Martina Gichenko and Neda Markoska

mg28561@seeu.edu.mk nm28679@seeu.edu.mk

Handwritten digit recognition

Final Project for the course Introduction to Artificial Intelligence

Introduction………………………………………………………………………………….2

Step-by-Step Process………………………………………………………………………...2

Results……………………………………………………………………………………….4

**Introduction**

Our project handwritten digit recognition is built using Neural Networks and TensorFlow in Python.

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

**Step-by-Step Process**

We have imported 4 essential libraries:

* cv2 – open cv library (which we need to import our own images). We first install it in CommandPromt with the command *pip install opencv-python*
* numpy
* matplotlib.pyplot – for some visualizations
* tensorflow – in order to build the neural network to get the data and also to test it, to apply the neural network

1. The first step we did was load the data set of the handwritten digits and we have the MNIST(Modified National Institute of Standards and Technology database) data set, which is a large database of handwritten digits that is commonly used for training various image processing systems.

mnist = tf.keras.datasets.mnist

1. Then we split the data set into training and testing data

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

1. The next step we did was normalize the data – we scaled it down between 0 and 1. We did this to make it easier to be computed.

x\_train = tf.keras.utils.normalize(x\_train, axis=1)

x\_test = tf.keras.utils.normalize(x\_test, axis=1)

1. The next step we define the model. We have taken an input layer, 2 dense layers in between(2 hidden layers) and one output. Which basically is a feedforward neural network.

* A feedforward neural network (FNN) is an artificial neural network wherein connections between the nodes do not form a cycle. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.
* The flatten layer is just a one dimensional layer, because we have 28x28 pixels in all of our images, so this would be more of a grid. And we flatten out the layers so that we have 784 pixels
* Dense layer means that all the neurons are connected to the previous layer and the next layer

model = tf.keras.models.Sequential() #an ordinary feedforward neural network  
  
#build the model  
  
#add some layers  
model.add(tf.keras.layers.Flatten(input\_shape=(28,28))) #flatten the layer with all the pixels off each individual image of a handwritten digit and we feed that into the input layer and then this input layer is followed by a dense layer  
#add the dense layer  
model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu)) #number of neurons that we are gonna have in this layer. The activation function is a simple rectify linear unit function  
#add the second hidden layer  
model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))  
#output layer  
model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax)) #the activation function is the function that tries to take all the outputs

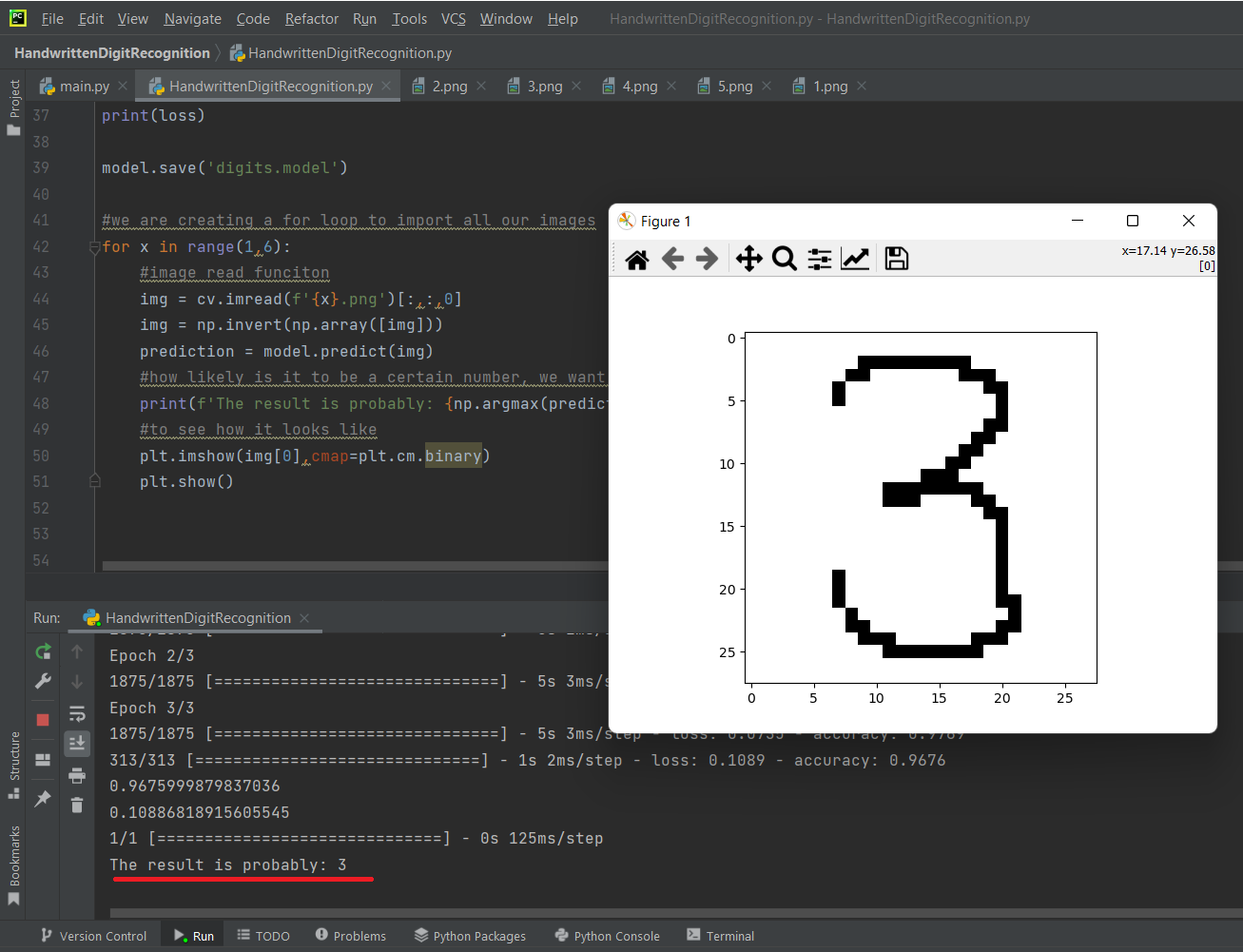
1. And finally in the fifth step we fit the model by training the neural network

