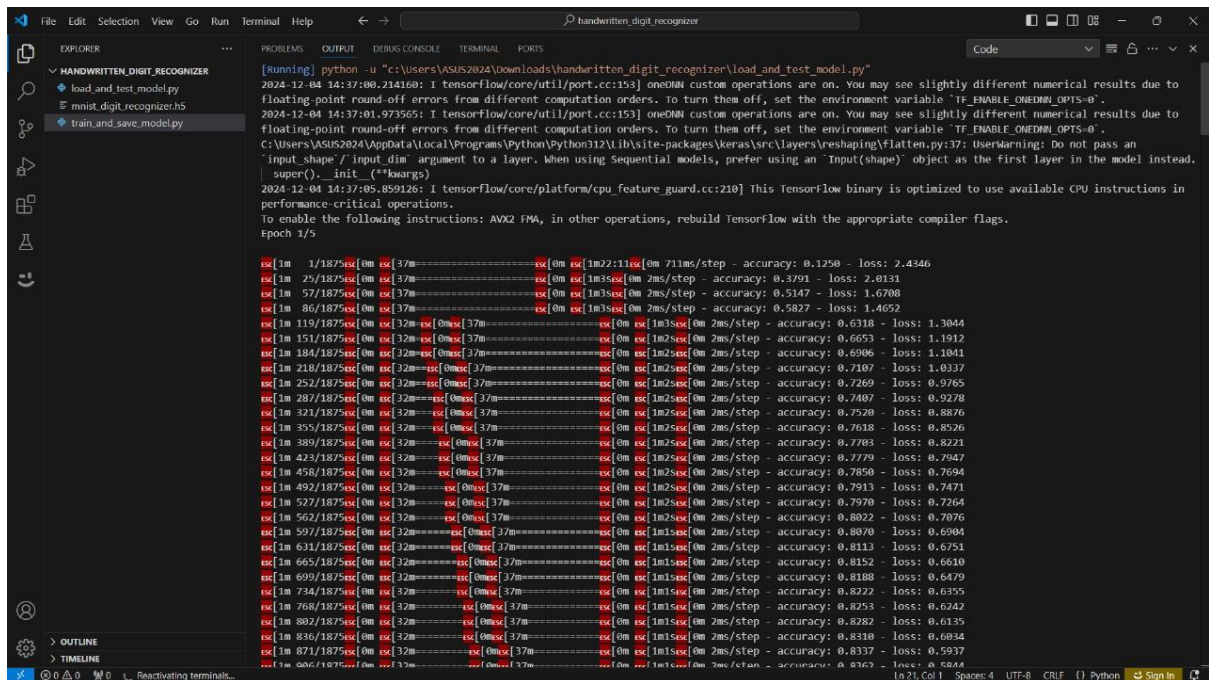
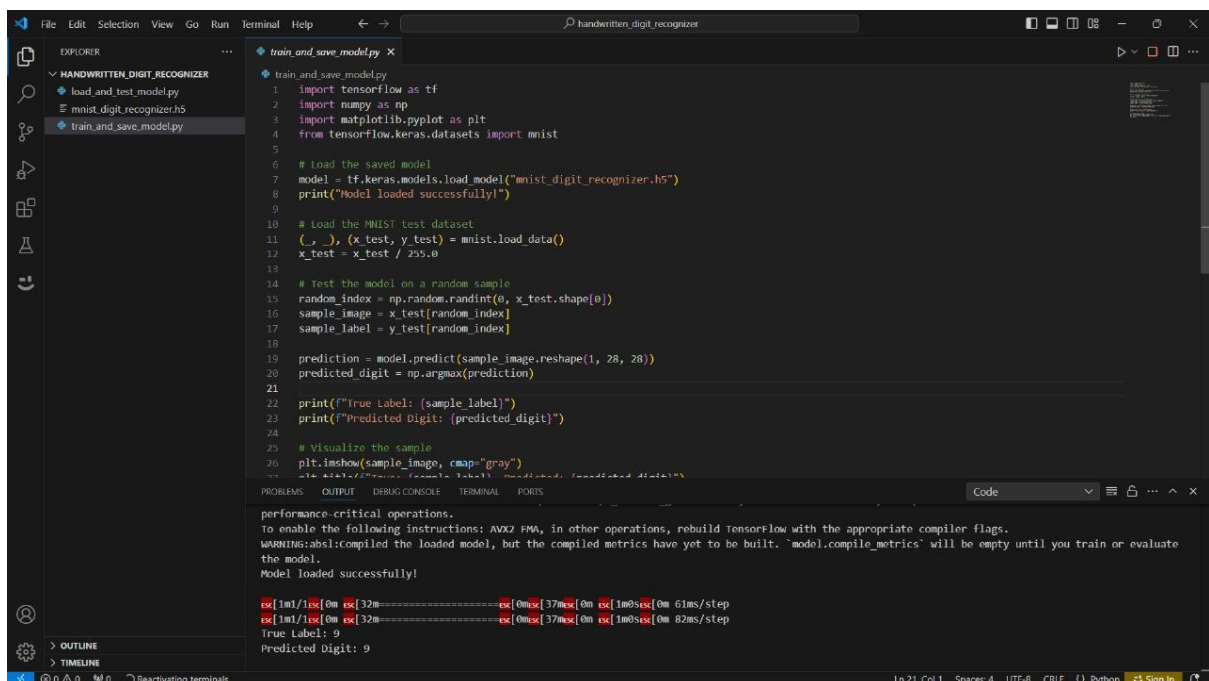


