Всероссийский форум научной молодежи
"Шаг в будущее"

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ: ПРИМЕНЕНИЕ В РАСПОЗНАВАНИИ ОБЪЕКТОВ НА ПРИМЕРЕ ЦИФР

Выполнил:
Мыреев Софрон Алексеевич, 10 класс
МБОУ «Политехнический лицей»
г. Мирного Республики Саха (Якутия)

Руководитель:
Давыдова Анна Юрьевна
Зам.начальника Научно-технического
отдела АК «АЛРОСА» (ПАО)

Москва, 2020

Содержание

Введение

Актуальность. В 50-х годах ученые не знали, почему человеческий мозг разумен. Они задались вопросом: «Как работает человеческий мозг в физическом смысле, и можно ли создать искусственную версию этого разума?»

Мозг состоит из миллиардов нейронов, которые соединяются друг с другом в сеть, передающую информацию с помощью маломощных электрических зарядов. Из этой, казалось бы, простой биологической системы возникает что-то более глубокое: разум, который может распознавать лица, вырабатывать философию, изучать физику частиц и другое. Ученые задались задачей воссоздать эту биологическую систему электронным путем, создать искусственный интеллект.

Таким образом, искусственная нейронная сеть (ИНС) - это математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей - сетей нервных клеток живого организма.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Для решения какой-либо задачи на компьютере традиционным методом необходимо знать правила (например, математические формулы), по которым можно найти решение задач. А с помощью ИНС можно найти решение, не зная правил, а имея несколько примеров.

Еще несколько лет назад нейросети считались чем-то диковинным. Сейчас же данный инструмент осваивает все больше людей и компаний. Эта отрасль развивается быстрыми темпами.

Проблемой в данной теме мы видим, что деятельность, связанная с нейросетями, требует глубокой подготовки, обширных знаний, использования нестандартных методик. Начинающему разработчику

сложно приобщиться к созданию ИНС в связи с отсутствием доступной для понимания литературы.

Чтобы по-настоящему в чем-то разобраться, нужно сделать это самому и продолжить по принципу «начинай с малого… затем наращивай».

Цель работы - создать собственную искусственную нейронную сеть, способную классифицировать объекты по образу на примере цифр.

Задачи:

- изучение теоретической информации и обзор использованной литературы;
- выбор среды разработки и реализации ИНС;
- постановка задачи для ИНС;
- подбор базы данных для обучения;
- создание ИНС;
- обучение ИНС;
- проверка адекватности обучения;
- экспериментальный подбор параметров эффективного обучения.

Основная часть

1. Обзор использованной литературы

Так как тема искусственных нейронных сетей является относительно «молодой», литературы по ней на русском языке мало. Основная литература, которая носит базовый характер, является переведенной с английского языка.

Кроме того, в сети интернет доступны небольшие статьи по данной теме, которые чаще всего носят прикладной характер.

Большая часть статей и книг, сведения в которых повторяются, написана сложным языком с приведением сложных формул из области высшей математики – начинающему исследователю-разработчику нейронных сетей данная литература трудна для восприятия.

Часть статей и книг посвящены разбору и описанию функционала готовых библиотек – направлены для использования и решения узконаправленных задач для специалистов уже имеющих опыт работы.

Более доступное изложение информации приведено в книге Рашида Тарика «Создаем нейронную сеть» [3], в которой рассмотрено изучение математических принципов, лежащих в основе нейронных сетей. Книга дает четкое понимание того, как работают нейросистемы, что помогает написать собственную нейросеть.

Кроме того, полезным оказалась книга Саймона Хайкина «Нейронные сети. Полный курс», где рассмотрены нейронные сети при решении задач распознавания образов [5].

Для написания нейронной сети на языке Python использована литература [1], [2], [4].

2. Среда разработки и реализации ИНС

Среда разработки - это система программных средств для разработки программного обеспечения.

