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»

Студент гр. 7383 _____ Левкович Д.В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

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

Порядок выполнения работы.

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

Выполнение работы.

0. Исходный код представлен в приложении А.
1. Выберем модель сети: один слой с 256 нейронами и функцией активации relu и второй с 10 нейронами и функцией активации softmax. В качестве оптимизатора будет выступать оптимизатор Adam. Количество эпох – 5, batch_size – 128, learning_rates по умолчанию.

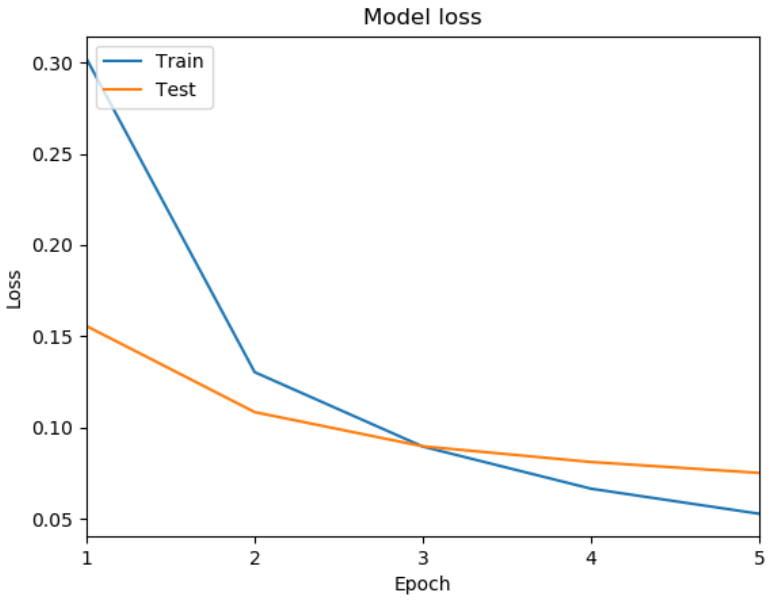

Рисунок 1 - график ошибок

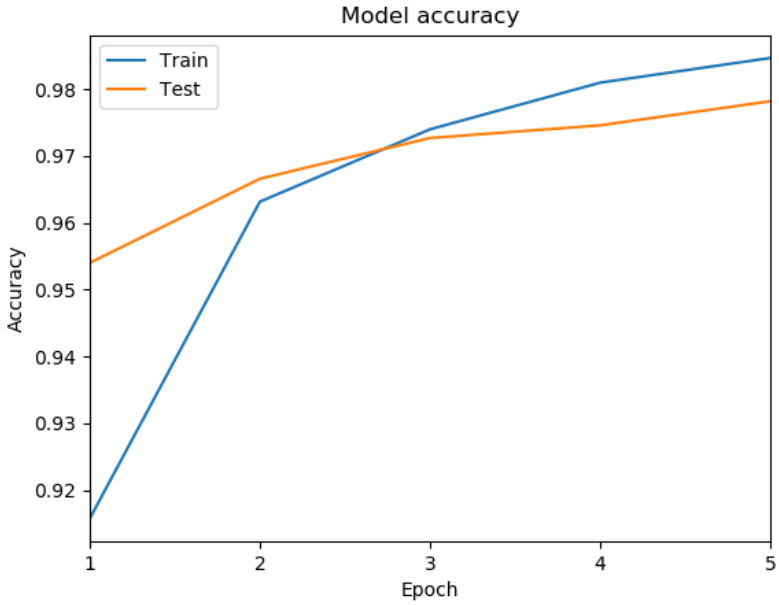

Рисунок 2 - график точности

Как видно из рис.2 достигнута точность не менее 0.97.

2. Проверим другие оптимизаторы с различными параметрами learning_rate. Проверим такие оптимизаторы: Adadelta, Adam, RMSprop, SDG, Adagrad с различными learning_rate (от 0.1 до 0.0001).

