МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студентка гр. 7382 _____ Головина Е.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Реализовать классификацию между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей.

60 входных значений показывают силу отражаемого сигнала под определенным углом. Входные данные нормализованы и находятся в промежутке от 0 до 1.

Задачи.

- Ознакомиться с задачей бинарной классификации
- Загрузить данные
- Создать модель ИНС в tf.Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Изменить модель и провести сравнение. Объяснить результаты

Требования.

- Изучить влияние кол-ва нейронов на слое на результат обучения модели.
- Изучить влияние кол-ва слоев на результат обучения модели
- Построить графики ошибки и точности в ходе обучения
- Провести сравнение полученных сетей, объяснить результат

Ход работы.

1. Знакомство с задачей бинарной классификации

Бинарная классификация относится к одному из основных классов машинного обучения, т.н. *обучение с учителем*. При обучении с учителем на вход подается тренировочный набор данных, на которых нейронная сеть учится

и по окончании обучения способна по новым данным (похожим на тренировочные данные) получать определенный результат.

Смысл задачи бинарной классификации: поданный на вход объект нужно определить в один из классов. Сам объект – это набор свойств (в данной лабораторной работе эти свойства – размеры пестиков и тычинок цветка). Тренировочные данные в таких задачах – это наборы свойств с указанием к какому классу относится данный набор и таких наборов должно быть достаточно много.

2. Загрузка данных

Данные, как было отмечено ранее, это значения силы отражаемого сигнала радара от поверхностей под определенными углами. Всего 2 класса: R – камни и M – металлические цилиндры. Данные представлены в следующем формате:

```
0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109
,0.2111,0.1609,0.1582,0.2238,0.0645,0.0660,0.2273,0.3100,0.299
9,0.5078,0.4797,0.5783,0.5071,0.4328,0.5550,0.6711,0.6415,0.71
04,0.8080,0.6791,0.3857,0.1307,0.2604,0.5121,0.7547,0.8537,0.8
507,0.6692,0.6097,0.4943,0.2744,0.0510,0.2834,0.2825,0.4256,0.
2641,0.1386,0.1051,0.1343,0.0383,0.0324,0.0232,0.0027,0.0065,0
.0159,0.0072,0.0167,0.0180,0.0084,0.0090,0.0032,R
0.0453,0.0523,0.0843,0.0689,0.1183,0.2583,0.2156,0.3481,0.3337
,0.2872,0.4918,0.6552,0.6919,0.7797,0.7464,0.9444,1.0000,0.887
4,0.8024,0.7818,0.5212,0.4052,0.3957,0.3914,0.3250,0.3200,0.32
71,0.2767,0.4423,0.2028,0.3788,0.2947,0.1984,0.2341,0.1306,0.4
182,0.3835,0.1057,0.1840,0.1970,0.1674,0.0583,0.1401,0.1628,0.
0621,0.0203,0.0530,0.0742,0.0409,0.0061,0.0125,0.0084,0.0089,0
.0048,0.0094,0.0191,0.0140,0.0049,0.0052,0.0044,R
……
```

Набор данных хранится в файле `sonar.csv` и загружается напрямую с помощью модуля `pandas` в `python`.

```
dataframe = pandas.read_csv("sonar.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:60].astype(float)
Y = dataset[:,60]
```

Таким образом, в X находится массив наборов свойств, а в Y соответствующий свойствам класс (камень или цилиндр).

Классы представлены строками («R» и «M»), которые необходимо перевести в целочисленные значения 0 и 1 соответственно. Для этого применяется LabelEncoder из scikit-learn. Код данного преобразования приведен ниже.

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

В начале находим все классы, потом заменяем их значения на присвоенные им порядковые номера (0, 1).

3. Создание модели ИНС в Keras

Следующий код создает модель ИНС:

```
model = Sequential()
model.add(Dense(60, input_dim=60, init='normal',
activation='relu'))
model.add(Dense(1, init='normal', activation='sigmoid'))
```

Разберем, что тут происходит:

- Sequential() – создает пустую модель, которую будем последовательно строить
- .add() – добавляет следующий слой искусственной нейронной сети, т.е. в нашей модели два слоя.
- Dense() – говорит о том, что будет создан тесно связанный (полносвязный) слой. Внутри конструктора указываем целое число – размерность выходных данных и вид активирующей функции.