Нами рассмотрены следующие общеизвестные среды разработки:

- PyCharm;

- Microsoft Visual Studio;

- Jupiter Notebook.

Для работы выбрана среда разработки - Jupiter Notebook - это командная оболочка для интерактивных вычислений.

Jupyter Notebook – мощный и удобный инструмент для интерактивной разработки и представления проектов в области наук о данных.

Jupyter поддерживает множество языков программирования и может быть легко запущен на любом сервере, необходим только доступ по ssh или http.

В сравнении с другими инструментами Jupyter Notebook вышеперечисленные возможности предлагает бесплатно.

Jupyter Notebook – удобный инструмент для создания красивых аналитических отчетов, так как он позволяет хранить вместе код, изображения, комментарии, формулы и графики.

С таким инструментом удобно экспериментировать с кодом, в которых часто приходится что-то поправлять. Он позволяет редактировать часть кода и при этом не перезапуская программу полностью.

Jupyter Notebook поддерживает множество языков программирования и может быть легко запущен на любом сервере.

Нами рассмотрены следующие языки программирования:

- Java;

- C++;

- Python.

Языком программирования нашей ИНС выбран Python - высокоуровневый язык программирования общего назначения.

Основными преимуществами Python является:

- минимализм синтаксиса ядра Python (легко писать и читать);

- простота кода;
- стандартная библиотека включает большой объём полезных функций.

Разработанный код ИНС приведен в приложении 1.

3. Задача для ИНС

ИНС подходят для распознавания образов и решения задач классификации, оптимизации и прогнозирования.

Они отлично решают поставленные задачи, когда отсутствует алгоритм или не известны принципы решения задач, но накоплено достаточное число примеров; проблема характеризуется большими объемами входной информации; данные неполны или избыточны, зашумлены, частично противоречивы.

Задача для нашей ИНС - классификация объектов по образам на примере распознавания цифр.

4. База данных для обучения

Важнейшим условием для успешного обучения ИНС является наличие обширной базы данных.

Мы использовали базу данных MNIST (сокращение от «Modified National Institute of Standards and Technology») – это объёмная база данных образцов рукописного написания цифр, которая содержит:

- 60 тысяч изображений для обучения;
- 10 тысяч изображений для тестирования.

5. Обучение ИНС и проверка адекватности обучения

Длительность обучения зависит от мощности компьютера (например, на моем компьютере обучение ИНС данное задаче длится 1-15 минут в зависимости от параметров сети и базы данных).

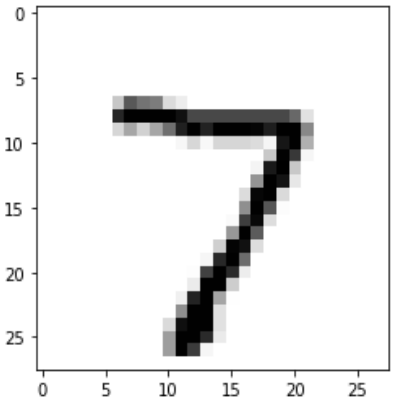
За это время ИНС обрабатывает очень большой объем информации – обучается изучая каждую цифру из базы данных.

Самым важным и трудоёмким моментом в обучении является создание базы данных.

Результат проверки адекватности нашей ИНС в результате тестирования оказался успешным.

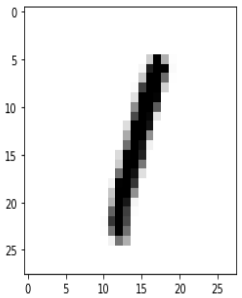
Например, рукописную цифру «7» наша ИНС предположила, что с вероятностью 99,6% это цифра «7».

```
Это: 7 (99.60751020917762 %)
Out[9]: <__main__.screen at 0x1f2671e9eb8>
```

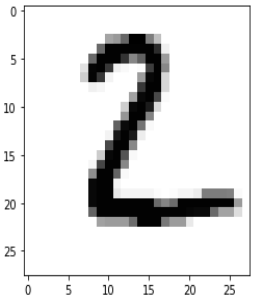


Таким образом, в результате тестирования были распознаны большинство предложенных рукописных цифр, примеры которых приведены ниже.

