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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искуссенные нейронные сети»

Тема: Многоклассовая классификация цветов

Студент гр. 7382 _____ Находько А.Ю.

Преподаватель _____ Жукова Н.В.

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 работы.

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

Требования работы.

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

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.

Эксперимент первый.

В программе из Приложения А установим первоначальные данные для нейронной сети. Протестируем модель при начальных данных.

Количество эпох — 75;

Количество слоёв — 2;

Количество нейронов на 1 слое — 4;

Количество нейронов на 2 слое — 3;

Полученные результаты:

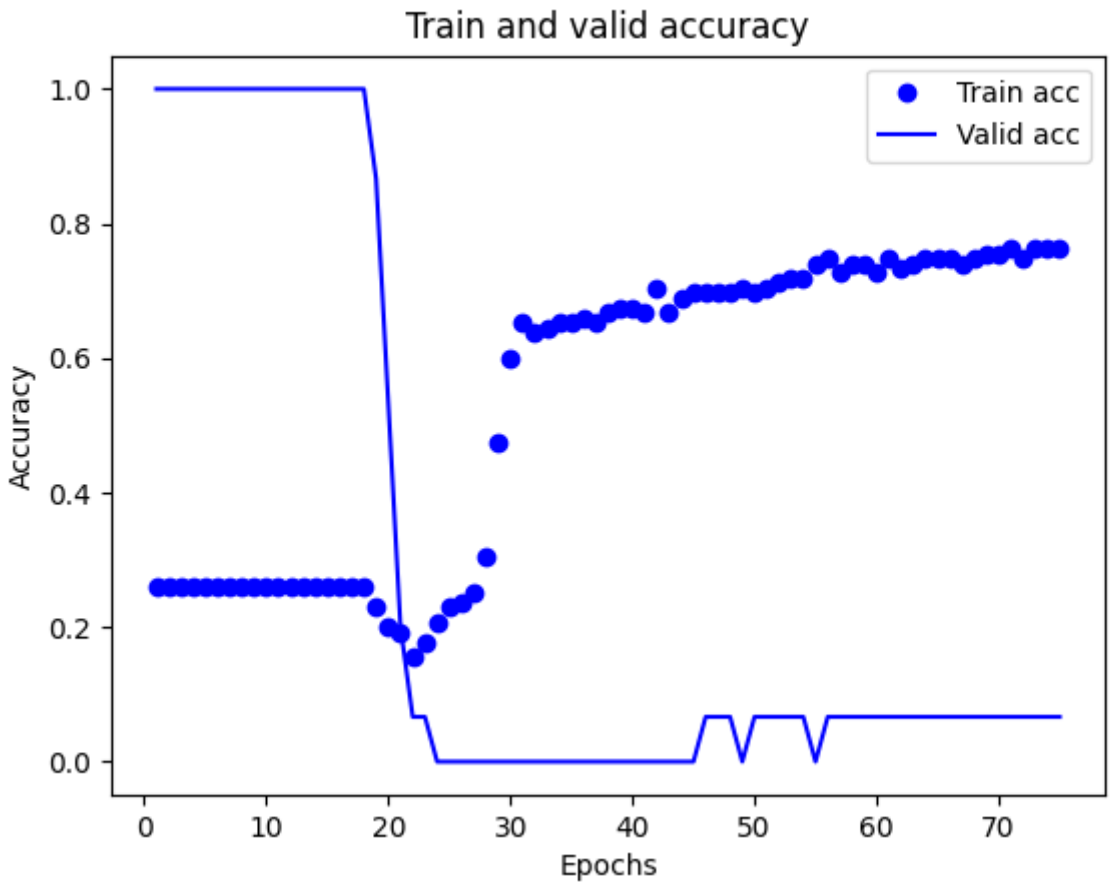

Рисунок 1 — График точности модели при исходных данных

Эксперимент второй.

Попробуем протестировать модель посредством увеличения количества проходов до 3000. Количество слоёв и нейронов в них останутся такими же, как в предыдущем эксперименте.

