

**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

\_\_\_\_\_

Медведев И.С.

Преподаватель

\_\_\_\_\_

Жукова Н. А.

Санкт-Петербург

2020

### **Цель работы.**

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

### **Выполнение работы.**

- 1) Была создана модель искусственной нейронной сети в Keras. Код программы представлен в приложении А.
- 2) Графики ошибок и точности строились с помощью библиотеки matplotlib.
- 3) При исследовании разных оптимизаторов, также менялись их параметры.

### **Оптимизатор RMSprop.**

В данном оптимизаторе при исследовании изменялась скорость обучения. Была взята стандартная скорость 0.001 и скорость, равная 0.03. Результаты представлены на рис. 1-4.

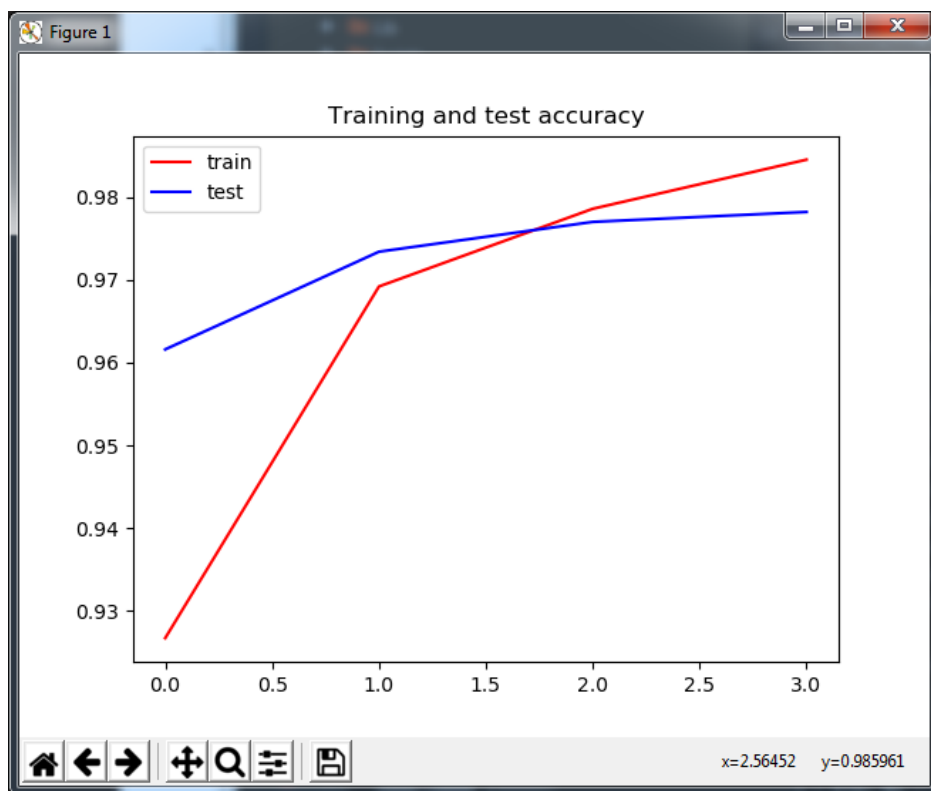


Рисунок 1 – График точности модели с оптимизатором RMSprop,  
 $\text{learning\_rate} = 0.001$

