Introduction to Artificial Intelligence Report

**Student ID: 2020600060**

**Major Class: COMPUTER SCIENCE AND TECHNOLOGY (AI)**

**Student name: CHIRWA ALEX JOSHUA**

**Advisor : Huang Qing**

**Score:**

1. Purpose and Task

The purpose of the introduction to Artificial Intelligence assignment is to train students how to combine theory and programming implementation on the basis of learning basic theories to realize the engineering application of artificial intelligence algorithms in practical scenarios. It allows students to have a deeper understanding of the basic concepts, methods and applications in the field of artificial intelligence, as well as master related tools and technologies.

Ii. Requirements

1. Systematically learn and master the basic concepts, principles and methods of artificial intelligence, including search, knowledge representation, reasoning, planning, machine learning, natural language processing, computer vision, etc.
2. Be familiar with and master commonly used AI tools and techniques, such as the Python programming language, PyTorch and other deep learning frameworks.
3. Understand and analyze different types of AI applications, such as speech recognition, image recognition, natural language processing, intelligent recommendation, autonomous driving, etc., and deeply discuss their principles, implementation methods and practical application scenarios.
4. To study and discuss the cutting-edge technologies and latest research results in the field of artificial intelligence, and understand the development trend and application prospects of artificial intelligence.
5. Complete an AI-related project or experiment, apply the learned knowledge and technology to solve practical problems, and improve practical operation ability and application skills

Iii. Content

You can choose one of the following three research contents:

**Application 1:** Given fundus blood vessel data set (which has been marked), the artificial intelligence algorithm taught in class or learned by ourselves is used to achieve segmentation of fundus blood vessel data set. Accuracy,Precision and Recall of the results of the data set should be evaluated.

**Application 2:** Neural network to realize handwriting recognition. Download MNIST's dataset and, using a deep learning network, train a handwritten digit recognition model. Compare the differences between two or three deep learning networks.

**Application 3:** Study the latest literature and explore the technology of autonomous driving. If you are going to make an autonomous driving system, what modules need to be included and what technology needs to be used to implement each module.

Iv. Summary and analysis

**Reporting requirements:**

**1. Explain the reasons for selecting the application.**

**2. Explain the rationale of the chosen method and the basic steps to implement it.**

**3. Explain the specific operation of the implementation.**

**4. Discuss the difficulties encountered and how they were solved.**

**5. Outlook: It can be the next step to improve or the next step to do or anything you want to say. The report needs charts.**

**Code requirements: Each statement clearly states what it is doing and why. The statement is clean and concise.**

**Neural network (Application 2)**

**Overview:**

*In this part of the introduction to the chosen application, I would like to give my opinions/reasons to why I chose the particular project.*

From my perspective, selecting the particular project “handwritten digit recognition” in Artificial Intelligence education it can offer a valuable learning experience.

Here are some of the reasons for selecting handwritten digit recognition:

* **Interest in Computer Vision –** based on how computers read and recognize characters that I have always wanted to know how it truly works, by building a digit recognition model allows me to seek deep knowledge into this exciting field.
* **Practical Application –** Handwritten digit recognition has practical applications in various domains, from automating data entry to enhancing OCR systems. I would love to work on something with real-world impact.
* **Learning Deep Learning Techniques –** I am eager to learn and apply deep learning techniques. This project provides a hands-on opportunity to work with neural networks, a powerful tool in AI.
* **Dataset Availability –** The MNIST dataset is a well-known benchmark for handwritten digit recognition. It is easily accessible and provides a solid foundation for training and testing my models.
* **Challenging and Rewarding –** I wanted a project that would challenge me intellectually and allow me to see tangible results. Building a model to recognize handwritten digits seems like a complex yet achievable goal.

**Steps implemented**

**Step 1: Setting Up Environment**

* We will have to first identify the programming language for this task and that will be Python.
* Installing the necessary libraries namely TensorFlow, Numpy, Matplotlib.pyplot, Pandas and OpenCV.

**Step 2: Download MNIST Dataset**

1. Downloading the MNSIT dataset will allow as to train and test model in machine learning

**Step 3: Load and Explore the Data**

1. Here we load the dataset and explore it. Understand its structure, the number of samples, and the format of the data. Familiarize yourself with the Images and labels.

**Step 4: Preprocess the Data**

1. Preprocessing is crucial. Normalize the images and encode the labels. This helps in better training of your neural network.

**Step 5: Build Neural Networks**

1. You can start with a simple feedforward neural network. Then, try a Convolutional Neural Network (CNN), a type of deep learning network known for image-related tasks. Compare their performances.

**Step 6: Train and Evaluate**

1. Train each model using the training data and evaluate their performance using the test data. Look at metrics like accuracy, precision, and recall.

**Step 7: Compare Results**

1. Compare the performance metrics of each model. Pay attention to things like training time, accuracy, and how well they generalize to new, unseen data.

**Step 8: Fine-tune and Experiment**

1. Experiment with different architectures, hyper parameters, and optimization techniques. Fine-tune the models to achieve better results.

**Codes and results**

The codes below are broken down into two parts the first set of codes set up a neural network using **TensorFlow** and **Keras** to perform handwritten digit recognition using the MNIST dataset. The second and final set of codes include image predictions using the trained model for four handwritten digit images (‘1.png’, ‘2.png’, ‘3.png’, ‘4.png’).

