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. 8383 _____ Бессуднов Г. И.

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.

Основные теоретические положения

Задача многоклассовой классификации является одним из основных видов задач, для решения которых применяются нейронные сети. В листинге 1 представлен пример данных.

Листинг 1 - Пример данных

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели (листинг 2).

Листинг 2 - Подключение модулей

```
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
```

Набор данных загружается напрямую с помощью pandas. Затем необходимо разделить атрибуты (столбцы) на входные данные(X) и выходные данные(Y) (листинг 3).

Листинг 3 - Загрузка данных

```
dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]
```

При решении задач многоклассовой классификации хорошей практикой является преобразование выходных атрибутов из вектора в матрицу к виду представленных в листинге 4. Для этого необходимо использовать функцию to_categorical()(Листинг 5)

Листинг 4 - Представление данных

```
Iris-setosa, Iris-versicolor, Iris-virginica
1,    0,    0
0,    1,    0
0,    0,    1
```

Листинг 5 - Переход от текстовых меток к категориальному вектору

```
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
```

Теперь можно задать базовую архитектуру сети (листинг 6)

Листинг 6 - Создание модели

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

Основным строительным блоком нейронных сетей является слой (или уровень), модуль обработки данных, который можно рассматривать как фильтр для данных. Он принимает некоторые данные и выводит их в более полезной форме. В частности, слои извлекают представления из подаваемых в них данных, которые, как мы надеемся, будут иметь больше смысла для решаемой задачи. Фактически методика глубокого обучения заключается в объединении простых слоев, реализующих некоторую форму поэтапной очистки данных. Модель глубокого обучения можно сравнить с ситом, состоящим из последовательности фильтров все более тонкой очистки данных — слоев.

В данном случае наша сеть состоит из последовательности двух слоев Dense, которые являются тесно связанными (их еще называют полносвязными) нейронными слоями. Второй (и последний) слой — это 3-переменный слой потерь (softmax layer), возвращающий массив с 3 оценками вероятностей (в сумме дающих 1). Каждая оценка определяет вероятность принадлежности текущего изображения к одному из 3 классов цветов.

Чтобы подготовить сеть к обучению, нужно настроить еще три параметра для этапа компиляции:

1.функцию потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении;

2.оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь;

3.метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

Листинг 6 - Инициализация параметров обучения

```
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])
```

Теперь можно начинать обучение сети (листинг 7), для чего в случае использования библиотеки Keras достаточно вызвать метод fit сети — он пытается адаптировать (fit) модель под обучающие данные.

