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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искусственные нейронные сети»

Тема: «Многоклассовая классификация цветов»

Студентка гр. 8383 Сырцова Е.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2021

Цель

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

Задачи

- Ознакомиться с задачей классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель

Требования

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

Ход работы

Была построена нейронная сеть, код которой представлен в приложении А.

1. Изучить различные архитектуры ИНС

После запуска программы с предложенным кодом были получены графики ошибок и точности, а значение точности работы ИНС равно 95,3%. Графики представлены на рис. 1-2.

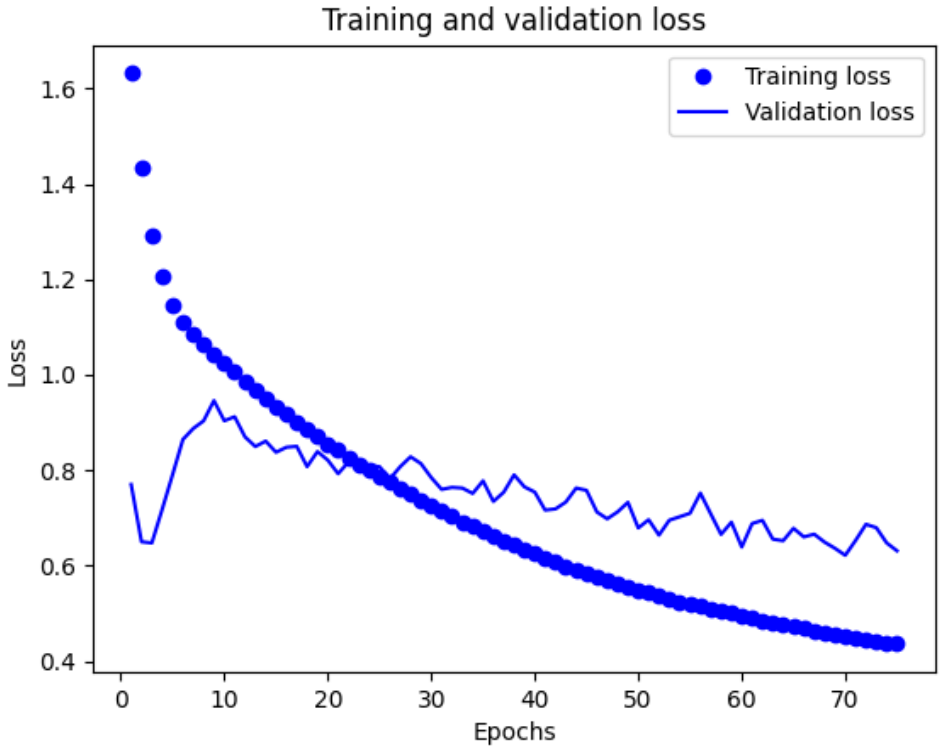

Рисунок 1 – График ошибок исходной сети

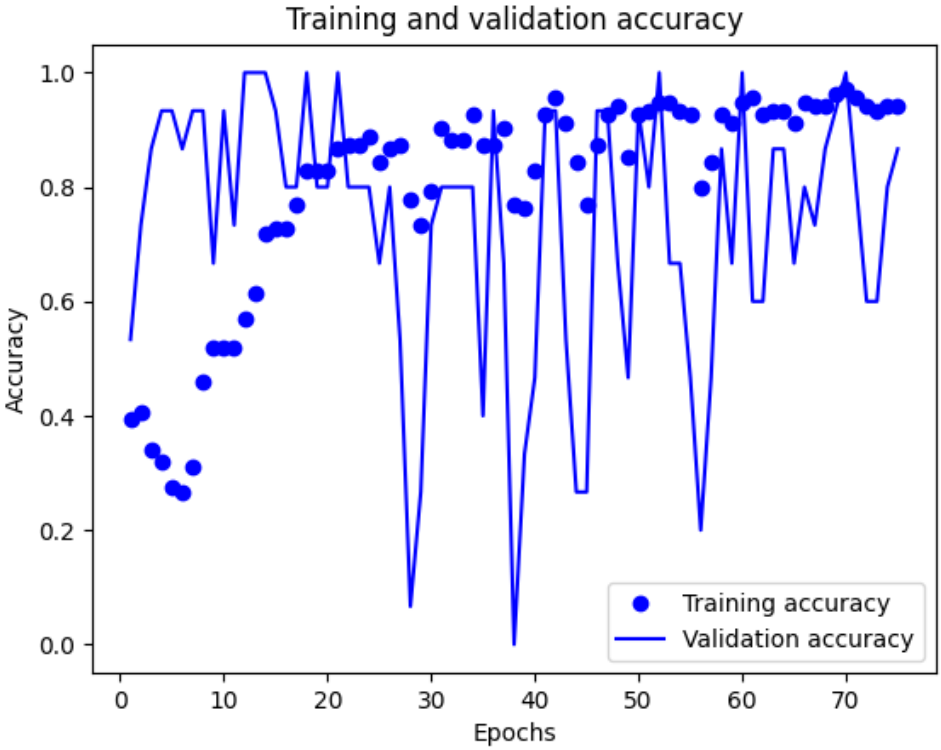

Рисунок 2 – График точности исходной сети

Как видно из графиков точность сети растет, а ошибки уменьшаются, что и ожидалось.

Теперь увеличим число нейронов в первом слое до 8. Точность вычислений сети равна 97,3%, это говорит о том, что число нейронов повысило точность вычислений. Графики представлены на рис. 3-4.

