МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание рукописных символов»

Студентка гр. 7382 Лящевская А.П.

Преподаватель Жукова Н. А.

Санкт-Петербург

2020

Цели.

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

Задачи.

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

Выполнение работы.

1. Найти архитектуру с точностью выше 95%

   Архитектура :

   - с 4-мя слоями:
     (a) *Flatten()*,
     (b) *Dense(64, activation='relu')*,
     (c) *Dense(32, activation='relu')*,
     (d) *Dense(10, activation='softmax')*,
   - оптимизатором – *adam*,
   - *batch_size* = 128,
   - функцией потерь – *categorical_crossentropy*,
   - количеством эпох = 5

   имеет точность ~ 97%.

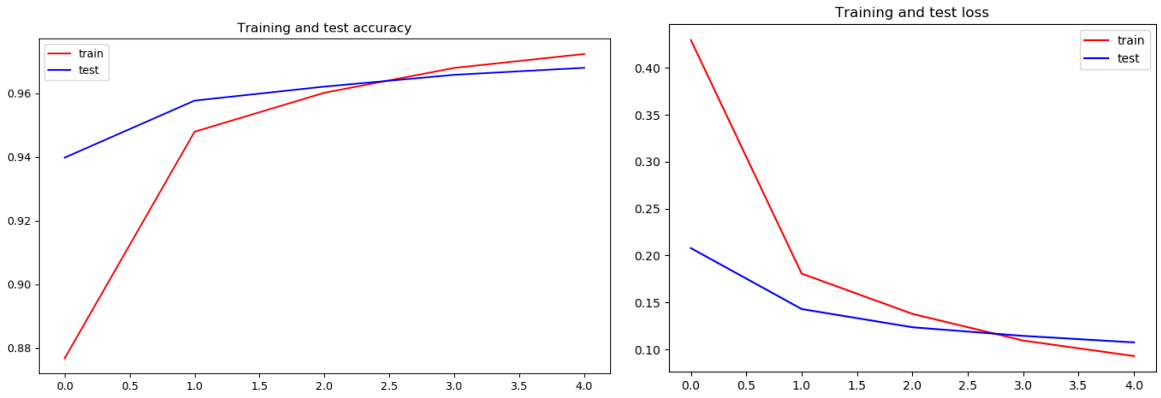


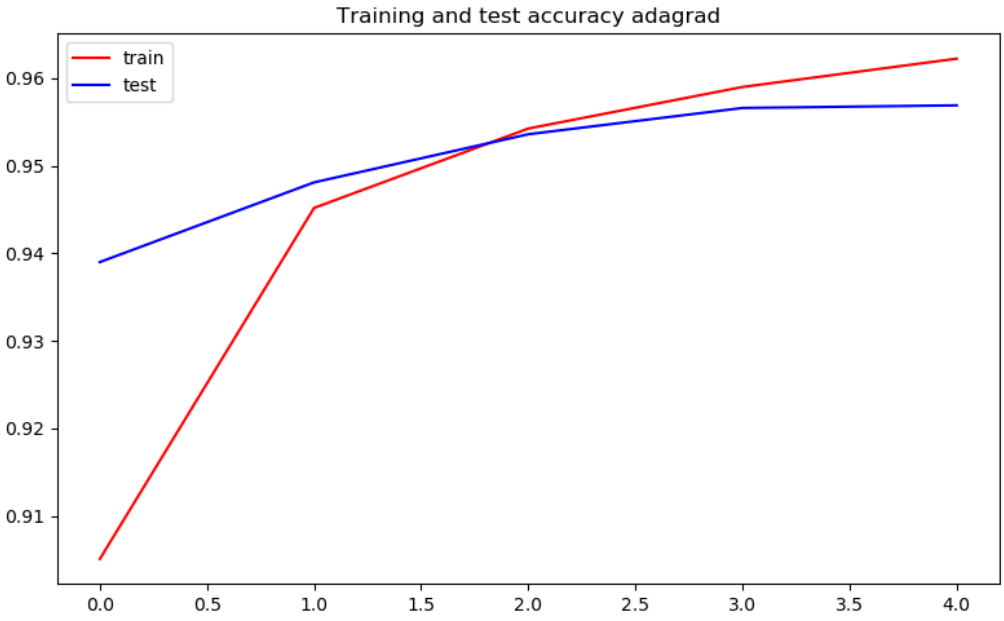
Рисунок 1 – графики точности и потерь для данной архитектуры.

2. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения.

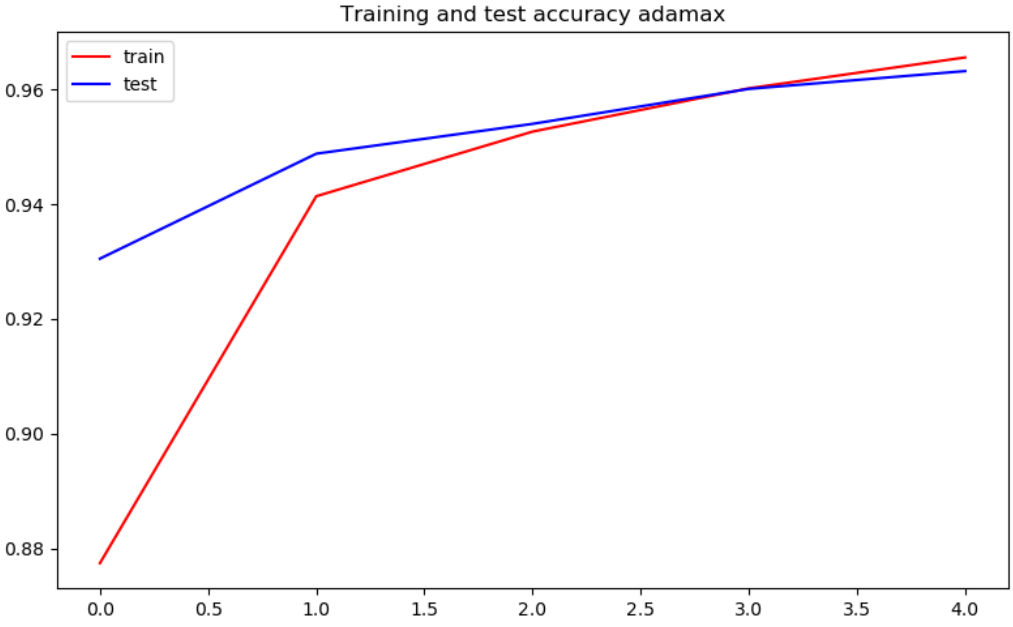
Посмотрим на результаты разных оптимизаторов.

- Оптимизатор – *adam* :

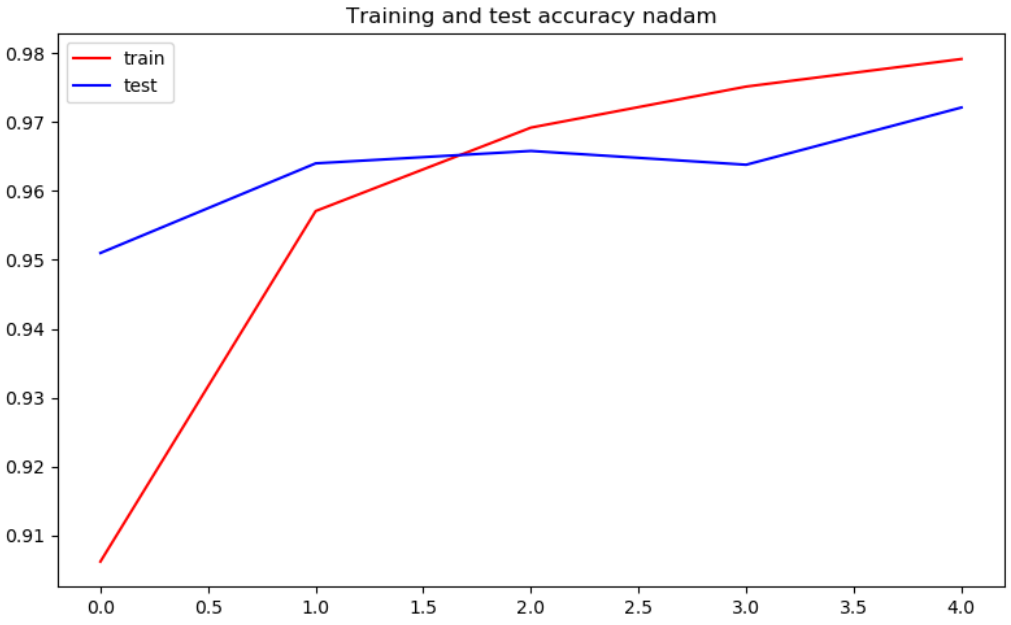
- Оптимизатор – *adagrad* :



