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

Студент гр. 8383 _____ Муковский Д. В.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2021

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.

Задание

- 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 функций *fit*)
3. Построить графики ошибок и точности в ходе обучения
4. Выбрать наилучшую модель

Выполнение работы.

1. Сначала были импортированы все нужные классы и функции (весь код представлен в приложении А). А именно: Keras - для собственно создания и обучения ИНС, Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели.
2. После чего были разделены атрибуты на входные данные(X) и выходные данные(Y). Выходные данные были переведены в категориальные вектора.
3. Далее была создана базовая архитектура сети, состоящая из двух полносвязных слоев (Dense), первый слой из 4 нейронов для входных

данных с активизирующей функцией RELU, а второй из 3 нейронов - переменный слой потерь с активизирующей функцией Softmax, которая возвращает массив с 3 оценками вероятностей, каждая оценка определяет вероятность принадлежности текущего набора данных к одному из 3 классов растений.

4. Далее были инициализированы параметры обучения. И теперь можно начинать обучение, для чего в случае использования библиотеки Keras достаточно вызвать метод fit сети — он пытается адаптировать (fit) модель под обучающие данные.
5. После чего происходила отрисовка графиков потери сети на обучающих данных и точность сети на обучающих данных, а также потери и точность на данных, не участвовавших в обучении.

Тесты

1. Результаты тестирования стартовой ИНС представлены на рис. 1 и 2.

