Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

БГТУ.СМК-Ф-4.2-К5-01

Факультет И Информационные и управляющие системы
шифр наименование
Кафедра И5 Информационные системы и программная инженерия
шифр наименование
Дисциплина Моделирование систем представления знаний

Индивидуальная практическая работа №5 на тему «Моделирование и обучение искусственных нейронных сетей» Вариант «P R S B»

Выполнил студент группы И967
Масанов И.В.
Фамилия И.О.

ПРЕПОДАВАТЕЛЬ
Гущин А.Н.
Фамилия И.О. Подпись

Оценка _____

«_____» _____ 2020 г.

САНКТ-ПЕТЕРБУРГ

2020 г.

Постановка задачи

1. Сформировать обучающую и тестовую выборки на основе имеющихся в варианте задания образов для распознавания, желательно путем автоматизированного зашумления (с помощью специально для этого написанной программы).

2. Выполнить разработку программы, моделирующий искусственную нейронную сеть для распознавания образов согласно варианту задания.

3. Обучить полученную нейронную сеть в соответствии с её типом на обучающей выборке и проверить качество обучения с помощью тестовой выборки, при необходимости внести коррективы в параметры обучения и повторить данный пункт.

Перечень фактически использованных технических средств

Программа, созданная в рамках практической работы, была создана под управлением операционной системы Windows 10, разрядность системы составляет 64 бита, ОЗУ – 4 Гб. Использовался процессор Intel Core i5 с частотой 2,6 ГГц.

Архитектура ИНС

Исходя из условий задачи была выбрана архитектура, показанная на рисунке 1.

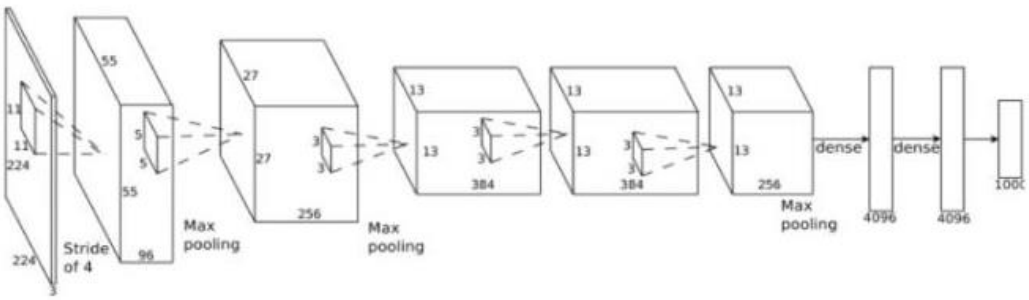

Рисунок 1 – Архитектура ИНС

Слои ИНС [1]:

1. Первым слоем ИНС является входной сверточный слой, размерность которого 64 нейронам, так как на вход мы получаем изображение 8x8. На выходе этого слоя мы получаем первые карты признаков;

2. слой «макспулинга», который снижает размерность карт признаков;

3. второй слой сверточной сети принимает полученные на предыдущем шаге карты, и на выходе дает другие карты признаков;

4. слой для преобразования полученных на предыдущем шаге карт в один вектор;

5. первый слой полносвязной сети принимает вектор, производит вычисления, которые дают значения для скрытого полносвязного слоя;

6. второй слой полносвязной сети, количество выходных нейронов которого равно количеству классов в используемом датасете;

7. выход всей модели подается в функцию потерь, которая сравнивает прогнозируемое значение с истинным, и вычисляет разницу между этими значениями.

Итоговая функция потерь является своего рода количественным “штрафом”, который можно рассматривать как меру качества прогноза модели. Это значение мы и будем использовать для обучения модели с помощью обратного распространения ошибки. Формулы, которые используют эту ошибку и «протягивают» ее сквозь все слои для обновления параметров и обучения модели.

