

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

Студент гр. 7383

Александров Р.А.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Постановка задачи.

1. Ознакомиться с представлением графических данных.
2. Ознакомиться с простейшим способом передачи графических данных нейронной сети.
3. Создать модель.
4. Настроить параметры обучения.
5. Написать функцию, позволяющая загружать изображение пользователя и классифицировать его.

Требования.

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.

В ходе работы была создана и обучена модель нейронной сети, весь код представлен в приложении А.

Были проверено влияние следующих оптимизаторов с различным параметром learning rate (lr) равным 0.1 и 0.001:

- SGD;
- RMSprop;
- Adagrad;
- Adadelat;
- Adam;
- Nadam.

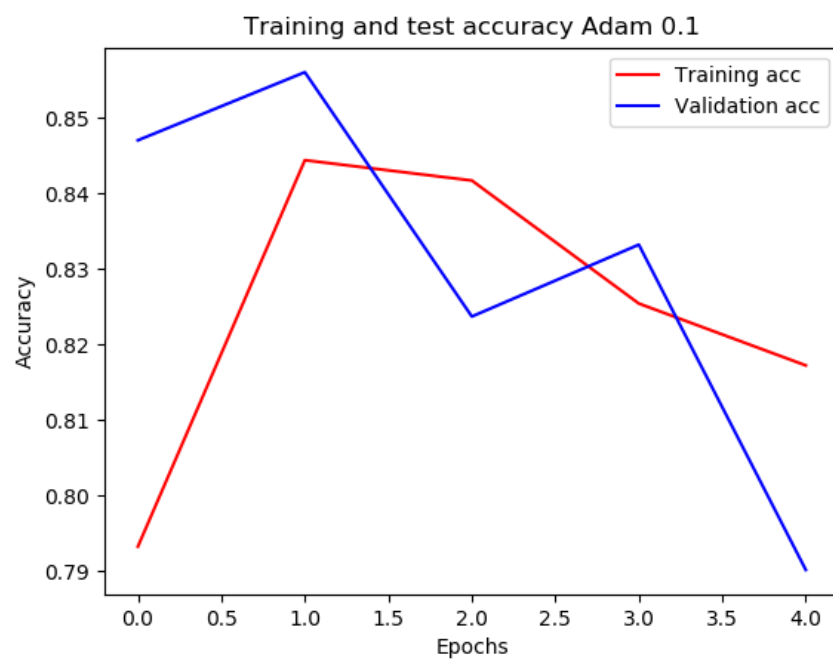


Рисунок 1 – Точность для оптимизатора Adam с $lr = 0.1$

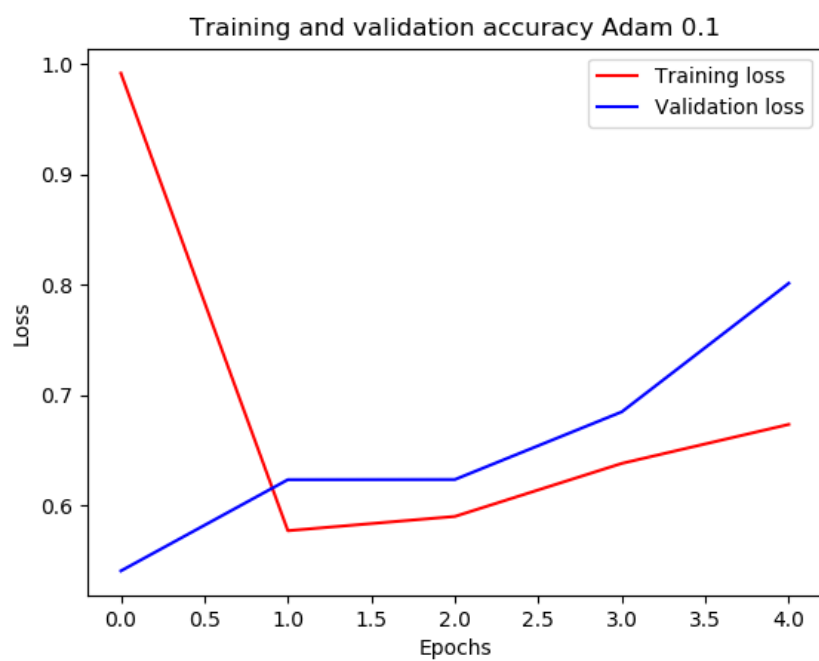


Рисунок 2 – Ошибки для оптимизатора Adam с $lr = 0.1$

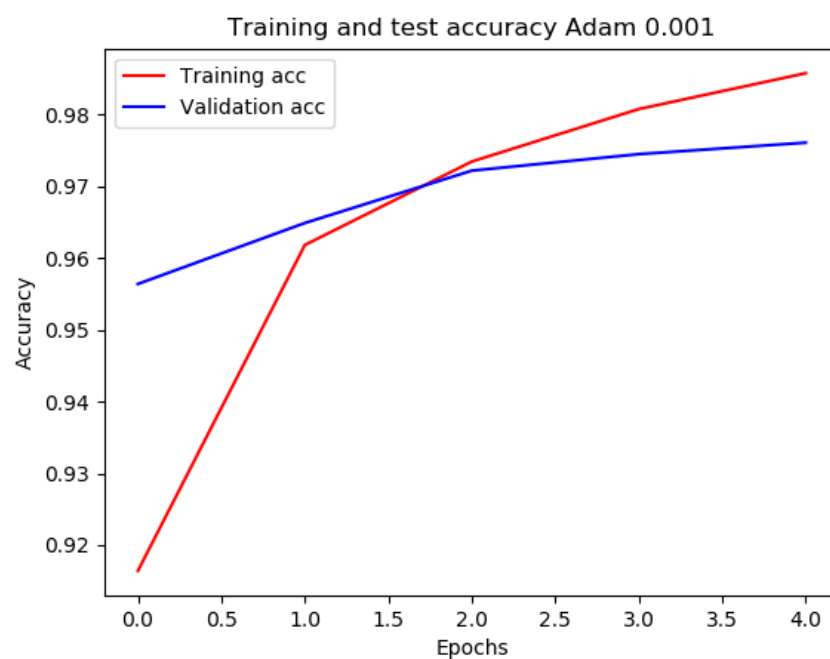


Рисунок 3 – Точность для оптимизатора Adam с $lr = 0.001$

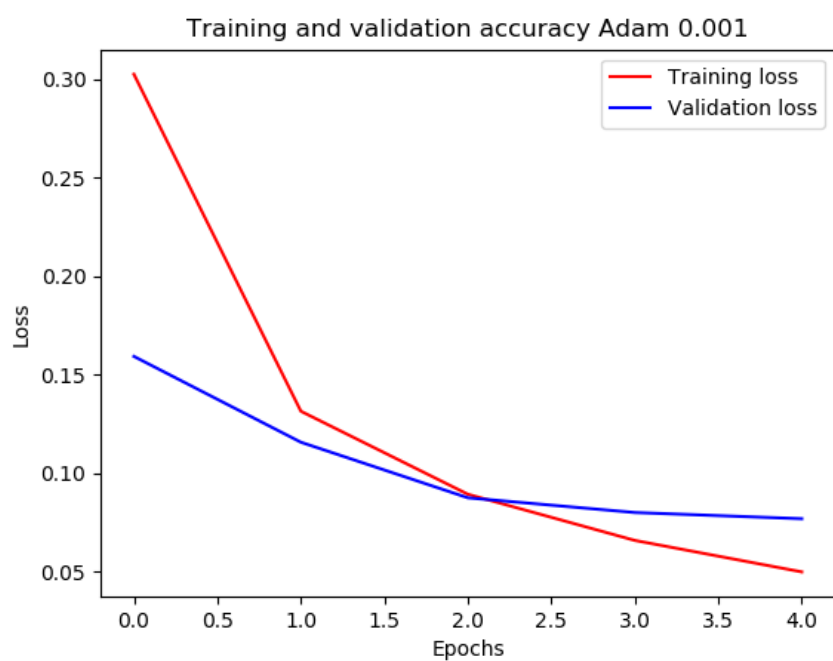


Рисунок 4 – Ошибки для оптимизатора Adam с $lr = 0.001$