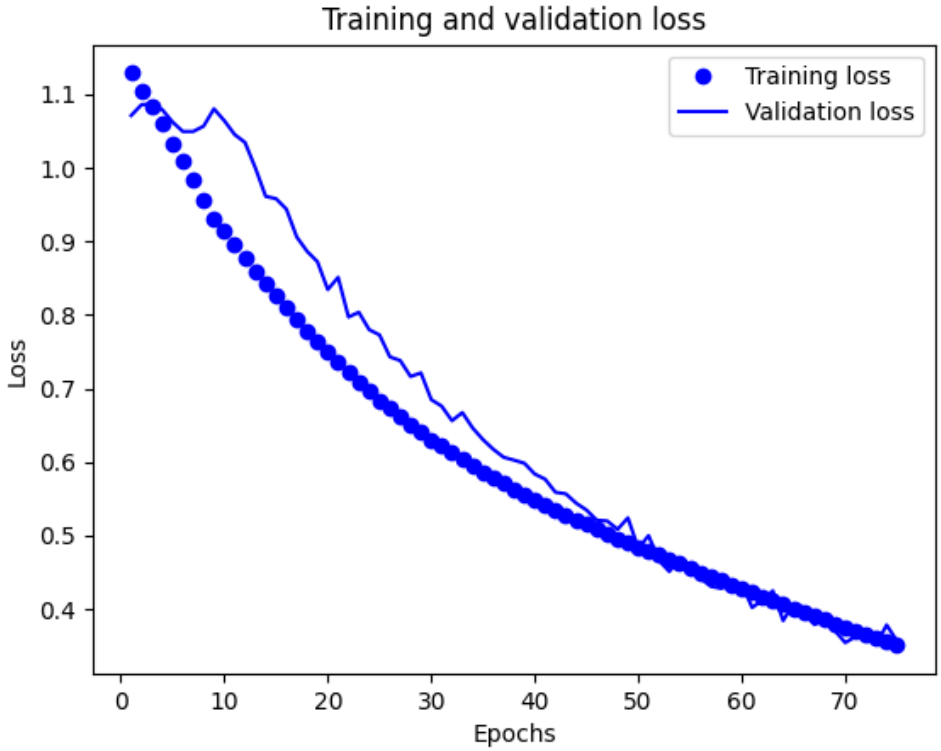

Рисунок 3 – График ошибок

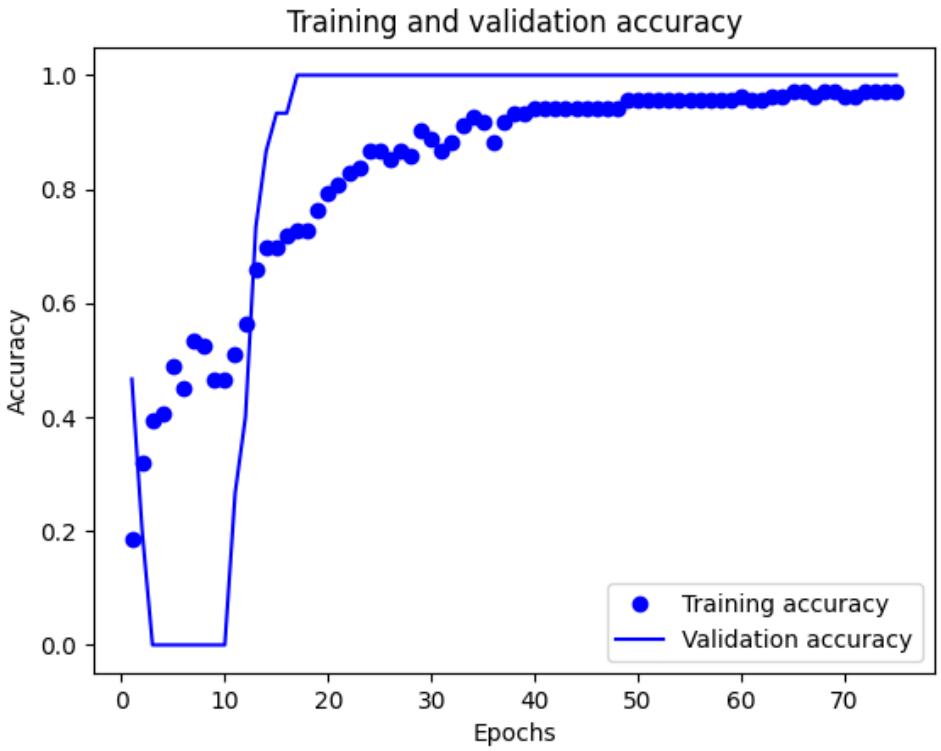

Рисунок 4 – График точности

Оставим прежнее число нейронов и добавим еще один слой между имеющимися. Этот слой совпадает с первым. Точность стала равна 97,3%. График точности стал хуже. Графики изображены на рис. 5-6.