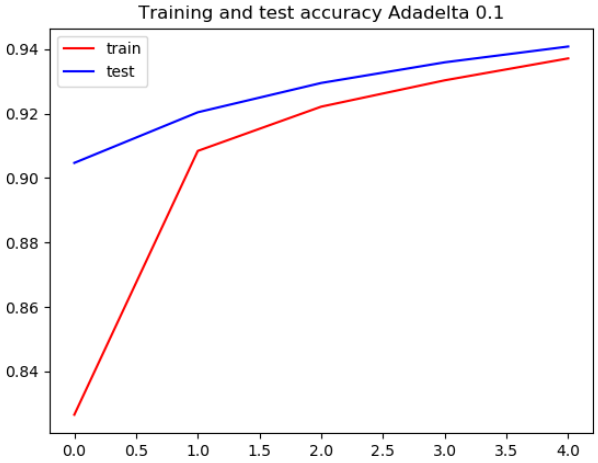

Рисунок 3 - точность Adadelta с параметром 0.1

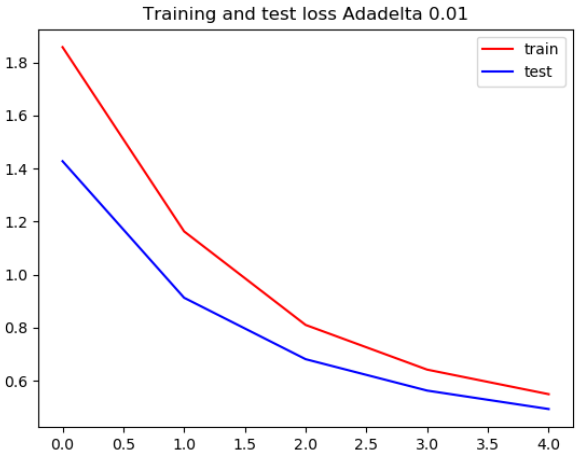

Рисунок 4 - ошибки Adadelta с параметром 0.1

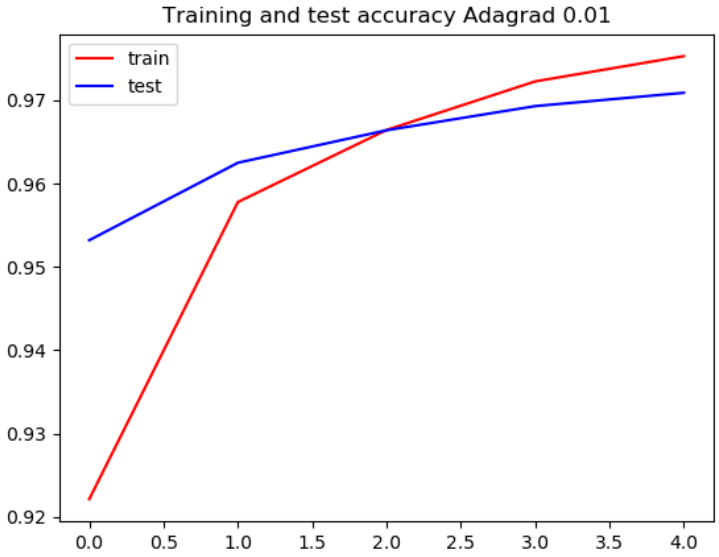

Рисунок 5 - точность Adagrad с параметром 0.01

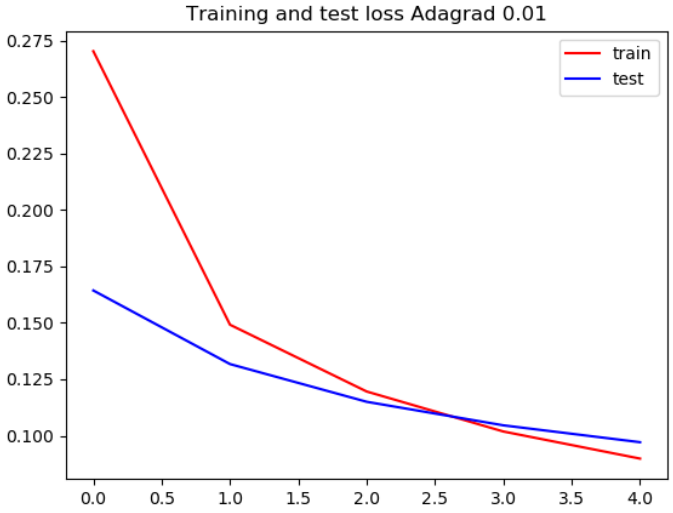

Рисунок 6 - ошибки Adagrad с параметром 0.01

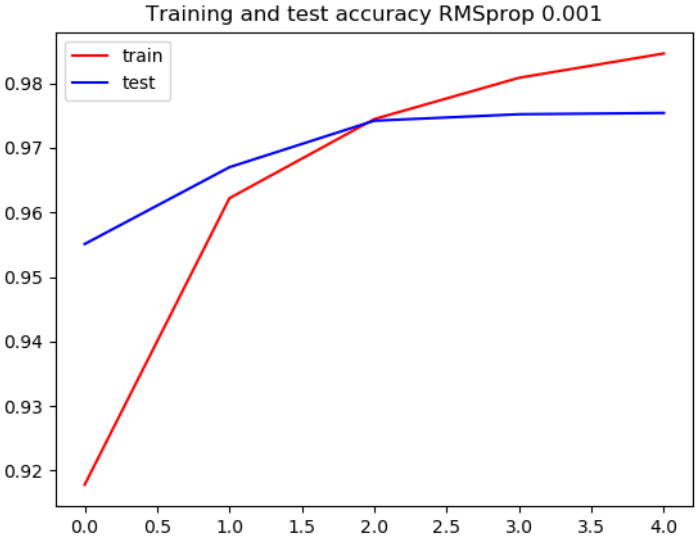

