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. 7381 _____ Трушников А.П.

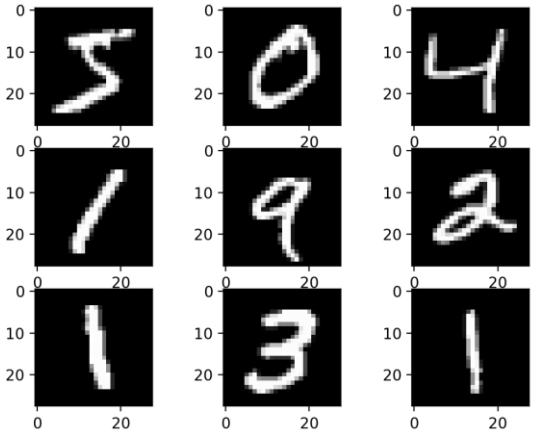
Преподаватель _____ Жукова Н.А..

Санкт-Петербург

2020

Цель работы.

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



Набор данных содержит 60,000 изображений для обучения и 10,000 изображений для тестирования.

Задачи.

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

Ход работы.

1. Найти архитектуру сети, при которой точность классификации будет не менее 95%.

   Была найдена архитектура, которая даёт точность $\approx 98\%$, параметры которой представлены в таблице 1–2.

Таблица 1

Оптимизатор	Функция потерь	Метрика качества обучения сети	Число эпох	batch_size
Adam(learning_rate=0.001)	Categorical_crossentropy	accuracy	7	110

Таблица 2

Номер слоя	Описание слоя
1	Flatten() – слой, преобразующий формат изображения из двумерного массива в одномерный. Слой извлекает строки пикселей из изображения и выстраивает их в один ряд. Этот слой не имеет параметров для обучения; он только переформатирует данные.
2	Dense(800, activation='relu')
3	Dense(10, activation='softmax')

Графики точности и ошибки представлены на рис.1–2.

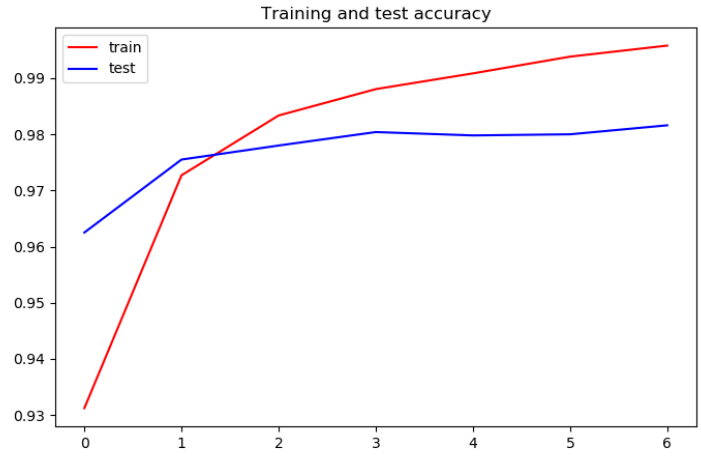


Рисунок 1 – График точности построенной сети

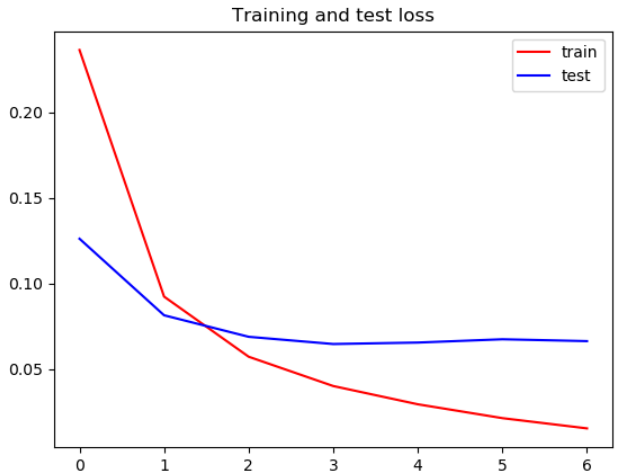


Рисунок 2 – График ошибок построенной сети

2. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения.

   В таблице 3 представлены параметры оптимизаторов с их описанием.

Таблица 3

Параметр	Описание

learning_rate	скорость обучения – влияет на то, как сильно веса изменяются каждый раз во время обучения
momentum	ускоряет оптимизатор
rho	коэффициент затухания скользящего среднего значения градиента

а) Протестируем оптимизатор SGD.

При значении $momentum = 0.0$ переберём значение $learning\_rate$ от 0.1 до 0.001. Результаты точности представлены в таблице 4. Графики точности и ошибок представлены на рис.3–8.