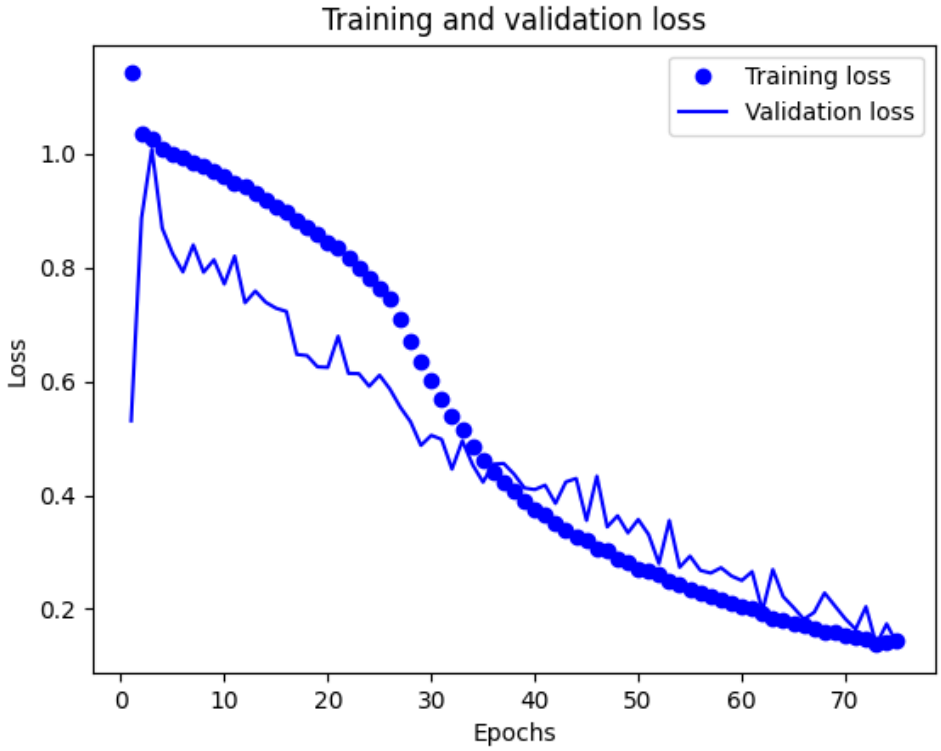

Рисунок 5 – График ошибок

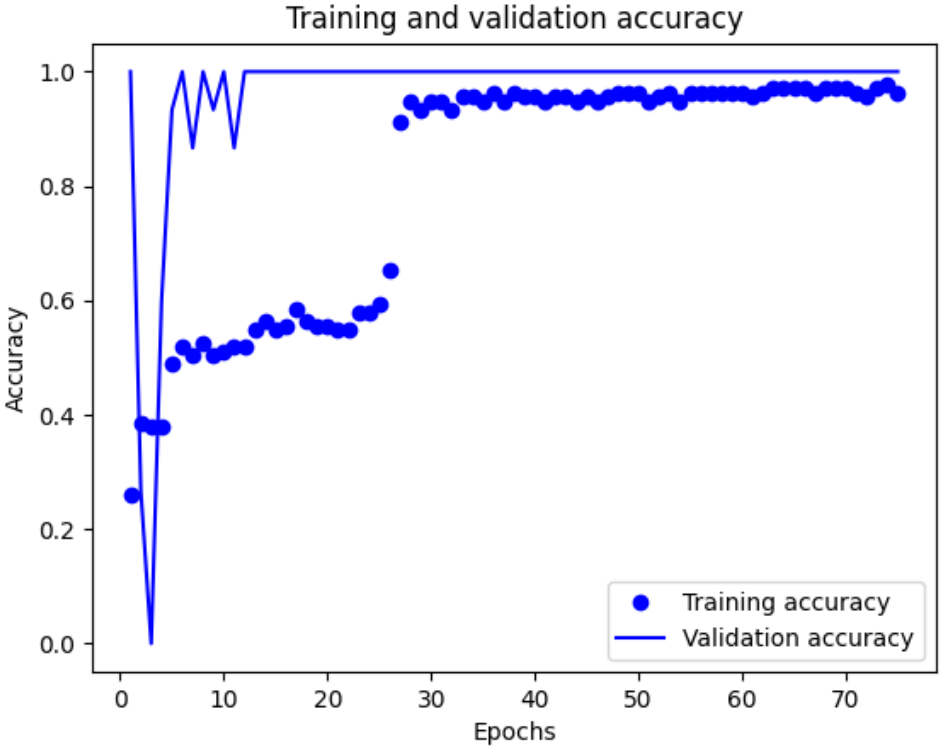

Рисунок 6 – График точности

Добавим 4 нейрона второму слою. Точность теперь равна 98,7%. График точности стал более ровным. Графики на рис. 7-8.

