**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. 8383 _____ Кормщикова А.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

**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 классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Изменить модель и провести сравнение. Объяснить результаты

**Ход работы**

Были импортированы  необходимые для работы классы и функции.

```
import matplotlib.pyplot as plt
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
```

С помощью pandas был загружен набор данных, который были разделен на входные(X) и выходные данные(Y), также выходные параметры "R" и "M" были переведены в целочисленные значение 0 и 1 соответственно.

Была построена следующая модель ИНС_1:

```
model = Sequential()
model.add(Dense(60, input_dim=60, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',loss='binary crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

Были построены графики, результаты запуска модели представлены на рис. 1

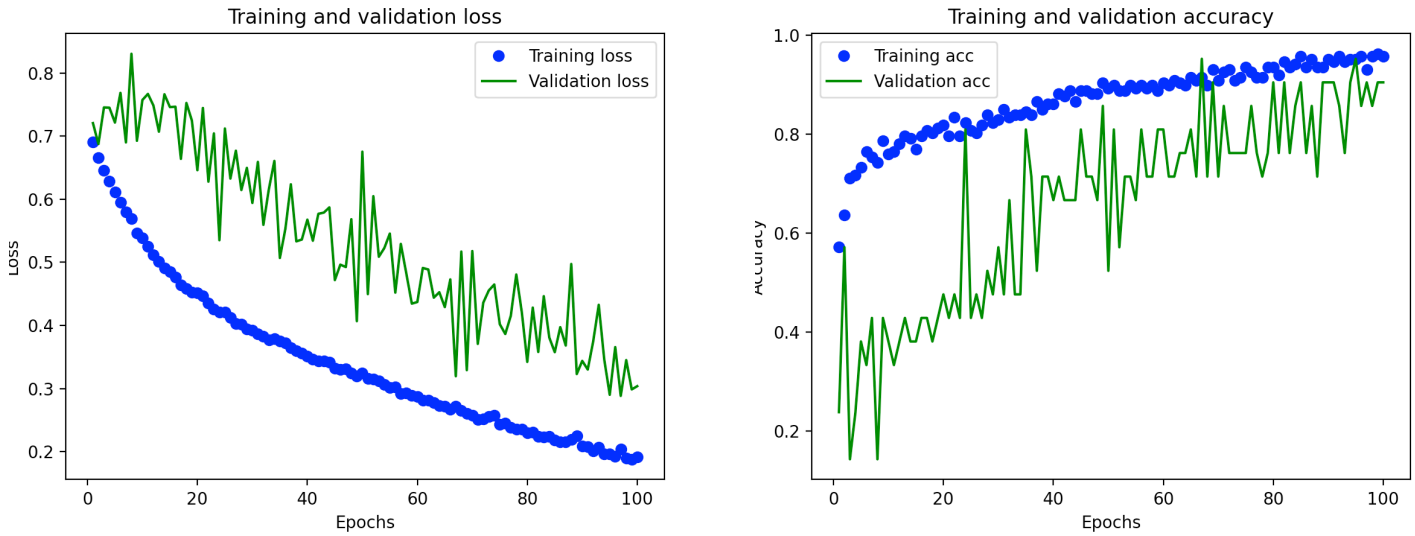


Рисунок 1 - Графики ошибок и точности ИНС_1

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.

Было уменьшено количество нейронов входного слоя. Результаты ИНС_2 представлены на рис.2

```
model = Sequential()
model.add(Dense(30, input_dim=60, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

