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. 8383 _____ Бабенко Н.С.

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

Выполнение работы.

Был скачан файл sonar.all-data и переименован в sonar.csv.

Первая нейронная сеть имеет промежуточный слой с ф. активации Relu и 60 нейронами, а также слой с 1 нейроном и ф. активации Sigmoid. Ниже приведены графики тестирования модели.

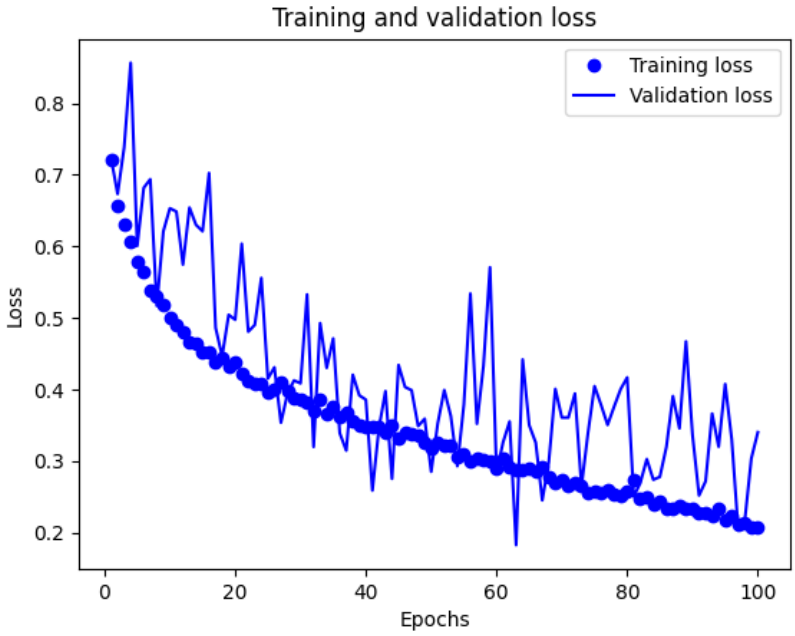

Рисунок 1 – Тестирование первой нейронной сети, ошибки

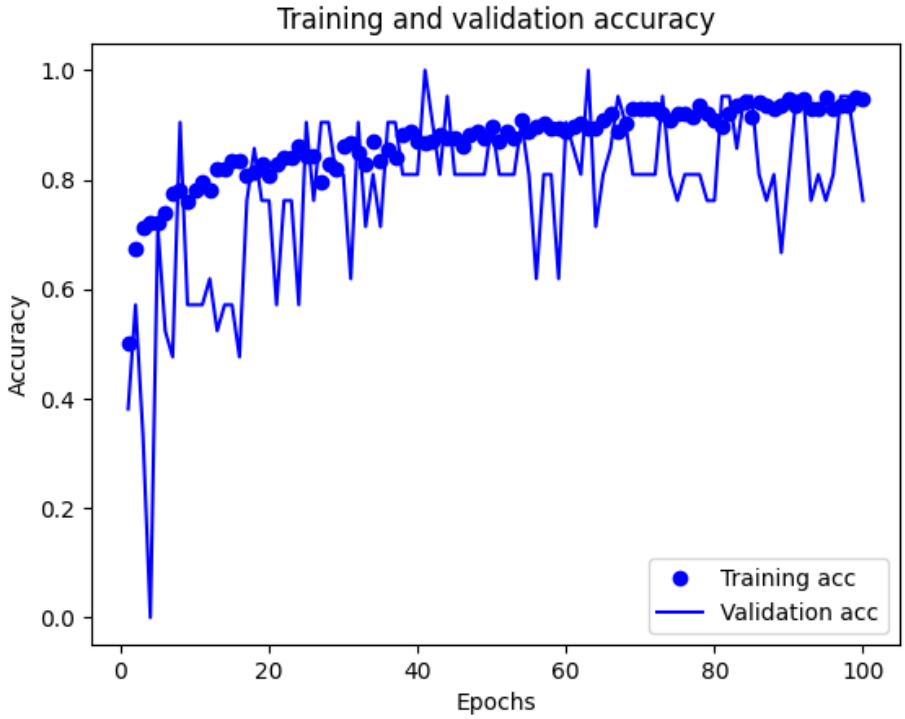

Рисунок 2 – Тестирование первой нейронной сети, точность

В наборе данных присутствует некоторая избыточность, т.к. с разных углов описывается один и тот же сигнал. В нейронной сети число нейронов во входном слое было изменено до 30 параметром input_dim. Результаты тестирования сети представлены ниже.