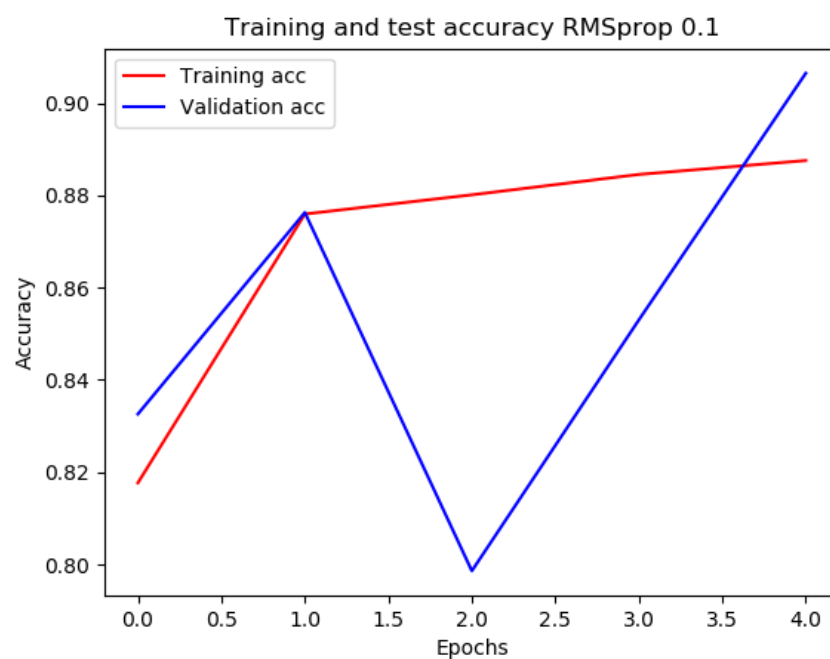


Рисунок 5 – Точность для оптимизатора RMSprop с $lr = 0.1$

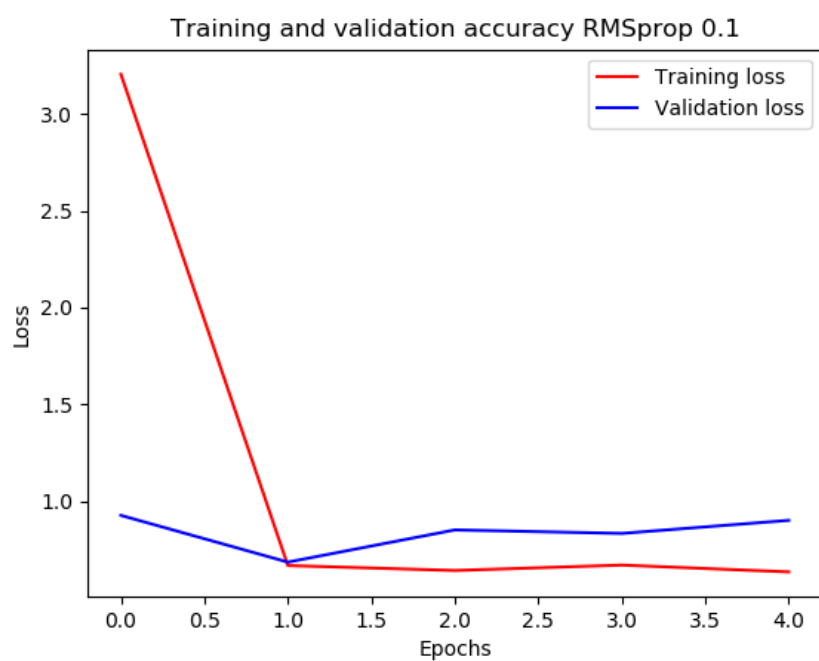


Рисунок 6 – Ошибки для оптимизатора RMSprop с $lr = 0.1$

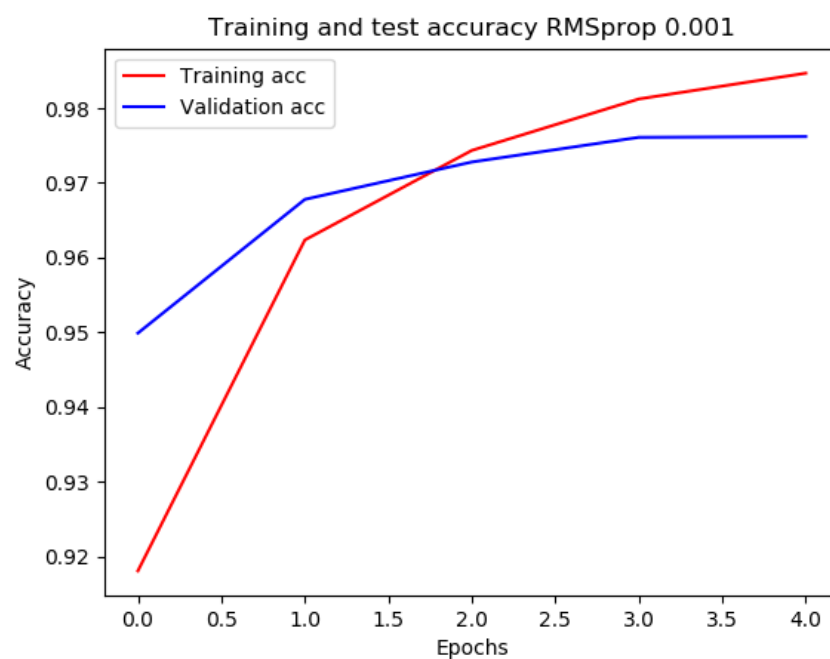


Рисунок 7 – Точность для оптимизатора RMSprop с $lr = 0.001$

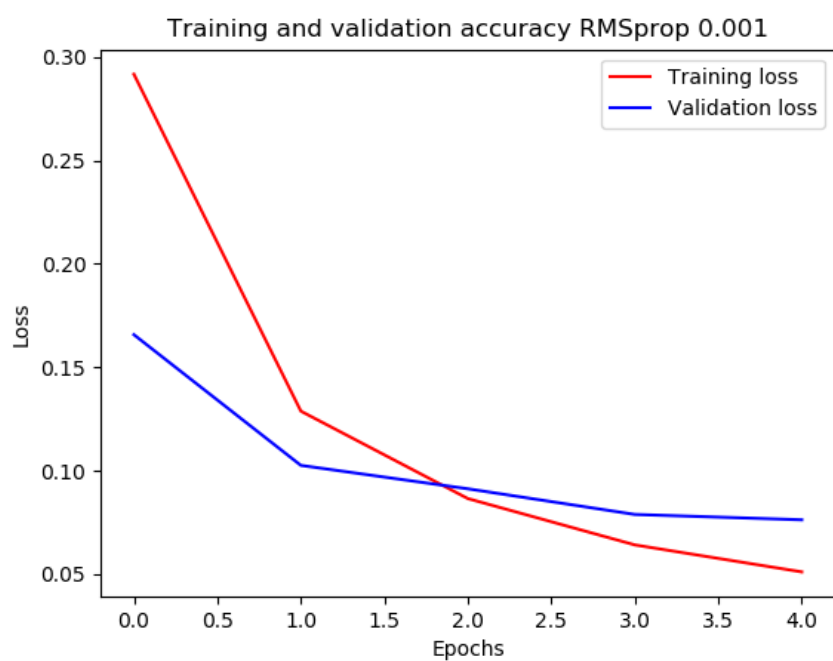


Рисунок 8 – Ошибки для оптимизатора RMSprop с $lr = 0.001$



Рисунок 9 – Точность для оптимизатора SGD с $lr = 0.1$

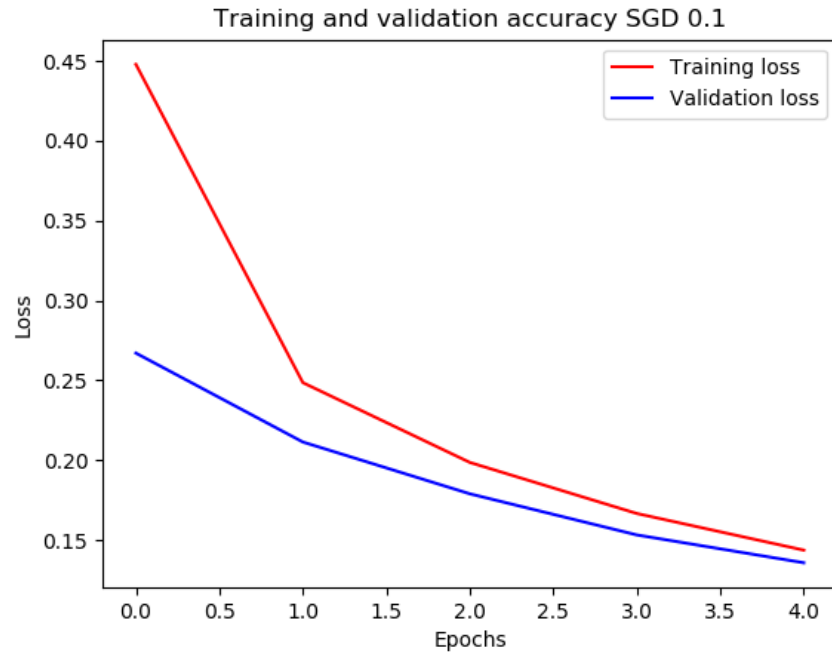


Рисунок 10 – Ошибки для оптимизатора SGD с $lr = 0.1$

