МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ МОРСКОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
 (СПБГМТУ)

| ФАКУЛЬТЕТ ЦИФРОВЫХ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ  КАФЕДРА КИБЕРФИЗИЧЕСКИХ СИСТЕМ |
| --- |

ЛАБОРАТОРНАЯ РАБОТА №5+

ВАРИАНТ 8

«РАБОТА С ДАТАСЕТОМ»

|  | |
| --- | --- |
|  | Выполнил  студент группы 20221 | | |
|  | Лаптев Иван Александрович | | |
|  | Проверила | | |
|  | Кайнова Татьяна Денисовна | | |
| Санкт-Петербург  2024 год | | |

**Оглавление**

[Цели работы…………………………………………………………………………………...](#_30j0zll) 3

[Результат работы………………………………………………..……………………………](#_1fob9te) 4

Листинг кода……[……………………………………………………………………...……...](#_3znysh7) 8

# **ЦЕЛИ РАБОТЫ**

В ходе выполнения лабораторной работы необходимо написать программу, которая выводит пять вариантов написания номера варианта по списку группы. Вывод необходимо осуществлять цифрами из набора MNIST.

Также необходимо разобраться в приведённом коде и заполнить комментарии, поясняющие работу соответствующего блока кода везде, где встречается символ «#».

# **РЕЗУЛЬТАТЫ РАБОТЫ**

Вариант работы 8, а значит нам нужно вывести цифру 8. Обращаемся к датасету MNIST и получаем следующие идентификаторы для цифры 0: 17, 31, 41, 46, 55. Листинг 1 содержит программный код, который позволяет загрузить и обработать данные из датасета MNIST, настроить нейронную сеть, а также провести обучение и визуализацию результатов

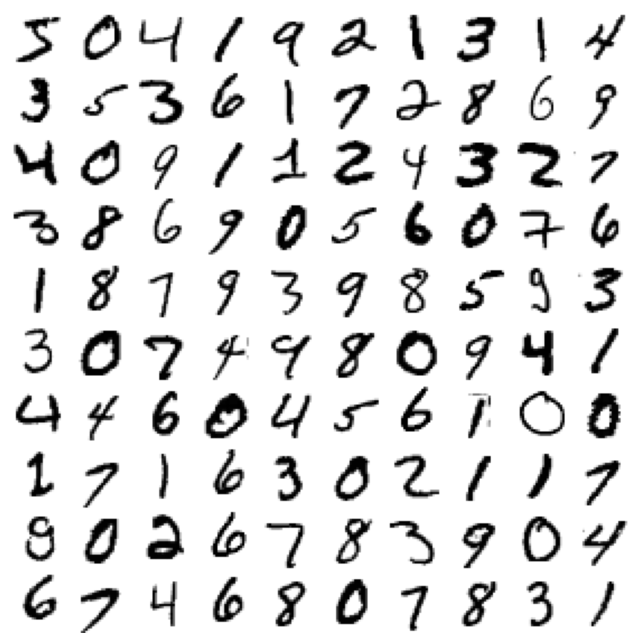


Рисунок 1 – Датасет MNIST

Полученные изображения:

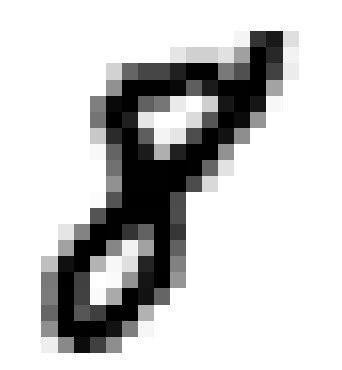


Рисунок 2 – Первое изображение

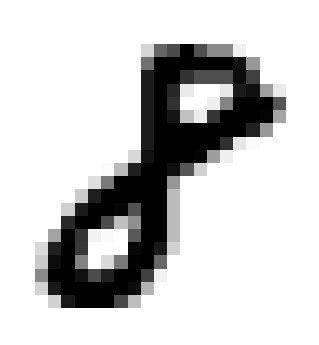


Рисунок 3 – Второе изображение

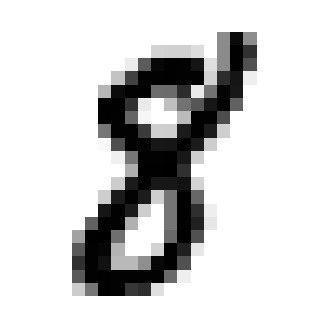


Рисунок 4 – Третье изображение

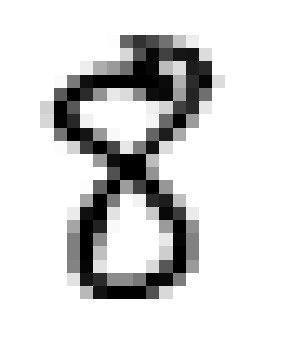


Рисунок 5 – Четвёртое изображение

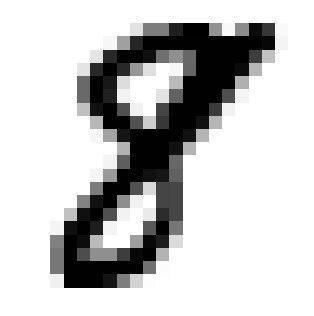


Рисунок 6 – Пятое изображение

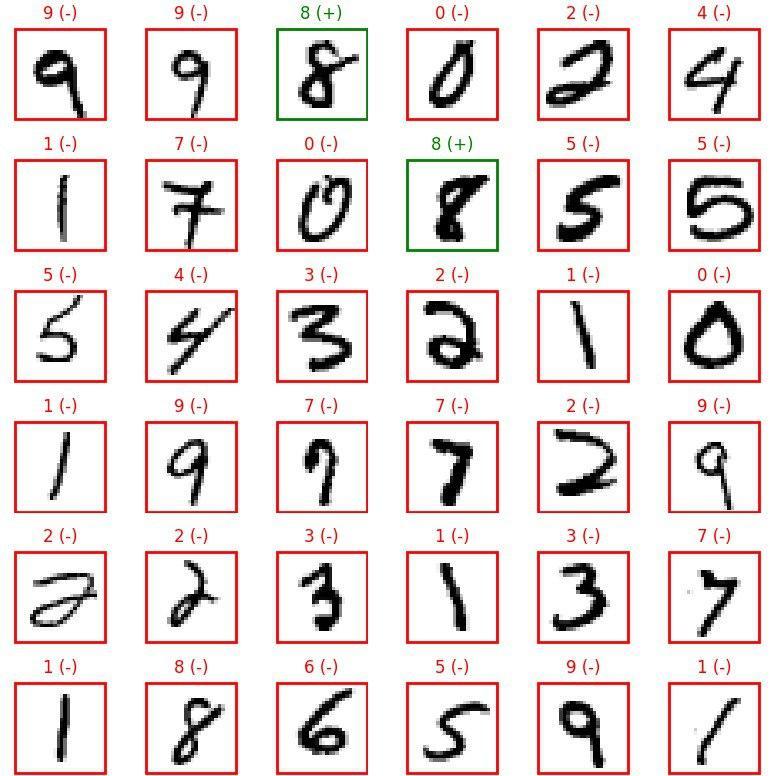


Рисунок 7 - Итоговое изображение

# 

Листинг 1 – Код с комментариями

import sys # Импортируем модуль sys для работы с параметрами системы

assert sys.version\_info >= (3, 5) # Проверяем, что версия Python не ниже 3.5

IS\_COLAB = "google.colab" in sys.modules # Проверяем, запущен ли код в Google Colab

IS\_KAGGLE = "kaggle\_secrets" in sys.modules # Проверяем, запущен ли код на платформе Kaggle

import sklearn # Импортируем библиотеку Scikit-learn

assert sklearn.\_\_version\_\_ >= "0.20" # Проверяем, что версия scikit-learn не ниже 0.20

import numpy as np # Импортируем библиотеку NumPy для работы с массивами

import os # Импортируем библиотеку os для работы с операционной системой

np.random.seed(42) # Устанавливаем фиксированное начальное значение для генератора случайных чисел (для воспроизводимости)

import matplotlib as mpl # Импортируем библиотеку Matplotlib для визуализации данных

import matplotlib.pyplot as plt # Импортируем модуль для рисования графиков

mpl.rc('axes', labelsize=14) # Настройка размеров шрифтов меток на осях графиков

mpl.rc('xtick', labelsize=12) # Настройка размера шрифта для меток на оси X

mpl.rc('ytick', labelsize=12) # Настройка размера шрифта для меток на оси Y

PROJECT\_ROOT\_DIR = "." # Устанавливаем текущую директорию как корень проекта

CHAPTER\_ID = "classification" # Идентификатор текущей главы проекта

IMAGES\_PATH = os.path.join(PROJECT\_ROOT\_DIR, "images", CHAPTER\_ID) # Путь для сохранения изображений