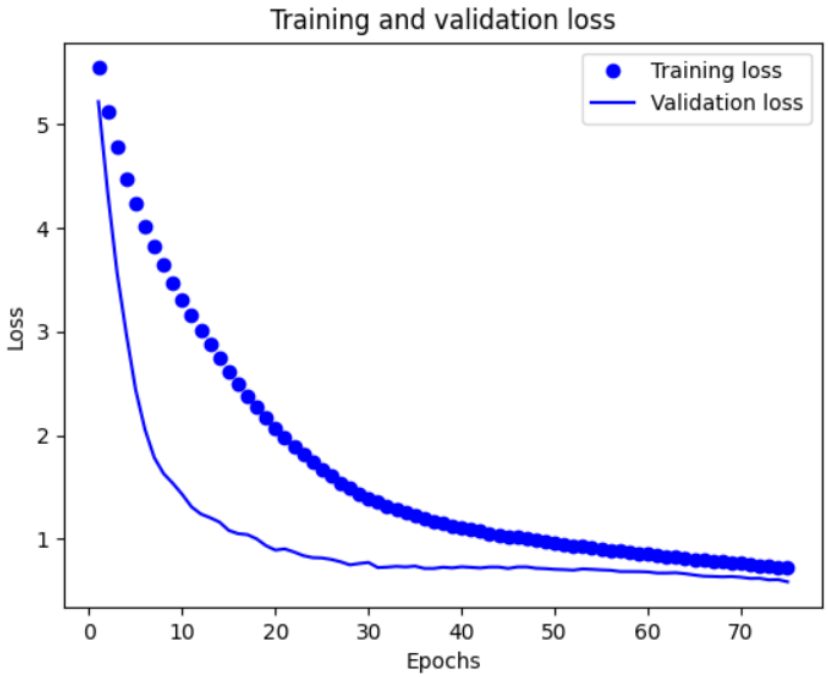

Рисунок 1 – График потери ИНС

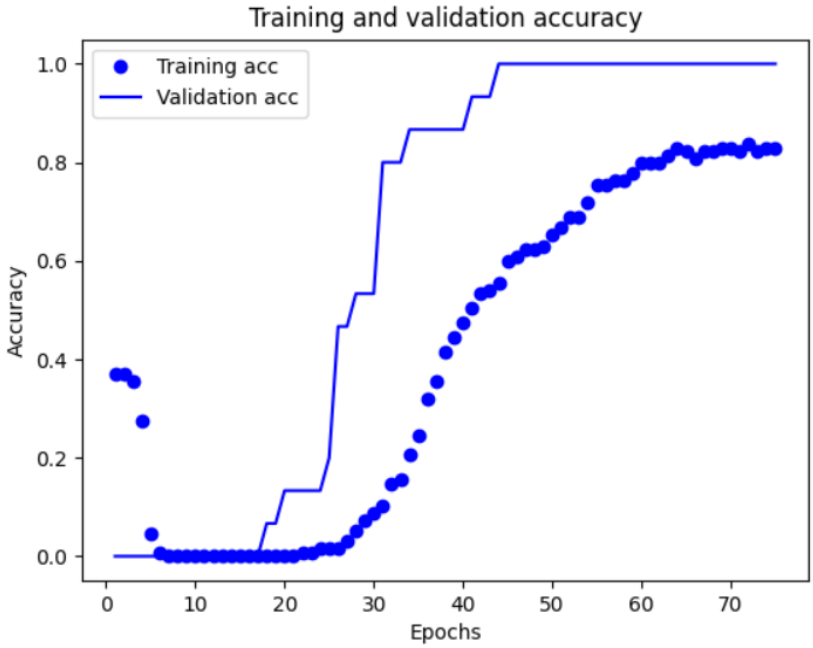

Рисунок 2 – График точности ИНС

Обучение данной сети нельзя назвать стабильным. Результаты, представленные на графиках, менялись раз за разом после каждого нового запуска обучения. Поэтому были внесены некоторые изменения в архитектуру сети.

2. Для начала посмотрим, как наша сеть будет вести себя при увеличении количества нейронов на первом слое. Увеличим их до 8.

Результаты тестирования стартовой ИНС представлены на рис. 3 и 4.

