Code:

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.datasets import mnist

import matplotlib.pyplot as plt

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

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

x\_train = x\_train.reshape((x\_train.shape[0], 28, 28, 1))

x\_test = x\_test.reshape((x\_test.shape[0], 28, 28, 1))

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

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

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())

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

model.add(layers.Dense(10, activation='softmax'))

model.summary()

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

early\_stopping = EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)

history = model.fit(x\_train, y\_train, epochs=10, batch\_size=64, validation\_split=0.2, callbacks=[early\_stopping])

Epoch 1/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **25s** 32ms/step - accuracy: 0.8504 - loss: 0.4955 - val\_accuracy: 0.9813 - val\_loss: 0.0625

Epoch 2/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **22s** 30ms/step - accuracy: 0.9808 - loss: 0.0596 - val\_accuracy: 0.9850 - val\_loss: 0.0498

Epoch 3/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **22s** 29ms/step - accuracy: 0.9883 - loss: 0.0387 - val\_accuracy: 0.9886 - val\_loss: 0.0409

Epoch 4/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **22s** 29ms/step - accuracy: 0.9907 - loss: 0.0316 - val\_accuracy: 0.9879 - val\_loss: 0.0407

Epoch 5/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **22s** 30ms/step - accuracy: 0.9924 - loss: 0.0233 - val\_accuracy: 0.9866 - val\_loss: 0.0443

Epoch 6/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **41s** 30ms/step - accuracy: 0.9933 - loss: 0.0200 - val\_accuracy: 0.9891 - val\_loss: 0.0409

Epoch 7/10

**750/750** ━━━━━━━━━━━━━━━━━━━━ **40s** 29ms/step - accuracy: 0.9946 - loss: 0.0151 - val\_accuracy: 0.9887 - val\_loss: 0.0408

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

print(f'Test accuracy: {test\_accuracy}')

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Accuracy over Epochs')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Loss over Epochs')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.figure(figsize=(10, 10))

plt.subplot(2, 1, 1)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Accuracy over Epochs')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.subplot(2, 1, 2)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Loss over Epochs')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

predictions = model.predict(x\_test)

predicted\_classes = tf.argmax(predictions, axis=1)

plt.figure(figsize=(10, 10))

for i in range(9):

    plt.subplot(3, 3, i + 1)

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

    plt.title(f'Predicted: {predicted\_classes[i].numpy()}, Actual: {y\_test[i].argmax()}')

    plt.axis('off')

plt.show()

Output



