МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №1

"Многоклассовая классификация цветов" по

дисциплине «Искусственные нейронные сети»

Студентка гр. 8383 _____ Аверина О.С

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.

Задание.

- 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.

1. Загрузка данных и создание ИНС.

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели (листинг 1).

Листинг 1 - 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 2 - Загрузка данных

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

Преобразуем выходные атрибуты из вектора в матрицу. Для этого необходимо использовать функцию to_categorical()(Листинг 3).

Листинг 3 - Переход от текстовых меток к категориальному вектору

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

Далее была создана простая модель с двумя слоями – первый состоит из 4 нейронов с функцией активации Relu (для обработки 4 входных параметров), второй – из 3 нейронов с функцией активации Softmax (каждый нейрон определяет класс цветка на выходе) (листинг 4).

Листинг 4 - Создание модели

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

Затем было выполнено обучение сети (листинг 5), для чего в случае использования библиотеки Keras нужно было вызвать метод fit сети — он попытается адаптировать (fit) модель под обучающие данные.

Листинг 5 - Обучение сети

```
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    history = model.fit(X, dummy_y, epochs=75, batch_size=10,
validation_split=0.5)
```

```
history_dict = history.history
```

Был реализован вывод 4-х графиков: ошибки во время обучения, точность во время обучения, ошибки на проверяемых данных, точность на проверяемых данных (листинг 6).

Листинг 5 – Вывод графиков

```
        loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

2. Анализ показателей различных ИНС

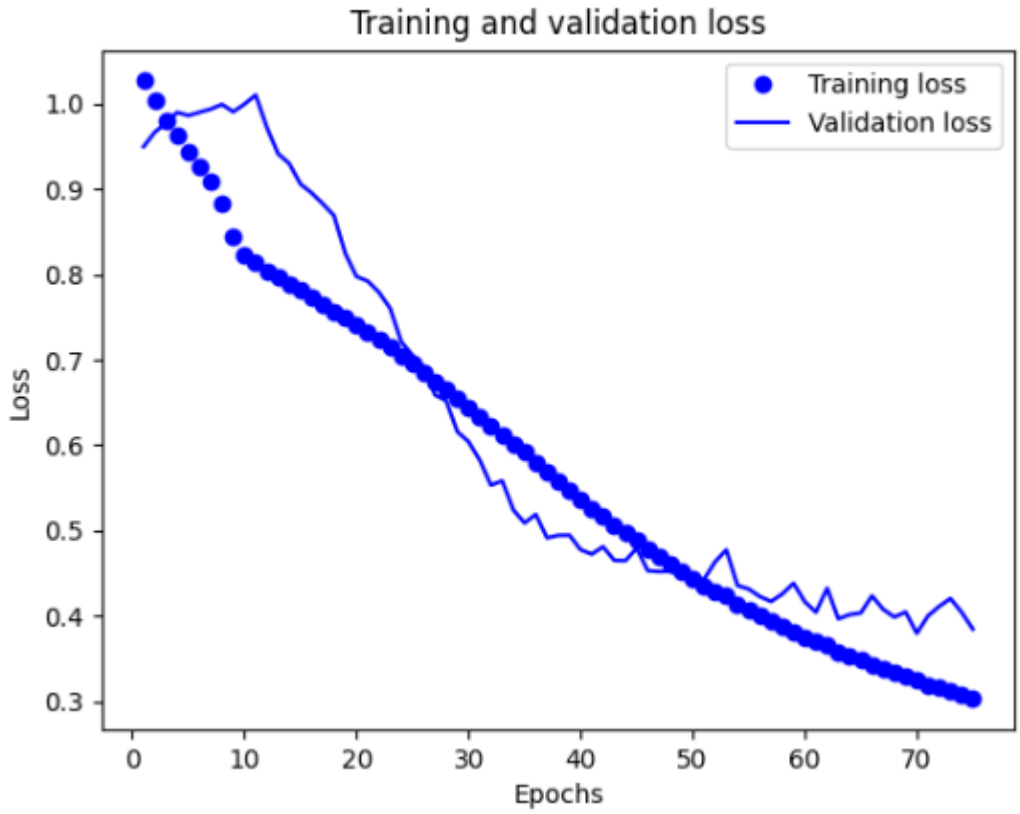
Модель 1.

Была построена модель, состоящая из двух слоев с 3 и 4 нейронами.

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

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
history_dict = history.history
```

Результаты тестирования модели №1 представлены на рис. 1.



