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 _____ Степанов В.Д.

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.

Выполнения работы.

Из методических материалов была построена следующая модель.

Модель №1

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

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

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

Результаты запуска модели:

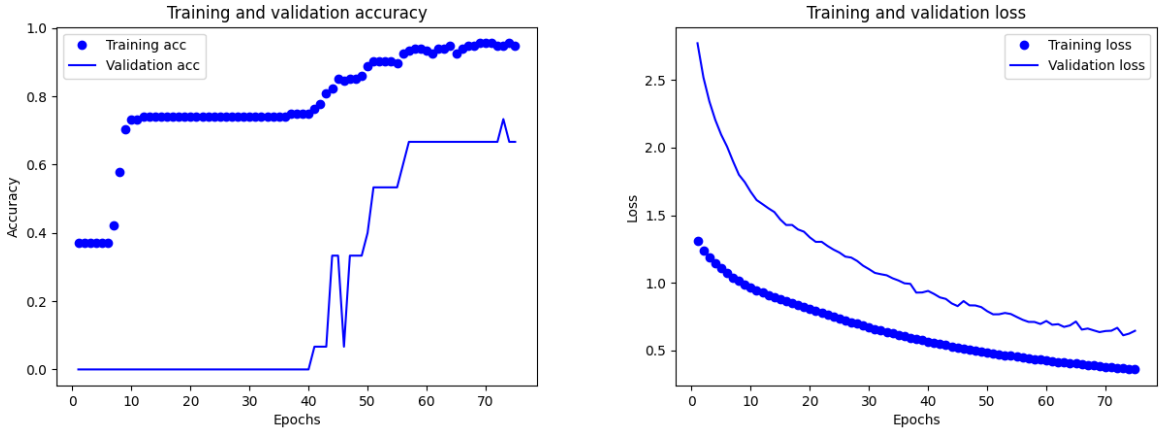

Рисунок 1 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.2 до 0.6
- Значение функция потерь на данных тестирования изменялось от 0.3 до 0.8

- Процент верных предположений на данных обучение изменялся от 75% до 98%
- Процент верных предположений на данных тестирования изменялся от 0% до 100%

Значения сильно отличаются друг от друга в зависимости от запуска.

Рассмотрим разные архитектуры ИНС. Добавим еще один слой.

Модель №2

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

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

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

Результаты запуска модели:

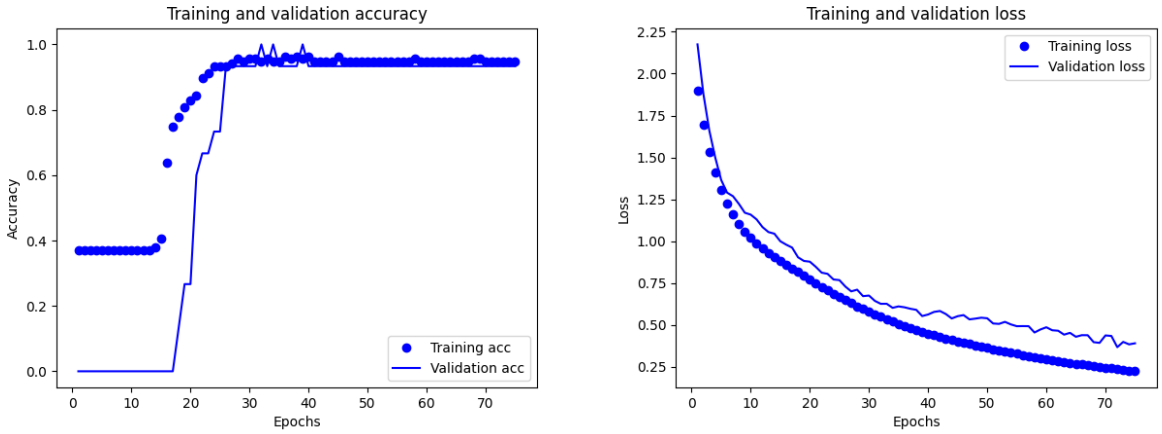

Рисунок 2 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.3 до 1.0

- Значение функция потерь на данных тестирования изменялось от 0.4 до 1.3
- Процент верных предположений на данных обучение изменялся от 37% до 95%
- Процент верных предположений на данных тестирования изменялся от 0% до 93%

Значения, как и в первом случае, сильно отличаются друг от друга в зависимости от запуска.