Листинг 7 - Обучение сети

```
model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```

В процессе обучения отображаются четыре величины: потери сети на обучающих данных и точность сети на обучающих данных, а также потери и точность на данных, не участвовавших в обучении.

Выполнение работы

По теоретическим сведениям была построена модель №1, она представлена на рис. 1. Для отображения модели использовалась функция plot_model(). Пример графика потерь и точности модели представлен на рис. 2.

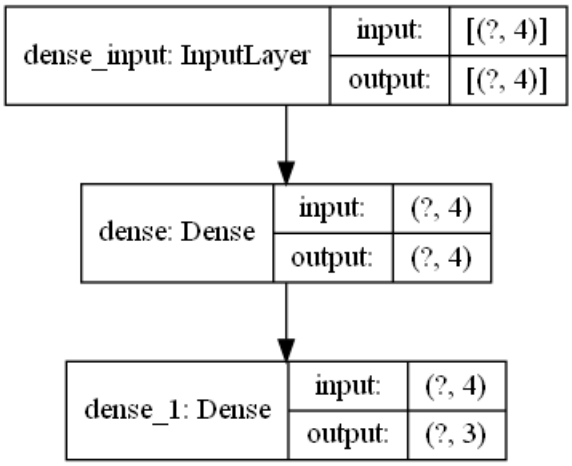

Рисунок 1: Схема модели №1

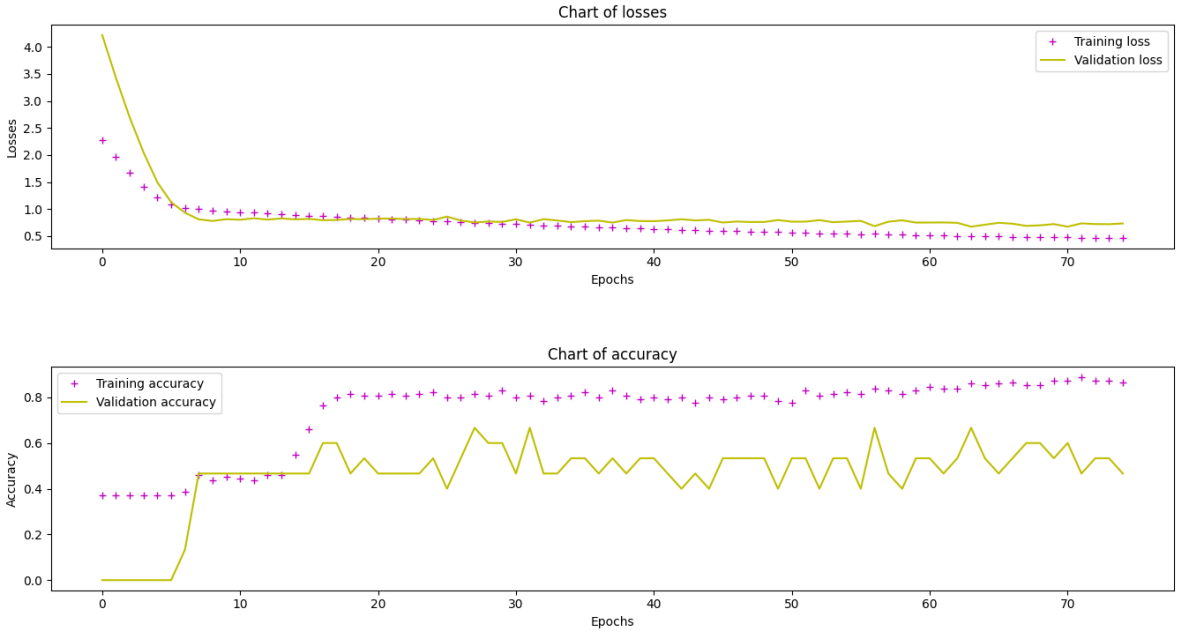

Рисунок 2: Результаты модели №1

Данная модель имеет довольно низкую точность, при этом результаты очень сильно меняются при каждом новом запуске.