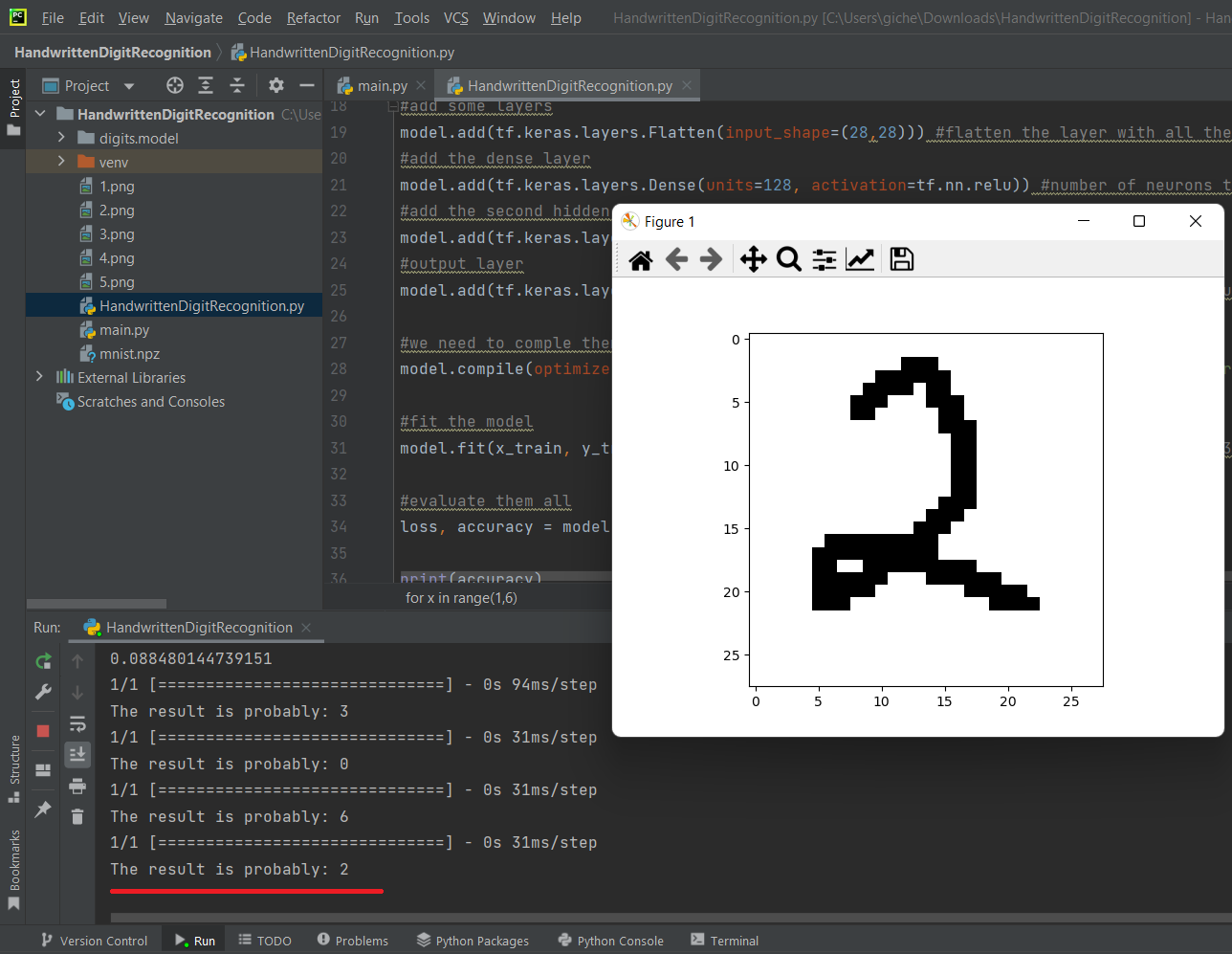
model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  
model.fit(x\_train, y\_train, epochs=3)   
loss, accuracy = model.evaluate(x\_test, y\_test)

1. We uploaded 5 images – digits written in paint: 3,0,8,2,4

for x in range(1,6):   
 img = cv.imread(f'{x}.png')[:,:,0]  
 img = np.invert(np.array([img]))  
 prediction = model.predict(img)  
 print(f'The result is probably: {np.argmax(prediction)}')   
 plt.imshow(img[0],cmap=plt.cm.binary)  
 plt.show()

**Results**

****

Graphical user interface

Description automatically generated