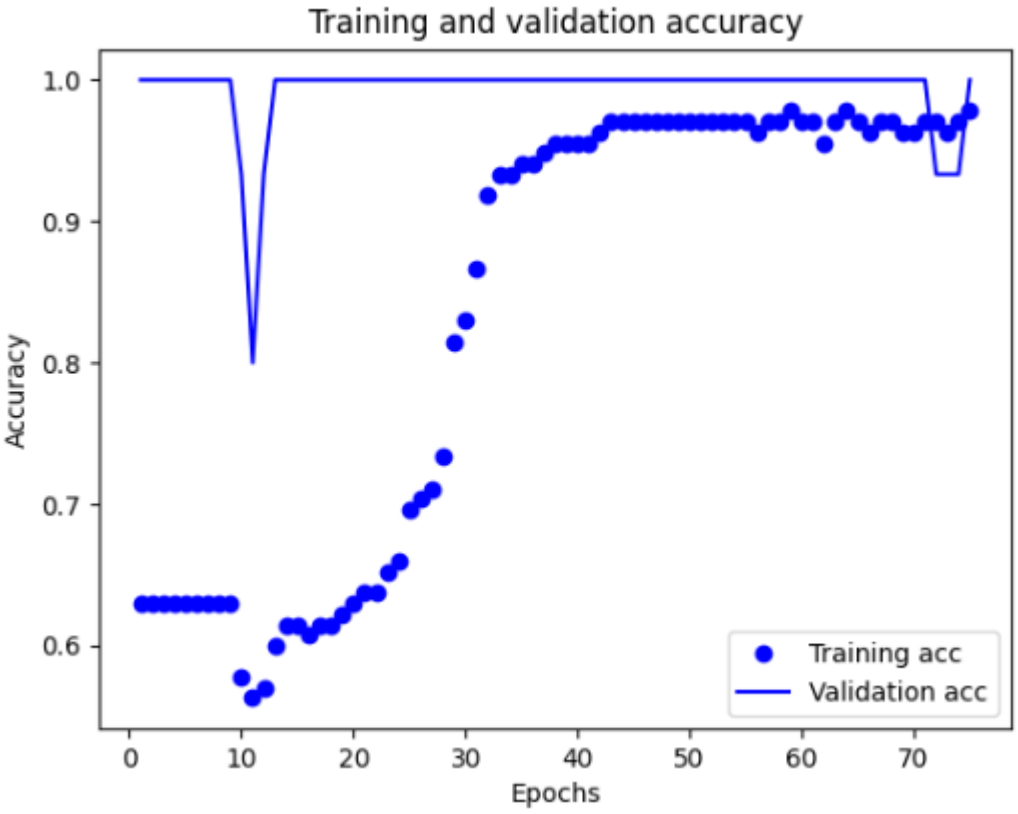


Рисунок 1 – Графики ошибок и точности ИНС #1

Из графика ошибок видно, что потери стремятся к нулю, а точность достаточно быстро возрастает.