**Here is the first part of the project:**

import pandas as pd  
import tensorflow as tf  
import numpy as np  
import cv2 as cv  
import matplotlib.pyplot as plt  
from tensorflow.python.keras import activations  
from tensorflow.python.keras.activations import softmax  
data=tf.keras.datasets.mnist  
from tensorflow.python.keras.metrics import accuracy  
from tensorflow.python.keras.layers import Dense  
  
(x\_train,y\_train),(x\_test,y\_test)=data.load\_data()  
  
x\_train=tf.keras.utils.normalize(x\_train,axis=1)  
x\_test=tf.keras.utils.normalize(x\_test,axis=1)  
model=tf.keras.models.Sequential()  
model.add(tf.keras.layers.Flatten(input\_shape=(28,28)))  
model.add(Dense(units=128,activation='relu'))  
model.add(Dense(units=128,activation='relu'))  
model.add(Dense(units=10,activation='softmax'))  
model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])  
model.fit(x\_train,y\_train,epochs=3)  
  
loss,accuracy=model.evaluate(x\_test,y\_test)  
print(accuracy)  
print(loss)

Here is a breakdown of the code:

1. **Importing Libraries:**

I imported the necessary libraries including Pandas for data handling, TensorFlow for machine learning, NumPy for numerical operations, OpenCV for computer vision tasks, and Matplotlib for data visualization.

1. **Loading the MNIST Dataset:**

I Loaded the MNIST dataset using TensorFlow’s built-in datasets. This dataset contains images of handwritten digits along with their corresponding labels.

1. **Data Preprocessing:**

I normalized the pixel values of the images to rage between 0 and 1. This helps improve the training process.

1. **Model Construction:**

I created a sequential model, which is a linear stack of layers. The model consists of:

* A Flatten layer to convert the 2D images into a 1D array.
* Two Dense layers with 128 units and ReLu activation functions.
* Another Dense layer with 10 units (for 10 digits) and a softmax activation function, which outputs probabilities for each class.

1. **Compiling the Model:**

I compiled the model using the Adam optimizer and sparse categorical cross-entropy as the loss function. The metric ‘accuracy’ is used to monitor the performance during training.

1. **Training the Model:**

I trained the model using the training data (‘x\_train’ and ‘y\_train’) for 3 epochs.

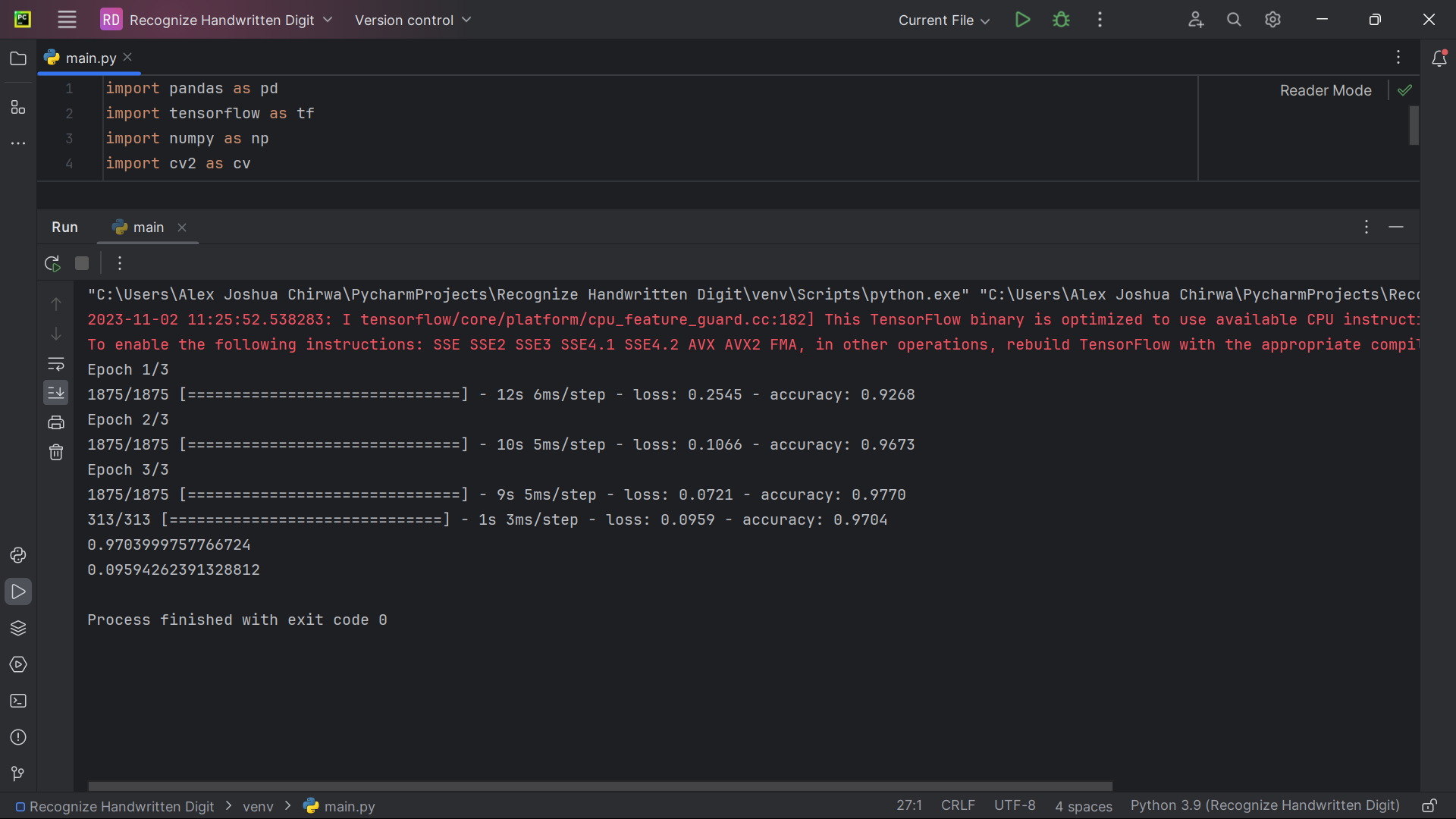
1. **Evaluating the Model:**

I evaluated the model’s performance using the test data (‘x\_test’ and ‘y\_test’) and print out the accuracy and loss.