Рисунок 7 - точность RMSprop с параметром 0.001

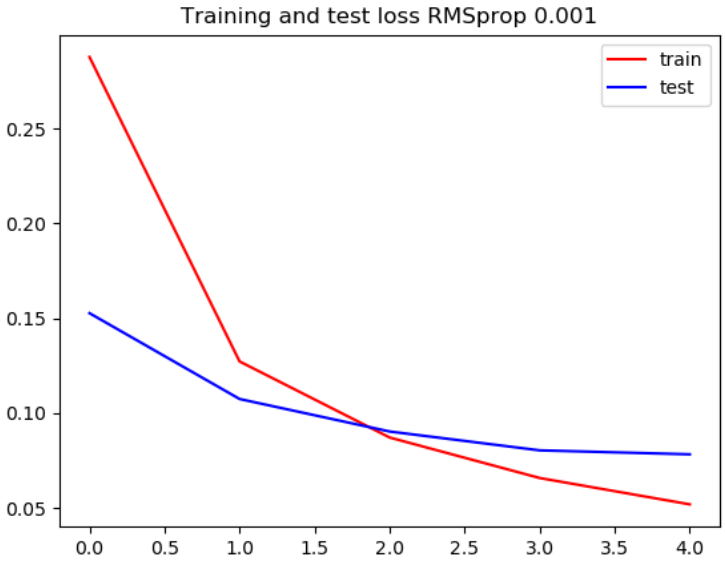

Рисунок 8-ошибки RMSprop с параметром 0.001

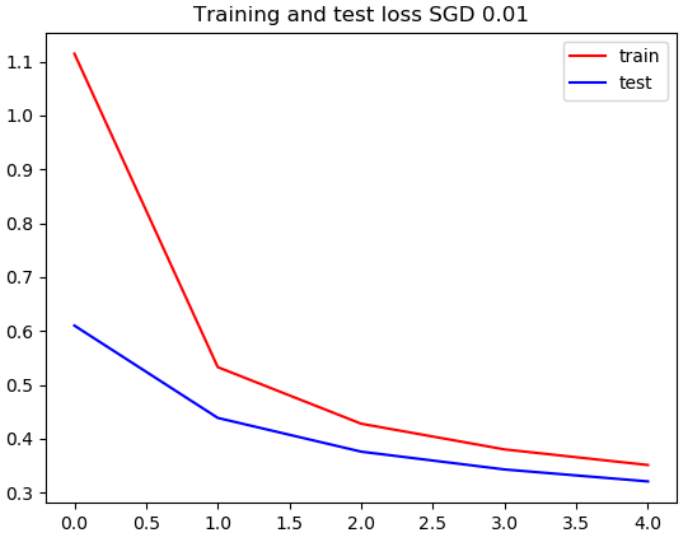

Рисунок 9 - ошибки SGD с параметром 0.01

Рисунок 10 - точность SGD с параметром 0.01

Выведем диаграммы точности каждого оптимизатора. Синим обозначены точность на тренировочных данных, а оранжевым на тестовых.

