Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №2

по дисциплине «Программные средства кибертехнических систем»

на тему «Работа с нейронной сетью»

Выполнили:

студенты группы 22ВВП1

Демин М. С.

Беляев Д. И.

Захаров А. С.

Приняли:

Дубинин В.Н.

Карамышева Н.С.

Пенза 2025

**Название**

Работа с нейронной сетью

**Цель работы**

Изучение программы по распознаванию рукописных цифр при помощи трехслойной нейронной сети.

**Задание**

Изучить теоретическую информацию и программу по распознаванию рукописных цифр на языке Python. Запустить программу

в среде Spyder(Anaconda), проанализировать полученные результаты. Затем внести правки в код в соответствии с вариантом и сравнить результаты с полученными ранее.

Опросить сеть с использованием тестовой базы данных (собственноручно написанных цифр).

**Варианты лабораторных заданий**

В соответствии с таблицей задаются коэффициент обучения,

количество эпох, количество скрытых узлов:

коэф = 0,3

кол-во эпох = 5

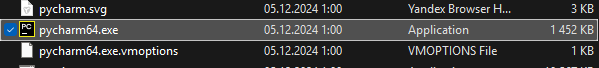
кол-во скрытых узлов = 140

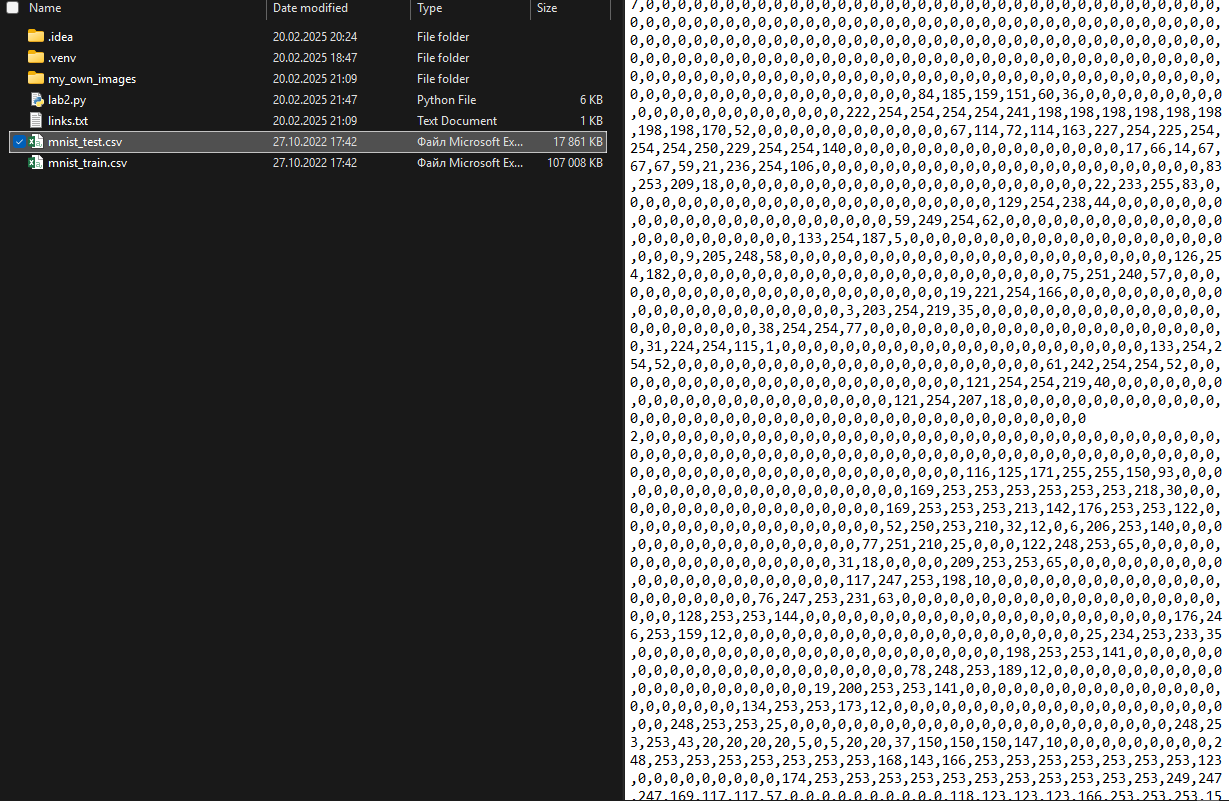
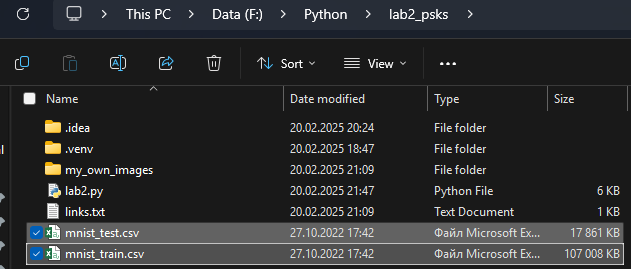
**Ход работы:**

1. Установливаем Python с официально сайта https://www.python.org/getit/, версию Python 3.12.

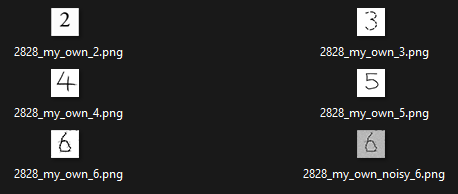


1. Затем с официального сайта установим среду PyCharm Community Edition 2024.2.4



1. Скачаем файлы тестового и тренировочного набора с сайта <https://pjreddie.com/projects/mnist-in-csv/>

Это наш тестовый и тренировочный наборы данных. Тренировочный набор содержит 60 000 промаркированных экземпляров, используемых для тренировки нейронной сети. Слово «промаркированные» означает, что для каждого экземпляра указан соответствующий правильный ответ. Он содержит маркер цифры в начале строки и затем значения 784 пикселей (размерность картинки 28×28).

1. Создали тестовую базу данных, используя собственноручно написанные цифры. Сымитировали различные виды почерка, а также зашумленные и прерывистые изображения, чтобы посмотреть, как с ними справится нейронная сеть. Берём изображения формата .png с соотношением сторон 1:1 размером 28х28 пикселей. 
2. Сделали программу. Модифицировали программную реализацию нейронной сети.

