МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искусственные нейронные сети»

Тема: Многоклассовая классификация цветов

Студент гр. 7383 _____ Рудоман В. А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы:

Реализовать классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков.

Задачи.

1. Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)
2. Изучить обучение при различных параметрах обучения (параметры функции fit)
3. Построить графики ошибок и точности в ходе обучения
4. Выбрать наилучшую модель

Ход работы.

1. Была создана и обучена модель искусственной нейронной сети в соответствии с условиями (весь код представлен в приложении А).
2. Для тестирования поведения сети в зависимости от числа нейронов была написана функция `test_num_of_neurons`.

Протестировано поведение при варьирующемся числе нейронов при двух скрытых слоях. Графики ошибок и точности показаны на рис. 1, 2.

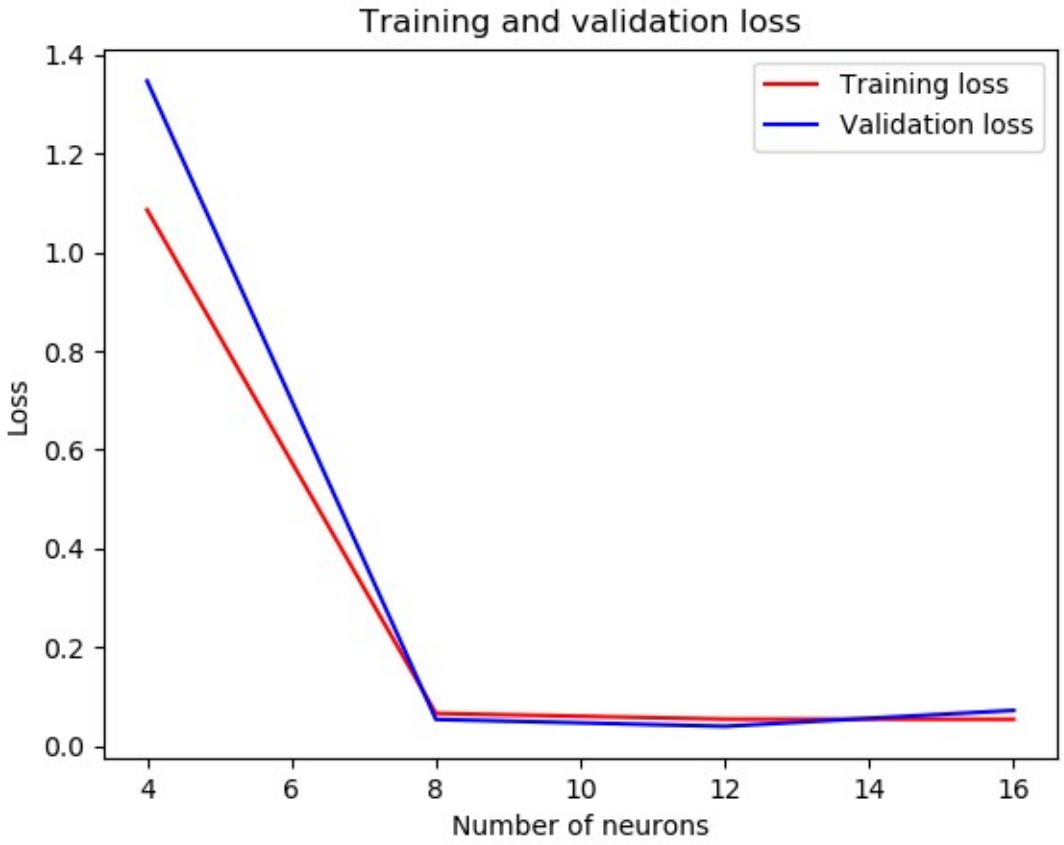
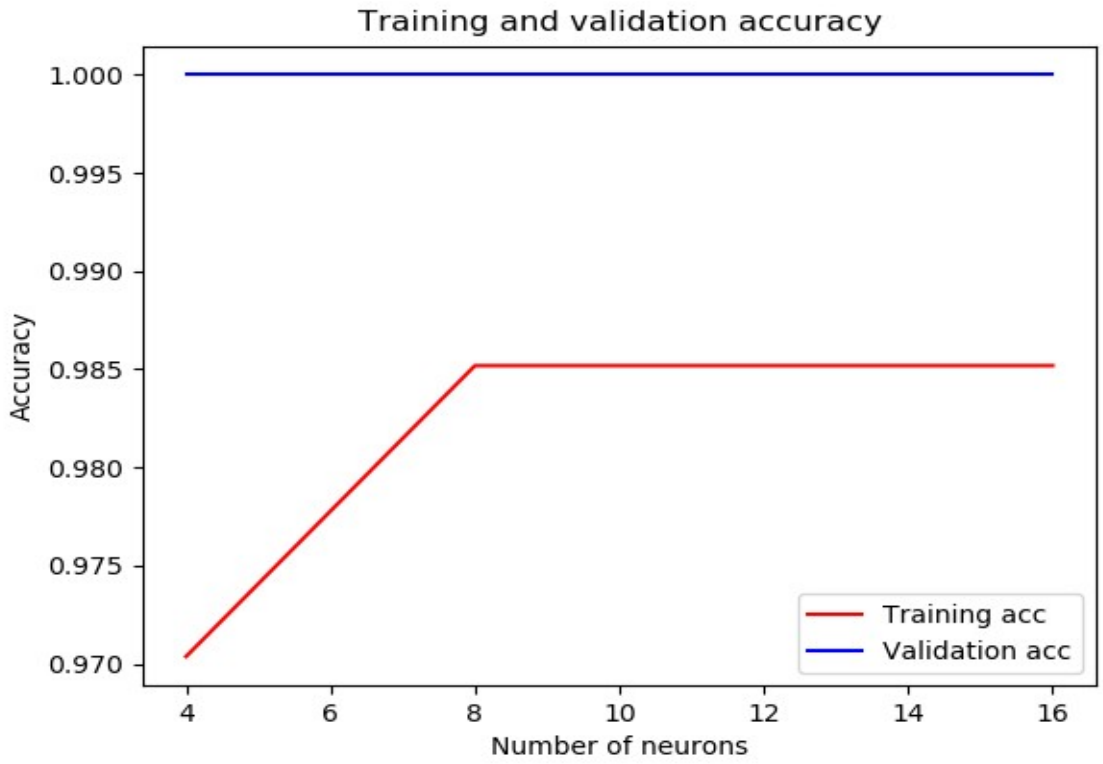