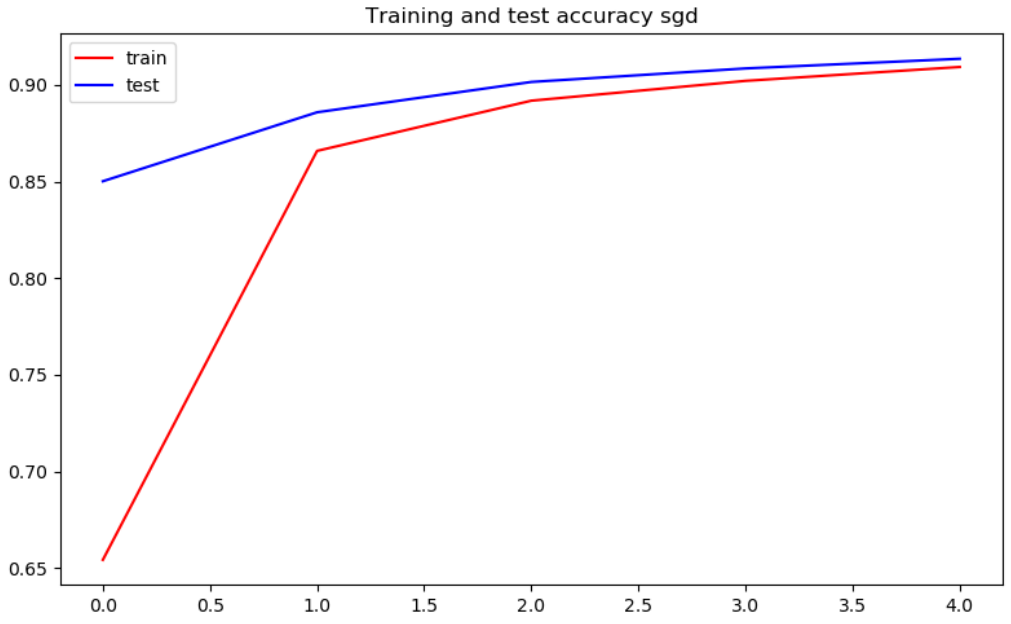
- Оптимизатор – *adamax* :



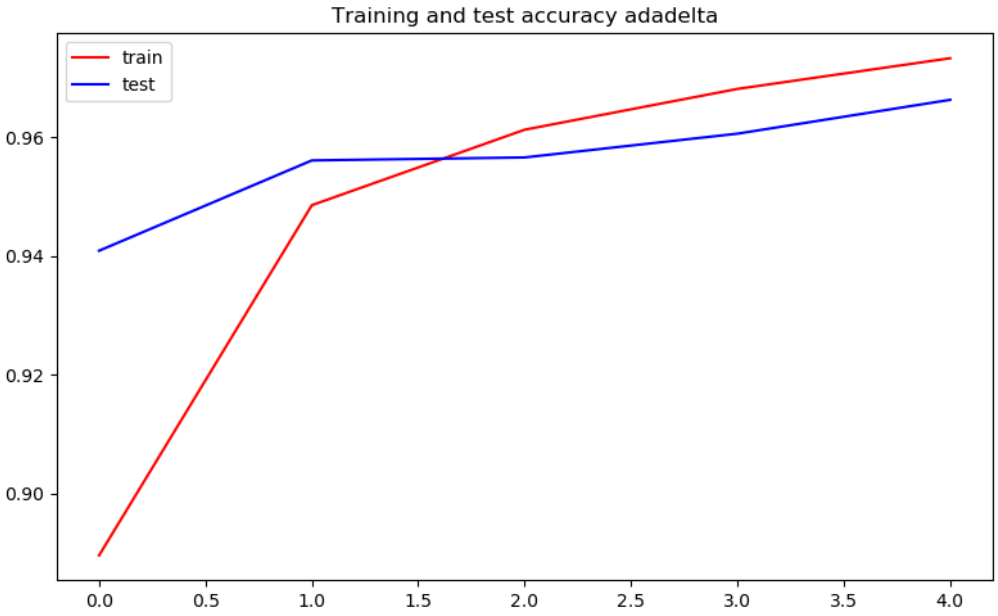
- Оптимизатор – *nadam* :