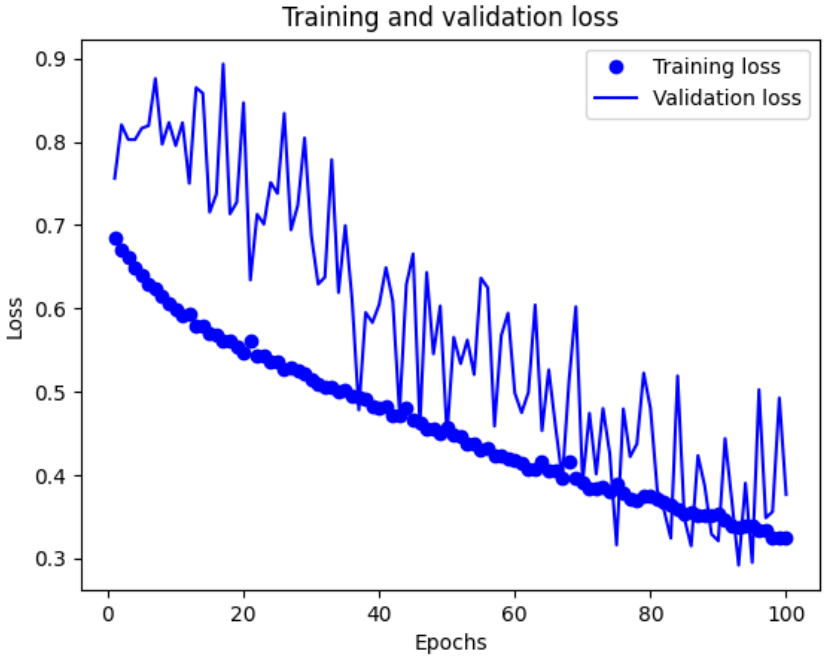

Рисунок 3 – Тестирование второй нейронной сети, ошибки

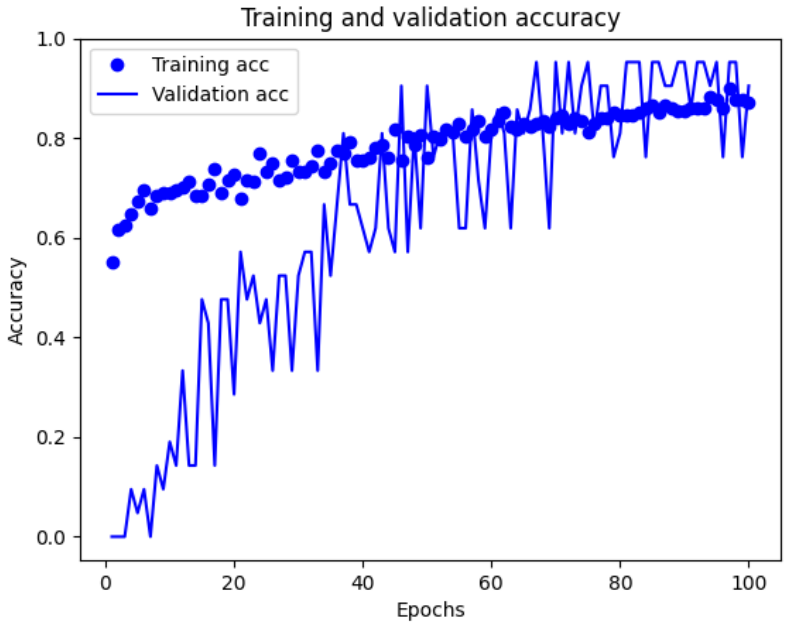

Рисунок 4 – Тестирование второй нейронной сети, точность

При уменьшении числа параметров на входе нейронной сети привело к небольшому увеличению ошибок, точность также возрастает медленнее, что можно списать на не самые удачные веса. Тем не менее, итоговая точность даже выше, чем в случае с 60 параметрами.

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.

Был добавлен еще один слой после первого скрытого – 15 нейронов с функцией активации Relu. Результаты тестирования третьей нейронной сети представлены ниже.

