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

Студентка гр. 7303 _____ Шишкин И.В.

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

Выполнение работы.

Был скачан файл с набором данных, на которых нейронная сеть будет обучаться. Так как в данной программе больше двух классов, то выходные атрибуты были преобразованы из вектора в матрицу, состоящую из нулей и единиц с помощью функции `to_categorical()`. Затем, была создана модель, состоящая из двух слоев (листинг 1) – она является моделью №1.

Листинг 1 – 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)
```

Графики этой модели представлены на рис. 1.

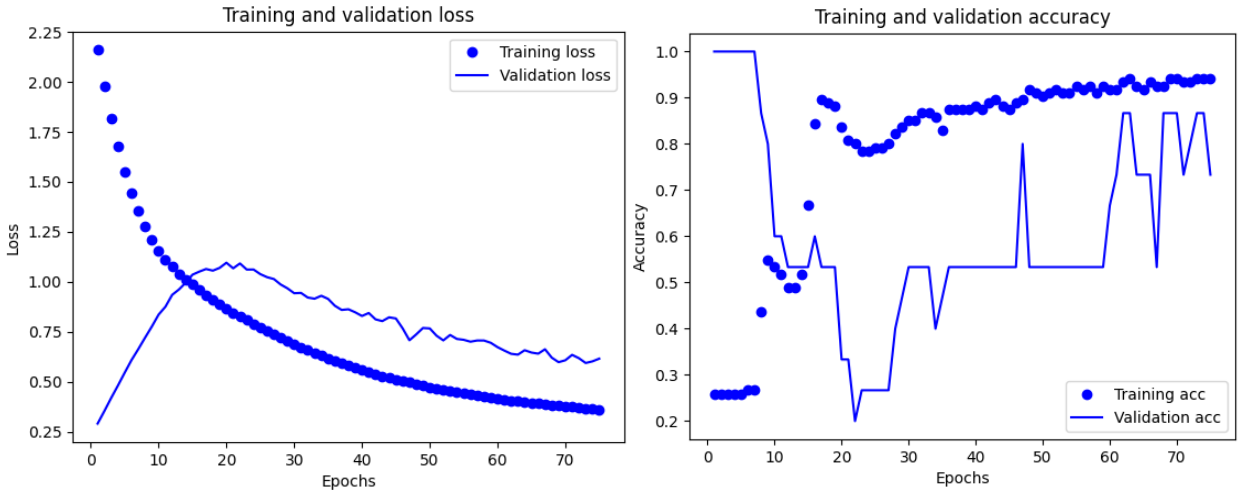

Рисунок 1 – Графики модели №1

От запуска к запуску результаты достаточно сильно меняются. И если точность при обучении после 75 эпох достигала колебалась примерно от 35 до 95%, то точность на неизвестной ей данных могла быть как 0%, так и 100%.

Для модели №2 количество нейронов на слое было увеличено до 16 (листинг 2).

Листинг 2 – Модель №2