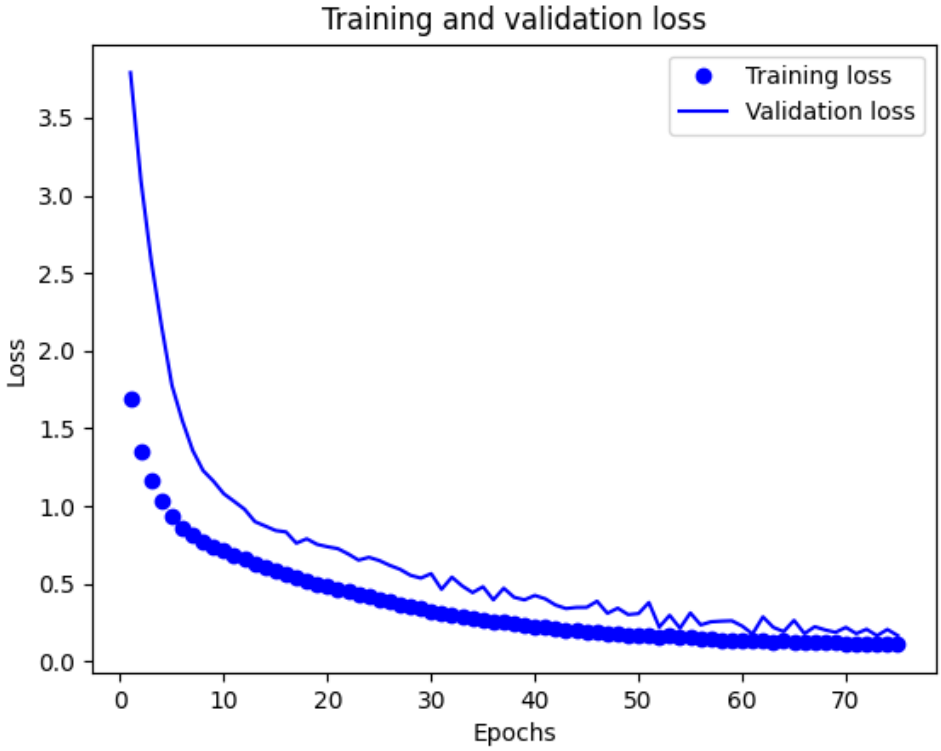

Рисунок 7 – График ошибок

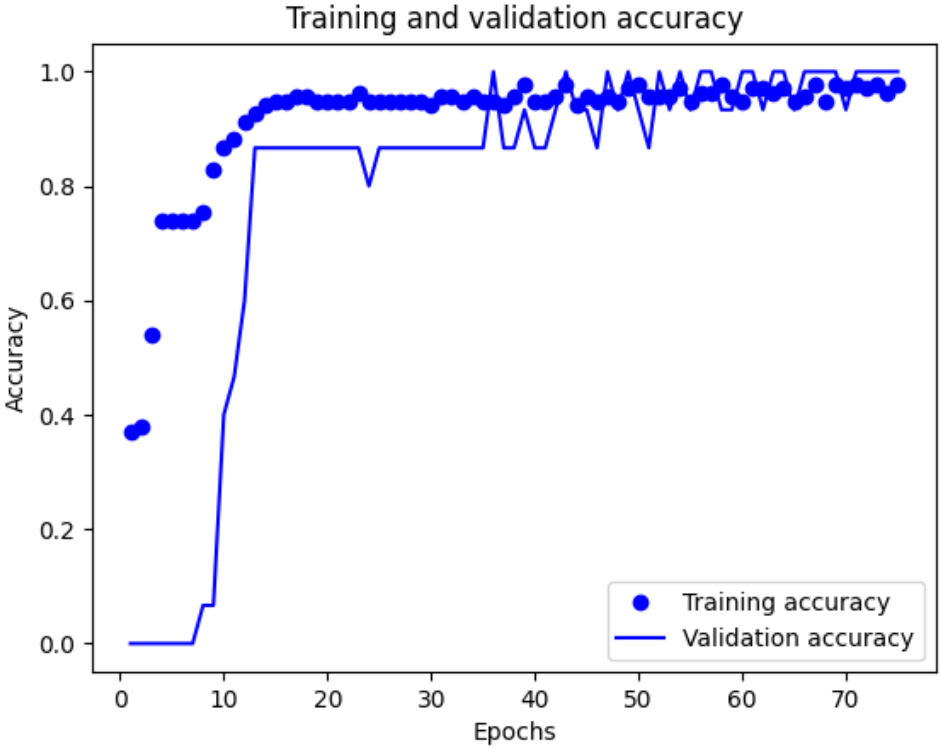

Рисунок 8 – График точности

2. Изучить обучение при различных параметрах обучения.

В модели с 8 нейронами на первом и 16 на втором слое и увеличим количество эпох до 100. Точность вычислений 98,7%. Графики представлены на рис. 9-10.

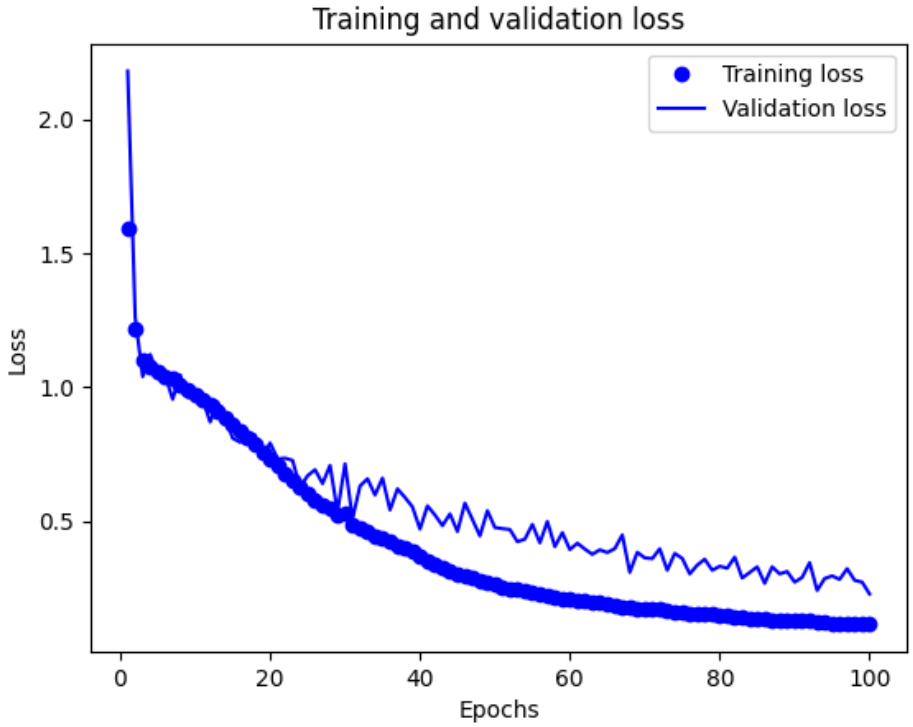

Рисунок 9 – График ошибок

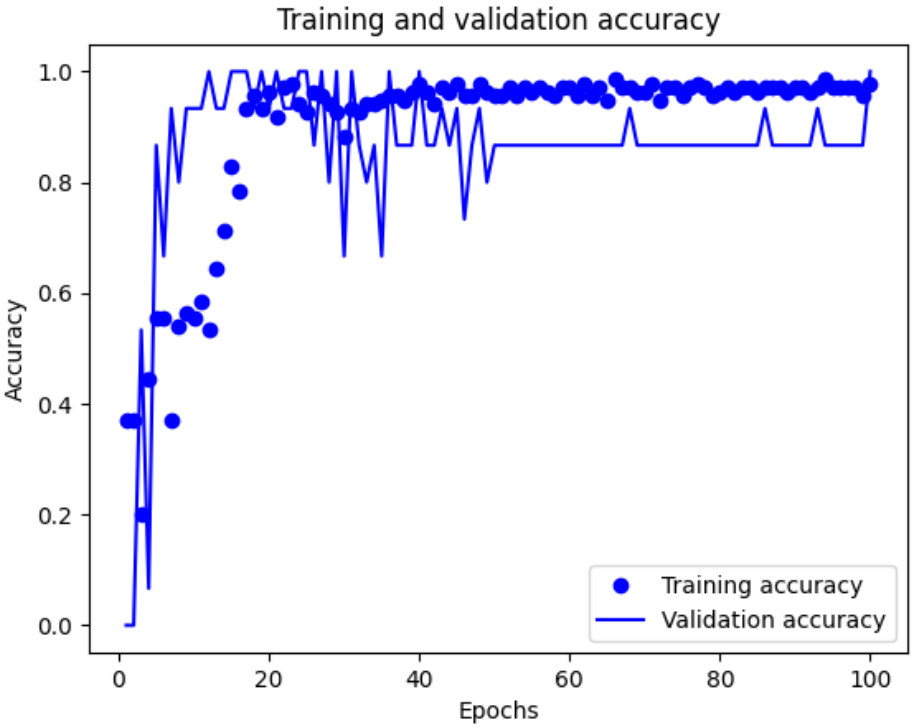

Рисунок 10 – График точности

Увеличим количество данных в одну итерацию до 20. Точность уменьшилась до 97,3%. Графики представлены на рис. 11-12.