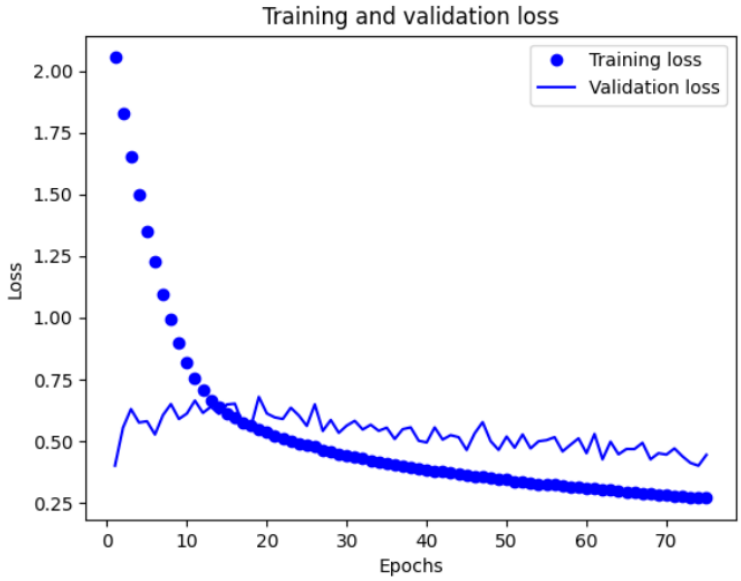

Рисунок 3 – График потери ИНС

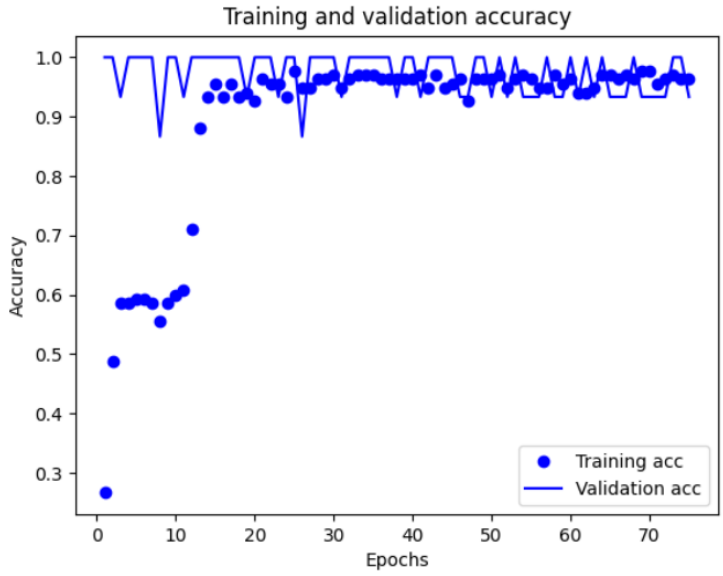

Рисунок 4 – График точности ИНС

В данном случае количество потерь значительно уменьшилась, как и точность. Притом точность тестируемых данных значительно увеличилась, а точность на данных, не участвовавших в тестировании, немного уменьшилась, я списываю это на погрешность.

3. При увеличении количества нейронов до 24 можно немного улучшить результаты. Функции потери и точности представлены на рис. 5 и 6.

