МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искуственные нейронные сети»

Тема: Многоклассовая классификация цветов

Студент гр. 8382 _____ Щемель Д.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

Ход работы

Задание константных параметров нейронной сети: Оптимизатор - “adam”

Функция потерь - “categorial_crossentropy” - для небинарных категорий

Метрика - “accuracy”

Размер пакета - 10 (размер всей выборки - 150)

Разделение на тестовые/обучающие данные - 0.1

Количество эпох во время подборки модели - 50. После нахождение наиболее точной модели - будет сокращено до оптимального.

Обучение с начальными параметрами

```
model.add(layers.Dense(4, activation="relu"))
```

Результаты:

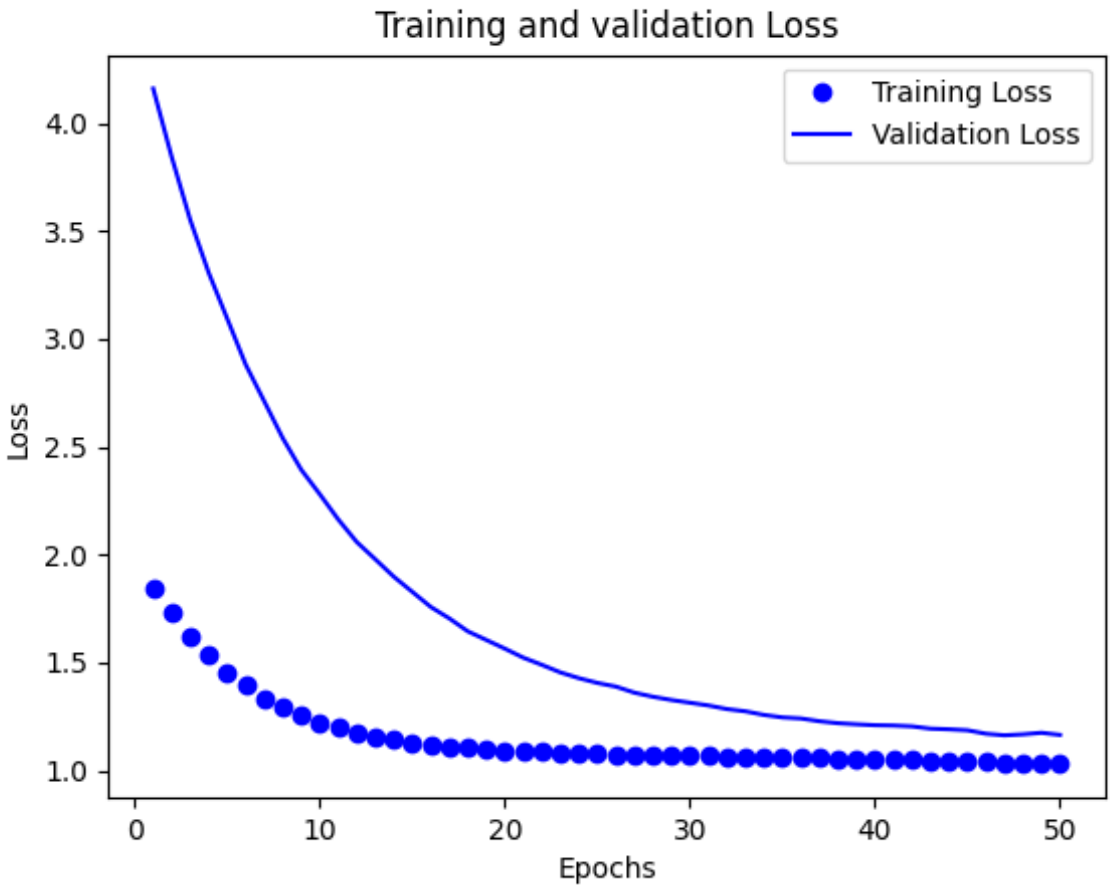

Рис. 1: 1_loss

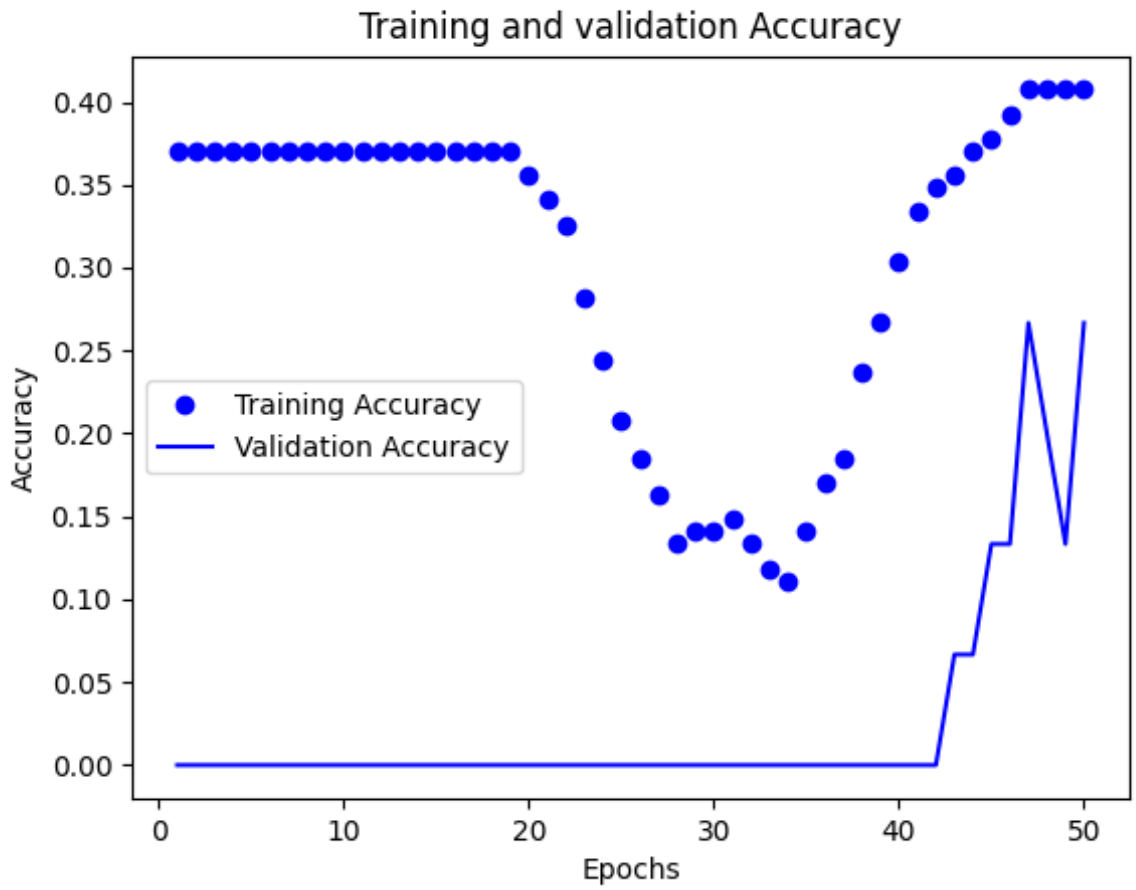

Рис. 2: 1_acc

Ошибки, точность: [1.0415226221084595, 0.40666666626930237]

Добавление количества нейронов

```
model.add(layers.Dense(100, activation="relu"))
```

Результаты:

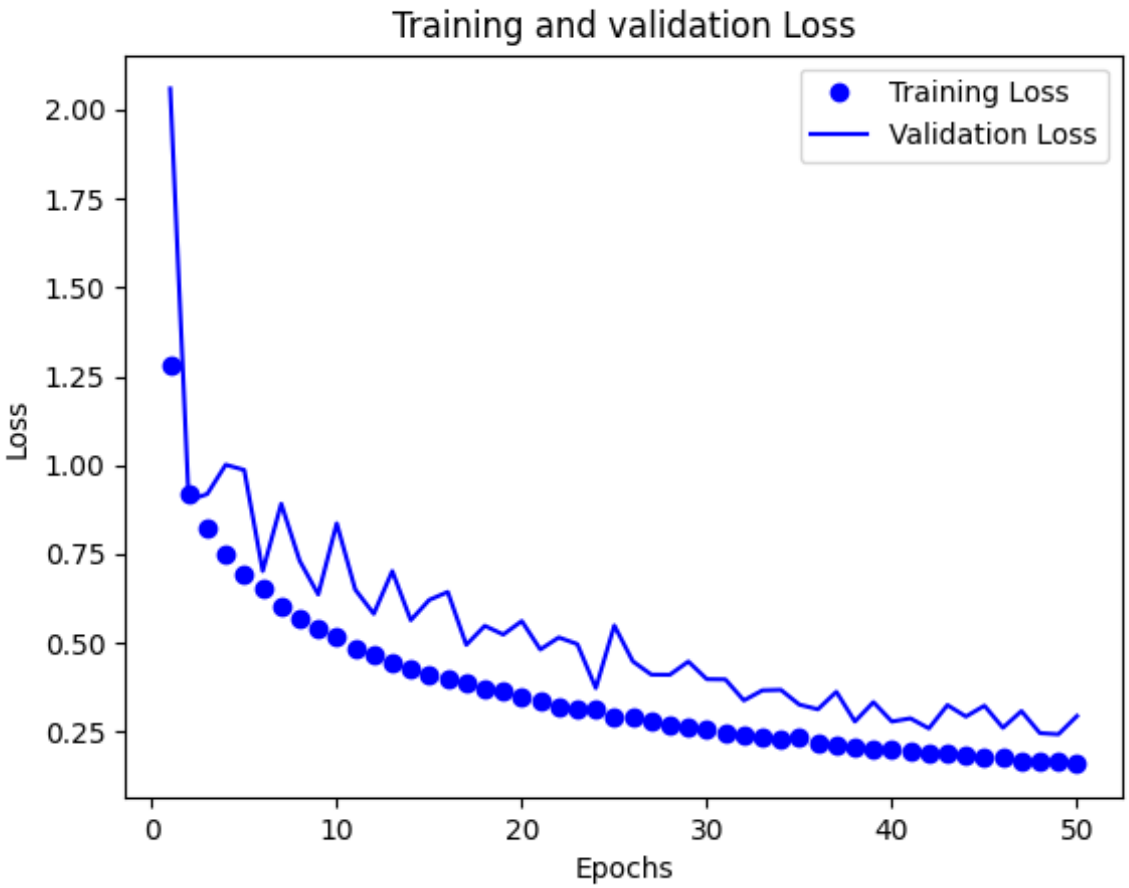

Рис. 3: 2_loss

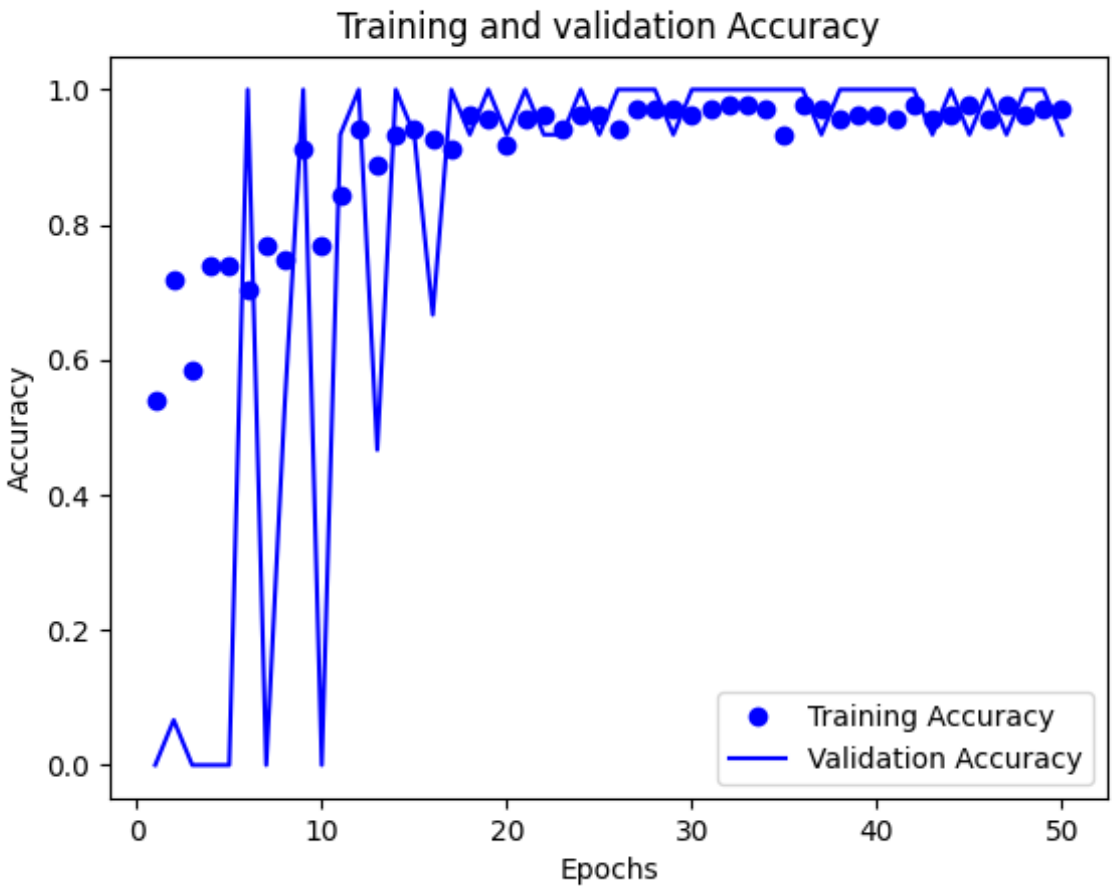

Рис. 4: 2_acc

Ошибки, точность: [0.1717270463705063, 0.9666666388511658]

Добавление количества слоёв

```