Рисунок 1 — Ошибки в зависимости от числа нейронов



Ри

Рисунок 2 — Зависимость точности от числа нейронов

Как видно, с ростом числа нейронов ошибки уменьшаются, а точность увеличивается

Было зафиксировано число нейронов протестировано с помощью функции `test_num_of_layers` поведение при изменяющемся числе слоев. Графики ошибок и точности показаны на рис. 3, 4.

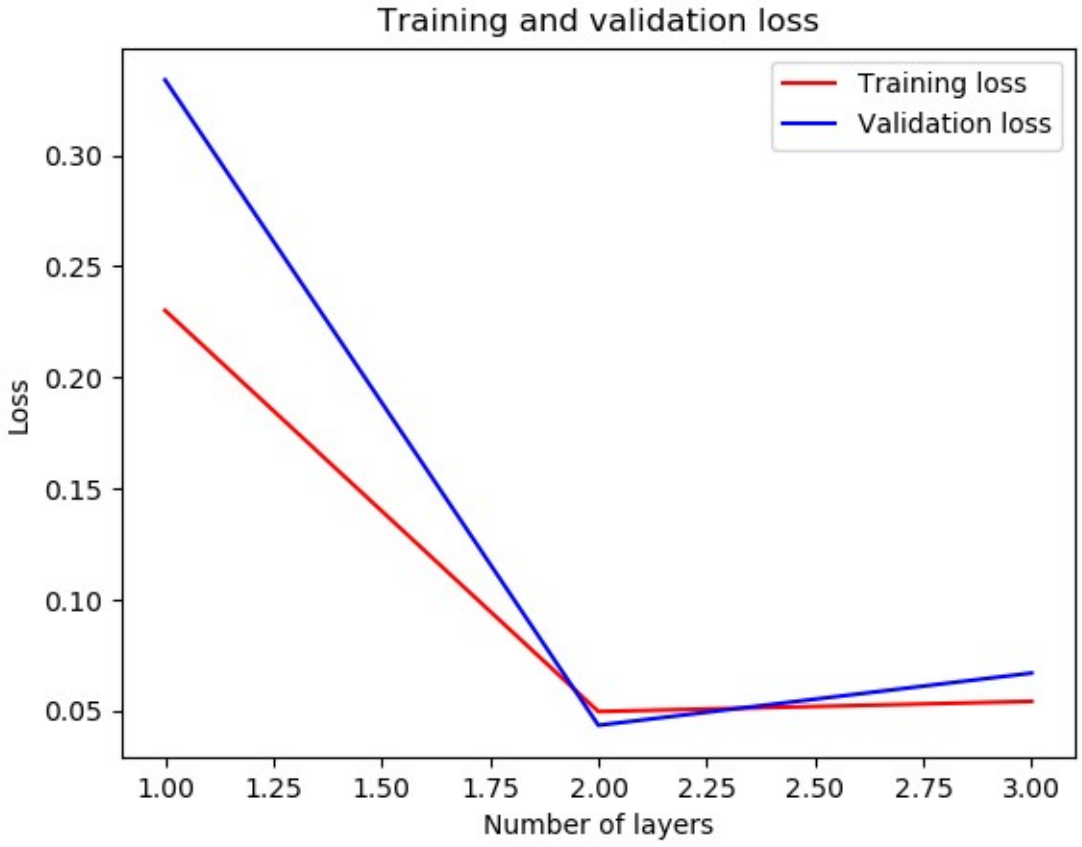


