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

Студент гр. 7383 _____ Сычевский 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).

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, позволяющую загружать изображение пользователя и классифицировать его

Ход работы.

Для исследования была разработана и использована программа. Код программы приведен в приложении А.

Исследуем поведение сети в зависимости от оптимизаторов. Результаты представлены на рисунках 1-4.

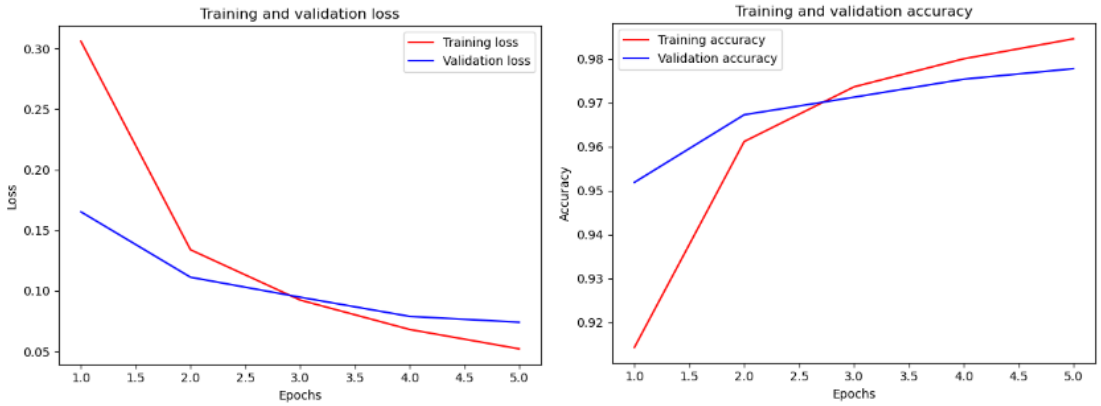