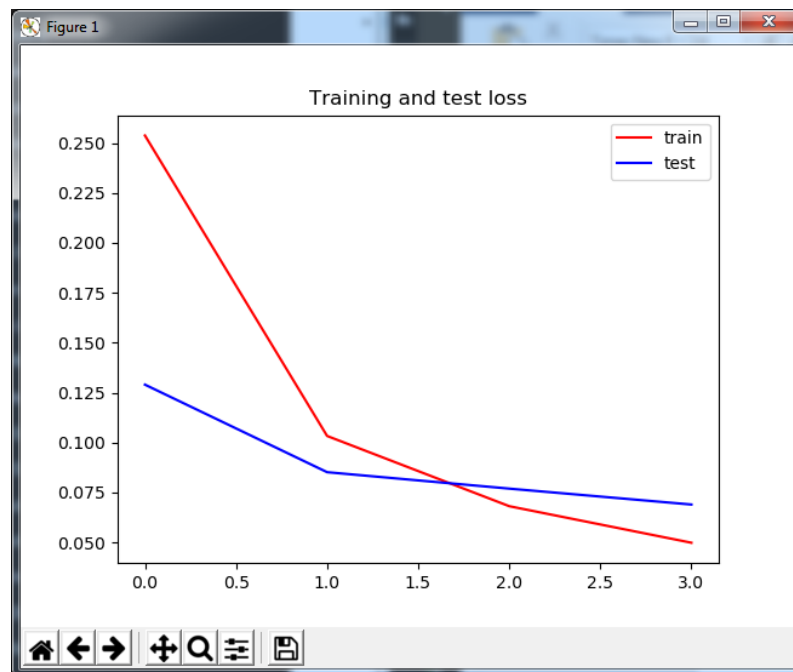


Рисунок 2 – График потерь модели с оптимизатором RMSprop,  
 $\text{learning\_rate} = 0.001$

С данным параметром и архитектурой модели была достигнута точность 0.9782.

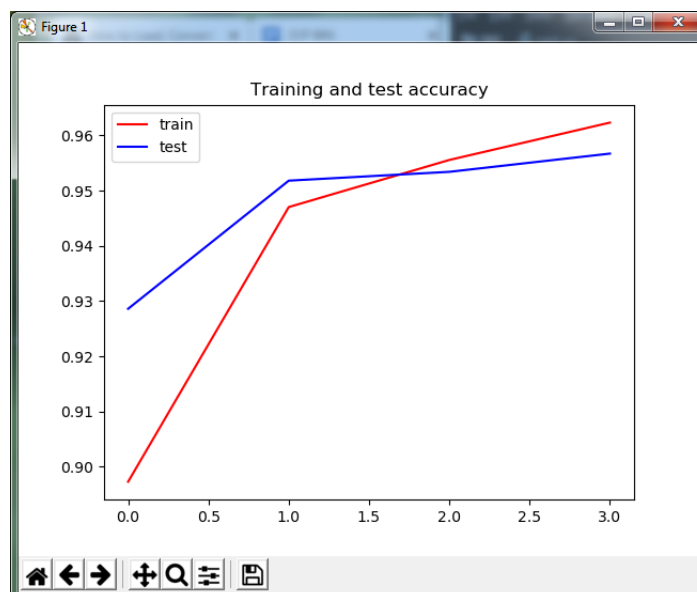


Рисунок 3 – График точности модели с оптимизатором RMSprop,  
 $\text{learning\_rate} = 0.03$

