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. 8382 _____ Янкин Д.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.

Постановка 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

Выполнение работы.

Данные для анализа считываются из файла sonar.csv, разделяются на входные параметры X и выход Y. Задана модель нейронной сети с одним скрытым слоев из 60 нейронов, обрабатывающая 60 входных параметров. Задаются параметры обучения и происходит обучение в течение 100 эпох.

```
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(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)
```

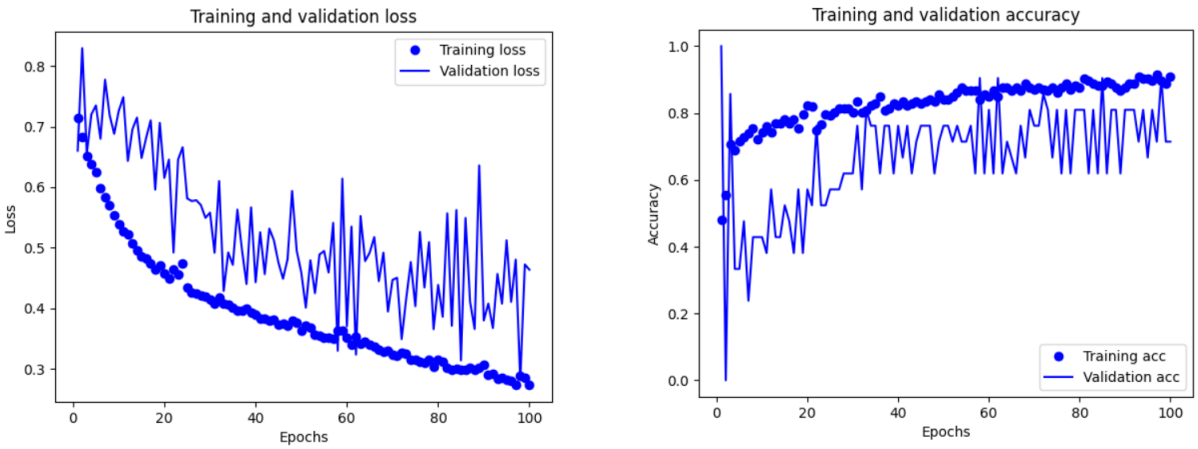

Рисунок 1. Ошибки и точточность изначальной сети

Показываемая сетью точность недостаточно высока, ошибки слишком большие.

Так как во входных данных может присутствовать избыточность, имеет смысл уменьшить количество обрабатываемых параметров. Размер входного слоя уменьшен в два раза до 30.

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

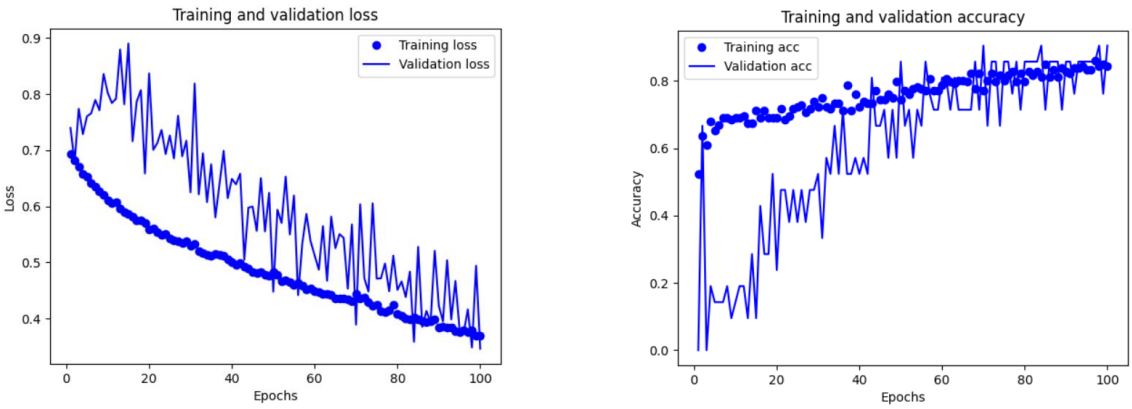

Рисунок 2. Ошибки и точность сети после уменьшения количества обрабатываемых параметров в два раза

Уменьшение числа входные параметров сказалось положительно: точность нейронной сети несколько возросла, а колебания ошибок и точности на графиках уменьшились, поэтому можно сделать вывод, что избыточность во входных данных действительно существовала.

Добавлен еще один скрытый слой с 15 нейронами, который позволит находит сложные закономерности в комбинации входных данных.

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

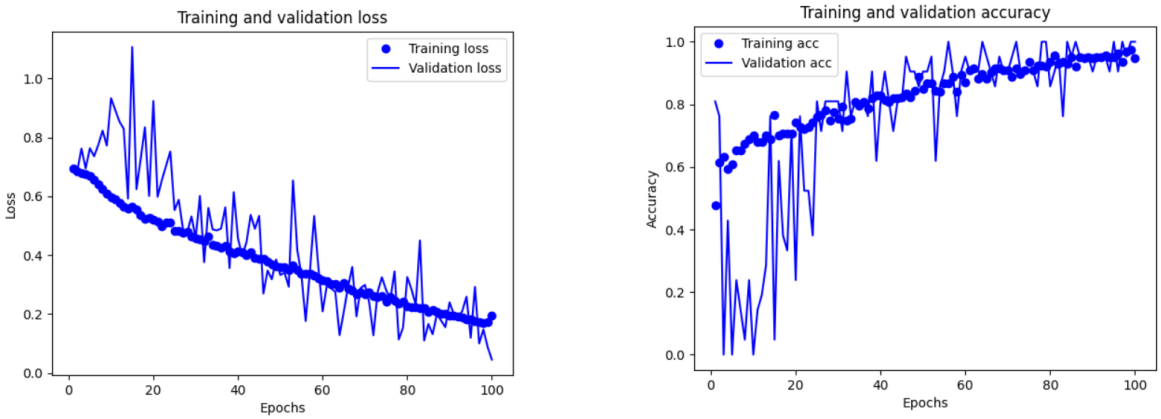

Рисунок 3. Ошибки и точность сети после добавления скрытого слоя с 15 нейронами

После добавления нового скрытого слоя точность обучаемой сети возросла, а ошибки уменьшились.

Итоговая сеть показывает хорошие результаты: точность >90% на обучающих и тестовых данных, ошибки <0.2 на обучающих и тестовых.

Выводы.

В ходе лабораторной работы была построена модель классификации между камнями и металлическими цилиндрами на основе данных об отражении сигналов радара от поверхностей. Было изучено влияние количества нейронов на входном слое и количества скрытых слоев на результат обучения.

ПРИЛОЖЕНИЕ А

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[:, 0:30].astype(float)
Y = dataset[:, 60]

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

model = Sequential()
model.add(Dense(60, input_dim=30, 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)

history_dict = history.history

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, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

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, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()
plt.show()
```