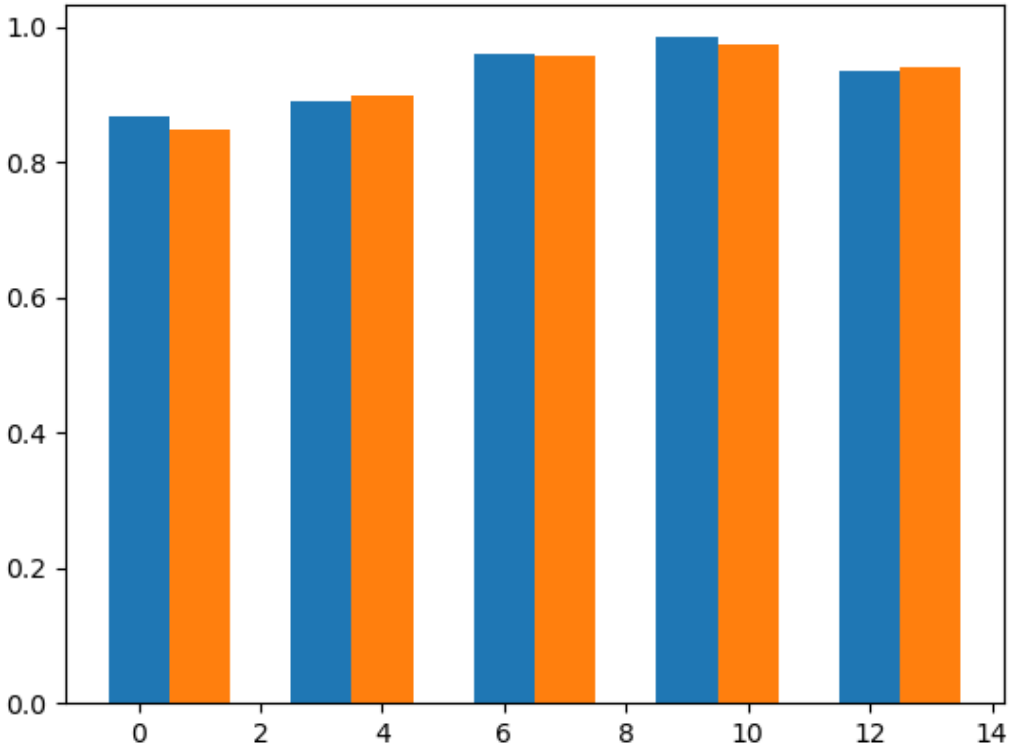

Рисунок 11 - диаграмма точности оптимизаторов Adam, RMSprop, SGD, Adagrad, Adadelta с параметром 0.1

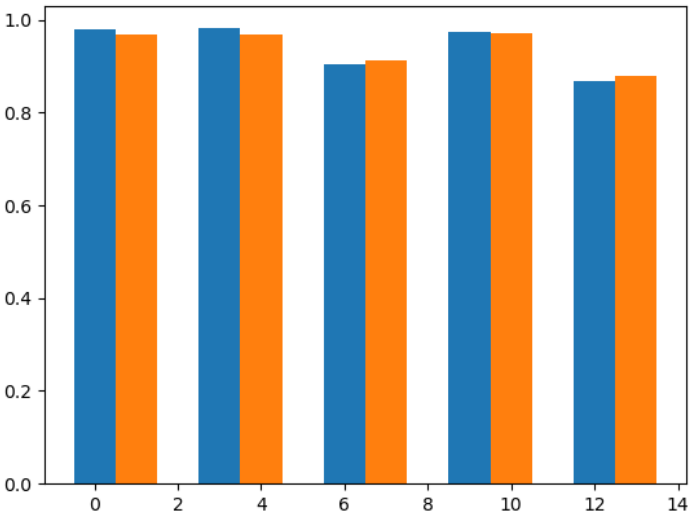

Рисунок 12 - диаграмма точности оптимизаторов Adam, RMSprop, optimizers.SGD, Adagrad, Adadelta с параметром 0.01

