import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import mnist

import matplotlib.pyplot as plt

# 1. Load MNIST dataset

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

# 2. Preprocess the data

# Normalize pixel values (0 to 1)

X\_train = X\_train.astype("float32") / 255.0

X\_test = X\_test.astype("float32") / 255.0

# Flatten images to vectors of 784 elements (28x28)

X\_train = X\_train.reshape(-1, 28 \* 28)

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

# One-hot encode labels

y\_train = tf.keras.utils.to\_categorical(y\_train, 10)

y\_test = tf.keras.utils.to\_categorical(y\_test, 10)

# 3. Build the model

model = models.Sequential([

tf.keras.Input(shape=(784,)), # Input layer for 28x28 flattened

layers.Dense(128, activation='relu'),

layers.Dense(64, activation='relu'),

layers.Dense(10, activation='softmax') # 10 classes for digits 0–9

])

# 4. Compile the model

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# 5. Train the model

history = model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_split=0.1)

# 6. Evaluate on test set

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

print(f"\nTest Loss: {test\_loss:.4f}")

print(f"Test Accuracy: {test\_accuracy:.4f}")