Рисунок 1 – Графики потерь и точности с оптимизатором Adam

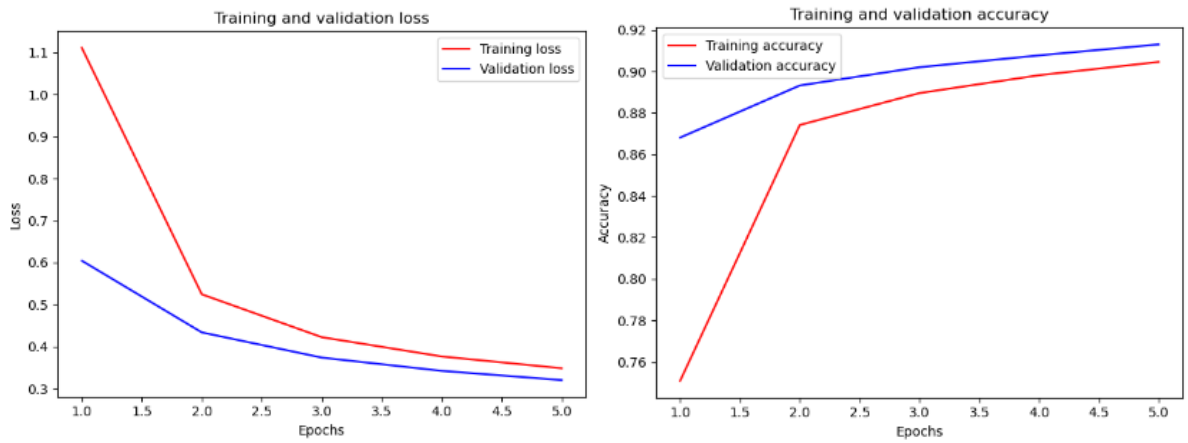

Рисунок 2 – Графики потерь и точности с оптимизатором SGD

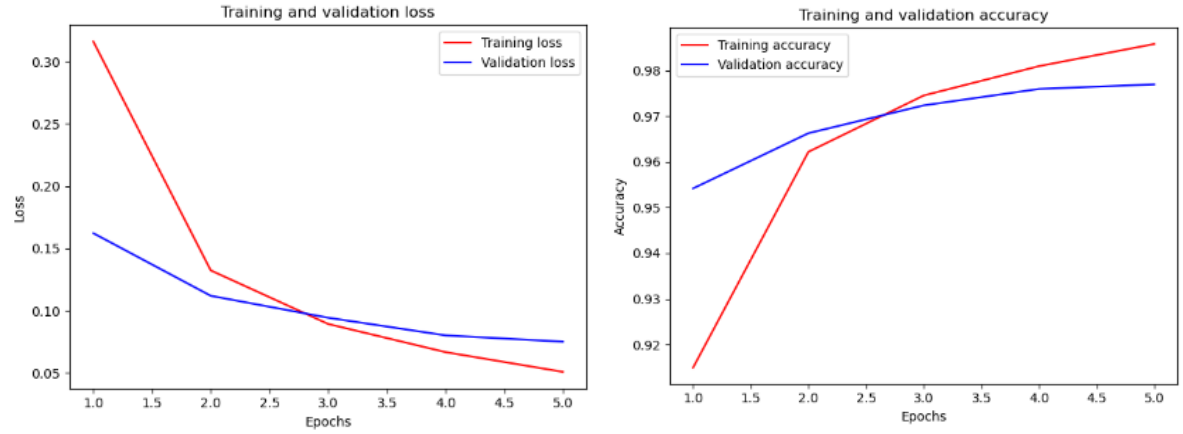

Рисунок 3 – Графики потерь и точности с оптимизатором Nadam

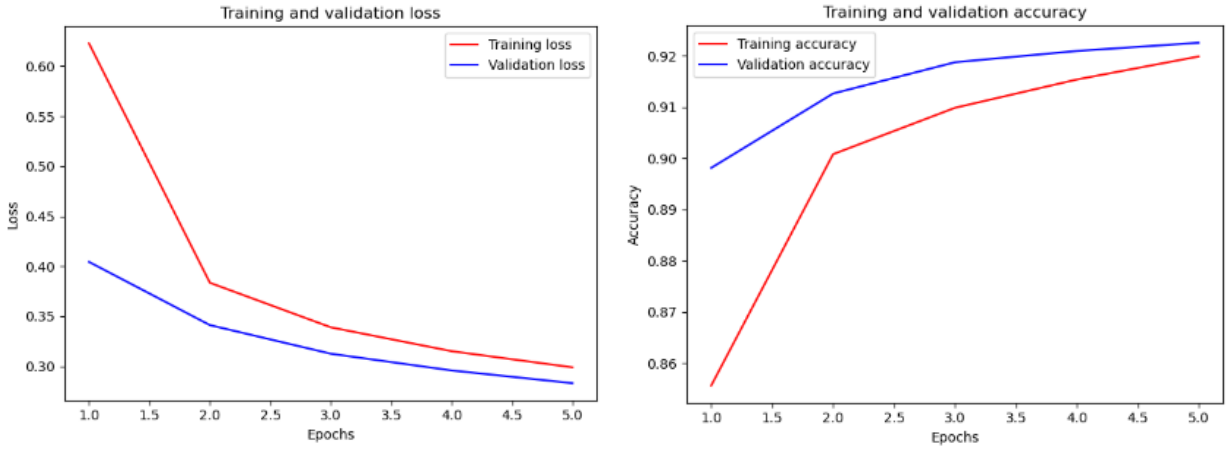

Рисунок 4 – Графики потерь и точности с оптимизатором Adagrad

Adam показал самую высокую точность: 97,78%, так что выберем его. Далее рассмотрим различные значение параметра скорости обучения оптимизатора Adam. Результаты показаны на рисунках 5-8.