Рисунок 3 - Ошибки в зависимости от числа слоев

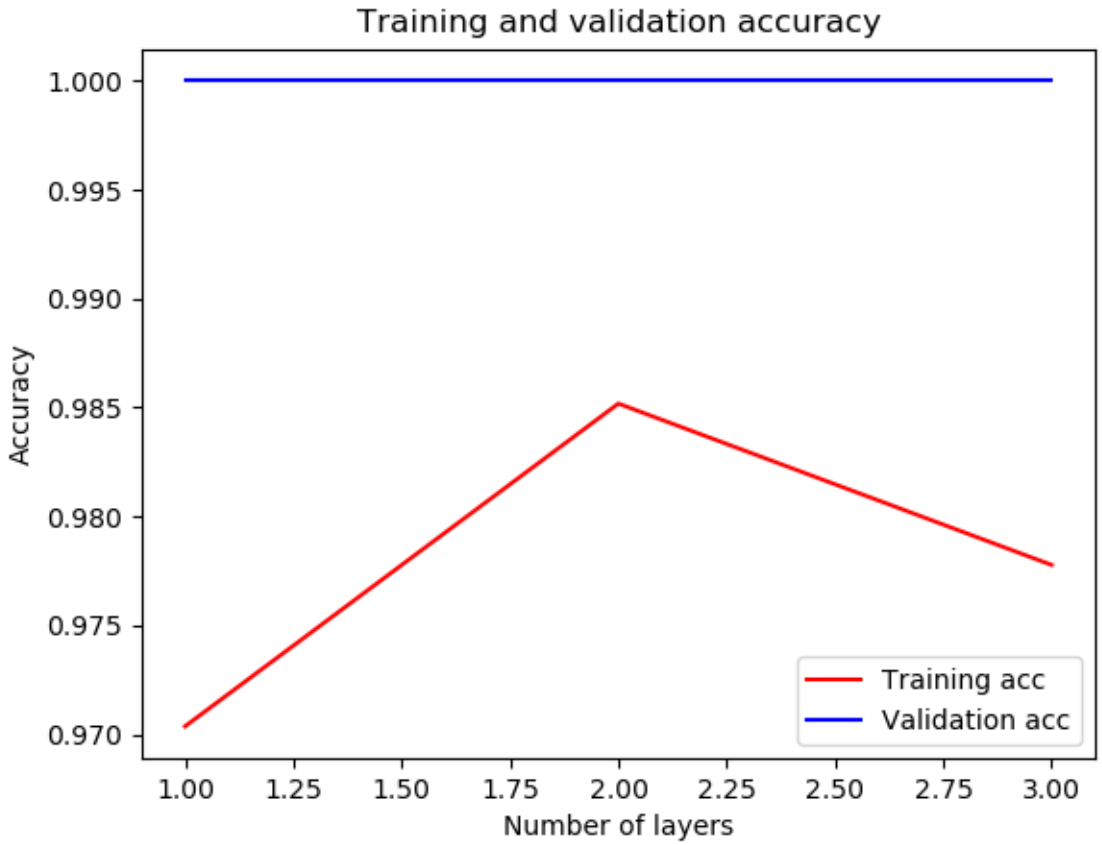


Рисунок 4 — Зависимость точности от числа слоев

Из графиков видно, что лучшие результаты сеть показывает при среднем числе слоев, равном 2.

3. Протестировано обучение функции в зависимости от параметров функции `fit`.

Используем функцию `test_epochs` для тестирования влияния параметра `epochs`. Графики ошибок и точности показаны на рис. 5, 6.