Таблица 4

$learning\_rate$	Значение точности
0.1	0.9706
0.01	0.9268
0.001	0.8579

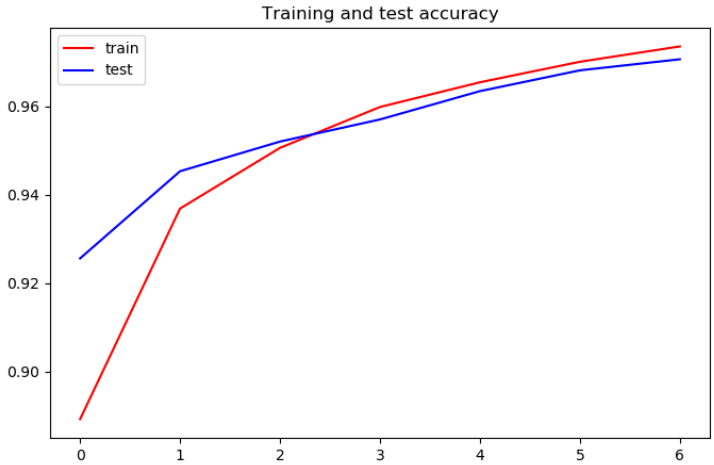


Рисунок 3 – График точности построенной сети при $learning\_rate = 0.1$

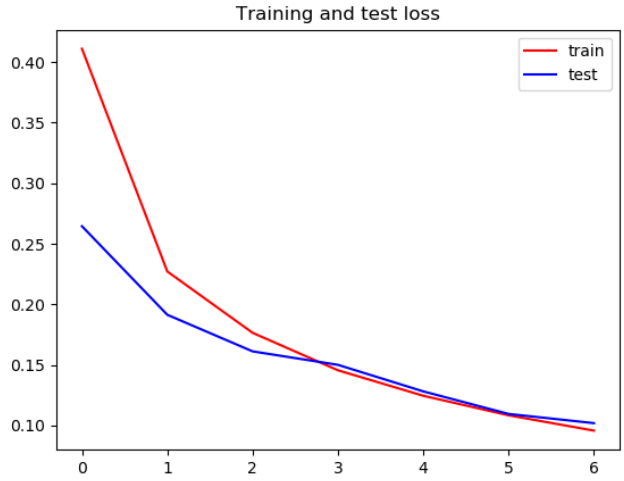


Рисунок 4 – График ошибок построенной сети при $learning\_rate = 0.1$

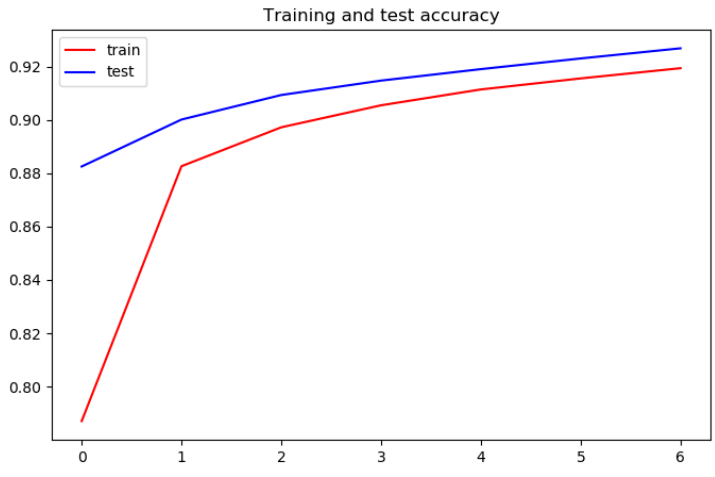


Рисунок 5 – График точности построенной сети при $learning\_rate = 0.01$

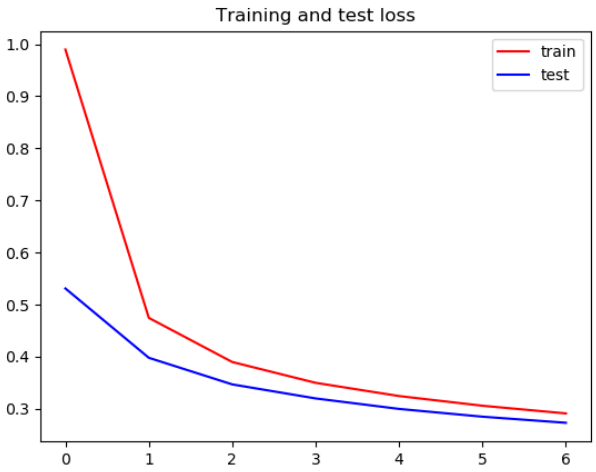


Рисунок 6 – График ошибок построенной сети при $learning\_rate = 0.01$

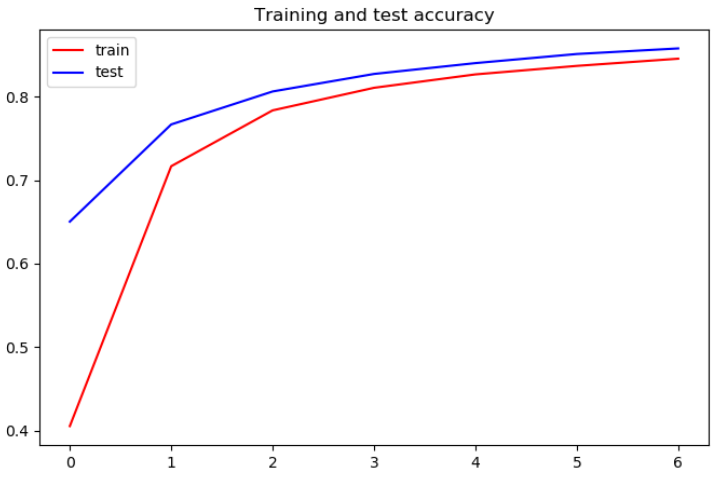


Рисунок 7 – График точности построенной сети при $learning\_rate = 0.001$

Рисунок 8 – График ошибок построенной сети при $learning\_rate = 0.001$

С уменьшением скорости обучения модели уменьшается и ее точность. Теперь при значении $learning\_rate = 0.01$ переберём значение *momentum* 0.1, 0.5, 0.9. Результаты точности представлены в таблице 5. Графики точности и ошибок представлены на рис.3–8.

