**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра математического обеспечения и применения ЭВМ**

# ОТЧЕТ

**по лабораторной работе №2**
**"Бинарная классификация отраженных сигналов радара"**
**по дисциплине «Искусственные нейронные сети»**

Студент гр. 8382 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

**Цель.**

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.

**Задание.**

- 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

**Выполнение работы.**

Работа выполнялась на базе операционной системы Windows 10 в среде разработки PyCharm.

**1. Загрузка данных и создание ИНС.**

Был скачан файл sonar.all-data и переименован в sonar.csv с исходными данными для анализа. Каждая строка данных содержит 60 параметров каждого объекта, а также буква, обозначающая, какой это объект.

В программе были подключены необходимые библиотеки, загружены данные из файла, параметры объектов были помещены в вектор X, а типы объектов в массив Y, который был приведен к категориальному виду с помощью метода transform() класса LabelEncoder. Листинг приведен ниже:

```
import pandas
import numpy
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

epochs = 100

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)
```

Далее была создана простая модель с двумя промежуточными слоями: первый имеет 60 нейронов с функцией активации Relu, второй – 1 нейрон с функцией активации Sigmoid. Значение на выходе – от 0 до 1 – определяет класс объекта, определенный ИНС.

Для анализа данных и построение графиков была использована библиотека matplotlib. Был реализован вывод 4-х графиков: ошибки во время обучения, точность во время обучения, ошибки на проверяемых данных, точность на проверяемых данных (зависимость показателей от эпох). Листинг приведен ниже:

```
loss_history = numpy.array(res.history["loss"])
val_loss_history = numpy.array(res.history["val_loss"])
accuracy_history = numpy.array(res.history["accuracy"])
val_accuracy_history = numpy.array(res.history["val_accuracy"])
history = [loss_history, accuracy_history, val_loss_history,
val_accuracy_history]
titles = ["Loss", "Accuracy", "Val Loss", "Val Accuracy"]
ylables = ["loss", "accur"]
```

```
for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.title(titles[i])
    plt.xlabel("epoch")
    plt.ylabel(ylables[i % 2])
    axes = plt.gca()
    if i % 2:
        axes.set_ylim([0, 1.1])
    else:
        axes.set_ylim([0, 3])
    plt.grid()
    plt.plot([i for i in range(epochs)], history[i])

plt.show()
```

## 2. Анализ показателей различных ИНС

Слои ИНС №1 представлены в табл. 1.

Таблица 1 – ИНС №1

| № | Ф-я активации | Кол-во нейронов |
|---|---|---|
| 1 | Relu | 60 |
| 2 | Sigmoid | 1 |

Результаты тестирования ИНС №1 представлены на рис. 1.

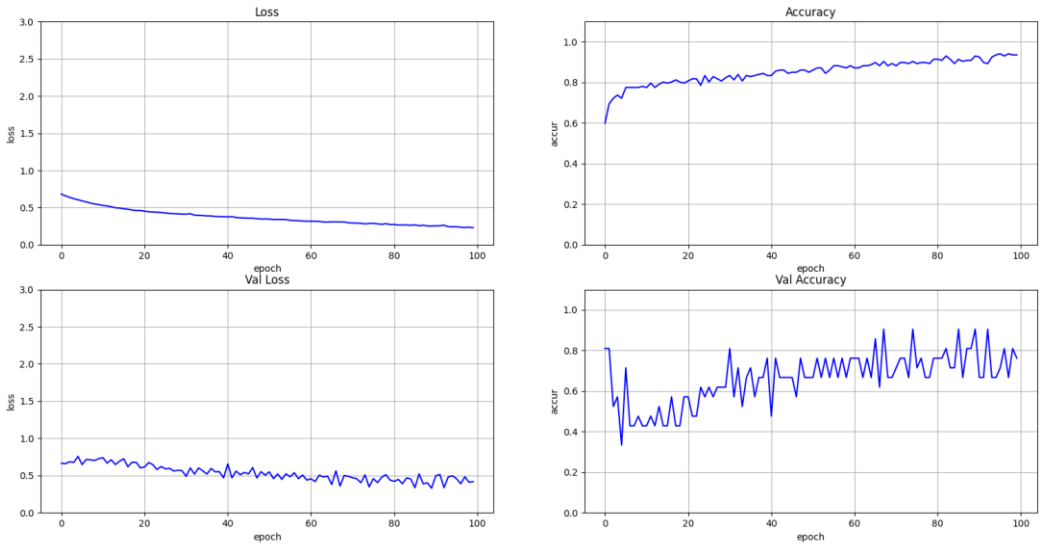


Рисунок 1 – Показатели ИНС №1

Из графика видно, что ИНС №1 не достигает высокой точности как на данных для обучения, так в особенности и на данных для проверки. Имеют место высокие значения ошибок, причем на данных для проверки после 60-й эпохи значения ошибок практически не улучшаются.