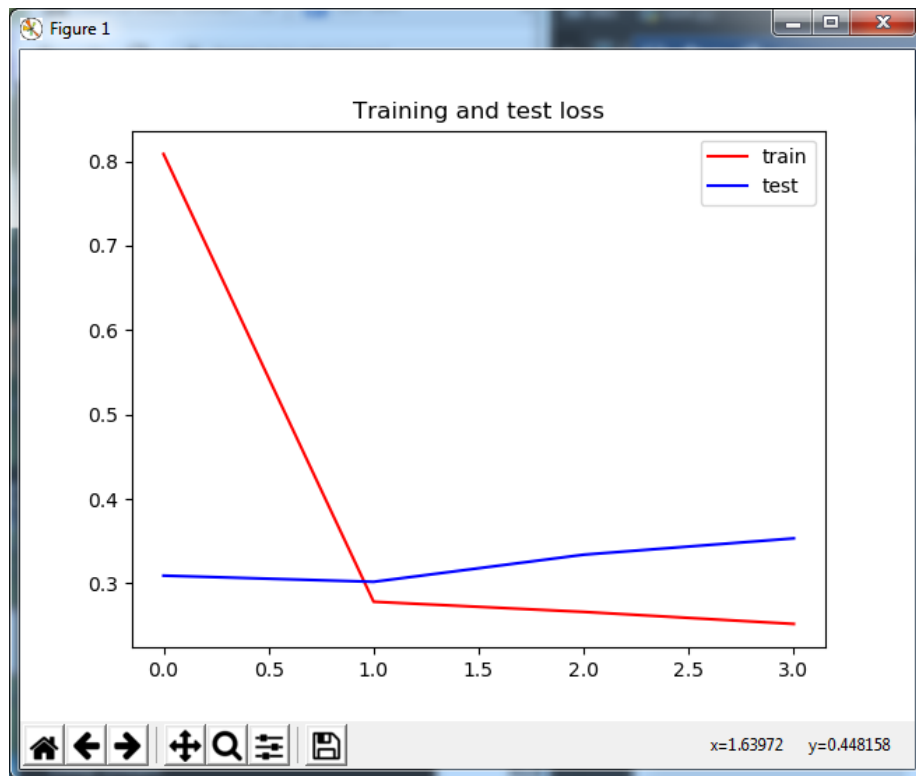


Рисунок 4 – График потерь модели с оптимизатором RMSprop,  $\text{learning\_rate} = 0.03$

С данным параметром и архитектурой модели была достигнута точность 0.9534.

### **Оптимизатор Adam.**

В данном оптимизаторе проведены исследования с 2-мя наборами параметров:

- 1) Стандартный:  $\text{learning\_rate}=0.001$ ,  $\text{beta}_1=0.9$ ,  $\text{beta}_2=0.999$
- 2)  $\text{learning\_rate}=0.002$ ,  $\text{beta}_1=0.7$ ,  $\text{beta}_2=0.9$

Результаты исследований представлены на рис. 5-8

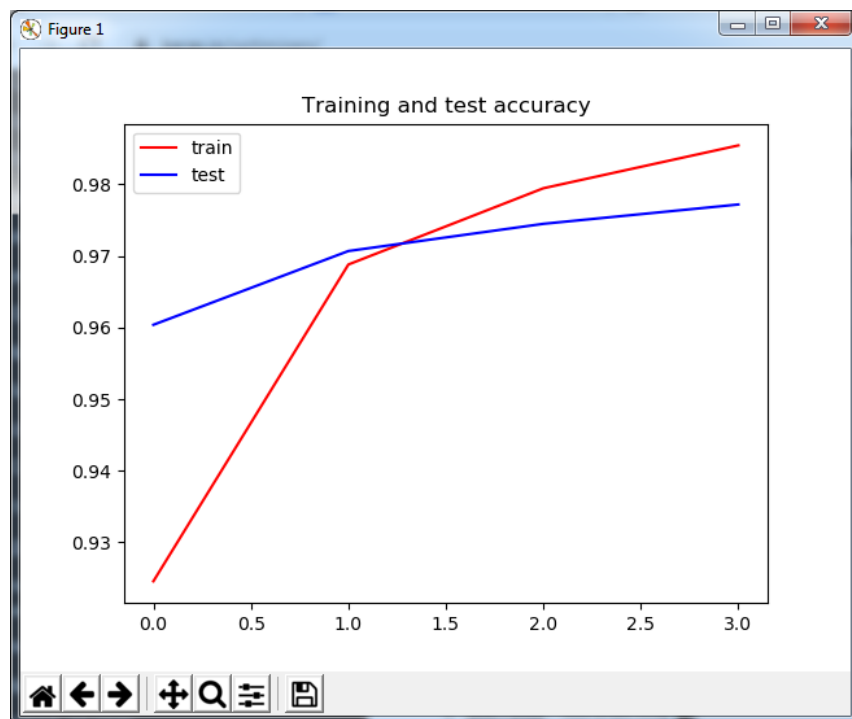


Рисунок 5 – График точности модели с оптимизатором Adam,  
 $\text{learning\_rate} = 0.001$ ,  $\text{beta}_1=0.9$ ,  $\text{beta}_2=0.999$