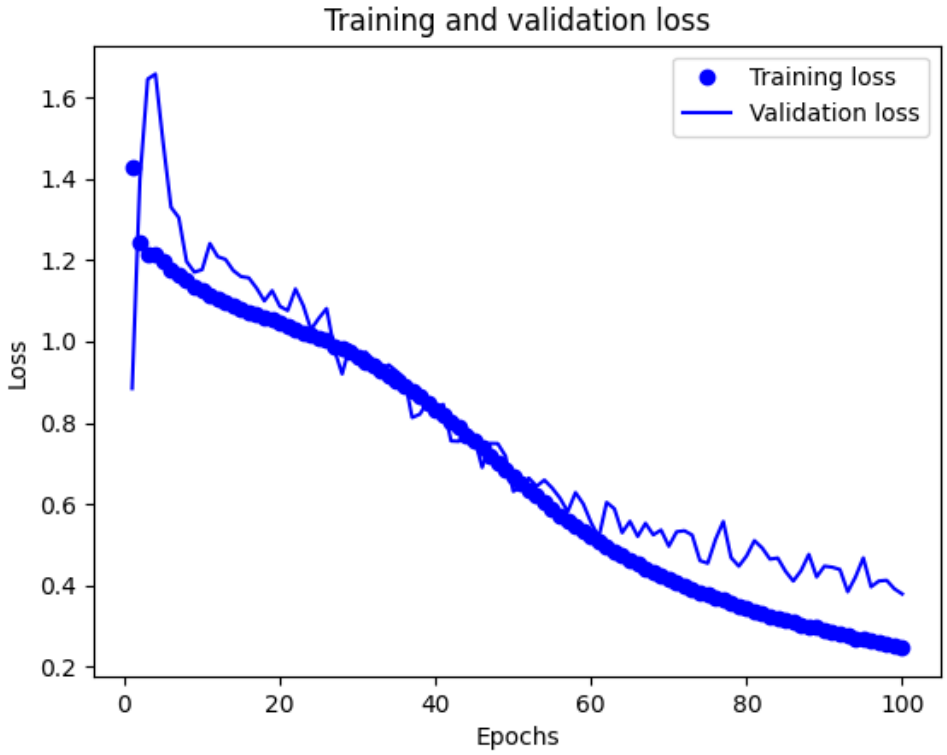

Рисунок 11 – График ошибок

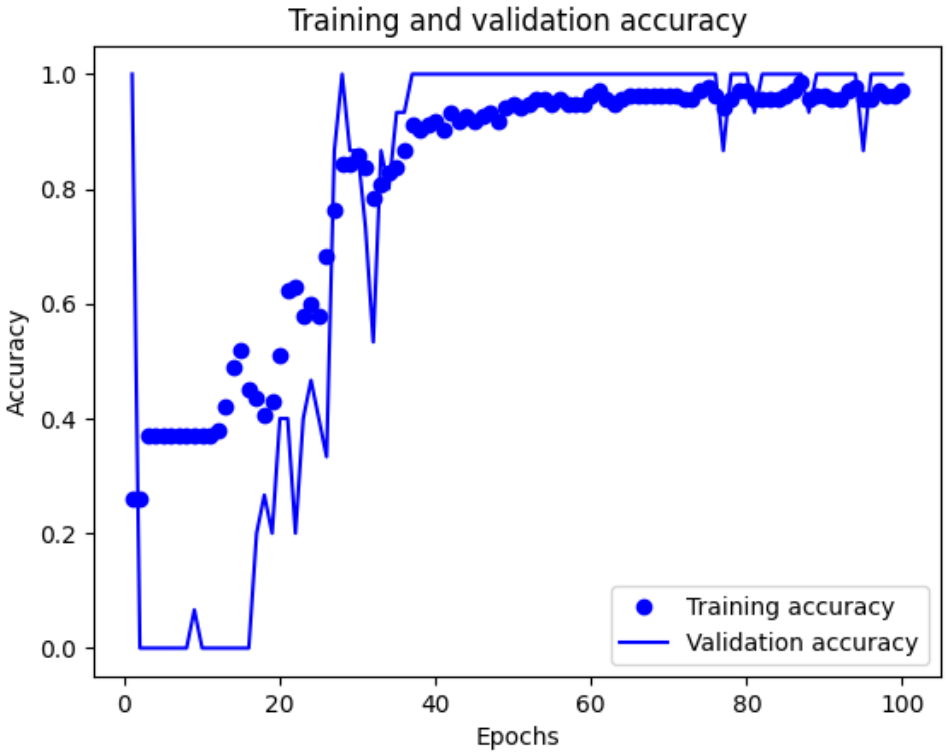

Рисунок 12 – График точности

Вернем количеству итераций значение 10 и увеличим число проверочных до 0,3. Точность падает до 83,3%. Графики изображены на рис. 13-14.