os.makedirs(IMAGES\_PATH, exist\_ok=True) # Создаем директорию для изображений, если она не существует

def save\_fig(fig\_id, tight\_layout=True, fig\_extension="png", resolution=300): # Функция для сохранения изображения

path = os.path.join(IMAGES\_PATH, fig\_id + "." + fig\_extension) # Формируем полный путь к файлу изображения

print("Saving figure", fig\_id) # Выводим сообщение о сохранении изображения

if tight\_layout: # Если tight\_layout=True, применяем его к графику

plt.tight\_layout()

plt.savefig(path, format=fig\_extension, dpi=resolution) # Сохраняем график в файл

from sklearn.datasets import fetch\_openml # Импортируем функцию для загрузки данных с OpenML

mnist = fetch\_openml('mnist\_784', version=1, as\_frame=False) # Загружаем датасет MNIST

mnist.keys() # Выводим ключи, доступные в загруженном датасете

X, y = mnist["data"], mnist["target"] # Разделяем данные на признаки (X) и целевую переменную (y)

X.shape # Получаем размерность матрицы данных

y.shape # Получаем размерность массива меток

28 \* 28 # Размер изображения (28x28 пикселей)

import matplotlib as mpl # Импортируем Matplotlib для рисования

import matplotlib.pyplot as plt # Импортируем функции для работы с графиками

some\_digit = X[17] # Выбираем 17-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем его в изображение размером 28x28

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

save\_fig("some\_digit\_plot") # Сохраняем изображение

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[31] # Выбираем 31-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[41] # Выбираем 41-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[46] # Выбираем 46-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

some\_digit = X[55] # Выбираем 55-й элемент из данных

some\_digit\_image = some\_digit.reshape(28, 28) # Преобразуем в изображение

plt.imshow(some\_digit\_image, cmap=mpl.cm.binary) # Отображаем изображение

plt.axis("off") # Отключаем оси

plt.show() # Показываем изображение

y[0] # Получаем метку первого элемента

y = y.astype(np.uint8) # Преобразуем метки в тип данных uint8

def plot\_digit(data): # Функция для отображения одной цифры

image = data.reshape(28, 28) # Преобразуем данные в изображение 28x28

plt.imshow(image, cmap=mpl.cm.binary, interpolation="nearest") # Отображаем изображение

plt.axis("off") # Отключаем оси

def plot\_digits(instances, images\_per\_row=10, \*\*options): # Функция для отображения нескольких цифр

size = 28 # Размер изображений (28x28)

images\_per\_row = min(len(instances), images\_per\_row) # Ограничиваем количество изображений в ряду

n\_rows = (len(instances) - 1) // images\_per\_row + 1 # Вычисляем количество рядов для изображений

n\_empty = n\_rows \* images\_per\_row - len(instances) # Вычисляем, сколько пустых изображений нужно добавить

padded\_instances = np.concatenate([instances, np.zeros((n\_empty, size \* size))], axis=0) # Добавляем пустые изображения

image\_grid = padded\_instances.reshape((n\_rows, images\_per\_row, size, size)) # Формируем сетку изображений

big\_image = image\_grid.transpose(0, 2, 1, 3).reshape(n\_rows \* size, images\_per\_row \* size) # Переставляем оси и получаем большое изображение

plt.imshow(big\_image, cmap=mpl.cm.binary, \*\*options) # Отображаем сетку изображений

plt.axis("off") # Отключаем оси

plt.figure(figsize=(9,9)) # Создаем фигуру для отображения

example\_images = X[:100] # Выбираем первые 100 изображений

plot\_digits(example\_images, images\_per\_row=10) # Отображаем изображения в виде сетки

save\_fig("more\_digits\_plot") # Сохраняем изображение

plt.show() # Показываем изображение

import time

from sklearn.datasets import fetch\_openml

mnist = fetch\_openml('mnist\_784', version=1, as\_frame=False)

# Данные, необходимые для обучения

X, y = mnist["data"], mnist["target"]

x\_train,y\_train=X,y

# Задаём исходные данные

num = 8

np.random.seed(5)