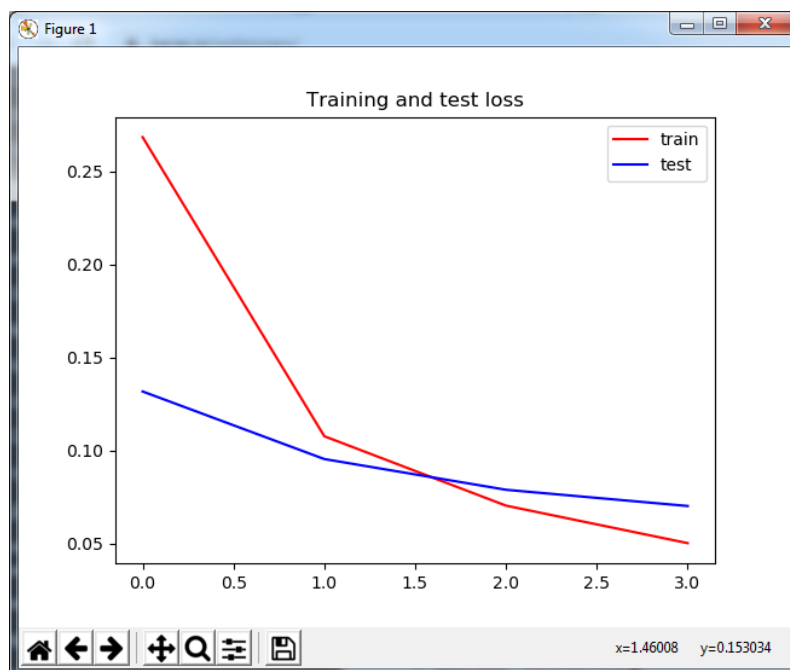


Рисунок 6 – График потерь модели с оптимизатором Adam,  
 $\text{learning\_rate} = 0.001$ ,  $\text{beta}_1=0.9$ ,  $\text{beta}_2=0.999$

С данными параметрами для оптимизатора и данной архитектурой была достигнута точность 0.9772

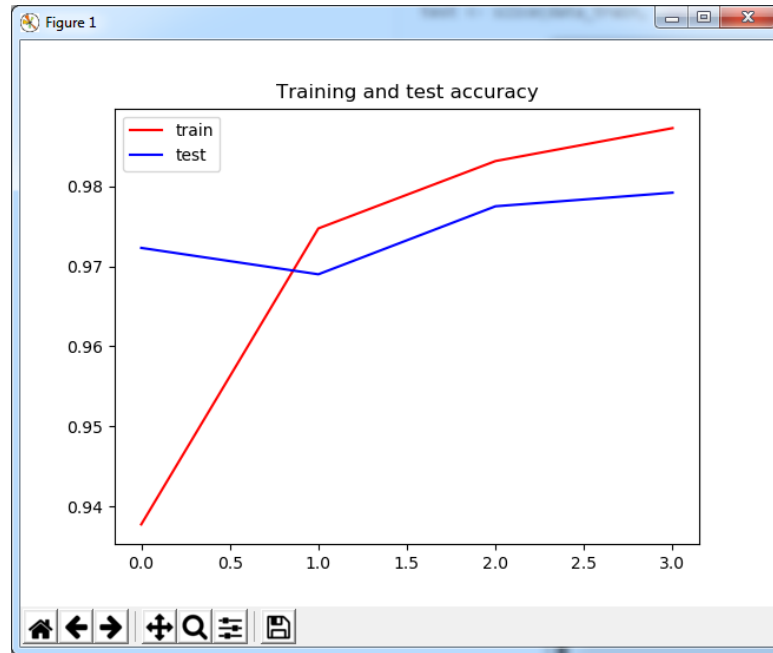


Рисунок 7 – График точности модели с оптимизатором Adam,  
 $\text{learning\_rate} = 0.002$ ,  $\text{beta}_1=0.7$ ,  $\text{beta}_2=0.9$