Можно сделать вывод, что ИНС №1 не подходит для решения данной задачи.

В представленном наборе данных присутствует некоторая избыточность, т.к. с разных углов описывается один и тот же сигнал. В ИНС №2 было уменьшено количество нейронов во входном слое до 30 с помощью параметра input_dim в 1 промежуточном слое. Также входные данные были исправлены таким образом, что берутся только первые 30 параметров. Слои ИНС №2 представлена в табл. 2.

Таблица 2 – ИНС №2

| №, параметры | Ф-я активации | Кол-во нейронов |
|---|---|---|
| 1, input_dim=30 | Relu | 60 |
| 2 | Sigmoid | 1 |

Результаты тестирования ИНС №1 и ИНС №2 представлены на рис. 2 (ИНС №1 – синим, ИНС №2 – красным).

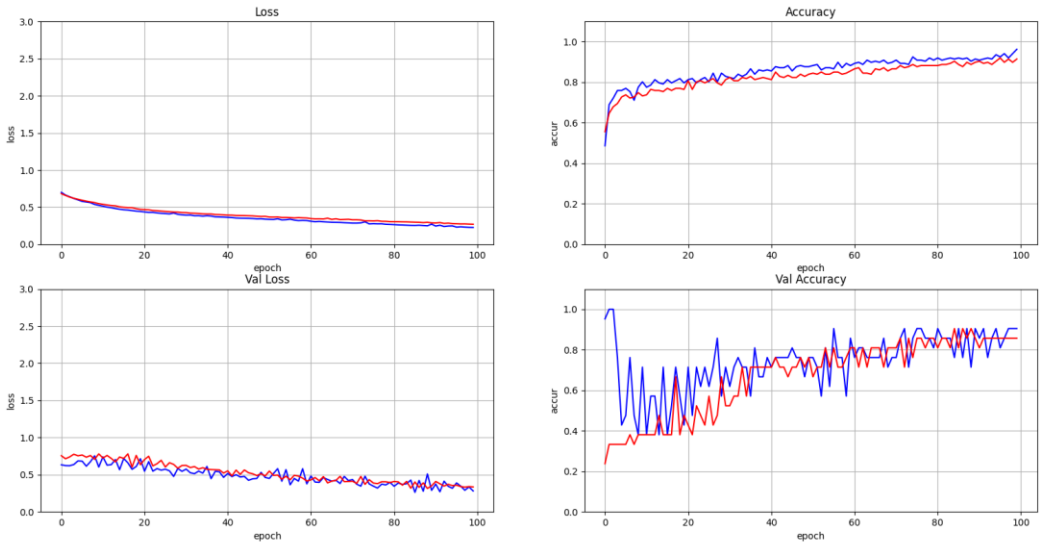


Рисунок 1 – Показатели ИНС №1 и ИНС №2

Из графиков видно, что изменение количества нейронов на входном слое практически не повлияло на показатели нейронной сети. Более того, наблюдается более стабильное обучение ИНС №2: меньше колебания точности на эпохах, точность на данных для проверки после 70-й эпохи в среднем лучше, чем у ИНС №1.

Из этого можно сделать вывод, что ввиду избыточности входных параметров, можно учитывать только их часть для обучения ИНС, то есть использовать меньшее количество нейронов во входном слое.

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.

В ИНС №3 был добавлен скрытый слой с 15 нейронами с функцией активации Relu. Слои ИНС №3 представлены в табл. 3.

Таблица 3 – ИНС №3

| №, параметры | Ф-я активации | Кол-во нейронов |
|---|---|---|
| 1, input_dim=30 | Relu | 60 |

| | | |
|---|---|---|
| 2 | Relu | 15 |
| 3 | Sigmoid | 1 |

Результаты тестирования ИНС №2 и ИНС №3 представлены на рис. 3 (ИНС №2 – синим, ИНС №3 – красным).