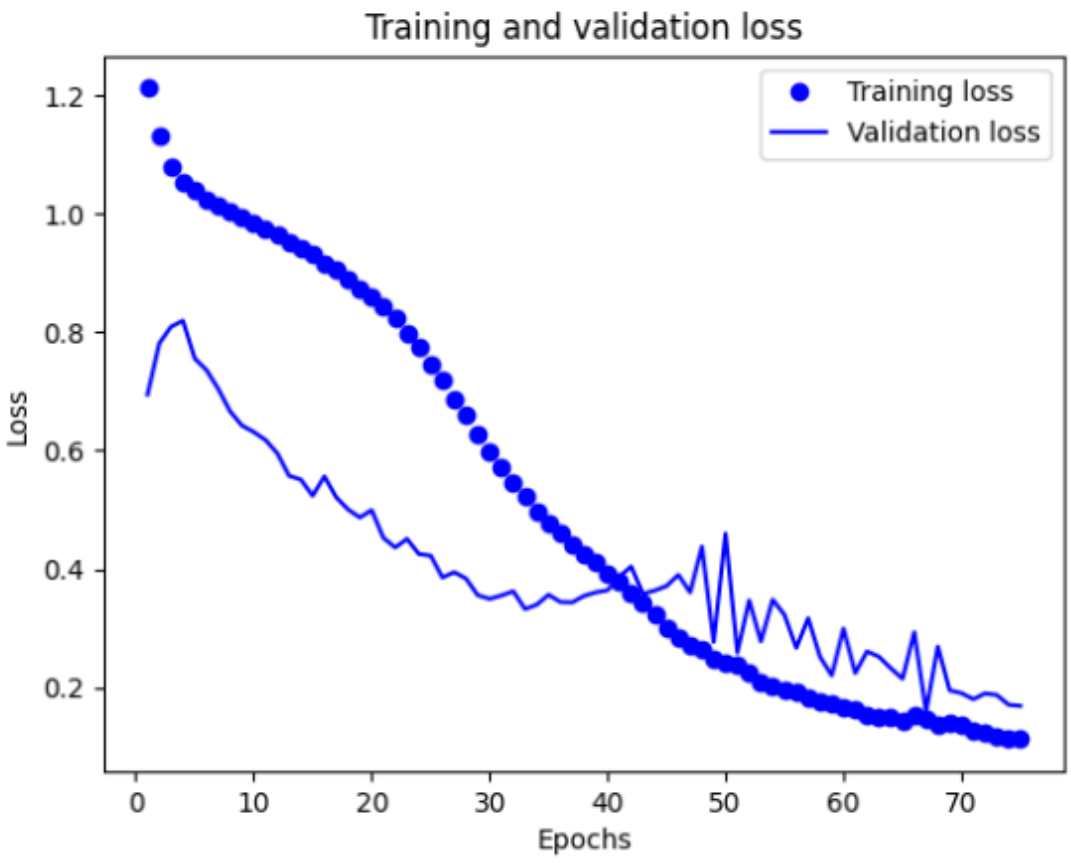
Результаты при каждом запуске нестабильны.

Модель 2.

Были добавлены два скрытых слоя с 10 и 5 нейронами с функцией активации Relu.

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(5, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```



