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. 8382 _____ Нечепуренко Н.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.

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.

Проведем сравнение нескольких архитектур ИНС, варьируя кол-во слоев и кол-во нейронов на слоях. Во всех представленных архитектурах первый слой будет состоять из 4 нейронов, так как каждое наблюдение описывается 4 признаками, в качестве функции активации выберем relu, так как отрицательных длин не бывает. Последний слой будет состоять из 3 нейронов с функцией активации softmax, который представит вероятности отнесения цветка к тому или иному классу.

Архитектуры моделей для сравнения опишем в файле models.csv.

```
epochs:100,4:relu;3:softmax
epochs:100,4:relu;6:relu;3:softmax
```

Для сравнения добавим один скрытый слой с 6 нейронами и функцией relu.

Сравним две представленные архитектуры при обучении на 100 эпохах. Так как веса ребер генерируются случайным образом, случается так, что функция потерь не минимизируется, либо выбирается локальный минимум с низким показателем требуемых метрик. На рисунках 1 и 2 приведены графики функции потерь и точности для заданных архитектур (лучшие результаты по трём запускам).

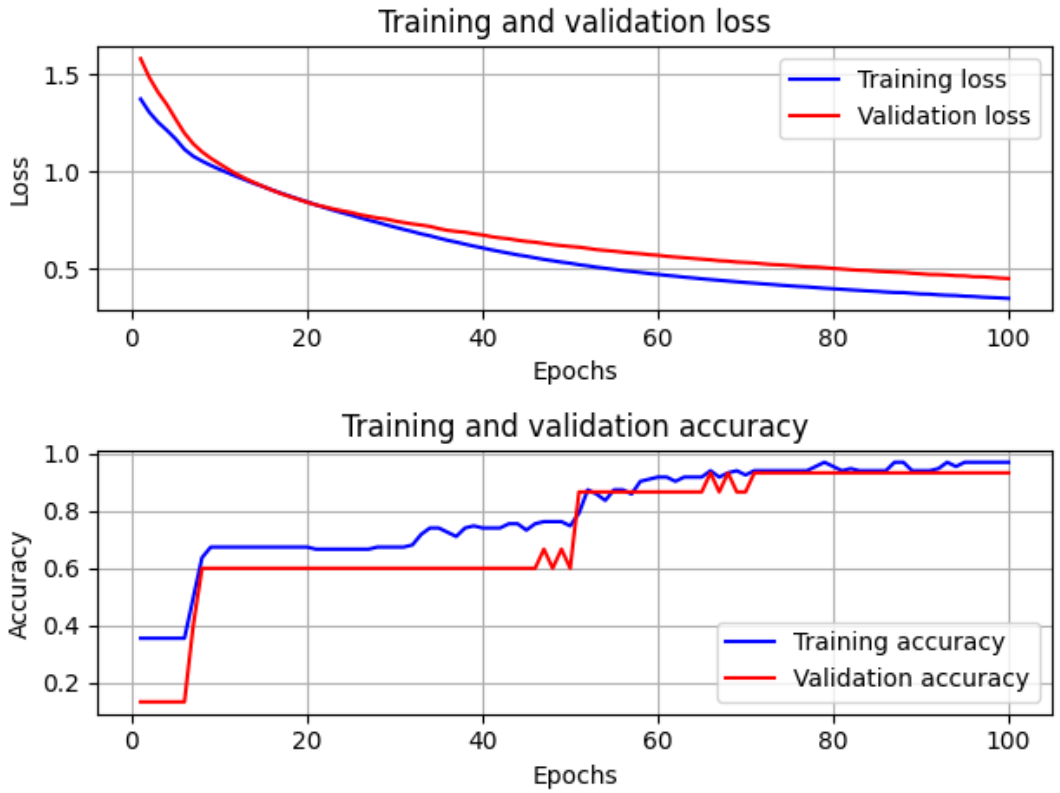

Рисунок 1 – Графики функции потерь и точности для архитектуры с двумя слоями

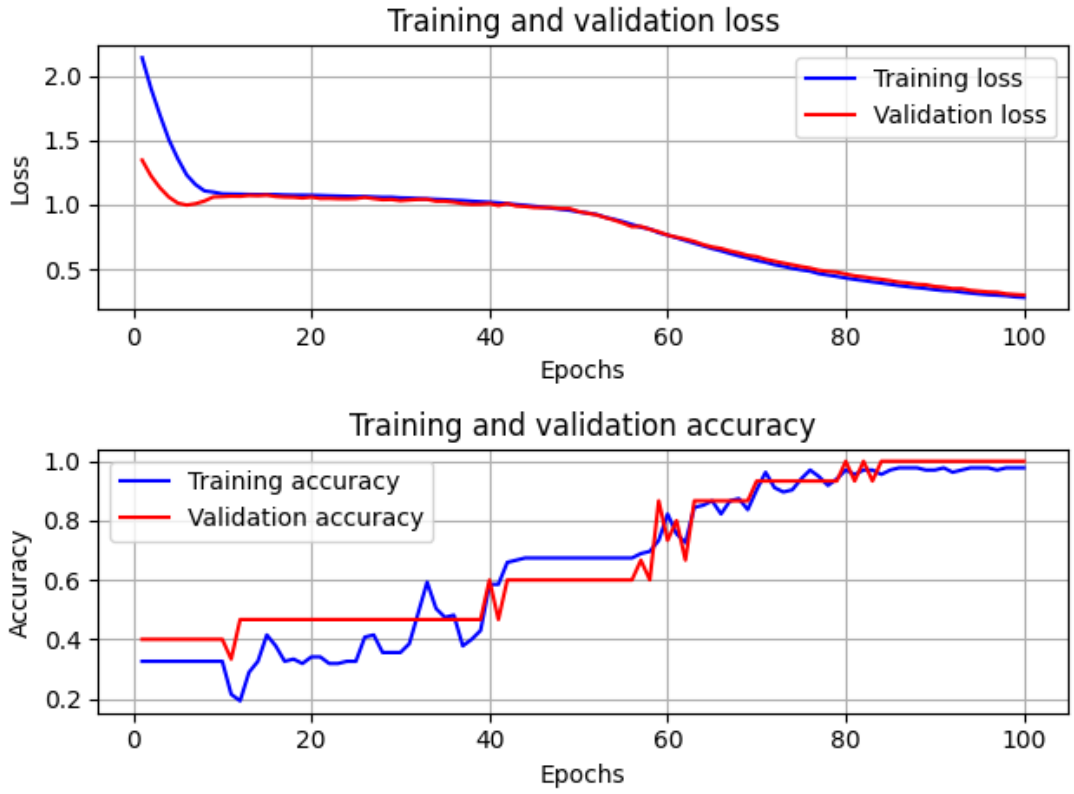

Рисунок 2 – Графики функции потерь и точности для архитектуры с тремя слоями

Первая архитектура показала точность на тренировочных данных порядка 0.95, на валидационных – порядка 0.93. Вторая архитектура с дополнительным слоем показала лучшие результаты, после 80 эпохи точность на тестовом множестве оказалась порядка 0.98, на валидационном 1. Показатели требуемой метрики улучшились, добавление слоя позволило более точно отражать влияние признаков. В целом, результаты архитектуры №2 достойные, но попробуем увеличить избыточность, добавив еще слои. Рассмотрим сеть:

```
epochs:100,4:relu;6:relu;10:relu;10:relu;4:relu;3:softmax
```

В результате одного из запусков алгоритм не смог сойтись (см. рис. 3).

Рисунок 3 – Алгоритм обучение сети не сошелся

Проведем несколько запусков и выберем лучший результат.