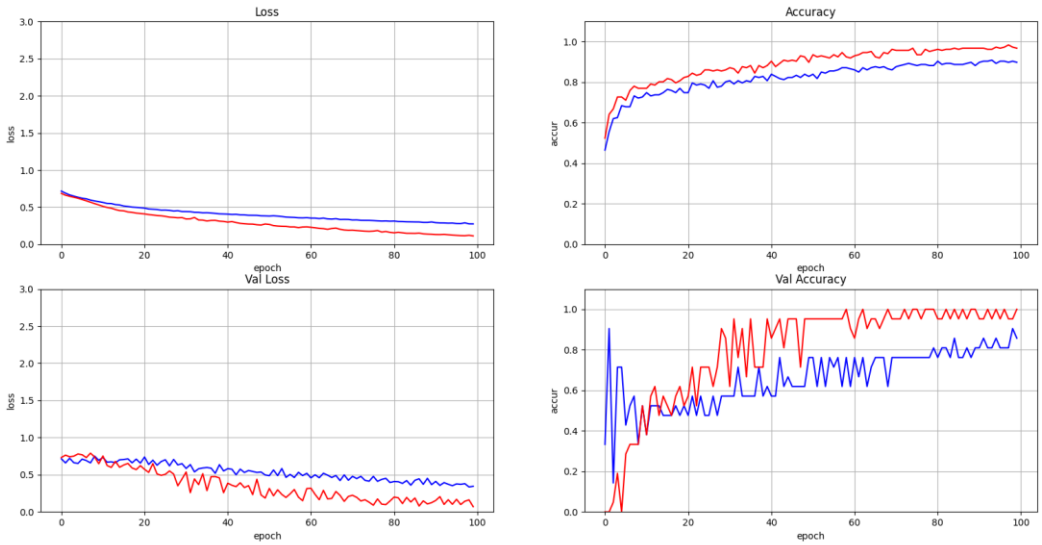


Рисунок 3 – Показатели ИНС №2 и ИНС №3

Из графиков видно, что ИНС №3 обучается быстрее и достигает значительно большей точности на данных для проверки за счет нелинейности в сети.

- ИНС №3 достигает стабильной точности в диапазоне 0.9-1 на данных для проверки на 70-й эпохе, и на ней же стабильных показателей ошибок (<0.2)
- Графики точности и ошибок на данных для обучения ИНС №2 и №3 похожи по характеру, однако ИНС №3 демонстрирует большую точность и меньшее значение ошибок на всем диапазоне эпох.

Можно сделать вывод, что добавление скрытого слоя нейронов положительно повлияло на показатели ИНС, так как это внесло нелинейность в сеть, что позволяет ей более точно распознать признаки, присущие тому или иному объекту.

В ИНС №4 был добавлен еще один скрытый слой. Слои ИНС №4 представлены в табл. 4.

Таблица 4 – ИНС №4

| № | Ф-я активации | Кол-во нейронов |
|---|---|---|
| 1, input_dim=30 | Relu | 60 |
| 2 | Relu | 15 |
| 3 | Relu | 15 |
| 4 | Sigmoid | 1 |

Результаты тестирования ИНС №3 и ИНС №4 представлены на рис. 4 (ИНС №3 – синим, ИНС №4 – красным).

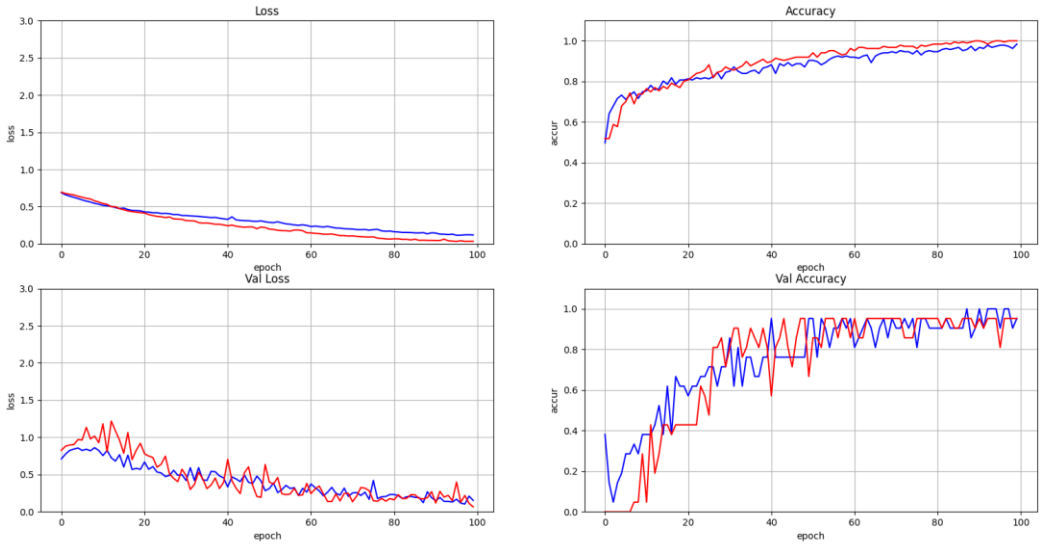


Рисунок 4 – Показатели ИНС №2 и ИНС №3

Из графика видно, что добавление еще одного слоя нейронов не привело к улучшению показателей точности и ошибок на данных для проверки. Кроме того, можно заметить, что ИНС №4 достигает точности в 100% на данных для обучения, а значит склонна к переобучению.

Можно сделать вывод, что добавление еще одного слоя нейронов в ИНС для решения поставленной задачи нецелесообразно.

### 3. Выбор ИНС, параметров обучения для данных и задачи

Итак, в результате анализа была выбрана модель ИНС №3. В табл. 5 приведена ее структура и параметры обучения.

