# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras import datasets, layers, models

from tensorflow.keras.utils import to\_categorical

from sklearn.model\_selection import train\_test\_split

# Load the MNIST dataset

(train\_images, train\_labels), (test\_images, test\_labels) = datasets.mnist.load\_data()

# Preprocess the data

# Reshape images to 28x28x1 (grayscale)

train\_images = train\_images.reshape((train\_images.shape[0], 28, 28, 1))

test\_images = test\_images.reshape((test\_images.shape[0], 28, 28, 1))

# Normalize the pixel values to be between 0 and 1

train\_images, test\_images = train\_images / 255.0, test\_images / 255.0

# Convert labels to one-hot encoding

train\_labels = to\_categorical(train\_labels, 10)

test\_labels = to\_categorical(test\_labels, 10)

# Define the CNN model

model = models.Sequential([

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

layers.MaxPooling2D((2, 2)),

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

layers.MaxPooling2D((2, 2)),

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

layers.Flatten(),

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

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

])

# Compile the model

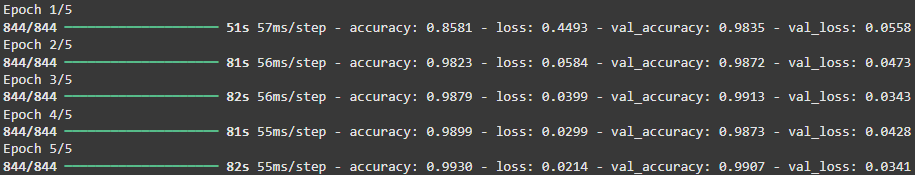
model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(train\_images, train\_labels, epochs=5, batch\_size=64, validation\_split=0.1)



# Evaluate the model on the test set

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print(f"Test accuracy: {test\_acc \* 100:.2f}%")



# Save the model (optional)

model.save('mnist\_cnn\_model.h5')

# Plot the training history (optional)

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

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

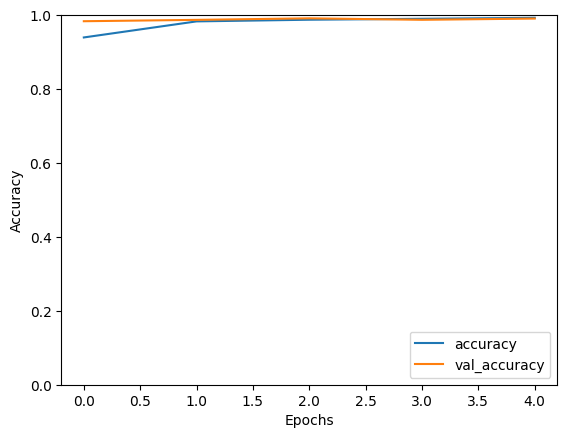
plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.ylim([0, 1])

plt.legend(loc='lower right')

plt.show()



# Predictions (optional)

predictions = model.predict(test\_images[:5])



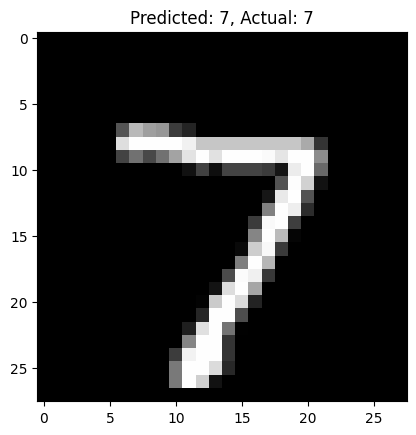
# Show the predicted digits and the actual digits

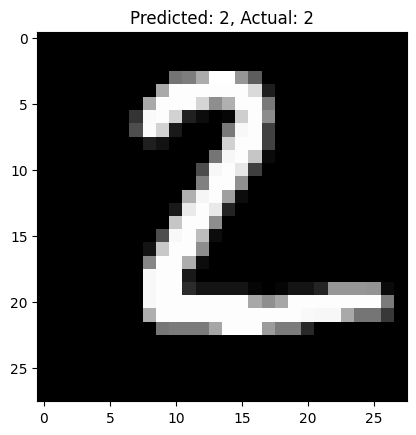
for i in range(5):

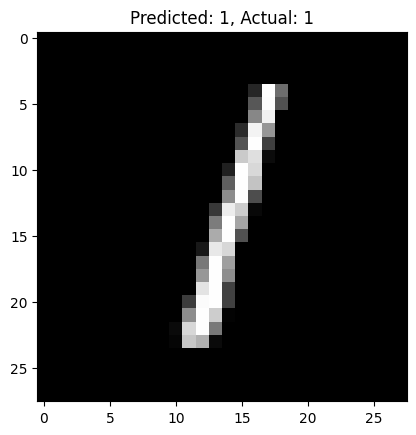
plt.imshow(test\_images[i].reshape(28, 28), cmap='gray')

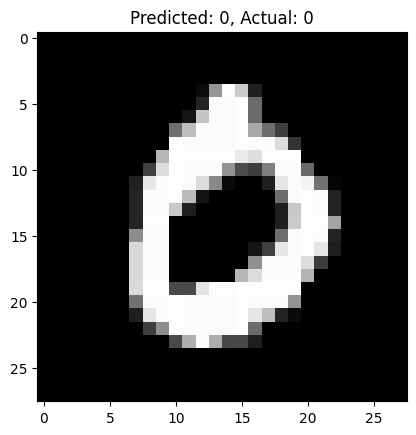
plt.title(f"Predicted: {np.argmax(predictions[i])}, Actual: {np.argmax(test\_labels[i])}")

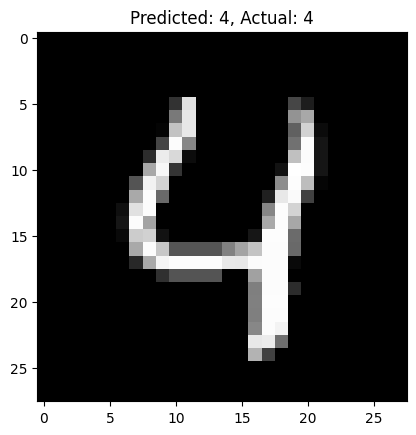
plt.show()











GOOGLE COLAB PROJECT LINK:

<https://colab.research.google.com/drive/1olnBH-v61GZBslZvkdY6HZ_1MjHDbGta>