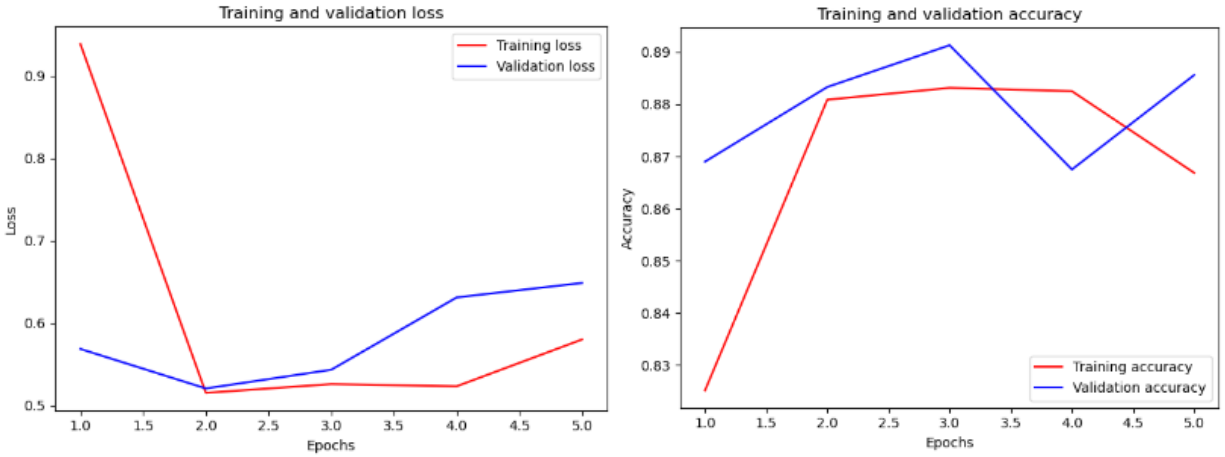

Рисунок 5 – Графики потерь и точности с learning_rate = 0.1

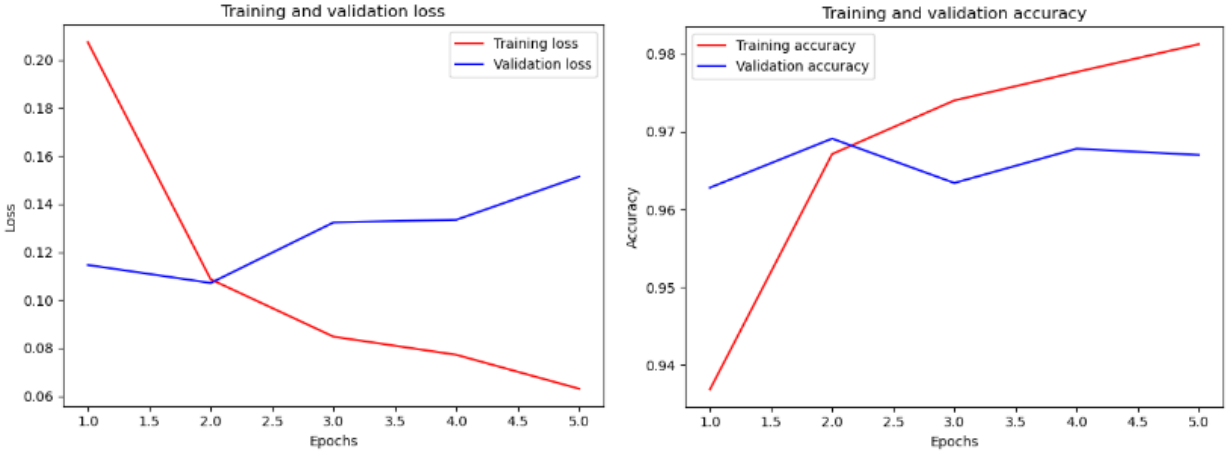

Рисунок 6 – Графики потерь и точности с learning_rate = 0.01

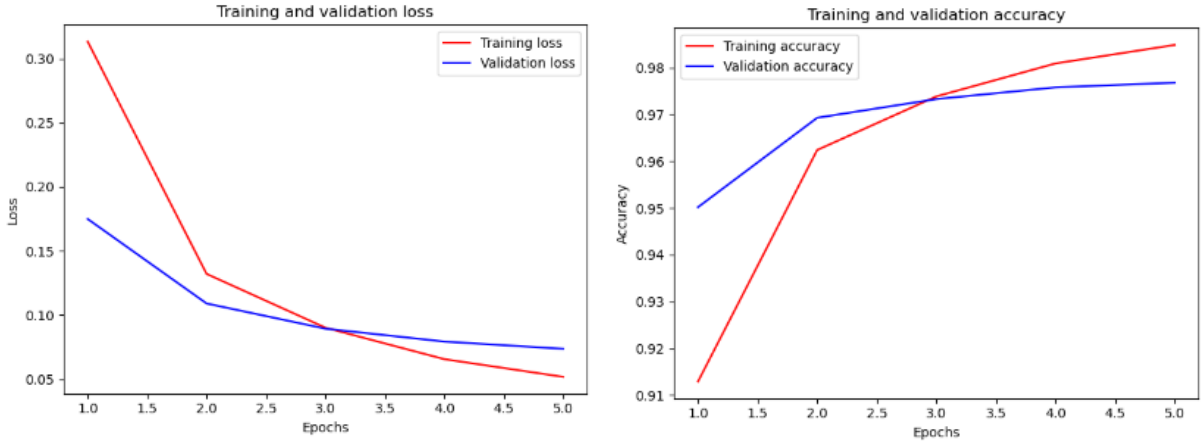

Рисунок 7 – Графики потерь и точности с learning_rate = 0.001

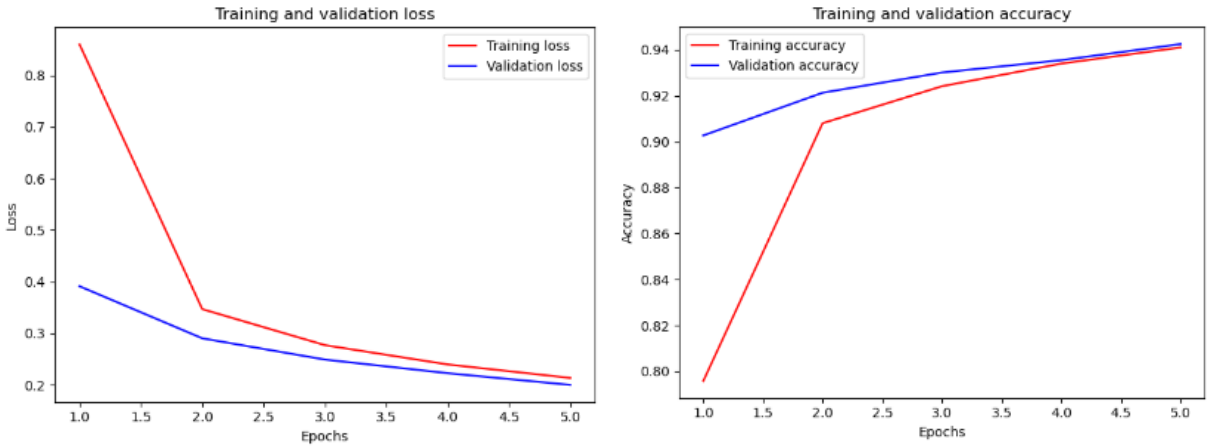

Рисунок 8 – Графики потерь и точности с learning_rate = 0.0001

После значения learning_rate = 0.0001 значения не сильно меняются, следовательно можно оставить это значение.

Так же была реализована функция loadImage для загрузки пользовательского изображения.

Выводы.

В ходе выполнения данной работы была создана сеть, которая может распознавать рукописные символы. Были рассмотрены различные оптимизаторы и их параметры, а так же их влияние на результат. Была реализована функция для загрузки своего изображения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import numpy as np
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
from keras.utils import to_categorical
import matplotlib.pyplot as plot
from tensorflow.keras import optimizers
import tensorflow as tf
from PIL import Image


def loadImage(path):
    image = Image.open(path)
