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

# ОТЧЕТ

**по лабораторной работе №4**

**по дисциплине «Искуссенные нейронные сети»**

**Тема: «Распознавание рукописных символов»**

Студент гр. 7382 _____ Находько А.Ю.

Преподаватель _____ Жукова Н.В.

Санкт-Петербург

2020

**Цель работы.**

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

**Задачи работы.**

•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

**Требования работы.**

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

**Основные теоретические положения.**

Искусственные нейронные сети - совокупность моделей, которые представляют собой сеть элементов - искусственных нейронов, связанных между собой синаптическими соединениями.

Нейронные сети используются как среда, в которой осуществляется адаптивная настройка параметров дискриминантных функций. Настройка происходит при последовательном предъявлении обучающих выборок образов из разных классов. Обучение - такой выбор параметров нейронной сети, при котором сеть лучше всего справляется с поставленной проблемой.

**Нейрон** — элемент, преобразующий входной сигнал по функции.

**Сумматор** — элемент, осуществляющий суммирование сигналов поступающих на его вход.

**Синапс —** элемент, осуществляющий линейную передачу сигнала.

**Экспериментальные результаты.**

В ходе выполнения лабораторной работы была создана и обучена модель нейронной сети, код которо представлен в приложении А.

Проведём исследование как на процесс обучения влияют различные оптимизаторы и их параметры.

