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

Студентка гр. 8383 _____ Гречко В.Д.

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.

1. Написание первой модели ИНС.

Данные были получены из файла iris.csv. Далее была выполнена загрузка данных и переход от текстовых меток к категориальному вектору. После этого была создана модель со следующими параметрами: два dense-слоя с функцией активации relu и 3-переменный слой потерь (softmax layer), возвращающий массив с 3 оценками вероятностей (в сумме дающих 1), 75 эпох и размер батча – 10. В качестве оптимизатора будет использоваться Adam, функцией потерь categorical_crossentropy, а в качестве метрики используется точность. Исходный код приведён в приложении А.

Графики потерь и точности представлены на рисунках 1 и 2.

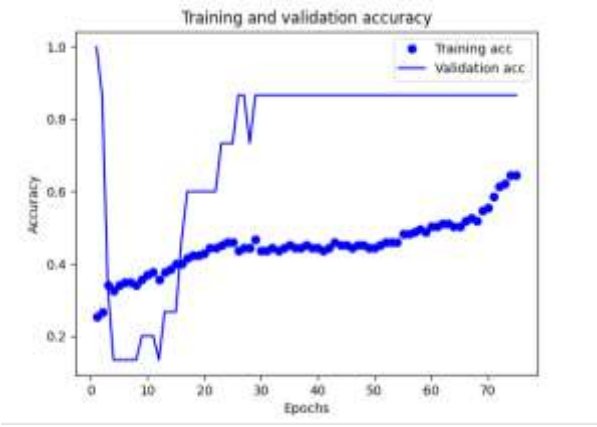


Рисунок 1 – график точности

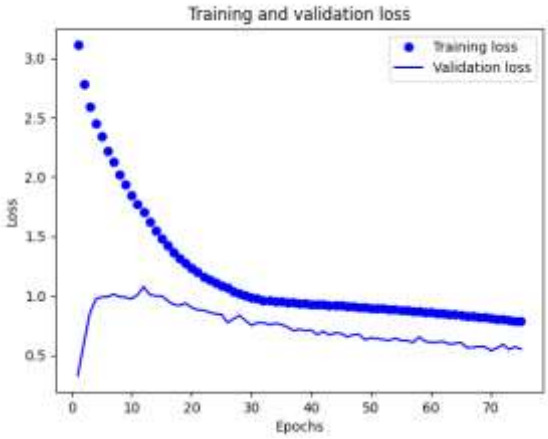


Рисунок 2 – график потерь

Как видно на графиках, потери стремятся к нулю, а точность возрастает, но с достаточно медленной скоростью.

2. Изучение различных архитектур ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)

Рассмотрим вариант с увеличением количества слоёв без увеличения количества нейронов на слоях.

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

Графики потерь и точности представлены на рисунках 3 и 4.

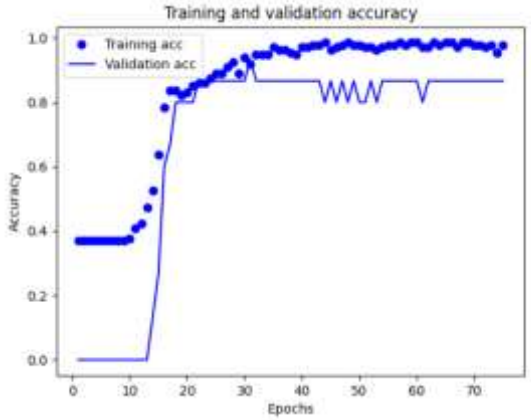


Рисунок 3 – график точности

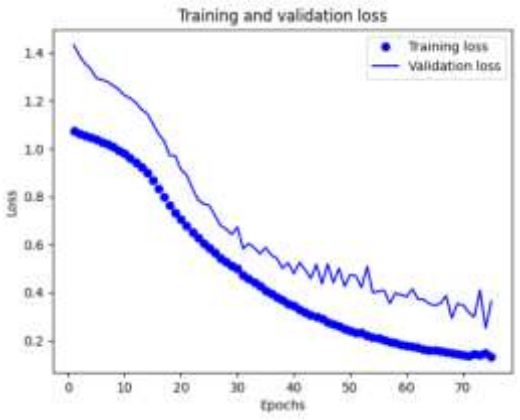


Рисунок 4 – график потерь

Вывод: увеличение количества слоев улучшает работу ИНС.