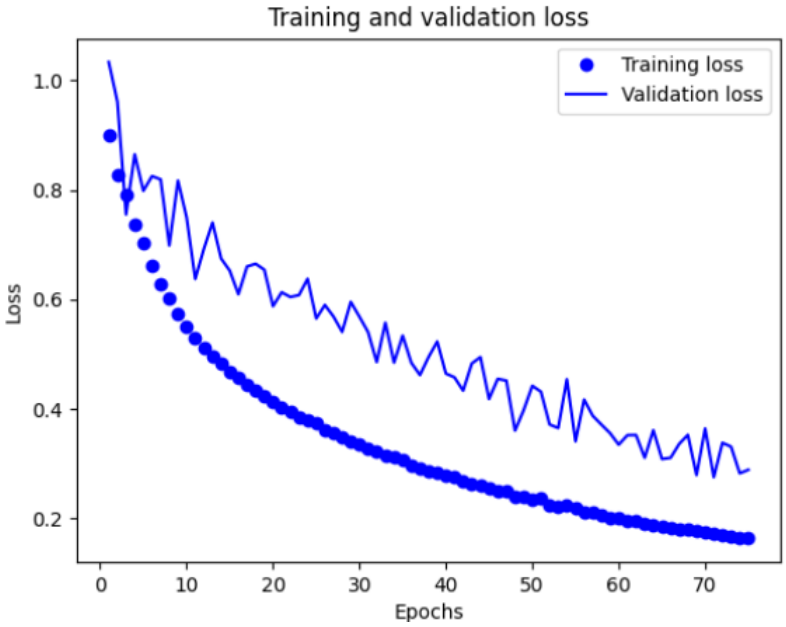

Рисунок 5 – График потери ИНС

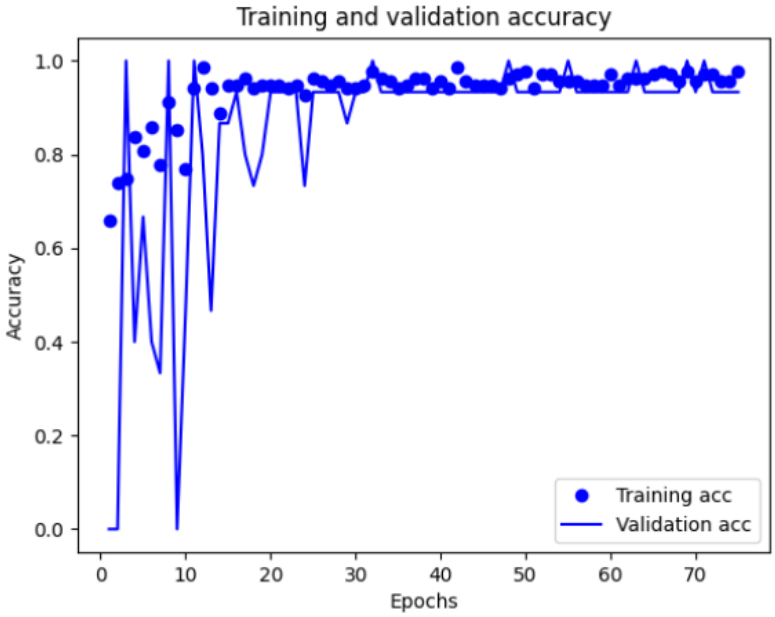

Рисунок 6 – График точности ИНС

4. При дальнейшем увеличении нейронов, ИНС улучшается незначительно, либо вовсе не улучшается, на рисунках 7 и 8 можно увидеть результат для 50 нейронов.

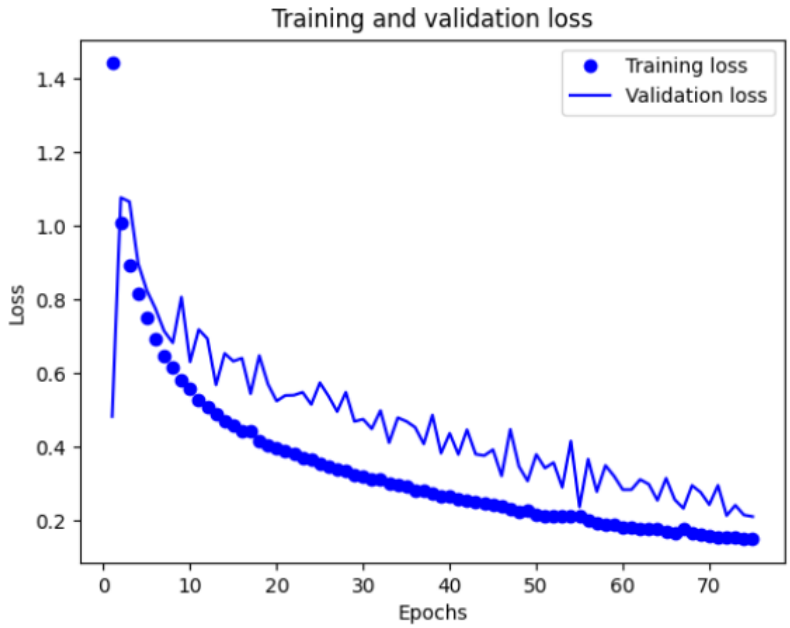

Рисунок 7 – График потери ИНС

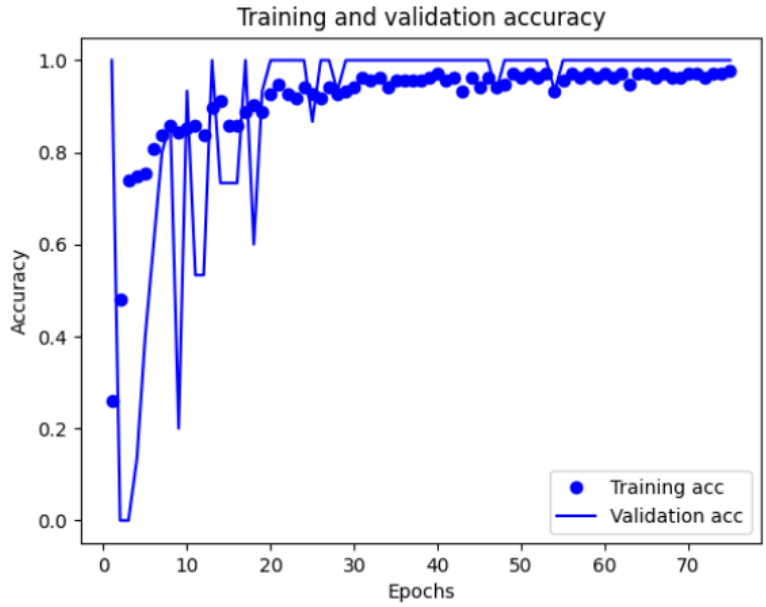

Рисунок 8 – График точности ИНС

Ошибка уменьшилась примерно с 0.17 до 0.15, притом точность на тестируемых данных тоже выросла и стала практически идеальной.