- Оптимизатор – *sgd* :



- Оптимизатор – *adadelta* :



Здесь мы видим, что все оптимизаторы показали примерно одинаковый результат, кроме *SDG*, у которого он значительно хуже.

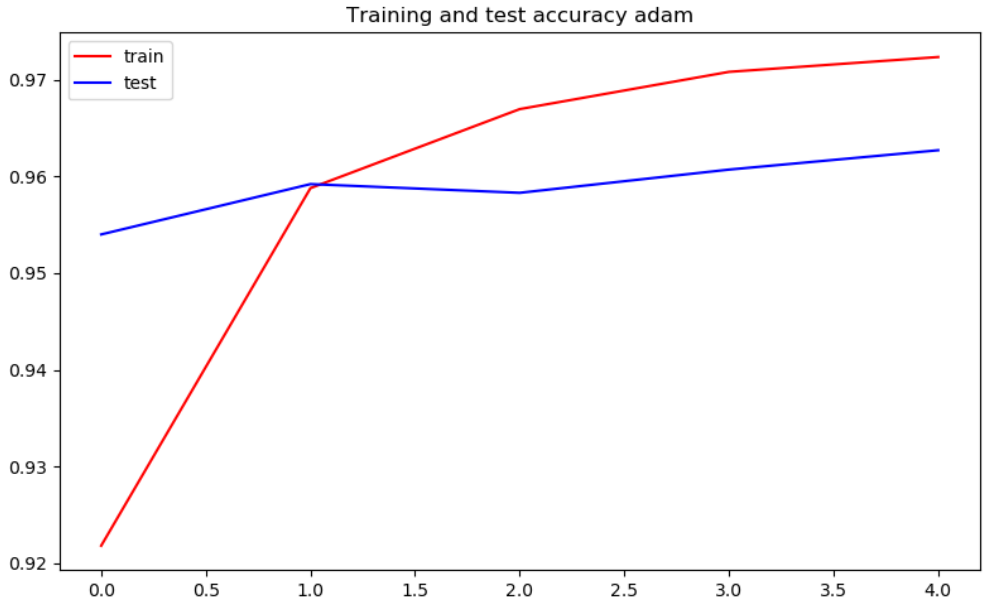
Далее, проверим влияние изменения параметра скорости обучения (*lr*) на обучение у оптимизатора *adam*.