Таблица 5

momentum	Значение точности
0.1	0.9267
0.5	0.9409
0.9	0.9714

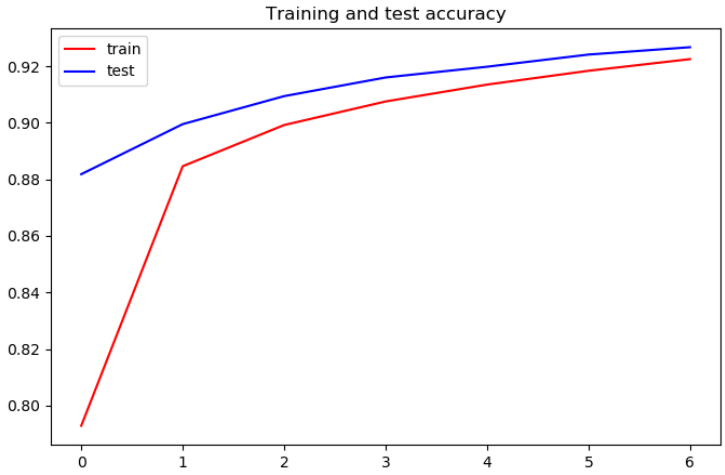


Рисунок 9 – График точности построенной сети при $momentum = 0.1$

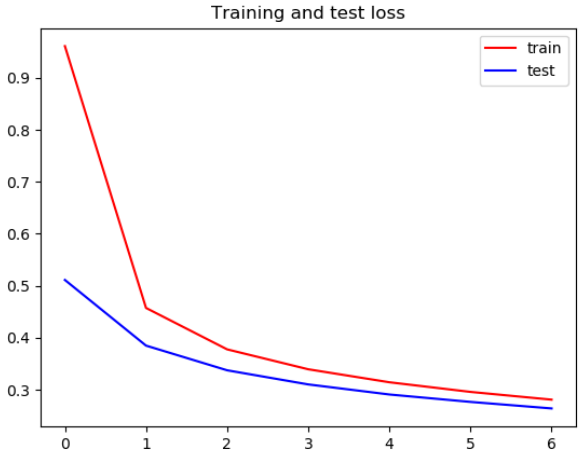


Рисунок 10 – График ошибок построенной сети при $momentum = 0.1$

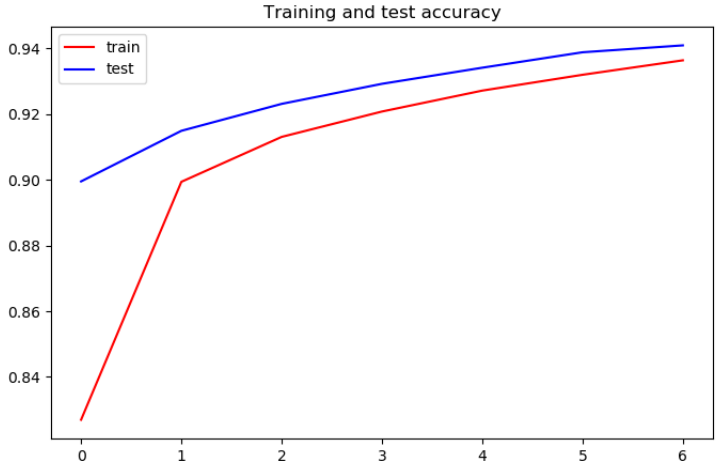


Рисунок 11 – График точности построенной сети при $momentum = 0.5$

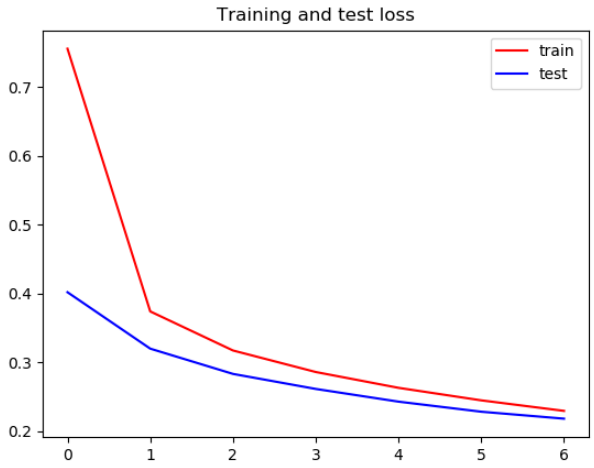


Рисунок 12 – График ошибок построенной сети при $momentum = 0.5$

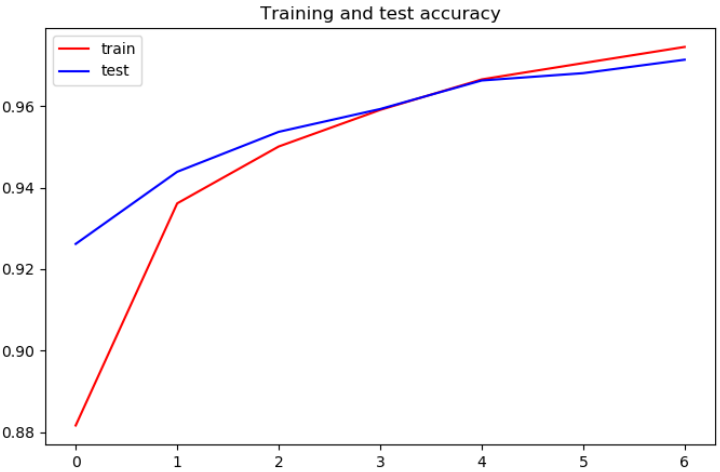


Рисунок 13 – График точности построенной сети при $momentum = 0.9$

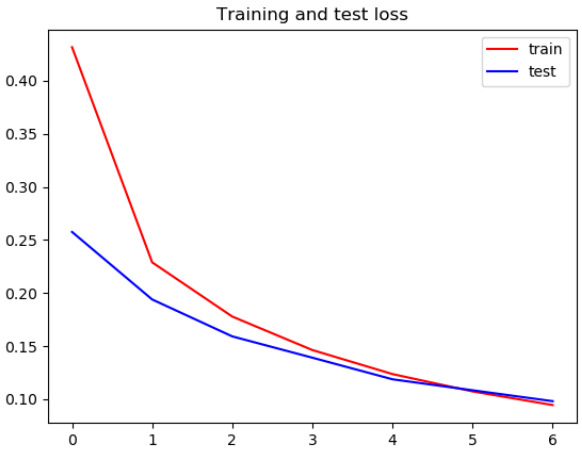


Рисунок 14 – График ошибок построенной сети при $momentum = 0.9$

С увеличением значения *momentum* увеличивается точность модели.

b) Протестируем оптимизатор RMSprop.

При значении $rho = 0.9$ переберём значение $learning\_rate$ от 0.1 до 0.001. Результаты точности представлены в таблице 6. Графики точности и ошибок представлены на рис.15–20.