Для следующей модели увеличивается количество слоев, но не изменяется количество нейронов на слое. Ее схема представлена на рис. 3, результаты — на рис. 4.

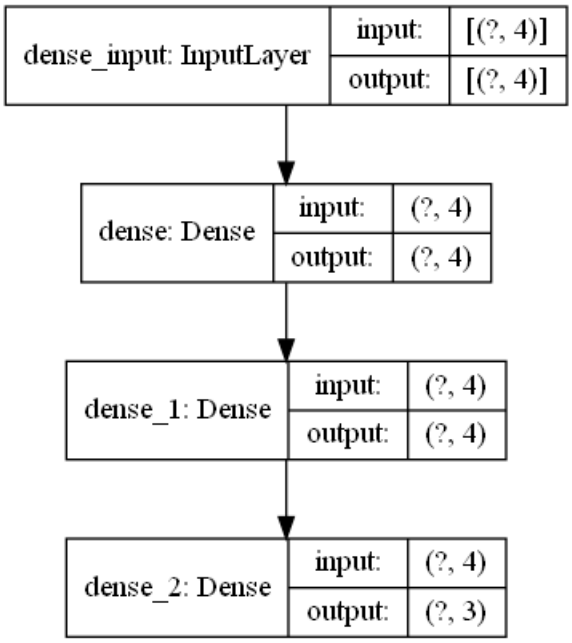

Рисунок 3: Схема модели №2

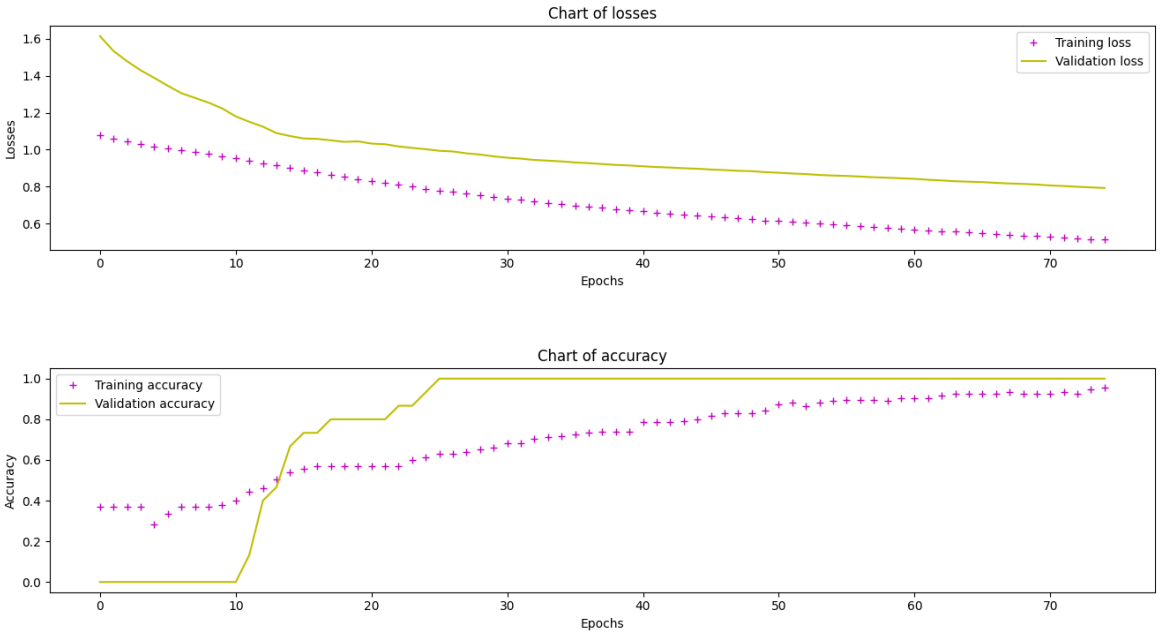

Рисунок 4: Результаты модели №2

Результат не сильно отличается первого варианта. Поведение все также разительно меняется с каждым запуском.

Для модели №3 увеличим количество нейронов на каждом слое до 16. Схема представлена на рис. 5, результаты — на рис. 6.

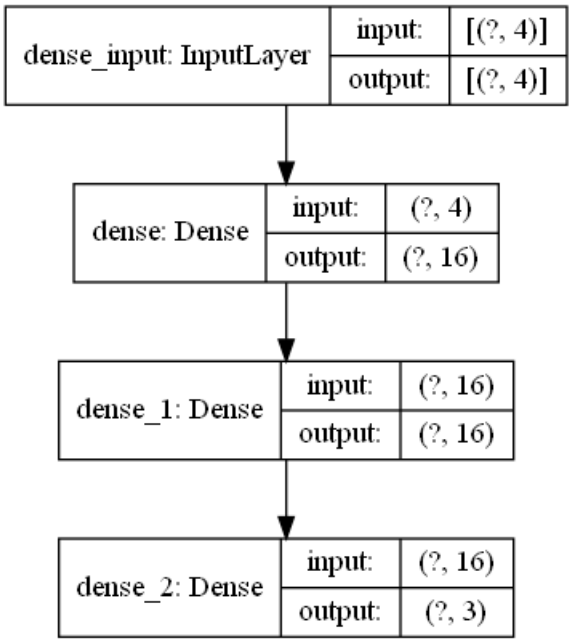

Рисунок 5: Схема модели №3

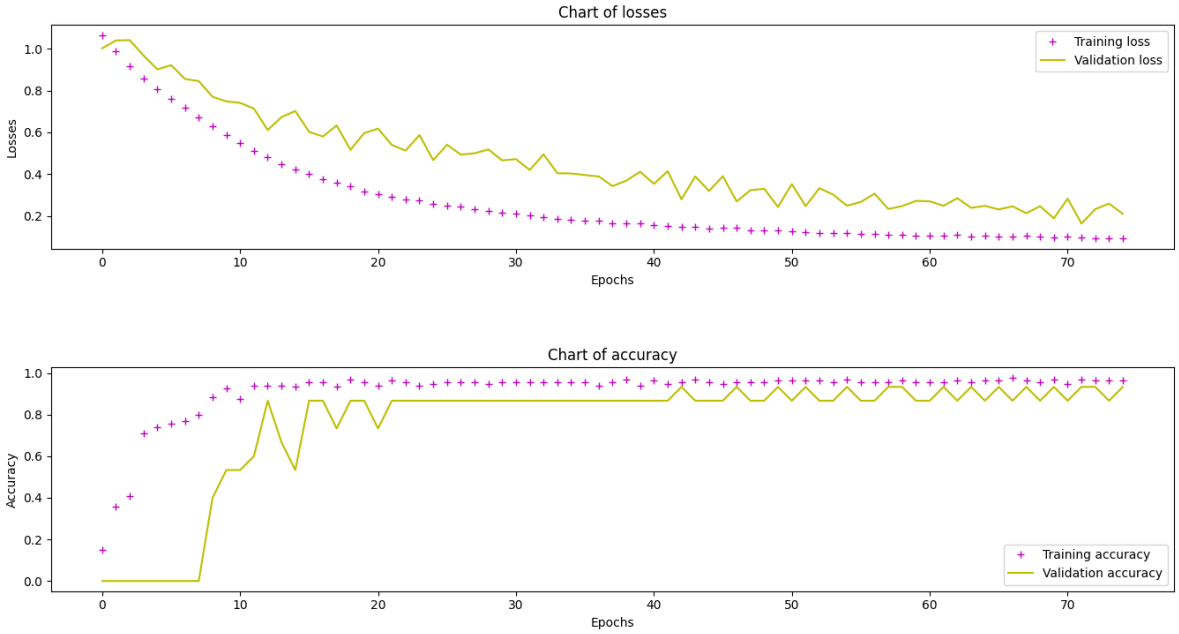

Рисунок 6: Результаты модели №3

По сравнению с другими вариантами эта модель показывает значительный прогресс. Во-первых с каждым новым запуском результаты

более стабильны, во-вторых модель дает довольно высокую точность — более 90%.