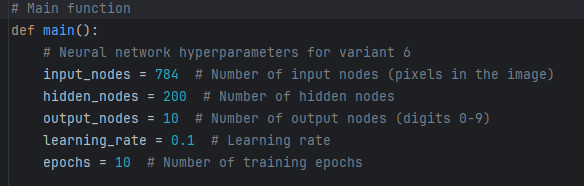
**Листинг**

import numpy as np  
import matplotlib.pyplot as plt  
import scipy.special  
import glob  
import imageio.v3 as imageio  
  
  
# Define the Neural Network class  
class NeuralNetwork:  
 def \_\_init\_\_(self, input\_nodes, hidden\_nodes, output\_nodes, learning\_rate):  
 self.inodes = input\_nodes  
 self.hnodes = hidden\_nodes  
 self.onodes = output\_nodes  
  
 # Initialize weight matrices with random values  
 self.wih = np.random.normal(0.0, pow(self.hnodes, -0.5), (self.hnodes, self.inodes))  
 self.who = np.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes))  
  
 self.lr = learning\_rate  
 self.activation\_function = lambda x: scipy.special.expit(x)  
  
  
 def train(self, inputs\_list, targets\_list):  
 # Convert input and target lists to 2D arrays  
 inputs = np.array(inputs\_list, ndmin=2).T  
 targets = np.array(targets\_list, ndmin=2).T  
  
 # Forward pass  
 hidden\_inputs = np.dot(self.wih, inputs)  
 hidden\_outputs = self.activation\_function(hidden\_inputs)  
  
 final\_inputs = np.dot(self.who, hidden\_outputs)  
 final\_outputs = self.activation\_function(final\_inputs)  
  
 # Compute errors  
 output\_errors = targets - final\_outputs  
 hidden\_errors = np.dot(self.who.T, output\_errors)  
  
 # Backpropagation and weight update  
 self.who += self.lr \* np.dot((output\_errors \* final\_outputs \* (1.0 - final\_outputs)), hidden\_outputs.T)  
 self.wih += self.lr \* np.dot((hidden\_errors \* hidden\_outputs \* (1.0 - hidden\_outputs)), inputs.T)  
  
  
 def query(self, inputs\_list):  
 # Convert input list to 2D array  
 inputs = np.array(inputs\_list, ndmin=2).T  
 hidden\_inputs = np.dot(self.wih, inputs)  
 hidden\_outputs = self.activation\_function(hidden\_inputs)  
 final\_inputs = np.dot(self.who, hidden\_outputs)  
 final\_outputs = self.activation\_function(final\_inputs)  
 return final\_outputs  
  
  
def load\_own\_images(image\_folder):  
 own\_dataset = []  
 for image\_file\_name in glob.glob(f'{image\_folder}/\*.png'):  
 label = int(  
 image\_file\_name[-5:-4]) # Предполагается, что имя файла заканчивается на '\_X.png', где X — метка  
 print(f"[INFO] Loading image {image\_file\_name}")  
 img\_array = imageio.imread(image\_file\_name, mode='F')  
 img\_data = 255.0 - img\_array.reshape(784)  
 img\_data = (img\_data / 255.0 \* 0.99) + 0.01  
 record = np.append(label, img\_data)  
 own\_dataset.append(record)  
 return own\_dataset  
  
  
def test\_own\_images(neural\_network, own\_dataset):  
 for item in own\_dataset:  
 correct\_label = int(item[0])  
 inputs = item[1:]  
 outputs = neural\_network.query(inputs)  
 label = np.argmax(outputs)  
 print(f"[INFO] Network prediction: {label}, Correct label: {correct\_label}")  
 plt.imshow(inputs.reshape(28, 28), cmap='Greys', interpolation='None')  
 plt.show()  
 if label == correct\_label:  
 print("Match!")  
 else:  
 print("No match!")  
  
  
# Main function  
def main():  
 # Neural network hyperparameters for variant 6  
 input\_nodes = 784 # Number of input nodes (pixels in the image)  
 hidden\_nodes = 140 # Number of hidden nodes  
 output\_nodes = 10 # Number of output nodes (digits 0-9)  
 learning\_rate = 0.3 # Learning rate  
 epochs = 5 # Number of training epochs  
  
 # Create a neural network instance  
 n = NeuralNetwork(input\_nodes, hidden\_nodes, output\_nodes, learning\_rate)  
  
 print("[INFO] Loading training dataset...")  
  
 # Load training dataset  
 try:  
 with open("mnist\_train.csv", 'r') as training\_data\_file:  
 training\_data\_list = training\_data\_file.readlines()  
 except FileNotFoundError:  
 print("[ERROR] mnist\_train.csv not found!")  
 return  
  
 print("[INFO] Training the neural network...")  
  
 # Train the neural network  
 for e in range(epochs):  
 print(f"[INFO] Epoch {e + 1}/{epochs} in progress...")  
 for record in training\_data\_list:  
 all\_values = record.split(',')  
 inputs = (np.asarray(all\_values[1:], dtype=np.float32) / 255.0 \* 0.99) + 0.01  
 targets = np.zeros(output\_nodes) + 0.01  
 targets[int(all\_values[0])] = 0.99  
 n.train(inputs, targets)  
  
 print("[INFO] Training completed!")  
  
 print("[INFO] Loading test dataset...")  
  
 # Load test dataset  
 try:  
 with open("mnist\_test.csv", 'r') as test\_data\_file:  
 test\_data\_list = test\_data\_file.readlines()  
 except FileNotFoundError:  
 print("[ERROR] mnist\_test.csv not found!")  
 return  
  
 print("[INFO] Testing the neural network...")  
  
 scorecard = []  
 for record in test\_data\_list:  
 all\_values = record.split(',')  
 correct\_label = int(all\_values[0])  
 inputs = (np.asarray(all\_values[1:], dtype=np.float32) / 255.0 \* 0.99) + 0.01  
 outputs = n.query(inputs)  
 label = np.argmax(outputs)  
 scorecard.append(1 if label == correct\_label else 0)  
  
 # Compute and print accuracy  
 scorecard\_array = np.asarray(scorecard)  
 accuracy = scorecard\_array.sum() / scorecard\_array.size  
 print(f"[INFO] Network accuracy = {accuracy:.4f}")  
  
 print("[INFO] Loading and testing on own images...")  
 own\_images = load\_own\_images('my\_own\_images')  
 test\_own\_images(n, own\_images)  
 print("[INFO] Program execution finished.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("[INFO] Neural Network program started...")  
 main()

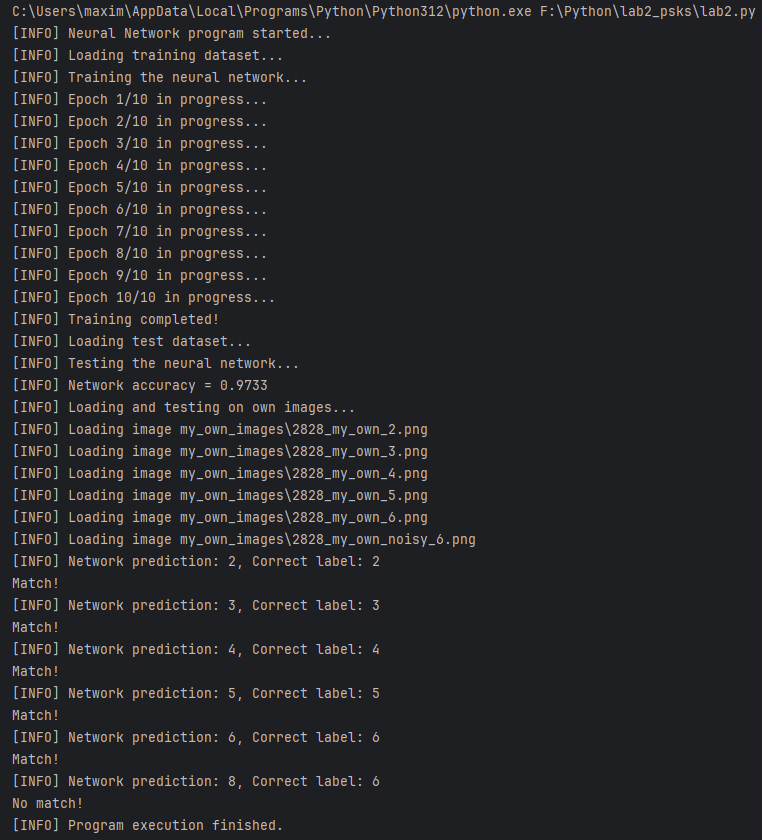
**Тестирование**

Провели тестирование на наборах изображений и с разными заданными параметрами для нейронной сети.