1. **Printing Results:**

Finally, I printed out the accuracy and loss obtained on the test set.

**Result:**



**Here is the Second part of the project:**

import pandas as pd  
import tensorflow as tf  
import numpy as np  
import cv2 as cv  
import matplotlib.pyplot as plt  
from tensorflow.python.keras import activations  
from tensorflow.python.keras.activations import softmax  
data=tf.keras.datasets.mnist  
from tensorflow.python.keras.metrics import accuracy  
from tensorflow.python.keras.layers import Dense  
  
(x\_train,y\_train),(x\_test,y\_test)=data.load\_data()  
  
x\_train=tf.keras.utils.normalize(x\_train,axis=1)  
x\_test=tf.keras.utils.normalize(x\_test,axis=1)  
model=tf.keras.models.Sequential()  
model.add(tf.keras.layers.Flatten(input\_shape=(28,28)))  
model.add(Dense(units=128,activation='relu'))  
model.add(Dense(units=128,activation='relu'))  
model.add(Dense(units=10,activation='softmax'))  
model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])  
model.fit(x\_train,y\_train,epochs=3)  
  
loss,accuracy=model.evaluate(x\_test,y\_test)  
print(accuracy)  
print(loss)  
  
for x in range(1,5):  
 img=cv.imread(f'C:/Users/Alex Joshua Chirwa/PycharmProjects/Recognize Handwritten Digit/Data/{x}.png', cv.IMREAD\_GRAYSCALE)  
 img=cv.resize(img,(28,28))  
 img=np.invert(np.array([img]))  
 img=img/255.0  
 prediction=model.predict(img)  
 print("----------------")  
 print("the predicted output is: ",np.argmax(prediction))  
 print("----------------")  
 plt.imshow(img[0],cmap=plt.cm.binary)  
 plt.show()

Here is a breakdown of the code:

1. **Reading and Preprocessing images:**

I used OpenCV (‘cv2’) to read the images and convert them to grayscale. Then, I resized the images to match the input shape of my model (28x28 pixels). Finally, I inverted the pixel values and normalize them to be between 0 and 1.

1. **Making Predictions:**

I passed the preprocessed image through your trained model using ‘model.predict()’ to get the prediction probabilities for each digit (0-9).

1. **Displaying Results:**

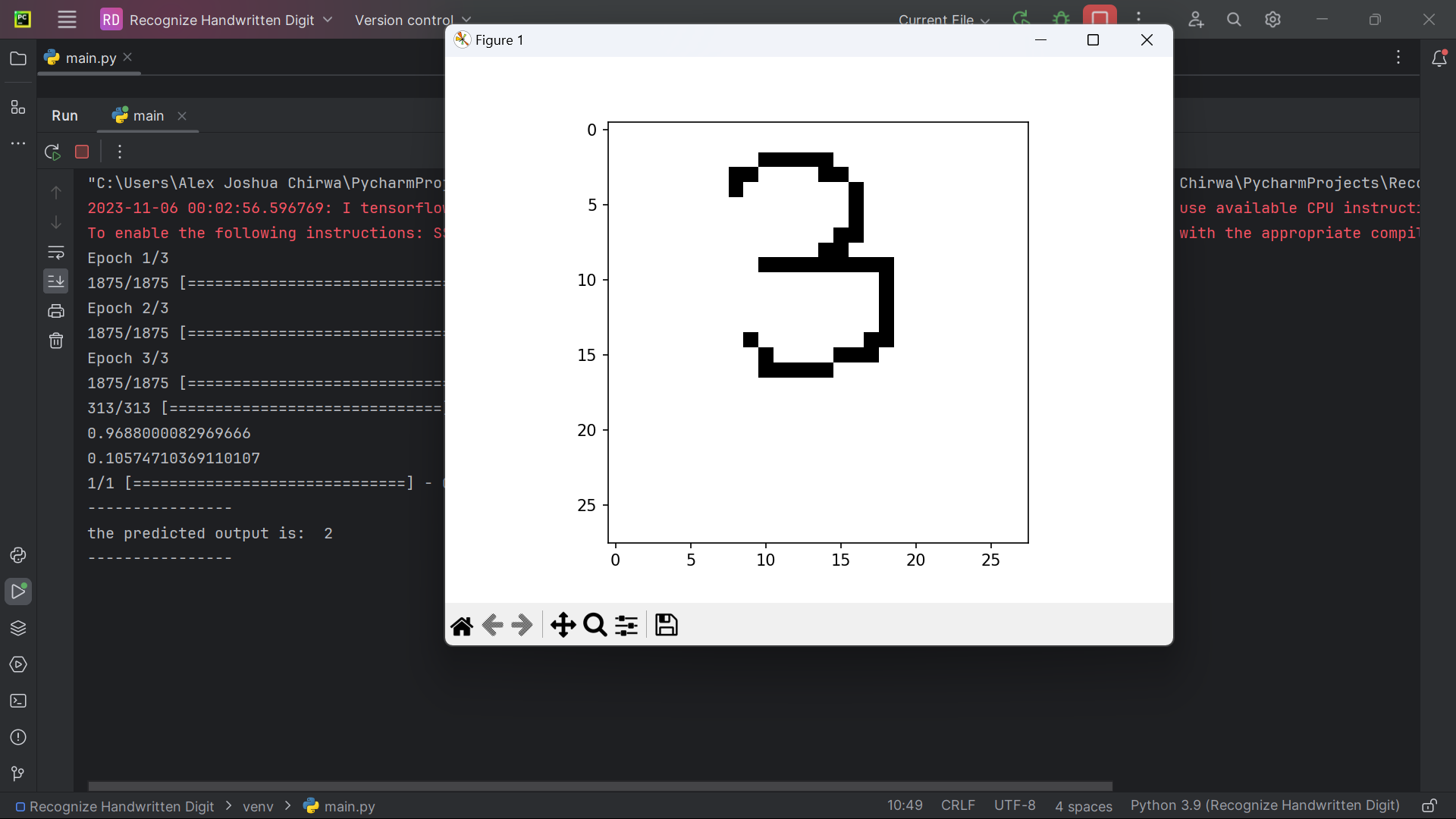
I printed out the predicted digit using ‘np.argmax(prediction)’. This gives you the digit using the highest probability according to the model’s predications.