Output Screenshots



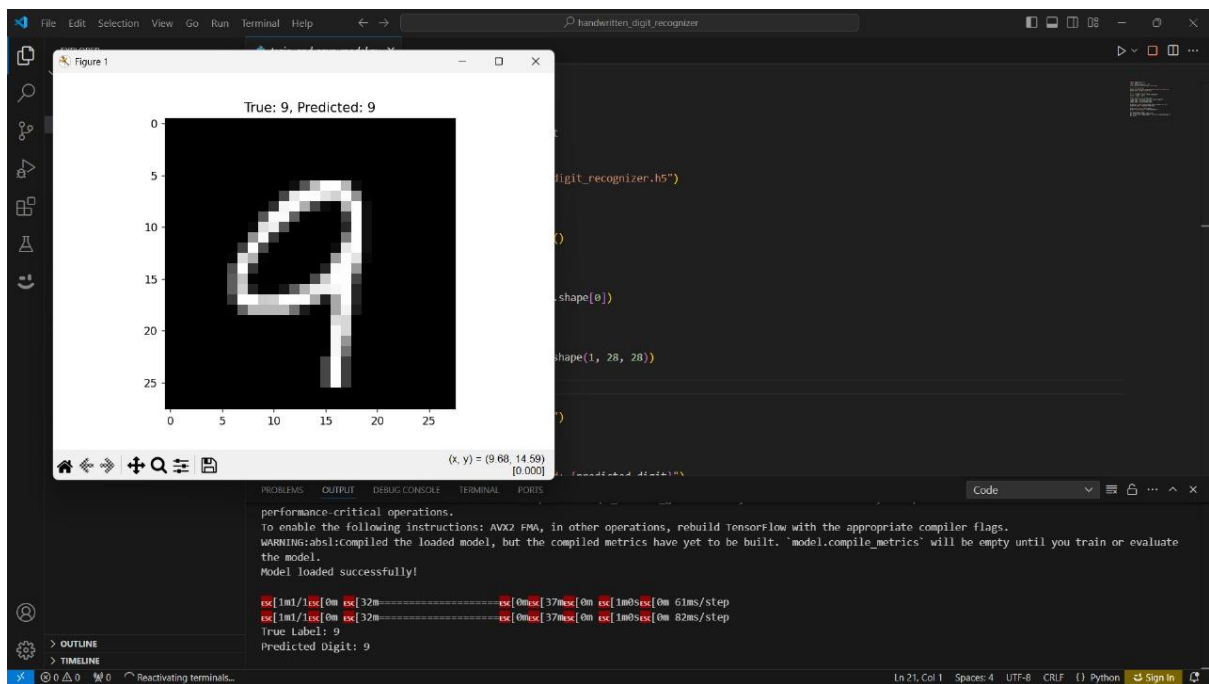
```
[running] python -u "c:\Users\VASUS2024\Downloads\handwritten_digit_recognizer\load_and_test_model.py"
2024-12-04 14:37:00.214160: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2024-12-04 14:37:01.977565: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
c:\Users\VASUS2024\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\srclayers\reshaping\flatten.py:37: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
  super().__init__(**kwargs)
2024-12-04 14:37:05.859126: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/5
es[1m 1/1875es[0m es[37m-----es[0m es[1m22:11es[0m 711ms/step - accuracy: 0.1250 - loss: 2.4346
es[1m 25/1875es[0m es[37m-----es[0m es[1m35es[0m 2ms/step - accuracy: 0.3791 - loss: 2.0131
es[1m 57/1875es[0m es[37m-----es[0m es[1m35es[0m 2ms/step - accuracy: 0.5147 - loss: 1.6708
es[1m 86/1875es[0m es[37m-----es[0m es[1m35es[0m 2ms/step - accuracy: 0.5827 - loss: 1.4652
es[1m 119/1875es[0m es[37m-----es[0m es[1m35es[0m 2ms/step - accuracy: 0.6318 - loss: 1.3044
es[1m 151/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.6653 - loss: 1.1912
es[1m 184/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.6906 - loss: 1.1041
es[1m 218/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7107 - loss: 1.0337
es[1m 252/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7269 - loss: 0.9765
es[1m 287/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7407 - loss: 0.9278
es[1m 321/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7520 - loss: 0.8876
es[1m 355/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7618 - loss: 0.8526
es[1m 389/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7703 - loss: 0.8221
es[1m 423/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7779 - loss: 0.7947
es[1m 458/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7850 - loss: 0.7694
es[1m 492/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7913 - loss: 0.7471
es[1m 527/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.7970 - loss: 0.7264