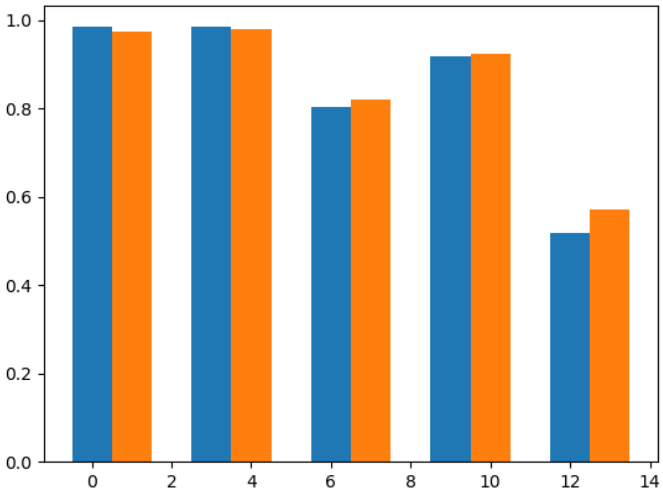

Рисунок 13 - диаграмма точности оптимизаторов Adam, RMSprop, optimizers.SGD, Adagrad, Adadelta с параметром 0.001

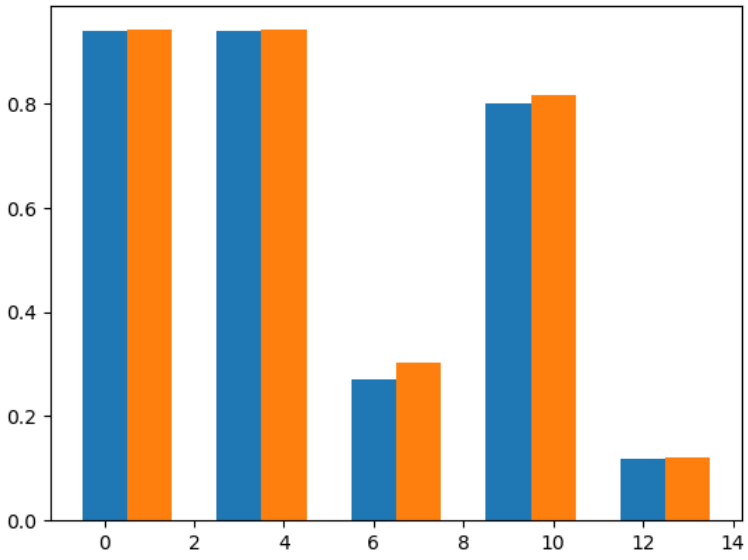

Рисунок 14- диаграмма точности оптимизаторов Adam, RMSprop, SGD, Adagrad, Adadelta с параметром 0.0001