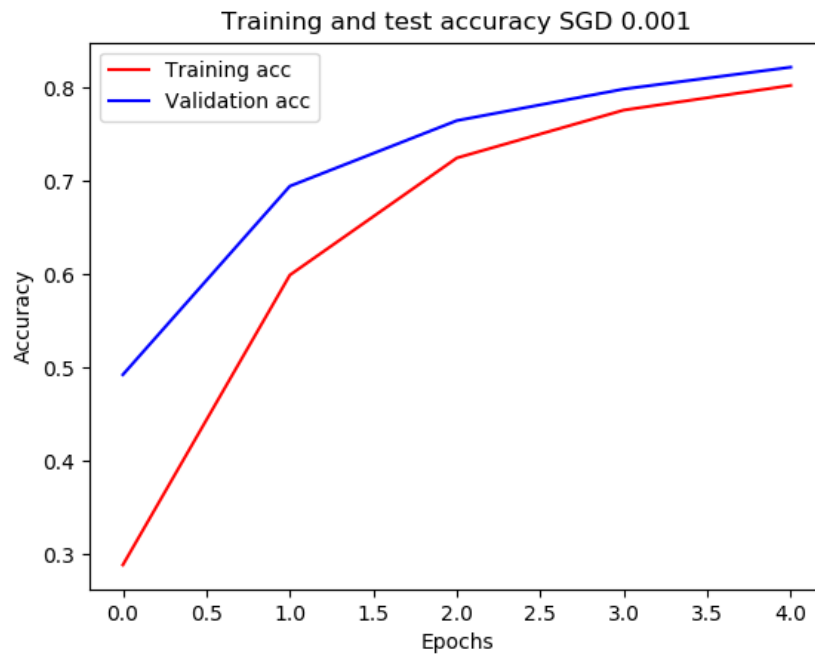


Рисунок 11 – Точность для оптимизатора SGD с $lr = 0.001$

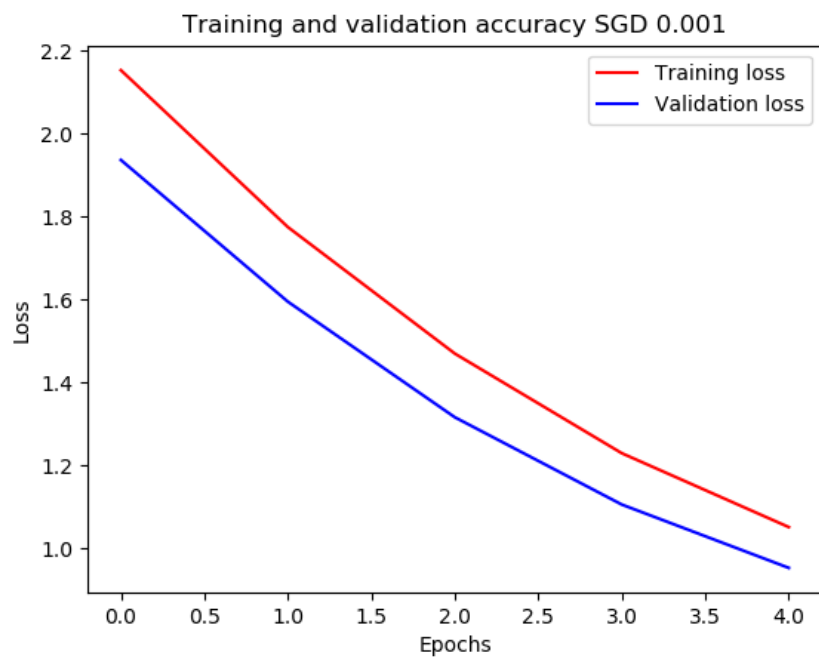


Рисунок 12 – Ошибки для оптимизатора SGD с $lr = 0.001$

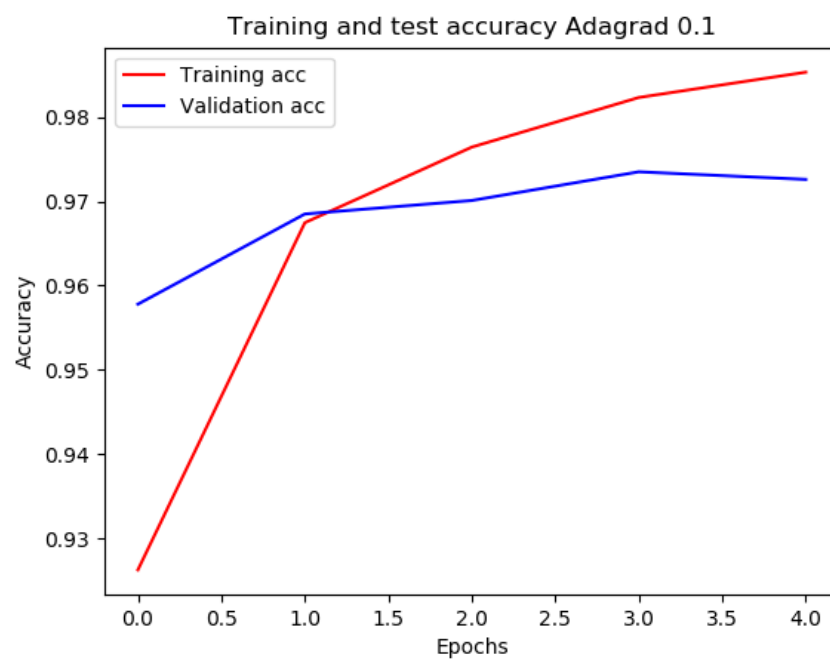


Рисунок 13 – Точность для оптимизатора Adagrad с $lr = 0.1$

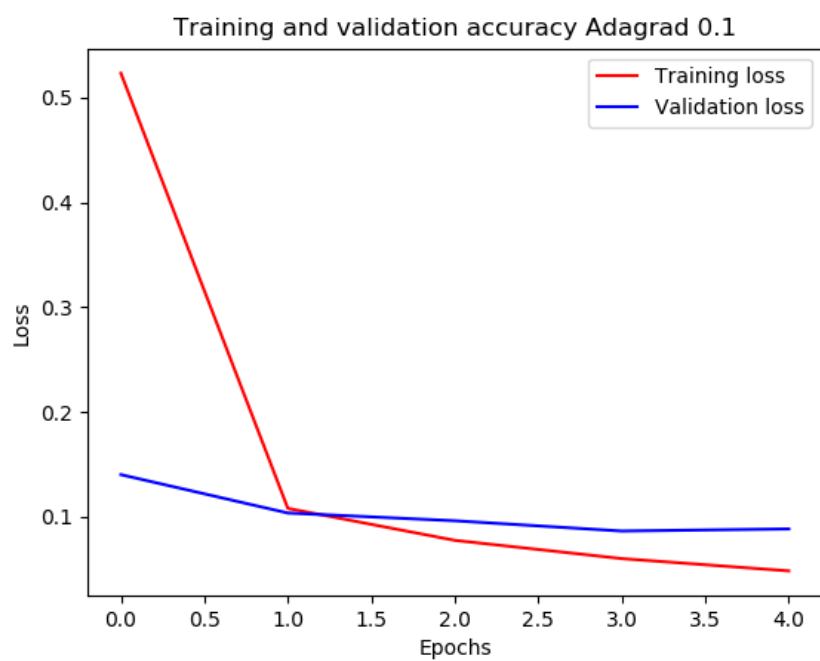


Рисунок 14 – Ошибки для оптимизатора Adagrad с $lr = 0.1$

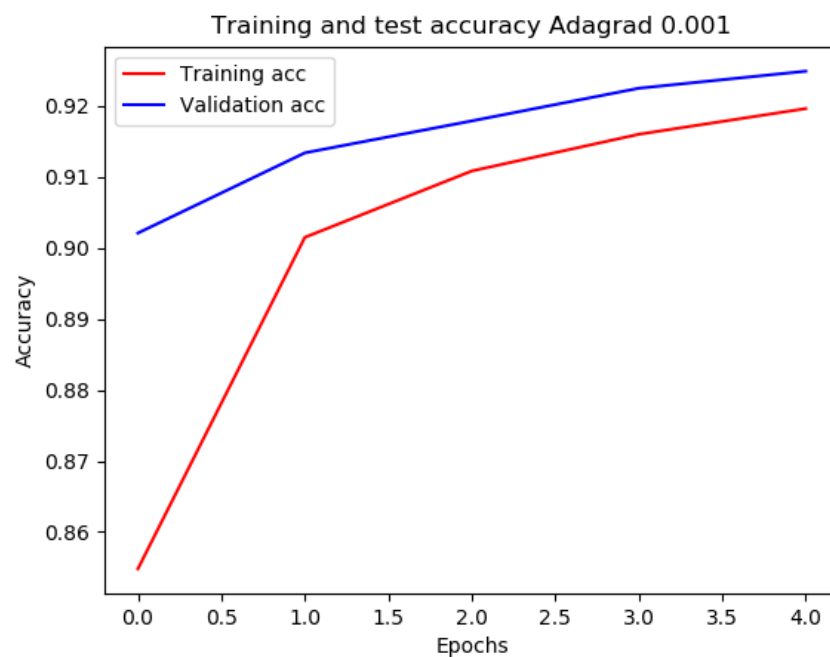


Рисунок 15 – Точность для оптимизатора Adagrad с $lr = 0.001$

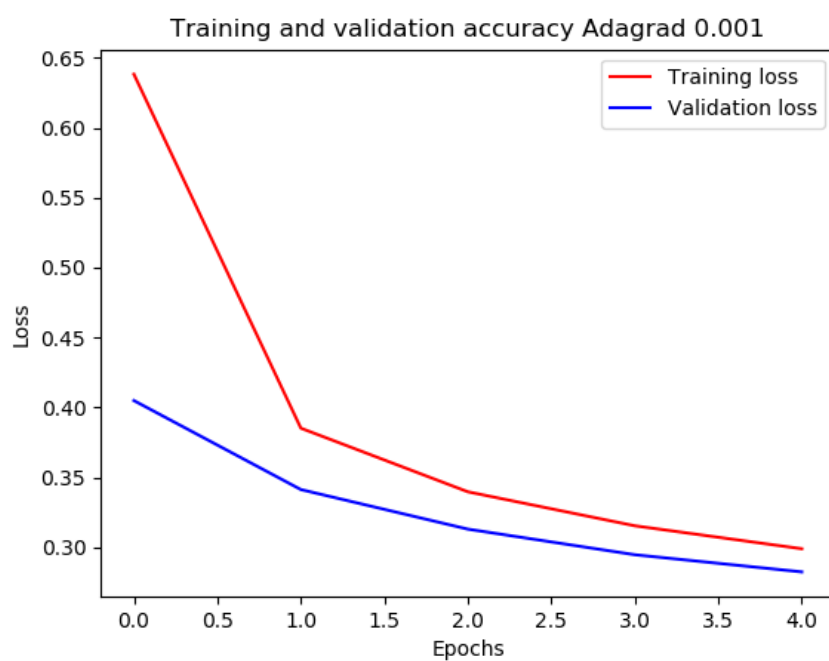


Рисунок 16 – Ошибки для оптимизатора Adagrad с $lr = 0.001$