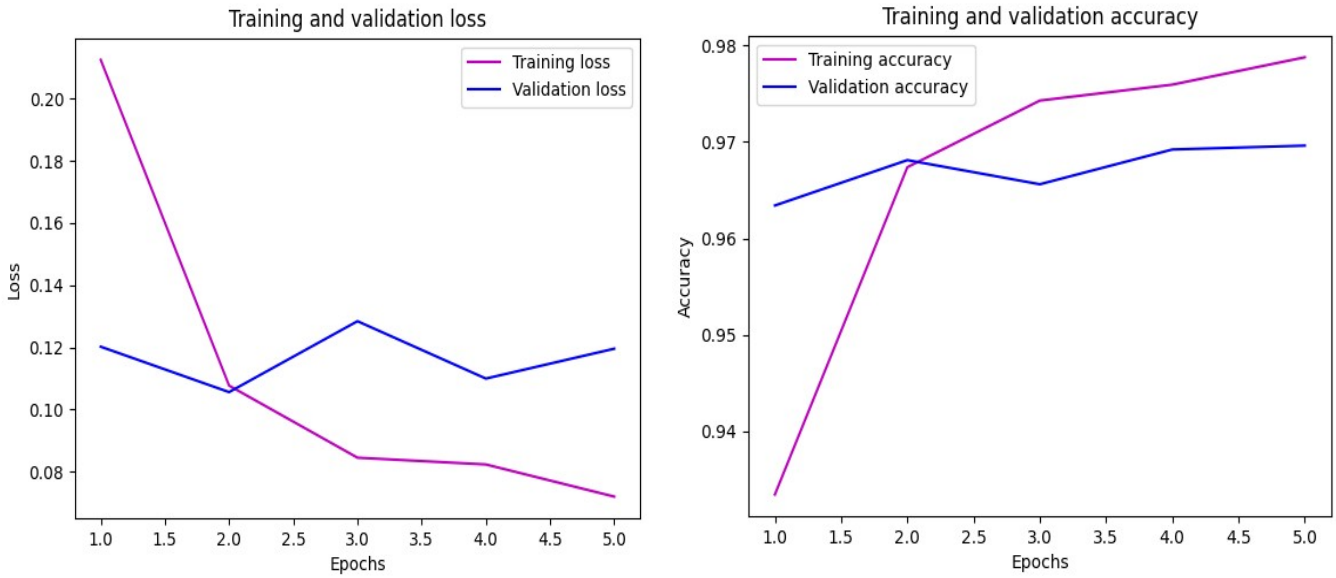



Рисунок 1 — Графики потерь и точности для оптимизатора Adam

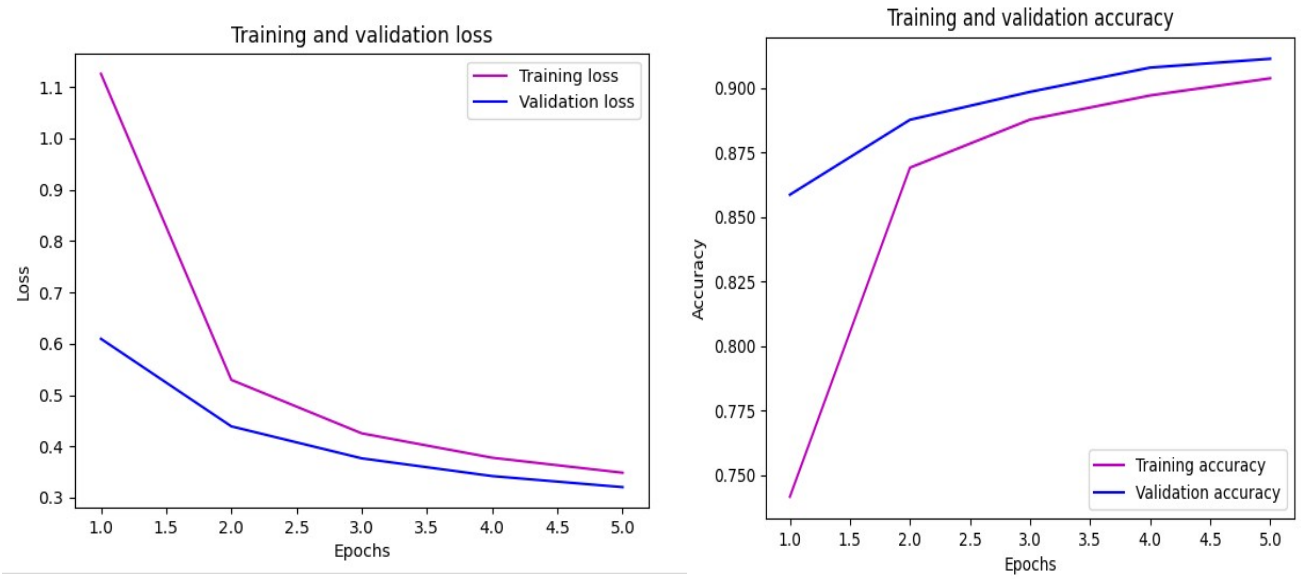
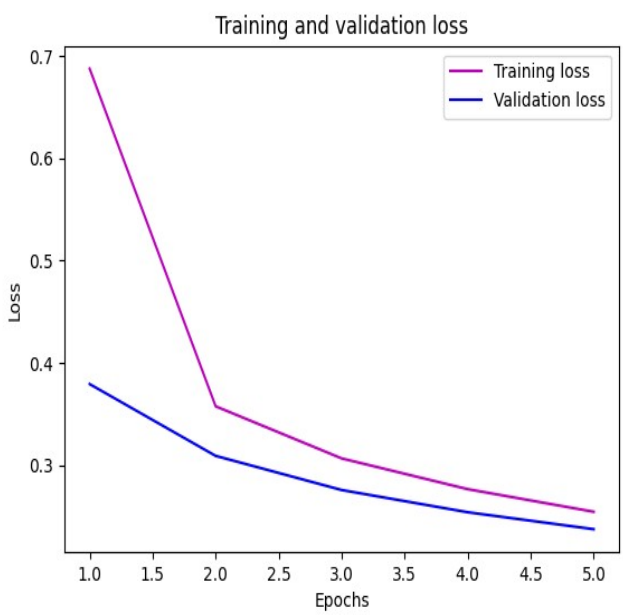