5. Попробуем изменить архитектуру ИНС, добавив в нее еще один слой. Итого два слоя по 24 нейрона и переменный слой потерь с 3 нейронами. Результаты можно увидеть на рисунках 9 и 10.

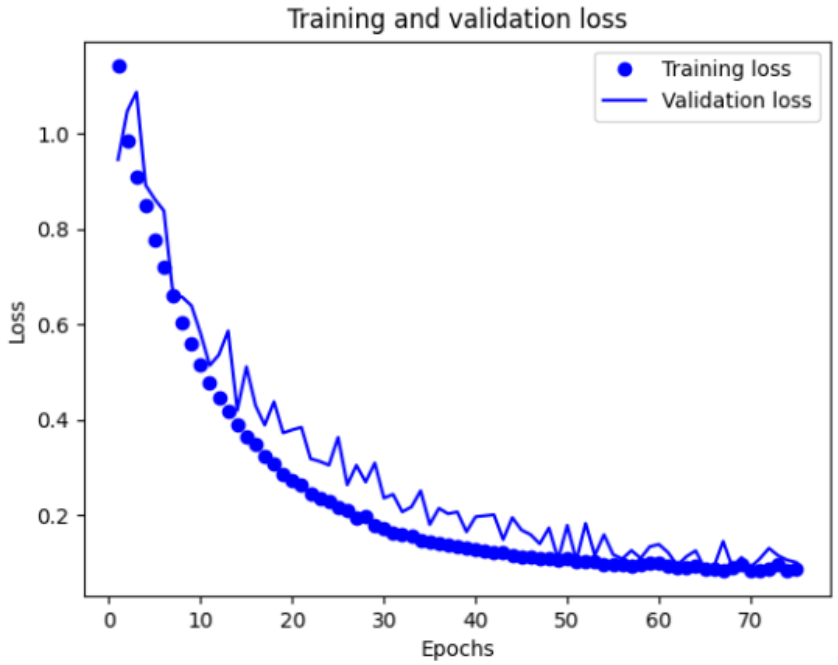

Рисунок 9 – График потери ИНС

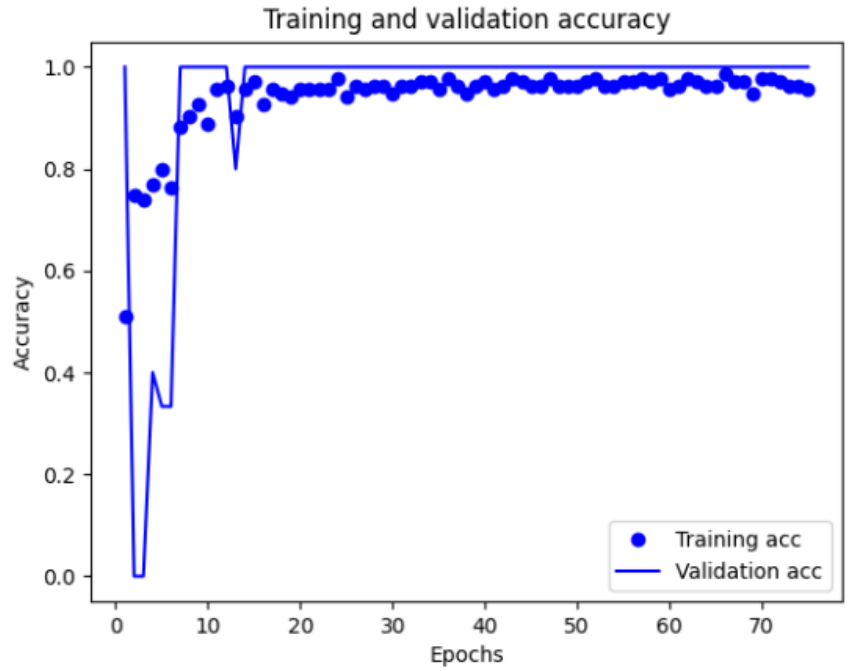

Рисунок 10 – График точности ИНС

Точность еще больше подросла, на тестируемых данных после 15 эпохи она стала идеальной. Ошибки также уменьшились с 0.15 до 0.9.