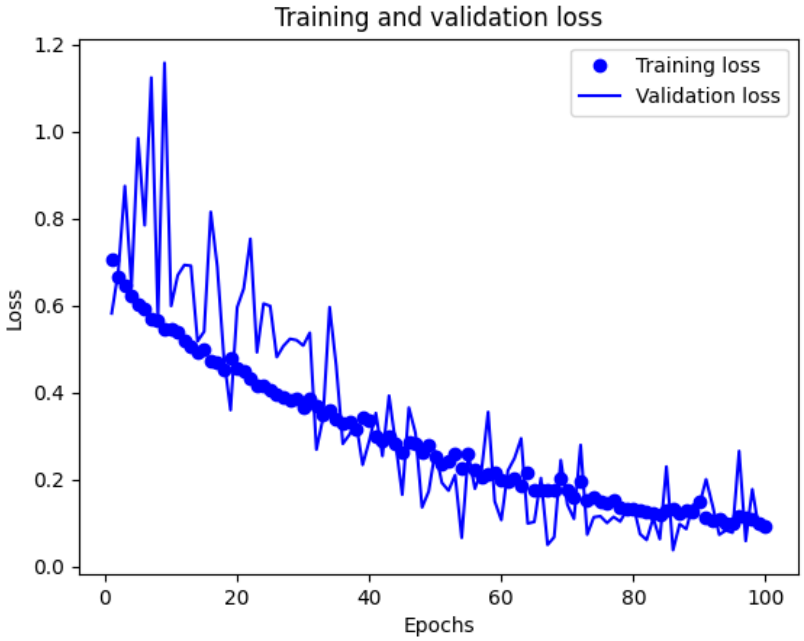

Рисунок 5 – Тестирование третьей нейронной сети, ошибки

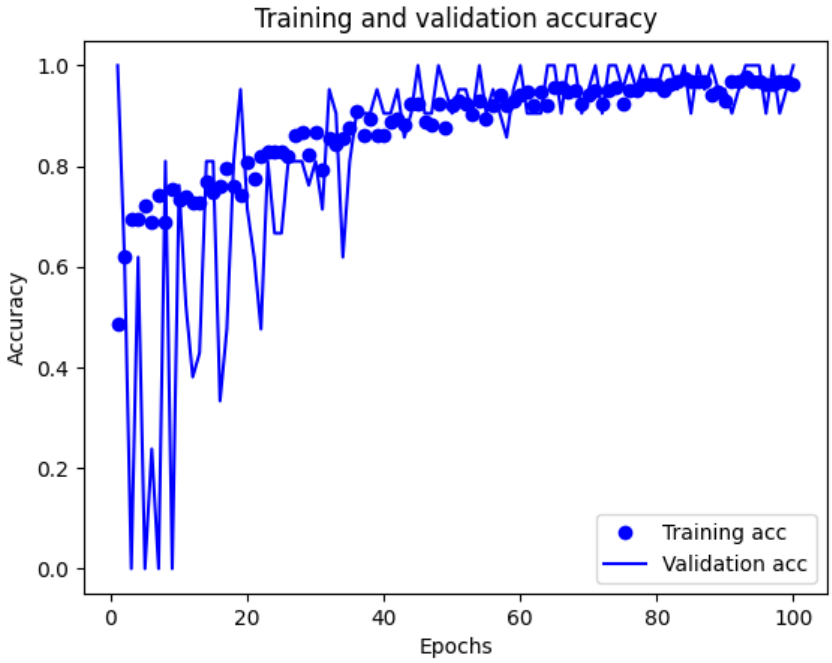

Рисунок 6 – Тестирование третьей нейронной сети, точность

Из графиков видно, что добавление еще одного слоя значительно уменьшило ошибки, повысило точность. Также видно, что точность 1 на данных для проверки достигается уже на 45 эпохе. Данную нейросеть можно считать наиболее подходящей для решения поставленной задачи. Спустя 100 эпох ошибки на всех данных < 0.1. Точность при обучении 0.96, при проверке 1.

Выводы.

В ходе выполнения лабораторной работы была построена модель классификации между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей, изучено влияние числа слоев, нейронов на входном слое на точность нейронной сети. В результате было выяснено, что в некоторых задачах возможно частично выбирать входные параметры и это не помешает классификации, а также, что добавление одного скрытого слоя позволило сети лучше справляться с задачей.

ПРИЛОЖЕНИЕ А.

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

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'])
H    =    model.fit(X,    encoded_Y,    epochs=epochs,    batch_size=5,
validation_split=0.1)

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()
```