Для следующей модели увеличим количество нейронов на слоях еще больше — до 32. Схема модели №4 представлена на рис. 7, результаты — на рис. 8.

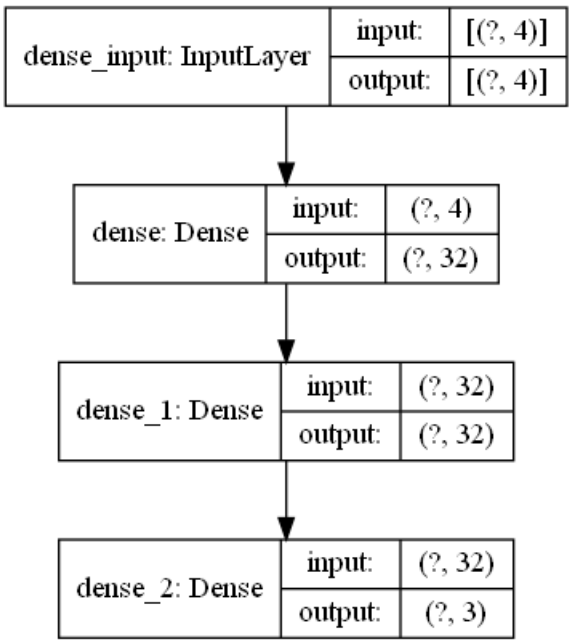

Рисунок 7: Схема модели №4

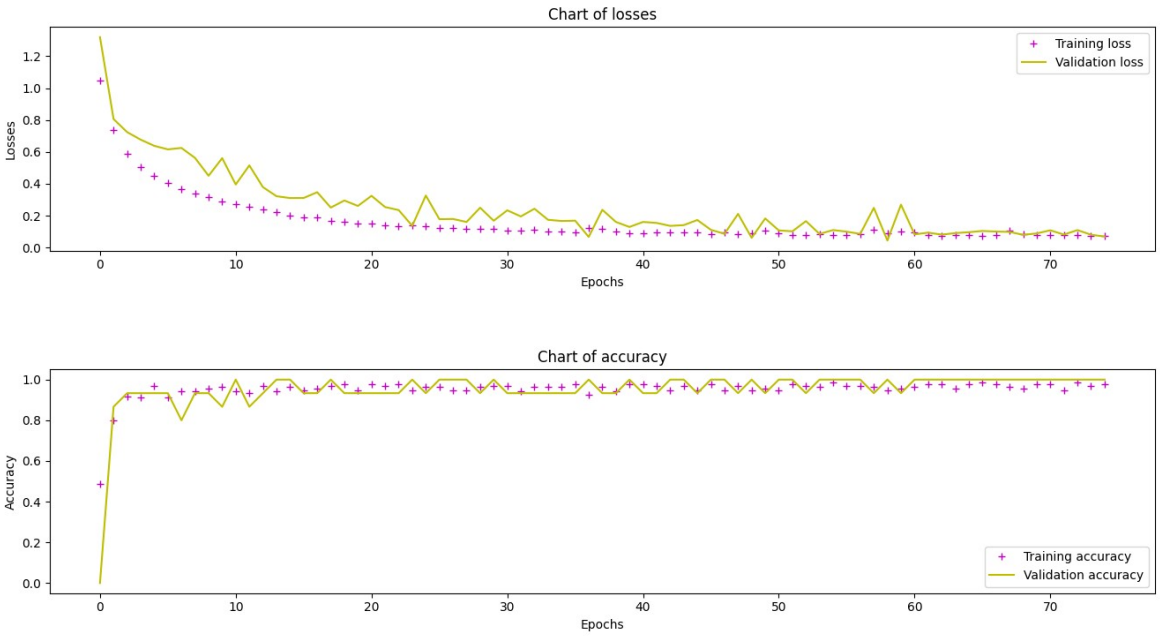

Рисунок 8: Результаты модели №4

Данная модель стабильна, ее точность высока и достигается достаточно быстро. Так же из результатов видно, что потери модели №4 меньше, чем у модели №3

Теперь, используя модель №4 изменим параметры обучения. Увеличим количество эпох до 100, параметр validation_slice до 0.25 (тем самым уменьшив количество тестовых образцов и увеличив количество образцов проверки), увеличим параметр batch_size до 15. Результаты приведены на рис. 9.

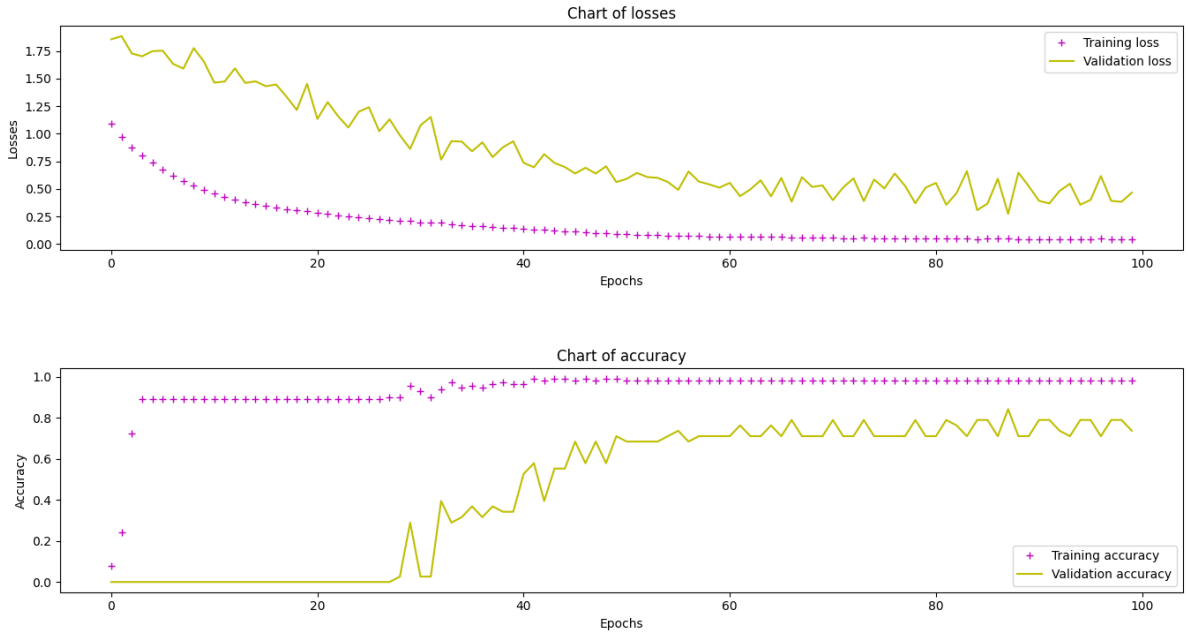

Рисунок 9: Результаты модели №5

Можно заметить, что по сравнения с опытом модели №4, увеличилось количество потерь, а так же упала точность данных проверки.