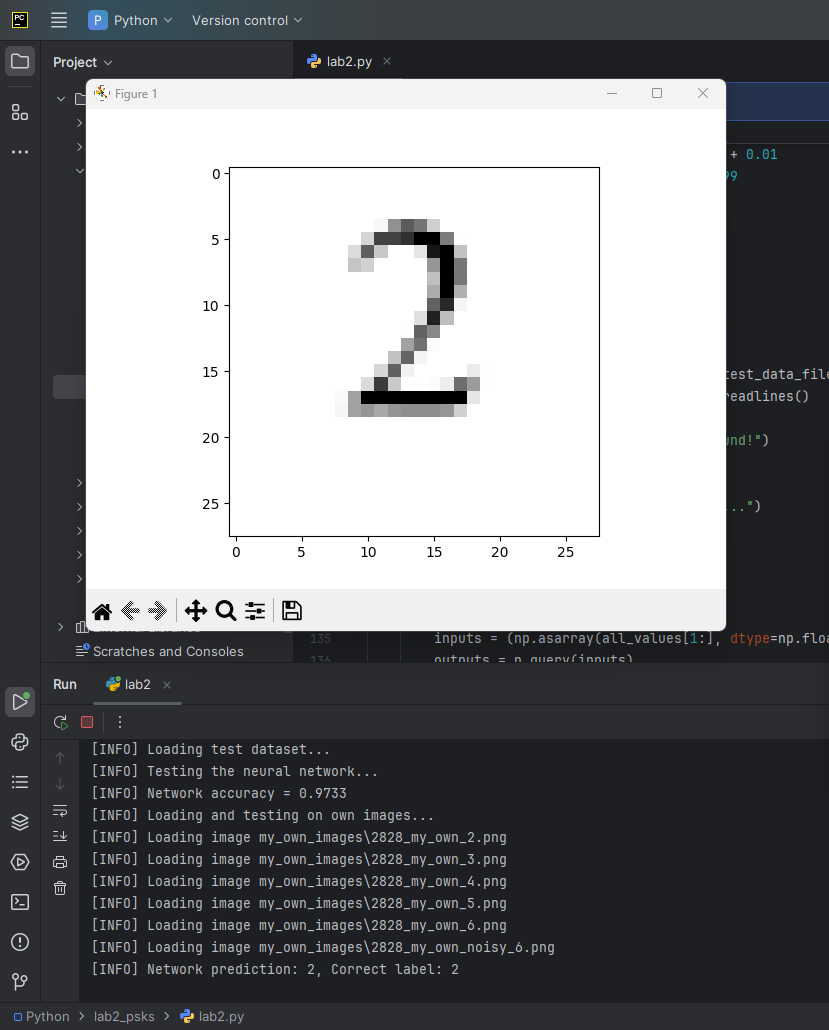
Сначала выбрали параметры:

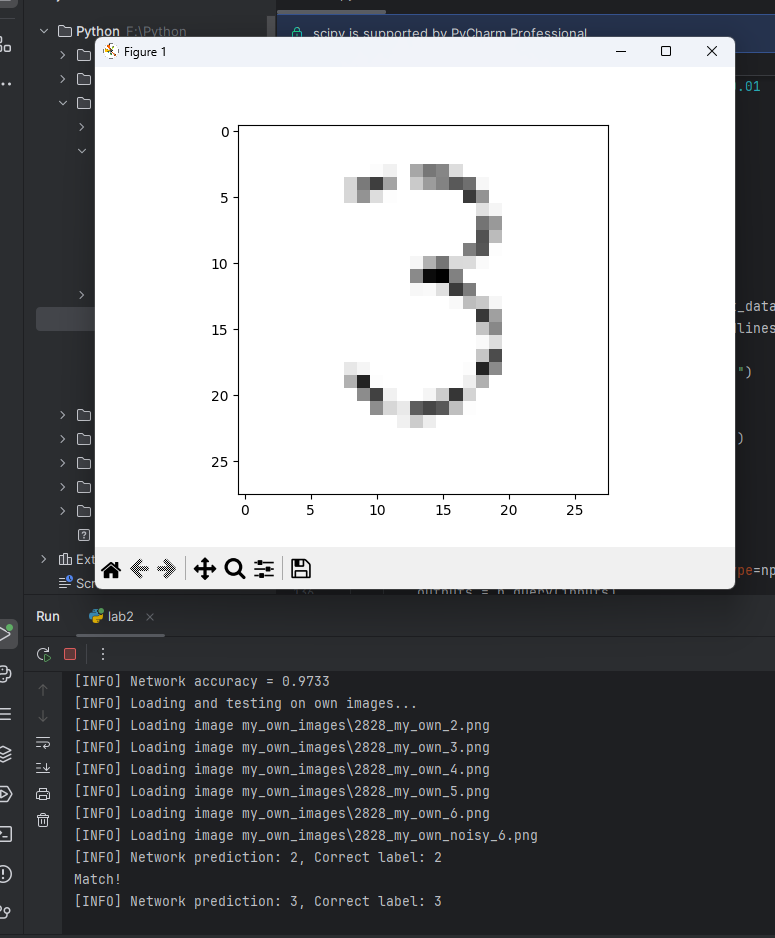


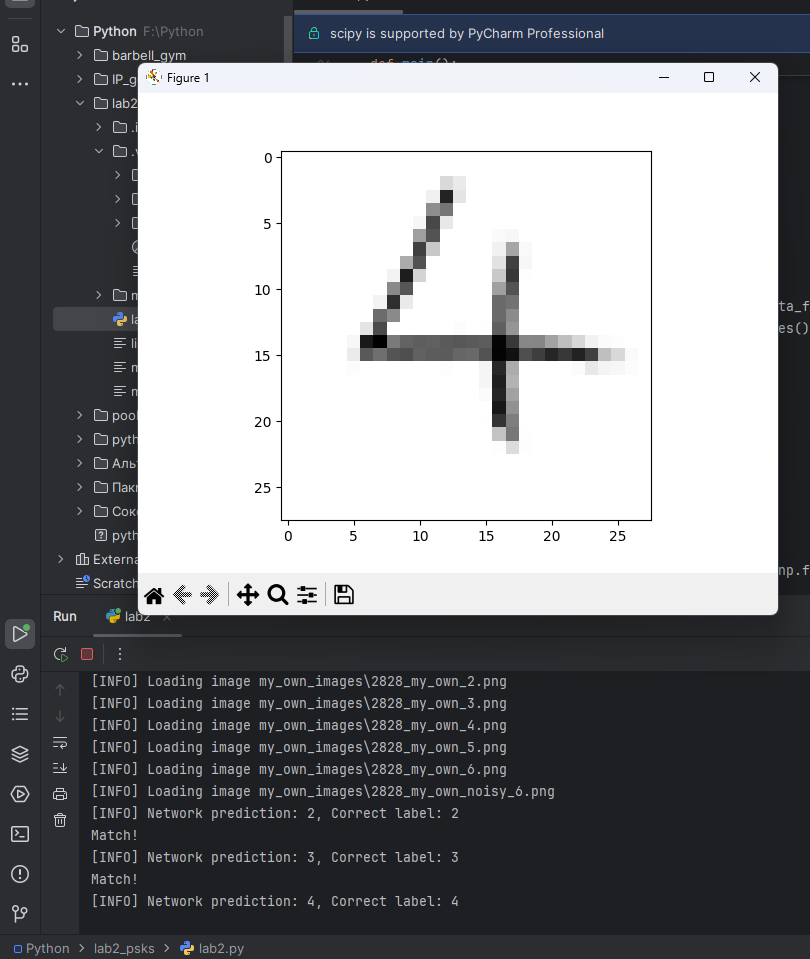
Получили:

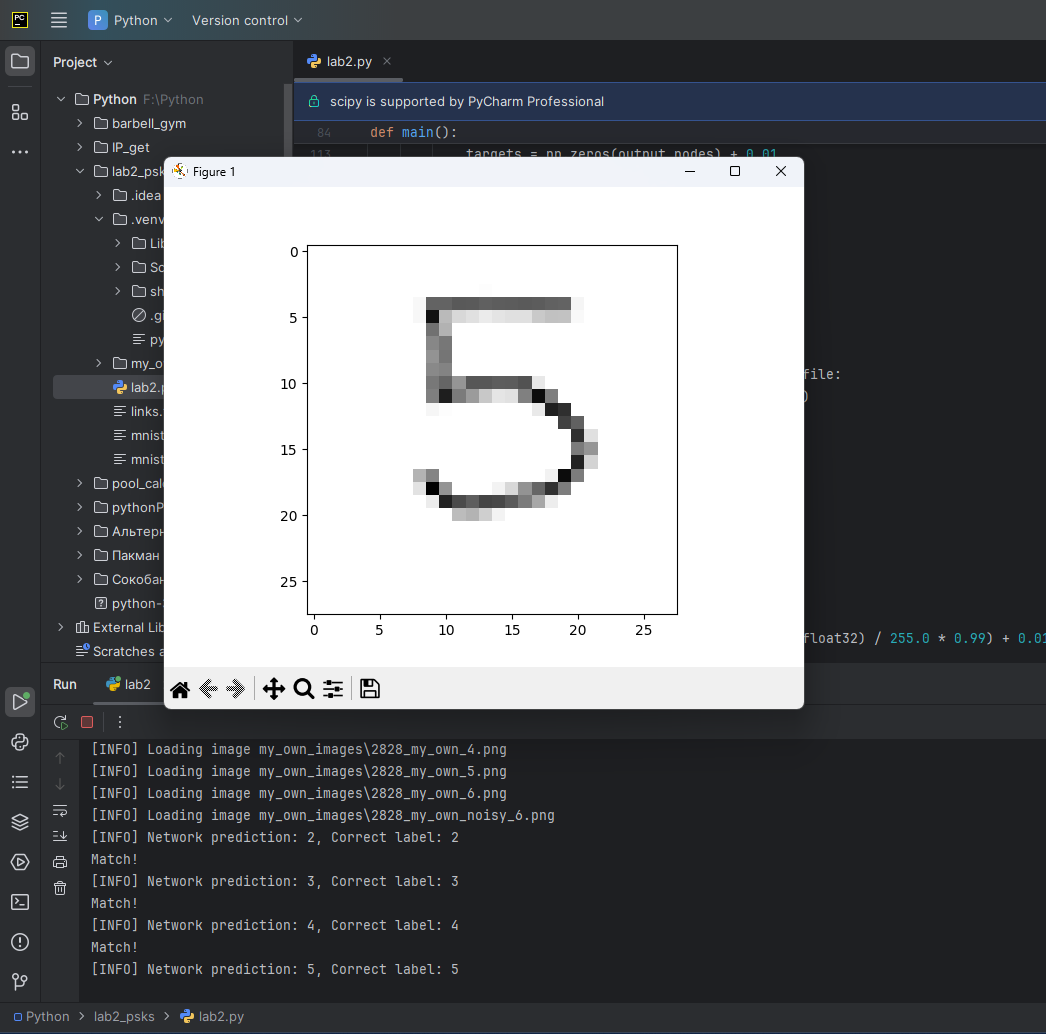


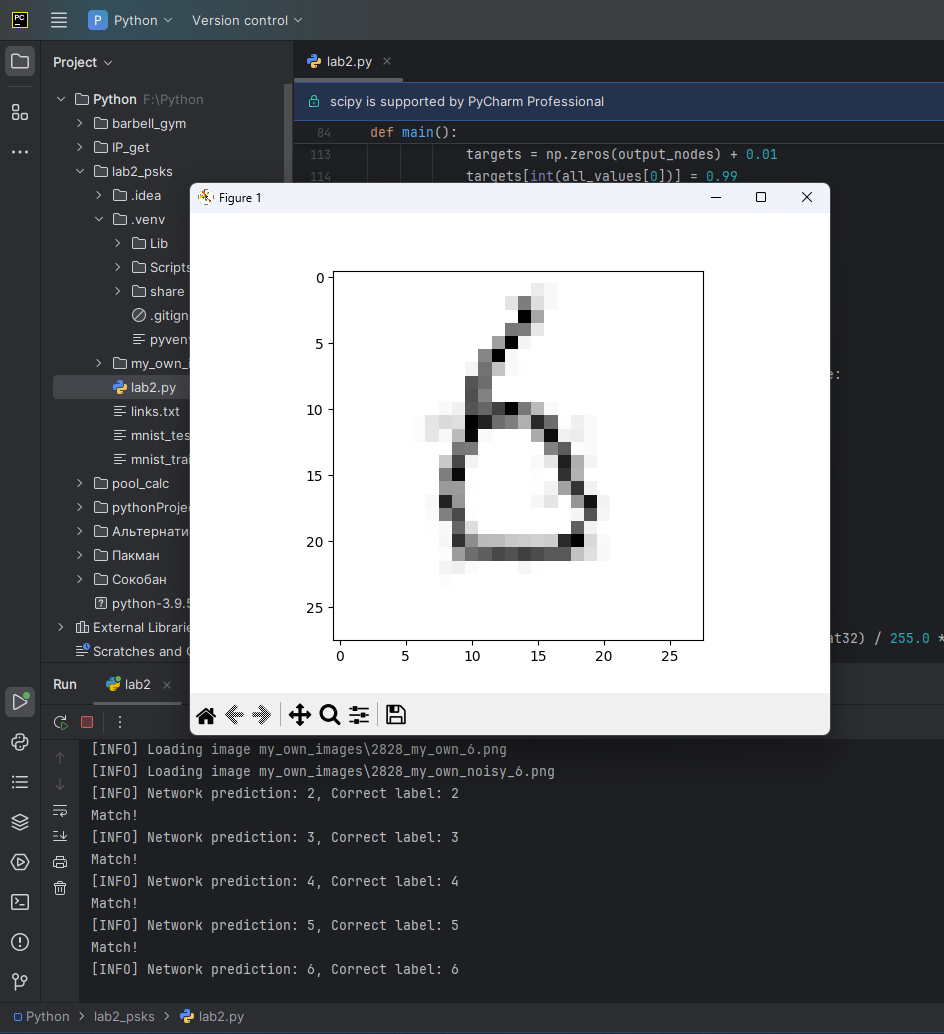
Далее результат работы на первом наборе данных.