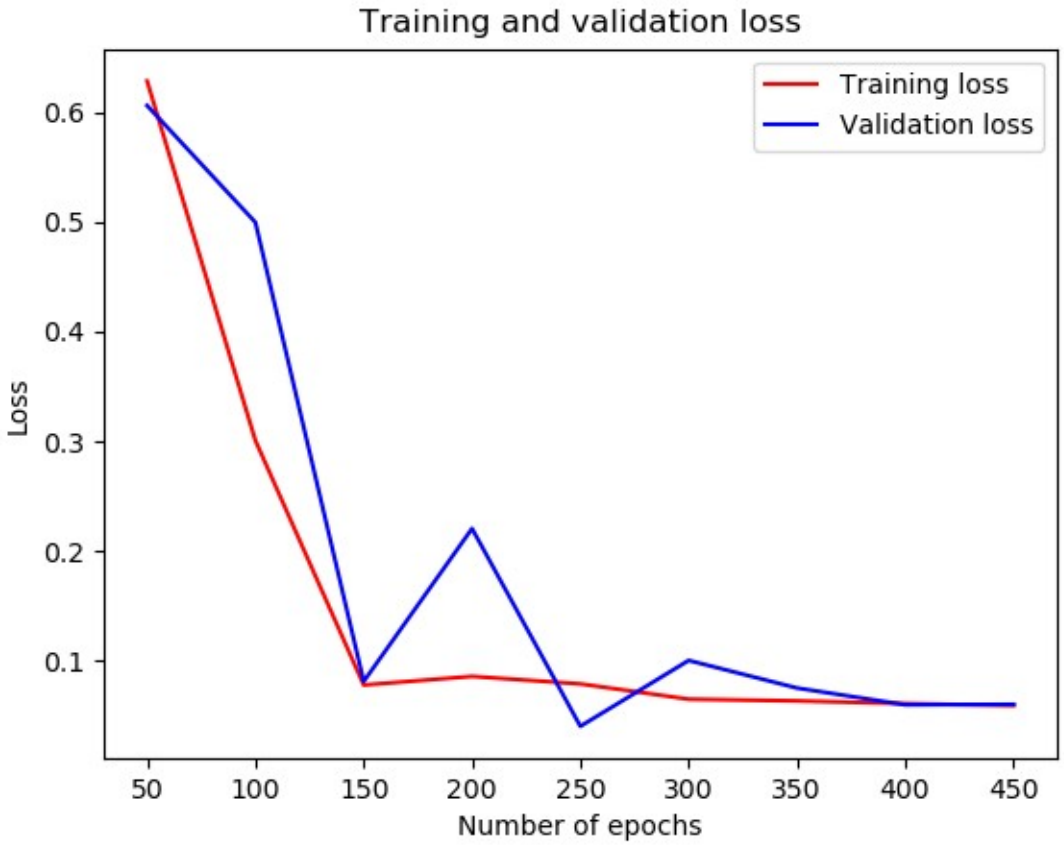
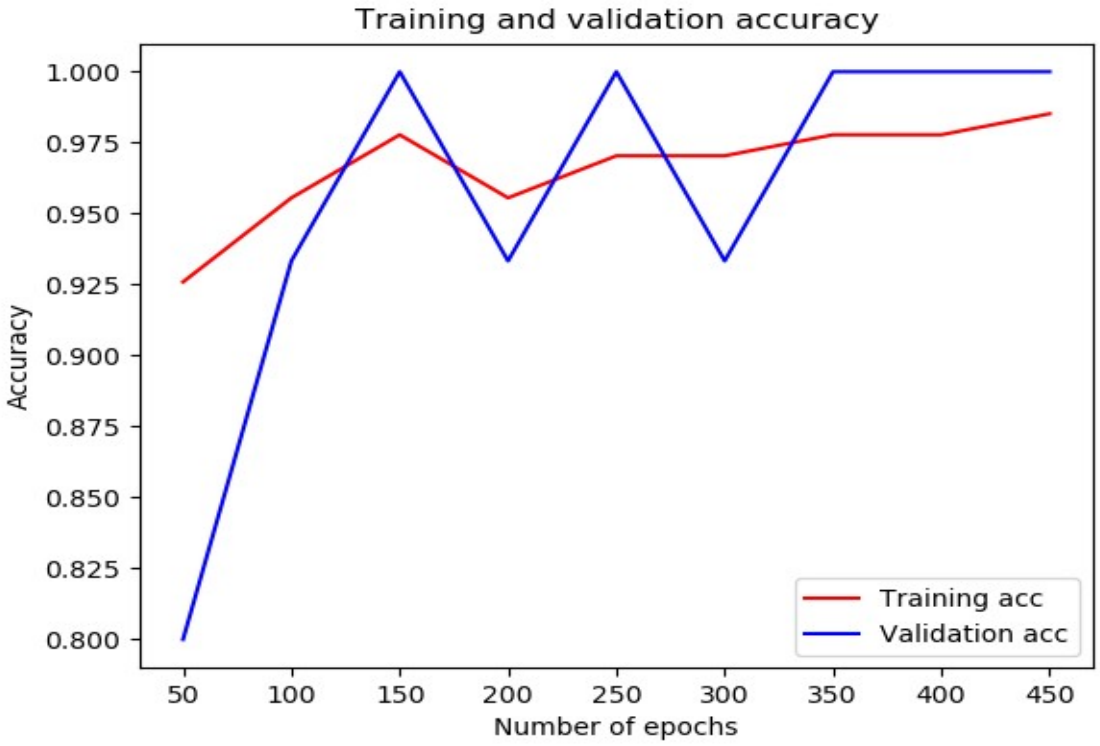


Рисунок 5 - Ошибки в зависимости от параметра epochs



Ри

Рисунок 6 — Зависимость точности от параметра epochs

Как видно из рисунков, результаты улучшаются при росте числа эпох.

Используем функцию `test_batch_size` для тестирования влияния параметра `batch_size`. Графики ошибок и точности показаны на рис. 7, 8.