В качестве функции активации выбрана функция soft-max, а в качестве функции потерь – cross-entropy, так как они наиболее подходят для задач классификации при количестве классов больше двух (показаны на рисунках 2 и 3 соответственно).

$$f_{softmax} = y_i^l = f(x_i^l) = \frac{e^{x_i^l}}{\sum_{k=0}^{n} e^{x_k^l}}$$

Рисунок 2 - Функция активации

$$E = -\sum_{i=0}^{n} y_i^{truth} ln(y_i^l)$$

Рисунок 3 - Функция потерь

Обратное распространение ошибки [2]

В качестве функции потерь была использована функция потерь cross-entropy, так как она наиболее подходит для задачи классификации (если у нас имеется более 2-х классов).

Формулы для обратного прохождения через функцию потерь:

$$\begin{aligned}
\frac{\partial E}{\partial y_i^l} &= \frac{\partial(-\sum_{i=0}^{n}(y_i^{truth} \cdot ln(y_i^l))}{\partial y_i^l} \\
&= \frac{\partial(-(y_0^{truth} ln(y_0^l) + \ldots + y_i^{truth} ln(y_i^l) + \ldots + y_n^{truth} ln(y_n^l))}{\partial y_i^l} \\
&= \frac{\partial(-y_i^{truth} ln(y_i^l))}{\partial y_i^l} = -\frac{y_i^{truth}}{y_i^l} \\
\frac{\partial y_i^l}{\partial x_j^l} &= \frac{\partial}{\partial x_j^l}\left(\frac{e^{x_i^l}}{\sum_{k=0}^{n} e^{x_k^l}}\right) = \frac{0 \cdot \sum_{k=0}^{n} e^{x_k^l} - e^{x_i^l} \cdot e^{x_j^l}}{\left(\sum_{k=0}^{n} e^{x_k^l}\right)^2} \\
&= \frac{e^{x_i^l} \cdot e^{x_j^l}}{\sum_{k=0}^{n} e^{x_k^l} \cdot \sum_{k=0}^{n} e^{x_k^l}} = -y_i^l \cdot y_j^l
\end{aligned}$$

Так как на выходной слой сети влияет каждый нейрон из предыдущего слоя, что показано на рисунке 4.

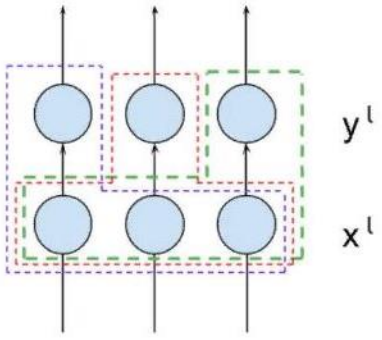

Рисунок 4 - Нейроны, влияющие на выходной слой

Следовательно, ошибка по каждому нейрону будет равна:

$$\frac{\partial E}{\partial x_i^l} = \frac{\partial E}{\partial y_0^l}\frac{\partial y_0^l}{\partial x_i^l} + ... + \frac{\partial E}{\partial y_1^l}\frac{\partial y_1^l}{\partial x_i^l} + ... + \frac{\partial E}{\partial y_n^l}\frac{\partial y_n^l}{\partial x_i^l} \qquad \forall i \in (0, ...n)$$

Далее обозначим δ_i^l, как $\frac{\partial E}{\partial x_i^l}$

Формулы для обратного прохождения через полносвязный слой:

Обратное распространение ошибки через полносвязный слой показан на рисунке 5 (w_i – веса, b_i – смещение).