Таблица 6

learning _ rate	Значение точности
0.1	0.857
0.01	0.9756
0.001	0.9787

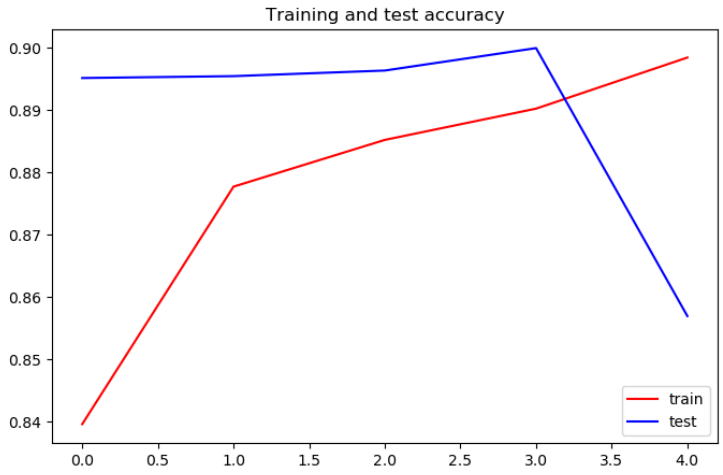


Рисунок 15 – График точности построенной сети при $learning\_rate = 0.1$

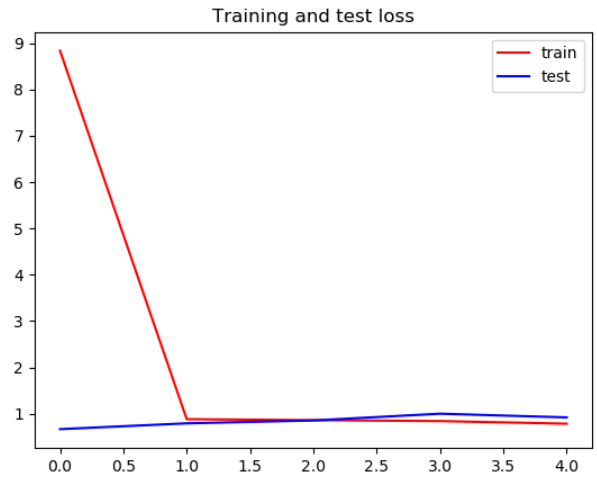


Рисунок 16 – График ошибок построенной сети при $learning\_rate = 0.1$

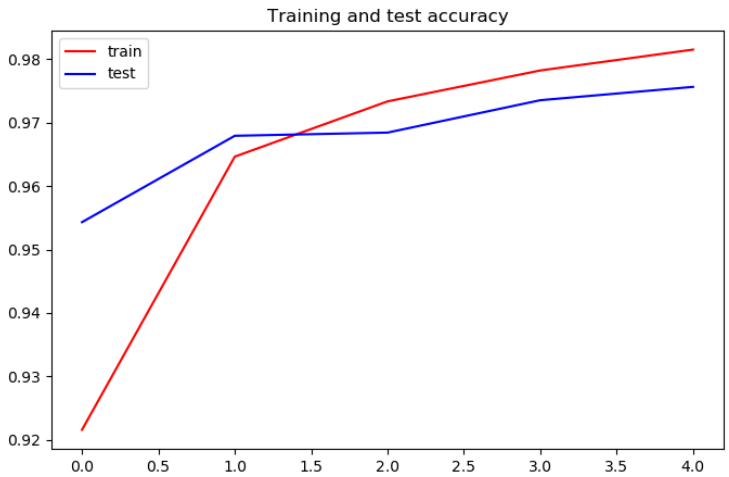


Рисунок 17 – График точности построенной сети при $learning\_rate = 0.01$

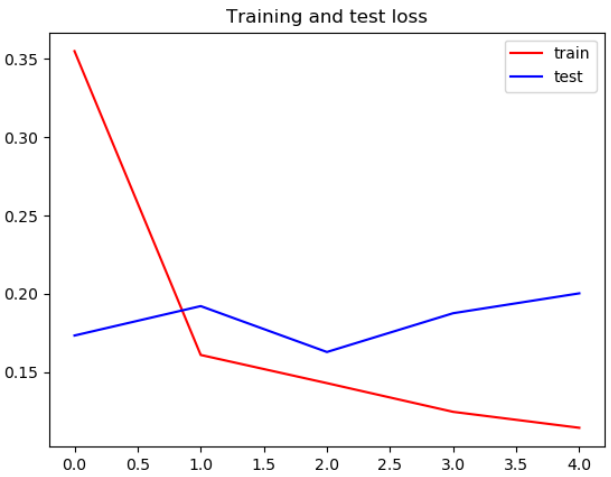


Рисунок 18 – График ошибок построенной сети при $learning\_rate = 0.01$

