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

ОТЧЕТ

по лабораторной работе 4

по дисциплине «Искусственные нейронные сети»

Тема: «Распознавание рукописных символов»

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель.

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

Задачи.

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

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

Ход работы.

Была построена нейронная сеть, код которой представлен в приложении А.

Было проверено соответствие метки и изображения. Они соответствуют друг другу. Были получены графики точности и ошибок для четырех функций оптимизаций. Графики представлены на рис. 1-16.

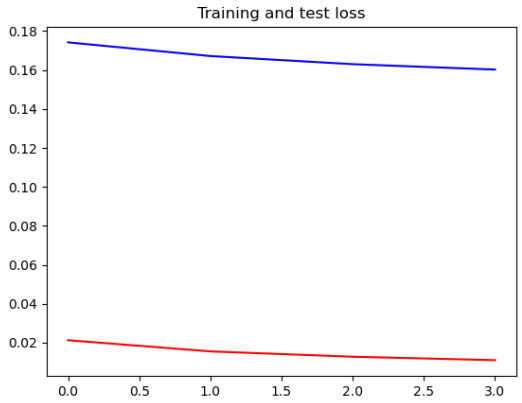

Рисунок 1 – График ошибок Adagrad 0.01

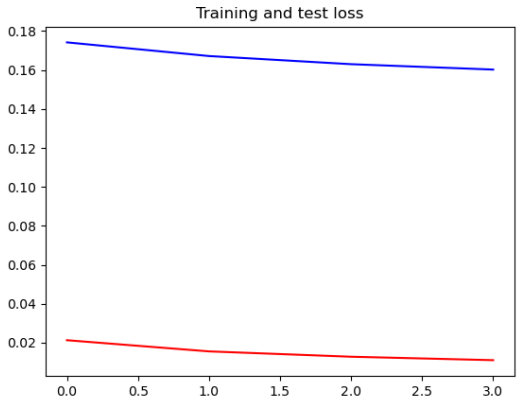

Рисунок 2 – График точности Adagrad 0.01

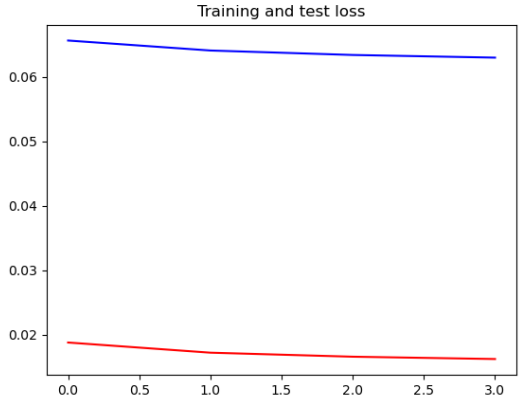

Рисунок 3 – График ошибок Adagrad 0.001

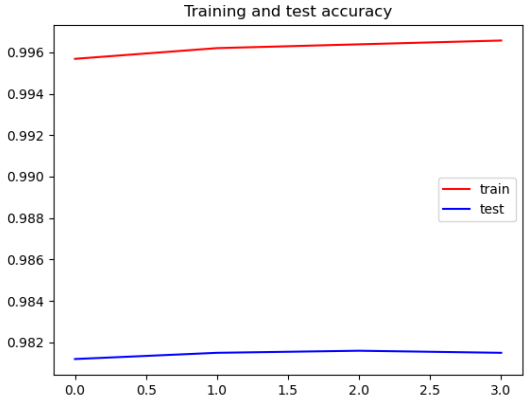

Рисунок 4 – График точности Adagrad 0.001

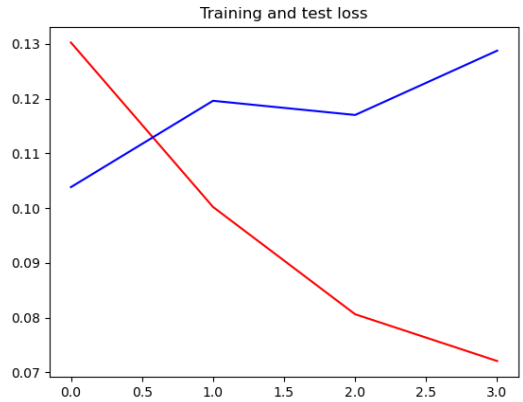

Рисунок 5 – График ошибок Adam 0.01

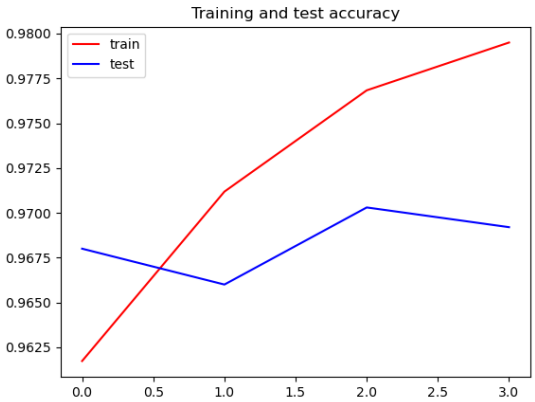

Рисунок 6 – График точности Adam 0.01

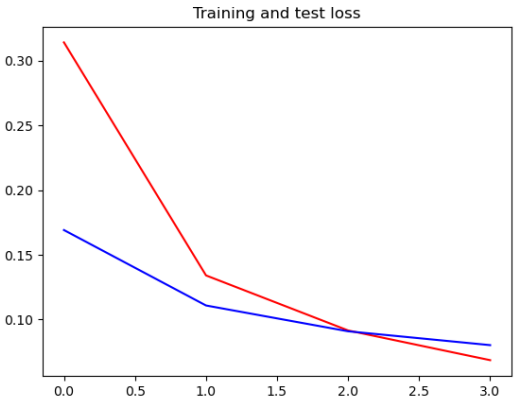

Рисунок 7 – График ошибок Adam 0.001

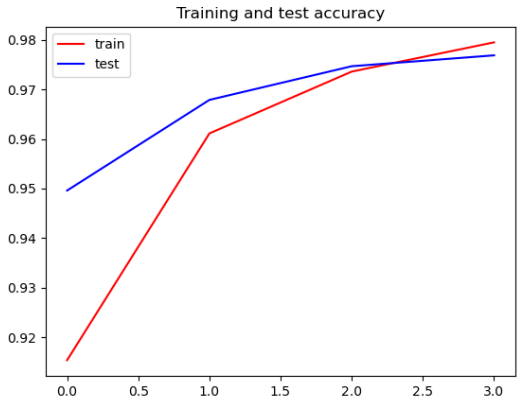

Рисунок 8 – График точности Adam 0.001

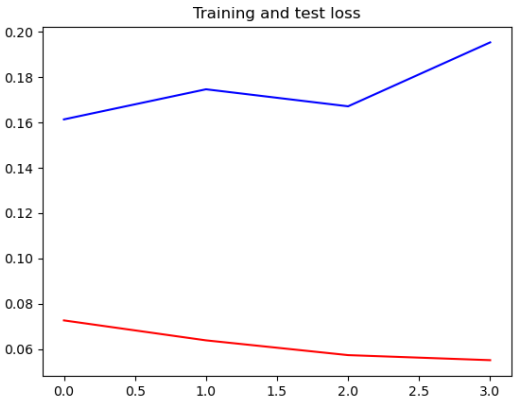

Рисунок 9 – График ошибок RMSprop 0.01

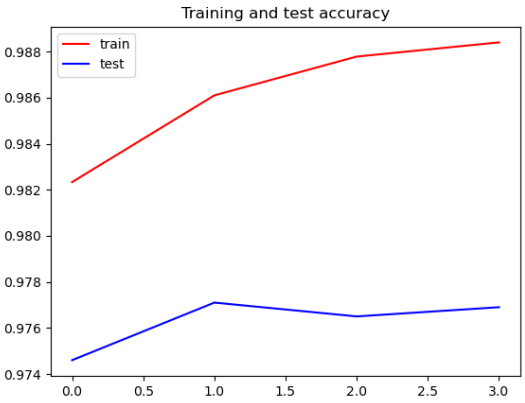