Вариант с увеличением количества нейронов на слоях и одновременно количество слоёв.

```
model.add(Dense(4, activation='relu'))
model.add(Dense(40, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Графики потерь и точности представлены на рисунках 5 и 6.

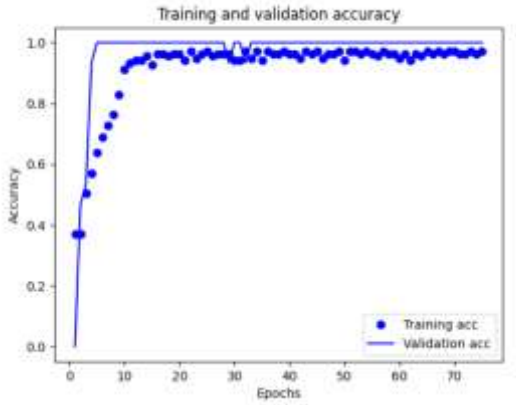


Рисунок 5 – график точности

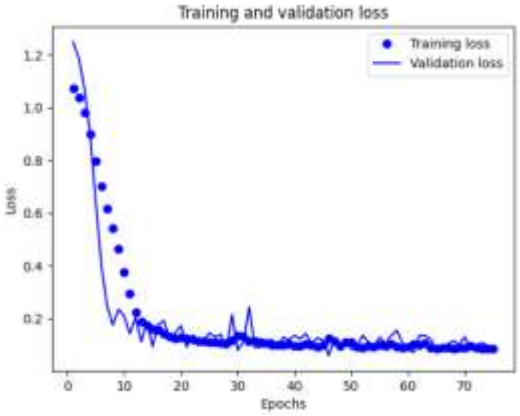


Рисунок 6 – график потерь

Вывод: увеличение количества слоев и одновременно нейронов улучшает работу ИНС. Более точные результаты достигаются быстрее.

Рассмотрим вариант с увеличением количества нейронов на слоях без увеличения количества слоёв.

```
model.add(Dense(40, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

Графики потерь и точности представлены на рисунках 7 и 8.

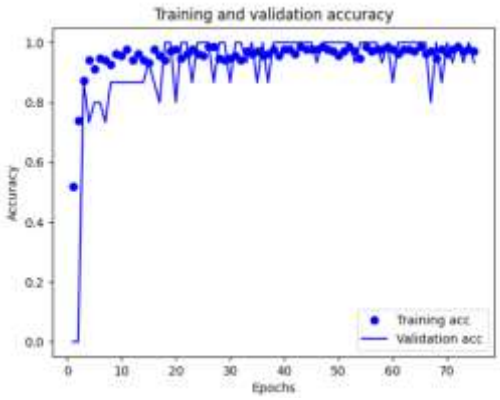


Рисунок 7 – график точности

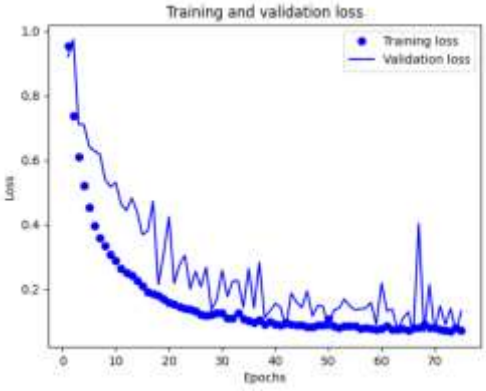


Рисунок 8 – график потерь

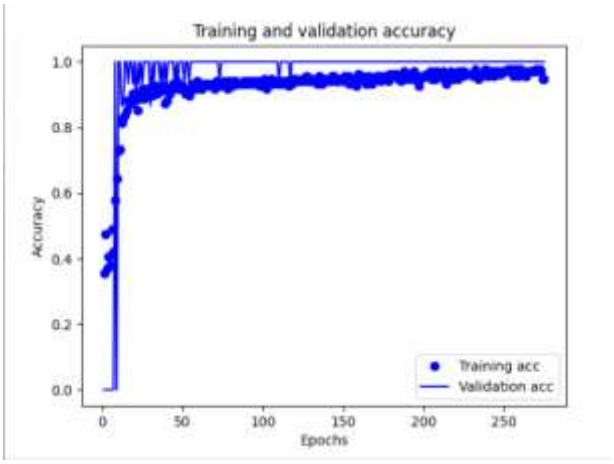
Вывод: увеличение только количества нейронов улучшает работу ИНС. Однако результат уступает варианту выше (точность в 0.2 достигается примерно в 15 – 18 эпохе против 11)

3. Изучение обучения при различных параметрах обучения (параметры функций fit)

Рассмотрим вариант с увеличением количества эпох.

```
history = model.fit(X, dummy_y, epochs=275, batch_size=10, validation_split=0.1)
```

Графики потерь и точности представлены на рисунках 9 и 10.