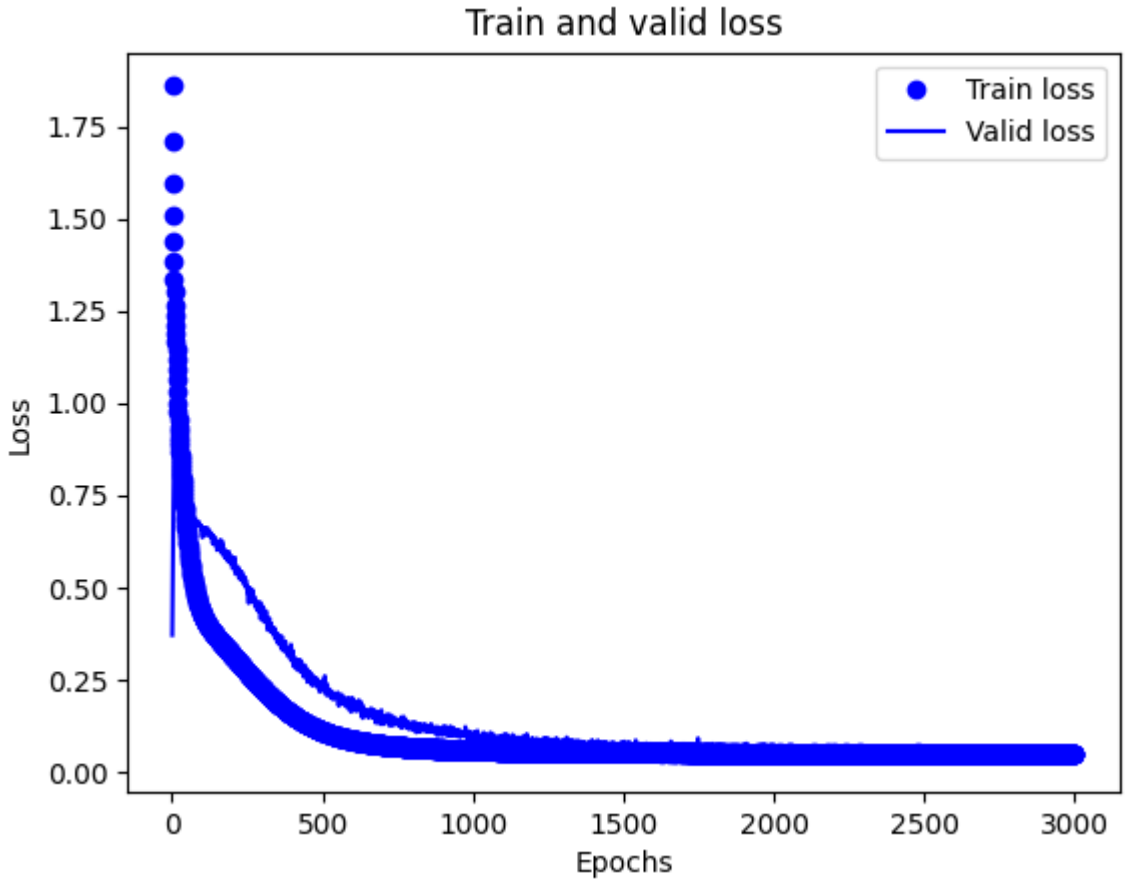

Рисунок 3 — График потерь модели при поднятии количества эпох

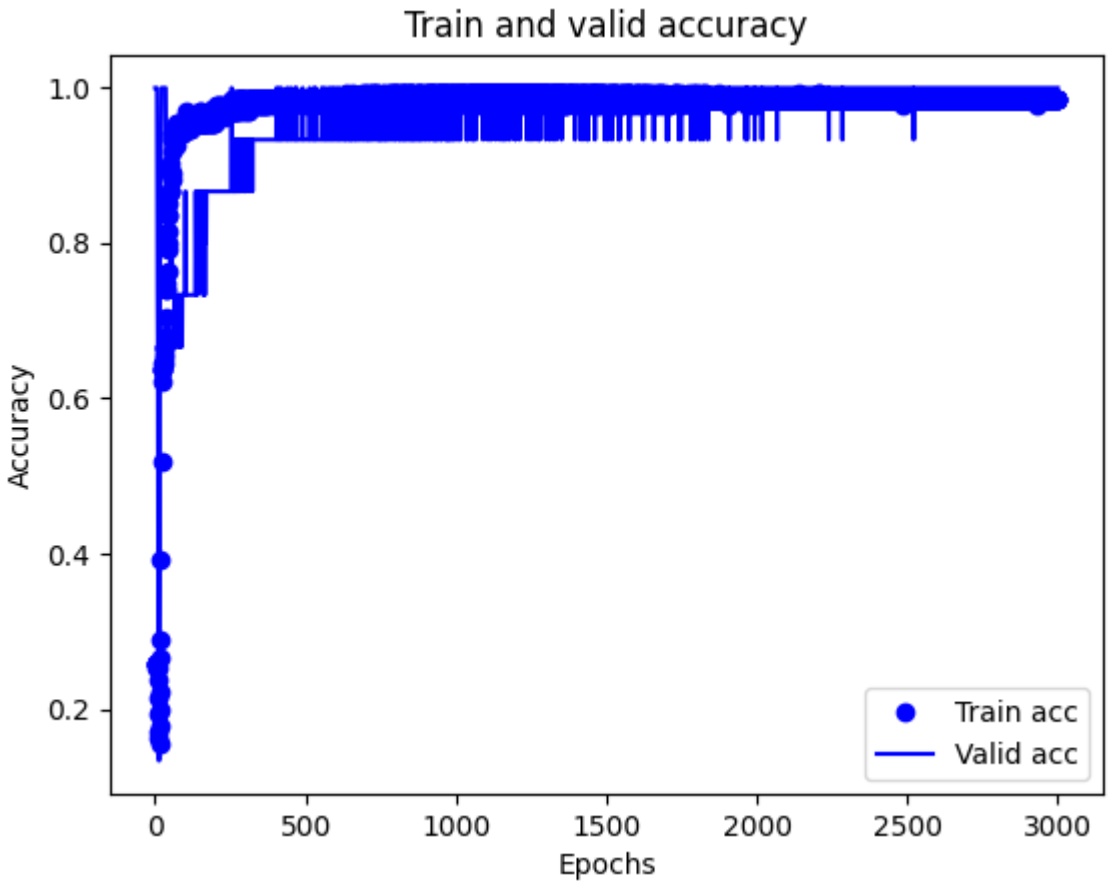

Рисунок 3 — График точности модели при поднятии количества эпох

Эксперимент третий.

Попробуем протестировать модель путём сокращения количества проходов до 300. Количество слоёв не поменяется, а количество нейронов увеличим на первом слое до 16.

Заметим проанализировав полученные ниже графики из данного эксперимента, что потери были минимизированы за 140 поколений, а максимальная точность была достигнута за 60 поколений.