```
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)
```

Графики этой модели представлены на рис. 2.

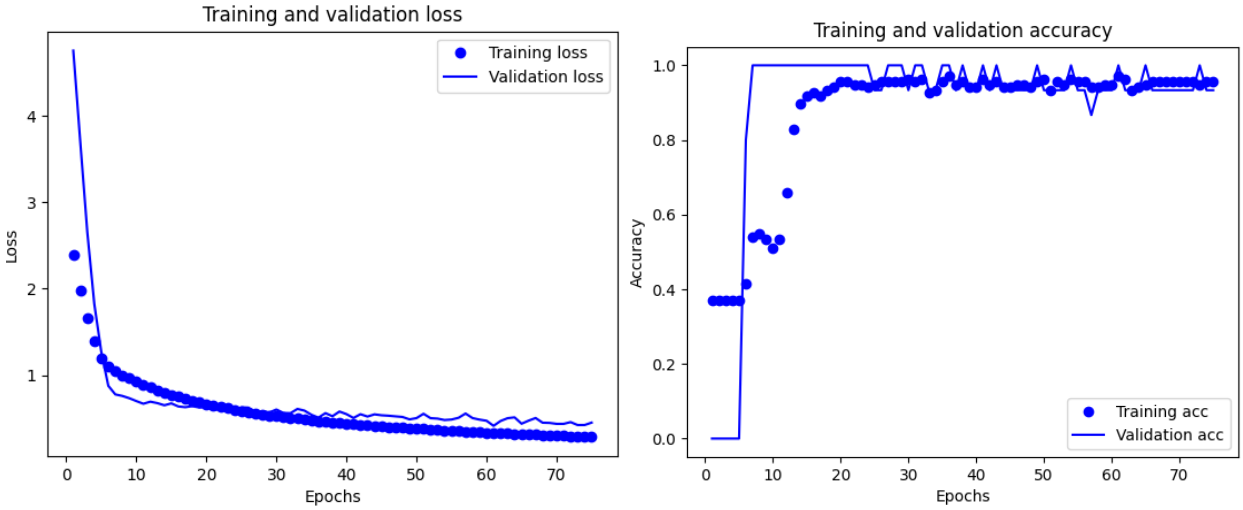

Рисунок 2 – Графики модели №2

Результат получается уже лучше: потери уменьшаются, точность на тестовых данных колеблется от 94 до 98%, на проверочных – от 33 до 100%.

Для модели №3 был добавлен еще один скрытый слой (листинг 3).

Листинг 3 – Модель №3

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(8, 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)
```

Графики этой модели представлены на рис. 3.

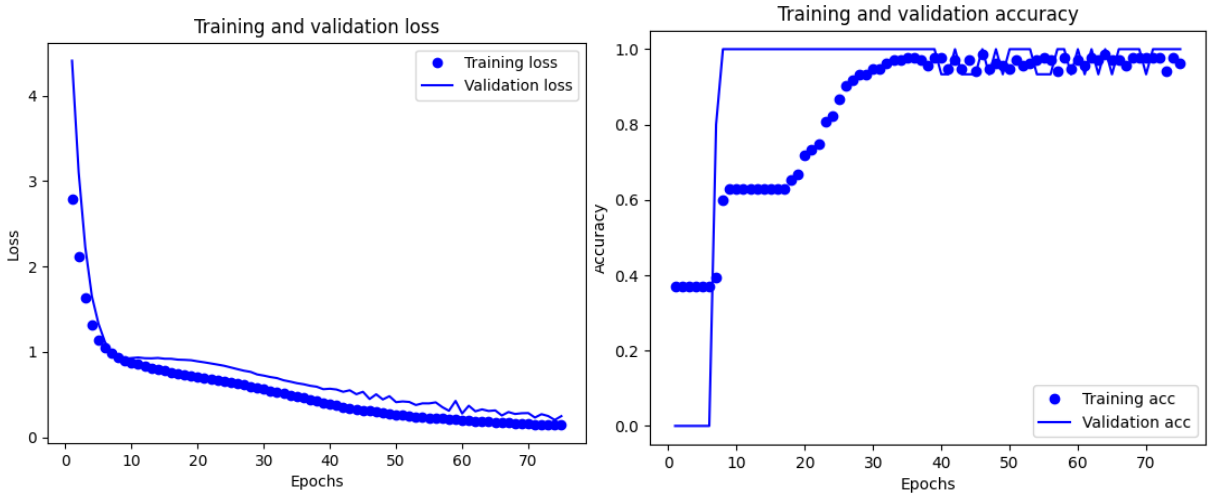

Рисунок 3 – Графики модели №3

Потери остались примерно такими же. Точность на тестовых данных колеблется от 92 до 99%, на проверочных – от 87 до 100%.

Параметр epochs в функции model.fit отвечает за количество эпох (итераций). Для модели №4 увеличим его в 2 раза (листинг 4).