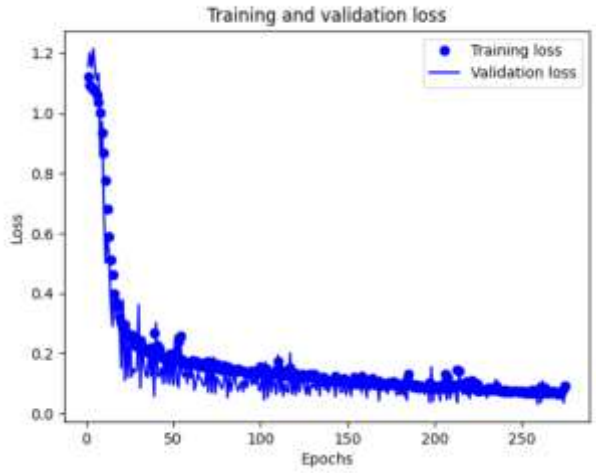


Рисунок 9 – график точности

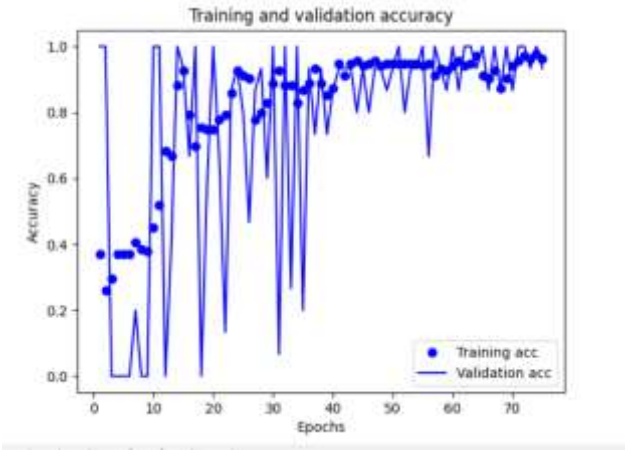
Рисунок 10 – график потерь

Вывод: увеличение количества итераций улучшает работу ИНС.

Рассмотрим вариант с увеличением размера батчей.

```
history = model.fit(X, dummy_y, epochs=75, batch_size= 20, validation_split=0.1)
```

Графики потерь и точности представлены на рисунках 11 и 12.



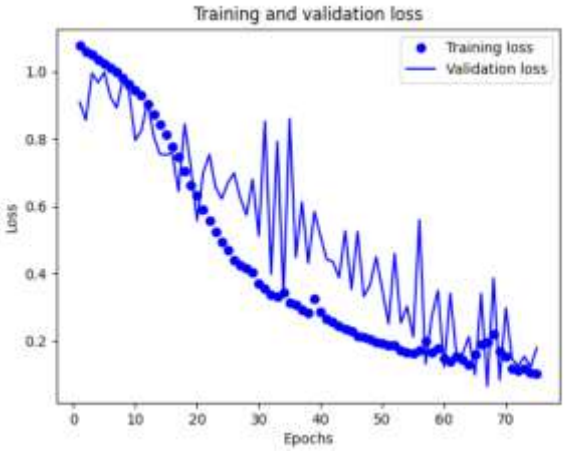


Рисунок 11 – график точности

Рисунок 12 – график потерь

Рассмотрим, наоборот, уменьшение размера батчей

```
history = model.fit(X, dummy_y, epochs=75, batch_size= 5, validation_split=0.1)
```

Графики потерь и точности представлены на рисунках 13 и 14.

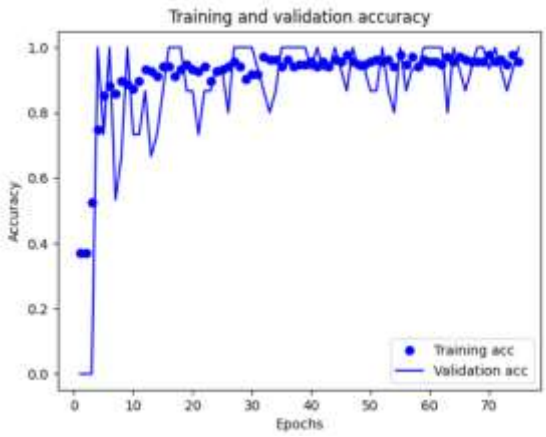


Рисунок 13 – график точности

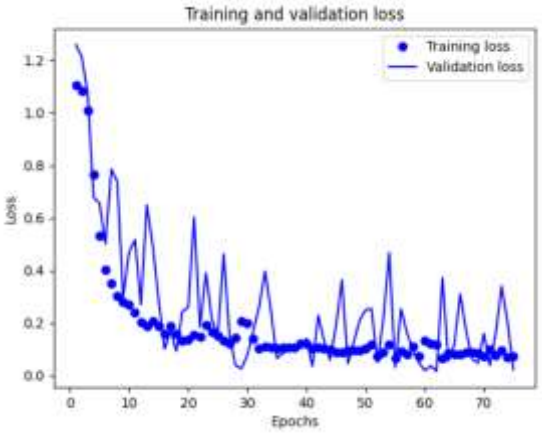


Рисунок 14 – график потерь

Вывод: уменьшение размера батчей улучшает работу ИНС (чем меньше размер, тем чаще модель переобучается).

