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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр. 8383 _____ Сосновский Д. Н.

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

Ход работы.

1. Загрузка файла с данными

Согласно методическим указаниям был скачан файл с данными sonar.all-data и переименован в файл sonar.csv

2. Написание программы

Была написана программа, которая создаёт модель, обучает её, и рисует графики ошибок и точности. Исходный код программы приведён в приложении. Графики ошибок и точности приведены на рис. 1 и рис.2

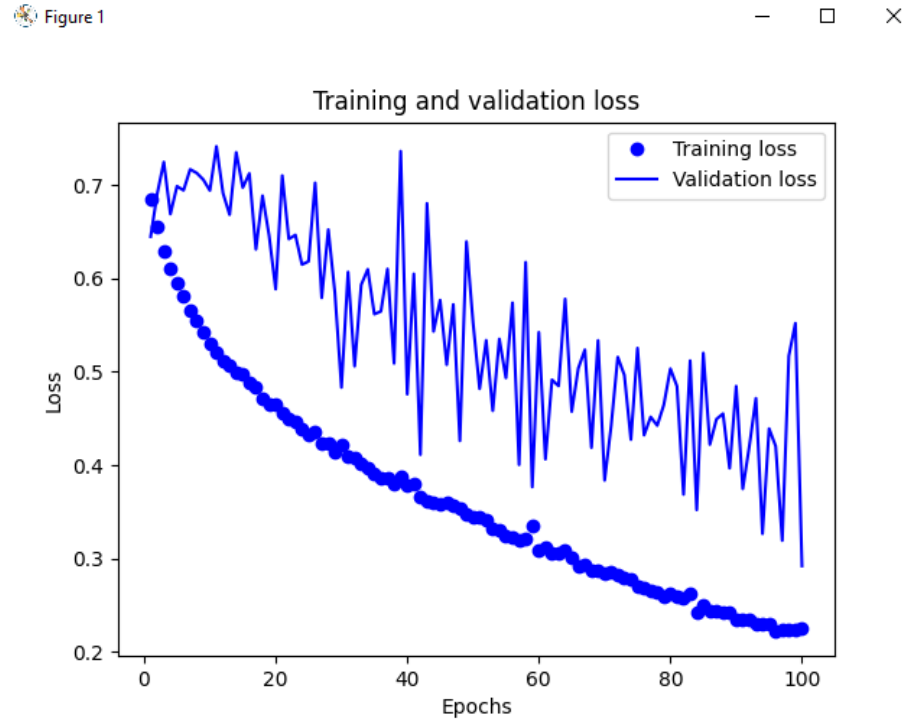

Рисунок 1 - тренировочные и проверочные ошибки

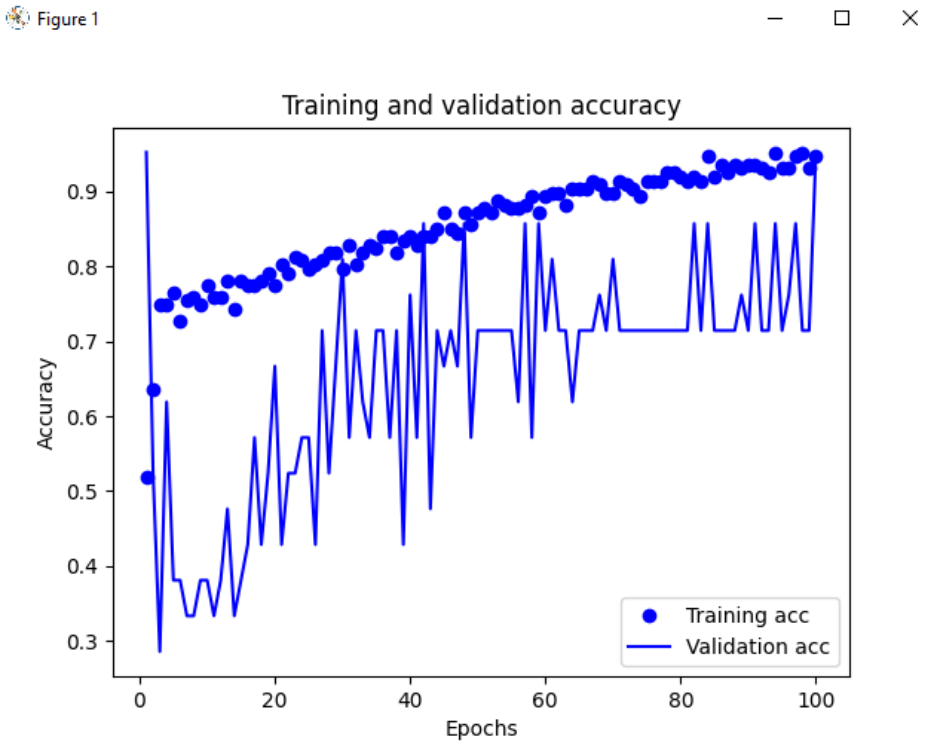

Рисунок 2 - тренировочная и проверочная точность

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.