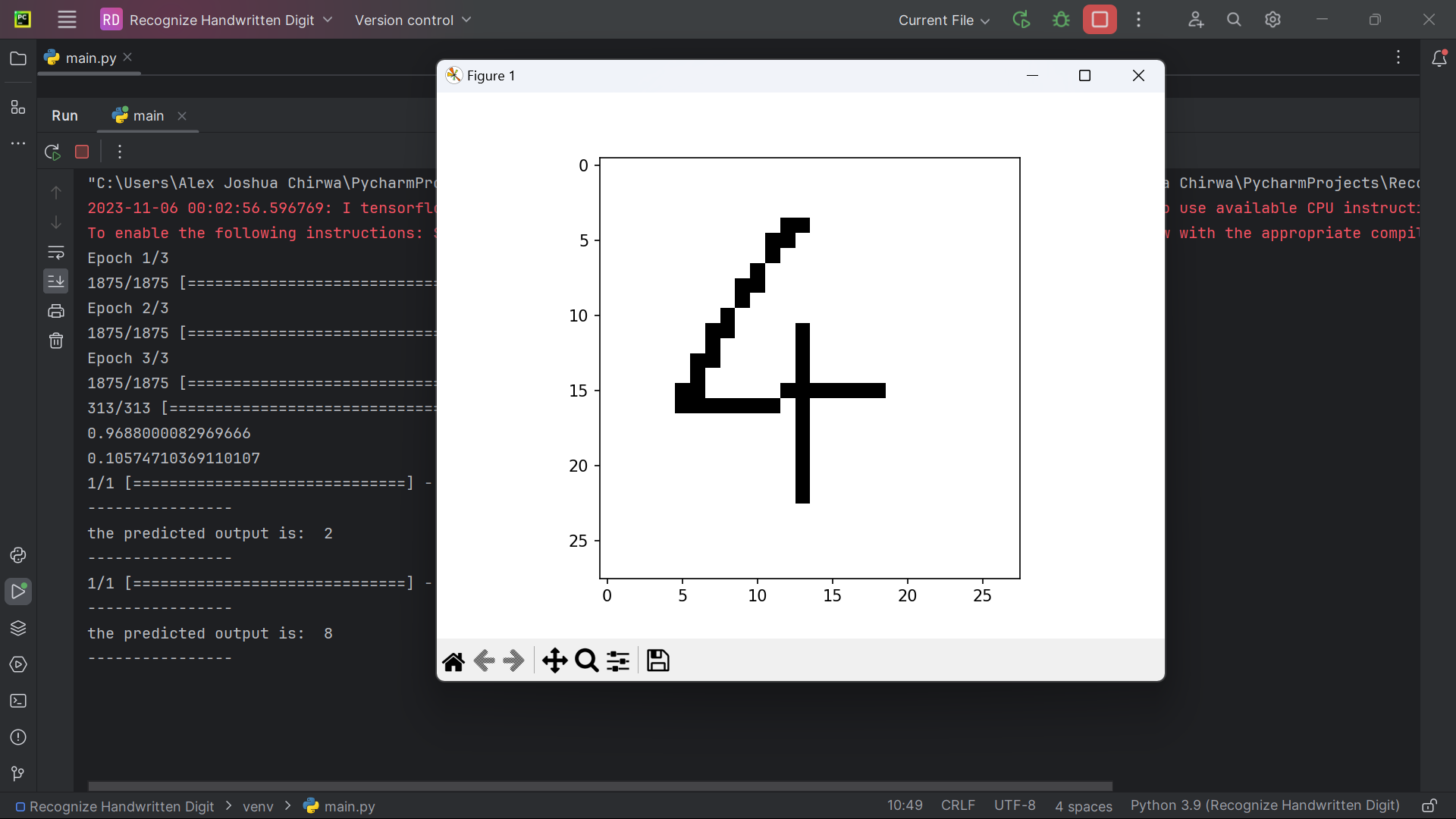
1. **Displaying images:**

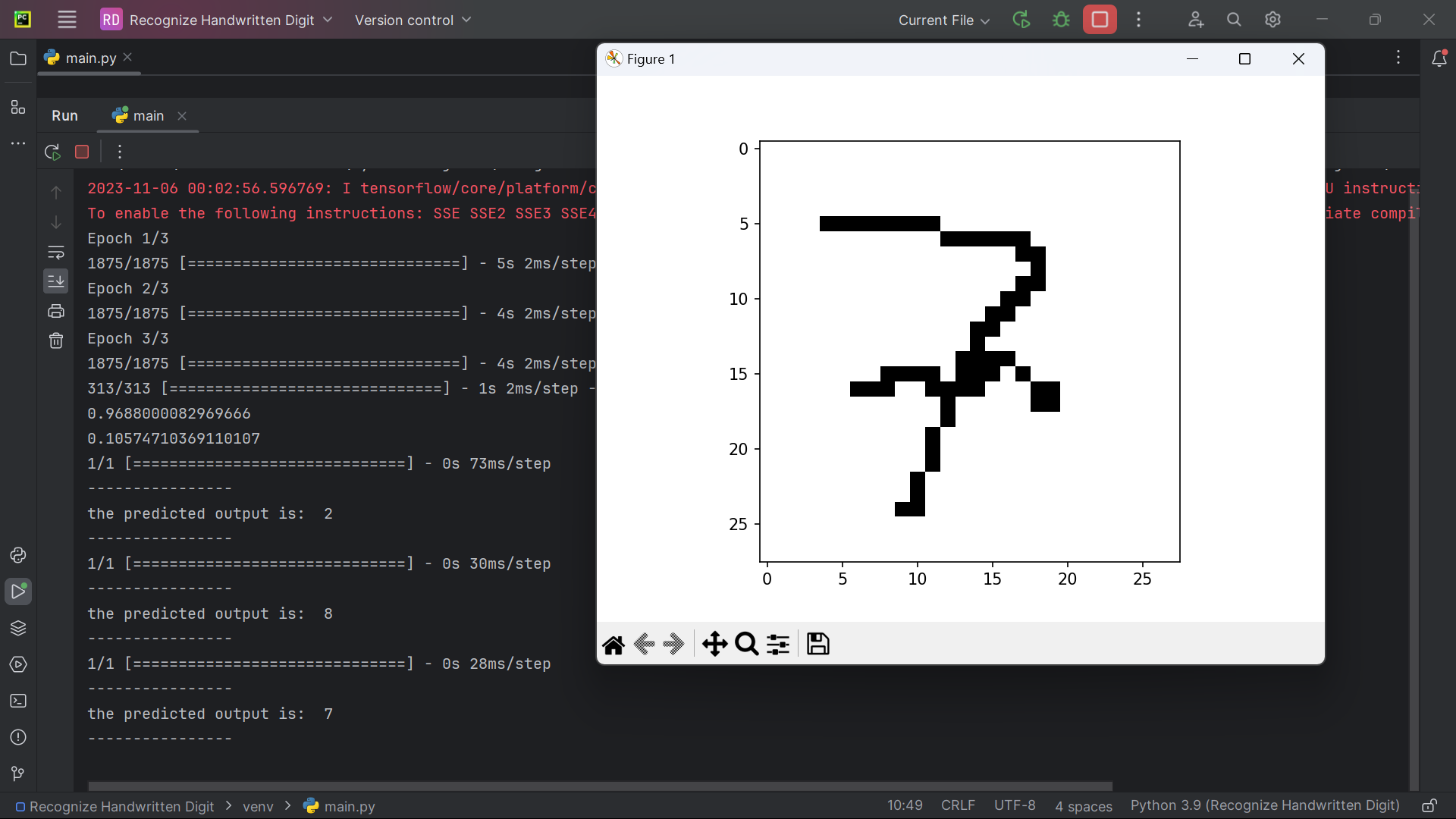
I used Matplotlib to display the processed images along with their predictions.