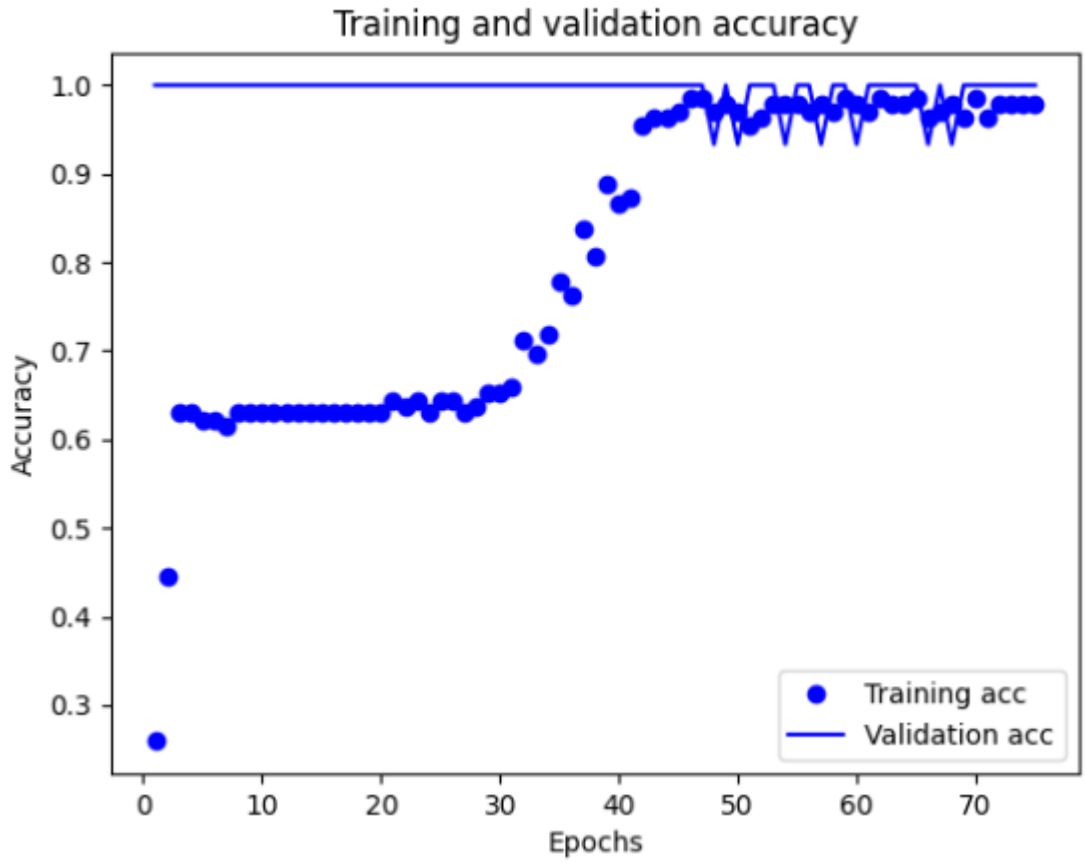


Рисунок 2 – Показатели ИНС #2

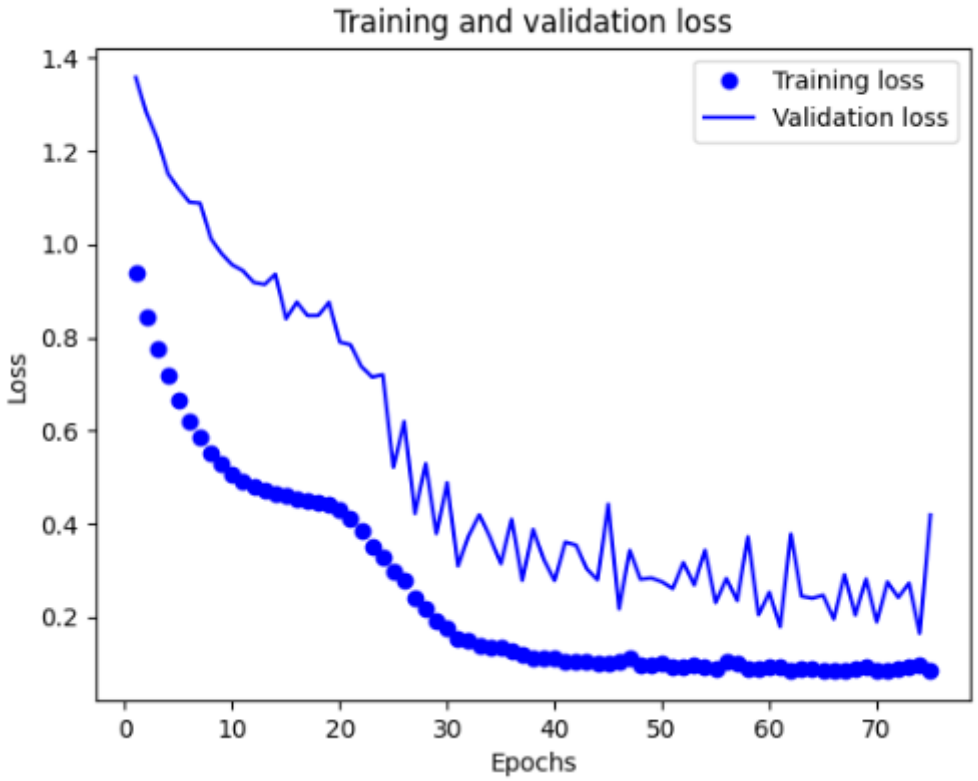
Из графиков видно, что при добавлении двух скрытых слоев с 10 и 5 нейронами с функцией активации Relu уменьшились показатели потерь и увеличилась точность на проверочных данных с 1 до 30 эпохах.