Увеличим количество нейронов в слоях.

Модель №3

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

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

Результаты запуска модели:

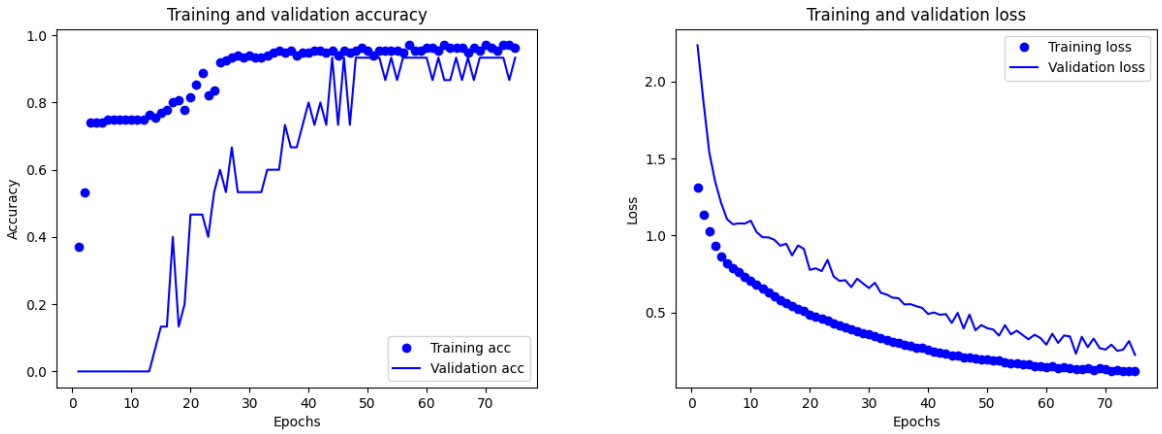

Рисунок 3 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.08 до 0.10
- Значение функция потерь на данных тестирования изменялось от 0.08 до 0.25
- Процент верных предположений на данных обучение изменялся от 93% до 99%
- Процент верных предположений на данных тестирования изменялся от 93% до 100%

По сравнению с предыдущими моделями, эта показывает более стабильные и точные результаты.

Удалим добавленный ранее слой.

Модель №4

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

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

Результаты запуска модели:

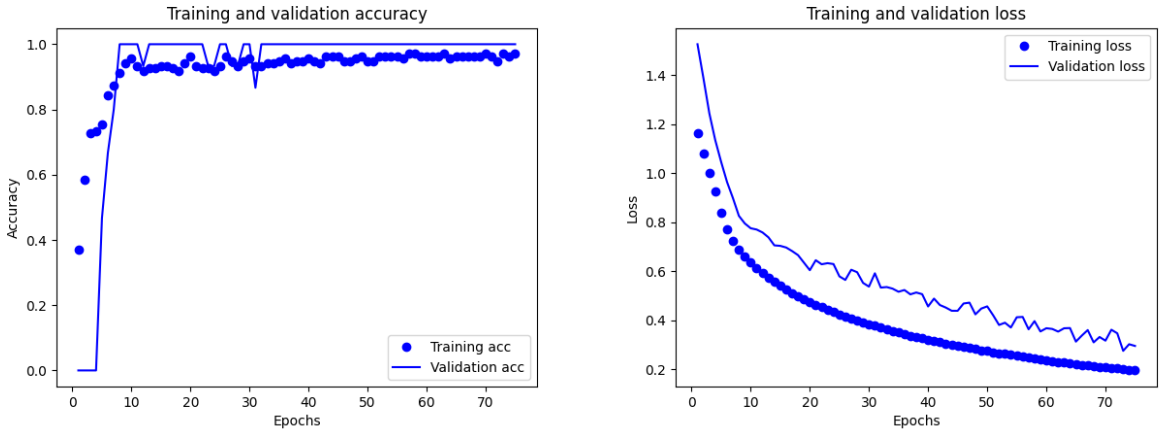

Рисунок 4 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.18 до 0.36
- Значение функция потерь на данных тестирования изменялось от 0.28 до 0.49
- Процент верных предположений на данных обучение изменялся от 88% до 99%
- Процент верных предположений на данных тестирования изменялся от 80% до 100%

По сравнению с первой и второй моделями, эта показывает более стабильные и точные результаты, но значение функции потерь у данной модели больше, чем у третьей модели.