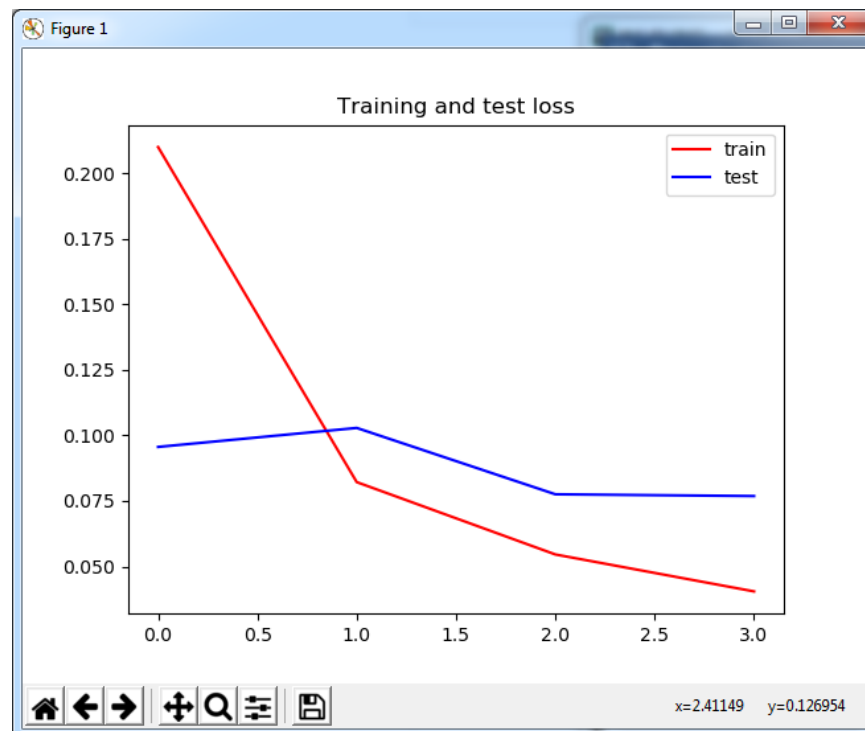


Рисунок 8 – График потерь модели с оптимизатором Adam,  
 $\text{learning\_rate} = 0.002$ ,  $\text{beta}_1=0.7$ ,  $\text{beta}_2=0.9$

С данными параметрами оптимизатора и данной архитектурой сети была достигнута точность 0.9792.

### Оптимизатор SGD.

В данном оптимизаторе проведены исследования с 2-мя наборами параметров:

- 3) Стандартный:  $\text{learning\_rate} = 0.01$ ,  $\text{momentum} = 0$
- 4)  $\text{learning\_rate} = 0.001$ ,  $\text{momentum} = 0.01$

Результаты исследований представлены на рис. 9-12

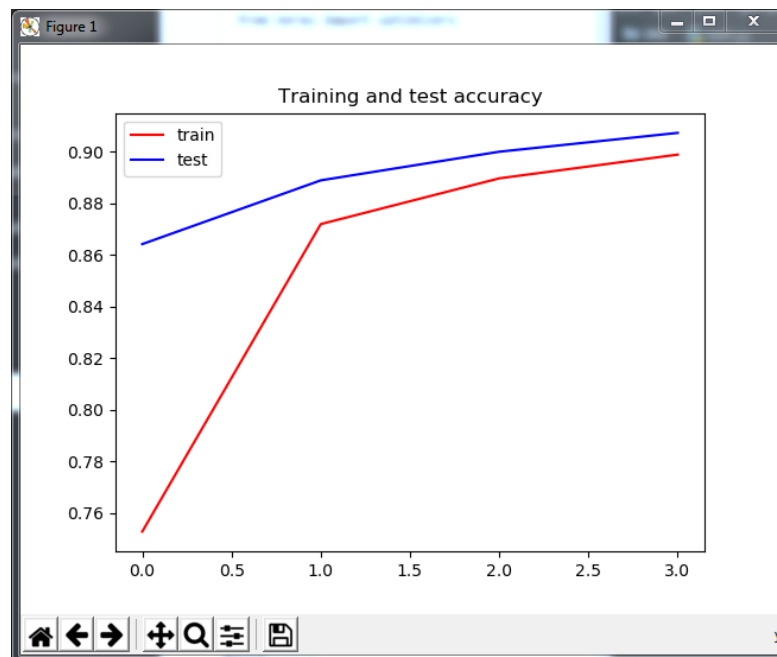


Рисунок 9 – График точности модели с оптимизатором SGD,  
 $\text{learning\_rate} = 0.01$ ,  $\text{momentum} = 0$