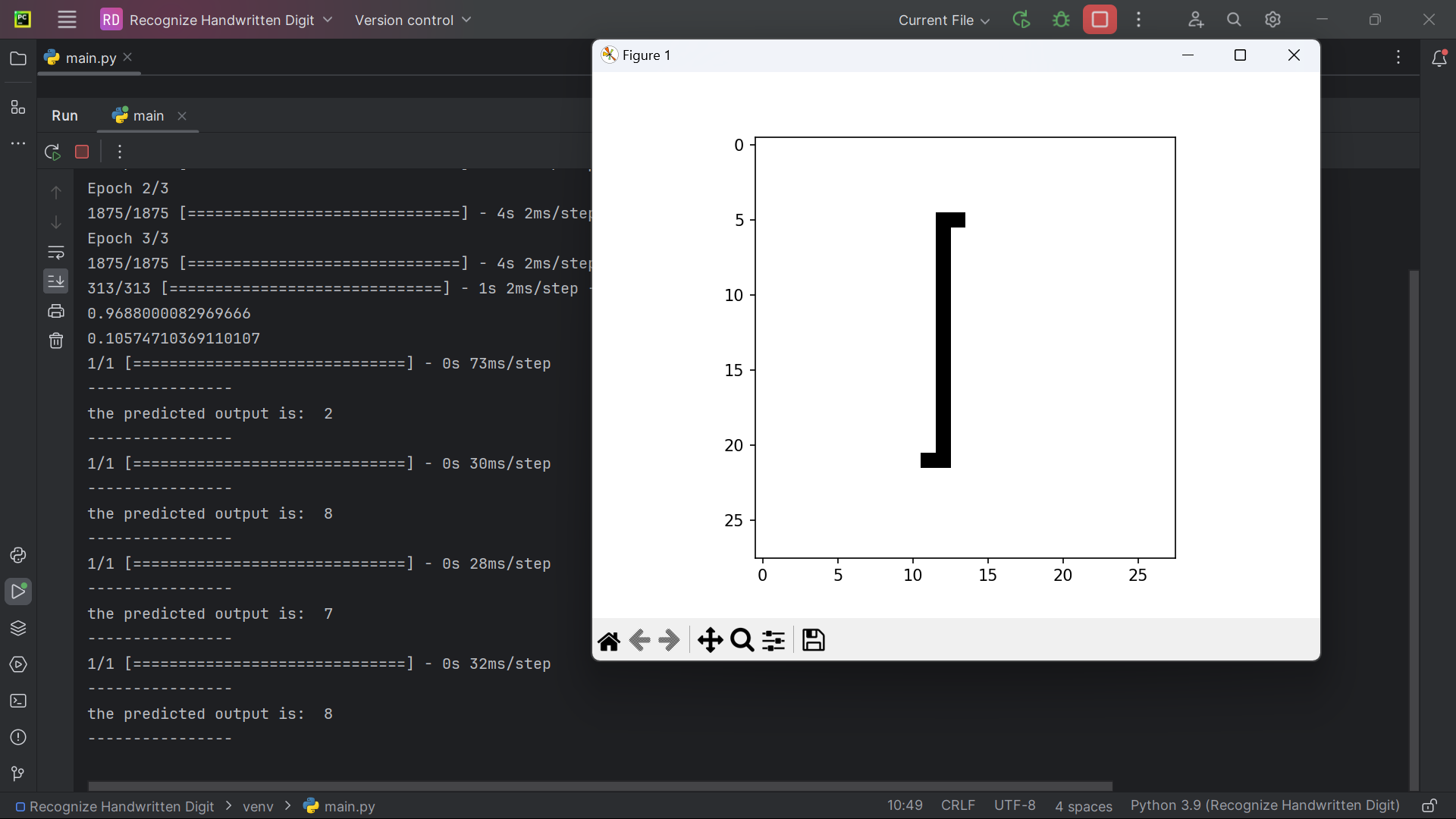
**Result:**

These below are the first sets of results and from the looks of things; the model gets only the prediction of the number 7 correct and the rest wrong.





