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. 8382 _____ Черницын П.А.

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.

Задачи

Ознакомиться с задачей классификации

Загрузить данные

Создать модель ИНС в Keras

Настроить параметры обучения

Обучить и оценить модель

Требования

Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)

Изучить обучение при различных параметрах обучения (параметры функции fit)

Построить графики ошибок и точности в ходе обучения

Выбрать наилучшую модель

Выполнение работы

В программу загружаются данные из файла *iris.csv*. Разделяются входные данные (размеры пестиков и тычинок цветков) и выходные данные (сорта растений). Выходные данные представляются в виде категориального вектора.

Задается начальная модель нейронной сети. В нее включается два слоя. Второй слой – выходной, который возвращает массив с тремя оценками вероятностей принадлежности текущего цветка к одному из трех классов.

```
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Обучение происходит в течение 75 эпох, размер батча – 10.

На рисунке 1 изображены графики потерь, а на рисунке 2 – графики точности на тренировочном и валидационном множествах. Как видно на графиках, потери стремятся к нулю, а точность возрастает, однако это происходит недостаточно быстро.

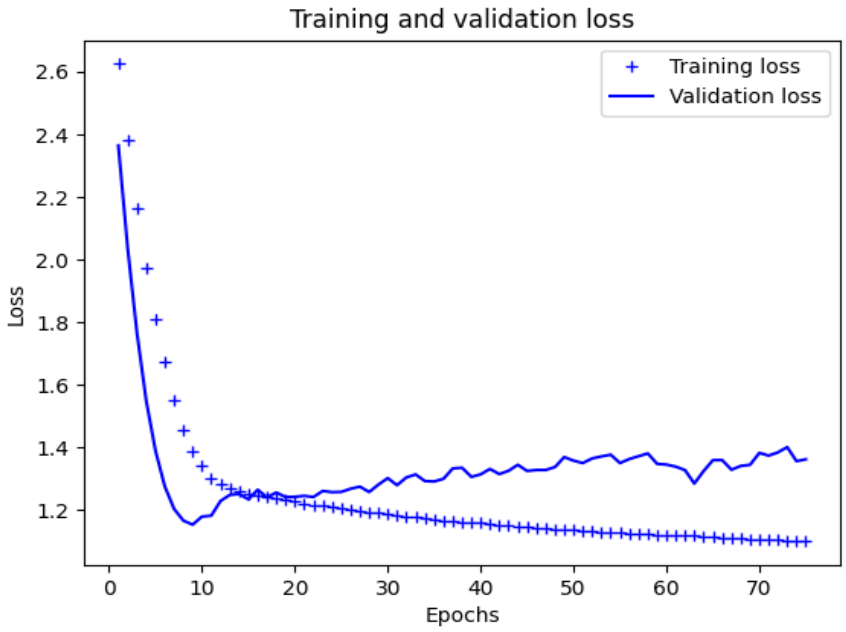

Рисунок 1

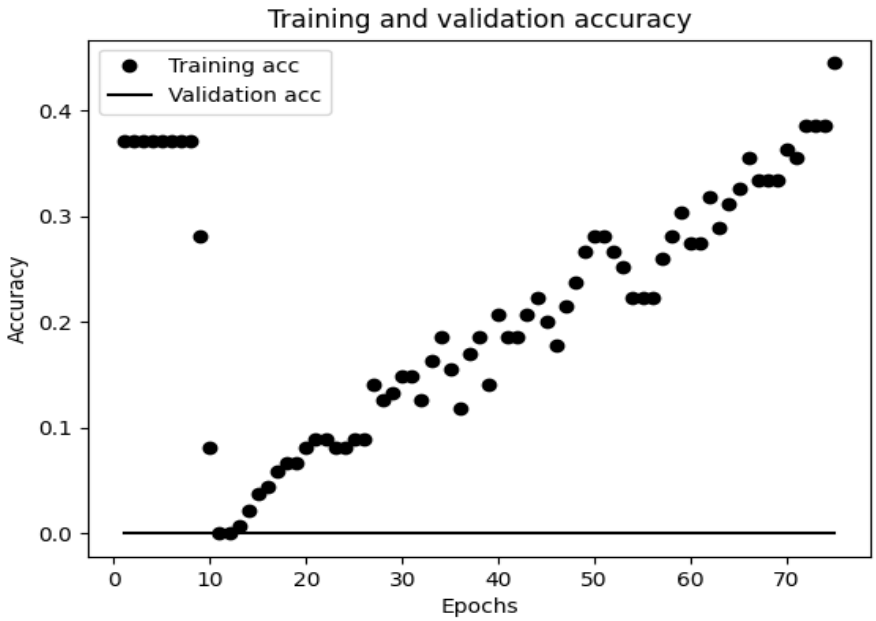

Рисунок 2

Изменим количество нейронов на промежуточном слое сети:

```
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Графики потерь изображены на рисунке 3, графики точности – на рисунке 4

4 Скорость обучения заметно возросла: точность модели на валидационном множестве становится близка к 1 уже примерно на 50-ой эпохе обучения.