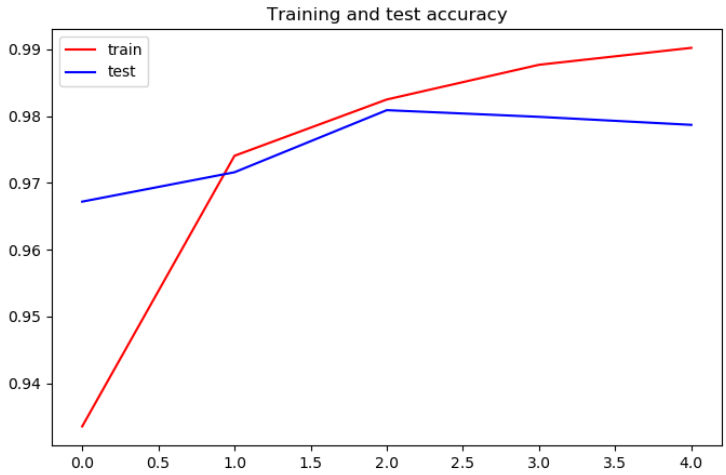


Рисунок 19 – График точности построенной сети при $learning\_rate = 0.001$

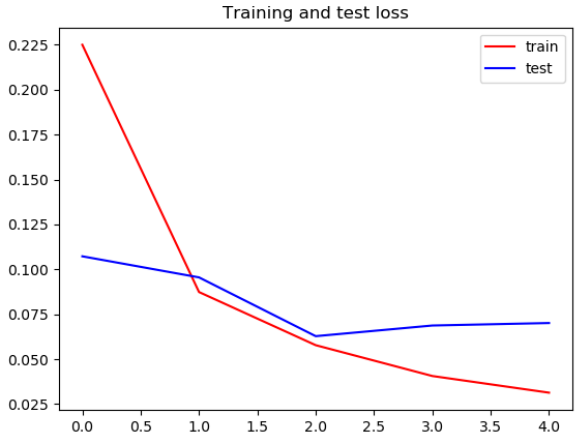


Рисунок 20 – График ошибок построенной сети при $learning\_rate = 0.001$

С уменьшением значения $learning\_rate$ увеличивается точность модели. Теперь при значении $learning\_rate = 0.001$ переберём значения rho 0.1, 0.5, 0.9. Результаты точности представлены в таблице 7. Графики точности и ошибок представлены на рис.21–26.