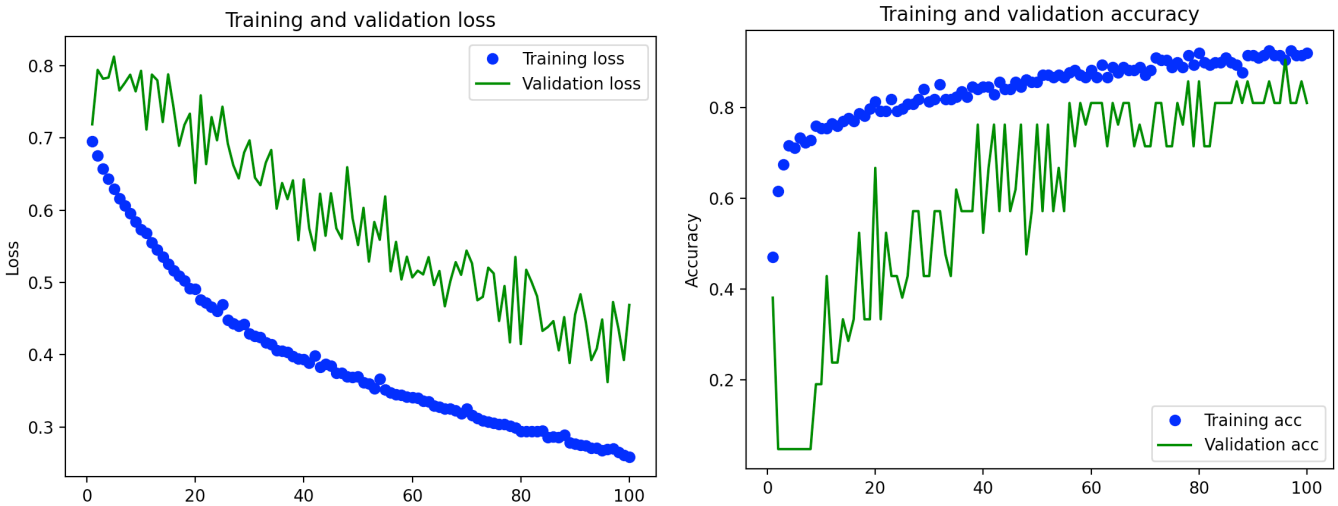


Рисунок 2 - Графики ошибок и точности ИНС_2

С изменением количества нейронов, нейросеть показала большие потери и меньшую точность.

Был добавлен еще один слой с 15 нейронами, количество нейронов во входном слое 30. Результаты ИНС_3 представлены на рис.3

```
model = Sequential()
model.add(Dense(30, input_dim=60, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

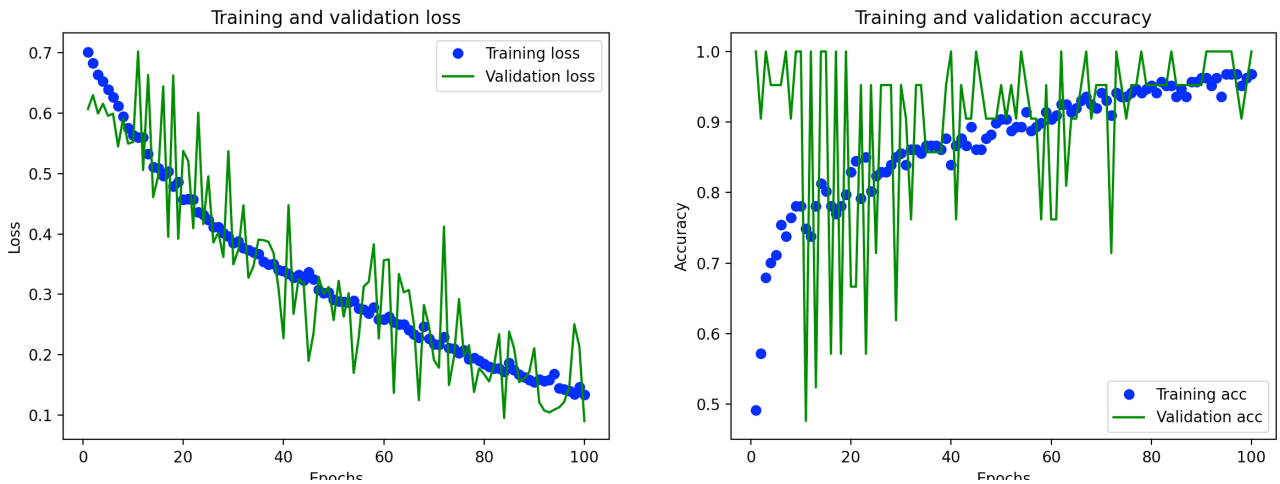


Рисунок 3 - Графики ошибок и точности ИНС_3

Количество нейронов на входном слое было увеличено до 60. Результаты ИНС_4 представлены на рис.4

```
model.add(Dense(60, input_dim=60, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