Модель 3.

Было увеличено количество нейронов в модели 2 с 10 на 20 и с 5 на 15.

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```



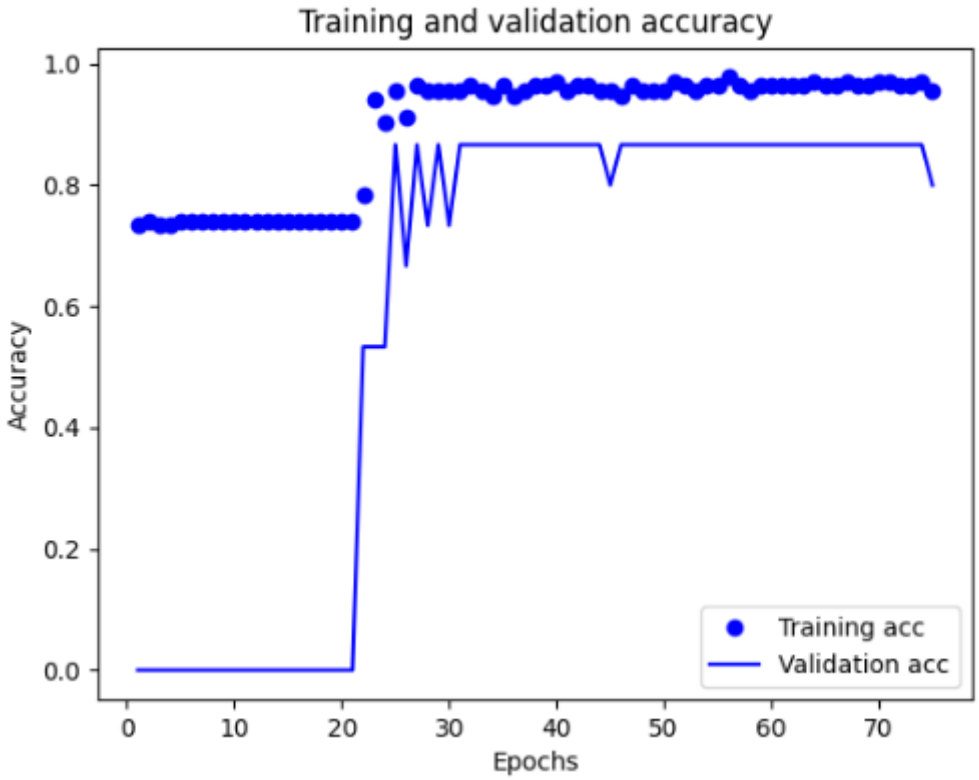


Рисунок 3 – Показатели ИНС №3

Из графиков видно, что увеличение числа нейронов в скрытом слое увеличило точность и уменьшило ошибки. На обучаемых данных точность быстро вырастает после 20 эпох до ~0,9, а на проверочных результат практически ~1.