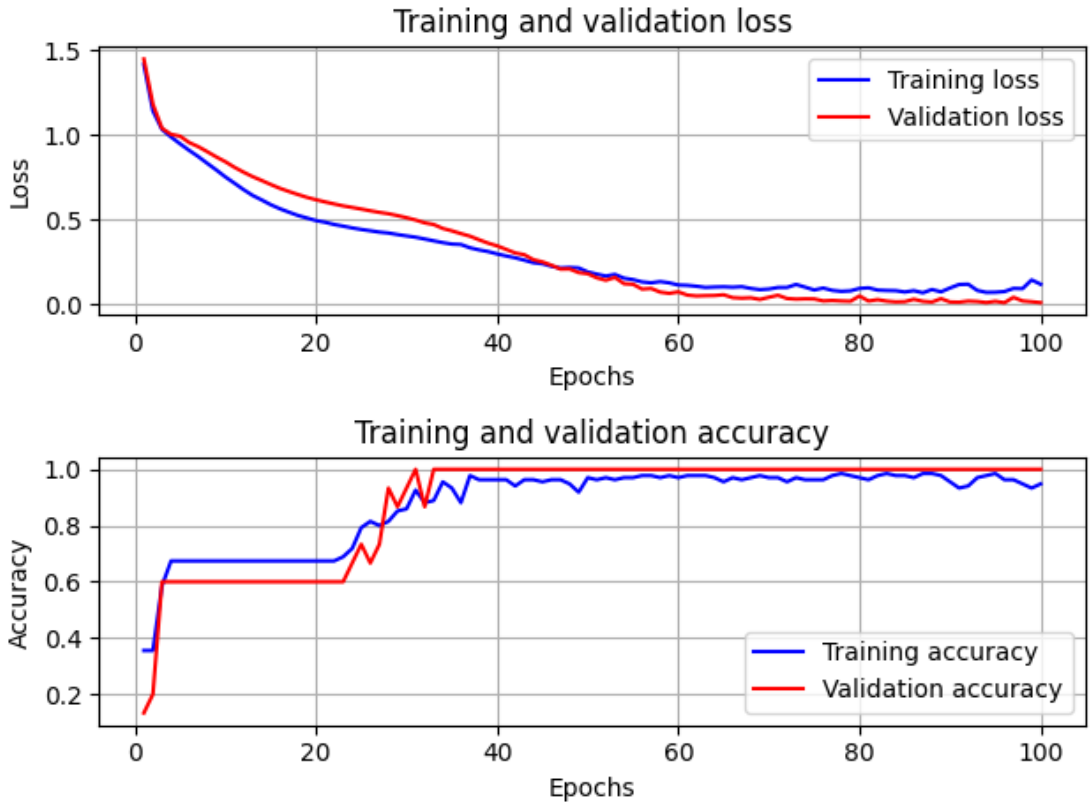

Рисунок 4 – Результат сети с 4 скрытыми слоями

Результаты сети схожи с результатами сети № 2, но достигаются за гораздо меньшее число эпох. Модель №3 гораздо сложнее первых двух, и так как заданием не установлен достаточный уровень точности модели, поэтому остановимся на моделях с одним скрытым слоем.

Исследуем влияние параметров обучения (функции fit) на результаты модели. Будем исследовать следующие конфигурации:

```
epochs:100;validation_split:0.2,4:relu;6:relu;3:softmax
epochs:100;validation_split:0.2;batch_size:5,4:relu;6:relu;3:softmax
epochs:100;validation_split:0.1;batch_size:5,4:relu;6:relu;3:softmax
epochs:200;validation_split:0.1;batch_size:10,4:relu;6:relu;3:softmax
```

При увеличении размера валидационного множества до 20% модель стала обучаться быстрее. Лучший результат из трех запусков представлен на рисунке 5.