Рисунок 2 — Графики потерь и точности для оптимизатора SGD



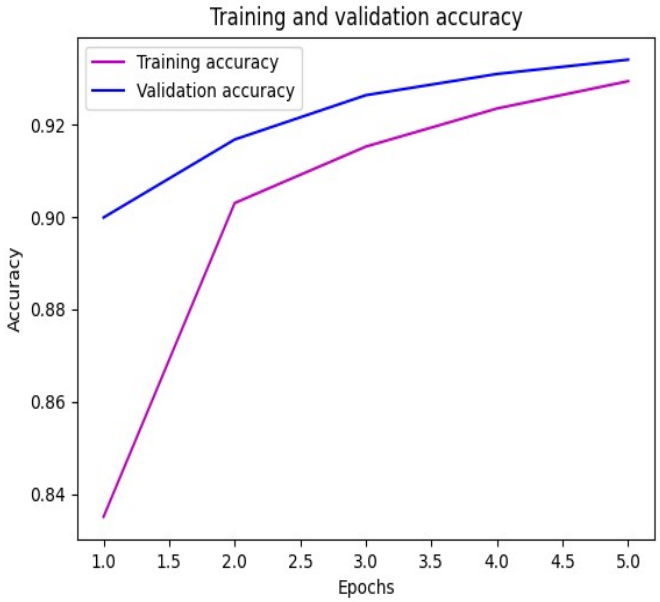
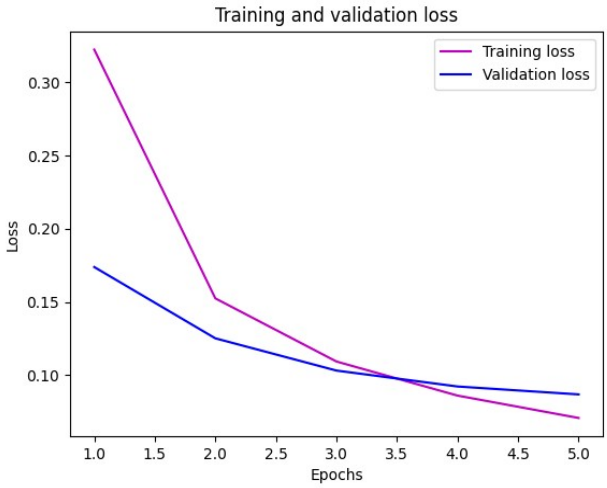




Рисунок 3 — Графики потерь и точности для оптимизатора Adagrad

Заметим, что лучшие результаты из представленных на графике показал оптимизатор Adam, также попробуем найти оптимальные параметры для него.



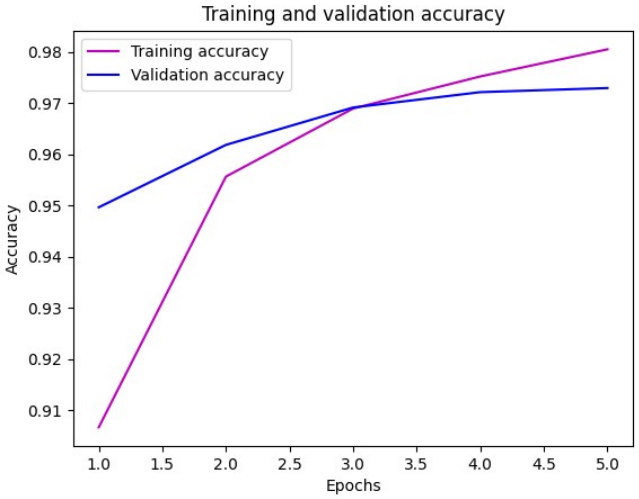
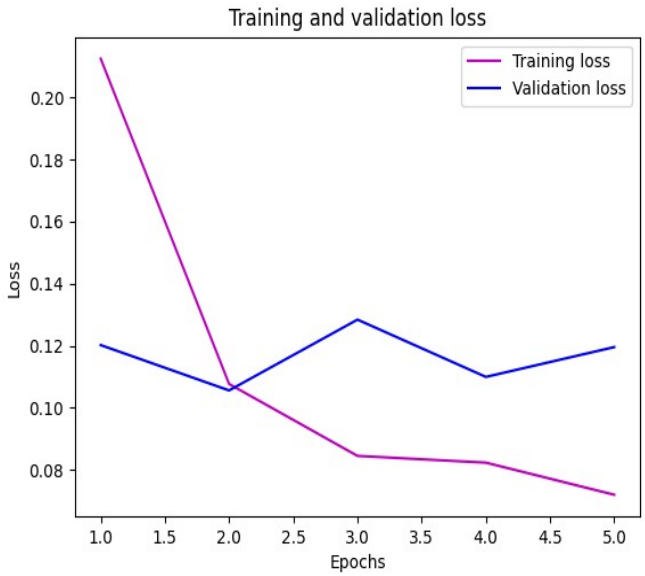