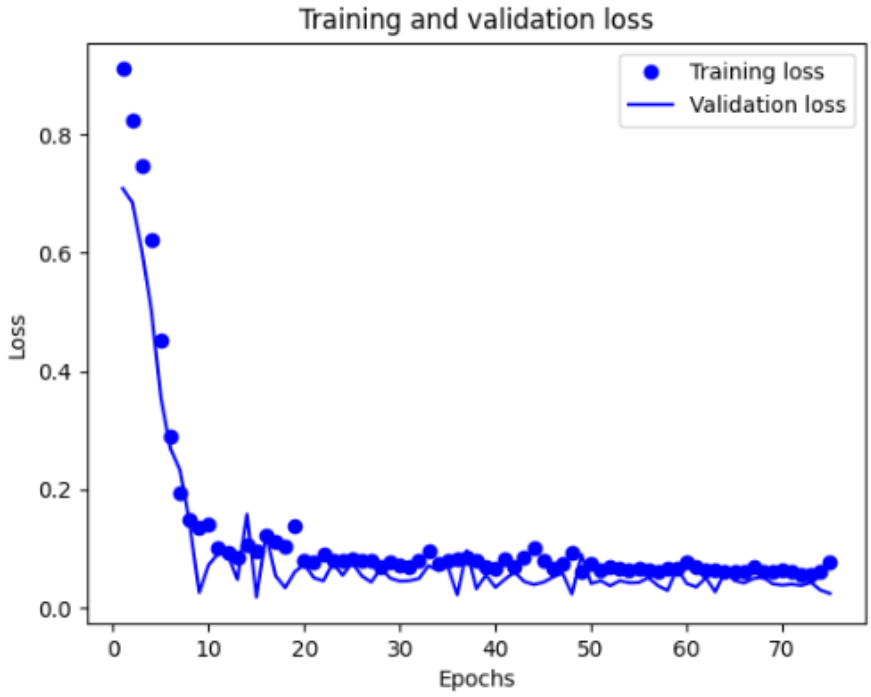
Значения при каждом запуске еще довольно отличаются друг от друга.

Модель 4.

Было увеличено количество нейронов в скрытых слоях с 20 на 200, с 15 на 100.

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.1)
```



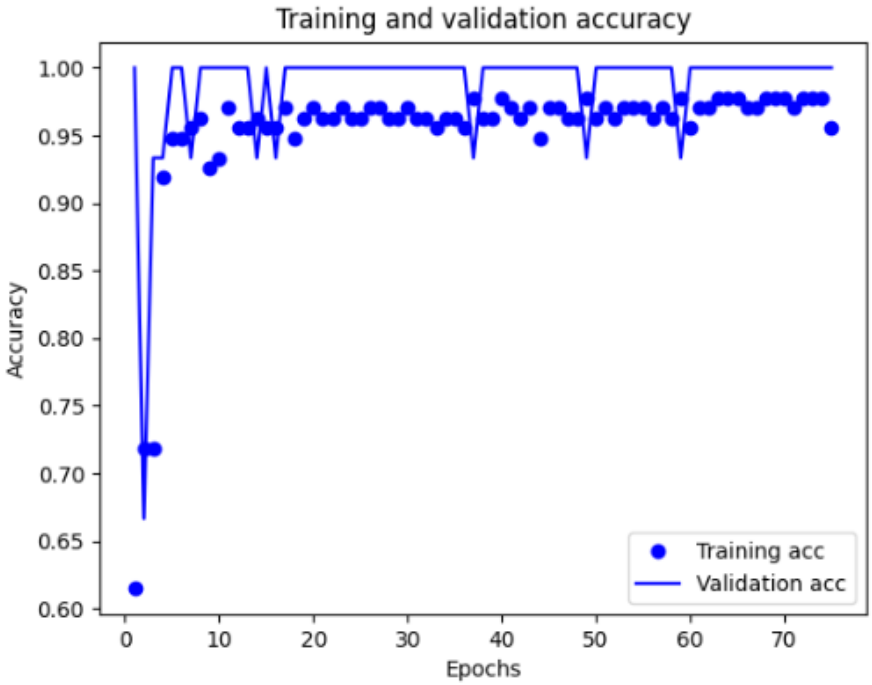


Рисунок 4 – Показатели ИНС #4

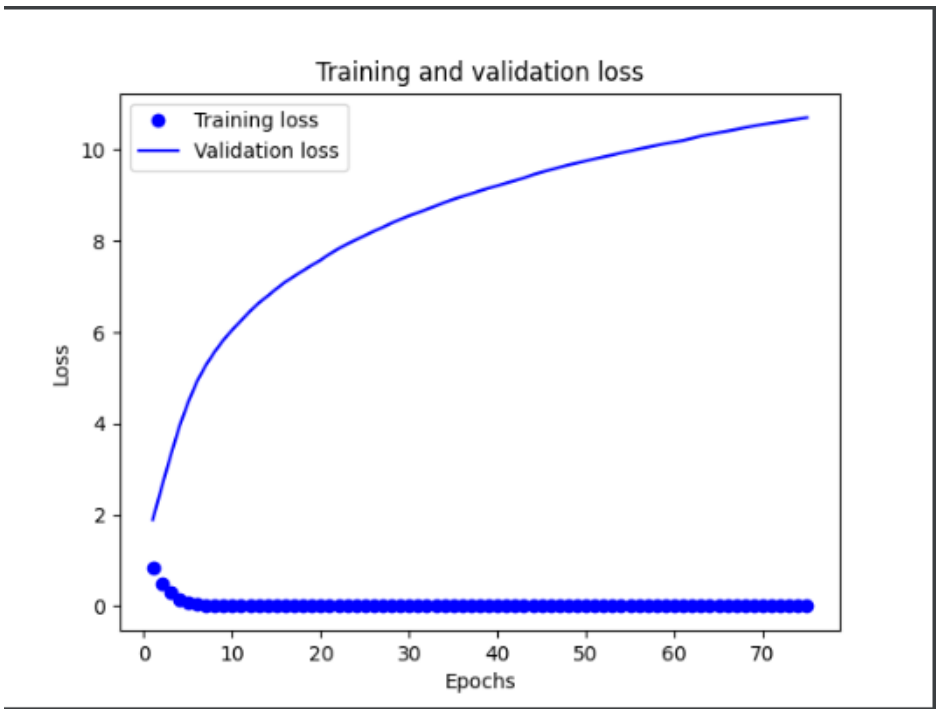
Из рисунков можно заметить, что увеличение числа нейронов в скрытых слоях значительно увеличило точность и уменьшило ошибки. Результаты на обучаемых и проверочных данных стали крайне близки.

Модель 5.

В модели 4 был изменен параметр validation_split с 0.1 на 0.5. Т.е. доля проверочных данных увеличилась с 10% до 50%.

