МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание рукописных символов

Студентка гр. 7382 _____ Дерябина П.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы

Реализовать классификацию черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9) на датасете MNIST.

Порядок выполнения работы

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющая загружать изображение пользователи и классифицировать его

Требования к выполнению задания

- Найти архитектуру сети, при которой точность классификации будет не менее 95%
- Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
- Написать функцию, которая позволит загружать пользовательское изображение не из датасета

Ход работы

Код программы, реализующий структуру ИНС, приведен в приложении А.

Будем последовательно рассматривать три оптимизатора: Adam, SGD, RMSprop с разными параметрами и выберем лучшую модель. На рис. 1 пример метрик для оптимизатора Adam с параметром learning_rate = 0.001, на рис. 2 – с параметром learning_rate = 0.05

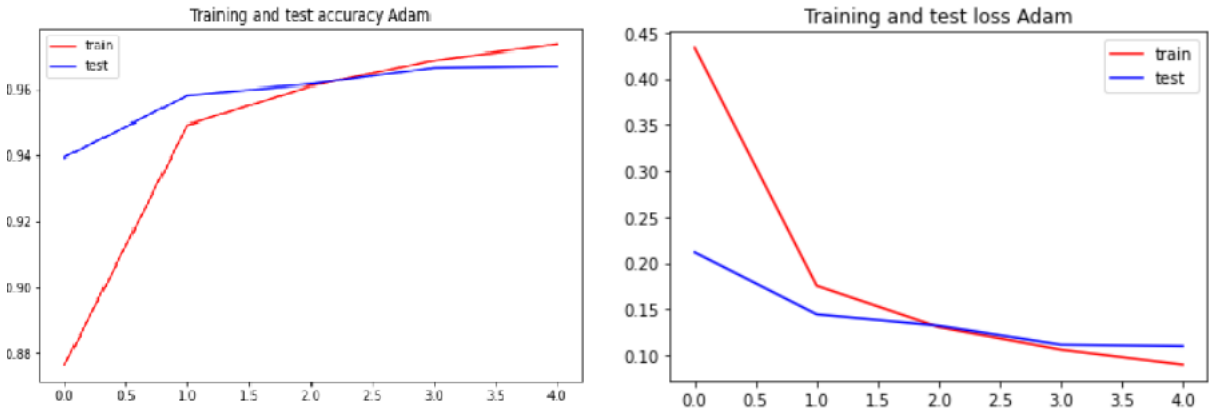

Рисунок 1 – Оптимизатор Adam, learning_rate = 0.001

Необходимая точность (не менее 95%) достигнута, получилось посмотрим на тот же оптимизатор с learning_rate = 0.05.

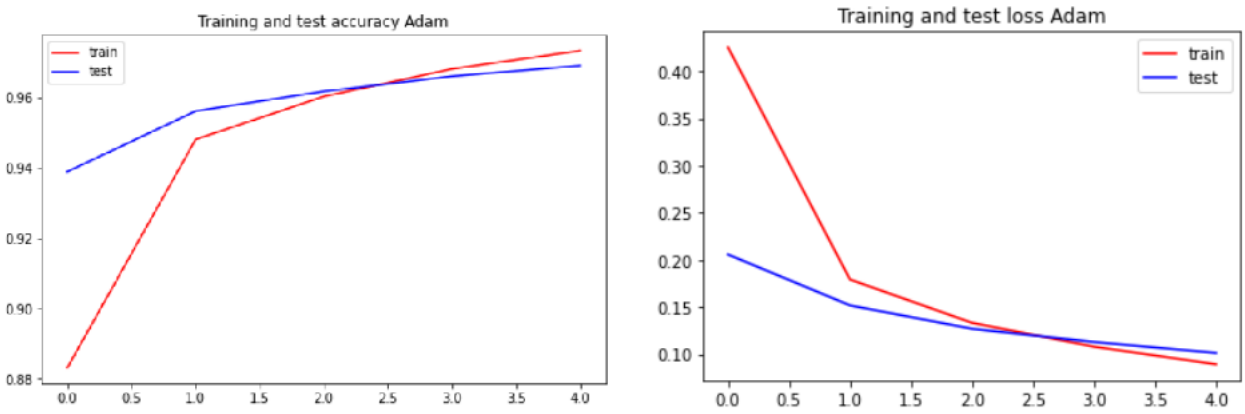

Рисунок 2 – Оптимизатор Adam, learning_rate = 0.05

Существенных различий с предыдущей моделью нет.