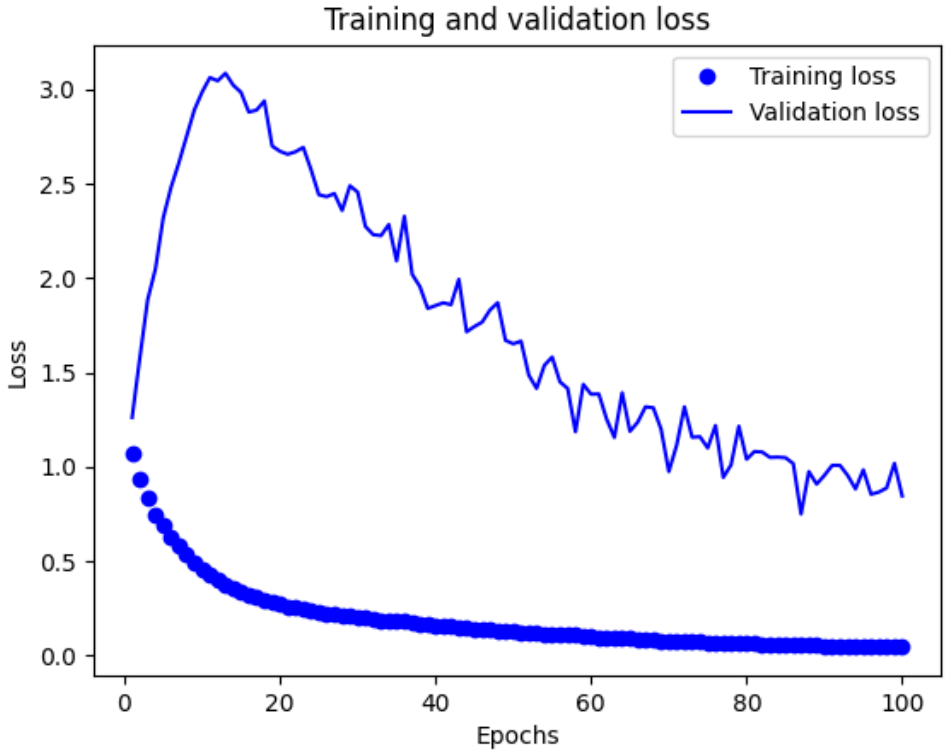

Рисунок 13 – График ошибок

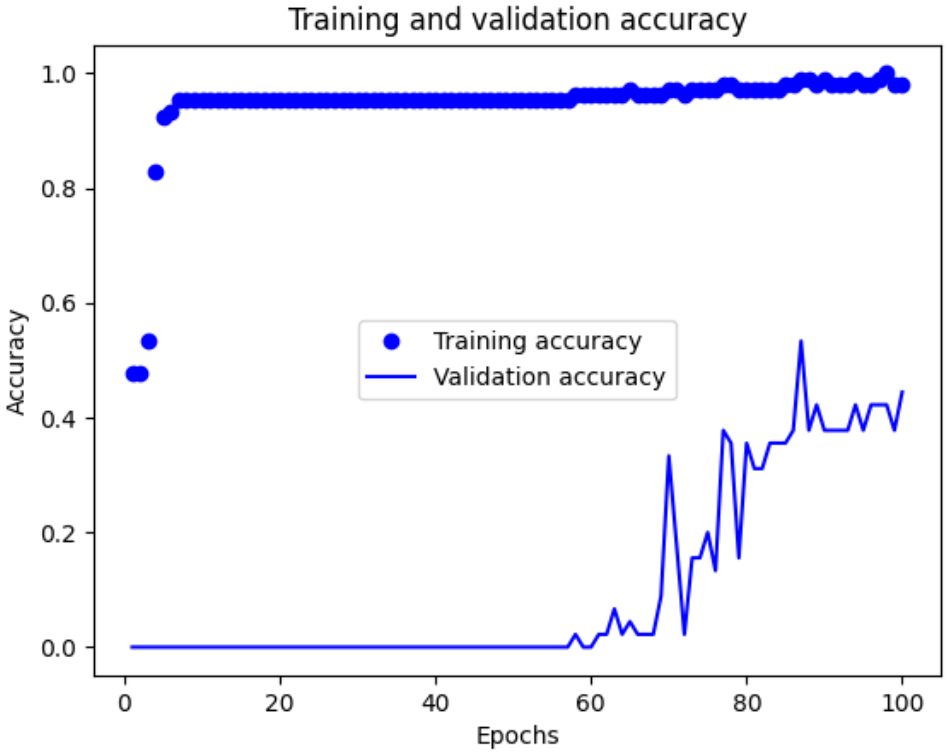

Рисунок 14 – График точности

Теперь уменьшим число проверочных до 0,005. Точность вычислений равна 98,7%, выросла по сравнению с исходной цепью. Графики представлены на рис. 15-16.