```
model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.5)
```



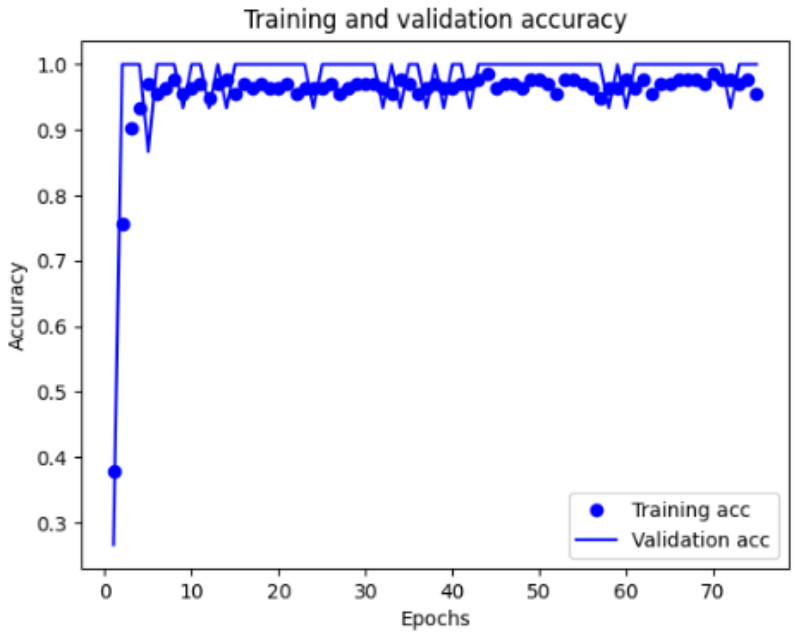


Рисунок 5 – Показатели ИНС #5

Результаты оказались неожиданными. Потери на проверяемых данных оказались равны нулю, а на обучаемых стремительно растут. Показатель точности стабильно был равен ~0.9-1.

Результат нестабилен при различных запусках.

Выводы.

Изучив результаты 5 моделей, можно прийти к выводу, что 4 модель дает самый лучший результат, т.к. точность крайне высокая, а потери малы.

ПРИЛОЖЕНИЕ А.

Исходный код программы. Файл main.py.

```
import pandas
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder

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(4, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.5)
history_dict = history.history

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```