# Import libraries import tensorflow as tf import numpy as np import matplotlib.pyplot as plt # Load the MNIST dataset (x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data() # Preprocess the data by normalizing the pixel values and one-hot encoding the labels x\_train = x\_train / 255.0 x\_test = x\_test / 255.0 y\_train = tf.keras.utils.to\_categorical(y\_train, 10) y\_test = tf.keras.utils.to\_categorical(y\_test, 10) # Define the model architecture model = tf.keras.Sequential([ tf.keras.layers.Flatten(input\_shape=(28, 28)), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dense(10, activation='softmax') ]) # Compile the model and specify the loss function, optimizer, and metrics to track model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']) # Train the model on the training data and evaluate it on the test data history = model.fit(x\_train, y\_train, epochs=10, validation\_data=(x\_test, y\_test)) # Visualize the training and validation accuracy and loss acc = history.history['accuracy'] val\_acc = history.history['val\_accuracy'] loss = history.history['loss'] val\_loss = history.history['val\_loss'] epochs\_range = range(10) plt.figure(figsize=(8, 8)) plt.subplot(2, 2, 1) plt.plot(epochs\_range, acc, label='Training Accuracy') plt.plot(epochs\_range, val\_acc, label='Validation Accuracy') plt.legend(loc='lower right') plt.title('Training and Validation Accuracy') plt.subplot(2, 2, 2) plt.plot(epochs\_range, loss, label='Training Loss') plt.plot(epochs\_range, val\_loss, label='Validation Loss') plt.legend(loc='upper right') plt.title('Training and Validation Loss') plt.show() #Finally, we can evaluate the model on the test data and print the test accuracy: test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2) print('\nTest accuracy:', test\_acc) Output 313/313 - 0s - loss: 0.0718 - accuracy: 0.9777 Test accuracy: 0.9777