Листинг 4 – Модель №4

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
H = model.fit(X, dummy_y, epochs=150, batch_size=10,
validation_split=0.1)
```

Графики этой модели представлены на рис. 4.

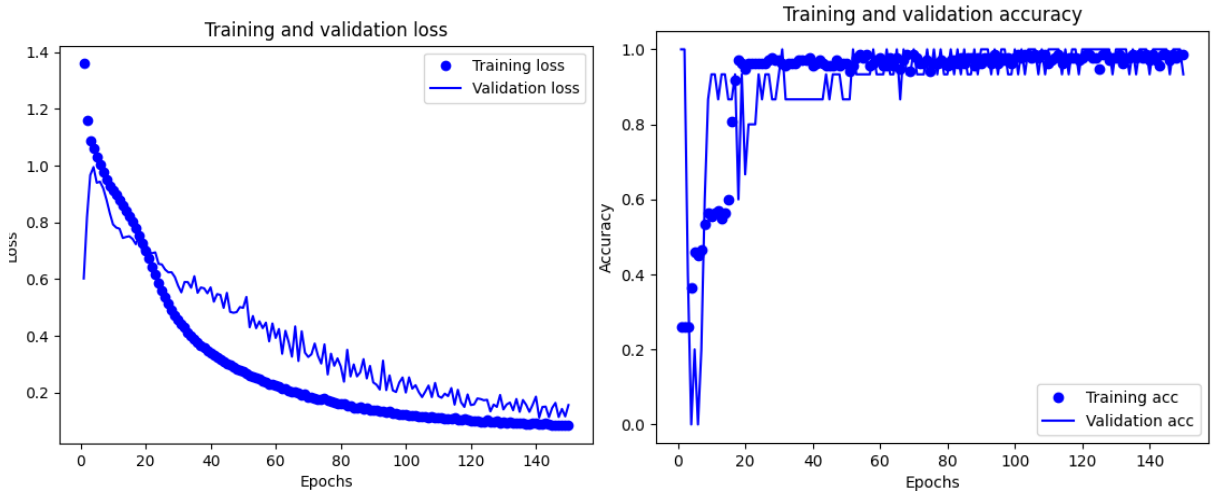

Рисунок 4 – Графики модели №4

Потери ожидаемо продолжают уменьшаться, при этом точность остается примерно такой же (на тестовых – от 95 до 99%, на проверочных – от 87 до 100%).

Для модели №5 поменяем значение параметра batch_size (то количество элементов, после которого будут меняться веса) с 10 на 100 (листинг 5).

Листинг 5 – Модель №5

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
```

```
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
H = model.fit(X, dummy_y, epochs=150, batch_size=100,
validation_split=0.1)
```

Графики этой модели представлены на рис. 5.