# Создаём класс

class NeuraNetwork:

def \_\_init\_\_(self, x, y, t = {0:4, 1:4, 2:4, 3:1}):

self.y\_train = y

self.x\_train = np.insert(x, 0, 1, axis=1)

self.top = t

self.learning\_rate = 0.001

self.index\_list = list(range(len(self.x\_train)))

self.n\_y = [[0]\*value for value in self.top.values()]

self.n\_error = [[0]\*value for value in self.top.values()]

self.n\_w = []

self.generate\_w()

def generate\_w(self):

for i in self.top:

count = len(self.x\_train[0]) if i == 0 else self.top[i-1]+ 1

shape = (self.top[i], count)

w = np.random.uniform(-1, 1, shape)

self.n\_w.append(w)

def forward\_pass(self, x, key = False):

# Прямой проход

for i in range(max(self.top)+1):

if i == 0:

for j in range(self.top[i]):

self.n\_y[i][j] = np.tanh(np.dot(self.n\_w[i][j], x))

elif i == (max(self.top)):

for j in range(self.top[i]):

n\_inputs = np.array([1.0]+ self.n\_y[i-1])

z2 = np.dot(self.n\_w[i][j], n\_inputs)

self.n\_y[i][j] = 1.0 / (1.0 + np.exp(-z2))

else:

for j in range(self.top[i]):

self.n\_y[i][j] = np.tanh(np.dot(self.n\_w[i][j], np.array([1.0] + self.n\_y[i-1])))

if key == True:

return self.n\_y[-1][0]

def backward\_pass(self, y):

# Обратный проход

if y == str(num):

error\_prime = -(1.0 - self.n\_y[-1][0])

else:

error\_prime = -(0.0 - self.n\_y[-1][0])

derivative = self.derivative\_log()

self.n\_error[-1][0] = error\_prime \* derivative

for i in reversed(range(max(self.top))):

for j in range(self.top[i]):

sum = 0

for k in range(self.top[i+1]): sum+=self.n\_error[i+1][k] \* self.n\_w[i+1][k][j+1]

self.n\_error[i][j] = sum \* self.derivative\_tanh(i, j)

def adjust\_weights(self, x):

# Корректируем веса

for i in range(max(self.top)+1):

if i == 0: inputs = x

else: inputs = np.array([1.0] + self.n\_y[i-1])

for j in range(self.top[i]): self.n\_w[i][j] -= self.learning\_rate \* self.n\_error[i][j] \* inputs

def derivative\_log(self, i = -1, j=0): return float(self.n\_y[i][j]) \* (1.0 - float(self.n\_y[i][j]))

def derivative\_tanh(self, i, j): return 1.0 - self.n\_y[i][j]\*\*2

n = NeuraNetwork(x\_train, y\_train, {0:3, 1:2, 2:1})

startTime = time.time()

# Процесс обучения

for i in range(1):

flag = True

np.random.shuffle(n.index\_list)

for i in n.index\_list:

n.forward\_pass(n.x\_train[i])

n.backward\_pass(n.y\_train[i])

n.adjust\_weights(n.x\_train[i])

for i in range(len(n.x\_train)):

n.forward\_pass(n.x\_train[i])

if(((int(n.y\_train[i]) != num) and (float(\*n.n\_y[-1]) >= 0.5)) or ((int(n.y\_train[i]) == num) and (float(\*n.n\_y[-1]) < 0.5))):

break

endTime = time.time()

totalTime = endTime - startTime

print("Время обучения ", totalTime)

# Проверка

np.random.shuffle(n.index\_list)

lst = []

for i in n.index\_list:

out = n.forward\_pass(n.x\_train[i], key = True)

if (n.y\_train[i] == str(num) and out >= 0.5) or (n.y\_train[i] != str(num) and out < 0.5):

lst.append(1)

else:

lst.append(0)

lst = np.asarray(lst)

print("Эффективность = ", (lst.sum() / lst.size) \* 100, "%")

startTime = time.time()

fig, axes = plt.subplots(6, 6, figsize=(8, 8))

# Вывод изображения

for i in range(36):

if lst[n.index\_list[i]] >= 0.5 and n.y\_train[n.index\_list[i]] == str(num):

sign, color = "+", "green"

else:

sign, color = "-", "red"

row = i//6

col = i % 6

number = X[n.index\_list[i]].reshape(28, 28)

ax = axes[row, col]

ax.imshow(number, cmap=mpl.cm.binary)

ax.axis("off")

ax.set\_title(f"{n.y\_train[n.index\_list[i]]} ({sign})", fontsize=12)

ax.add\_patch(plt.Rectangle((0, 0), 27, 27, fill=False, edgecolor=color, linewidth=2))

ax.title.set\_color(color)

plt.tight\_layout()

plt.show()