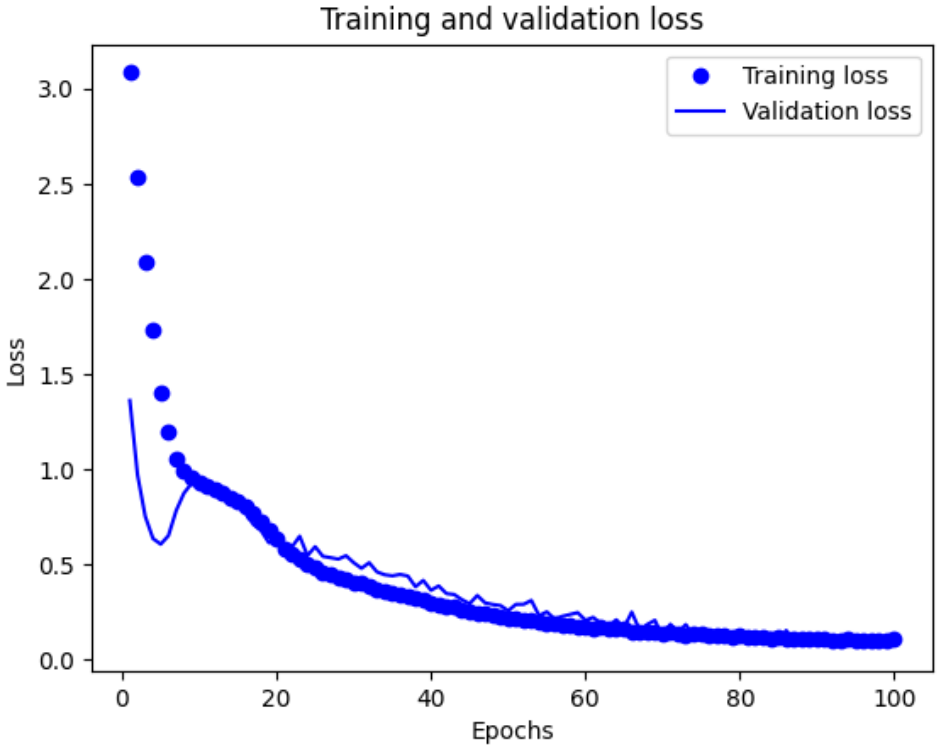

Рисунок 15 – График ошибок

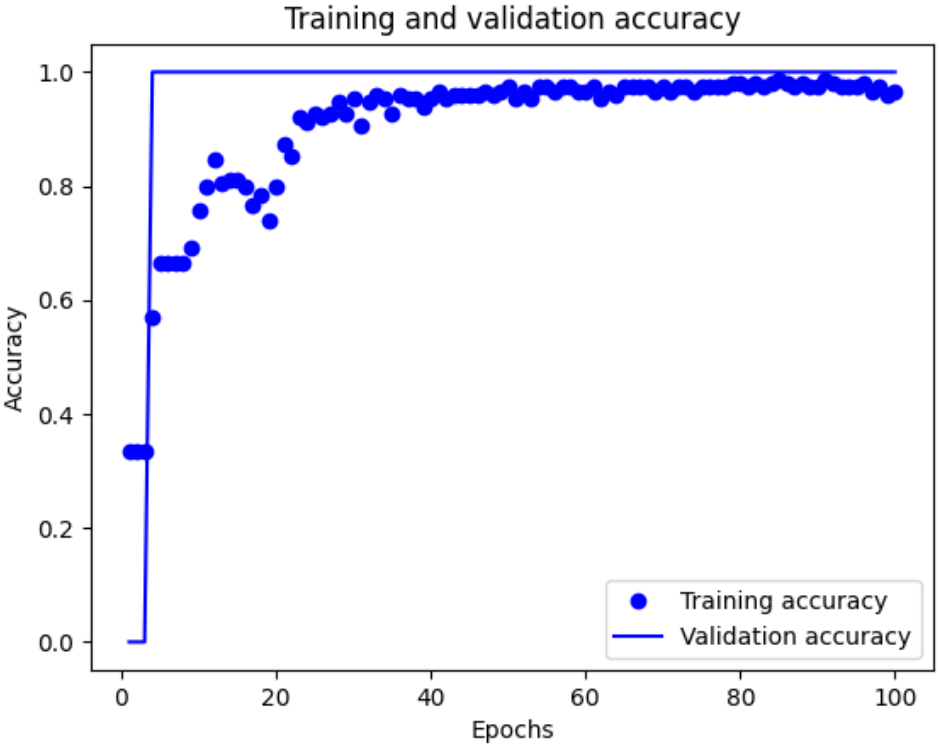

Рисунок 16 – График точности

Сделаем число эпох 150. Точность практически не изменилась, 98,7%. Графики изображены на рис. 17-18.

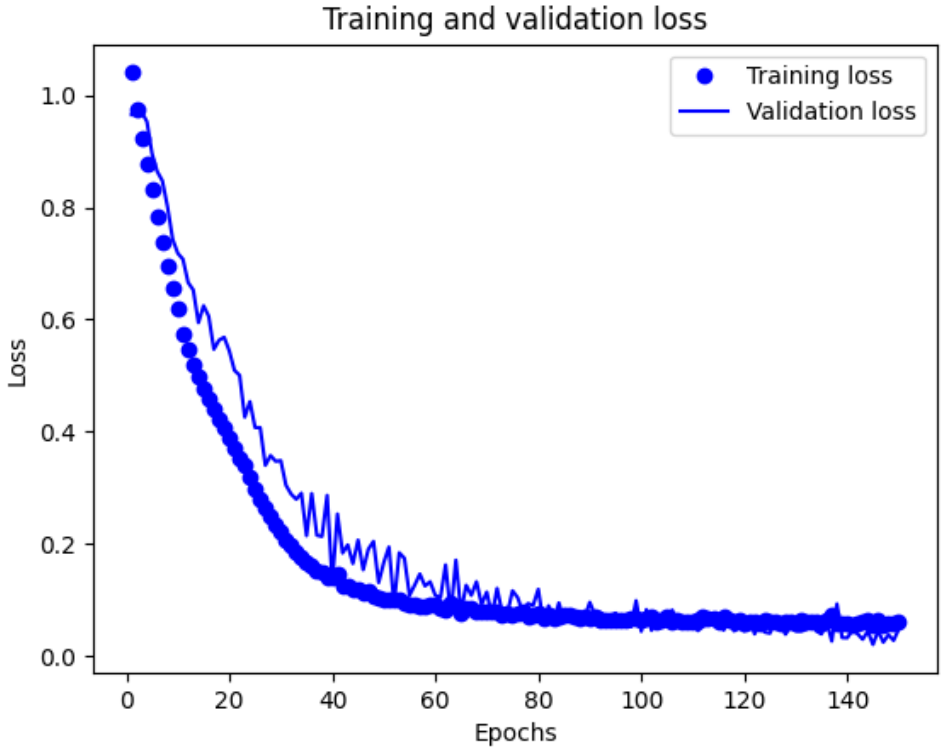

Рисунок 17 – График ошибок

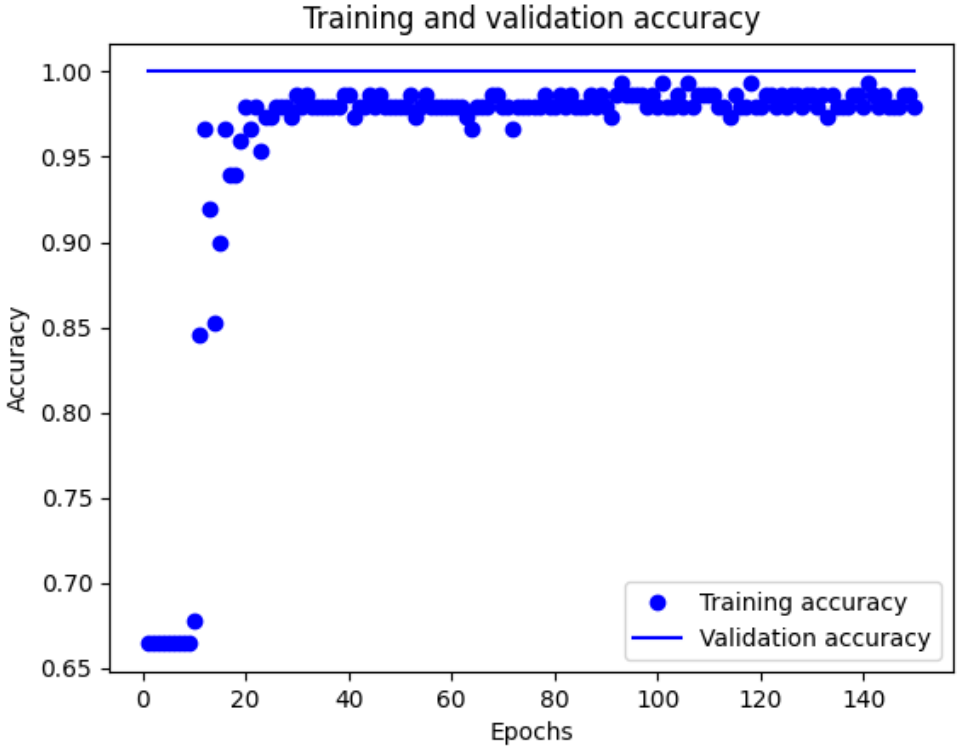

Рисунок 18 – График точности

Увеличим число проверочных до 0,01. Точность 98,8% и значение ошибки 0,08, что ниже исходного. Графики представлены на рис. 19-20.