Таким образом, построена нейронная сеть с двумя слоями.

4. Настройка параметров обучения

Далее нужно настроить еще три параметра:

- *функцию потерь* (определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении). Для задач бинарной классификации применяется функция `binary_crossentropy`.
- *оптимизатор* (механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь). Мы выбрали алгоритм оптимизации Adam (разновидность градиентного спуска).
- выбрать *метрики* для мониторинга на этапах обучения и тестирования - здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

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

5. Обучение и оценка модели

Для того, чтобы провести обучение используем метод `.fit()`:

```
model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

Обучение проводится пачками, где batch_size – это размер одной пачки. Количество эпох (epochs) – это количество проходов по всем входным примерам. Validation_split – доля данных, которые будут использоваться в качестве данных проверки.

Во время обучения – информация о каждом проходе во время обучения будет выводиться в терминал в следующем виде:

```
Train on 187 samples, validate on 21 samples
Epoch 1/100
 10/187 [>.............................] - ETA: 9s - loss:
0.6961
187/187 [==============================] - 1s 3ms/step -
loss: 0.6907 - accuracy: 0.5829 - val_loss: 0.6848 -
val_accuracy:0.8095
……
Epoch 100/100
```

```
 10/187 [>.............................] - ETA: 0s - loss: 0.1682
187/187 [==============================] - 0s 123us/step - loss: 0.3093 - accuracy: 0.8770 - val_loss: 0.4399 - val_accuracy: 0.7619
208/208 [==============================] - 0s 19us/step
```

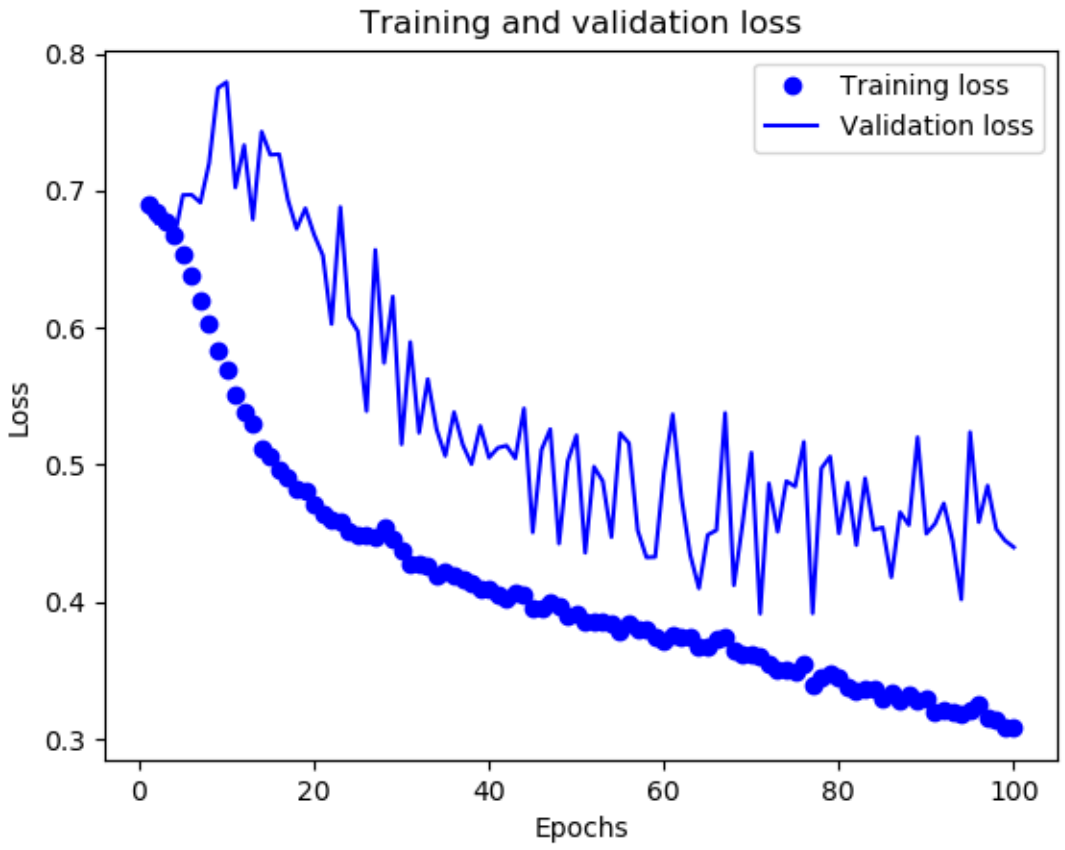
Для оценки полученной модели используем функцию `.evaluate()`.

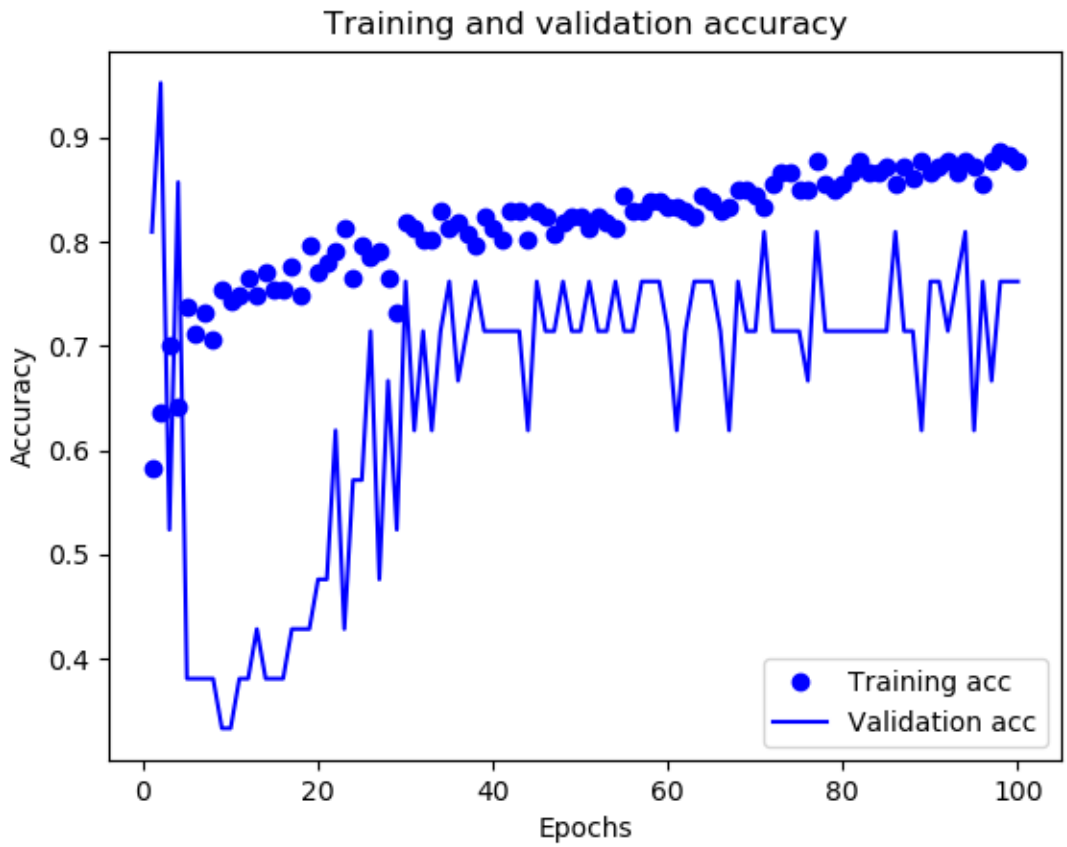
```
results = model.evaluate(X, encoded_Y)
print("Accurancy: ", results[1])
```

Полученные результаты:

```
Accurancy:  0.875
```

Также выведем графики ошибок и точности, используя данные, которые были сгенерированы во время обучения.





6. Эксперименты и выбор модели

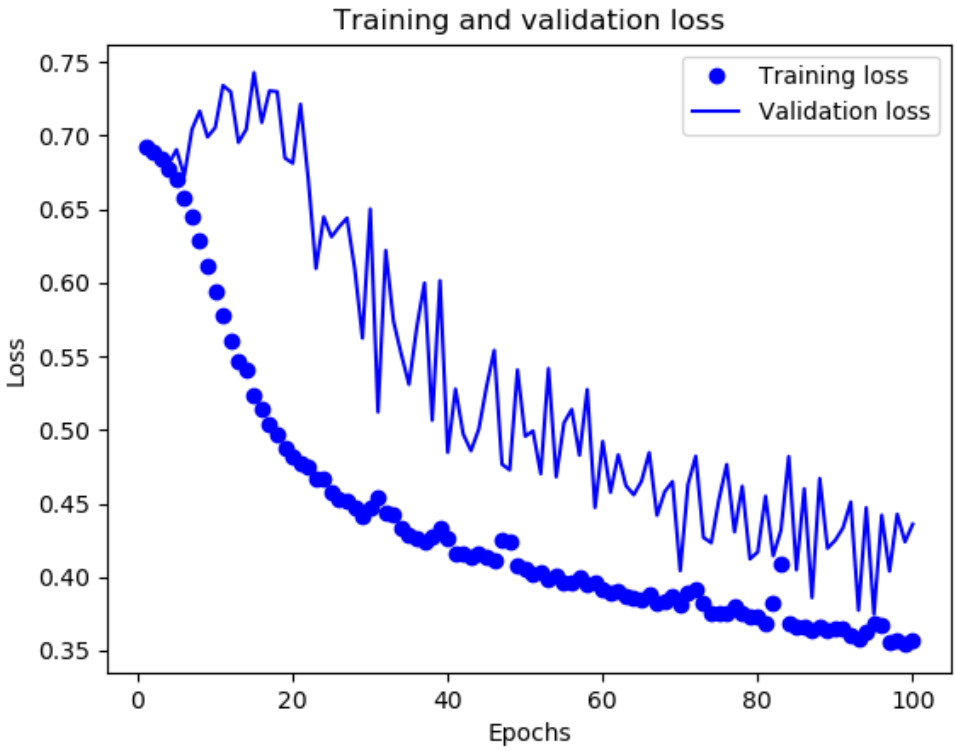
В представленном наборе данных присутствует некоторая избыточность, т.к. с разных углов описывается один и тот же сигнал. Вероятно, что некоторые углы отражения сигнала имеют большую значимость, чем другие. Изменение количества нейронов во входном слое напрямую влияет на количество признаков, с которыми будет работать нейронная сеть.

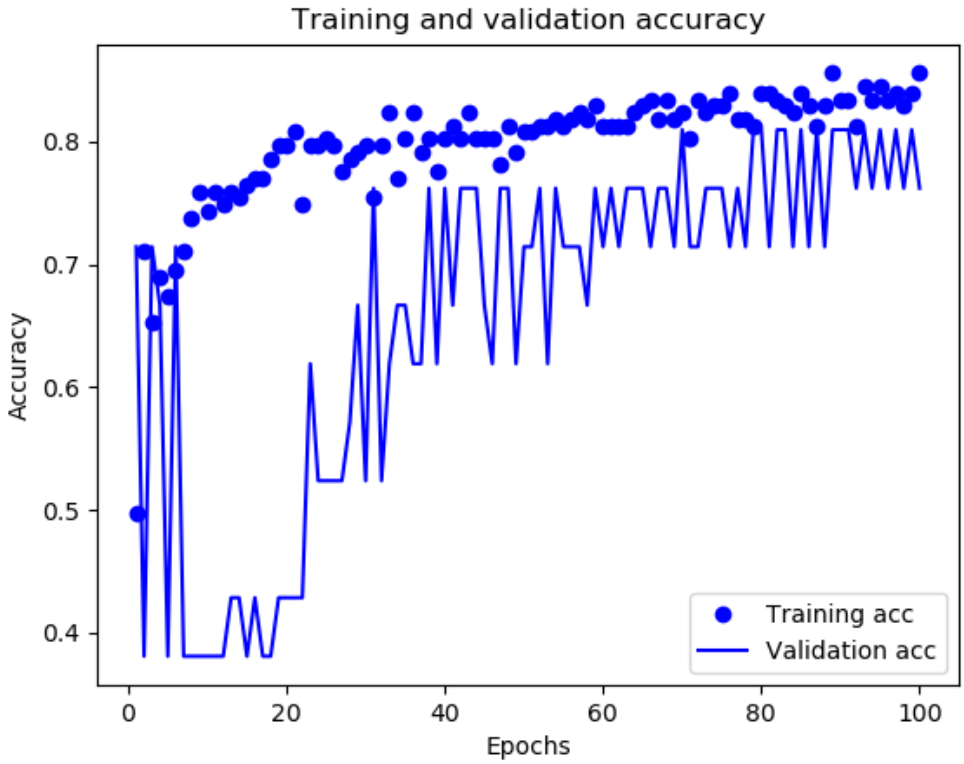
Эксперимент 1.

Уменьшение размера входного слоя в два раза и сравнение с результатами первоначальной архитектуры.