Рисунок 4 — Оптимизатор — Adam, learning_rate = 0.1



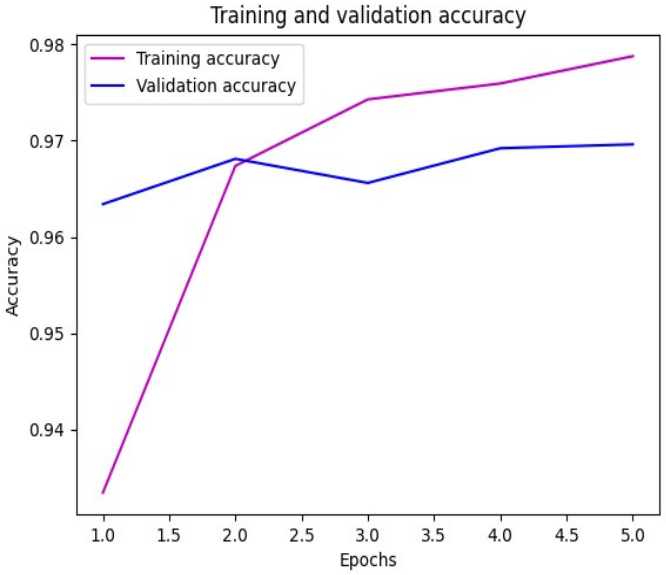
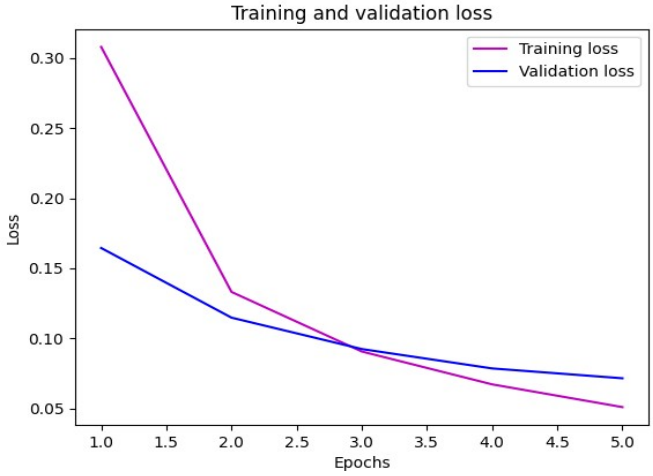




Рисунок 5 — Оптимизатор — Adam, learning_rate = 0.01



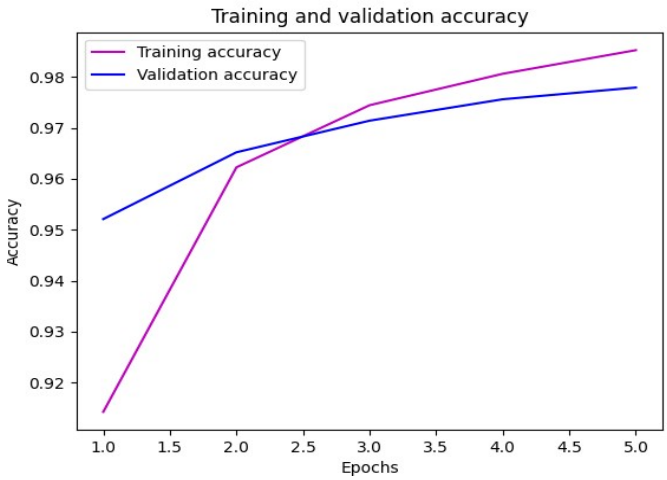




Рисунок 5 — Оптимизатор — Adam, learning_rate = 0.001

При дальнейшем уменьшении learning_rate результаты существенно не изменяются, поэтому принимаем данное значение как наилучшее для модели.

**Выводы.**

В ходе выполнения лабораторной работы была реализована классификация черно-белых изображений рукописных цифр по 10 категориям. Ознакомился с представлением графических данных, с простейшим способом передачи графических данных нейронной сети, была написана функция, которая позволяет загружать изображения пользователя и классифицировать его. Была корректна построена модель и настроены параметры её обучения.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД

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

def load_img(path):
image = Image.open(path)
image = image.resize((28, 28))
image = np.dot(np.asarray(image), np.array([1 / 3, 1 / 3, 1 / 3]))
image /= 255
image = 1 - image
return image.reshape((1, 28 * 28))

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=Adam(learning_rate=0.001),loss='categorical_cros
sentropy', metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=5,
```

```
batch_size=128, validation_data=(test_images, test_labels))
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plot.plot(epochs, loss, 'm', 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 = history.history['accuracy']
val_acc = history.history['val_accuracy']
plot.plot(epochs, acc, 'm', 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_loss:', test_loss)
print('test_acc:', test_acc)
```