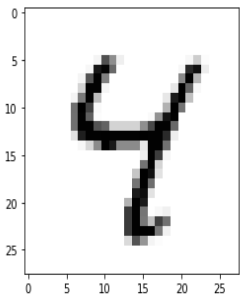
```
In [14]: screen(5)
Это: 1 (98.92564249189962 %)
Out[14]: <__main__.screen at 0x1f26f663a90>
```



```
In [10]: screen(1)
Это: 2 (95.05645589891988 %)
Out[10]: <__main__.screen at 0x1f26f4d3ef0>
```

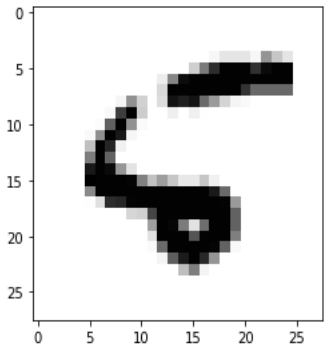


```
In [15]: screen(6)
Это: 4 (97.80828656646203 %)
Out[15]: <__main__.screen at 0x1f270684518>
```



Ниже приведен результат тестирования на более сложном варианте.

```
In [17]: screen(8)
         Это: 6 (54.17249250764067 %)
Out[17]: <__main__.screen at 0x1f270753198>
```



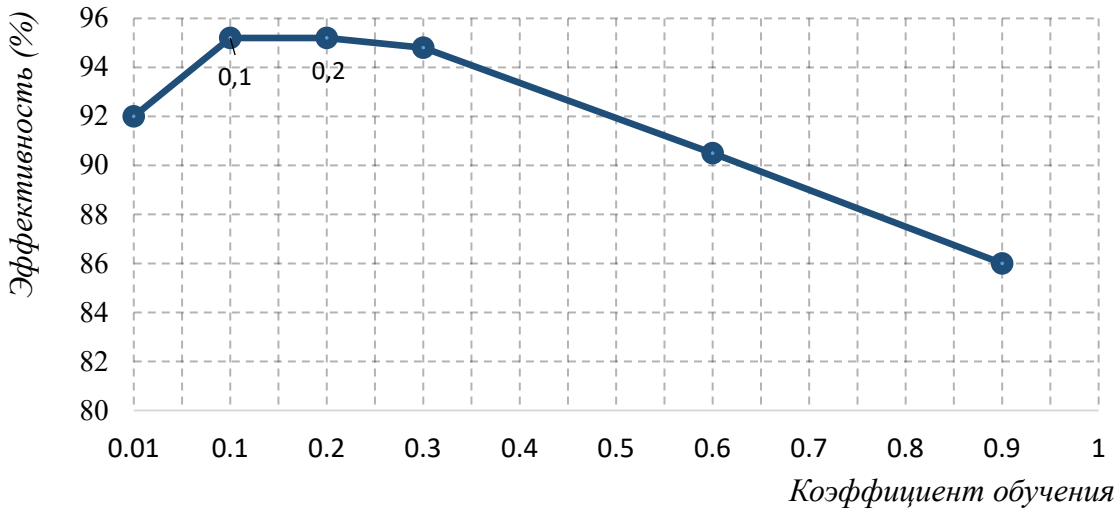
ИНС предположила, что с вероятностью 54% это цифра «6».

Данный пример показывает насколько ИНС «умна» и распознает на уровне человеческого интеллекта. Даже человек видя эту цифру не предположит на 100%, что это цифра «6».