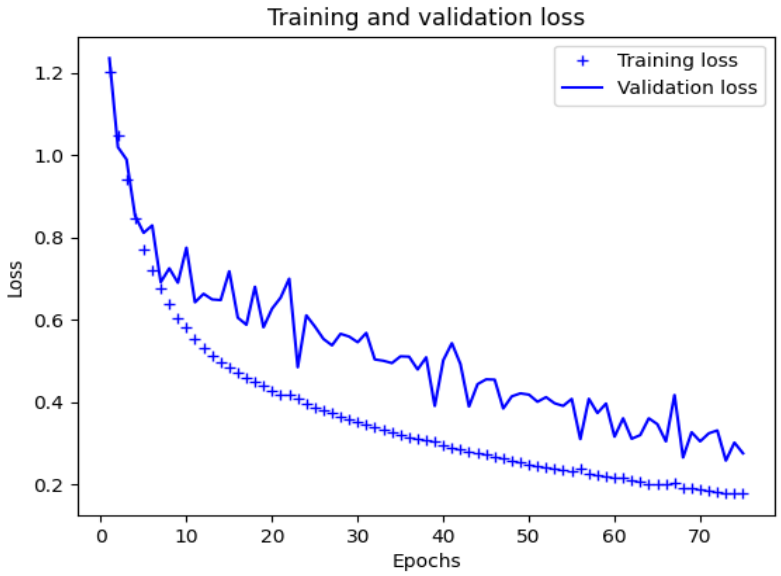

Рисунок 3

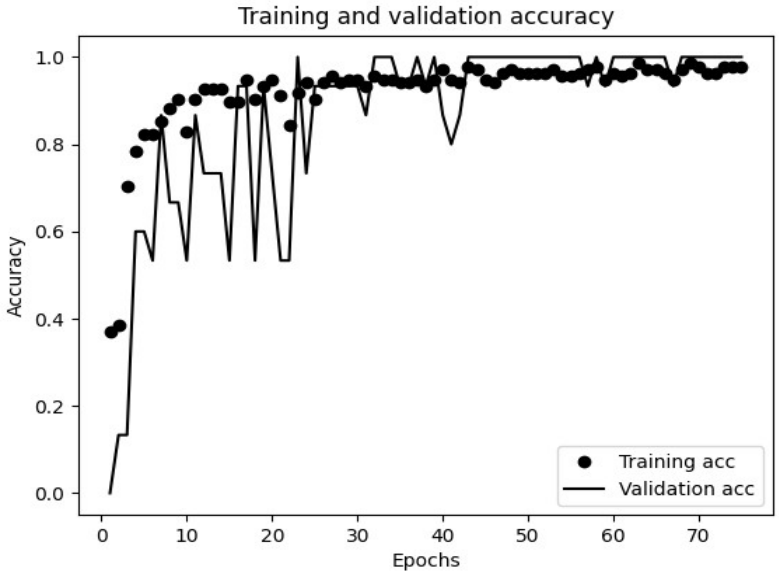

Рисунок 4

Добавим в модель еще один промежуточный слой:

```
model.add(Dense(30, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

На рисунке 5 изображены графики потерь. Можно заметить, что потери несколько снизились по сравнению с моделью с одним промежуточным слоем (рис. 3). На рисунке 6 изображены графики точности. Точность по сравнению с рассмотренной ранее моделью практически не изменилась.

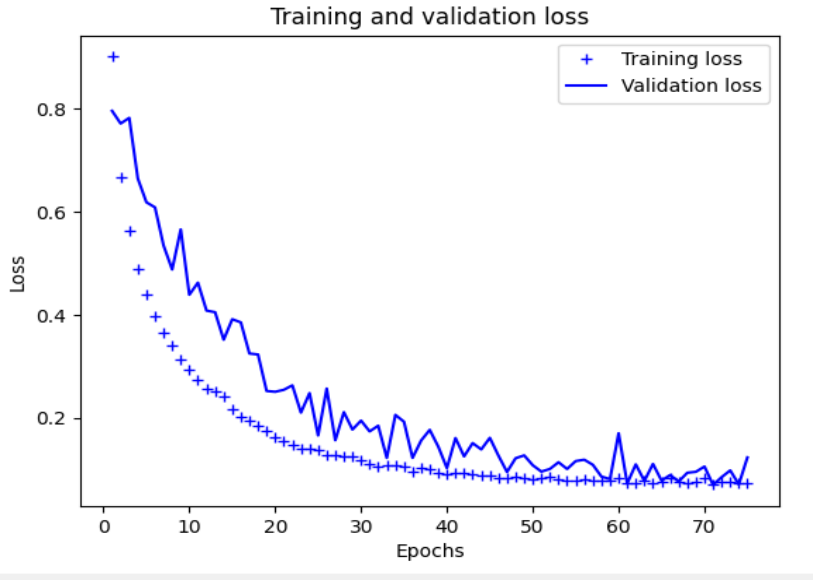

Рисунок 5

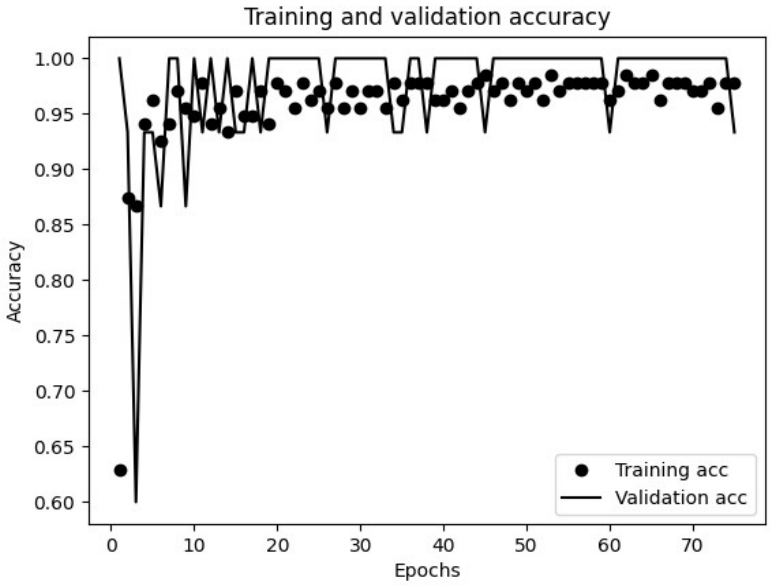

Рисунок 6

Попробуем изменить размер батча с 10 до 15:

```
model.fit(X, dummy_y, epochs=75, batch_size=15,
validation_split=0.1)
```

На рисунках 7 и 8 изображены графики потерь и точности соответственно. На графике точности видно, что, по сравнению с предыдущей моделью, текущая модель дает менее стабильный результат, так что имеет смысл вернуться к предыдущим параметрам обучения.

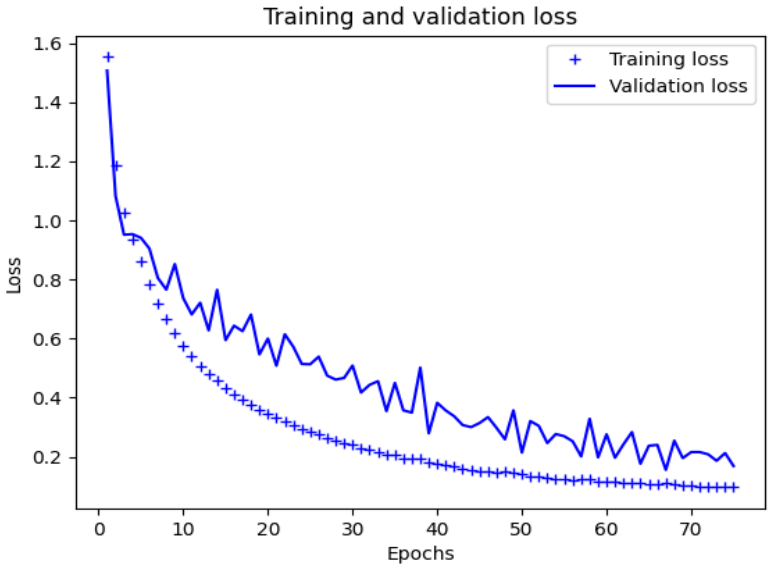

Рисунок 7

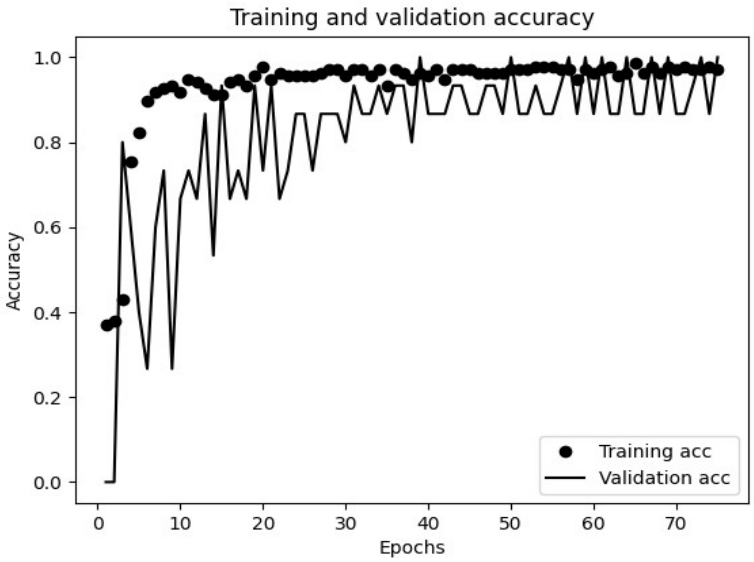

Рисунок 8

Увеличим число этапов обучения:

model.fit(X, dummy_y, epochs=100, batch_size=10, validation_split=0.1)

На рисунке 9 изображены графики ошибок, на рисунке 10 – графики точности.

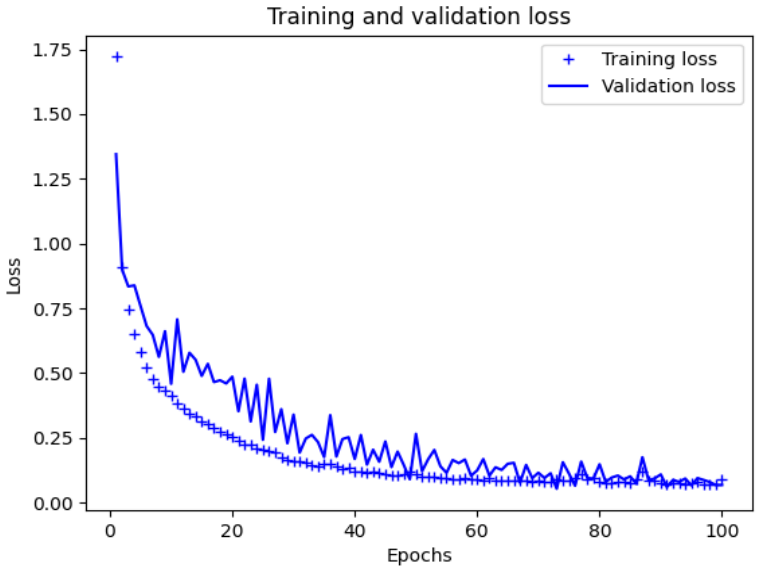

Рисунок 9

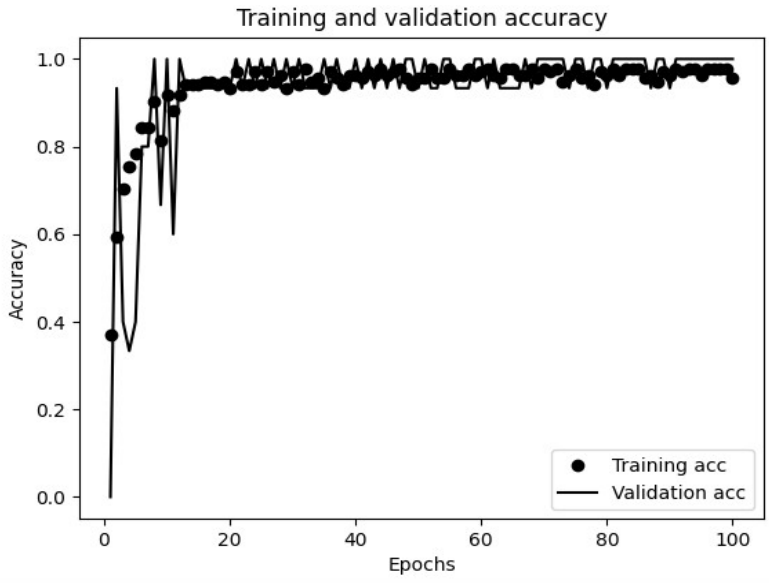

Рисунок 10

Данная модель обеспечивает наилучшую точность и наименьшие потери из всех рассмотренных. Дальнейших попыток улучшения предпринимать не

будем. Полный код программы, реализующей эту модель, приведен в приложении А.

Выводы

Была создана и обучена модель искусственной нейронной сети, осуществляющей многоклассовую классификацию цветов, при помощи библиотеки Keras. Были рассмотрены и оценены различные архитектуры сети, т. е. модели сети с различным количеством слоев и нейронов на слоях, а также обучения с различными параметрами. В результате выбрана модель, обеспечивающая наилучшую точность.

ПРИЛОЖЕНИЕ А

Полный код программы

```
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(30, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
h = model.fit(X, dummy_y, epochs=100, batch_size=10, validation_split=0.1)

#graphics
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)
#1
plt.plot(epochs, loss, 'b+', 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()
#2
plt.plot(epochs, acc, 'ko', label='Training acc')
plt.plot(epochs, val_acc, 'k', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```