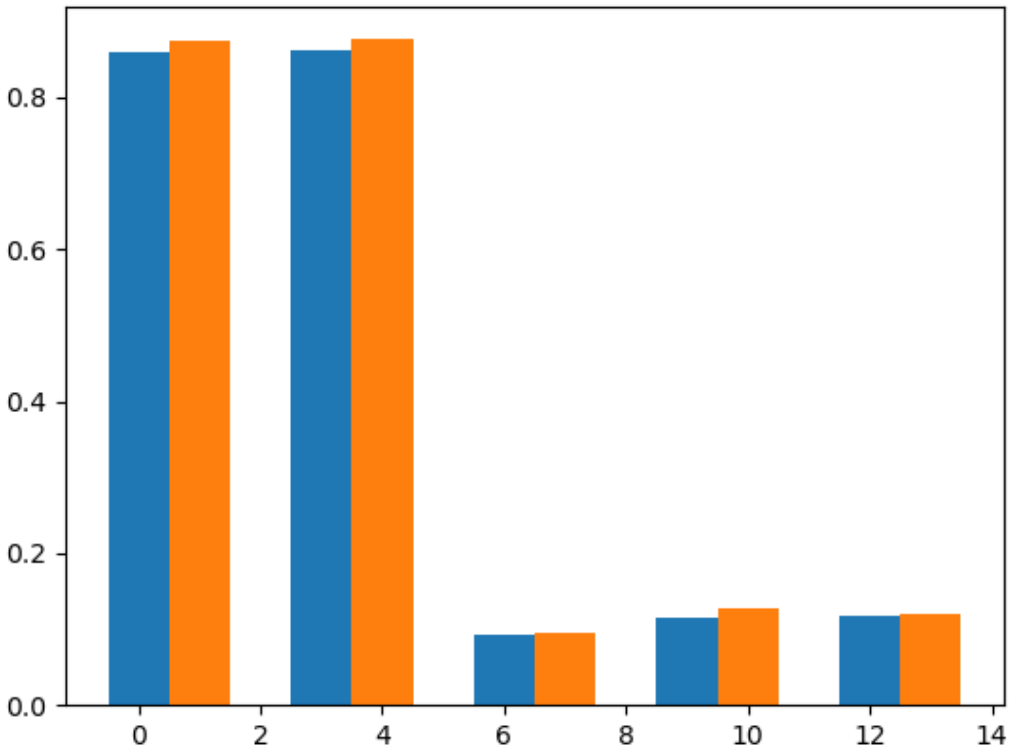

Рисунок 15- диаграмма точности оптимизаторов Adam, RMSprop, optimizers.SGD, Adagrad, Adadelta с параметром 0.00001

Возьмем оптимизатор RMSprop с параметром rho.