Изменим размер входного слоя в два раза и сравним с результатами первоначальной архитектуры. Код изменённой архитектуры сети приведён ниже.

Листинг 1

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

Графики ошибок и точности приведены на рис. 3 и рис.4

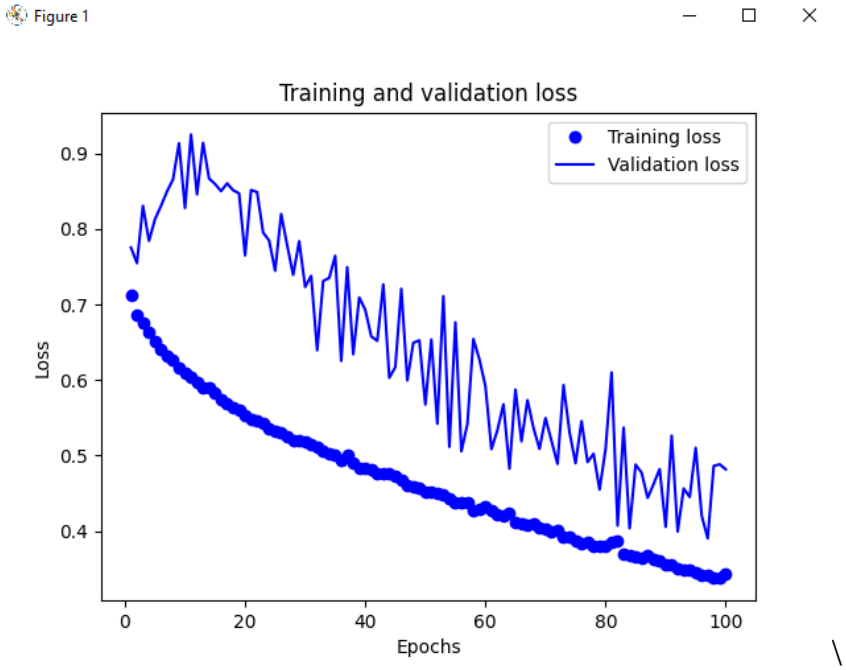

Рисунок 3 - тренировочные и проверочные ошибки

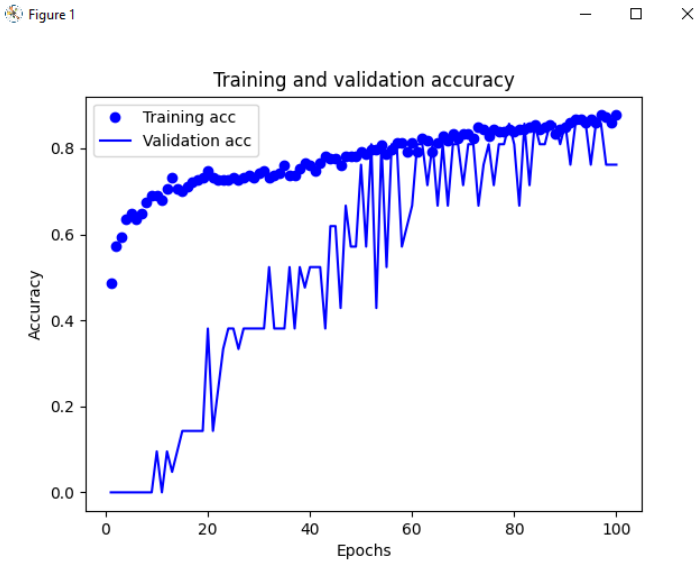

Рисунок 4 - тренировочная и проверочная точность

Исходя из полученных результатов видно, что показатели ошибок выросли, а точность понизилась.

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 слой в архитектуру сети с 15 нейронами. Листинг изменённой архитектуры приведён ниже.