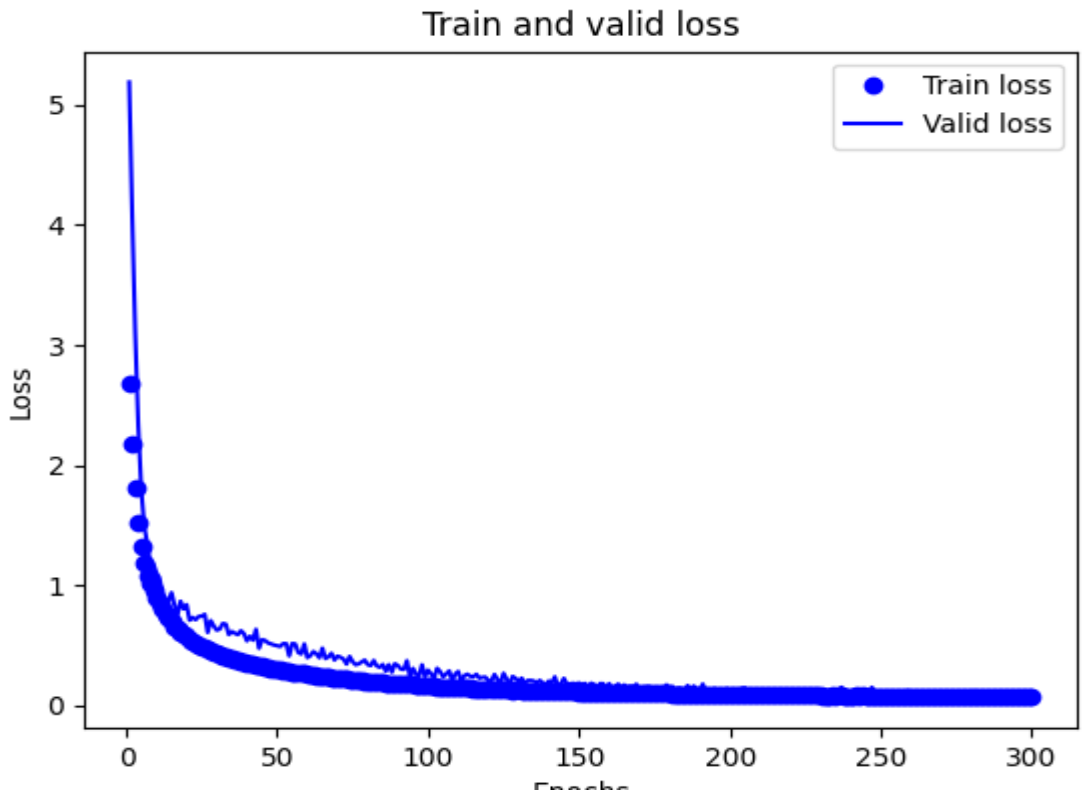

Рисунок 4 — График потерь модели при увеличении количества нейронов

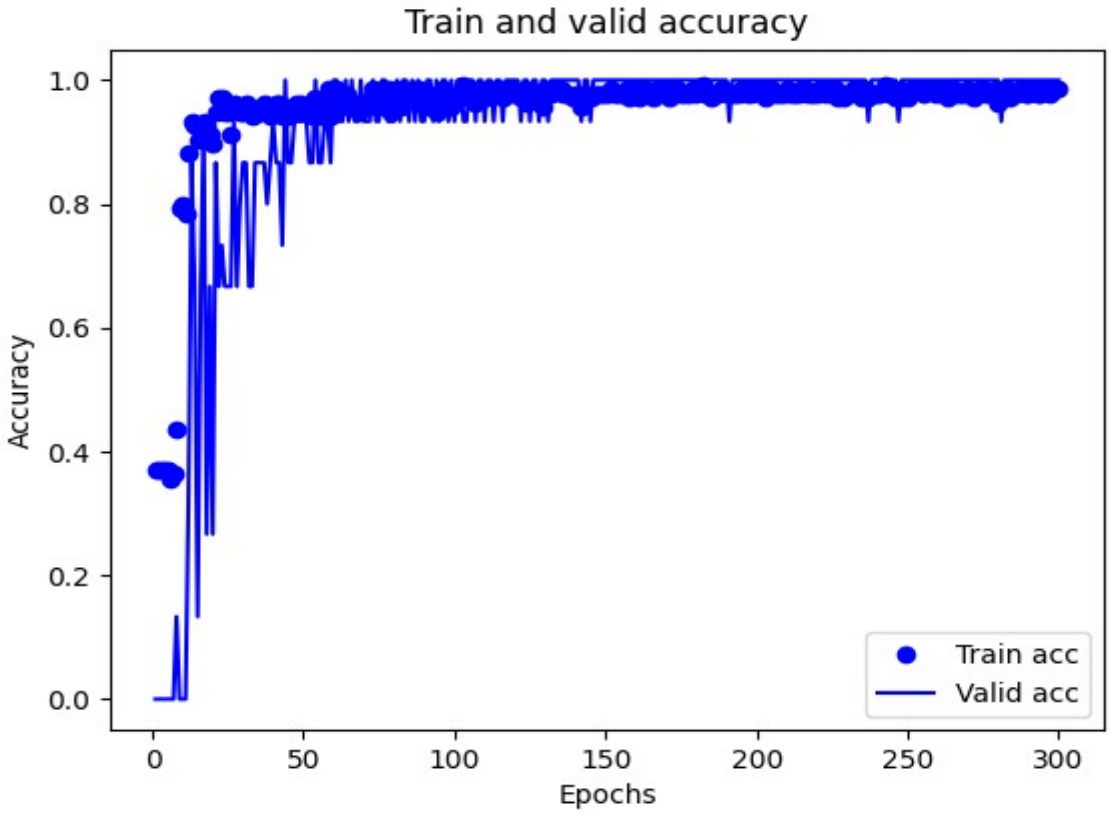

Рисунок 5 — График точности модели при увеличении количества нейронов

При проведении дальнейших экспериментов с увеличением количества слоёв, результаты не становились существенно лучше, поэтому установим что данная конфигурация модели считается наилучшей.

Выводы.

В ходе выполнения лабораторной работы были изучены различные архитектуры ИНС, также провёл обучение сети при различных параметрах обучения. Были построены графики ошибок и точности в ходе обучения и выбрана наилучшая модель.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import pandas
from keras.layers import Dense
from keras.models import Sequential
from 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(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossent
ropy', metrics=['accuracy'])
H = model.fit(X, dummy_y, epochs=300, 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, 'bo', label='Train loss')
plt.plot(epochs, val_loss, 'b', label='Valid loss')
plt.title('Train and valid loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()
plt.plot(epochs, acc, 'bo', label='Train acc')
plt.plot(epochs, val_acc, 'b', label='Valid acc')
plt.title('Train and valid accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```