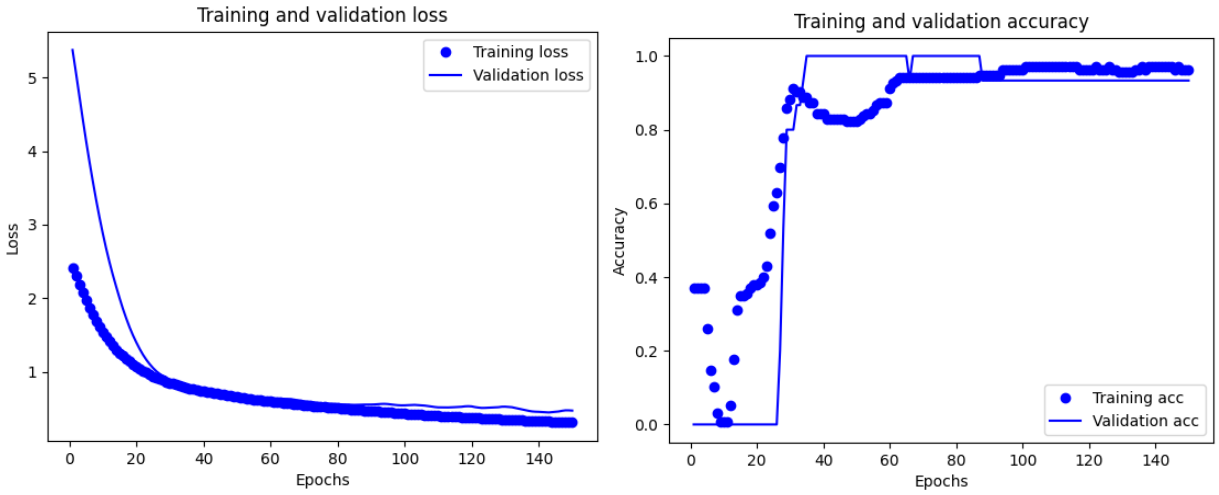

Рисунок 5 – Графики модели №5

В данном случае модель себя показывает так же, как и первая. Потери увеличились, точность на тестовых данных изменяется от 73 до 98%, на проверочных – от 0 до 100%. Причем чем больше это значение, тем хуже модель показывает результаты. Если поставить это значение равным 50, то точность достигнет уровня четвертой модели, но потери возрастут. Если же наоборот уменьшить количество образцов до 5, то модель показывает немного лучше результаты, чем в четвертой модели.

Для модели №6 было изменено значение параметра validation_split (часть обучающих данных, которая будет использоваться в качестве данных проверки.) с 0.1 на 0.2 (листинг 6).

Листинг 6 – Модель №6

```
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

```
H = model.fit(X, dummy_y, epochs=150, batch_size=100, validation_split=0.2)
```

Графики этой модели представлены на рис. 6.

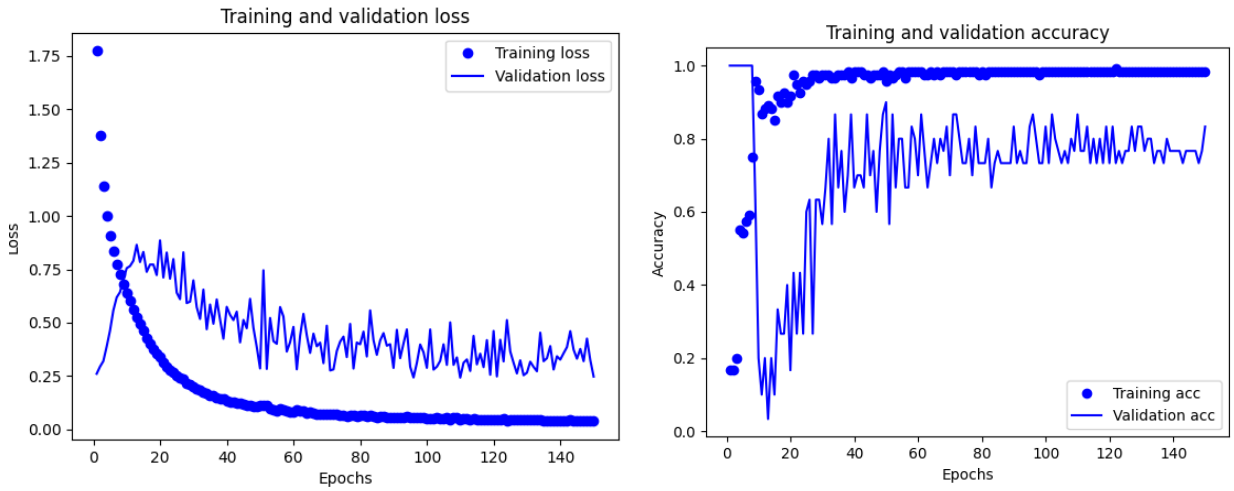

Рисунок 6 – Графики модели №6

Потери на тестовых данных уменьшились, точность почти всегда 98%, но точность на проверочных данных, в среднем, около 80%.

Таким образом, лучше всего себя показала модель с параметрами epochs = 150, batch_size = 5, validation_split = 0.1.

Выводы.

Была реализована классификация сортов растения ирис по четырем признакам: размерам пестиков и тычинок его цветков. Изучены различные архитектуры ИНС (разное кол-во слоев, разное кол-во нейронов на слоях). Изучено обучение при различных параметрах обучения (параметры ф-ций fit). Построены графики ошибок и точности в ходе обучения. Наилучшей моделью оказалась модель под номером 5 с параметром batch_size = 5.