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dense(100, activation="relu"))
```

Результаты:

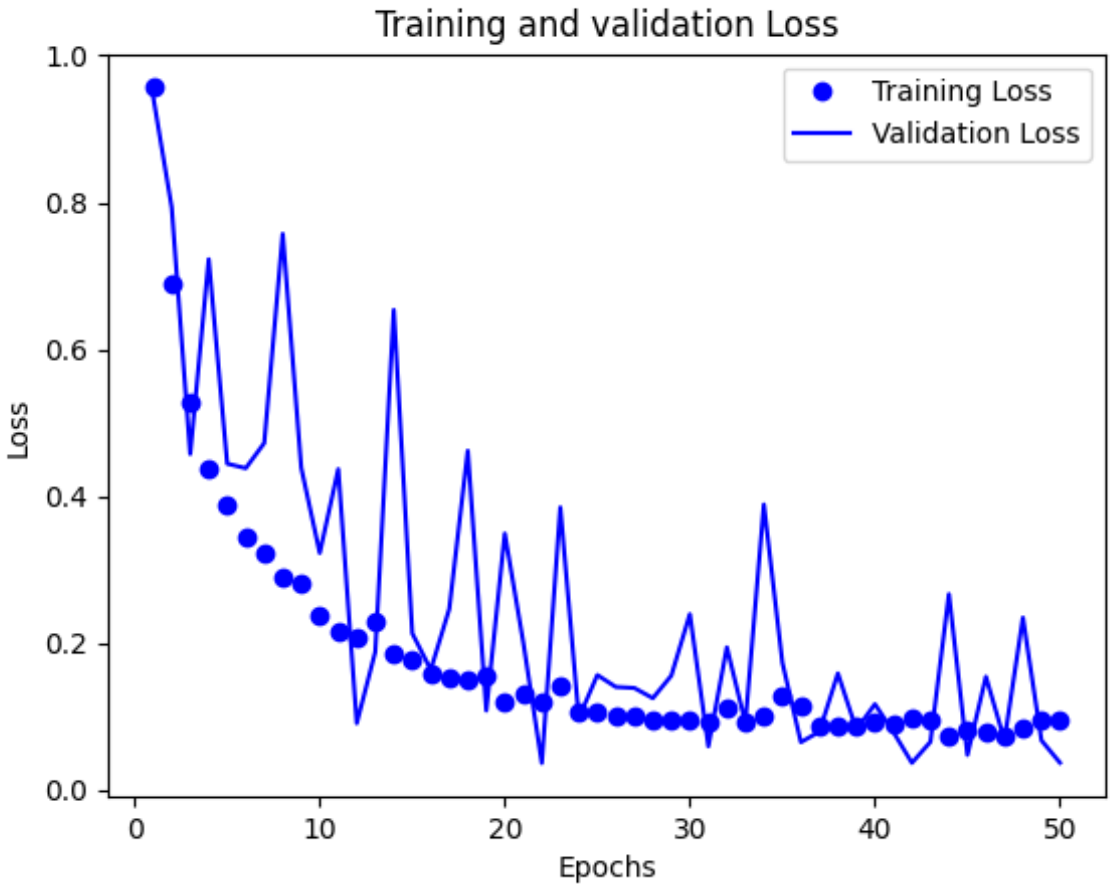
Training and validation Loss
Training Loss
Validation Loss
Loss
Epochs
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50