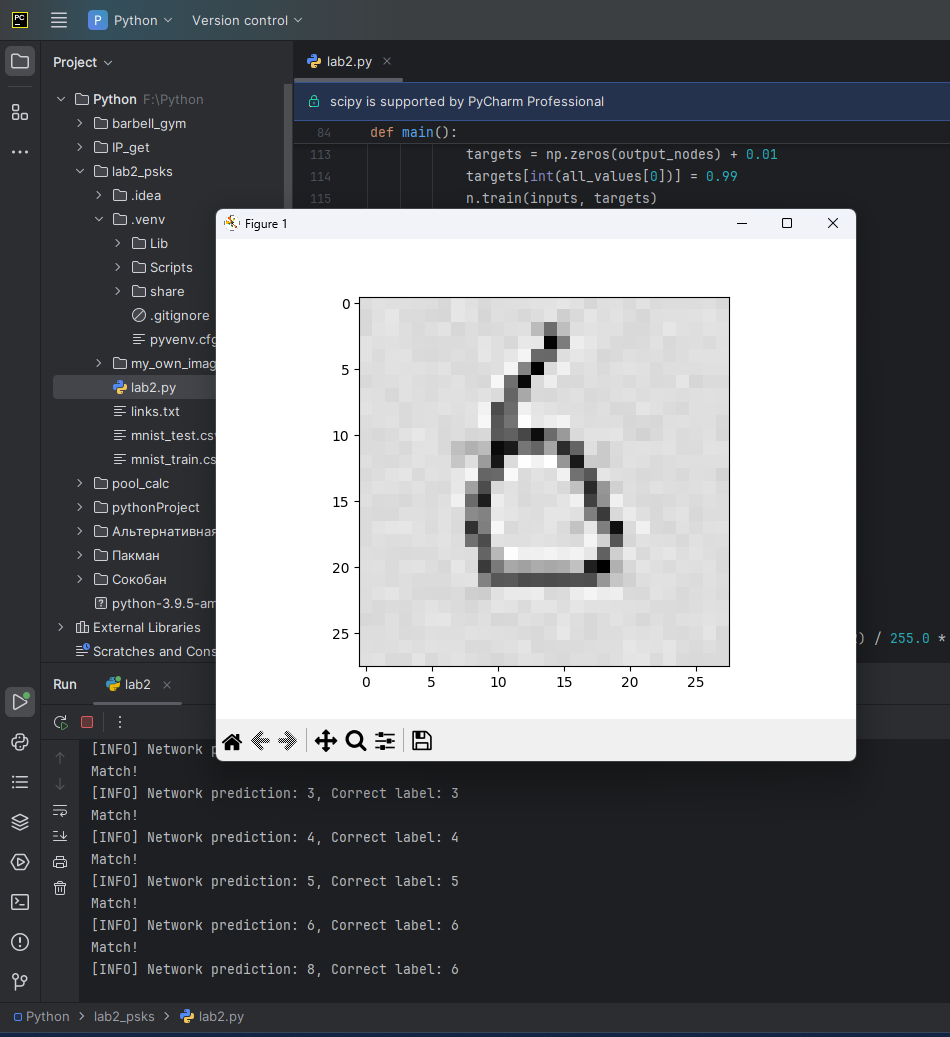


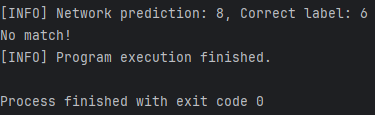




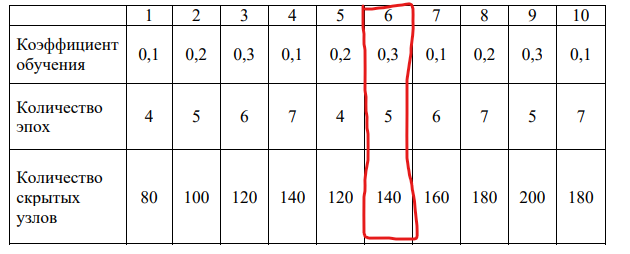


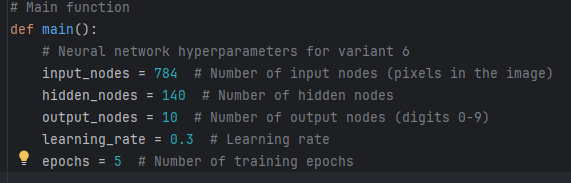