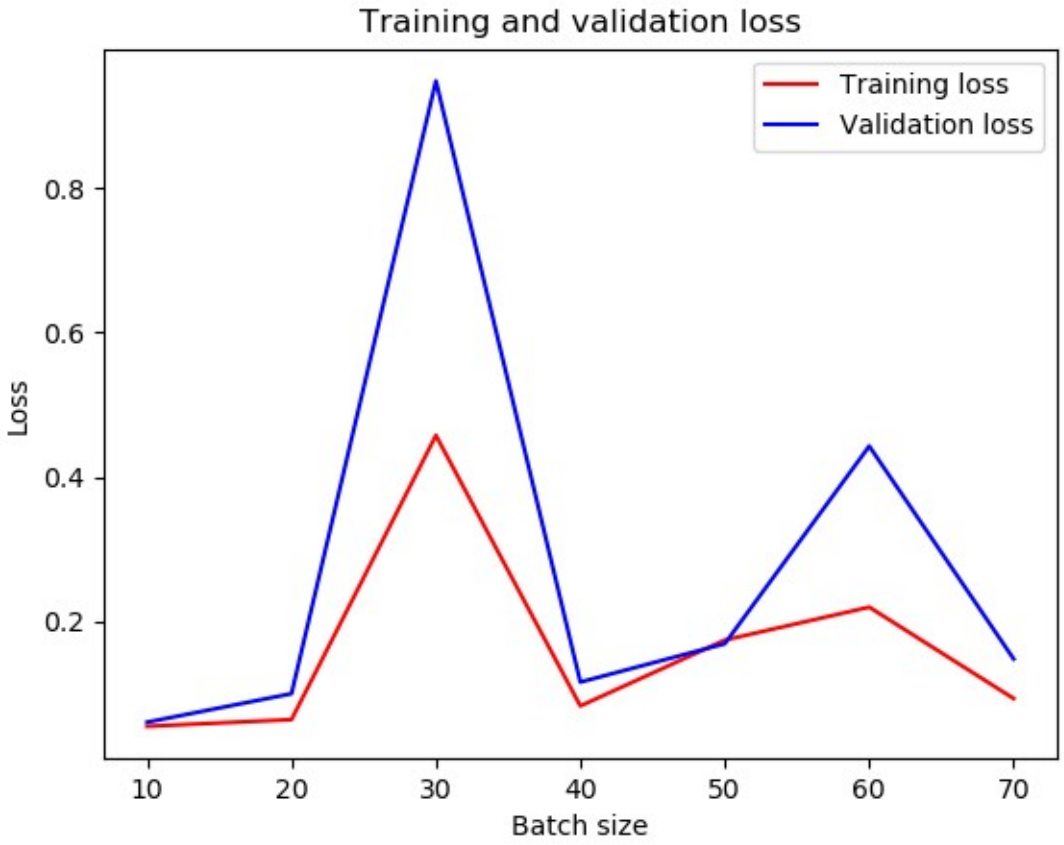


Рисунок 7 - Ошибки в зависимости от параметра batch_size

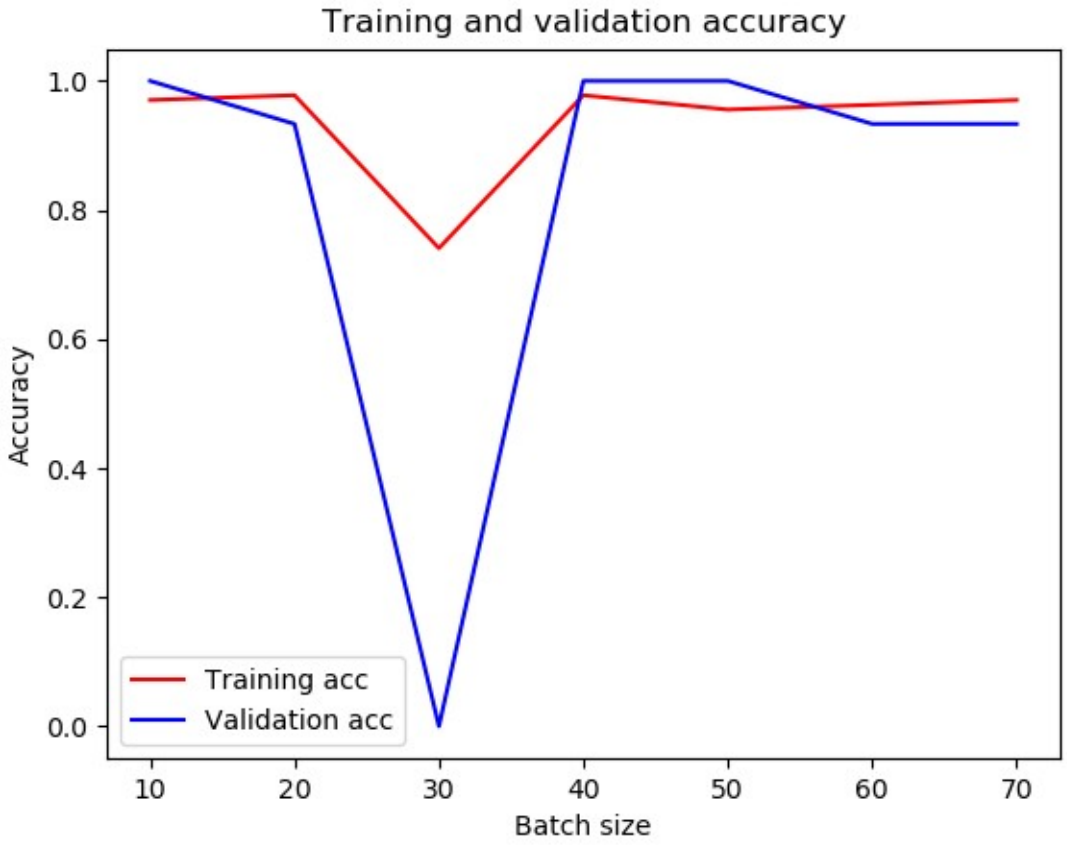


Рисунок 8 — Зависимость точности от параметра batch_size

Как видно, наибольшая точность и наименьшая ошибка при параметре, равном 10.

Используем функцию `validation_test` для тестирования влияния параметра `validation_split`. Графики ошибок и точности показаны на рис. 9, 10.