Рис. 5: 3_loss

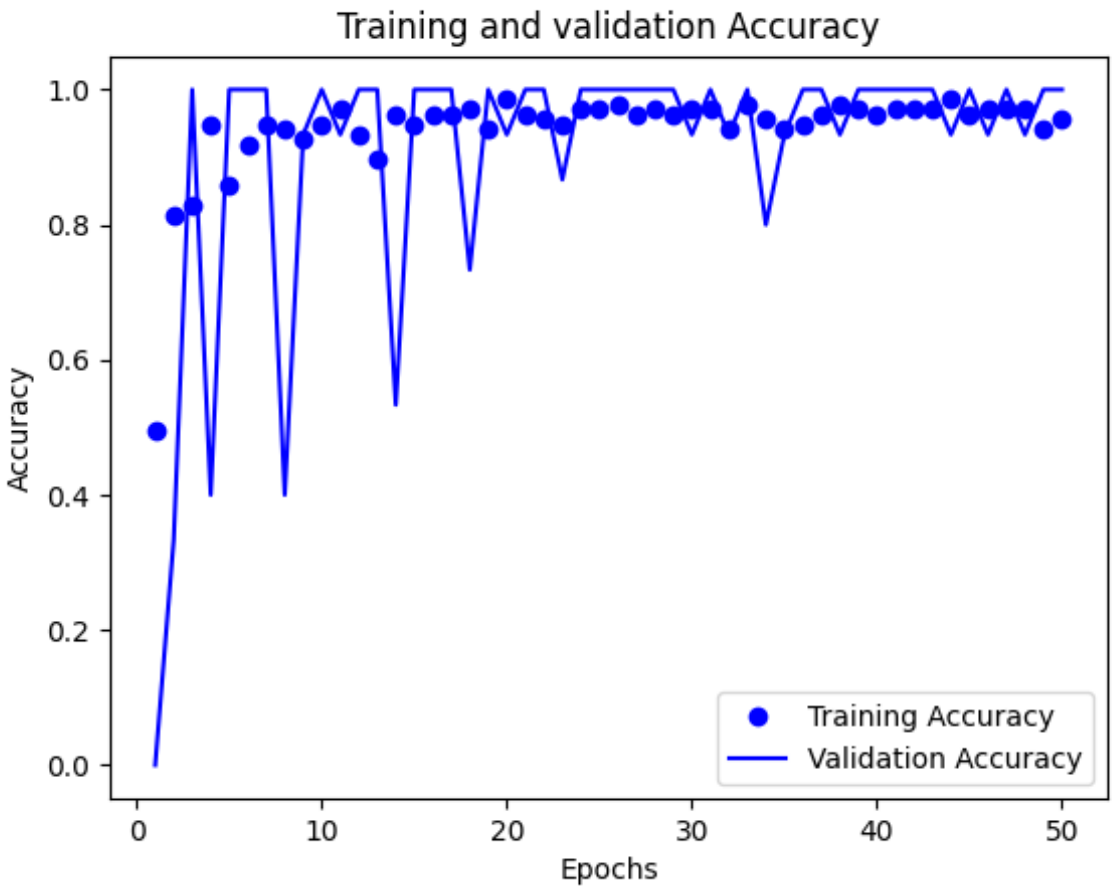

Рис. 6: 3_acc

Ошибки, точность: [0.07677149772644043, 0.9733333587646484]

Увеличение количества эпох Увеличим колчество эпох до 90.

```
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dense(100, activation="relu"))
...
epochs = 90
model.fit(data, labels, batch_size=batch_size, epochs=epochs, validation_split=vali
```

Результаты:

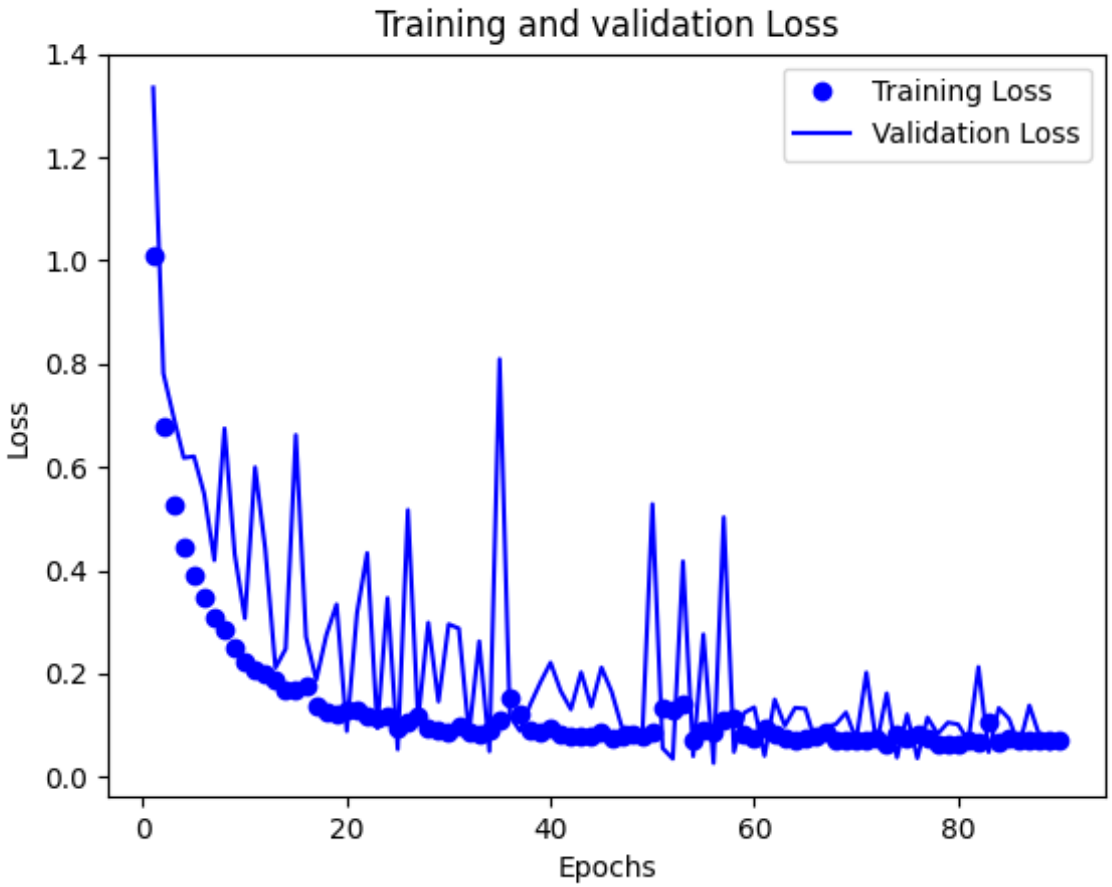
Training and validation Loss
Training Loss
Validation Loss
Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80