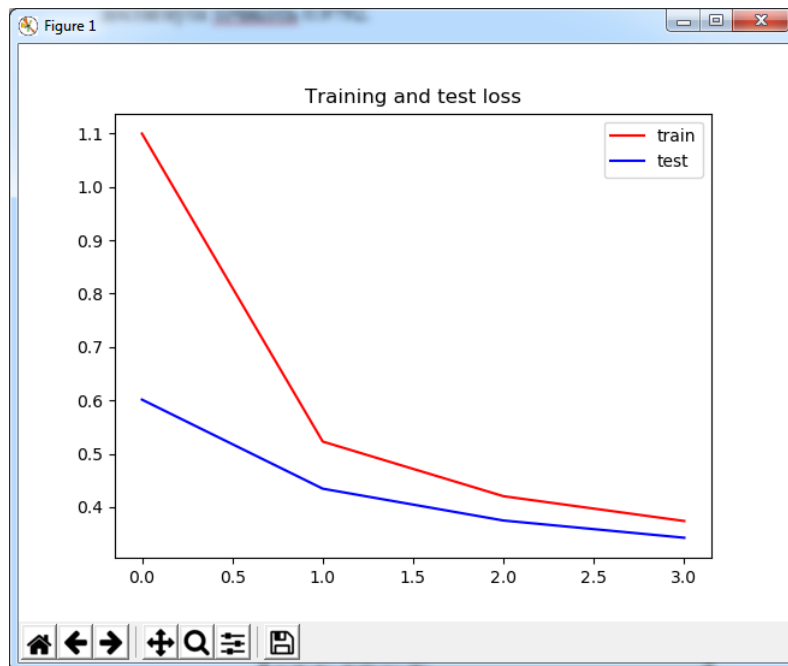


Рисунок 10 – График точности модели с оптимизатором SGD,  
 $\text{learning\_rate} = 0.01$ ,  $\text{momentum} = 0$

С данными параметрами для оптимизатора и данной архитектурой была достигнута точность 0.9074.

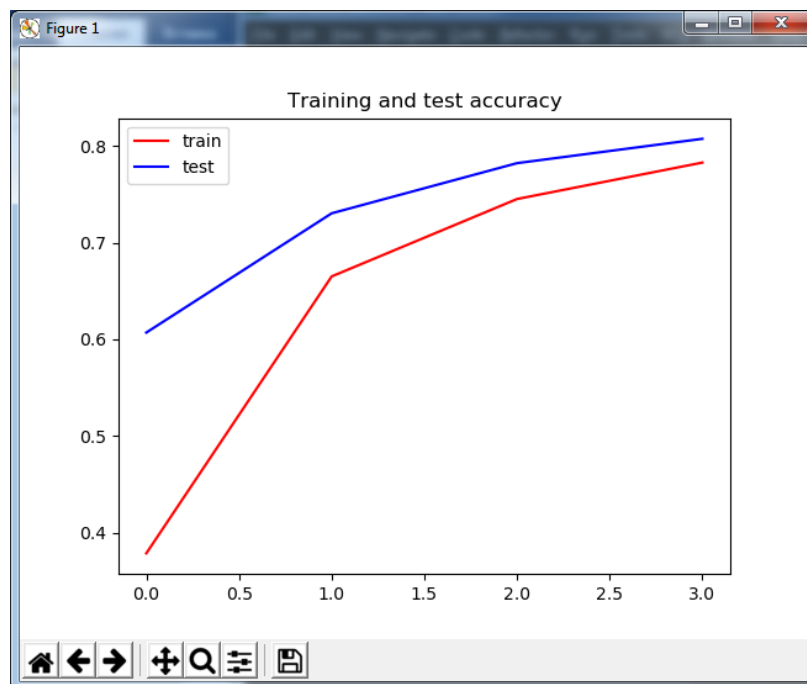


Рисунок 11 – График точности модели с оптимизатором SGD,  
 $\text{learning\_rate} = 0.001$ ,  $\text{momentum} = 0.01$