Таблица 7

rho	Значение точности
0.1	0.9737
0.5	0.9774
0.9	0.9816

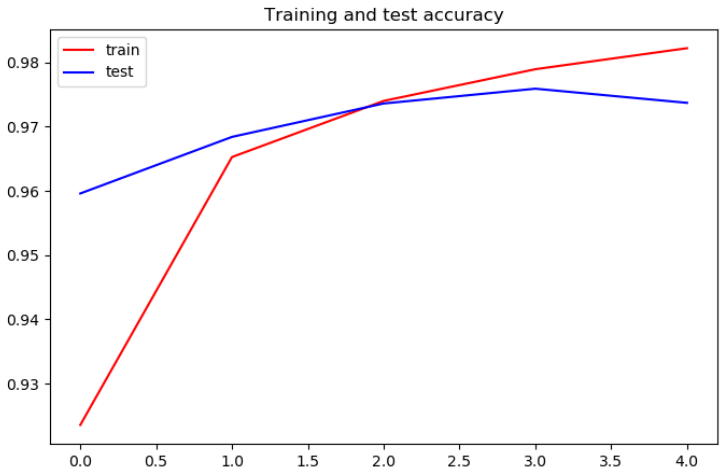


Рисунок 21 – График точности построенной сети при $rho = 0.1$

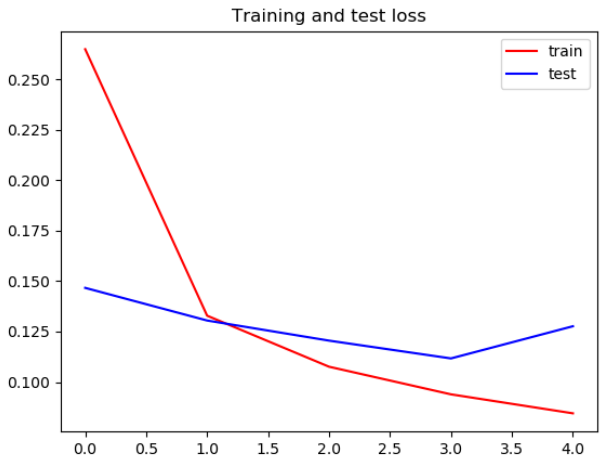


Рисунок 22 – График ошибок построенной сети при $rho = 0.1$

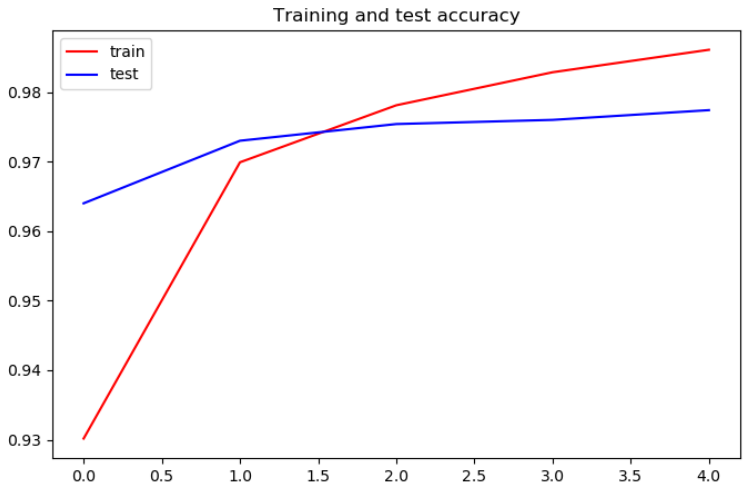


Рисунок 23 – График точности построенной сети при $rho = 0.5$

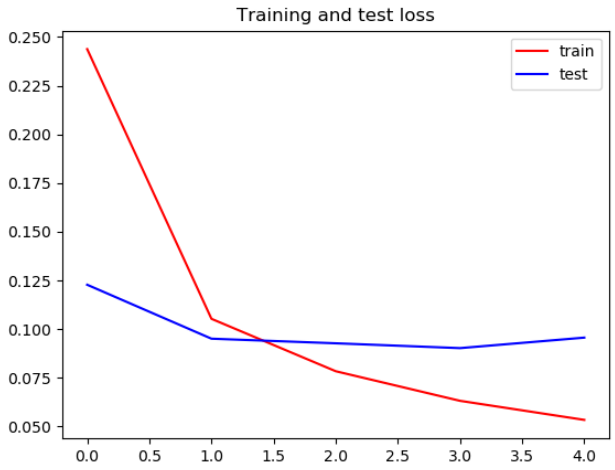


Рисунок 24 – График ошибок построенной сети при $rho = 0.5$

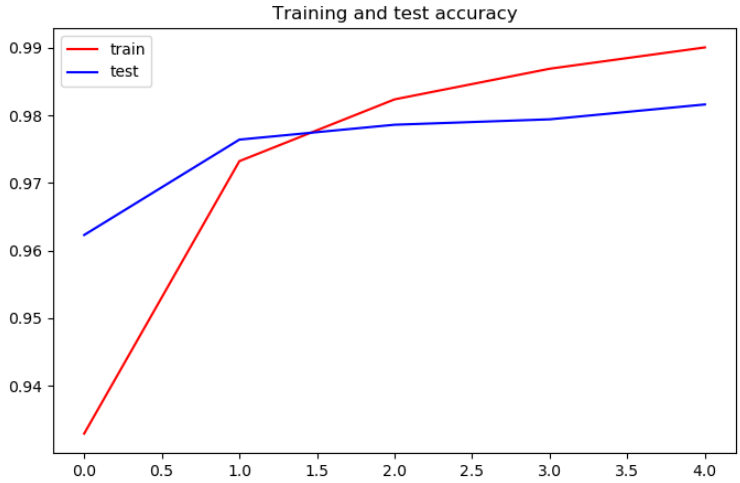


Рисунок 25 – График точности построенной сети при $rho = 0.9$

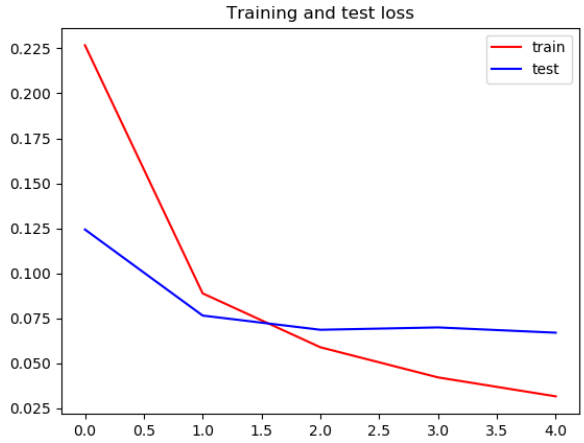


Рисунок 26 – График ошибок построенной сети при $rho = 0.9$

С увеличением значения *rho* увеличивается точность модели.

c) Протестируем оптимизатор Adam.