На рис. 3 пример метрик для оптимизатора SGD с параметром learning_rate = 0.01, на рис. 4 – с параметром learning_rate = 0.5

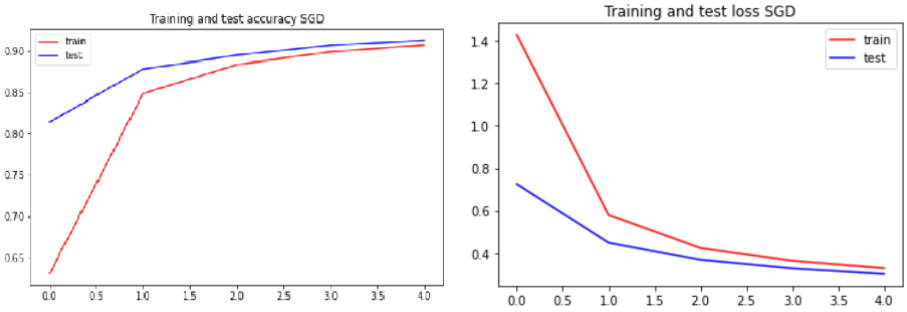

Рисунок 3 – Оптимизатор SGD, learning_rate = 0.01

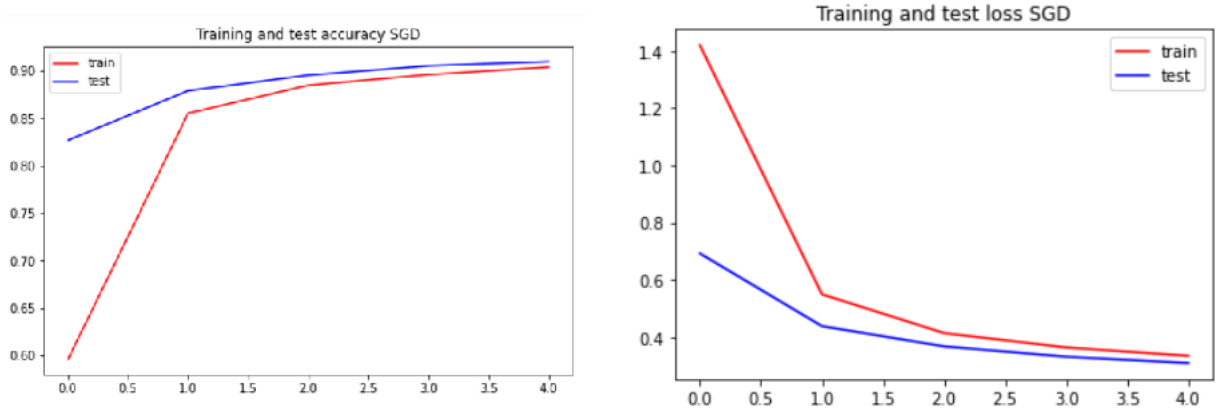

Рисунок 3 – Оптимизатор SGD, learning_rate = 0.5

Необходимой точности (не менее 95%) достичь не удалось

На рис. 5 пример метрик для оптимизатора RMSprop с параметром learning_rate = 0.001, на рис. 6 – с параметром learning_rate = 0.05