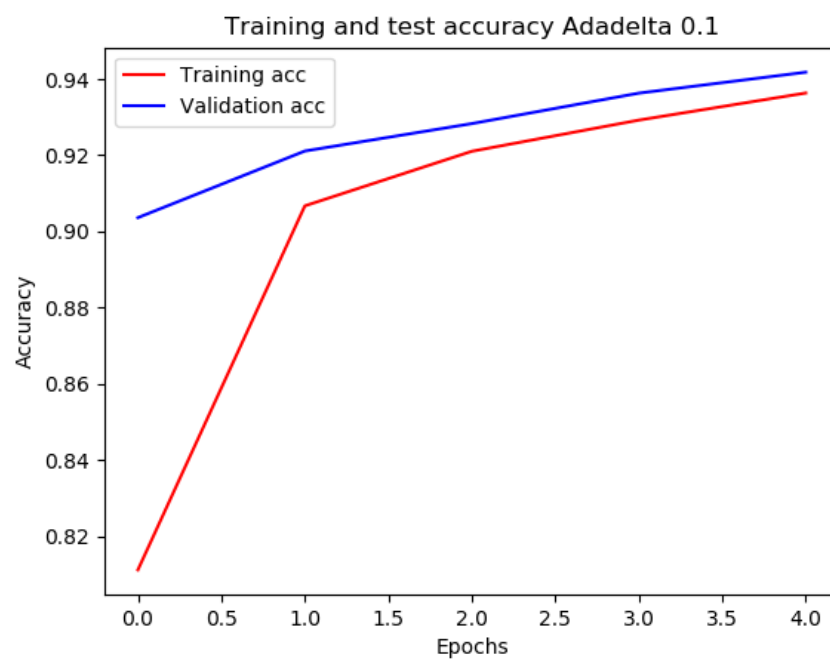


Рисунок 17 – Точность для оптимизатора Adadelta с $lr = 0.1$

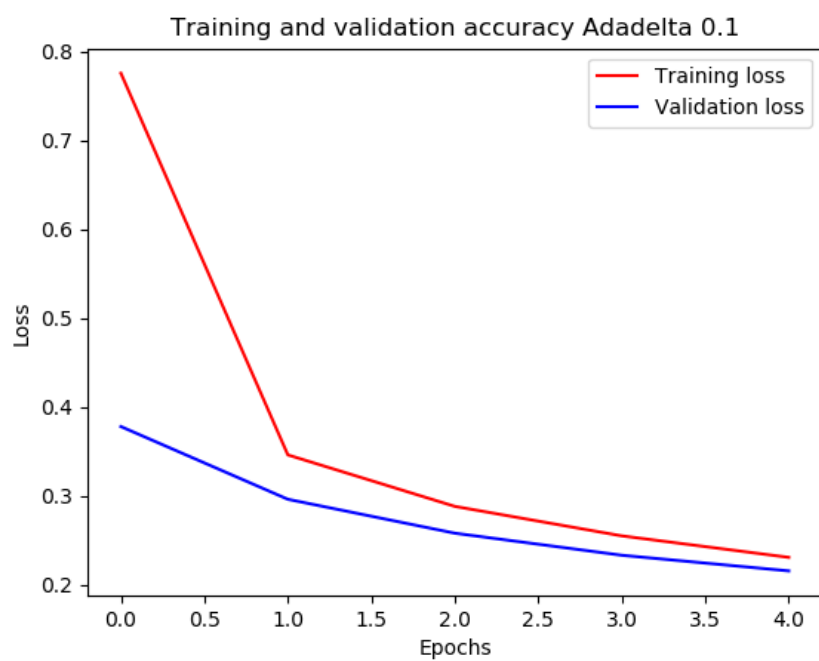


Рисунок 18 – Ошибки для оптимизатора Adadelta с $lr = 0.1$

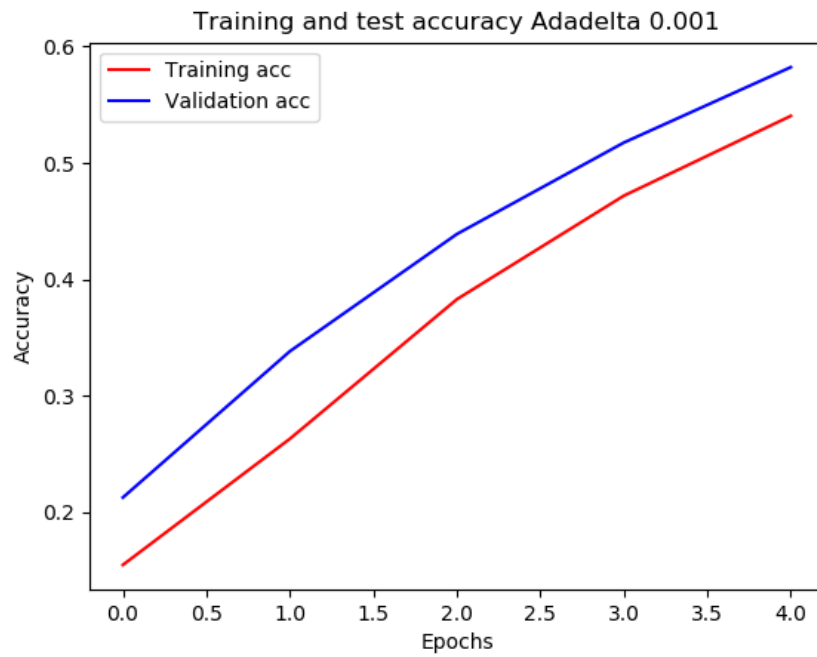


Рисунок 19 – Точность для оптимизатора Adadelta с $lr = 0.001$

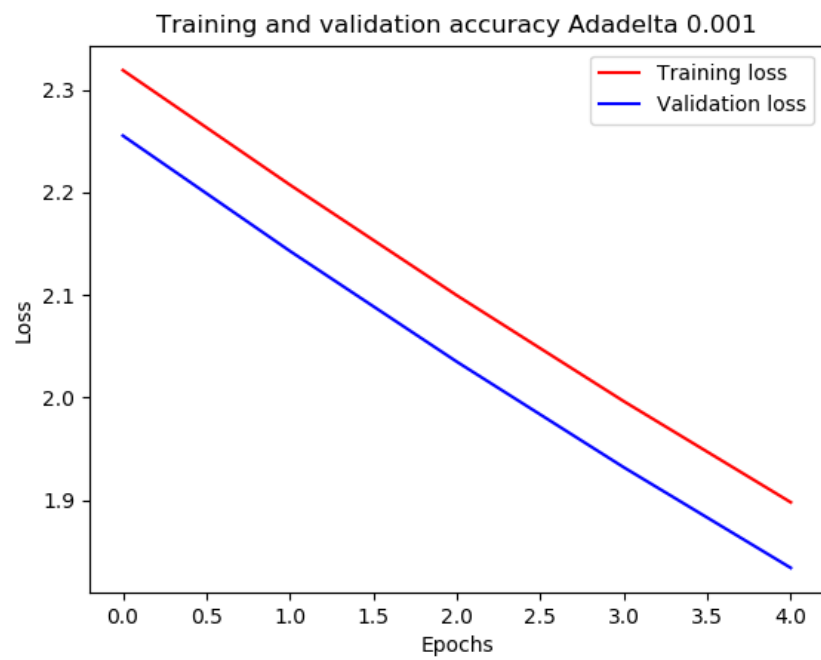


Рисунок 20 – Ошибки для оптимизатора Adadelta с $lr = 0.001$

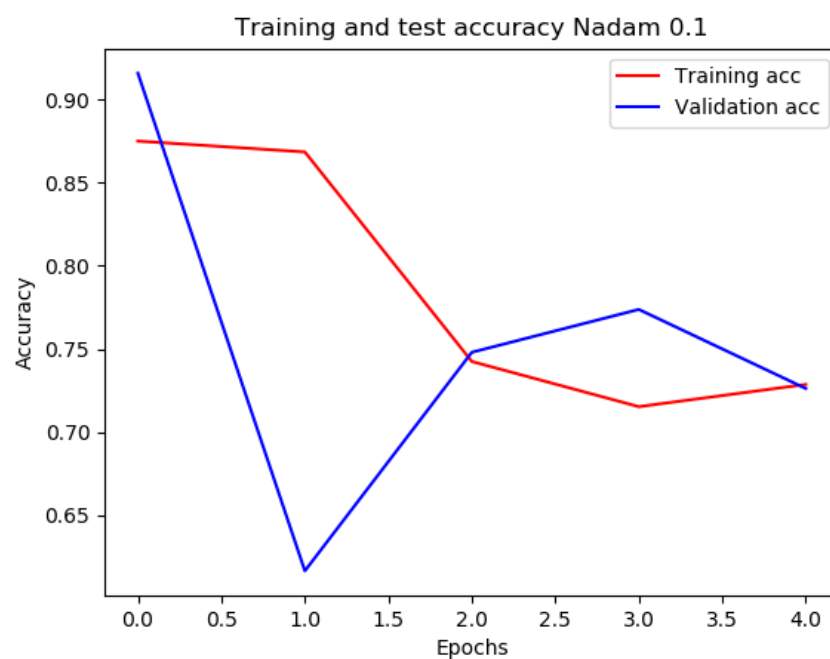


Рисунок 21 – Точность для оптимизатора Nadam с $lr = 0.1$

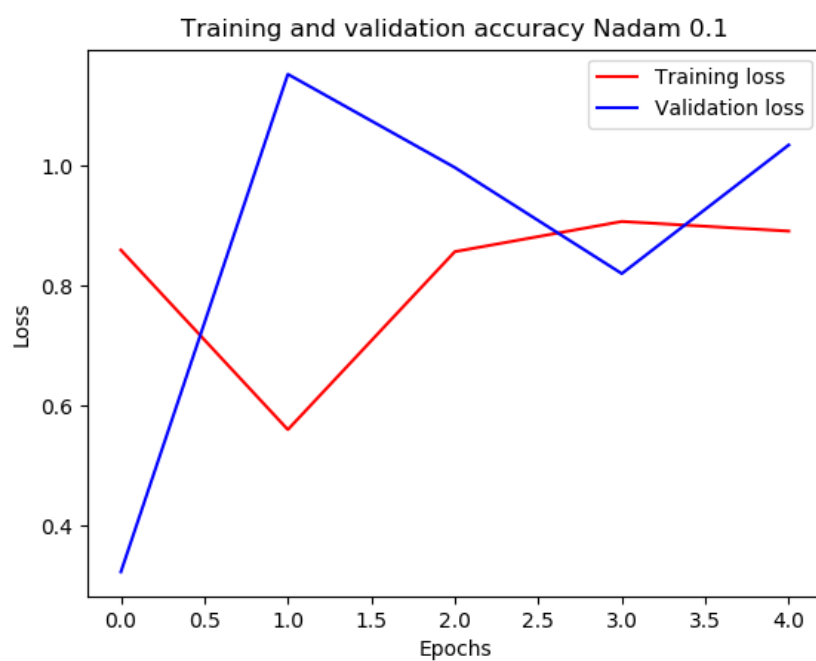