Рисунок 10 – График точности RMSprop 0.01

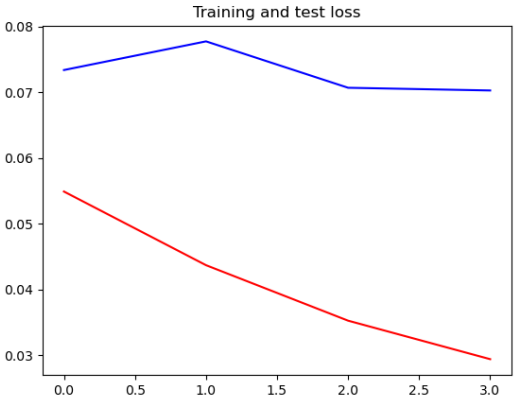

Рисунок11 – График ошибок RMSprop 0.001

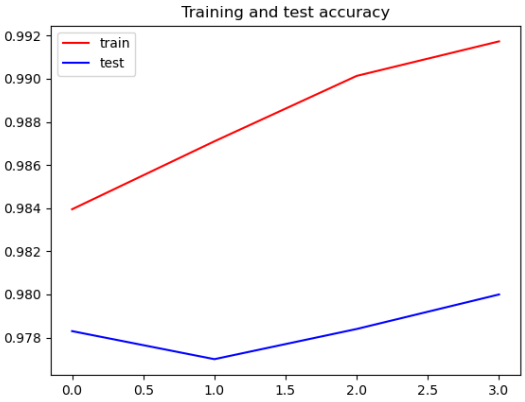

Рисунок 12 – График точности RMSprop 0.001

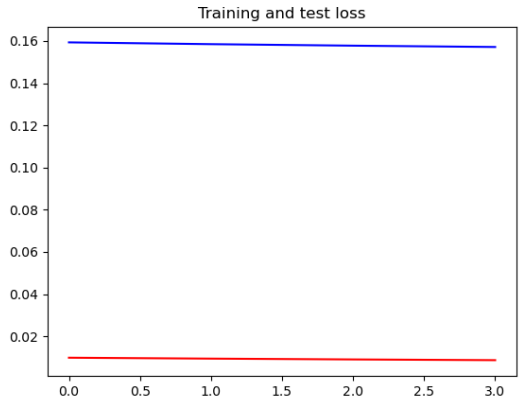

Рисунок 13 – График ошибок SGD 0.01

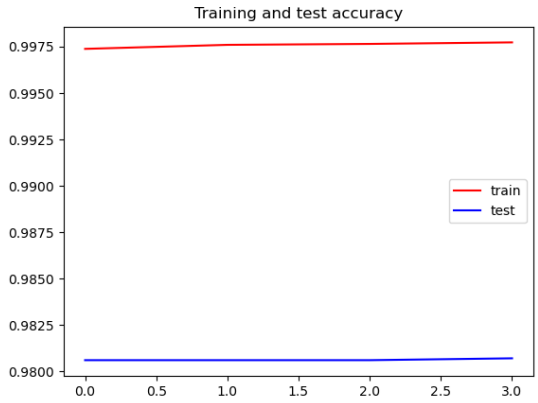

Рисунок 14 – График точности SGD 0.01

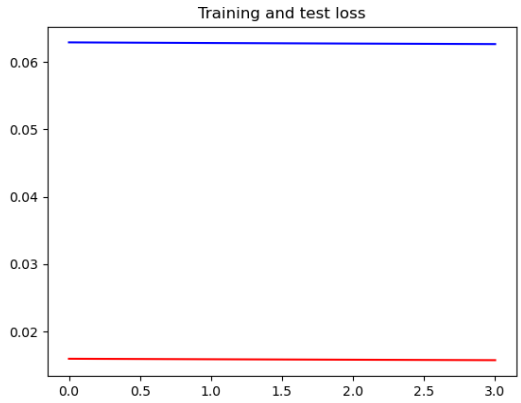

Рисунок 15 – График ошибок SGD 0.001

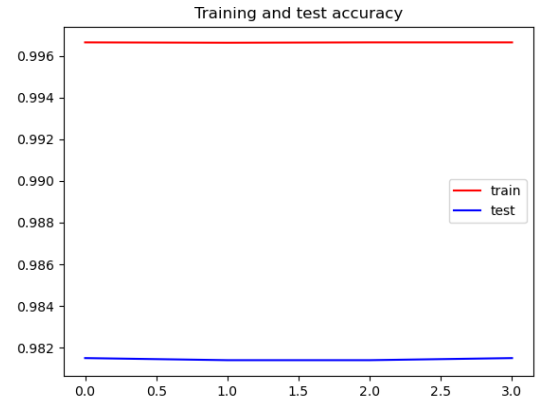

Рисунок 16 – График точности SGD 0.001

В табл. 1 приведены значения ошибок и точности.

Таблица 1 – Значения точности

	Adagrad	Adam	RMSprop	SGD
0.01	0.9804	0.9692	0.9769	0.9807
0.001	0.9815	0.9756	0.98	0.9815

Из табл. 1 видно, что лучшие показатели у Adagrad и SGD при learning_rate = 0.01.

Выводы.

В ходе выполнения лабораторной работы была изучено представление графических данных и способом передачи графических данных нейронной сети. Была создана модель искусственной нейронной сети и настроены параметры обучения. Были построены графики ошибок и точности в ходе обучения. Была написана функция, позволяющая загружать изображение пользователем и классифицировать его.

ПРИЛОЖЕНИЕ А

```
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras import optimizers

def load_img(path):
    img = load_img(path=path, target_size=(28, 28))
    return img_to_array(img)

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

plt.imshow(train_images[0], cmap=plt.cm.binary)