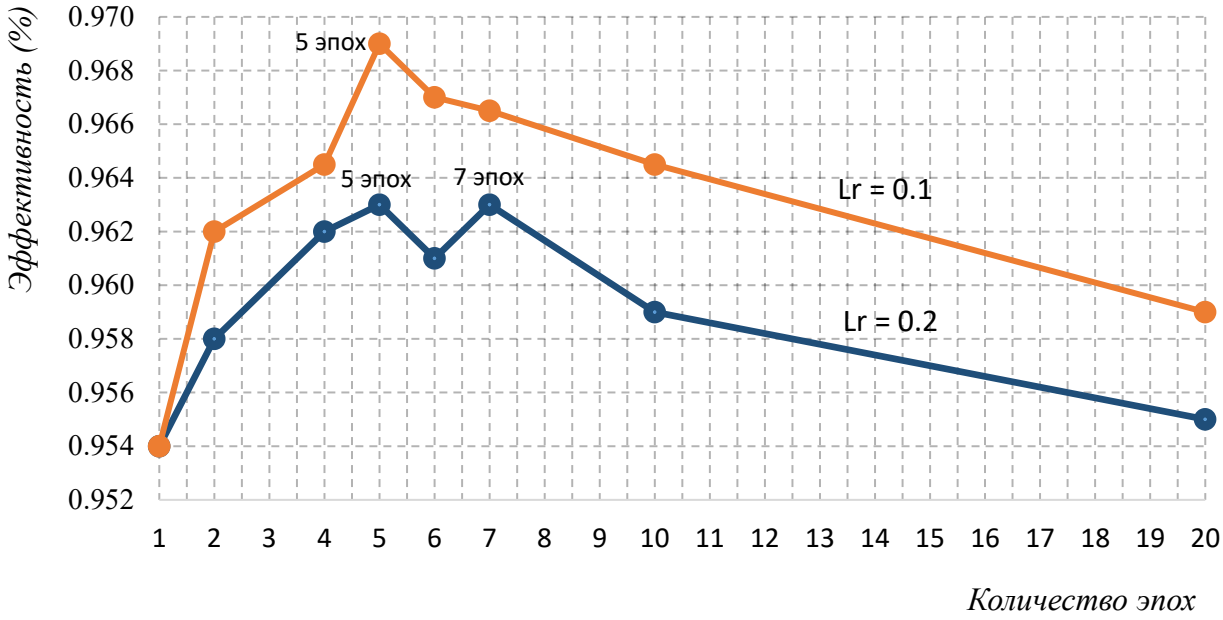
6. Экспериментальное определение зависимости эффективности ИНС от параметров обучения

Для определения зависимости эффективности нашей ИНС от параметров ее обучения провели эксперименты с различными параметрами:

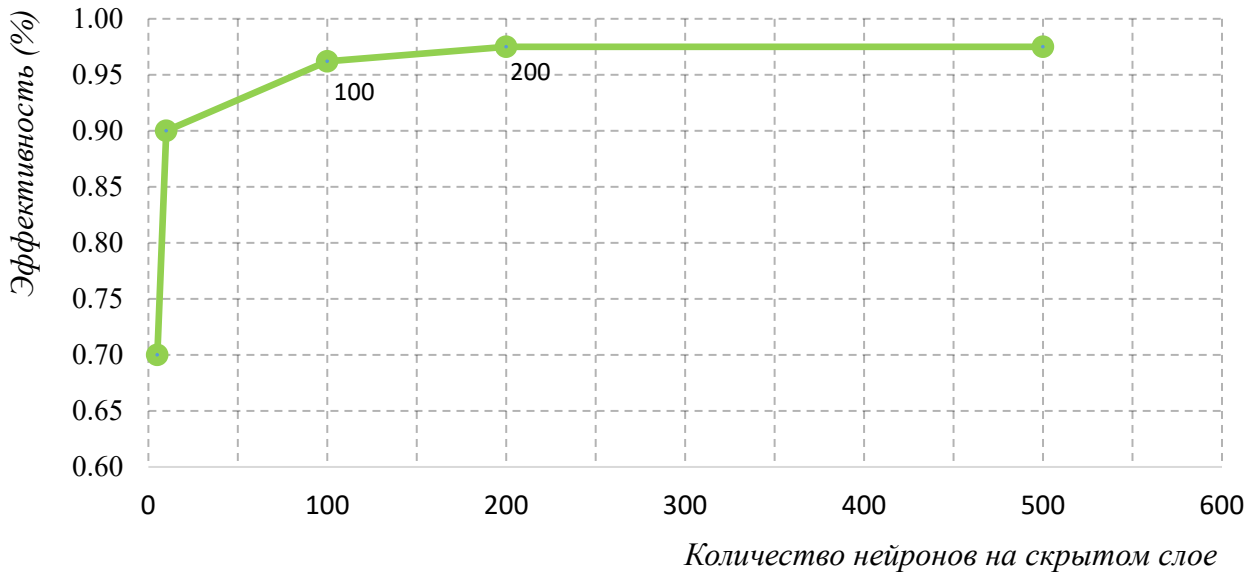
1. с различными ***коэффициентами обучения***; результаты показали, что наибольшая эффективность ИНС находится в диапазоне коэффициента обучения от 0,1 до 0,2;



