Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Кафедра вычислительной математики и механики

**ЛАБОРАТОРНАЯ РАБОТА № 5**

**«Реализация нейронных сетей для решения задачи распознавания образов»**

Выполнила:

студентка группы ИСТ-19-2б

Семёнова А.С.

Проверил:

ассистент кафедры ВММБ

Нетбай Г.В.

Пермь, 2022

**Цель:** сформировать у студентов способность построения нейросетевой интеллектуальной системы для решения задач распознавания образов.

**Описание:** в рамках данной работы необходимо реализовать приложение, решающее задачу распознавание образов с использованием искусственных нейронных сетей. Возможный вариант реализации – многослойный персептрон с методом обратного распространения ошибки, где каждый входной сигнал соответствует пикселю образа.

Вариативность задания заключается в различной предметной

области:

1. распознавание геометрических фигур (не менее 8);
2. распознавание знаков валют (не менее 8);
3. распознавание направления стрелки, указывающей на 8 направлений сторон света (север, северо-восток и т.д.);
4. распознавание римских цифр.

**Требования:**

1. Должна быть возможность добавления, удаления, изменения элементов обучающей выборки.
2. Должна быть возможность рисования распознаваемого образа мышью компьютера.
3. Не допускается непосредственное копирование кода из открытых библиотек, реализующих алгоритмы с искусственными нейронными сетями.
4. Программный код должен соответствовать требованиям:

* код должен быть единообразно отформатирован;
* должны присутствовать комментарии.

**Используемые технологии**

1. База данных MNIST — это база данных, в которой хранятся образцы написания рукописных цифр. Она состоит из 70 тысяч картинок одинакового размера, где изображены написанные от руки цифры. Набор данных, или датасет, используют для машинного обучения - на нем программы учат распознавать числа.
2. Numpy-библиотека для работы с массивами;
3. ImageIO- библиотека для работы с изображениями;
4. Tensorflow- библиотека для построения нейронной сети.

**Модель нейронной сети**

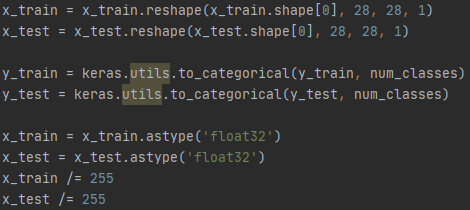
1. Реализуем модель Персептрона с использованием библиотеки Keras на языке Python. Задача будет решаться помощью распознавания рукописных цифр от 0 до 9;
2. Загрузим базу тестирований из MNIST:



1. Импортируем библиотеки для обучения Персептрона:



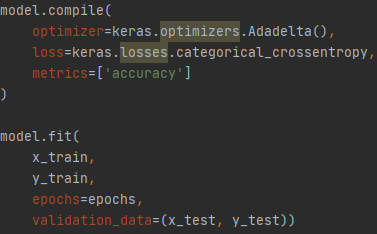
1. Приводим данные к значениям от нуля до единицы:



1. Создаем модель и добавляем слои:



1. Сборка модели:



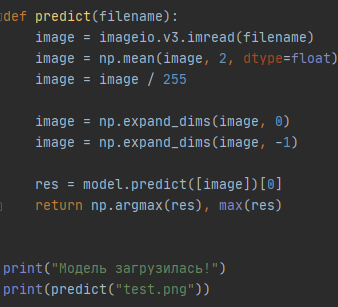
1. Сохранение модели:



1. Запуск обучение модели:



1. Вычисление значения точности модели и сохранение(загрузка):



**Обучение нейронной сети**

Обучения проводятся на базе данных MNIST, в 8 эпох. По завершению обучения и прогонки тестовых данных, точность модели составит примерно %.

**Результат работы программы**

Исходное изображение:



Результат распознавания:



**Приложение 1(код нейронной сети)**

import keras

from keras.datasets import mnist

from tensorflow.python.ops import nn

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

num\_classes = 10

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

epochs = 5

model = keras.models.Sequential([

keras.layers.Conv2D(

input\_shape=(28, 28, 1),

filters=32,

kernel\_size=(5, 5),

padding='same',

activation='relu'

),

keras.layers.MaxPool2D(pool\_size=(2, 2)),

keras.layers.Conv2D(

filters=64,

kernel\_size=(5, 5),

padding='same',

activation='relu'

),

keras.layers.MaxPool2D(pool\_size=(2, 2)),

keras.layers.Flatten(),

keras.layers.Dense(1024, activation=nn.relu),

keras.layers.Dropout(0.2),

keras.layers.Dense(10, activation=nn.softmax)

])

model.compile(

optimizer=keras.optimizers.Adadelta(),

loss=keras.losses.categorical\_crossentropy,

metrics=['accuracy']

)

model.fit(

x\_train,

y\_train,

epochs=epochs,

validation\_data=(x\_test, y\_test))

model.save('fit\_neuronet.h5')

print("Модель сохранилась!")

**Приложение 2 (код распознавания нейронной сети)**

import imageio

import numpy as np

from keras.models import load\_model

model = load\_model(fit\_neuronet.h5)

def predict(filename):

image = imageio.v3.imread(filename)

image = np.mean(image, 2, dtype=float)

image = image / 255

image = np.expand\_dims(image, 0)

image = np.expand\_dims(image, -1)

res = model.predict([image])[0]

return np.argmax(res), max(res)

print ("Модель загрузилась!")

print (predict("test.png"))