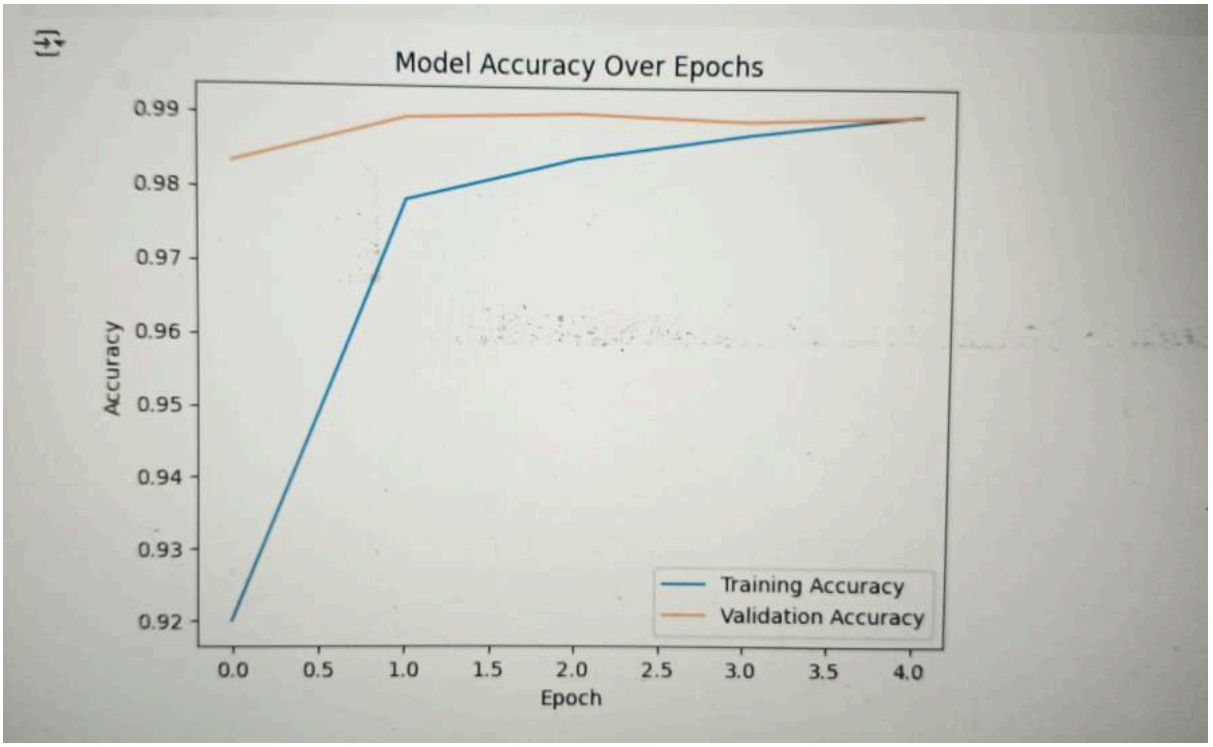
> Objective:
To develop a machine learning model that recognizes handwritten digits using Convolutional Neural Networks (CNN) and the MNIST dataset.

>Tools Used:
Google Colab
Python
TensorFlow / Keras
MNIST dataset (loaded via keras.datasets)

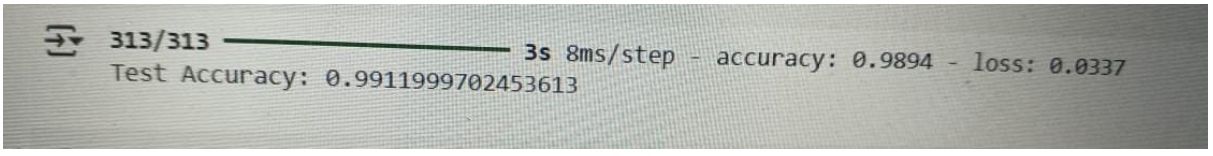
>Approach:
Loaded and preprocessed the MNIST dataset
Built a CNN with two convolutional + pooling layers
Trained model with dropout and dense layers
Achieved over 98% test accuracy

>Result:
The trained model successfully recognizes handwritten digits and can be used in digital exam checking and EdTech applications.

📊 Accuracy Graph:



✅ Final Test Accuracy Output:



🧠 Code Used (Google Colab):

- Import Libraries

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

- Load & Preprocess MNIST

```
# Load the MNIST dataset directly
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Reshape and normalize the 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

# One-hot encode the labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

- Build Model

```
model = Sequential([
    Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(64, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(10, activation='softmax')
])
```

- Train model

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

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

- Evaluate & Plot

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_acc)

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.legend()
plt.show()
```

> 🔍 This model can be further extended by adding a digit drawing UI using Python libraries like Tkinter or Streamlit.