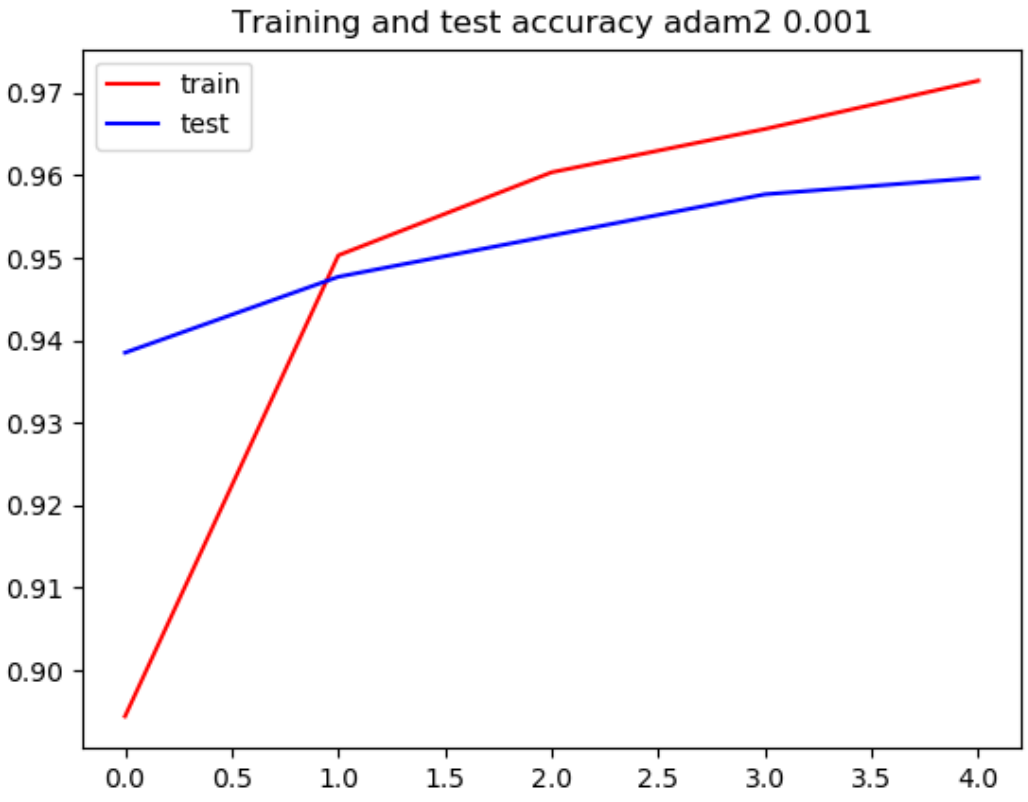

Рисунок 16 - Значение точности с оптимизатором rmsprop и параметром 0.99999

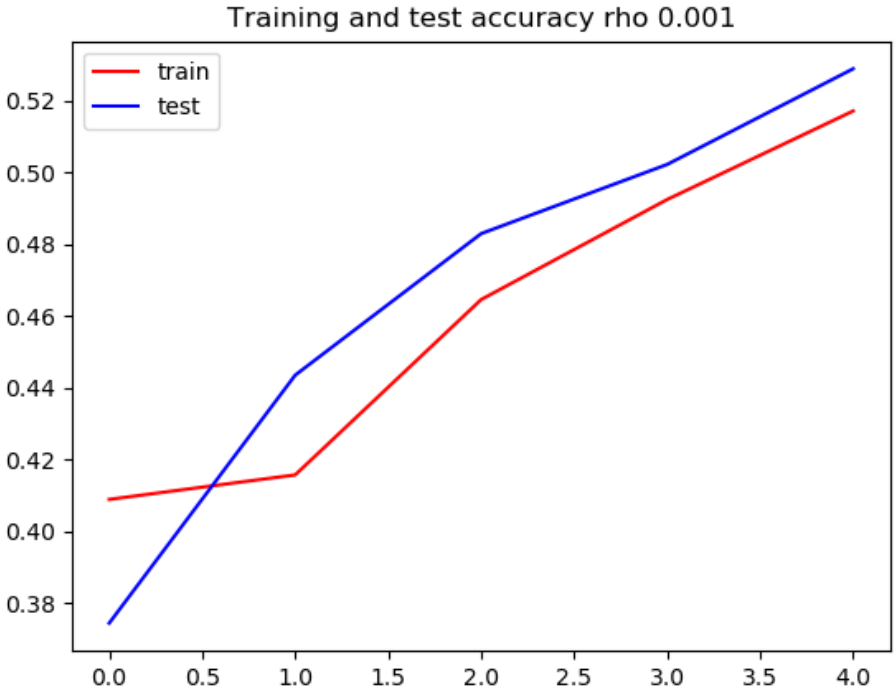

Рисунок 17- Значение точности с оптимизатором rmsprop и параметром 0.9999999

При стремлении rho к 1 начинает падать точность.

3. Функция считывания числа представлена в приложении А

Выводы.

Как видно из диаграмм, оптимизаторы Adam и RMSprop хорошо себя показывают при всех параметрах, кроме 0.1 (точность упала сильнее по сравнению с другими оптимизаторами). В то же время оптимизаторы SGD, Adagard и Adadelta стали хуже себя показывать при уменьшении параметра.

ПРИЛОЖЕНИЕ А

Исходный код программы

```
import tensorflow as tf
import keras.backend as K
import matplotlib.pyplot as plt
import pylab
from keras.utils import to_categorical
from keras.layers import Dense, Activation, Flatten
from keras.models import Sequential
import matplotlib
import numpy as np
from PIL import Image
from keras import optimizers

mnist = tf.keras.datasets.mnist
(train_images, train_labels),(test_images, test_labels) = mnist.load_data()



train_images = train_images / 255.0
test_images = test_images / 255.0

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

def build_model():
    model = Sequential()
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    return model




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

```
    test_loss, test_acc = model.evaluate(test_images, test_labels)

    print('test_acc:', test_acc)
    print('test_loss:', test_loss)#+" "+K.eval(model.optimizer.lr)
    res_train_data.append(history.history['accuracy'][-1])
    res_test_data.append(test_acc);
    plt.title('Training and test accuracy '+name+"
"+str(K.eval(model.optimizer.lr)))
    plt.plot(history.history['accuracy'], 'r', label='train')
    plt.plot(history.history['val_accuracy'], 'b', label='test')
    plt.legend()
    plt.savefig(name+str(K.eval(model.optimizer.lr))+'_acc' + '.png')
    plt.clf()

    plt.title('Training and test loss '+name+" "+str(K.eval(model.optimizer.lr)))
    plt.plot(history.history['loss'], 'r', label='train')
    plt.plot(history.history['val_loss'], 'b', label='test')
    plt.legend()
    plt.savefig(name+str(K.eval(model.optimizer.lr))+'_loss' + '.png')
    plt.clf()

def loadImage(filename):
    image = Image.open(filename).convert('L')
    image = image.resize((28, 28))
    image = np.array(image)
    image = image/255
    return np.expand_dims(image, axis=0)
optimizerslist = [optimizers.Adam, optimizers.RMSprop,  optimizers.SGD,
optimizers. Adagrad, optimizers.Adadelta]
learning_rates = [0.1, 0.01, 0.001, 0.0001, 0.00001]
list = learning_rates
res_train_data = []
res_test_data = []
for learn_rt in learning_rates:
    for optimizer in optimizerslist:
        test_optimizer(optimizer(learning_rate=learn_rt), optimizer.__name__)
    plt.bar(np.arange(len(optimizerslist)) * 3, res_train_data, width=1)
    plt.bar(np.arange(len(optimizerslist)) * 3 + 1, res_test_data, width=1)
    plt.show()
    plt.clf()
    res_test_data = []
```

```
res_train_data = []
```