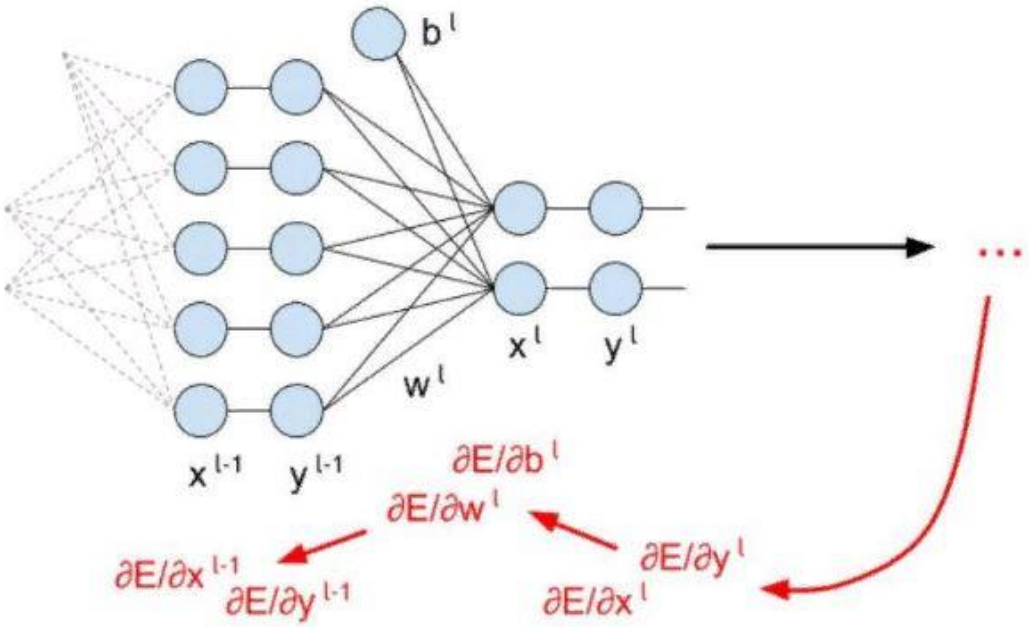

Рисунок 5 - Полносвязный слой

$$\begin{aligned}\frac{\partial E}{\partial w_{ki}^l} &= \frac{\partial E}{\partial y_i^l}\frac{\partial y_i^l}{\partial x_i^l}\frac{\partial x_i^l}{\partial w_{ki}^l} = \delta_i^l \cdot \frac{\partial x_i^l}{\partial w_{ki}^l} = \delta_i^l \cdot \frac{\partial \left(\sum_{k'=0}^{m} w_{k'i}^l y_{k'}^{l-1} + b_i^l\right)}{\partial w_{ki}^l} \\ &= \delta_i^l \cdot \frac{\partial \left(w_{0i}^l y_0^{l-1} + \ldots + w_{ki}^l y_k^{l-1} + ...w_{mi}^l y_m^{l-1} + b_i^l\right)}{\partial w_{ki}^l} = \delta_i^l \cdot y_k^{l-1} \\ &\forall i \in (0, ..., n) \;\; \forall k \in (0, ..., m)\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial b_i^l} &= \frac{\partial E}{\partial y_i^l}\frac{\partial y_i^l}{\partial x_i^l}\frac{\partial x_i^l}{\partial b_i^l} = \delta_i^l \cdot \frac{\partial x_i^l}{\partial b_i^l} \\ &= \delta_i^l \cdot \frac{\partial \left(\sum_{k'=0}^{m} w_{k'i}^l y_{k'}^{l-1} + b_i^l\right)}{\partial b_i^l} = \delta_i^l \\ &\forall i \in (0, ..., n)\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial y_k^{l-1}} &= \sum_{i=0}^{n} \delta_i^l \cdot \frac{\partial x_i^l}{\partial y_k^{l-1}} = \sum_{i=0}^{n} \delta_i^l \cdot \frac{\partial \left(\sum_{k'=0}^{m} w_{k'i}^l y_{k'}^{l-1} + b_i^l\right)}{\partial y_k^{l-1}} \\ &= \sum_{i=0}^{n} \delta_i^l \cdot \frac{\partial \left(w_{0i}^l y_0^{l-1} + \ldots + w_{ki}^l y_k^{l-1} + \ldots w_{mi}^l y_m^{l-1} + b_i^l\right)}{\partial y_k^{l-1}} \\ &= \sum_{i=0}^{n} \delta_i^l \cdot w_{ki}^l \\ &\quad \forall i \in (0, \ldots, n) \;\; \forall k \in (0, \ldots, m)\end{aligned}$$