- *Lr* = 0.0001

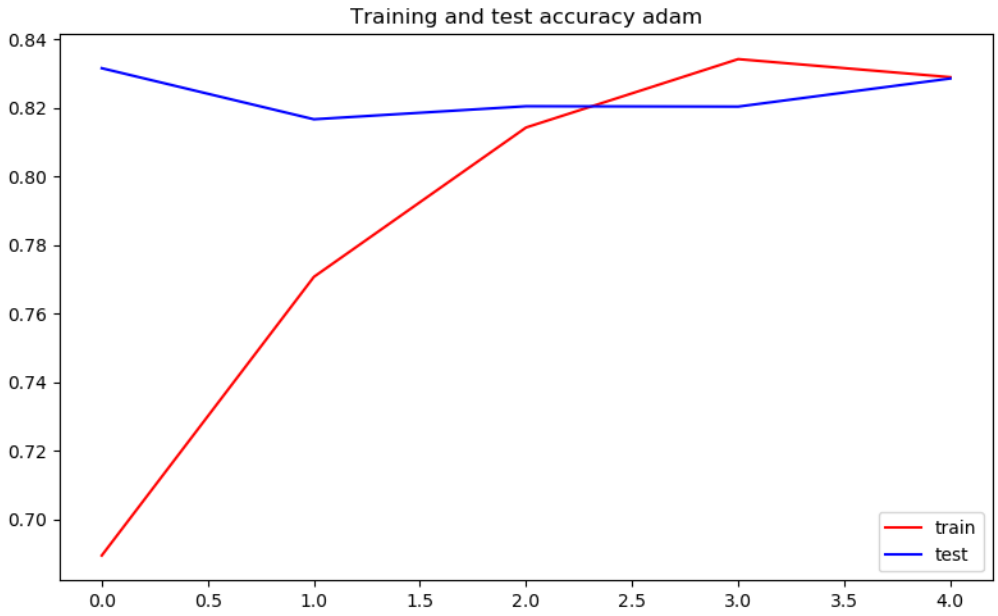
- *Lr* = 0.001

- $Lr = 0.005$

- $Lr = 0.01$



- *Lr* = 0.5



Как видно, при отклонении значения параметра *lr* от стандартного(0.001) обучаемость сети становится хуже.

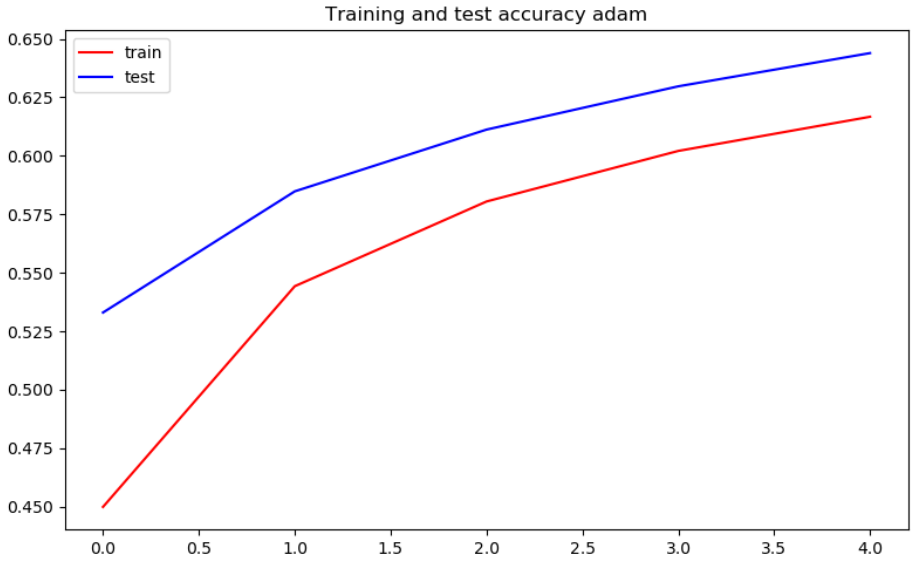
Теперь, проверим влияние еще одного параметра: *decay* (коэффициент убывания скорости обучения) на том же оптимизаторе *adam*.

- *Decay* = 0.0

- *Decay* = 0.05

- *Decay* = 0.1

- *Decay* = 0.5



- *Decay* = 2

Изменение параметра *Decay* не принесло хороших результатов.

3. Написать функцию, которая позволит загружать пользовательское изображение не из датасета.

```
def openImage(path):
    return numpy.asarray(Image.open(path))
```

Вывод.

В ходе выполнения данной работы была создана сеть для распознавания рукописных символов, а также исследованы разные оптимизаторы в обучении сети. Лучше всего в данной задаче себя показал оптимизатор *Adam* со стандартными параметрами, с лучшей точностью ~97%.

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ

КОД ПРОГРАММЫ

```
from keras.layers import Dense, Activation,
Flatten, Dropout
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


def openImage(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 compile_fit_print(optimizer, name):
    model.compile(optimizer=optimizer,
loss='categorical_crossentropy',
metrics=['accuracy'])
    H = model.fit(train_images, train_labels,
epochs=5, batch_size=128,
validation_data=(test_images, test_labels))

    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()


compile_fit_print(optimizers.Adam(), 'adam')
# compile_fit_print('adagrad')
# compile_fit_print('adamax')
# compile_fit_print('nadam')
# compile_fit_print('sgd')
# compile_fit_print('adadelta')
```