6. Попробуем увеличить количество эпох с 75 до 200. Результаты приведены на рис. 11 и 12.

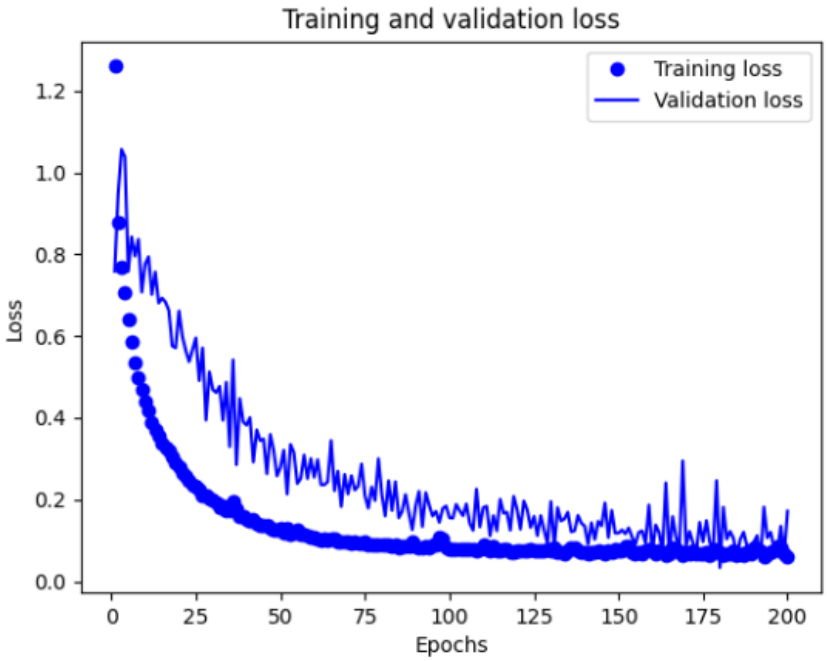

Рисунок 11 – График потери ИНС

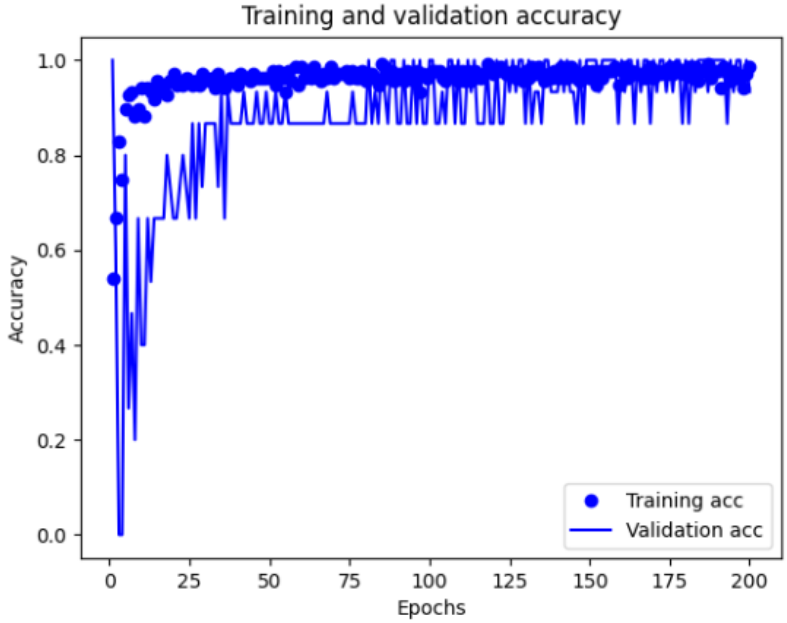

Рисунок 12 – График точности ИНС

Как видно точность и потери после 75 эпохи особенно не изменились, поэтому вернем количество эпох обратно.