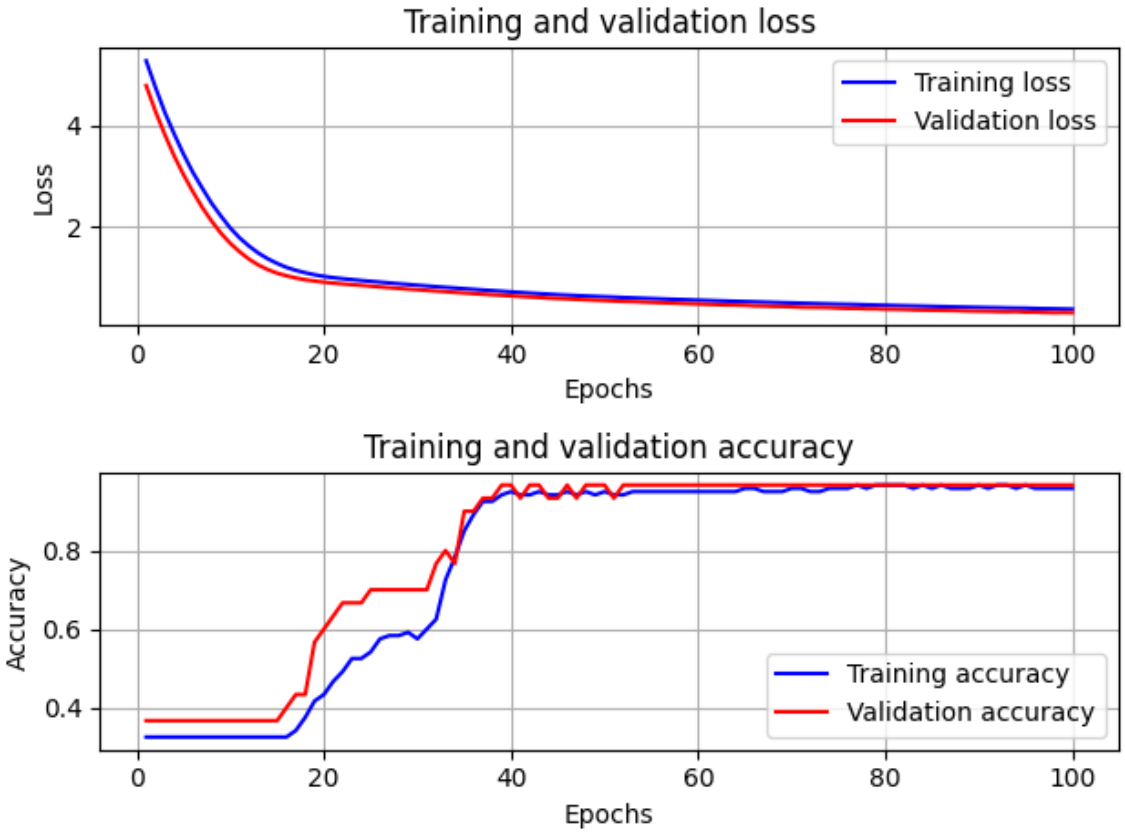

Рисунок 5 – Результаты ИНС при увеличенном валидационном множестве

Изменим размер батча с 10 до 5, при том же validation_split. Результаты приведены на рисунке 6 (лучший из трех запусков).