Переберём значение *learning _rate* от 0.1 до 0.001. Результаты точности представлены в таблице 8. Графики точности и ошибок представлены на рис.27–32.

learning _rate	Значение точности
0.1	0.8609
0.01	0.9654
0.001	0.9814

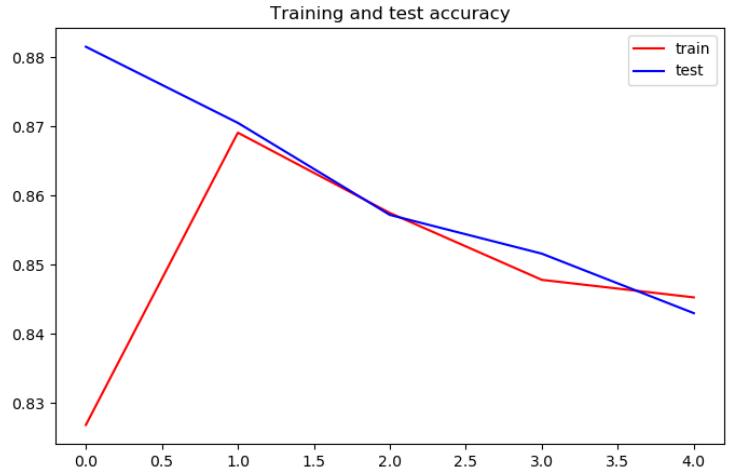


Рисунок 27 – График точности построенной сети при $learning\_rate = 0.1$

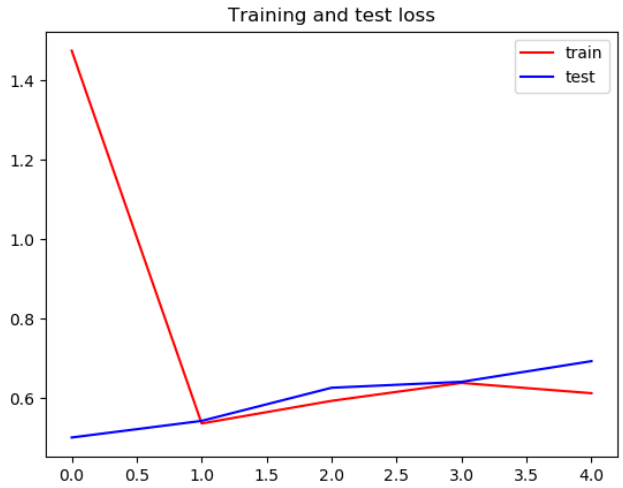


Рисунок 28 – График ошибок построенной сети при $learning\_rate = 0.1$

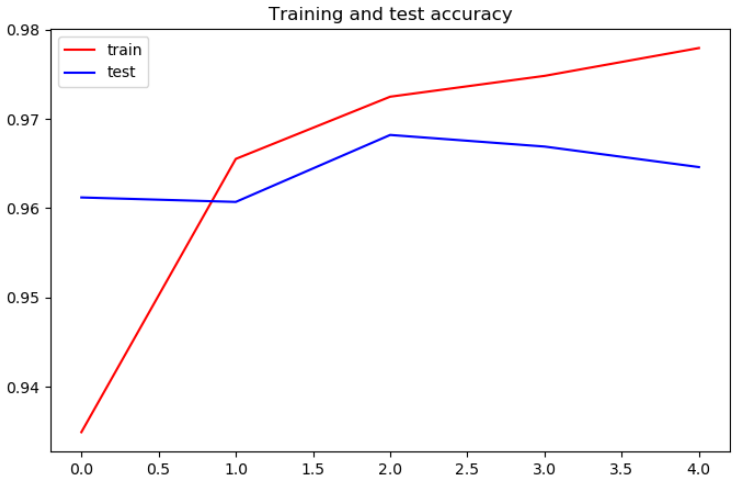


Рисунок 29 – График точности построенной сети при $learning\_rate = 0.01$

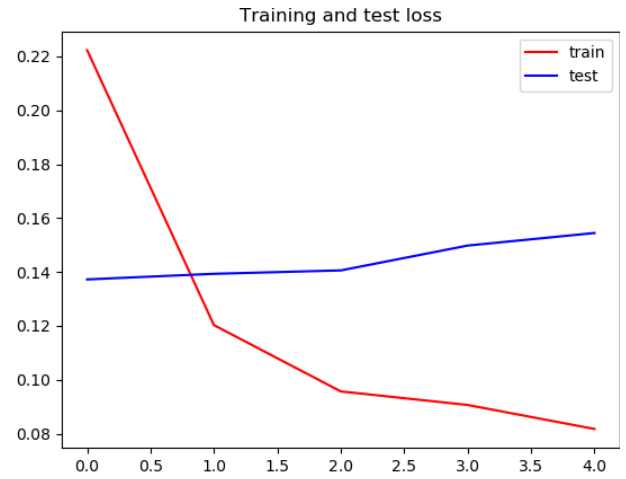


Рисунок 30 – График ошибок построенной сети при $learning\_rate = 0.01$

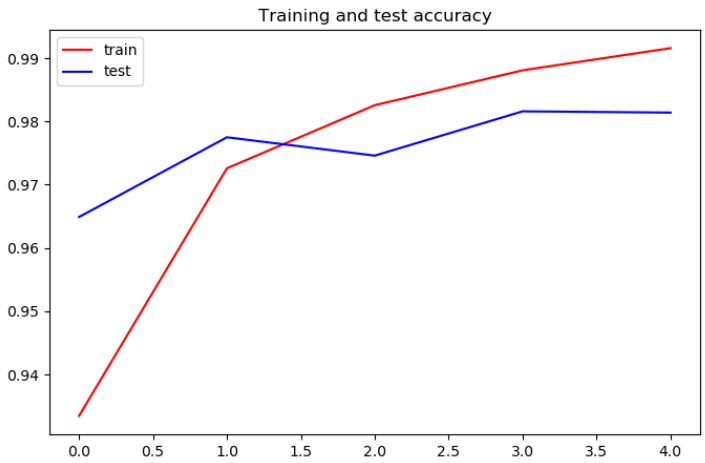


Рисунок 31 – График точности построенной сети при $learning\_rate = 0.001$

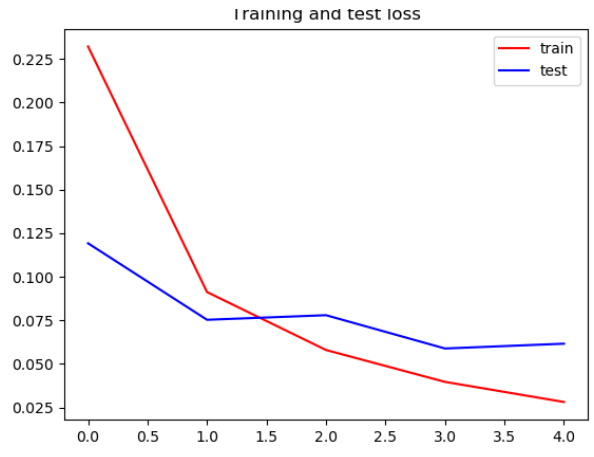


Рисунок 32 – График ошибок построенной сети при $learning\_rate = 0.001$

С увеличением значения $learning\_rate$ увеличивается точность модели.

3. Написать функцию, которая позволит загружать пользовательское изображение не из датасета.

   Была написана функция, в которую необходимо передать путь к изображению и модель, которая будет определять ее принадлежность классу. Функция представлена на рис. 33.

```
def classify_img(path, model):
    x = image.img_to_array(image.load_img(path, target_size=(28, 28), color_mode="grayscale")).reshape(1, 784)
    return np.argmax(model.predict((255 - x) / 255))
```

Рисунок 33 – Функция

Протестируем ее:

a) Подадим на вход изображение единицы, представленное на рис.34. Результат работы программы представлен на рис. 35.

Рисунок 34 – Изображение цифры 1

```
(array([[0.00284781, 0.52586925, 0.03160701, 0.06761608, 0.0393732 ,
        0.03098067, 0.02009719, 0.05931531, 0.19948085, 0.02281272]],
      dtype=float32), 1)
```

Рисунок 35 – Результат раюоты программы

b) Подадим на вход изображение двойки, представленное на рис.36. Результат работы программы представлен на рис. 37.

Рисунок 36 – Изображение цифры 2

```
(array([[5.2306576e-10, 1.7593628e-06, 9.9939084e-01, 6.0739776e-04,