    image = image.resize((IMAGE_HEIGHT, IMAGE_WIDTH))
    image = np.dot(np.asarray(image), np.array([1 / 3, 1 / 3, 1 / 3]))
    image /= 255
    image = 1 - image
    image = image.reshape((1, IMAGE_HEIGHT * IMAGE_WIDTH))
    return image

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

model.compile(optimizer=optimizers.Adam(learning_rate=0.0001),loss='ca
tegorical_crossentropy', metrics=['accuracy'])
H = model.fit(train_images, train_labels, epochs=5, batch_size=128,
validation_data=(test_images, test_labels))

loss = H.history['loss']
val_loss = H.history['val_loss']
epochs = range(1, len(loss) + 1)
plot.plot(epochs, loss, 'r', label='Training loss')
plot.plot(epochs, val_loss, 'b', label='Validation loss')
plot.title('Training and validation loss')
plot.xlabel('Epochs')
plot.ylabel('Loss')
```

```
plot.legend()
plot.show()
plot.clf()

acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
plot.plot(epochs, acc, 'r', label='Training accuracy')
plot.plot(epochs, val_acc, 'b', label='Validation accuracy')
plot.title('Training and validation accuracy')
plot.xlabel('Epochs')
plot.ylabel('Accuracy')
plot.legend()
plot.show()

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
print('test_loss', test_loss)
```