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. 7382 _____ Токарев А.П.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

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 пользователи и классифицировать его.

Требования к выполнению задания.

1. Найти архитектуру сети, при которой точность классификации будет не менее95%.
2. Исследовать влияние различных оптимизаторов, а также и х параметров, на процесс обучения.
3. Написать функцию,которая позволит загружать пользовательское изображение не из датасета.

Основные теоретические положения.

MINIST – база данных рукописных цифр, имеющая подготовленный набор обучающих значений в размере 60000 примеров и тестовых значений из 10000 примеров. Это подмножество более широкого набора, доступное из NIST. Наш набор состоит из изображений размером 28x28, каждый пиксель которого представляет собой оттенок серого, цифры нормализированы по размеру и имеют фиксированный размер изображения. Таким образом, есть 10 цифр (от 0 до 9) или 10 классов для прогнозирования. Результаты сообщаются с использованием ошибки прогнозирования,которая является не чем иным, как инвертированной точностью классификации.

Ход работы.

Была создана и обучена модель искусственной нейронной сети. Код предоставлен в приложении А.

```
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(784,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Данная архитектура дает точность ~ 95% (loss: 0.1717 - accuracy: 0.9507 - val_loss: 0.1526 - val_accuracy: 0.9582) . Графики точности и ошибки предоставлены на рис. 1 и рис. 2 соответственно.

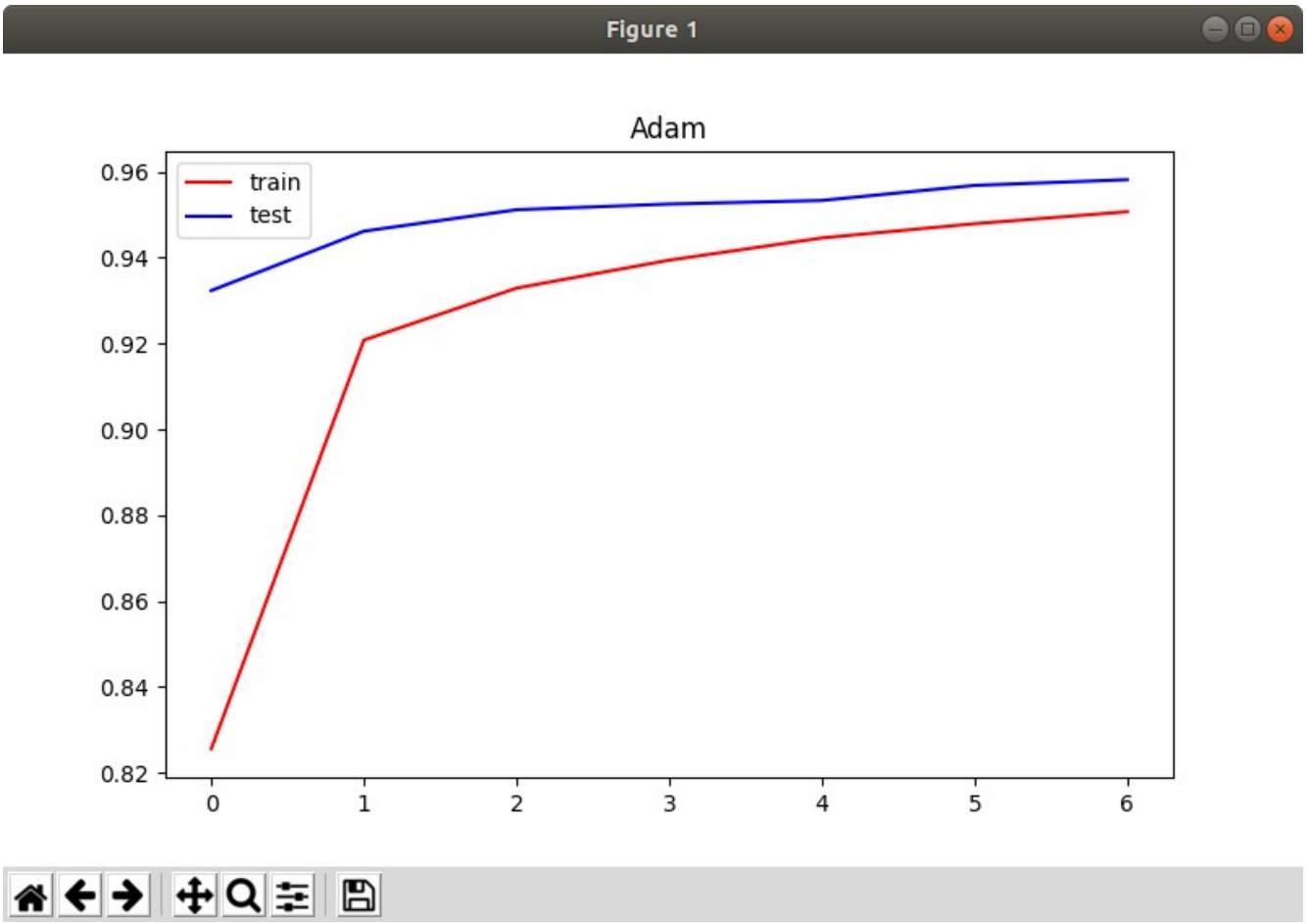

Рисунок 1 – График точности для оптимизатора adam

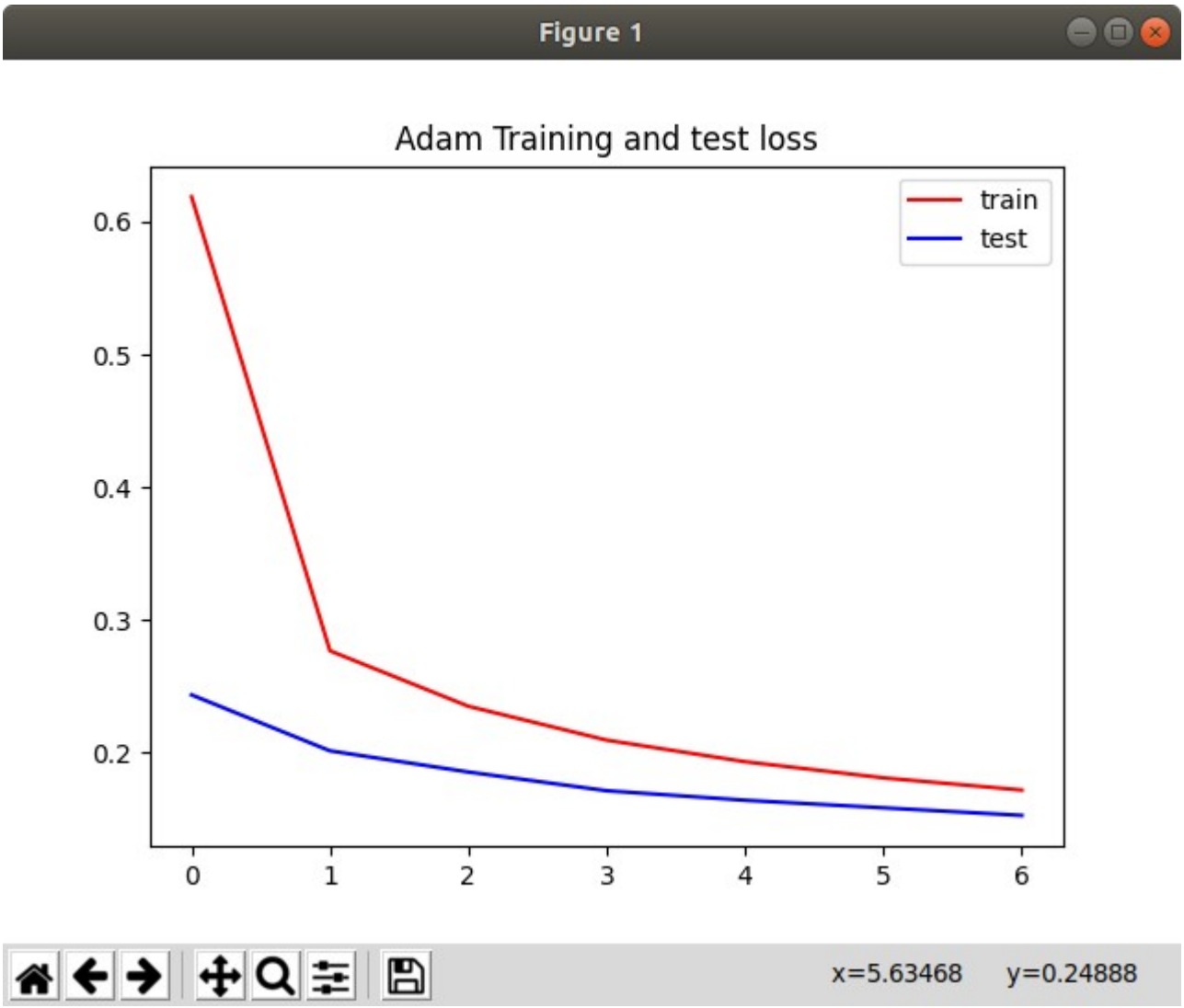

Рисунок 2 – График потерь для оптимизатора adam

Напишем функцию, которая позволит загружать пользовательское изображение не из датасета:

```
def upload_image(path):
    img = image.load_img(path=path, grayscale=True, target_size=(28, 28, 1))
    img = image.img_to_array(img)
    return img.reshape((1, 784))
```

Выводы.

В ходе работы была изучена задача классификации рукописных цифр с помощью базы данных MINIST. Подобрана архитектура, дающая точность свыше 95%, таковой оказалась adam. Также была написана функция загрузки изображения в память программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import keras
import matplotlib.pyplot as plt
from keras import Sequential
from keras import optimizers as opt
from keras.layers import Dense
from keras.preprocessing import image
from keras.utils import to_categorical
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape(60000, 784)
test_images = test_images.reshape(10000, 784)
train_images = train_images / 255.0
test_images = test_images / 255.0
train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)
def upload_image(path):
    img = image.load_img(path=path, grayscale=True, target_size=(28, 28, 1))
    img = image.img_to_array(img)
    return img.reshape((1, 784))
def build_model():
    model = Sequential()
    model.add(Dense(16, activation='relu', input_shape=(784,)))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    return model
def draw(H, title_name):
    plt.figure(1, figsize=(8, 5))
    plt.title(title_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".format(title_name))
    plt.plot(H.history['loss'], 'r', label='train')
    plt.plot(H.history['val_loss'], 'b', label='test')
    plt.legend()
    plt.show()
    plt.clf()
model = build_model()
model.compile(optimizer=opt.Adam(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
H = model.fit(train_images, train_labels, epochs=7, batch_size=100, validation_split=0.1)
test_loss, test_acc = model.evaluate(test_images, test_labels)
draw(H, "Adam")
model = build_model()
```