Обратное прохождение ошибки через слой макспуллинга:

Ошибка “проходит” только через те значения исходной матрицы, которые были выбраны максимальными на шаге макспулинга. Остальные значения ошибки для матрицы будут равны нулю (что логично, ведь значения по этим элементам не были выбраны функцией макспулинга во время прямого прохождения через сеть и, соответственно, никак не повлияли на итоговый результат), показано на рисунке 6.

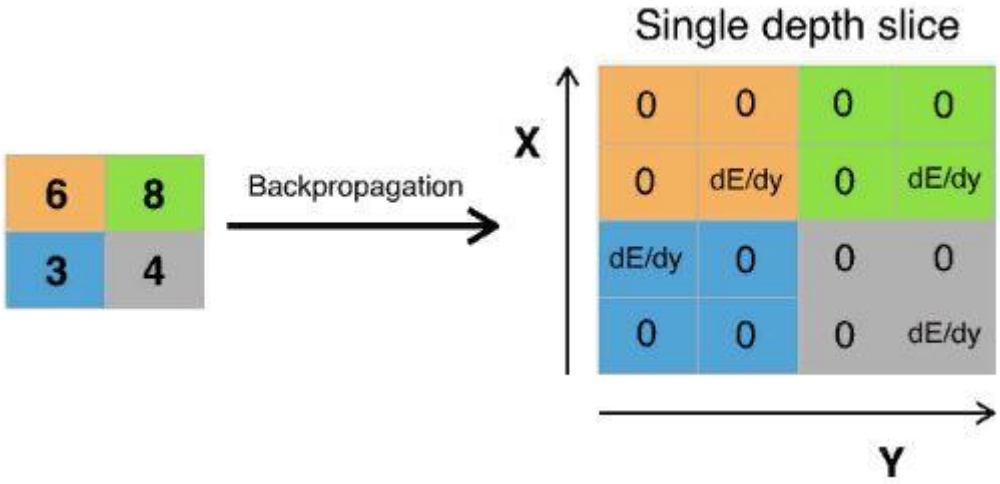

Рисунок 6 - Слой maxpooling

Формулы для обратного прохождения через сверточные слои:

Обратное распространение ошибки через сверточныый слой показан на рисунке 7 (w_i – веса).