2. с различным ***количеством эпох*** (количество прохождения обучения на одной базе данных) при коэффициентах 0,1 и 0,2; результаты показали, что наибольшая эффективность ИНС достигается при количестве эпох 5 с коэффициентом 0,1; следует отметить, что после 5 эпох обучения начинается спад эффективности ИНС;



3. с различным ***количеством нейронов на скрытом слое***; результаты показали, что оптимальным количеством нейронов на скрытом слое является 100-200 скрытых узлов.



7. Испытания ИНС на устойчивое распознавание искаженных входных данных

В связи с тем, что почерки людей бывают самые различные, кроме того цифры могут быть расположены не в центре области (наверху, внизу, с краю), нами проведены испытания нашей ИНС по распознаванию искаженных входных данных.

Для определения устойчивости нашей ИНС к искаженным входным данным проведены испытания посредством ручного ввода данных (рукописных цифр) с использованием планшета и графического редактора.

Так как ИНС была обучена на «правильной» базе данных, где цифры центрованы, расположены относительно ровно, первые испытания при распознавании рукописных цифр показали значительное количество ошибок.

В связи с этим, программный код ИНС был доработан – добавлено центрирование картин, в результате чего нейросеть стала правильно распознавать рукописные цифры.

Заключение

В работе разработана искусственная нейронная сеть распознавания объектов на примере цифр. На примере распознавания цифр показаны возможности обучения нейронных сетей.

Данные принципы обучения ИНС можно применить для распознавания картин, лиц, голоса, для решения сложных задач и др.

Результат проверки адекватности нашей ИНС во время тестирования оказался успешным (97% предложенных цифр были распознаны верно).

Для определения зависимости эффективности нашей ИНС от параметров ее обучения проведены эксперименты, по результатам которых выявлено, что наибольшая эффективность достигается при количестве эпох 5 с коэффициентом обучения 0,1, с количеством нейронов на скрытом слое 100-200 узлов.

Таким образом используя в работе методы исследования:

- теоретические (сбор информации путем изучения литературы и других источников информации по нейронным сетям и языкам программирования);
- экспериментальные (создание нейронной сети, обучение и тестирование ее возможностей и эффективности распознавания)

мы разработали свою собственную нейронную сеть, основанную на простых идеях машинного обучения, для решения сложной задачи по распознаванию образов.

Данная работа будет полезной школьникам и начинающим разработчикам нейронных сетей для изучения машинного обучения и позволит приобщить к пониманию искусственного интеллекта широкий круг лиц с перспективой более глубокого развития данной темы. В умелых руках этот инструмент может делать удивительные вещи.