es[1m 562/1875es[0m es[37m-----es[0m es[1m25es[0m 2ms/step - accuracy: 0.8022 - loss: 0.7076
es[1m 597/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8070 - loss: 0.6904
es[1m 631/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8113 - loss: 0.6751
es[1m 665/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8152 - loss: 0.6610
es[1m 699/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8188 - loss: 0.6479
es[1m 734/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8222 - loss: 0.6355
es[1m 768/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8253 - loss: 0.6242
es[1m 802/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8282 - loss: 0.6135
es[1m 836/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8310 - loss: 0.6034
es[1m 871/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8337 - loss: 0.5937
es[1m 906/1875es[0m es[37m-----es[0m es[1m15es[0m 2ms/step - accuracy: 0.8363 - loss: 0.5844
```

Loading Images



```
train_and_save_model.py X
train_and_save_model.py
1 import tensorflow as tf
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from tensorflow.keras.datasets import mnist
5
6 # Load the saved model
7 model = tf.keras.models.load_model("mnist_digit_recognizer.h5")
8 print("Model loaded successfully!")
9
10 # Load the MNIST test dataset
11 (_, _), (x_test, y_test) = mnist.load_data()
12 x_test = x_test / 255.0
13
14 # Test the model on a random sample
15 random_index = np.random.randint(0, x_test.shape[0])
16 sample_image = x_test[random_index]
17 sample_label = y_test[random_index]
18
19 prediction = model.predict(sample_image.reshape(1, 28, 28))
20 predicted_digit = np.argmax(prediction)
21
22 print(f"True Label: {sample_label}")
23 print(f"Predicted Digit: {predicted_digit}")
24
25 # Visualize the sample
26 plt.imshow(sample_image, cmap="gray")
27 plt.show()
28
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
Model loaded successfully!
es[1m1/es[0m es[32m-----es[0m es[37mes[0m es[1m05es[0m 61ms/step
es[1m1/es[0m es[32m-----es[0m es[37mes[0m es[1m05es[0m 82ms/step
True Label: 9
Predicted Digit: 9
```

Training Data



Predicted Value