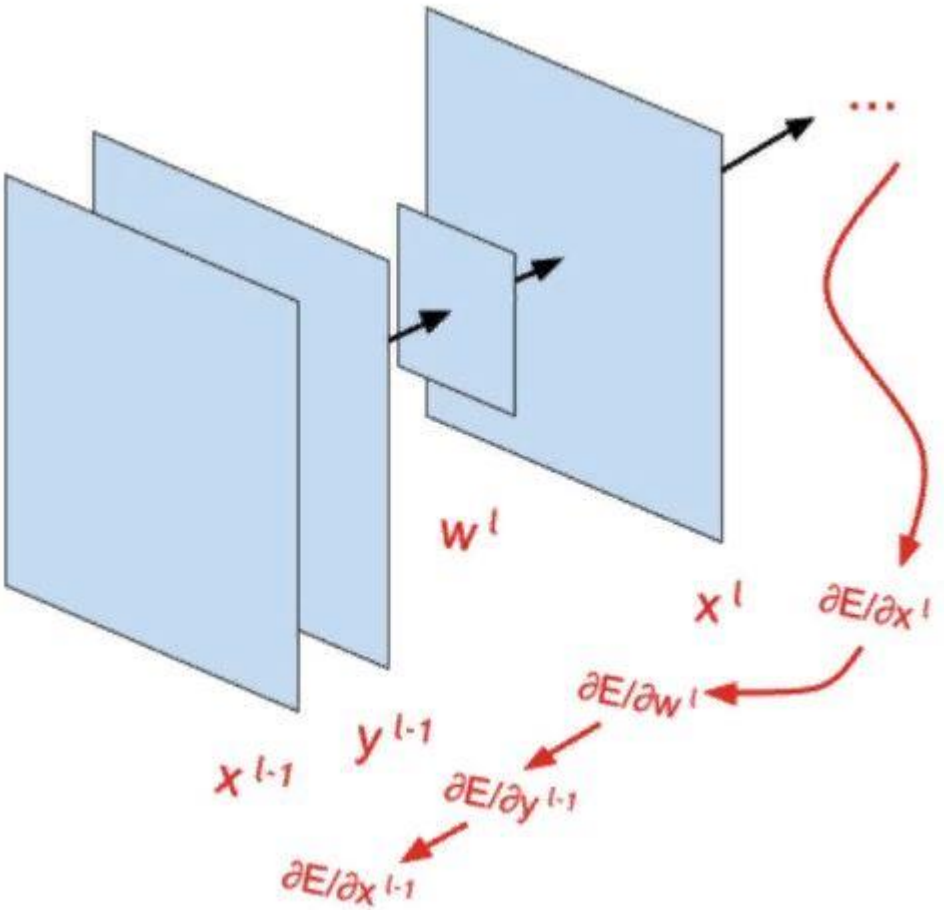

Рисунок 7 - Сверточный слой

Формулы для обновления ядра свертки:

$$\begin{aligned}\frac{\partial E}{\partial w_{ab}^l} &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial x_{ij}^l}{\partial w_{ab}^l} \\ &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \cdot \frac{\partial \left(\sum_{a'=-\infty}^{+\infty} \sum_{b'=-\infty}^{+\infty} w_{a'b'}^l \cdot y_{(is-a')(js-b')}^{l-1} + b^l \right)}{\partial w_{ab}^l} \\ &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \cdot y_{(is-a)(js-b)}^{l-1}\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial b^l} &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial x_{ij}^l}{\partial b^l} \\ &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \cdot \frac{\partial \left(\sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} w_{ab}^l \cdot y_{(is-a)(js-b)}^{l-1} + b^l \right)}{\partial b^l} \\ &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l}\end{aligned}$$

Так как все операции производятся над матрицами, было решено использовать специализированную библиотеку для работы с массивами –

numpy, так как она предоставляет удобные функции для произведения операций над матрицами.

Инструкция для запуска программы

Для запуска программы на операционной системе Kubuntu 18.04.2, необходимо загрузить и установить установщик пакетов Python3 – pip3 (для этого потребуются права администратора):

sudo apt install python3-pip

Далее, установить с помощью установщика пакетов pip3 используемую программой библиотеку для оперирования матрицами – numpy [3]:

pip3 install numpy

python3 main.py

Для запуска программы на операционной системе Windows 10, необходимо установить с помощью установщика пакетов pip используемую программой библиотеку для для оперирования матрицами – numpy:

pip install numpy

python main.py

В случае если потребуется заново обучить ИНС, то потребуется установить библиотеки OpenCV [4] для работы с изображениями и matplotlib для отрисовки графиков обучения.

Инструкция по работе с программой

После запуска программы потребуется ввести изображение в виде матрицы 8x8 (цифры в каждой строки должны быть введены через пробел), после нажатия клавиши Enter будут выведены вероятности принадлежности введенного изображения к классам.