Теперь установим параметр batch_size равным 5, остальные настройки останутся без изменения. Результат представлен на рис. 10.

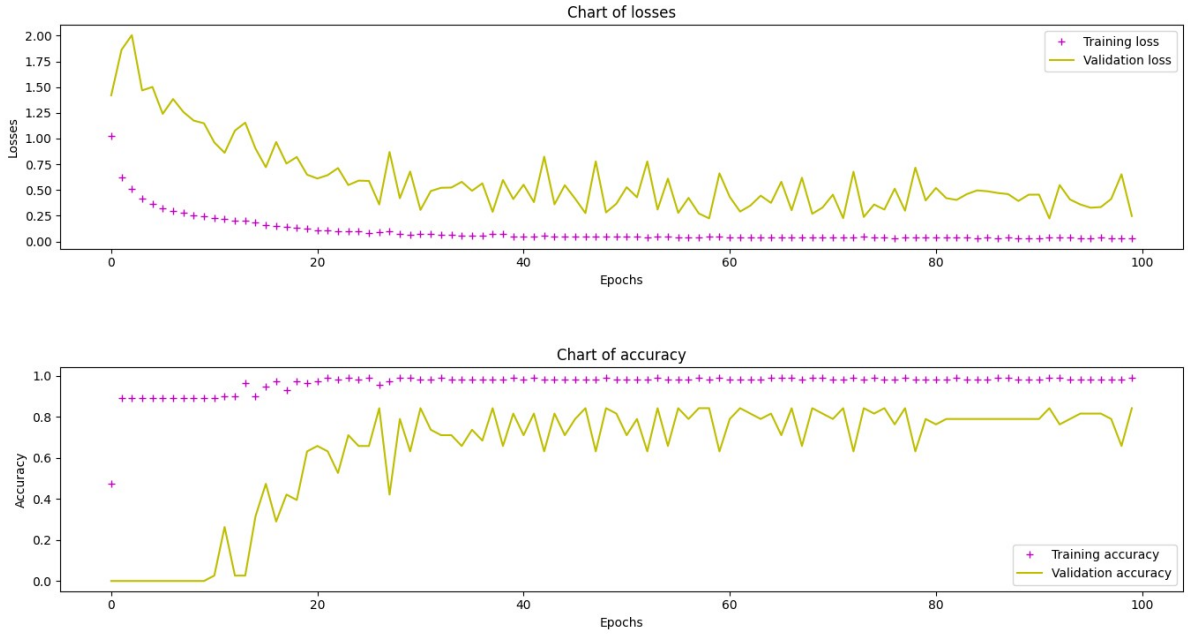

Рисунок 10: Результаты модели №6

Из результатов видно, что по сравнению с прошлым экспериментом, точность и потери не сильно изменились, но их значения стали достаточно часто «прыгать» и график стал более шумным.

Выводы

Была реализована искусственная нейронная сеть для классификации сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков. Были получены навыки в построение, настройке, тренировке и анализе искусственных нейронных сетей. Так же можно сделать вывод, что лучше всего себя показала модель №4 с начальными значениями функции fit().

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