Для изучения разных параметров обучения возьмём за основу модель №3.

Параметр `validation_split` сообщает какую часть от входных данных взять для тестирования сети после каждой эпохи (эти данные не будут учувствовать в тестировании). Увеличим этот параметр с 10% до 20%.

Модель №5

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

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

H = model.fit(X, dummy_y, epochs=75, batch_size=10, validation_split=0.2)
```

Результаты запуска модели:

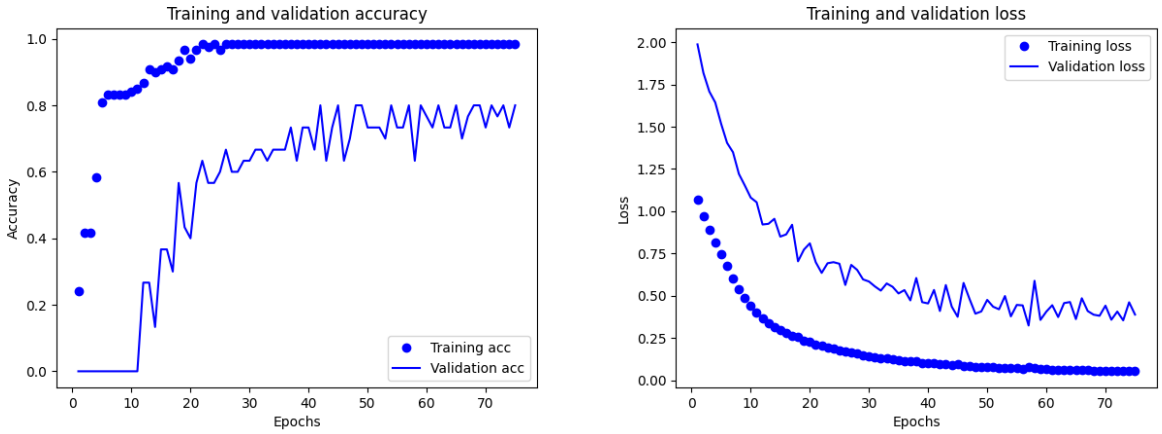

Рисунок 5 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.05 до 0.13
- Значение функция потерь на данных тестирования изменялось от 0.38 до 0.61
- Процент верных предположений на данных обучение изменялся от 98% до 99%
- Процент верных предположений на данных тестирования изменялся от 63% до 80%

Сравнивая результаты с моделью №3 видно, что значение функции потерь данных увеличился, а процент верных предположений уменьшился. Предположительно это связано, что количество обучающих данных уменьшилось на 10% (по отношению ко всему количеству данных), а количество тестовых данных увеличилось на 10%.

Параметр `epochs` сообщает повторов обучения сети на обучающих данных. Увеличим его с 75 до 100 эпох.

Модель №6

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

H = model.fit(X, dummy_y, epochs=100, batch_size=10, validation_split=0.2)
```

Результаты запуска модели:

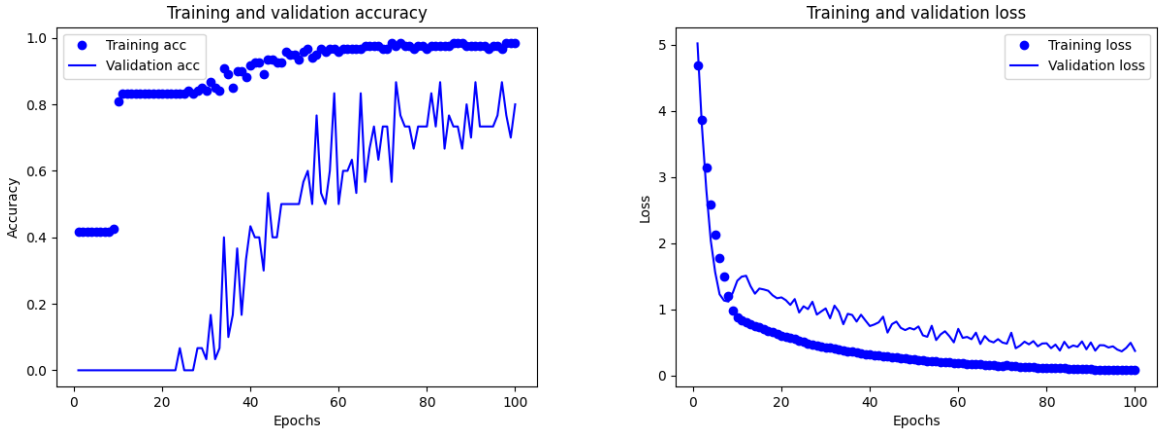

Рисунок 6 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.02 до 0.10
- Значение функция потерь на данных тестирования изменялось от 0.29 до 0.45
- Процент верных предположений на данных обучение изменялся от 96% до 99%
- Процент верных предположений на данных тестирования изменялся от 73% до 80%

Сравнивая с моделью №5 видно, что значения функции потерь уменьшились, а минимальное значение процента верных предположений на данных тестирования увеличилось.

Параметр `batch_size` сообщает после какого количества пропущенных элементов необходимо пересчитать веса. Увеличим с 10 до 50.

Модель №7

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

H = model.fit(X, dummy_y, epochs=100, batch_size=50, validation_split=0.2)
```

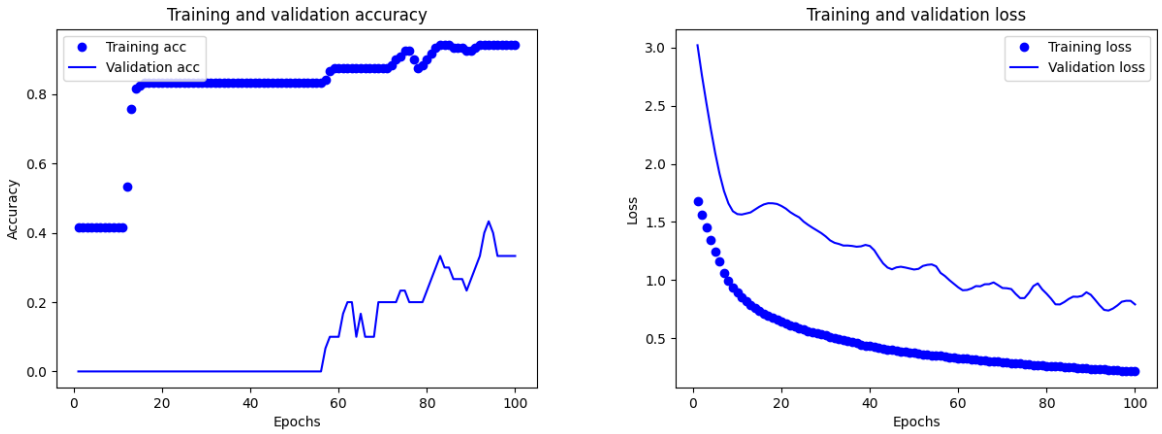

Рисунок 7 – Графики точности и ошибки

При нескольких запусках обучения результаты после последней эпохи такие:

- Значение функции потерь на данных обучение изменялось от 0.20 до 0.35
- Значение функция потерь на данных тестирования изменялось от 0.79 до 1.00
- Процент верных предположений на данных обучение изменялся от 85% до 93%
- Процент верных предположений на данных тестирования изменялся от 0% до 33%

Сравнивая с моделью №6 видно, что значения функции потерь увеличились, а минимальное значение процента верных предположений на данных тестирования сильно уменьшились.

Лучше всего показала себя модель №3.

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

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

Если бы у нас было больше данных для обучения, то модель № 6 тоже может показать хорошие результаты

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

H = model.fit(X, dummy_y, epochs=100, batch_size=10, validation_split=0.2)
```

Выводы.

При выполнение лабораторной работы ознакомились с задачей классификации, создали модель ИНС и настроили оптимальные параметры обучения, при которых ИНС выдавала приемлемые результаты.