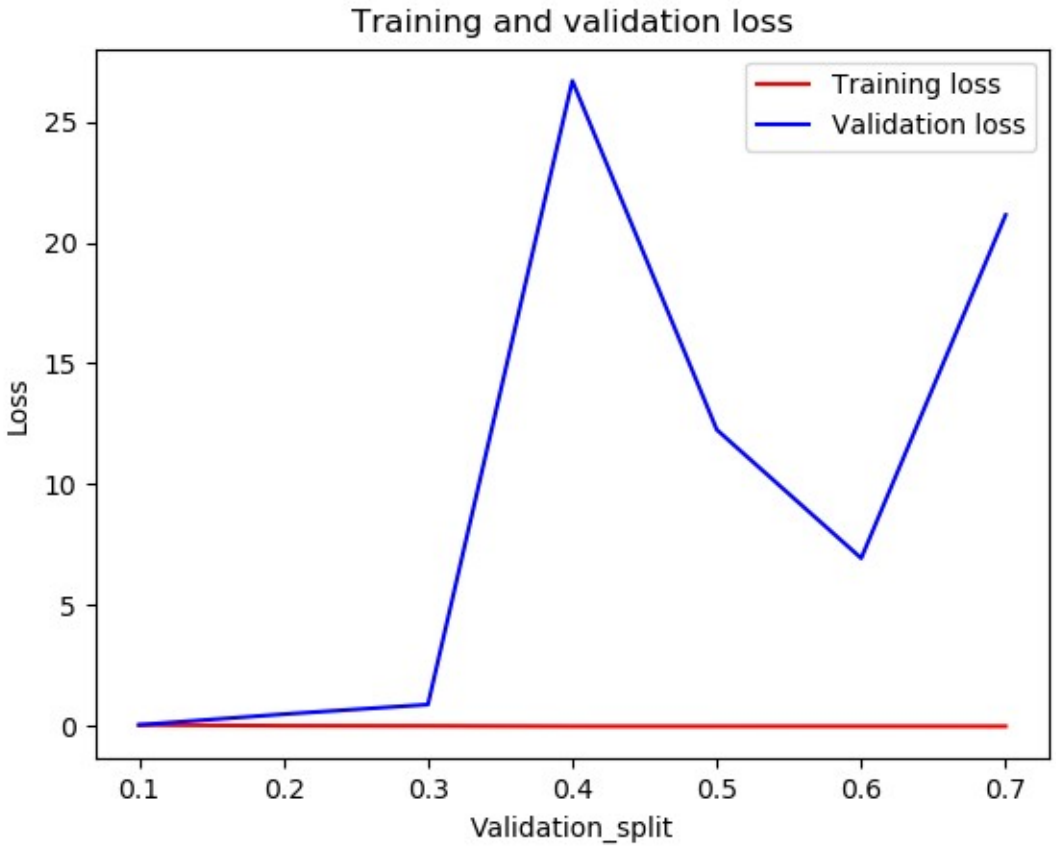
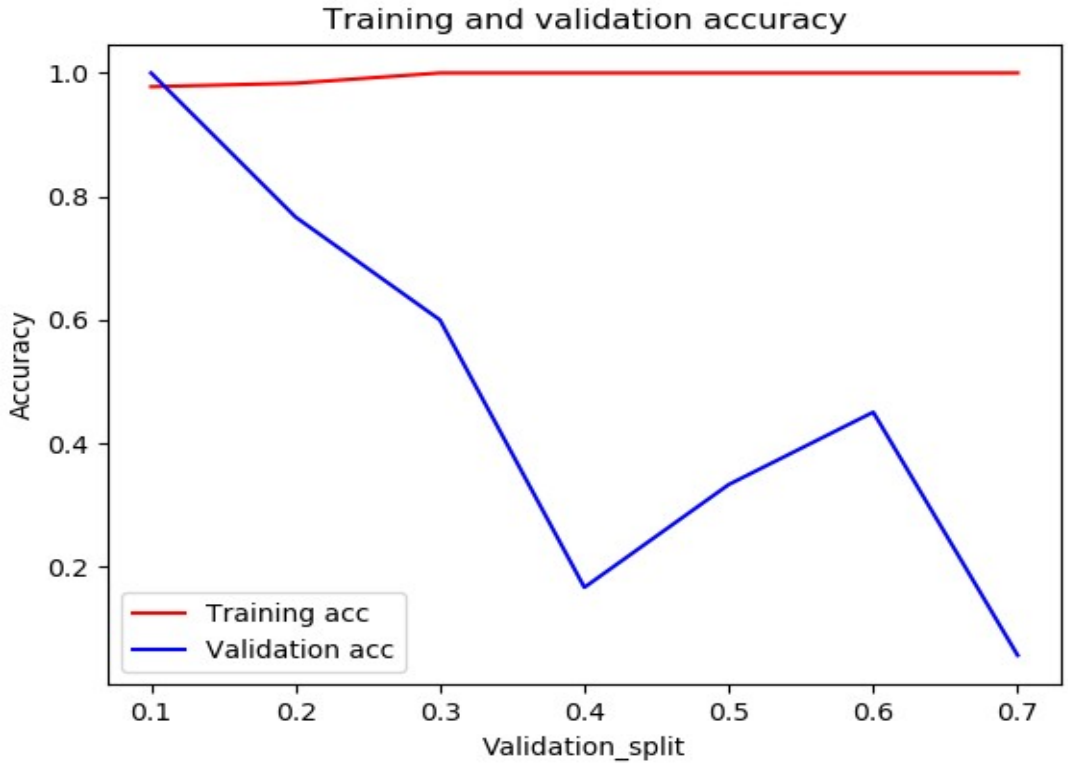


Рисунок 9 - Ошибки в зависимости от параметра validation_split



Ри

Рисунок 10 — Зависимость точности от параметра validation_split

Как видно, при увеличении параметра растет ошибка и уменьшается точность.

4. Из проделанного исследования выберем лучшую модель. Выберем модель из двух слоев с 8 нейронами на каждом с параметрами epochs=450, batch_size=10, validation_split=0.1. Графики ошибок и точности показаны на рис. 13, 14.