Список литературы

1. Джоши Пратик. Искусственный интеллект с примерами на Python /П. Джоши – М.: Издательство «Диалектика/Вильямс», 2019 – 448 с.
2. Мюллер Андреас. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными: Пер. с англ. /А. Мюллер – СПб.: ООО «Альфа-книга», 2018 – 480 с.
3. Рашид Тарик. Создаем нейронную сеть: Пер. с англ. /Т. Рашид – СПб.: ООО «Альфа-книга», 2017 – 272 с.
4. Сузи Р.А. Язык программирования Python /Р.А. Сузи – М.: НОУ «Интуит», 2007 – 292 с.
5. Хайкин Саймон. Нейронные сети. Полный курс. 2-е издание: Пер с англ. /С. Хайкин – М.: Издательский дом «Вильямс», 2006 – 1104 с.

Прочие источники информации

Электронные ресурсы:

- https://en.wikipedia.org/wiki/Artificial_neural_network;
- https://en.wikipedia.org/wiki/Neural_network;
- https://ru.wikipedia.org/wiki/Искусственная_нейронная_сеть.

Приложение 1

Программный код

Импортирование библиотек

```
import numpy
import scipy.special

import matplotlib.pyplot
from PIL import Image
import pickle
```

Центрирование картинки

```
def alignment(image):
    image = numpy.array(image).reshape(28,28)

    sum_rows = numpy.array([])
    sum_columns = numpy.array([])

    num_row = [0]
    num_arr = 0
    for bl in image:
        if (numpy.sum(bl)>0.28) == num_arr:
            num_row[-1] += 1
        else:
            num_arr = 1-num_arr
            num_row.append(1)

    if num_arr == 1:
        num_row.append(0)
        num_arr = 0

    rot_image = image.transpose()
    num_column = [0]
    for bl in rot_image:
        if (numpy.sum(bl)>0.28) == num_arr:
            num_column[-1] += 1
        else:
```

```
            num_arr = 1-num_arr
            num_column.append(1)

    if num_arr == 1:
        num_column.append(0)

    num_y, num_x = num_row[0] - (num_row[0]+num_row[2])//2,
num_column[0] - (num_column[0]+num_column[2])//2

    if num_x > -1:
        for del_x in range(num_x):
            rot_image = numpy.delete(rot_image, 0, 0)
            rot_image = numpy.append(rot_image,[[0.01 for l in range(28)]], axis=0)
    else:
        for del_x in range(-num_x):
            rot_image = numpy.delete(rot_image, -1, 0)
            rot_image = numpy.append([[0.01 for l in range(28)]], rot_image,
axis=0)
    image = rot_image.transpose()

    if num_y > -1:
        for del_y in range(num_y):
            image = numpy.delete(image, 0, 0)
            image = numpy.append(image,[[0.01 for l in range(28)]], axis=0)
    else:
        for del_y in range(-num_y):
            image = numpy.delete(image, -1, 0)
            image = numpy.append([[0.01 for l in range(28)]], image, axis=0)

    return numpy.asfarray(image.flatten())
```