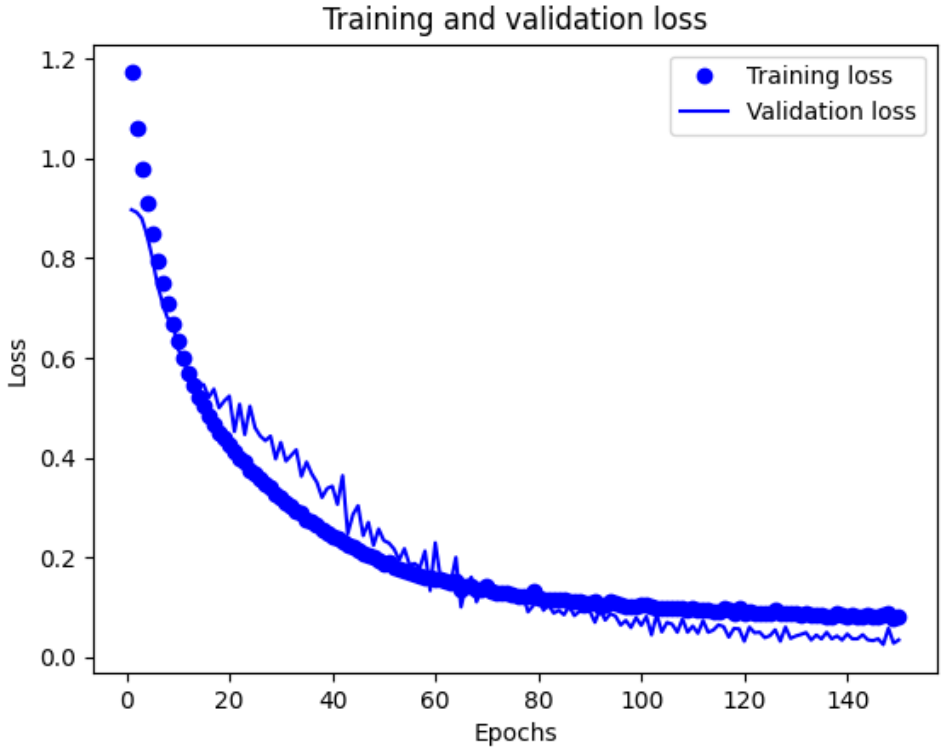

Рисунок 19 – График ошибок

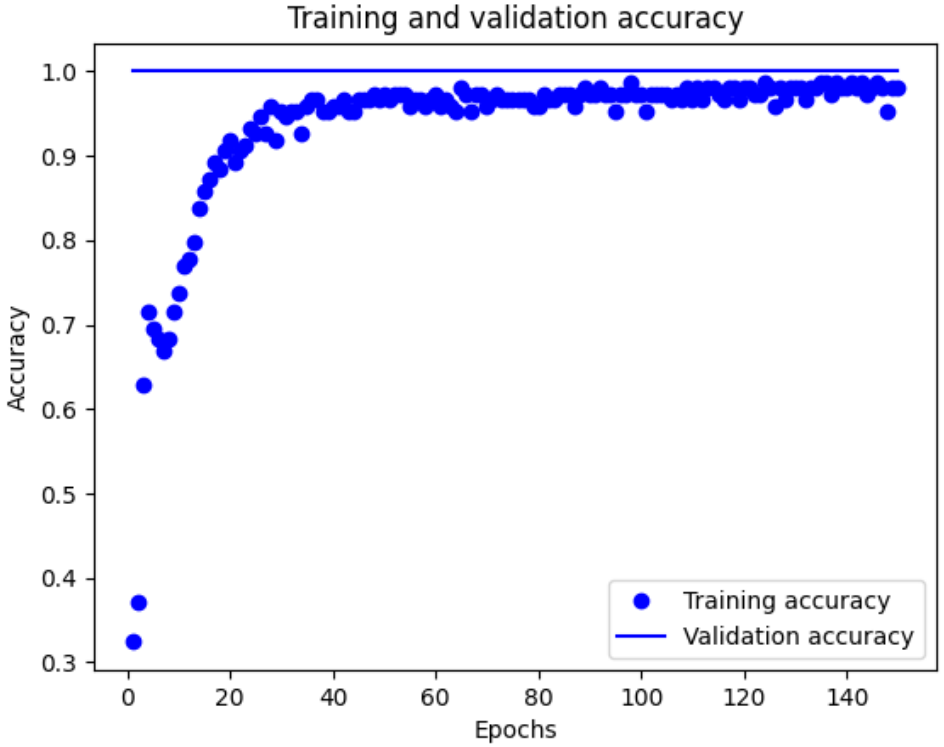

Рисунок 20 – График точности

3. Выбрать наилучшую модель

Была выбрана следующая модель с параметрами обучения:

```
model = Sequential()
model.add(Dense(8, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
hist = model.fit(X, dummy_y, epochs=150, batch_size=10,
validation_split=0.01)
```

Ее точность составила 98,8%. Графики представлены на рис. 19-20.

Вывод

В процессе выполнения лабораторной работы были изучены задачи классификации и создана модель искусственной нейронной сети. Также были исследованы разные виды архитектур и обучение с различными параметрами, построены графики ошибок и точности в ходе обучения. Была выбрана лучшая модель.

Было установлено, что малое количество нейронов может быть недостаточно для решения задачи, а количество слоев в сети повышает точность; с увеличением числа эпох точность растет, а с сильным увеличением процента проверочных данных – падает.

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
import matplotlib.pyplot as plt
# загрузка данных
dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:4].astype(float)
Y = dataset[:, 4]
# переход от текстовых меток к категориальному виду
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
# создание модели
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))
# инициализация параметров обучения
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
# обучение сети
hist = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
history_dict = hist.history
# потери сети на обучающих данных и потери на данных, не участвовавших в
обучении
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
# точность сети на обучающих данных и точность на данных, не участвовавших в
обучении
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
plt.plot(epochs, acc_values, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc_values, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
res = model.evaluate(X, dummy_y)
print(res)
```