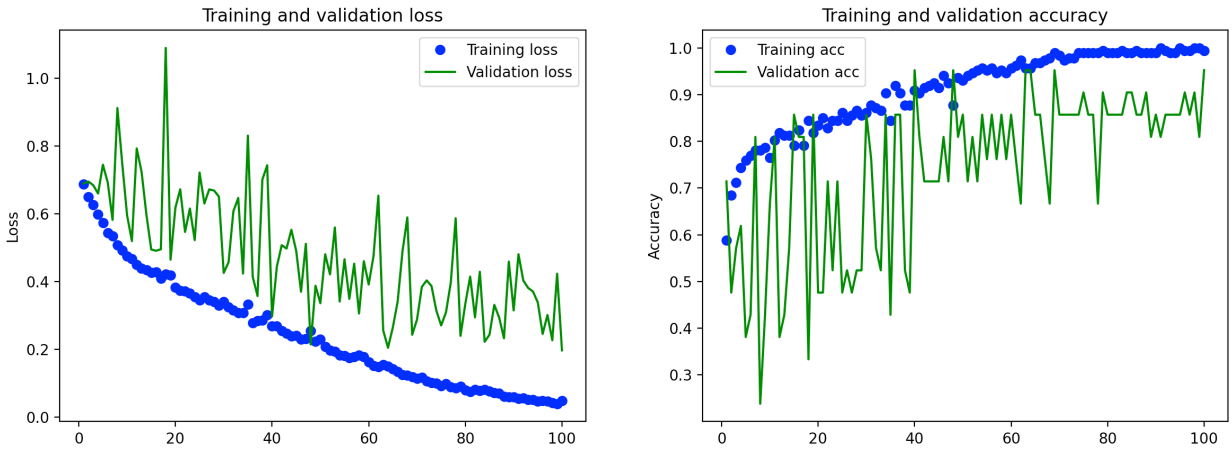


Рисунок 4 - Графики ошибок и точности ИНС_4

Точность увеличилась, а потери стали меньше, эта модель показывает результаты лучше чем 1-2 модели. Сравнивая с 3 моделью, можно сказать, что на тренировочных данных лучше 4 модель, на проверочных данных лучшие результаты показывает 3 модель.

В скрытом слое было увеличено количество нейронов с 15 до 30. Результаты запуска ИНС_5 представлены на рис.5

```
model.add(Dense(60, input_dim=60, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

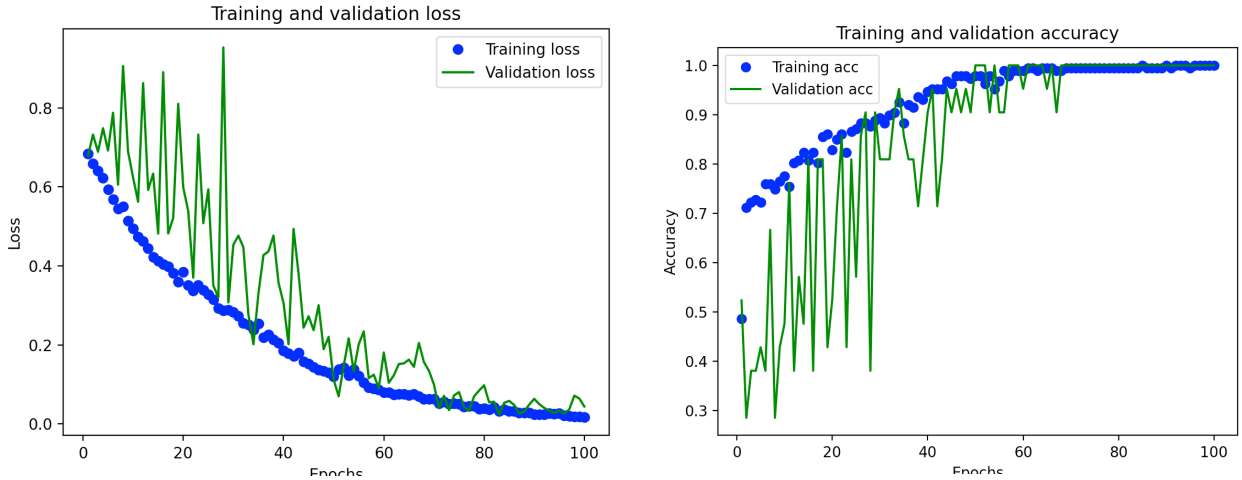


Рисунок 5 - Графики ошибок и точности ИНС_5

Данная модель показывает лучшие результаты среди всех предыдущих: она достигает высокой точности и меньших потерь, данные к концу обучения не имеют широкой амплитуды значений.

**Выводы.**

Во время выполнения лабораторной работы была реализована классификация между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей.. Были исследованы различные конфигурации ИНС, построены графики ошибок и точности в ходе обучения, а также была выбрана наилучшая модель.

## ПРИЛОЖЕНИЕ А

```
import matplotlib.pyplot as plt  #импорт модуля для графиков
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


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

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

#архитектура сети
model = Sequential()
model.add(Dense(30, input_dim=60, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)

history_dict = history.history
# график ошибки
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, 'g', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
# график точности
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, 'g', label='Validation acc')
```

```
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```