```
#Создание модели
model = Sequential()
model.add(Dense(30, input_dim=60, init='normal', activation='relu'))
model.add(Dense(1, init='normal', activation='sigmoid'))
```

Результат: Accurancy: 0.8365384340286255





Видно, что графики обучающих и тренировочных данных сблизились по сравнению с оригинальной моделью, а значит обучение происходит более эффективно, но значительных улучшений в точности замечено не было.

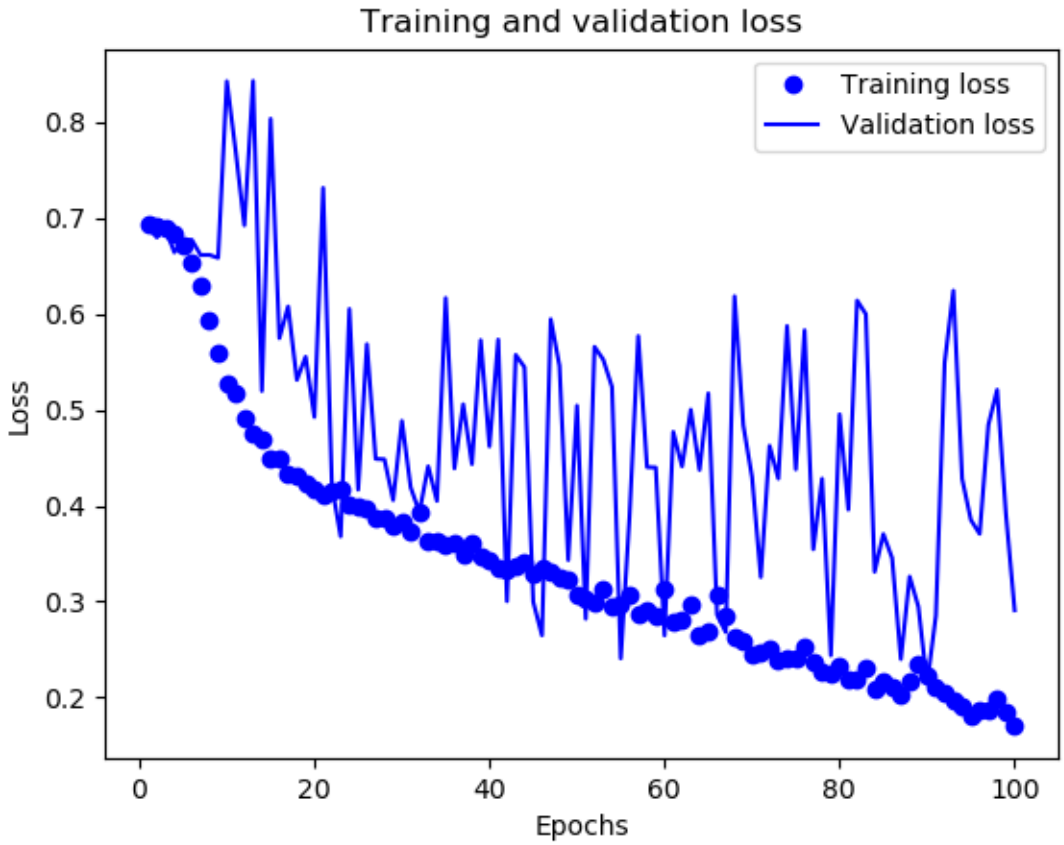
Эксперимент 2.

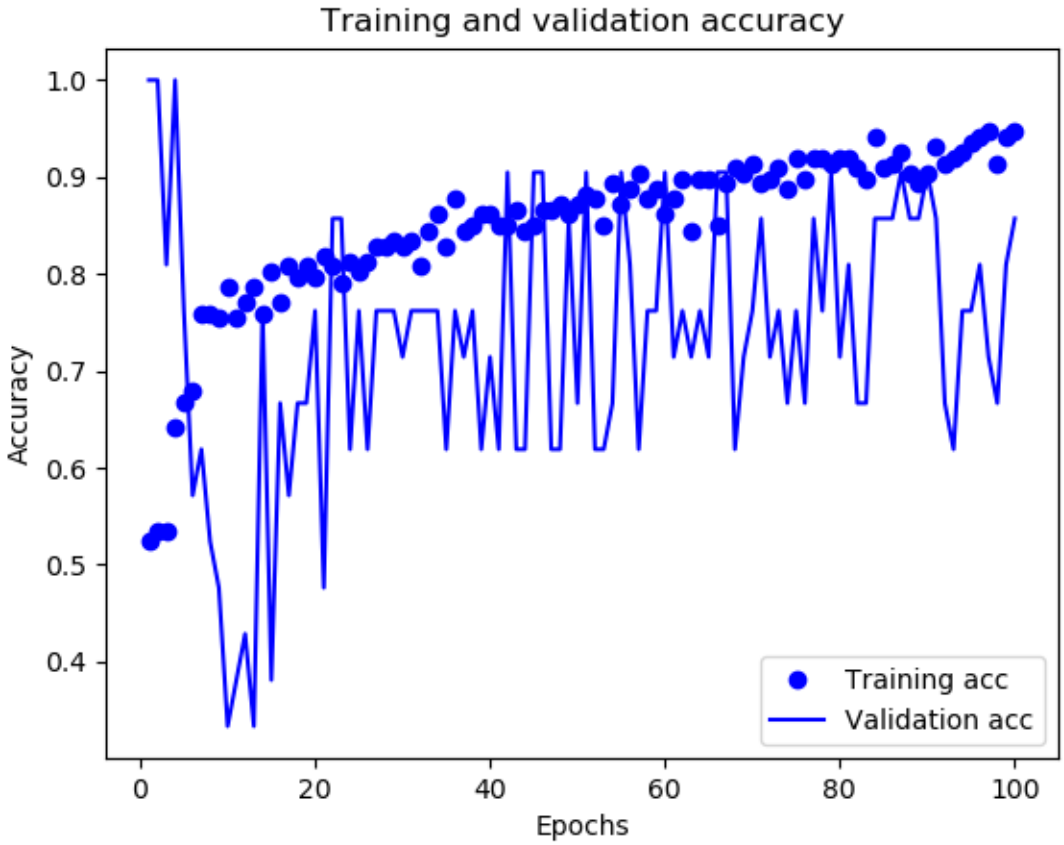
Нейронная сеть с несколькими слоями позволяет находить закономерности не только во входных данных, но и в их комбинации. Также, дополнительные слои позволяют ввести нелинейность в сеть, что позволяет получать более высокую точность.

Добавим промежуточный (скрытый) слой Dense в архитектуру сети с 15 нейронами и проанализируем результаты.

```
#Создание модели
model = Sequential()
model.add(Dense(60, input_dim=60, init='normal', activation='relu'))
model.add(Dense(15, init='normal', activation='relu'))
model.add(Dense(1, init='normal', activation='sigmoid'))
```

Accurancy: 0.9182692170143127


Training and validation loss
Training loss
Validation loss
Loss
Epochs
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
20
40
60
80
100

Как видно из графиков, результаты оказались намного лучше. Точность выше и потери меньше.

Вывод.

В ходе выполнения работы реализовали бинарную классификацию между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей, для этого изучили как конструировать искусственную нейронную сеть с помощью библиотеки keras в python, протестировали получившуюся модель, затем, проведя ряд экспериментов, выбрали наилучшую конфигурацию модели.