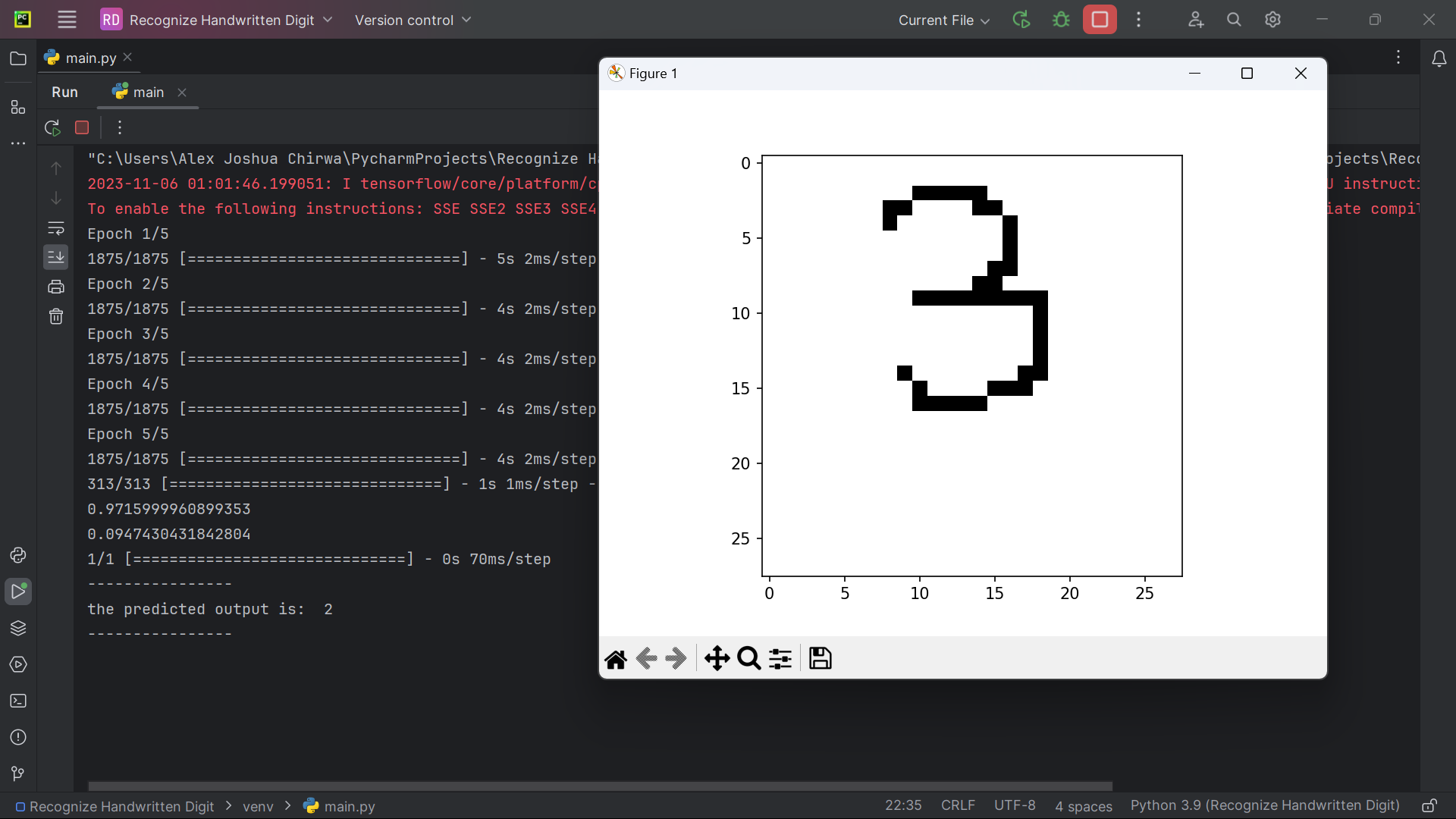




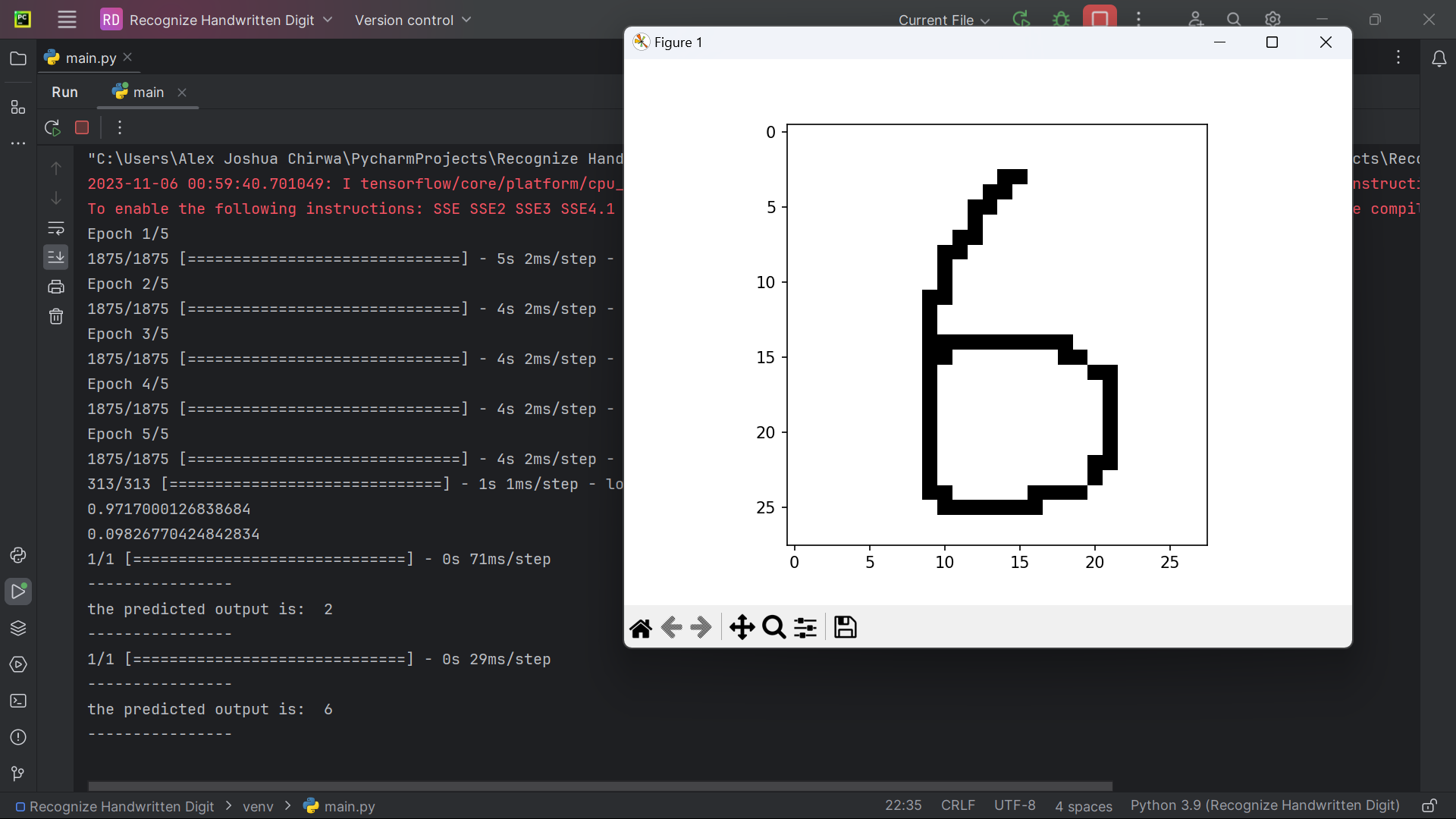
**Second run on the model:**

In this second trail we see an improvement where we have 3 out of 4 sets correct which is impressive, note that when I increased the Epoch to 5 it allowed the model to see and learn from the entire training dataset. This means that it had an opportunity to understand a wide range of patterns and relationships present in the data.