Рассмотрим изменение параметра validation_split.

```
history = model.fit(X, dummy_y, epochs=75, batch_size=5, validation_split=0.3)
```

Графики потерь и точности представлены на рисунках 15 и 16.

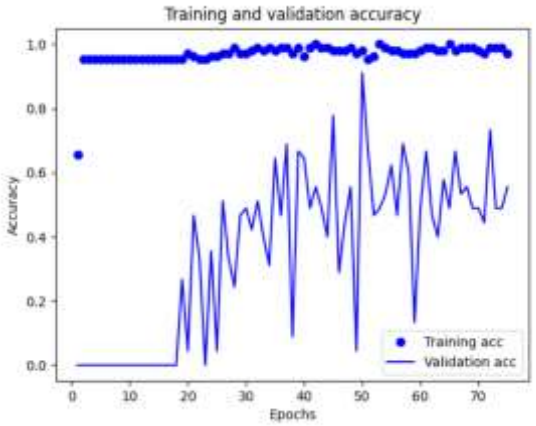


Рисунок 15 – график точности

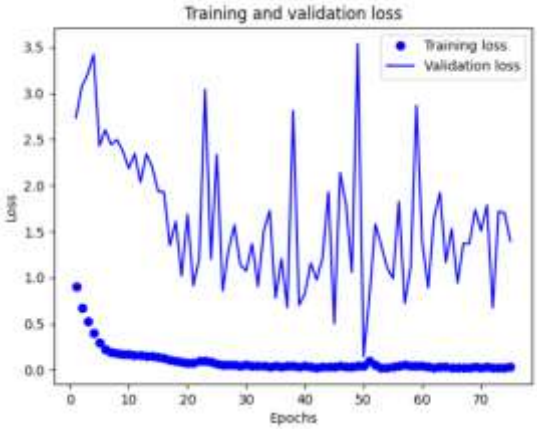


Рисунок 16 – график потерь

Из-за большого процента валидационных данных и малого процента тренировочных графики не сходятся, и результат трудно оценим.

```
history = model.fit(X, dummy_y, epochs=75, batch_size=5, validation_split=0.3)
```

Графики потерь и точности представлены на рисунках 15 и 16.

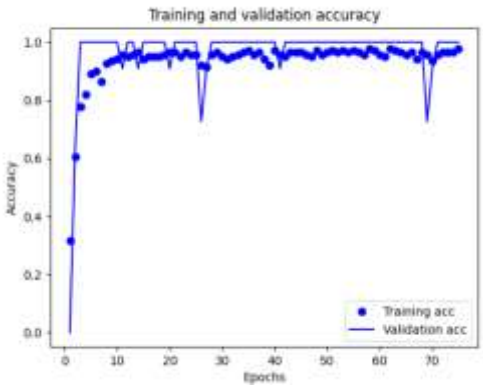


Рисунок 15 – график точности

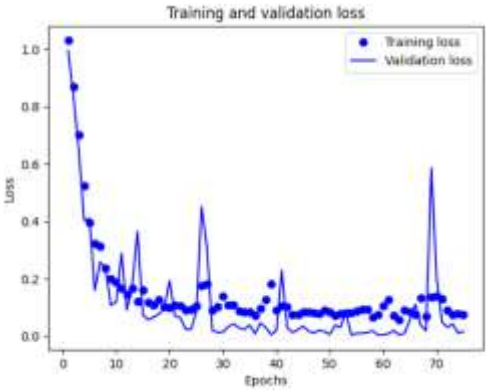


Рисунок 16 – график потерь

Вывод: самый оптимальный вариант – 10% валидационных данных

Исходя из всех исследований можно определить наилучшую модель:

- Одновременное увеличение и числа нейронов, и числа слоёв
- Сохранение 10% валидационных данных
- Увеличение числа итераций
- Размер батчей в пределах от 5 до 10

Наилучшая модель представлена в приложении А.

Выводы.

В результате выполнения лабораторной работы была создана и обучена модель искусственной нейронной сети. Также было проведено исследование на нахождение наилучшей модели и её параметров.

ПРИЛОЖЕНИЕ А

```
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
IMPORT MATPLOTLIB.PYPLOT AS PLT #ИМПОРТ МОДУЛЯ ДЛЯ ГРАФИКОВ

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(40, ACTIVATION='RELU'))
MODEL.ADD(DENSE(30, ACTIVATION='RELU'))
MODEL.ADD(DENSE(50, 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=5,
VALIDATION_SPLIT=0.1)

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()
```