7. Увеличим процентное соотношение данных, которое не участвуют в обучении до 20. Результаты приведены на рис. 13 и 14.

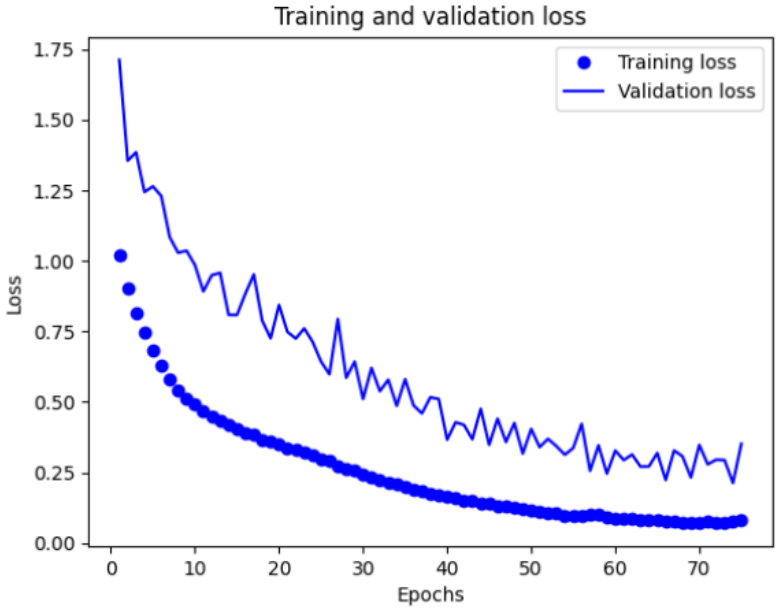

Рисунок 13 – График потери ИНС

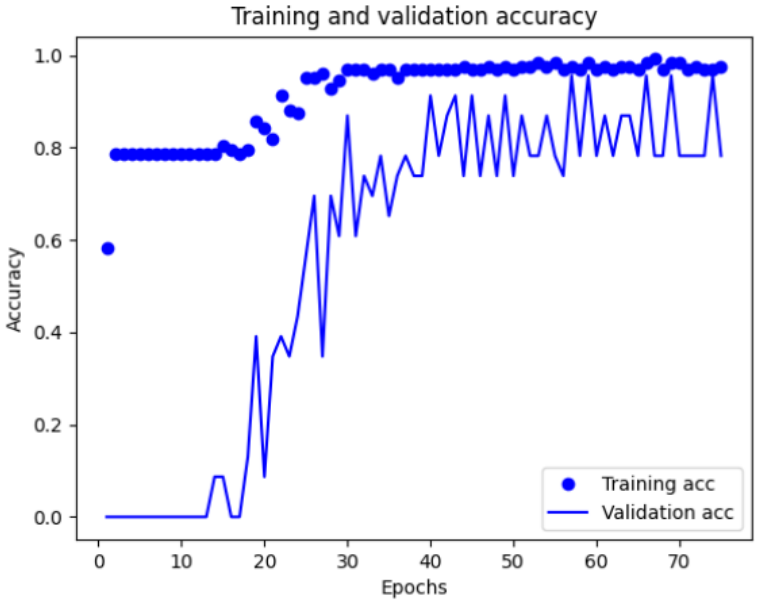

Рисунок 14 – График точности ИНС

Количество данных, на которых тренируется ИНС уменьшилось, а количество данных, на которых тестируется увеличилась. По графикам видно, что ИНС научилась довольно хорошо определять данные, на которых она тренировалась за 40 эпох, но она не улучшила результат для тестируемых данных, что означает, что надо вернуть предыдущее значение.