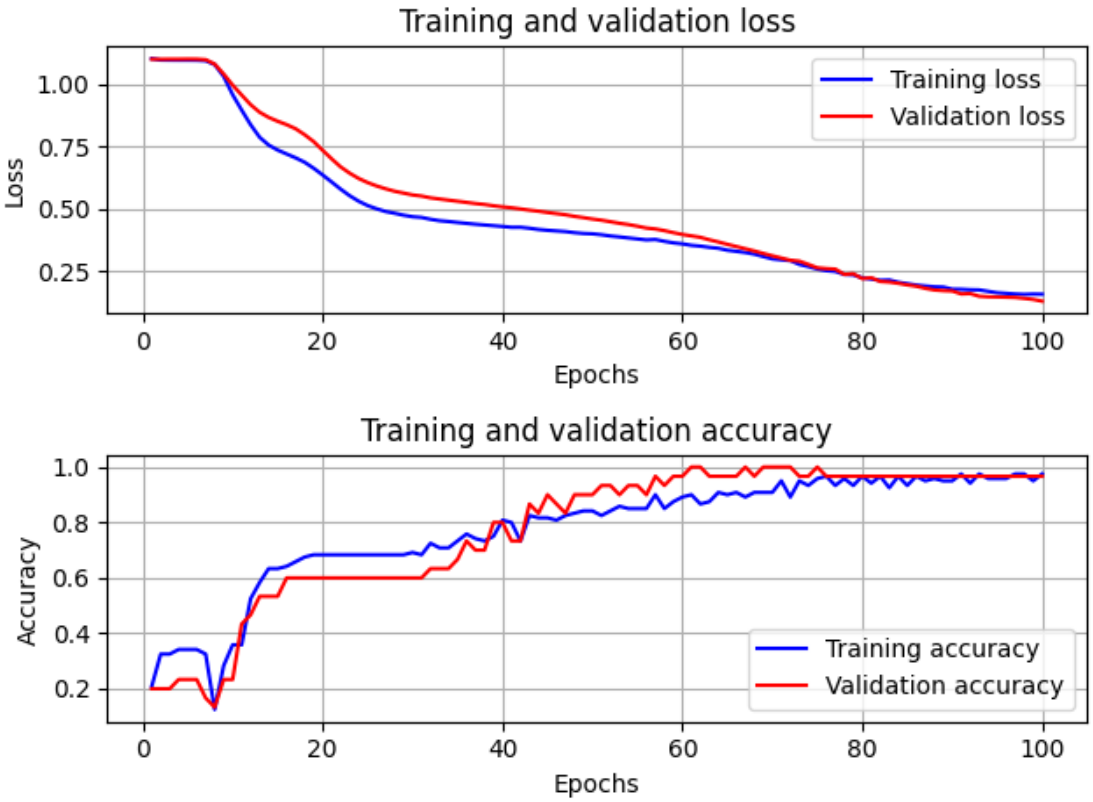

Рисунок 6 – Результаты ИНС при уменьшении размера батча

Вернем параметры validation_split и batch_size к изначальным значениям 0.1, и 10 соответственно. Увеличим число эпох до двухсот. Результат запуска приведен на рисунке 7.