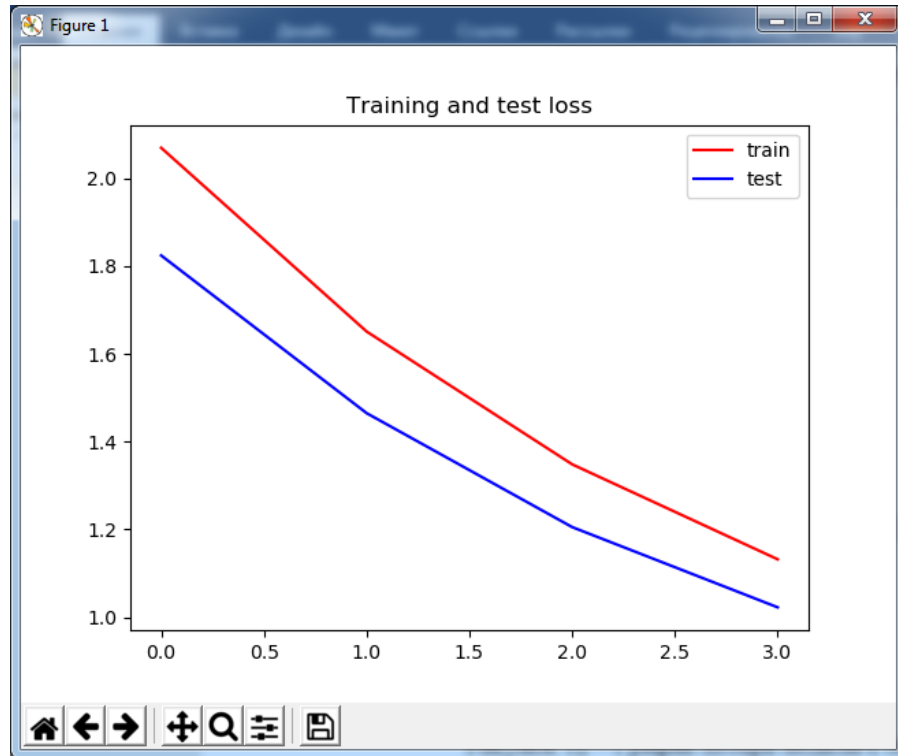


Рисунок 12 – График потерь модели с оптимизатором SGD,  
 $\text{learning\_rate} = 0.001$ ,  $\text{momentum} = 0.01$

С данными параметрами для оптимизатора и данной архитектурой была достигнута точность 0.8074.

### **Выводы.**

В ходе выполнения данной работы была создана сеть для распознавания рукописных символов, а также исследованы разные оптимизаторы. Лучше всего в данной задаче показал себя оптимизатор RMSprop со стандартными параметрами, т.к. была достигнута точность 0.9782.

## ПРИЛОЖЕНИЯ

### ПРИЛОЖЕНИЕ А: КОД ПРОГРАММЫ

```
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
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()
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images / 255.0
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images / 255.0
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
network = Sequential()
network.add(Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(Dense(10, activation='softmax'))

def run_research(optimizer):
    network.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
    history = network.fit(train_images, train_labels, epochs=4,
batch_size=128, validation_data=(test_images,
test_labels))
    test_loss, test_acc = network.evaluate(test_images, test_labels)
    print('test_acc:', test_acc)
    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.show()
    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.legend()
```

```
plt.show()  
plt.clf()  
  
run_research(optimizers.SGD(learning_rate=0.001,momentum=0.01))
```