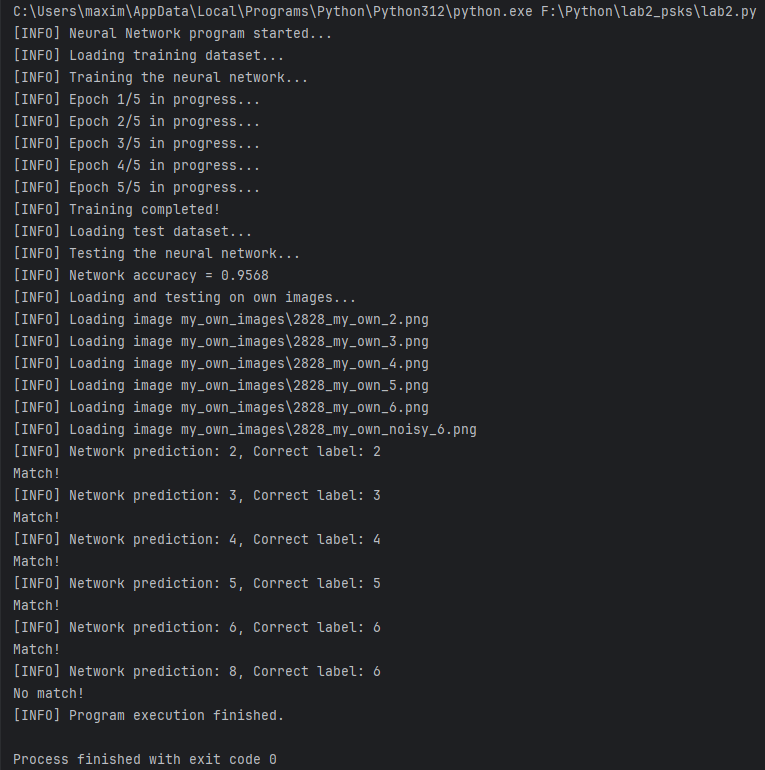


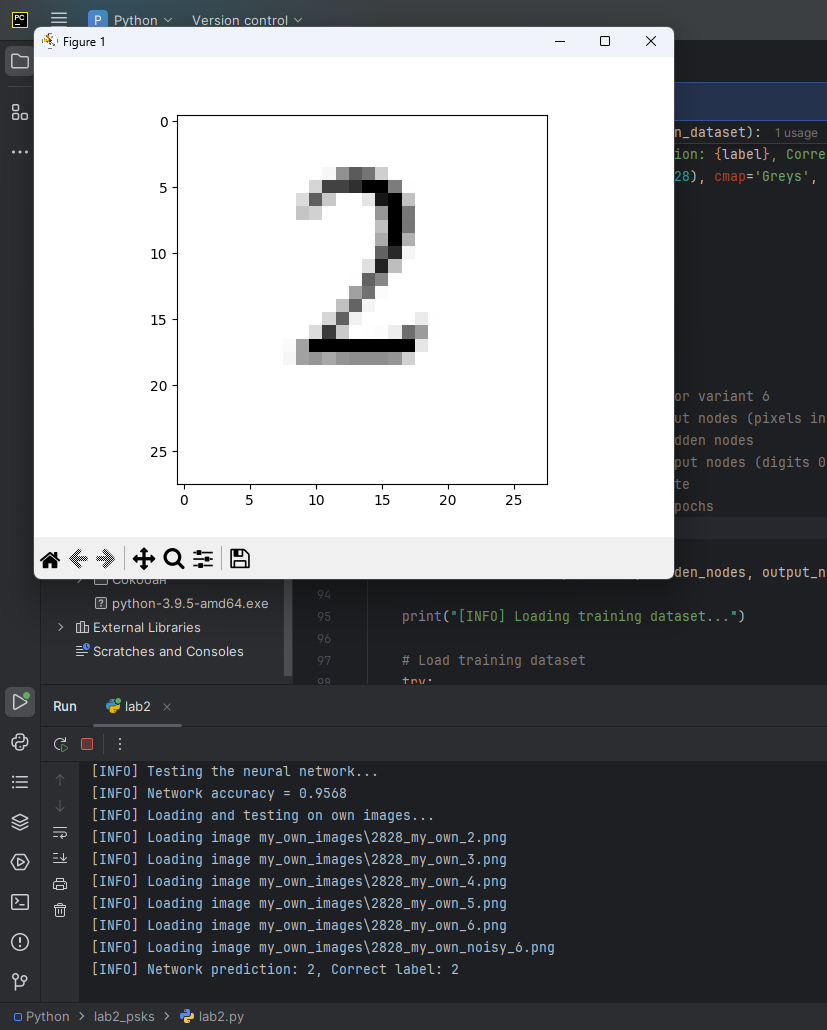


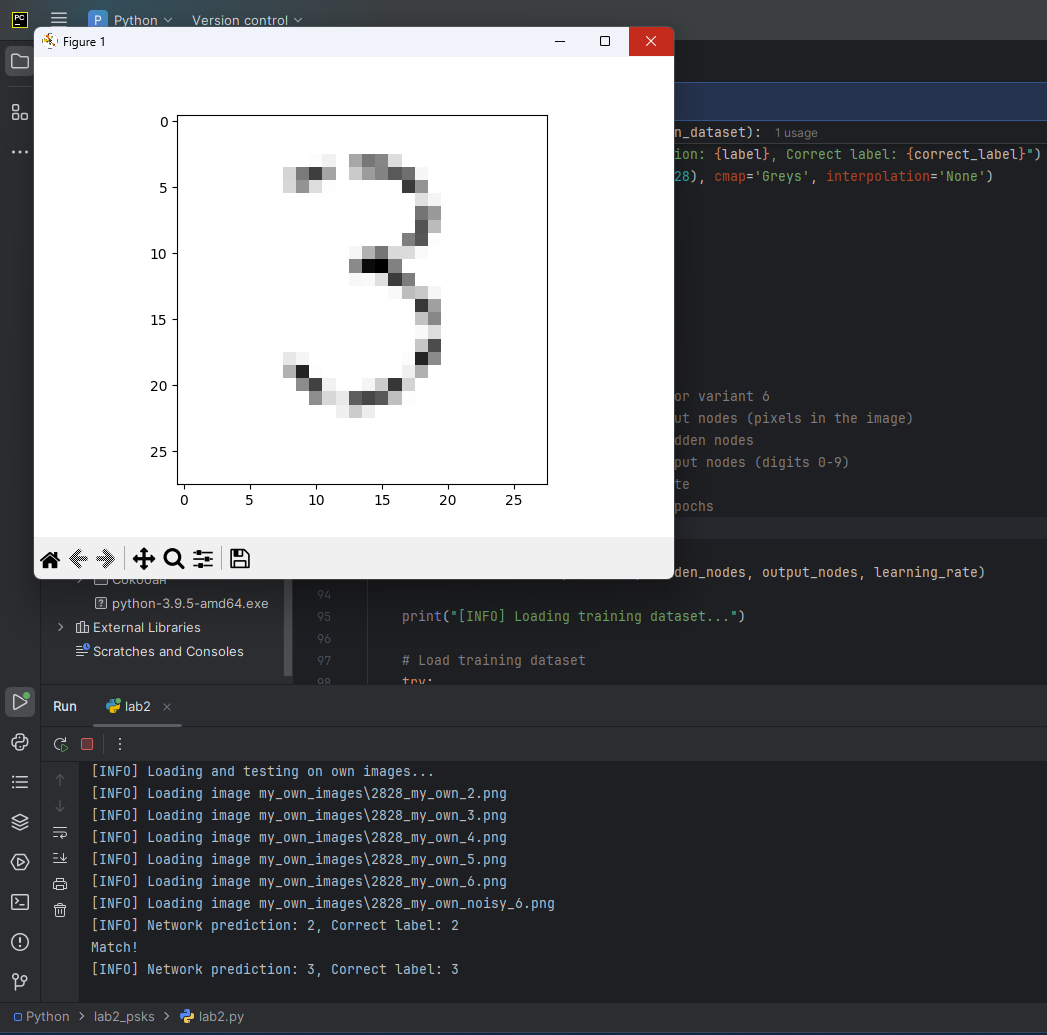
Далее взяли набор данных из нашего варианта и провели тестирование еще раз:

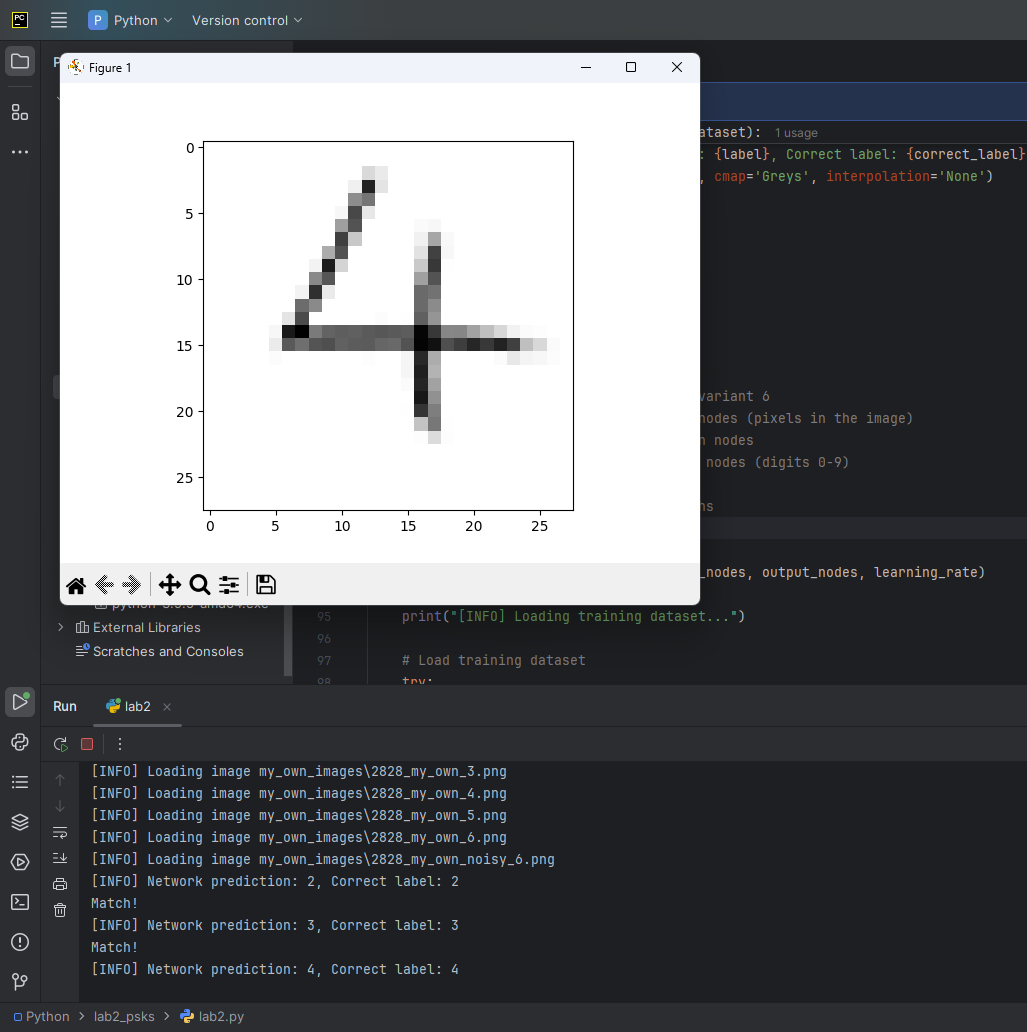


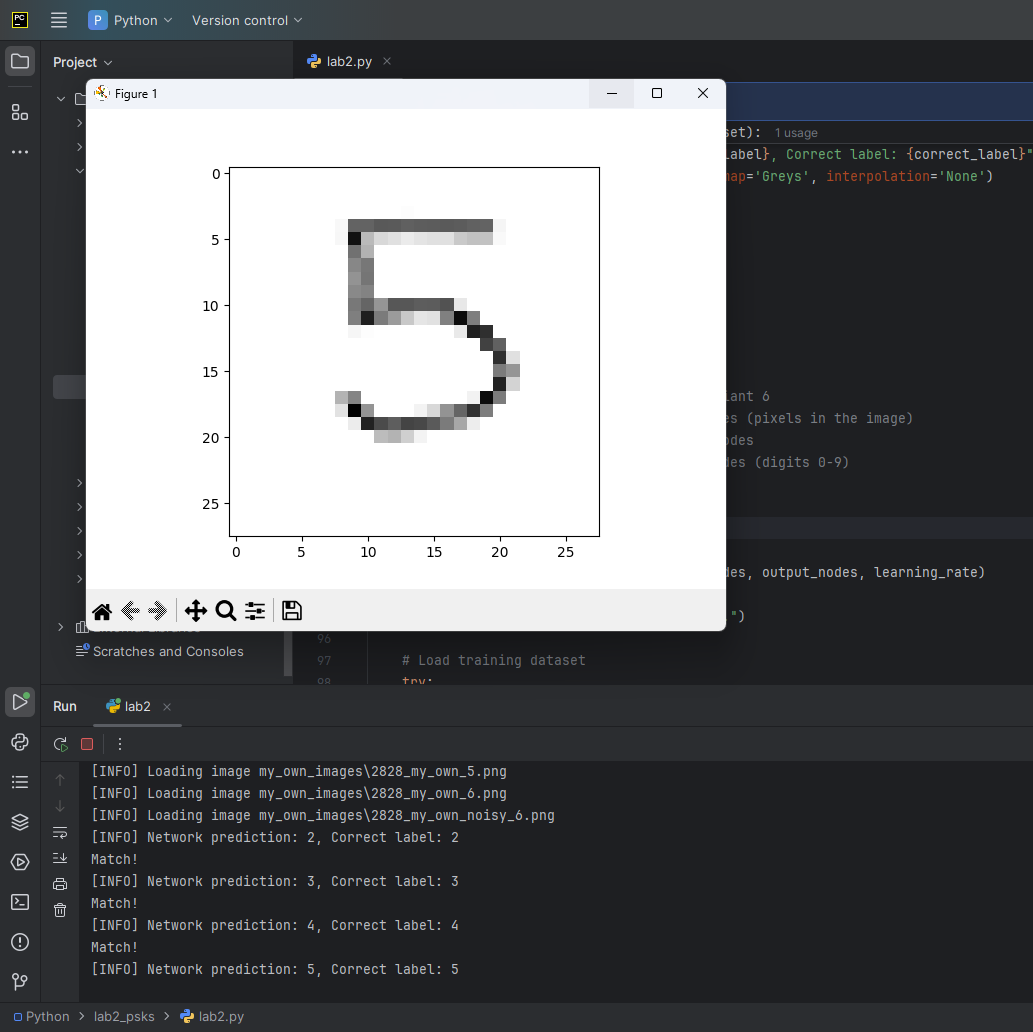


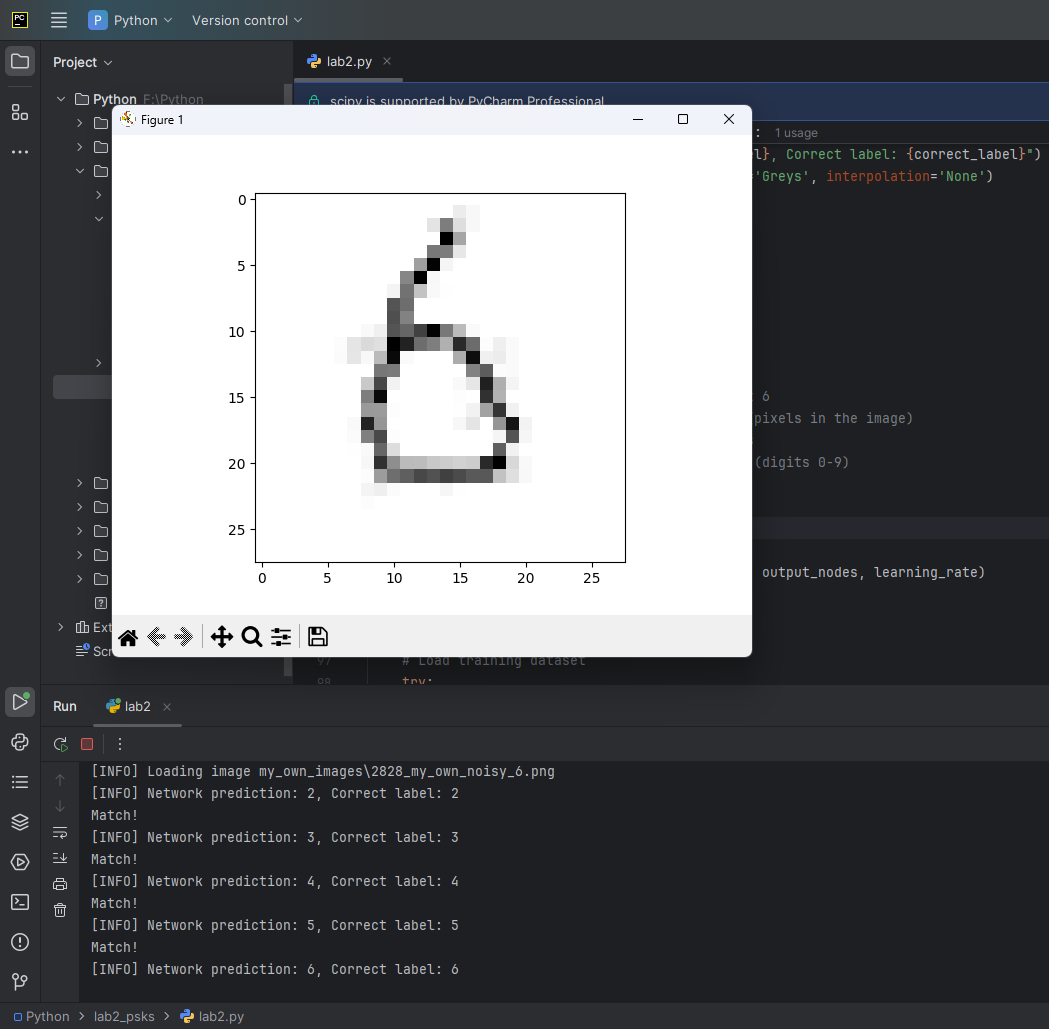


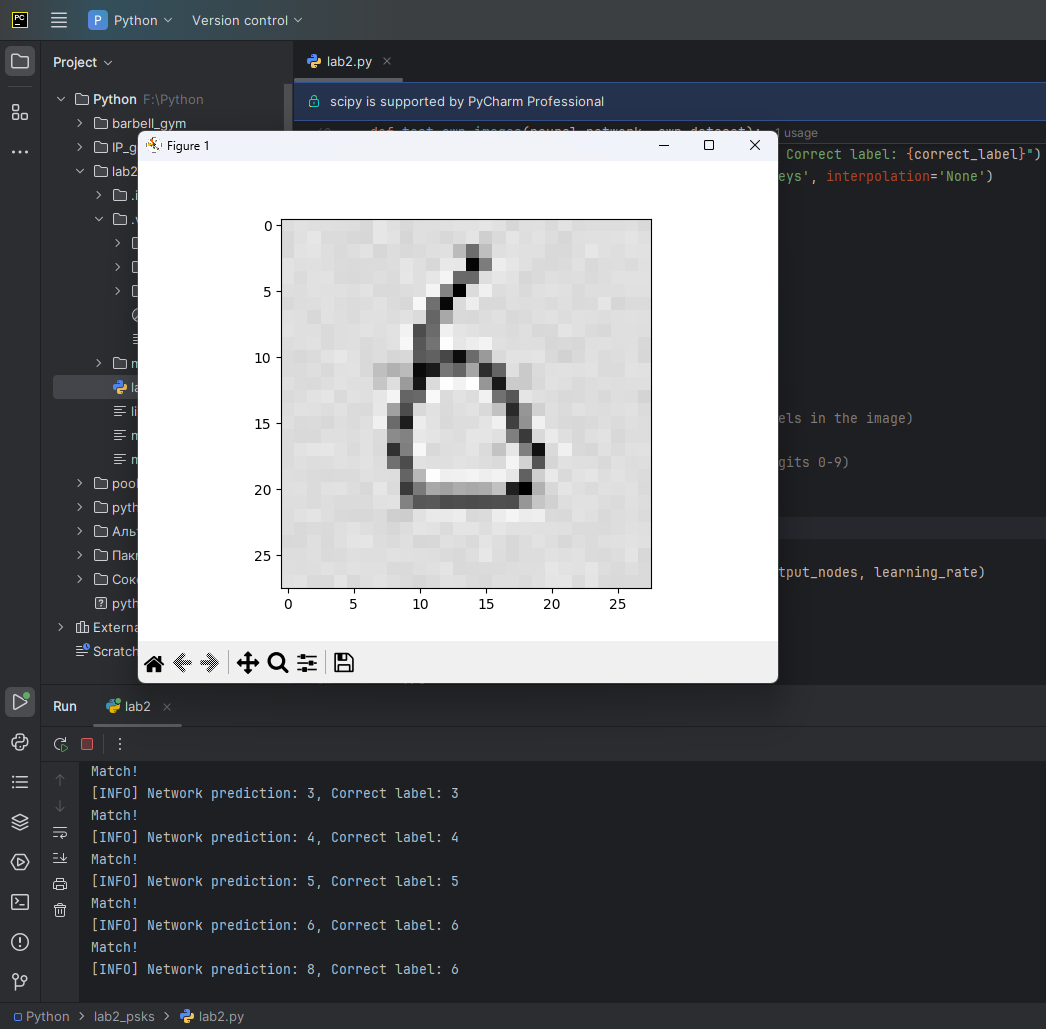


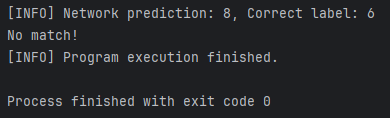












**Вывод**

Нейросети не удалось распознать последнюю цифру. Она угадывает в рукописи цифру «8», но это 6. Не зависимо от коэффициента обучения, количества эпох, количества скрытых узлов нейросеть выдаёт одинаковый результат. Мы изучили программы по распознаванию рукописных цифр при помощи трехслойной нейронной сети.