МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр.8382 _____ Фильцин И.В.

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.

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

Ход работы

Рассмотрим первую архитектуру ИНС. Первый слой состоит из 60 нейронов, функция активации - relu, второй слой из 1 нейрона, функция активации - sigmoid. Результаты приведены на рис. 1 и рис. 2.

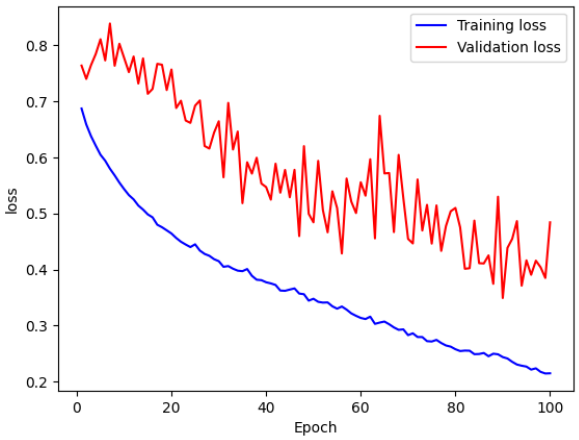

Рис. 1: "Арх.1"

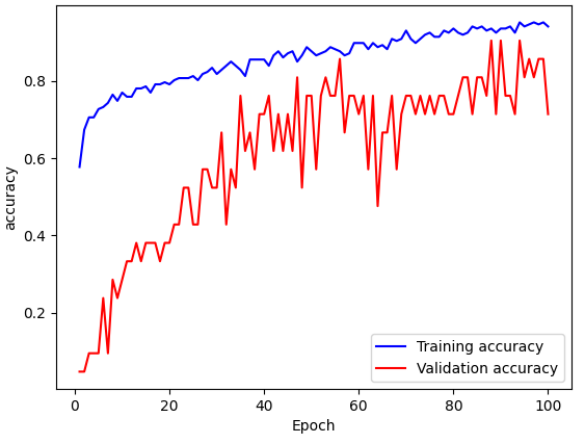

Рис. 2: "Арх.1"

На графиках видно, что сеть имеет не очень большую точность на валидационных данных. Можно заметить, что потери на валидационных данных также имеют высокие значения.

В представленном наборе данных присутствует некоторая избытычность, 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.

Уменьшим рахмер входного слоя в два раза и сравним результаты с 1ой архитектурой.

Вторая архитектура ИНС. Первый слой состоит из 30 нейронов, функция активации - relu, второй слой из 1 нейрона, функция активации - sigmoid. Результаты приведены на рис. 3 и рис. 4.

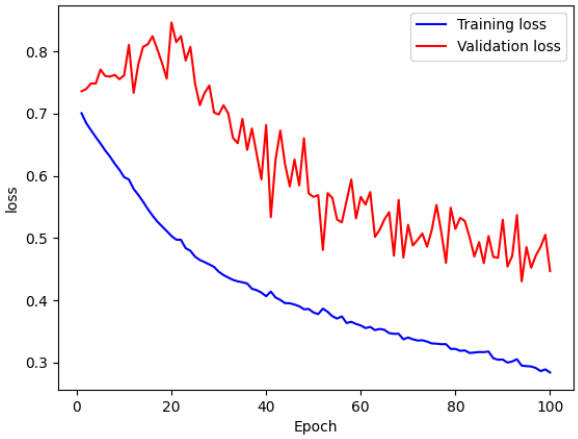

Рис. 3: "Арх.2"

На графиках видно, что сеть начала иметь еще меньшую точность на валидационном множестве.

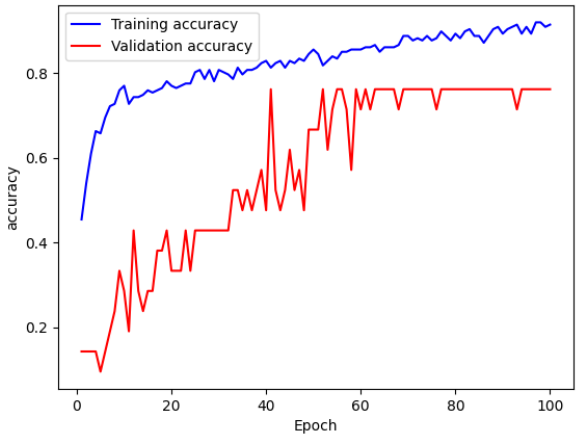

Рис. 4: "Арх.2"

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. Рассмотрим 3ью архитектуру ИНС. Первый слой состоит из 60 нейронов, функция активации - relu, второй слой из 15 нейронов, функция активации - relu, третий слой из 1 нейрона, функция активации - sigmoid.

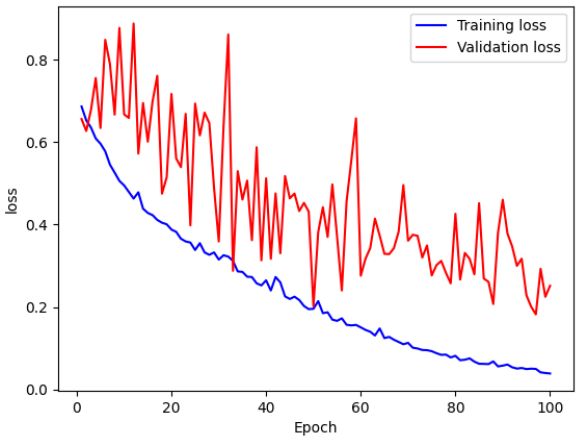

Рис. 5: "Арх.3"

На графиках видно, что точность сети на валидационном наборе стала лучше (в среднем 0.95).

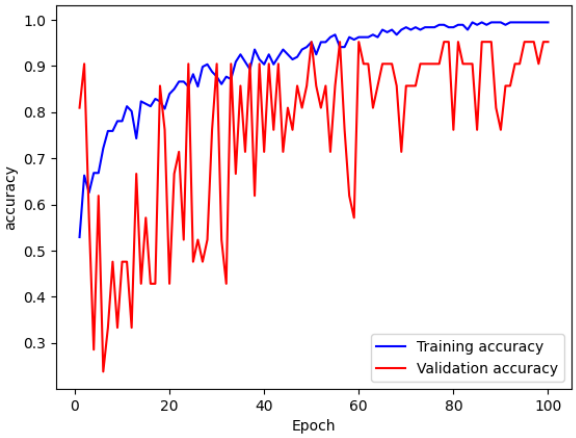

Рис. 6: "Арх.3"

Вывод

В ходе лабораторной работы была реализована классификация между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Были исследованы различных архитектуры, проведен анализ результатов.

Приложение А.

Исходный код

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


def draw_plot(xrange, training_data, validation_data, label):
    plt.plot(xrange, training_data, 'b', label='Training {}'.format(label))
    plt.plot(xrange, validation_data, 'r', label='Validation {}'.format(label))
    plt.xlabel('Epoch')
    plt.ylabel(label)
    plt.legend()
    plt.show()
    plt.clf()


def draw_result(history, nepochs):
    epochs = range(1, nepochs + 1)
    draw_plot(epochs, history['loss'], history['val_loss'], 'loss')
    draw_plot(epochs, history['accuracy'], history['val_accuracy'], 'accuracy')


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)

model = Sequential()
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'])
h = model.fit(X, encoded_Y, epochs=100, batch_size=10, validation_split=0.1)

draw_result(h.history, 100)
```