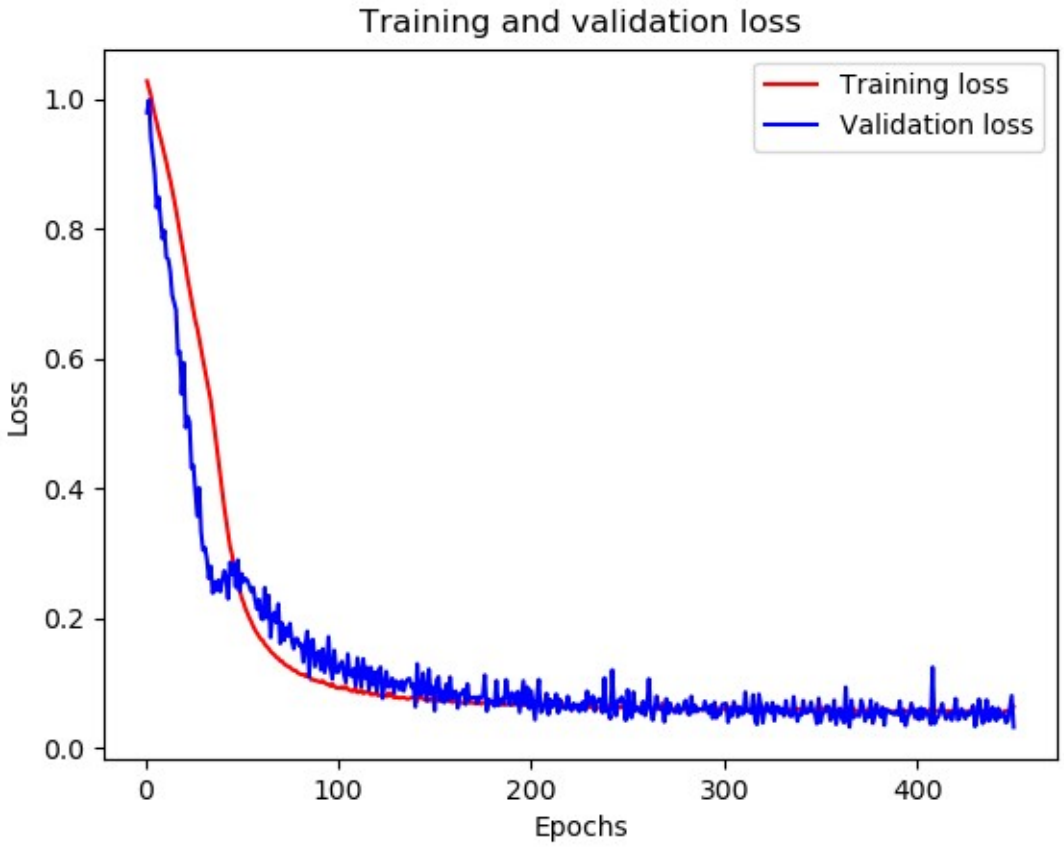


Рисунок 13 - Ошибки в ходе обучения выбранной лучшей модели

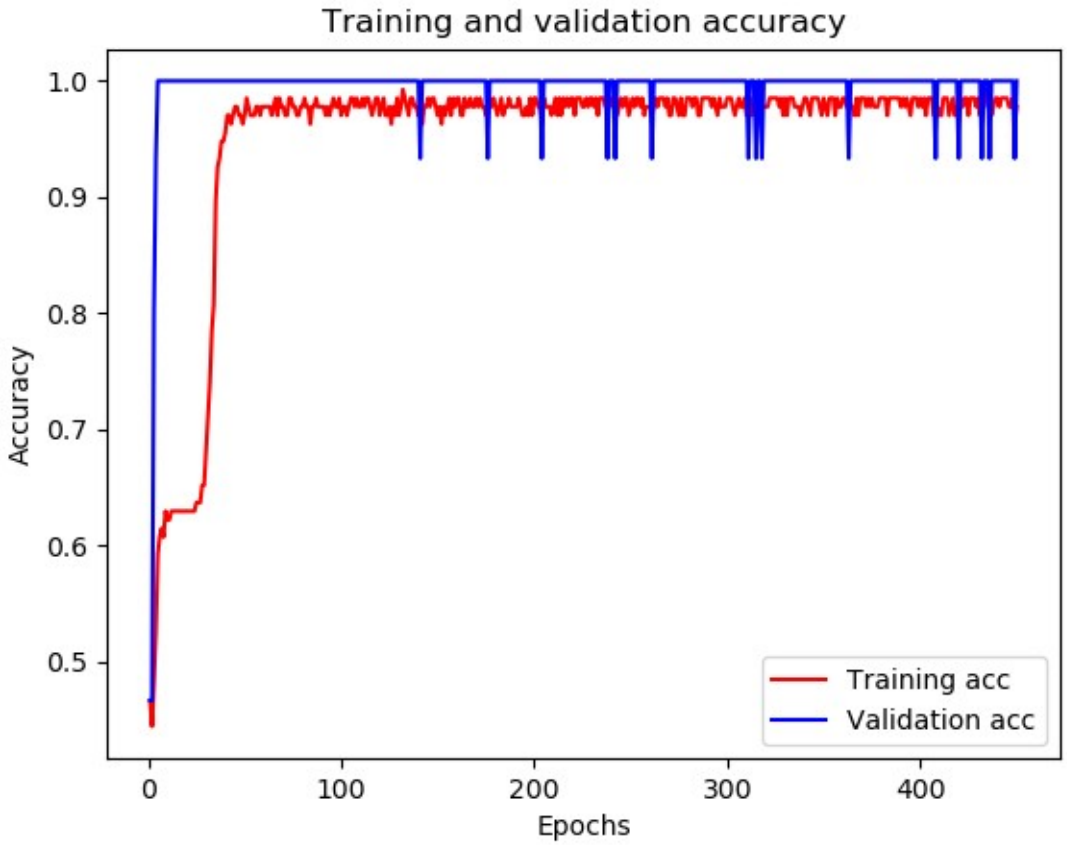


Рисунок 14 - Точность в ходе обучения выбранной лучшей модели

Выводы:

Были изучены основы работы с искусственными нейронными сетями на языке python. Было исследовано поведение сети в зависимости от ее модели и параметров обучения. Была выбрана наилучшая модель.