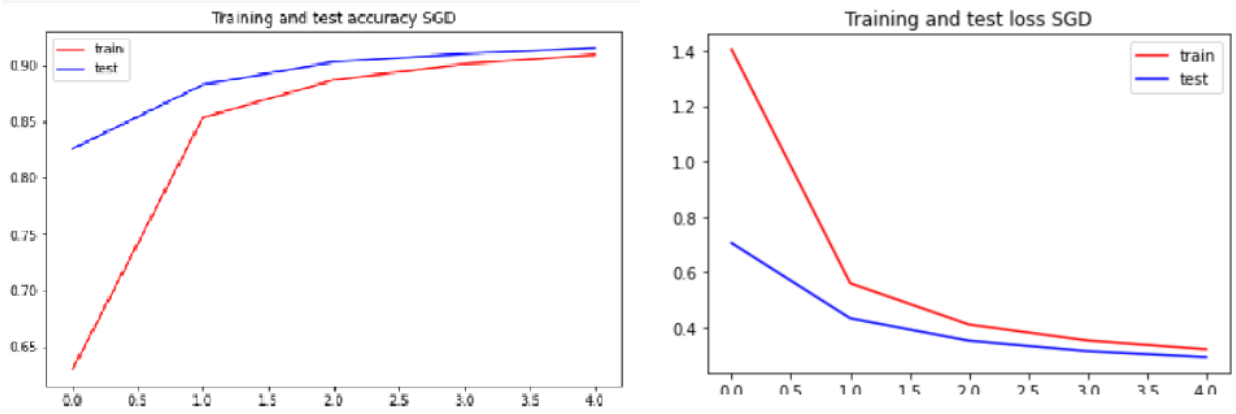

Рисунок 5 – Оптимизатор RMSprop, learning_rate = 0.001

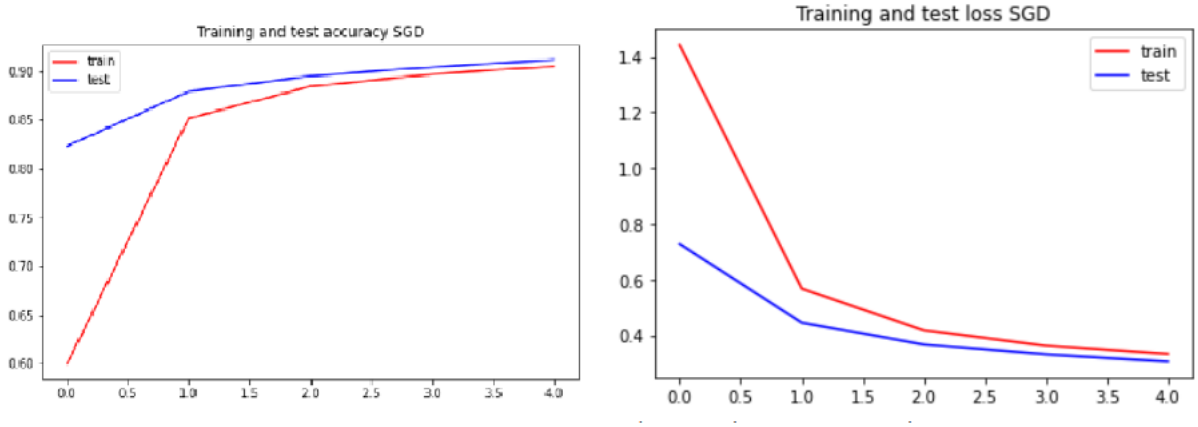

Рисунок 5 – Оптимизатор RMSprop, learning_rate = 0.05

В двух моделях с оптимизатором RMSprop не достигается нужной точности

Вывод

В ходе работы были изучены особенности работы таких оптимизаторов, как Adam, SGD, RMSprop. Была разработана модель нейронной сети, наибольшая точность (0.97205) была достигнута на модели с оптимизатором RMSprop и параметрами по умолчанию.

ПРИЛОЖЕНИ А
ИСХОДНЫЙ КОД

```
from keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import optimizers
from PIL import Image
import numpy
import matplotlib.pyplot as plt
from keras.optimizers import SGD
from keras.optimizers import  RMSprop
from keras.optimizers import  Adam

def loadImg(path):
    return numpy.asarray(Image.open(path))


(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

train_images = train_images / 255.0
test_images = test_images / 255.0

model = Sequential()
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))


def getModel(optimizer, name):
    model.compile(optimizer=name, loss='categorical_crossentropy', metrics=[
'accuracy'])
    h = model.fit(train_images, train_labels, epochs=5, verbose=0, batch_siz
e=128, validation_data=(test_images, test_labels))

    print((h.history['accuracy']), h.history['val_accuracy'])
    plt.figure(1, figsize=(8, 5))
    plt.title('Training and test accuracy ' + name)
    plt.plot(h.history['accuracy'], 'r', label='train')
    plt.plot(h.history['val_accuracy'], 'b', label='test')
    plt.legend()
    plt.show()
    plt.clf()

    plt.figure(1, figsize=(8, 5))
    plt.title('Training and test loss ' + name)
```

```
    plt.plot(h.history['loss'], 'r', label='train')
    plt.plot(h.history['val_loss'], 'b', label='test')
    plt.legend()
    plt.show()
    plt.clf()

opt = Adam(learning_rate=0.05)
getModel(opt, 'Adam')
```