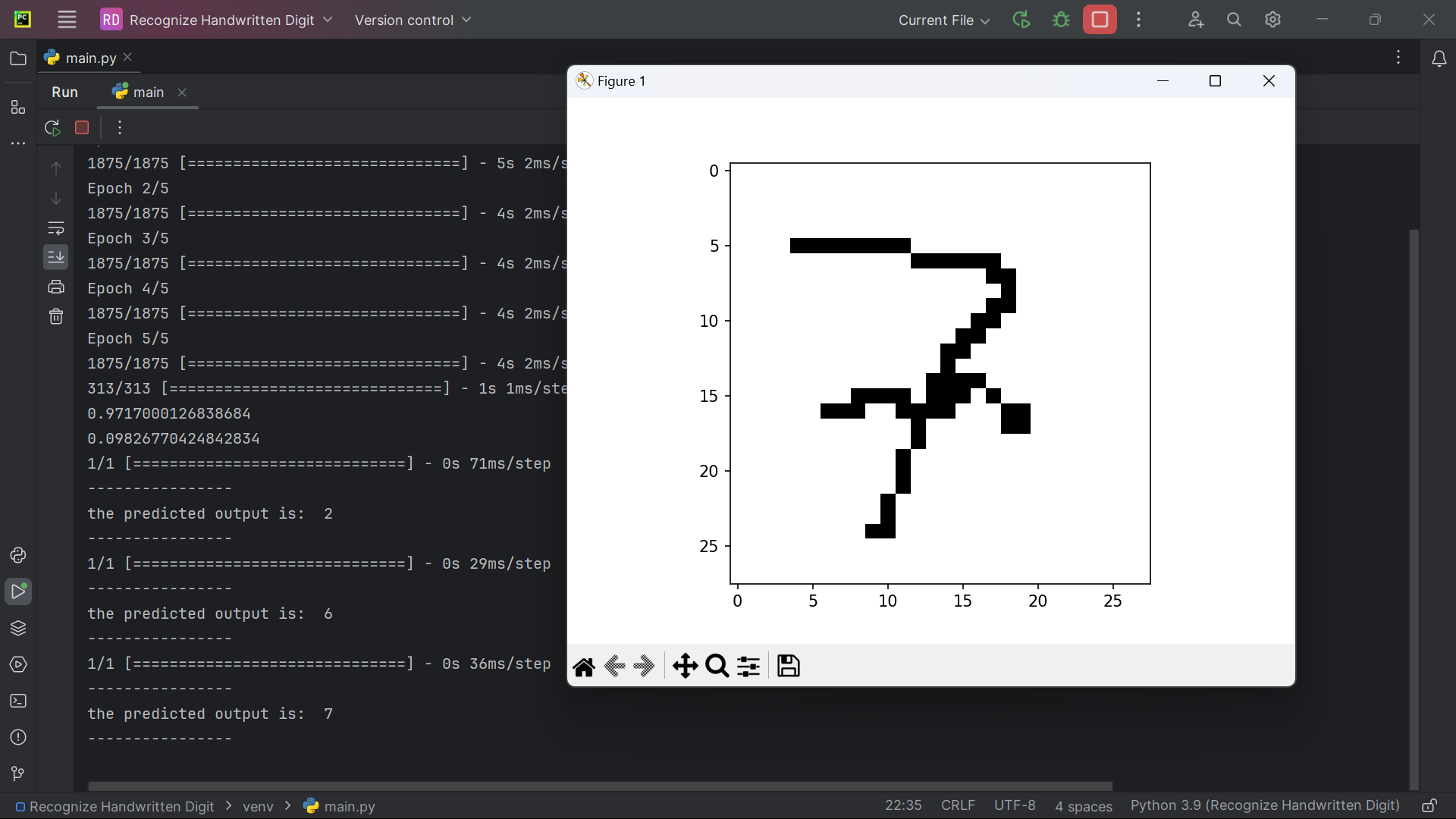
**Results:**



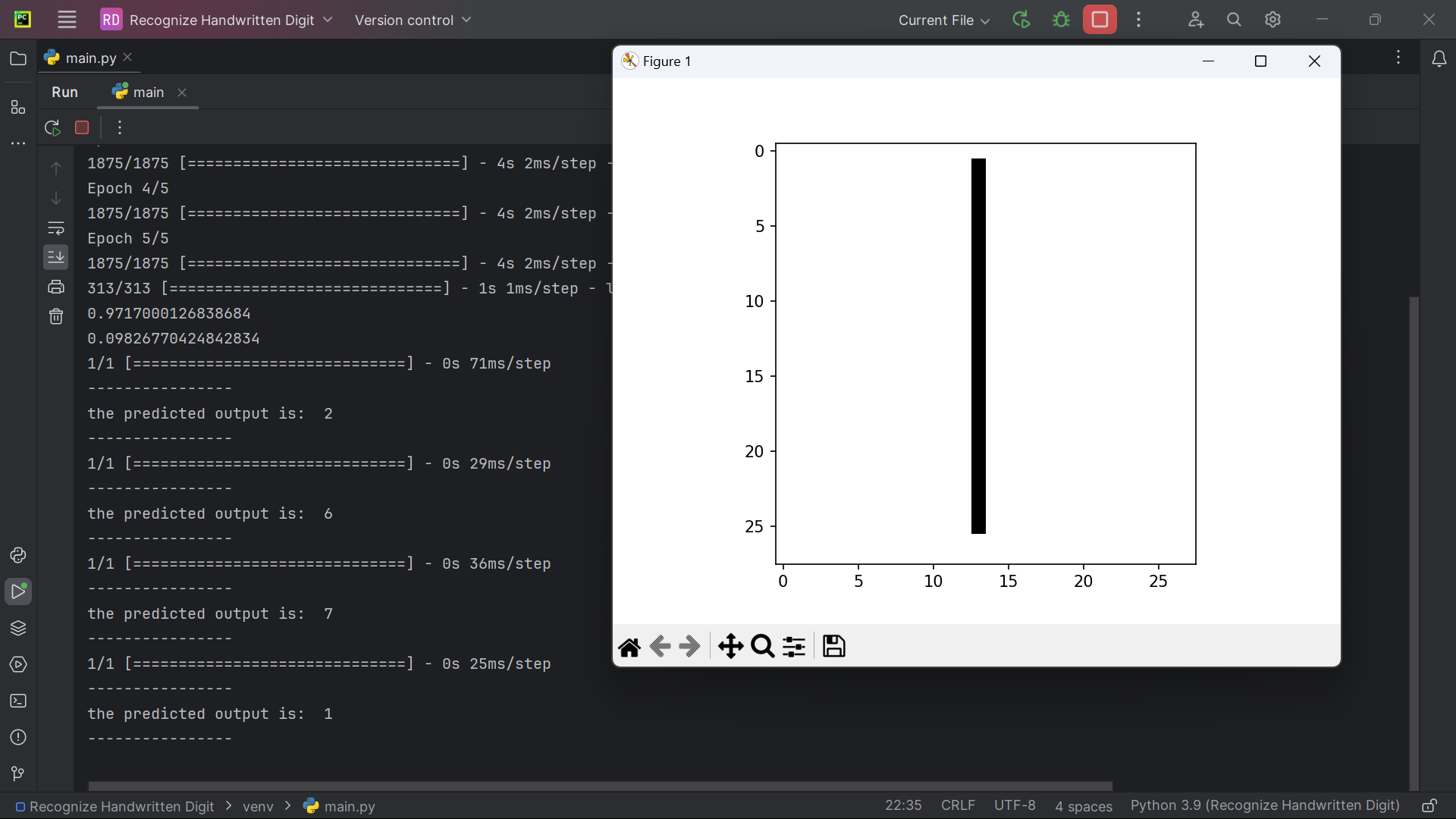
the predicted output is: 2 (**X)**



the predicted output is: 6 (✔️)



the predicted output is: 7 (✔️)



the predicted output is: 1 (✔️)

Process finished with exit code 0

**Conclusion:**

Overall, I have successfully built and trained a neural network for handwritten digit recognition using the MNIST dataset. Additionally, I demonstrated how to make predictions on new images, which is an important step in assessing the model’s real-world applicability.