Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Программные средства кибертехнических систем»

# на тему «Работа с нейронной сетью»

Вариант №3

Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Зинкин С. И.

Карамышева Н. С

Пенза 2022

**Цель работы**: изучение программы по распознаванию рукописных цифр при помощи трехслойной нейронной сети.

**Лабораторное 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.

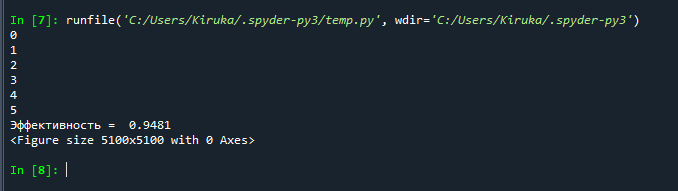
Опросить сеть с использованием тестовой базы данных (собственноручно написанных цифр).

|  |  |
| --- | --- |
|  | 3 |
| Коэф. Обучения | 0.3 |
| Кол. Эпох | 6 |
| Кол. Скр. Узлов | 120 |

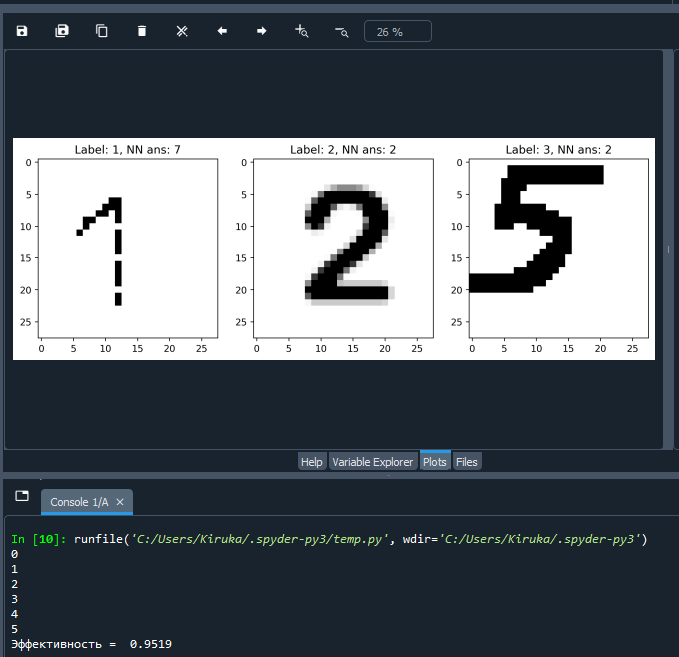
**Первая часть задания.**

1. Написали код в рабочую область и проанализировали работоспособность программы.
2. Протестировали работу программы с учетом параметров по варианту.

Результат теста программы:

  
**Вторая часть задания.**

1. После проверки на тестовых наборах данных, приступили к созданию собственных цифр
2. Нарисовали цифры 1, 2, 5 и сохранили в формате PNG с размером 28х28пикселей, для корректной работы с набором тренировочных данных.



**Вывод:** Научились изучать писать программу по распознаванию рукописных цифр при помощи трехслойной нейронной сети.

Сравнивая результаты пришли к выводу, что эффективность больше всего зависит от кол-ва скрытых узлов и скорости обучения. Чем больше скрытых узлов и меньше скорость обучения, тем эффективность больше.

**Листинг.**

**import numpy**

**# библиотека scipy.special содержитсигмоиду expit ()**

**import scipy.special**

**import scipy.misc**

**import glob**

**import matplotlib.pyplot as plt**

**import os**

**import imageio**

**# определение класса нейронной сети**

**class neuralNetwork:**

**def \_\_init\_\_(self, inputnodes, hiddennodes, outputnodes, learningrate):**

**self.inodes = inputnodes**

**self.hnodes = hiddennodes**

**self.onodes = outputnodes**

**self.wih = numpy.random.normal(0.0, pow(self.inodes, -0.5), (self.hnodes, self.inodes))**

**self.who = numpy.random.normal(0.0, pow(self.hnodes, -0.5), (self.onodes, self.hnodes))**

**self.lr = learningrate**

**self.activation\_function = lambda x: scipy.special.expit(x)**

**pass**

**def train(self, inputs\_list, targets\_list):**

**inputs = numpy.array(inputs\_list, ndmin=2).T**

**targets = numpy.array(targets\_list, ndmin=2).T**

**hidden\_inputs = numpy.dot(self.wih, inputs)**

**hidden\_outputs = self.activation\_function(hidden\_inputs)**

**final\_inputs = numpy.dot(self.who, hidden\_outputs)**

**final\_outputs = self.activation\_function(final\_inputs)**

**output\_errors = targets - final\_outputs**

**hidden\_errors = numpy.dot(self.who.T, output\_errors)**

**self.who += self.lr \* numpy.dot((output\_errors \* final\_outputs \* (1.0 - final\_outputs)), numpy.transpose(hidden\_outputs))**