#plt.show()
#print(train_labels[0])

train_images = train_images / 255.0
test_images = test_images / 255.0

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

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

def optimizer_research(optimizer):
    optimizerConf = optimizer.get_config()
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['acc'])
    history = model.fit(train_images, train_labels, epochs=4, 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('optimizer', optimizerConf)
    plt.title('Training and test accuracy')
    plt.plot(history.history['acc'], 'r', label='train')
    plt.plot(history.history['val_acc'], 'b', label='test')
    plt.legend()
    plt.savefig("%s_%s_%s_acc.png" % (optimizerConf["name"],
optimizerConf["learning_rate"], test_acc), format='png')
    plt.clf()

    plt.title('Training and test loss')
    plt.plot(history.history['loss'], 'r', label='train')
    plt.plot(history.history['val_loss'], 'b', label='test')
    plt.savefig("%s_%s_%s_loss.png" % (optimizerConf["name"],
optimizerConf["learning_rate"], test_acc), format='png')
    plt.legend()
    plt.clf()
    return model

for learning_rate in [0.001, 0.01]:
    optimizer_research(optimizers.Adam(learning_rate=learning_rate))
    optimizer_research(optimizers.RMSprop(learning_rate=learning_rate))
```

```
optimizer_research(optimizers.Adagrad(learning_rate=learning_rate))
optimizer_research(optimizers.SGD(learning_rate=learning_rate))
```