Рисунок 22 – Ошибки для оптимизатора Nadam с $lr = 0.1$

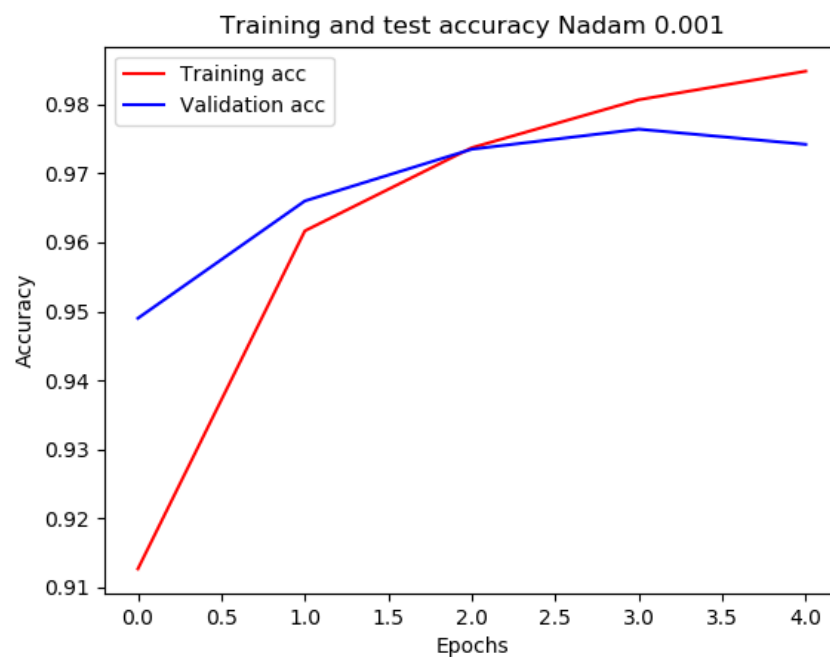


Рисунок 23 – Точность для оптимизатора Nadam с $lr = 0.001$

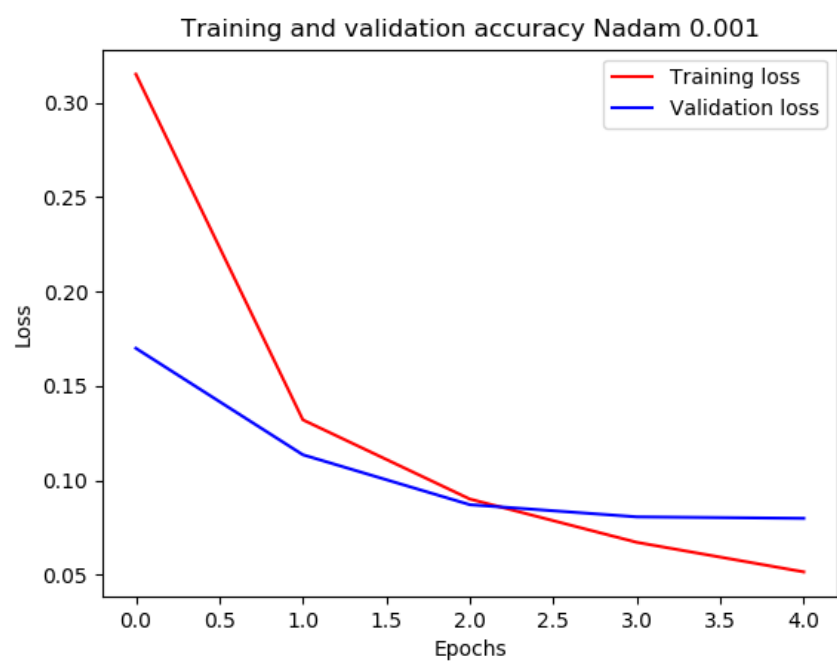


Рисунок 24 – Ошибки для оптимизатора Nadam с $lr = 0.001$

Из графиков видно, что точности не менее 95% удовлетворяют Adam, RMSprop и Nadam с $lr = 0.001$, Adagrad с $lr = 0.1$.

Выводы.

В ходе работы были изучены представление графических данных, простейший способ передачи графических данных нейронной сети, написана функция, позволяющая загружать изображение пользователи и классифицировать его, было исследовано влияние различных оптимизаторов, а также их параметров на процесс обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.utils import to_categorical
from keras.layers import Dense, Flatten
from keras.models import Sequential
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 get_image(filename):
    im = Image.open(filename)
    im = im.resize((28, 28))
    im = np.dot(np.asarray(im), np.array([1/3, 1/3, 1/3]))
    im /= 255
    im = 1 - im
    im = im.reshape((1, 28, 28))

    return im

model = build_model()
model.compile(optimizer=optimizers.Adam(learning_rate=0.1), loss='c
ategorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=5,
batch_size=128, validation_data=(test_images, test_labels))
```



```
plt.title('Training and test accuracy Adam 0.1')
plt.plot(history.history['accuracy'], 'r', label='Training acc')
plt.plot(history.history['val_accuracy'], 'b', label='Validation
acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.clf()

plt.plot(history.history['loss'], 'r', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation accuracy Adam 0.1')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.clf()
```