**self.wih += self.lr \* numpy.dot((hidden\_errors \* hidden\_outputs \* (1.0 - hidden\_outputs)), numpy.transpose(inputs))**

**return output\_errors**

**pass**

**def query(self, inputs\_list):**

**inputs = numpy.array(inputs\_list, ndmin=2).T**

**hidden\_inputs = numpy.dot(self.wih, inputs)**

**hidden\_outputs = self.activation\_function(hidden\_inputs)**

**final\_inputs = numpy.dot(self.who, hidden\_outputs)**

**final\_outputs = self.activation\_function(final\_inputs)**

**return final\_outputs**

**# количество входных, скрытых и выходных узлов**

**input\_nodes = 784**

**hidden\_nodes =140**

**output\_nodes =10**

**# коэффициент обучения**

**learning\_rate = 0.1**

**# создать экземпляр нейронной сети**

**n = neuralNetwork(input\_nodes,hidden\_nodes,**

**output\_nodes,learning\_rate)**

**# загрузить в список тренировочный набор данных**

**# CSV-файланабора MNIST**

**training\_data\_file = open("./mnist\_train.csv", 'r')**

**training\_data\_list = training\_data\_file.readlines()**

**training\_data\_file.close()**

**# тренировка нейронной сети**

**# переменная epochs указывает, сколько раз**

**#тренировочныйнабор данных используется для тренировки #сети**

**epochs = 7**

**loss\_list = []**

**for e in range(epochs):**

**# перебрать все записи в тренировочном наборе данных**

**for record in training\_data\_list:**

**# получить список значений из записи, используя символы**

**# запятой (',') в качестве разделителей**

**all\_values = record.split(',')**

**# масштабировать и сместить входные значения**

**inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01**

**# создать целевые выходные значения (все равны #0,01, заисключением желаемого маркерного значения, #равного 0,99)**

**targets = numpy.zeros(output\_nodes) + 0.01**

**# all\_values[0] - целевое маркерное значение**

**# для данной записи**

**targets[int(all\_values[0])] =0.99**

**loss = n.train(inputs, targets)**

**loss\_list.append(loss)**

**pass**

**print(e)**

**pass**

**# загрузить в список тестовый набор данных**

**# CSV-файланабора MNIST**

**test\_data\_file = open("./mnist\_test.csv", 'r')**

**test\_data\_list = test\_data\_file.readlines()**

**test\_data\_file.close()**

**# тестирование нейронной сети**

**# журнал оценок работы сети, первоначально пустой**

**scorecard = []**

**# перебрать все записи в тестовом наборе данных**

**for record in test\_data\_list:**

**# получить список значений из записи, используя #симво-лызапятой (',\*) в качестве разделителей**

**all\_values = record.split(',')**

**# правильный ответ - первое значение**

**correct\_label = int(all\_values[0])**

**# масштабировать и сместить входные значения**

**inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01**

**# опроссети**

**outputs = n.query(inputs)**

**# индекс наибольшего значения является маркерным #значе-нием**

**label = numpy.argmax(outputs)**

**# присоединить оценку ответа сети к концу списка**

**if (label == correct\_label):**

**# в случае правильного ответа сети присоединить**

**# к списку значение 1**

**scorecard.append(1)**

**else:**

**# в случае неправильного ответа сети**

**# присоединитьк списку значение 0**

**scorecard.append(0)**

**pass**

**pass**

**# рассчитать показатель эффективности в виде**

**# долиправильныхответов**

**scorecard\_array = numpy.asarray(scorecard)**

**print ("Эффективность = ", scorecard\_array.sum() / scorecard\_array.size)**

**our\_own\_dataset = []**

**data\_root = './img'**

**for image\_file\_name in glob.glob(os.path.join(data\_root, '\*.png')):**

**# label = int(image\_file\_name.split('\\')[-1].split('\_')[0])**

**label = int(image\_file\_name[-5:-4])**

**img\_array = imageio.imread(image\_file\_name, as\_gray = True)**

**img\_data = 255.0 - img\_array.reshape(784)**

**img\_data = (img\_data / 255.0 \* 0.99) + 0.01**

**record = numpy.append(label,img\_data)**

**our\_own\_dataset.append(record)**

**pass**

**# выводим результат работы на собственном датасете**

**fig = plt.figure(figsize=(4, 5), dpi = 255)**

**fig.set\_size\_inches(20, 20)**

**for item in range(0, len(our\_own\_dataset)):**

**correct\_label = our\_own\_dataset[item][0]**

**inputs = our\_own\_dataset[item][1:]**

**outputs = n.query(inputs)**

**label = numpy.argmax(outputs)**

**fig.add\_subplot(4, 5, item+1)**

**plt.imshow(our\_own\_dataset[item][1:].reshape(28,28), cmap='Greys', interpolation='None')**

**plt.title("Label: {0}, NN ans: {1}".format(int(correct\_label), label))**