Результаты выполнения программы

Создание обучаещей и тестовой выборок

Для предварительной загрузки и обработки изображений для составления датасета была использована библиотека OpenCV

```
# -*- coding: utf-8 -*-

import numpy as np
import os  # для работы с файлами на диске
import random
import glob
import cv2 as cv
import time

random.seed(time.time())

def train_test_split(x, y, percent):
    num = int(percent * len(x))
    trainx = x[:len(x) - num]
    trainy = y[:len(y) - num]
    testx = x[len(x) - num:]
    testy = y[len(y) - num:]
    to_file = (np.array(trainx), np.array(testx), np.array(trainy),
np.array(testy))
    np.save("train_test_data.npy", {'data': to_file})
    return to_file

def make_train_test(num):
    np.random.seed(int(time.time()))
    data = []
    labels = []
    imagePathsMain = sorted(list(glob.glob("data/[1, 2, 3, 4]/*.jpg",
recursive=True)))
    imagePathsTemp = sorted(list(glob.glob("data\\5\\*.jpg",
recursive=True)))
    # цикл по изображениям
    for i in range(num):
        # загружаем изображение, меняем размер на 8x8 пикселей (без
учёта соотношения сторон)
        # добавляем в список
        # переводим изображение в черно-белое
        path = random.choice(imagePathsMain)
        image = cv.imread(path)
        gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
        gray = gaussian_blur(gray)
        data.append(gray)

        # извлекаем метку класса из пути к изображению и обновляем
        # список меток
        label = path.split(os.path.sep)[-2]
        labels.append(int(label))
    for i in range(4):
        # загружаем изображение, меняем размер на 8x8 пикселей (без
учёта соотношения сторон)
        # добавляем в список
        # переводим изображение в черно-белое
        for path in imagePathsTemp:
            image = cv.imread(path)
            gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

```
                gray = gaussian_blur(gray)
                data.append(gray)

                # извлекаем метку класса из пути к изображению и обновляем
                # список меток
                label = path.split(os.path.sep)[-2]
                labels.append(int(label))
        # масштабируем интенсивности пикселей в диапазон[0, 1]
        data = np.array(data, dtype="float")
        data = data.reshape(data.shape[0], 8, 8)
        data /= 255.0
        labels = np.array(labels, dtype="int")
        list_storage = zip(data, labels)
        list_storage = list(list_storage)
        np.random.shuffle(list_storage)
        data, labels = zip(*list_storage)
        # разбиваем данные на обучающую и тестовую выборки, используя 75%
данных
        # для обучения и оставшиеся 25% для тестирования
        return train_test_split(data, labels, 0.25)

    def gaussian_blur(img):
        kernel = np.array([[1.0, 2.0, 1.0], [2.0, 4.0, 2.0], [1.0, 2.0,
1.0]])
        kernel = kernel / np.sum(kernel)
        arraylist = []
        for y in range(3):
            temparray = np.copy(img)
            temparray = np.roll(temparray, y - 1, axis=0)
            for x in range(3):
                temparray_X = np.copy(temparray)
                temparray_X = np.roll(temparray_X, x - 1, axis=1) *
kernel[y, x]
                arraylist.append(temparray_X)

        arraylist = np.array(arraylist)
        arraylist_sum = np.sum(arraylist, axis=0)
        return arraylist_sum

    make_train_test(1000)
```

Результаты обучения нейросети представлены на рисунке 8.

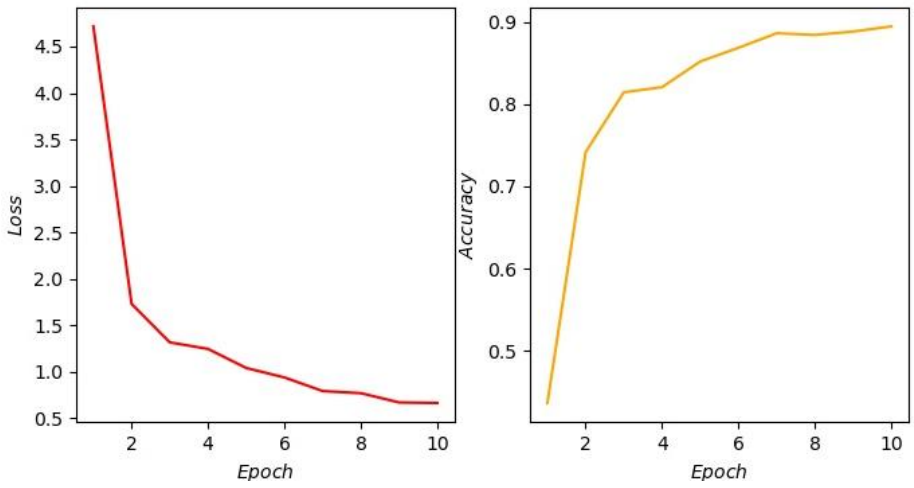

Рисунок 8 - История обучения ИНС

Проверка ИНС на образцах

Результат выполнения при входном образце «B» представлен на рисунке 9.