Листинг 2

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

Графики ошибок и точности приведены на рис. 5 и рис. 6

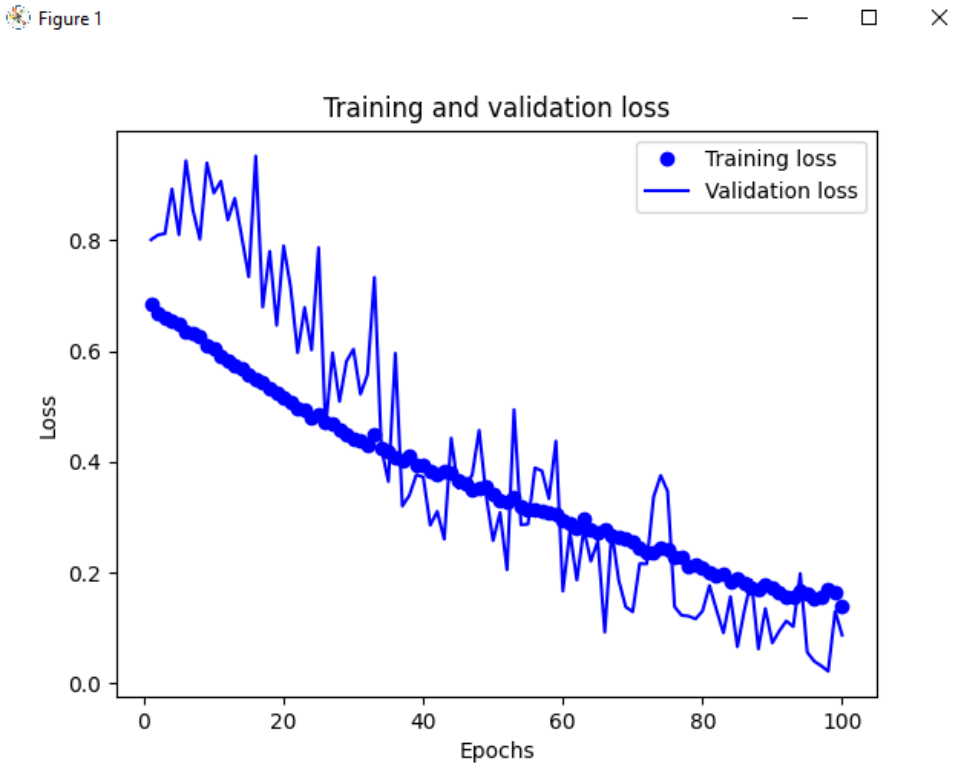

Рисунок 5 - тренировочные и проверочные ошибки

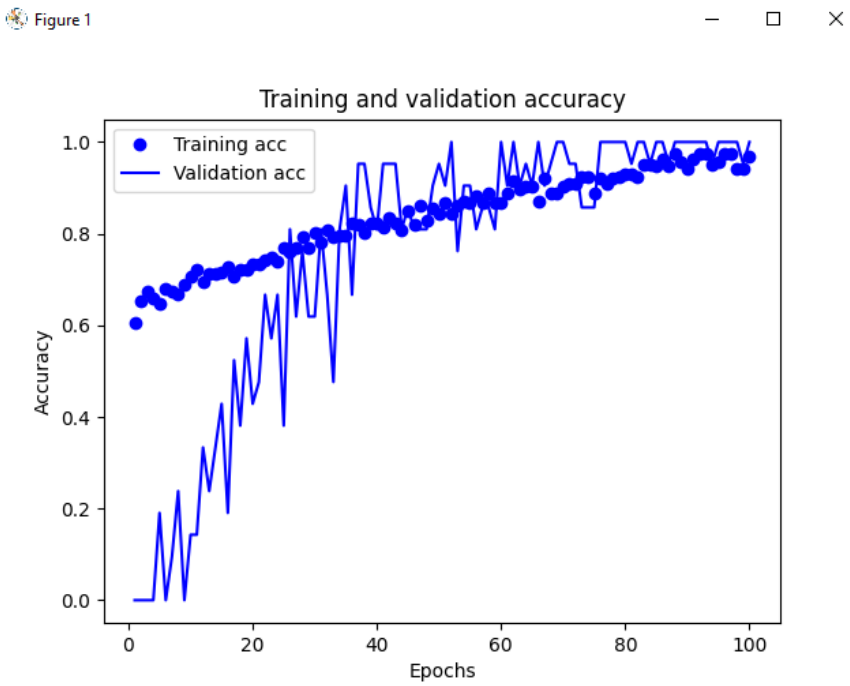

Рисунок 6 - тренировочная и проверочная точность

Из графиков видно, что ошибки уменьшились, а точность снова возросла. Таким образом, последняя модель показала наилучший результат.

Вывод

В ходе выполнения данной лабораторной работы была реализована классификация между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Были исследованы различные параметры построения архитектуры ИНС.

ПРИЛОЖЕНИЕ

Листинг – исходный код программы

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

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)

# Базовая архитектура сети
model = Sequential()
model.add(Dense(60, input_dim=60, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Инициализация параметров обучения
model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

# Обучение сети
H = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)

# Построение графиков
loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)

# Построение графика ошибки
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Построение графика точности
plt.clf()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```