        7.0108553e-18, 2.2492830e-10, 7.3133765e-15, 5.9494267e-09,
        4.5610360e-09, 2.4810006e-11]], dtype=float32), 2)
```

Рисунок 37 – Результат раюоты программы

c) Подадим на вход изображение восьмёрки, представленное на рис.38. Результат работы программы представлен на рис. 39.

Рисунок 38 – Изображение цифры 8

```
(array([[9.2991342e-10, 5.0443436e-09, 1.0372619e-07, 8.0463421e-03,
        1.1882680e-06, 1.7961976e-05, 8.2933991e-07, 5.0849081e-08,
        9.9193341e-01, 8.5715612e-08]], dtype=float32), 8)
```

Рисунок 39 – Результат раюоты программы

Сеть определила изображения цифр верно.

Выводы.

В ходе выполнения данной работы было изучено представление графических данных. Была построена и протестирована на пользовательских изображениях сеть с точностью ≈98%.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

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


def classify_img(path, model):
    x = image.img_to_array(image.load_img(path, target_size=(28, 28),
color_mode="grayscale")).reshape(1, 784)
    return model.predict((255 - x) / 255),np.argmax(model.predict((255 - x) /
255))


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)

model = Sequential()
model.add(Flatten())
model.add(Dense(800, activation='relu'))
model.add(Dense(10, activation='softmax'))

optimizer = optimizers.Adam(learning_rate=0.001)
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

h = model.fit(train_images, train_labels, epochs=5, batch_size=100,
validation_data=(test_images, test_labels), verbose=0)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
print(np.argmax(test_labels[0]),np.argmax(model.predict(test_images)[0]))

print(classify_img("8.jpg", model))

plt.figure(1, figsize=(8, 5))
plt.title("Training and test accuracy")
plt.plot(h.history['acc'], 'r', label='train')
plt.plot(h.history['val_acc'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()
```

```
plt.figure(1, figsize=(8, 5))
plt.title("Training and test loss")
plt.plot(h.history['loss'], 'r', label='train')
plt.plot(h.history['val_loss'], 'b', label='test')
plt.legend()
plt.show()
plt.clf()
```