```
Модель загружена
Введите изображение (8x8) построчно:
0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1
Вероятность совпадения с образцами:
B: 0.998272300287081
P: 2.6243533980129757e-10
R: 0.00044425078424377114
S: 0.00016319381149241494
Ни на что не похоже: 0.001120254854747501
```

Рисунок 9 - Образ «B»

Результат выполнения при входном образце «P» представлен на рисунке 10.

```
Модель загружена
Введите изображение (8x8) построчно:
0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
Вероятность совпадения с образцами:
B: 4.3421284387782535e-27
P: 0.9999999979166619
R: 1.4092344445928683e-09
S: 3.180267951168931e-31
Ни на что не похоже: 6.741035169078279e-10
```

Рисунок 10 - Образ «P»

Результат выполнения при входном образце «R» представлен на рисунке 11.

```
Модель загружена
Введите изображение (8x8) построчно:
0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1
0 1 1 1 1 1 0 1
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
Вероятность совпадения с образцами:
B: 1.2121808171363025e-15
P: 0.029076299392928915
R: 0.9709028143488294
S: 3.300992334920654e-24
Ни на что не похоже: 2.088625824059657e-05
```

Рисунок 11 - Образ «R»

Результат выполнения при входном образце «S» представлен на рисунке 12.

```
Модель загружена
Введите изображение (8x8) построчно:
1 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
1 0 0 0 0 0 0 1
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
1 0 0 0 0 0 0 1
Вероятность совпадения с образцами:
B: 1.5189762475583056e-11
P: 9.679295768676975e-29
R: 5.747927767643895e-17
S: 0.9999999999848102
Ни на что не похоже: 9.2550549866127e-17
```

Рисунок 12 - Образ «S»

Результат выполнения при входном образце «?» представлен на рисунке 13.

```
Модель загружена
Введите изображение (8x8) построчно:
1 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0
1 1 1 1 1 1 0 1
1 1 1 1 1 0 1 1
1 1 1 1 0 1 1 1
1 1 1 1 0 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 0 1 1 1
Вероятность совпадения с образцами:
B: 2.508953944931814e-19
P: 1.0181394925282504e-09
R: 9.123949910038735e-12
S: 2.769619763456082e-13
Ни на что не похоже: 0.9999999989724596
```

Рисунок 13 - Образ «?»

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Определение основных параметров сверточной. - URL: https://m.habr.com/ru/company/ods/blog/344008/ (дата обращения 2020-04-30).

2. Вывод формул для обратного распространения ошибки. - URL: https://m.habr.com/ru/company/ods/blog/344116/ (дата обращения 2020-04-30).

3. Numpy Documentation. – URL: https://numpy.org/index.html (дата обращения 2020-04-20).

4. OpenCV Documentation. – URL: https://docs.opencv.org/2.4/index.html (дата обращения 2020-04-20).

ПРИЛОЖЕНИЕ А

Исходные тексты программы представлены в файлах main.py (главный файл программы), make_data.py (подготовка датасета), model.py (функции описывающие слои ИНС).