Рис. 7: 4_loss

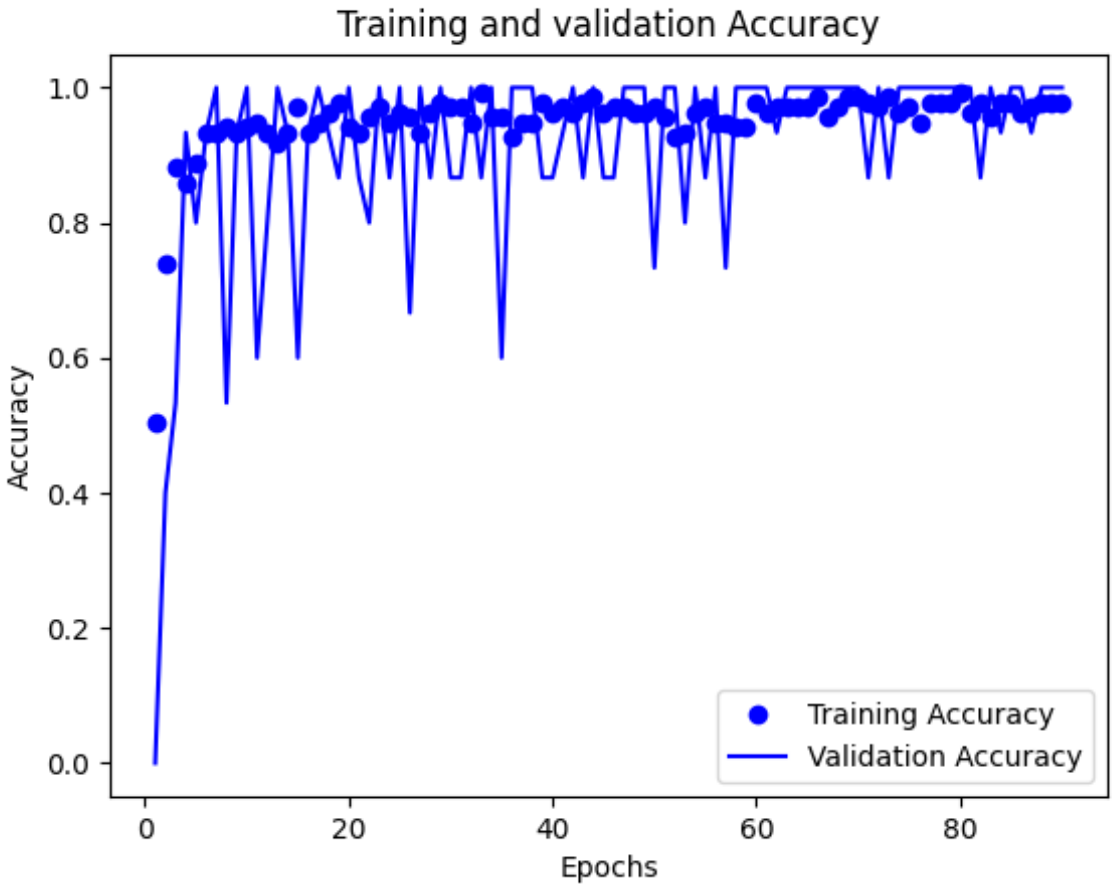

Рис. 8: 4_acc

Ошибки, точность: [0.062131308019161224, 0.9800000190734863]

На графике можно наблюдать локальные падения точности и увеличения ошибок. Это связано с переобучением (overfitting) модели. Поэтому увеличение эпох не всегда оправдано.

Выбор модели Для решения задачи можно оставновиться на последней модели, т.к. она даёт точность 0.98.

Вывод

В ходе выполнения лабораторной работы были получены практические навыки построения нейронных сетей для многоклассовой классификации.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

```
from typing import List, Iterator, Tuple
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
from tensorflow.python.keras import models
from tensorflow.python.keras.callbacks import History


def load_data(filename: str) -> Tuple[List[int], List[int]]:
    dataframe = pandas.read_csv(filename, header=None)
    dataset = dataframe.values
    data = dataset[:, :4].astype(float)
    string_labels = dataset[:, 4]
    encoder = LabelEncoder()
    encoder.fit(string_labels)
    encoded_labels = encoder.transform(string_labels)
    labels = to_categorical(encoded_labels)
    return data, labels


def create_model() -> models.Model:
    model = models.Sequential()

    model.add(layers.Dense(100, activation="relu"))
    model.add(layers.Dense(100, activation="relu"))
    model.add(layers.Dense(3, activation="softmax"))

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

```
def train_model(model: models.Model, data: np.array, labels: np.array, batch_size:
                validation_split: float) -> History:
    return model.fit(data, labels, batch_size=batch_size, epochs=epochs, validation


def draw_plot_for(data_type: str, epochs: Iterator[int], train_data_value: List[int
    plt.plot(epochs, train_data_value, "bo", label=f"Training {data_type}")
    plt.plot(epochs, test_data_value, "b", label=f"Validation {data_type}")
    plt.title(f"Training and validation {data_type}")
    plt.xlabel("Epochs")
    plt.ylabel(f"{data_type}")
    plt.legend()
    plt.show()


def main():
    data, labels = load_data("iris.csv")

    model = create_model()
    history = train_model(model, data, labels, 10, 90, 0.1).history

    loss = history["loss"]
    val_loss = history["val_loss"]
    acc = history["accuracy"]
    val_acc = history["val_accuracy"]
    epochs = range(1, len(loss) + 1)

    draw_plot_for("Loss", epochs, loss, val_loss)

    plt.clf()
    draw_plot_for("Accuracy", epochs, acc, val_acc)

    results = model.evaluate(data, labels)
    print(results)
```

```
if __name__ == "__main__":
    main()
```