8. Изменим batch_size с 10 на 5, то есть ИНС будет чаще корректировать веса. Но также количество вычислений увеличиться, а следовательно, и производительность. Результаты приведены на рис. 15 и 16.

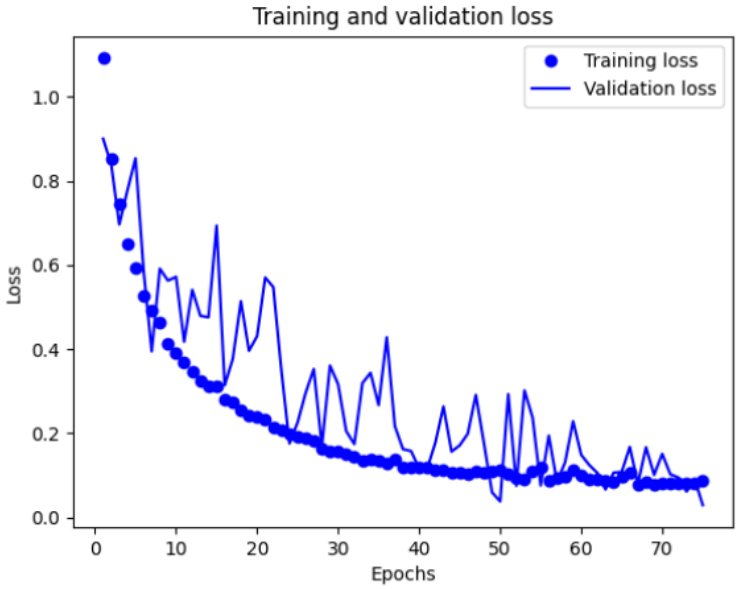

Рисунок 15 – График потери ИНС

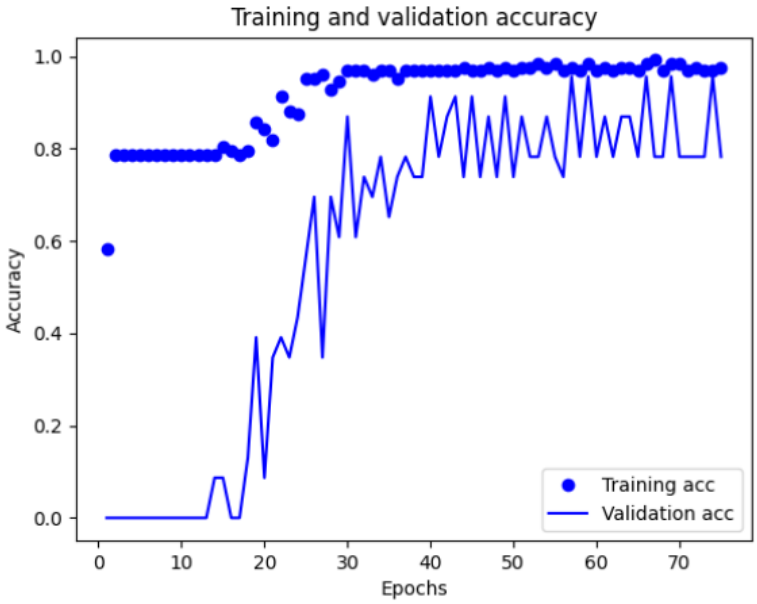

Рисунок 16 – График точности ИНС

Результат не сильно изменился с новым параметром batch_size, поэтому увеличение количества вычислений и перебалансировки весов бессмысленна. Лучшая конфигурация нейросети представлена в приложении А.

Выводы.

При выполнении работы мы ознакомились с задачей классификации, создали модель ИНС и настроили оптимальные параметры обучения, при которых ИНС работает наилучшим способом.

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

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)

model = Sequential()

model.add(Dense(24, activation='relu', input_shape=(4,)))
model.add(Dense(24, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

res = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1,
verbose=1)

history_dict = res.history

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()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```