import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

# Load MNIST dataset

mnist = tf.keras.datasets.mnist

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Preprocess the data

X\_train, X\_test = X\_train / 255.0, X\_test / 255.0  # Normalize pixel values to range [0,1]

X\_train = X\_train.reshape(-1, 28, 28, 1)  # Reshape for CNN input

X\_test = X\_test.reshape(-1, 28, 28, 1)

# Define CNN model

model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(28, 28, 1)),  # First convolutional layer

    tf.keras.layers.MaxPooling2D(2,2),  # Max pooling to reduce spatial size

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  # Second convolutional layer

    tf.keras.layers.MaxPooling2D(2,2),  # Max pooling

    tf.keras.layers.Flatten(),  # Flatten feature maps for dense layers

    tf.keras.layers.Dense(128, activation='relu'),  # Fully connected layer

    tf.keras.layers.Dense(10, activation='softmax')  # Output layer (10 classes for digits 0-9)

])

# Compile and train the model

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=5, validation\_data=(X\_test, y\_test))  # Train for 5 epochs

# Evaluate the model

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print(f"✅ Model Accuracy: {test\_acc \* 100:.2f}%")  # Print test accuracy

# Predict a new handwritten digit

index = np.random.randint(0, X\_test.shape[0])  # Select a random image

image = X\_test[index].reshape(1, 28, 28, 1)

prediction = model.predict(image)

predicted\_digit = np.argmax(prediction)  # Get predicted digit

# Display the image and model prediction

plt.imshow(X\_test[index].reshape(28, 28), cmap='gray')

plt.title(f"Predicted Digit: {predicted\_digit}")

plt.show()

Epoch 1/5

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **64s** 33ms/step - accuracy: 0.9116 - loss: 0.2990 - val\_accuracy: 0.9841 - val\_loss: 0.0466

Epoch 2/5

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **79s** 31ms/step - accuracy: 0.9877 - loss: 0.0408 - val\_accuracy: 0.9864 - val\_loss: 0.0395

Epoch 3/5

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **83s** 32ms/step - accuracy: 0.9922 - loss: 0.0258 - val\_accuracy: 0.9888 - val\_loss: 0.0341

Epoch 4/5

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **80s** 31ms/step - accuracy: 0.9936 - loss: 0.0189 - val\_accuracy: 0.9917 - val\_loss: 0.0256

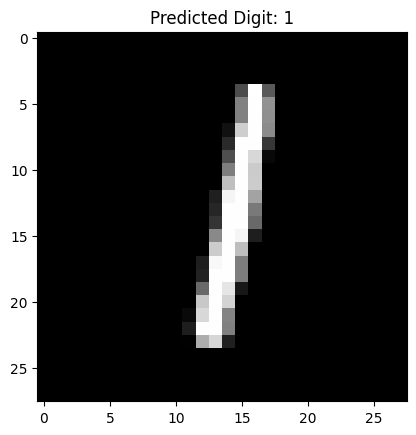
Epoch 5/5

**1875/1875** ━━━━━━━━━━━━━━━━━━━━ **60s** 32ms/step - accuracy: 0.9961 - loss: 0.0117 - val\_accuracy: 0.9892 - val\_loss: 0.0355

**313/313** ━━━━━━━━━━━━━━━━━━━━ **3s** 9ms/step - accuracy: 0.9859 - loss: 0.0442

Model Accuracy: 98.92%

**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 110ms/step



**مراحل کار :**

**مجموعه داده MNIST را بارگذاری کرده ایم (اعداد دست‌نویس از 0 تا 9).**

**تصاویر را پردازش کرده و با نرمال‌سازی و تغییر ابعاد برای ورودی CNN آماده کردیم.**

**یک مدل CNN با لایه‌های کانولوشنی و تجمیعی (Pooling) تعریف کردیم**

**مدل را به مدت 5 دوره (Epoch) با بهینه‌ساز Adam آموزش دادیم.**

**دقت مدل آموزش‌دیده را ارزیابی کردیم.**

**یک عدد دست‌نویس تصادفی را پیش‌بینی کرده و آن را نمایش دادیم**