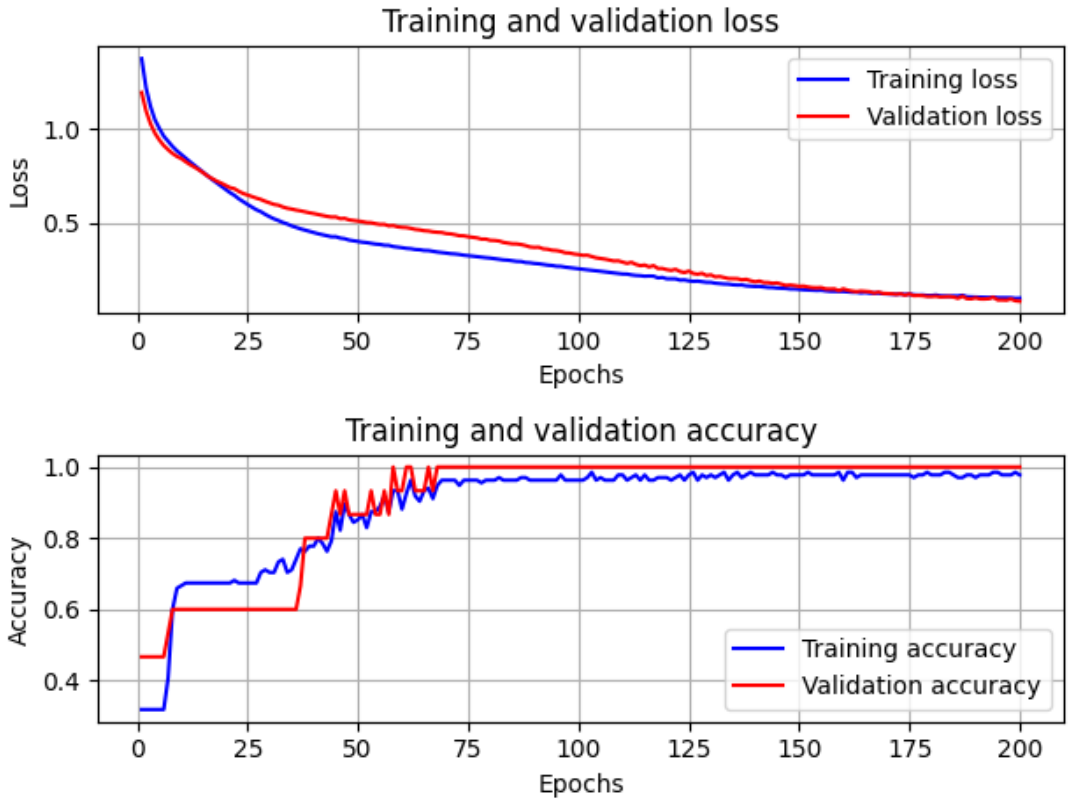

Рисунок 7 – Результаты ИНС при увеличении числа эпох до 200

На рисунке 7 видно, что увеличение числа эпох для заданной модели не дает никаких преимуществ, точность на валидационном множестве держится стационарно на 1, а на тренировочном колеблется в незначительных пределах. Для данной архитектуры достаточно 75 эпох (из рисунка).

При выборе наилучшей модели стоит учитывать множество факторов, например, скорость обучения, размер сети, значение целевых метрик (точность в данном случае), а также минимально достаточный уровень значения этой самой метрики и другие. Исходя из того, что количество наблюдений довольно мало, из-за чего трудно сделать объективные выводы, можно принять, что модель с одним скрытым слоем достаточно успешно решает заданную задачу классификации.

Выводы.

В результате выполнения данной лабораторной работы были получены навыки создания ИНС для решения задачи многоклассовой классификации. Были исследованы различные архитектуры сетей, а также влияние параметров

обучения на результаты классификации. Была выбрана оптимальная модель классификации цветов на три вида по четырем признакам.

ПРИЛОЖЕНИЕ А.

Исходный код программы. Файл main.py.

```
import pandas as pd
import numpy as np
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# %matplotlib inline
from lb1.models import ModelBuilder, get_models_to_compare


def read_data():
    dataframe = pd.read_csv("sample_data/iris.csv", header=None)
    dataset = dataframe.values
    np.random.seed(42)
    np.random.shuffle(dataset)
    x = dataset[:, 0:4].astype(float)
    y = dataset[:, 4]
    return x, y


def encode_y(raw_y):
    encoder = LabelEncoder()
    encoder.fit(raw_y)
    encoded_y = encoder.transform(raw_y)
    y_mapping = dict(zip(encoder.classes_,
encoder.transform(encoder.classes_)))
    return to_categorical(encoded_y), y_mapping


def plot(epochs, train, validation, metrics):
    plt.plot(epochs, train, 'b', label=f'Training {metrics}')
    plt.plot(epochs, validation, 'r', label=f'Validation {metrics}')
    plt.title(f'Training and validation {metrics}')
    plt.xlabel('Epochs')
    plt.ylabel(metrics.capitalize())
    plt.grid(True)
    plt.legend()


def summary(h):
    print(f"Model #{h.idx + 1} has best val_accuracy
{max(h.history.get('val_accuracy'))}")
    # maybe next time


def plot_history(history):
    loss = history['loss']
    val_loss = history['val_loss']
    acc = history['accuracy']
    val_acc = history['val_accuracy']
    epochs = range(1, len(loss) + 1)
```

```
    plt.figure()
    plt.subplot(211)
    plot(epochs, loss, val_loss, "loss")
    plt.subplot(212)
    plot(epochs, acc, val_acc, "accuracy")
    plt.show()


X, Y = read_data()
Y, mapping = encode_y(Y)

history_list = []
for model in get_models_to_compare():
    model_cfg, model_layers = model
    model_builder = ModelBuilder()
    model_builder.set_params(**model_cfg)
    model = model_builder.set_layers(model_layers).build()
    history_list.append(model.fit(X, Y))

for idx, history_item in enumerate(history_list):
    history_item.idx = idx
    summary(history_item)
    plot_history(history_item.history)
```