Таблица 5 – Структура ИНС №3

| **Слои** | **Ф-я акт.** | **Кол-во нейр., параметры** |
|---|---|---|
| | Relu | 60, input_dim=30 |
| | Relu | 15 |
| | Softmax | 3 |
| **Параметры обучения** | | |
| **Validation_split** | **Batch_size** | **Epochs** |
| 0.1 | 10 | 100 |

Результаты нескольких тестирований ИНС №3 с выбранными параметрами обучения представлены на рис. 5. На графике 4 линии – 4 последовательных запуска аналогичных моделей ИНС №3 с одинаковыми параметрами тестирования.

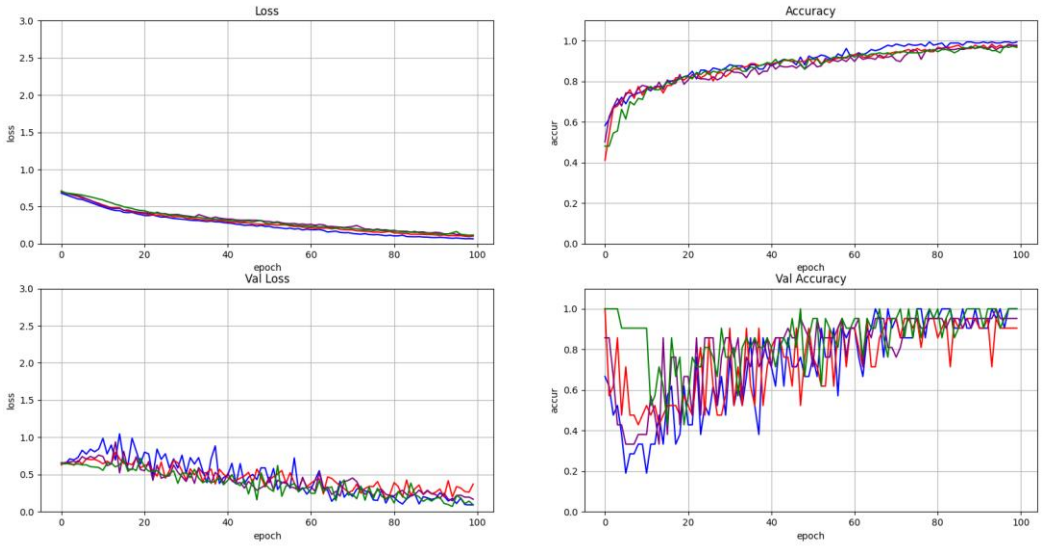


Рисунок 5 – Тестирование ИНС №3 с подобранными параметрами

В целом, показатели таковы:

- Обучение за 80 эпох. При достижении этой эпохи:

- Потери на данных для обучения: $< 0.2$
- Точность на данных для обучения: $> 0.95$
- Потери на данных для проверки: $0.25$ в среднем
- Точность на данных для проверки: $0.95$ в среднем

В приложении А приведен код программы, в котором реализована модель ИНС №3.

**Выводы.**

В ходе выполнения лабораторной работы была построена модель классификации между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Экспериментально было изучено, как различное количество нейронов во входном слое влияет на обучение модели, а также, как нелинейность модели при использовании скрытых слоев улучшает показатели ИНС при обучении. В результате была выбрана модель, которая показывает достаточно высокую точность и небольшие ошибки после 80 эпох обучения.

## ПРИЛОЖЕНИЕ А.

### Исходный код программы. Файл lr2.py.

```
import pandas
import numpy
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

epochs = 100

dataframe = pandas.read_csv("sonar.csv", header=None)
dataset = dataframe.values
X = dataset[:, 0:30].astype(float)
Y = dataset[:, 60]

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

model = Sequential()
model.add(Dense(60, activation='relu', input_dim=30))
model.add(Dense(15, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
res = model.fit(X, encoded_Y, epochs=epochs, batch_size=10,
validation_split=0.1)

loss_history = numpy.array(res.history["loss"])
val_loss_history = numpy.array(res.history["val_loss"])
accuracy_history = numpy.array(res.history["accuracy"])
val_accuracy_history = numpy.array(res.history["val_accuracy"])

# graph maker
history = [loss_history, accuracy_history, val_loss_history,
val_accuracy_history]
titles = ["Loss", "Accuracy", "Val Loss", "Val Accuracy"]
ylables = ["loss", "accur"]

for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.title(titles[i])
    plt.xlabel("epoch")
    plt.ylabel(ylables[i % 2])
```

```
    axes = plt.gca()
    if i % 2:
        axes.set_ylim([0, 1.1])
    else:
        axes.set_ylim([0, 3])
    plt.grid()
    plt.plot([i for i in range(epochs)], history[i], color="blue")
plt.show()
```