Нейронная сеть

```
class neuralNetwork:
    def __init__(self, iNodes, hNodes, oNodes, lrRate):
        self._iNodes = iNodes
        self._hNodes = hNodes
        self._oNodes = oNodes
        self._lrRate = lrRate
```

```
        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.exp = lambda x: scipy.special.expit(x)
        pass

    def load(self, name_of_file="neuro.data"):
        with open(name_of_file, 'rb') as filehandle:
            info, self._iNodes, self._hNodes, self._oNodes, self._lrRate, self.wih,
self.who = pickle.load(filehandle)
        return info

    def save(self, name_of_file="neuro.data"):
        info = [txt, efficient_data, epochs_data]
        placesList = [info, self._iNodes, self._hNodes, self._oNodes, self._lrRate,
self.wih, self.who]
        with open(name_of_file, 'wb') as filehandle:
            pickle.dump(placesList, filehandle)
        pass

    def train(self, input_list, target_list):
        input_list = numpy.array(alignment(input_list), ndmin=2).T
        target_list = numpy.array(target_list, ndmin=2).T

        hidden_inputs = numpy.dot(self.wih, input_list)
        hidden_outputs = self.exp(hidden_inputs)
        final_inputs = numpy.dot(self.who, hidden_outputs)
        final_outputs = self.exp(final_inputs)

        output_errors = target_list - final_outputs
        hidden_errors = numpy.dot(self.who.T, output_errors)

        self.who += self._lrRate * numpy.dot((output_errors* final_outputs* (1.0 -
final_outputs)), numpy.transpose(hidden_outputs))
        self.wih += self._lrRate * numpy.dot((hidden_errors* hidden_outputs* (1.0
- hidden_outputs)), numpy.transpose(input_list))
        pass

    def query(self, input_list):
        out_inputs = numpy.array(alignment(input_list), ndmin=2).T
```

```
        hidden_inputs = numpy.dot(self.wih, out_inputs)
        hidden_outputs = self.exp(hidden_inputs)
        final_inputs = numpy.dot(self.who, hidden_outputs)
        final_outputs = self.exp(final_inputs)

        return final_outputs
```

Загрузка весов и кое-других данных, загрузка базы данных для тестирования, процесс обучения нейронной сети

```
ffnn = neuralNetwork(1, 1, 1, 0.0)
txt, efficient_data, epochs_data = ffnn.load('neuro_013.data') #File name!!!
print('information: ', txt)
print('efficient: ', efficient_data)
print('epochs: ', epochs_data)
```

```
information:  Feed forward neuronal network v.013
efficient:  0.9709
epochs:  5
```

```
test_data_file = open("mnist_test.csv", 'r')
test_data_list = test_data_file.readlines()
test_data_file.close()


scoreboard = []

for record in test_data_list:
    output = ffnn.query((numpy.asfarray(record.split(',')[1:])/255.0*0.99) + 0.01)
    label = numpy.argmax(output)
    scoreboard.append(int(label == int(record[0])))

scoreboard = numpy.asarray(scoreboard)
efficient_data = scoreboard.sum() / scoreboard.size
print("Эффективность: ", efficient_data)
```

Training

```
inputNodes = 784
hiddenNodes = 200
outputNodes = 10
learningRate = 0.1

ffnn = neuralNetwork(inputNodes, hiddenNodes, outputNodes, learningRate)


training_data_file = open("mnist_train.csv", 'r')
training_data_list = training_data_file.readlines()
training_data_file.close()


test_data_file = open("mnist_test.csv", 'r')
test_data_list = test_data_file.readlines()
test_data_file.close()


epochs_data = 3
for e in range(epochs_data):
   print(f'epoch {e+1} - in process...')
   for record in training_data_list:
      all_values_training = record.split(',')
      inputs_training = (numpy.asfarray(all_values_training[1:])/255.0*0.99) +
0.01
      targets = numpy.zeros(outputNodes) + 0.01
      targets[int(all_values_training[0])] = 0.99

      ffnn.train(inputs_training, targets)
```

Saving

```
#efficient_data = -1.0
epochs_data = 5
txt = 'Feed forward neuronal network v.013'

ffnn.save('neuro_013.data') #Don't forget to change file name!!!
```

Распознавание сторонних картинок

```
def image_info(file):
    img = Image.open(file)
    img.thumbnail((28, 28))
    img = numpy.asarray(img.convert('L'), dtype='uint8')
    img = 255 - img
    full_img=[]
    for lst in img:
        full_img.extend(lst)
    output = ffnn.query(full_img).flatten()
    hk = -1
    for h in output:
        hk += 1
        if h > 0.1:
            print(f'>>> {hk}: {h*95}/100')
    matplotlib.pyplot.imshow(img, cmap="Greys", interpolation="nearest")



image_info("Untitled.png")
```