ПРИЛОЖЕНИЕ А

```
import pandas
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
import numpy as np

dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:4].astype(float)
Y = dataset[:, 4]
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
# Тестирование при различных параметрах


def test_num_of_neurons():
    loss = []
    val_loss = []
    acc = []
    val_acc = []
    vect_num_neurs = (4, 8, 12, 16)  # 2 слоя
    for i in vect_num_neurs:
        model = Sequential()
        model.add(Dense(i, activation='relu'))
        model.add(Dense(i, activation='relu'))
        model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
            H = model.fit(X, dummy_y, epochs=450, batch_size=10,
validation_split=0.1)
        loss.append(H.history['loss'][-1])
        val_loss.append(H.history['val_loss'][-1])
        acc.append(H.history['accuracy'][-1])
        val_acc.append(H.history['val_accuracy'][-1])
    draw_test(vect_num_neurs, 'Number of neurons', loss, val_loss,
acc, val_acc)


def test_num_of_layers():
    loss = []
    val_loss = []
    acc = []
    val_acc = []
    vect_num_layers = (1, 2, 3)
    for i in vect_num_layers:
        model = Sequential()
```

```
        for j in range(1, i):
            model.add(Dense(8, activation='relu'))
            model.add(Dense(8, activation='relu'))
        model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
             H = model.fit(X, dummy_y, epochs=450, batch_size=10,
validation_split=0.1)
        loss.append(H.history['loss'][-1])
        val_loss.append(H.history['val_loss'][-1])
        acc.append(H.history['accuracy'][-1])
        val_acc.append(H.history['val_accuracy'][-1])
    draw_test(vect_num_layers, 'Number of layers', loss, val_loss,
acc, val_acc)


def test_epochs():
    loss = []
    val_loss = []
    acc = []
    val_acc = []
    vect_epochs = range(50, 451, 50)
    for i in vect_epochs:
        model = Sequential()
        model.add(Dense(8, activation='relu'))
        model.add(Dense(8, activation='relu'))
        model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
              H = model.fit(X, dummy_y, epochs=i, batch_size=10,
validation_split=0.1)
        loss.append(H.history['loss'][-1])
        val_loss.append(H.history['val_loss'][-1])
        acc.append(H.history['accuracy'][-1])
        val_acc.append(H.history['val_accuracy'][-1])
      draw_test(vect_epochs, 'Number of epochs', loss, val_loss,
acc, val_acc)


def test_batch_size():
    loss = []
    val_loss = []
    acc = []
    val_acc = []
    vect_batch = range(10, 80, 10)
    for i in vect_batch:
        model = Sequential()
        model.add(Dense(8, activation='relu'))
        model.add(Dense(8, activation='relu'))
        model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
             H = model.fit(X, dummy_y, epochs=450, batch_size=i,
```

```
validation_split=0.1)
        loss.append(H.history['loss'][-1])
        val_loss.append(H.history['val_loss'][-1])
        acc.append(H.history['accuracy'][-1])
        val_acc.append(H.history['val_accuracy'][-1])
       draw_test(vect_batch, 'Batch size', loss, val_loss, acc,
val_acc)


def validation_test():
    loss = []
    val_loss = []
    acc = []
    val_acc = []
    vect_validation = []
    for i in range(1, 8):
        vect_validation.append(i*0.1)
        model = Sequential()
        model.add(Dense(8, activation='relu'))
        model.add(Dense(8, activation='relu'))
        model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
            H = model.fit(X, dummy_y, epochs=450, batch_size=10,
validation_split=i*0.1)
        loss.append(H.history['loss'][-1])
        val_loss.append(H.history['val_loss'][-1])
        acc.append(H.history['accuracy'][-1])
        val_acc.append(H.history['val_accuracy'][-1])
     draw_test(vect_validation, 'Validation_split', loss, val_loss,
acc, val_acc)


def draw_test(arg, label, loss, val_loss, acc, val_acc):
    plt.plot(arg, loss, 'r', label='Training loss')
    plt.plot(arg, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel(label)
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(arg, acc, 'r', label='Training acc')
    plt.plot(arg, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel(label)
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def best_model():
    model = Sequential()
```

```
    model.add(Dense(8, activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(3, activation='softmax'))
                                    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
         H  =  model.fit(X,  dummy_y,  epochs=450,  batch_size=10,
validation_split=0.1)
    loss = H.history['loss']
    val_loss = H.history['val_loss']
    acc = H.history['accuracy']
    val_acc = H.history['val_accuracy']
    epochs = range(1, len(loss) + 1)
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(epochs, acc, 'r', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


test_num_of_neurons()
test_num_of_layers()
test_epochs()
test_batch_size()
validation_test()
best_model()    plt.show()
```