```
import pandas
import numpy as np
from matplotlib import pyplot
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

from pathlib import Path

##Загрузка данных
path = Path("iris.csv")
dataframe = pandas.read_csv(path.absolute(), header = None)
dataset = dataframe.values
X = dataset[:, 0:4].astype(float)
Y = dataset[:,4]



##Создание категориального вектора
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_Y = to_categorical(encoded_Y)

##Создание модели
model = Sequential()
model.add(Dense(32, activation="relu", input_shape=(4,)))
model.add(Dense(32, activation="relu"))
model.add(Dense(3, activation="softmax"))

##Установка параметров обучения
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
##Обучение модели
data = model.fit(X, dummy_Y, epochs = 75, batch_size = 10, validation_split =
0.1, verbose=0)

##Печать модели
plot_model(model, to_file="model.png", show_shapes = True)

##Отрисовка графиков потерь и точности
loss = data.history['loss']
val_loss = data.history['val_loss']
accuracy = data.history['accuracy']
val_accuracy = data.history['val_accuracy']
epochs = range(0, 75)
pyplot.figure()

##График потерь
pyplot.subplot(2, 1, 1)
pyplot.plot(epochs, loss, 'm+', label='Training loss')
pyplot.plot(epochs, val_loss, 'y-', label='Validation loss')
pyplot.title('Chart of losses')
pyplot.xlabel('Epochs')
pyplot.ylabel('Losses')
pyplot.legend()

##График точности
pyplot.subplot(2, 1, 2)
pyplot.plot(epochs, accuracy, 'm+', label='Training accuracy')
pyplot.plot(epochs, val_accuracy, 'y-', label='Validation accuracy')
pyplot.title('Chart of accuracy')
pyplot.xlabel('Epochs')
pyplot.ylabel('Accuracy')
pyplot.legend()

pyplot.tight_layout()
pyplot.show()
```