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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искуссенные нейронные сети»

Тема: «Бинарная классификация отраженных сигналов радара»

Студент гр. 7382 _____ Находько А.Ю.

Преподаватель _____ Жукова Н.В.

Санкт-Петербург

2020

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.

Задачи работы.

•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 работы.

1.Изучить влияние кол-ва нейронов на слое на результат обучения модели.
2.Изучить влияние кол-ва слоев на результат обучения модели
3.Построить графики ошибки и точности в ходе обучения
4.Провести сравнение полученных сетей, объяснить результат

Основные теоретические положения.

Искусственные нейронные сети - совокупность моделей, которые представляют собой сеть элементов - искусственных нейронов, связанных между собой синаптическими соединениями.

Нейронные сети используются как среда, в которой осуществляется адаптивная настройка параметров дискриминантных функций. Настройка происходит при последовательном предъявлении обучающих выборок образов из разных классов. Обучение - такой выбор параметров нейронной сети, при котором сеть лучше всего справляется с поставленной проблемой.

Нейрон — элемент, преобразующий входной сигнал по функции.

Сумматор — элемент, осуществляющий суммирование сигналов поступающих на его вход.

Синапс — элемент, осуществляющий линейную передачу сигнала.

Экспериментальные результаты.

В приложении А представлен код созданной и обученной нейронной сети в соответствии с требуемыми к лабораторной работе условиями.

Эксперимент 1.

В соответствии с заданием лабораторной работы, был уменьшен размер входного слоя в два раза и полученные данные сравнил с результатами первоначальной архитектуры.

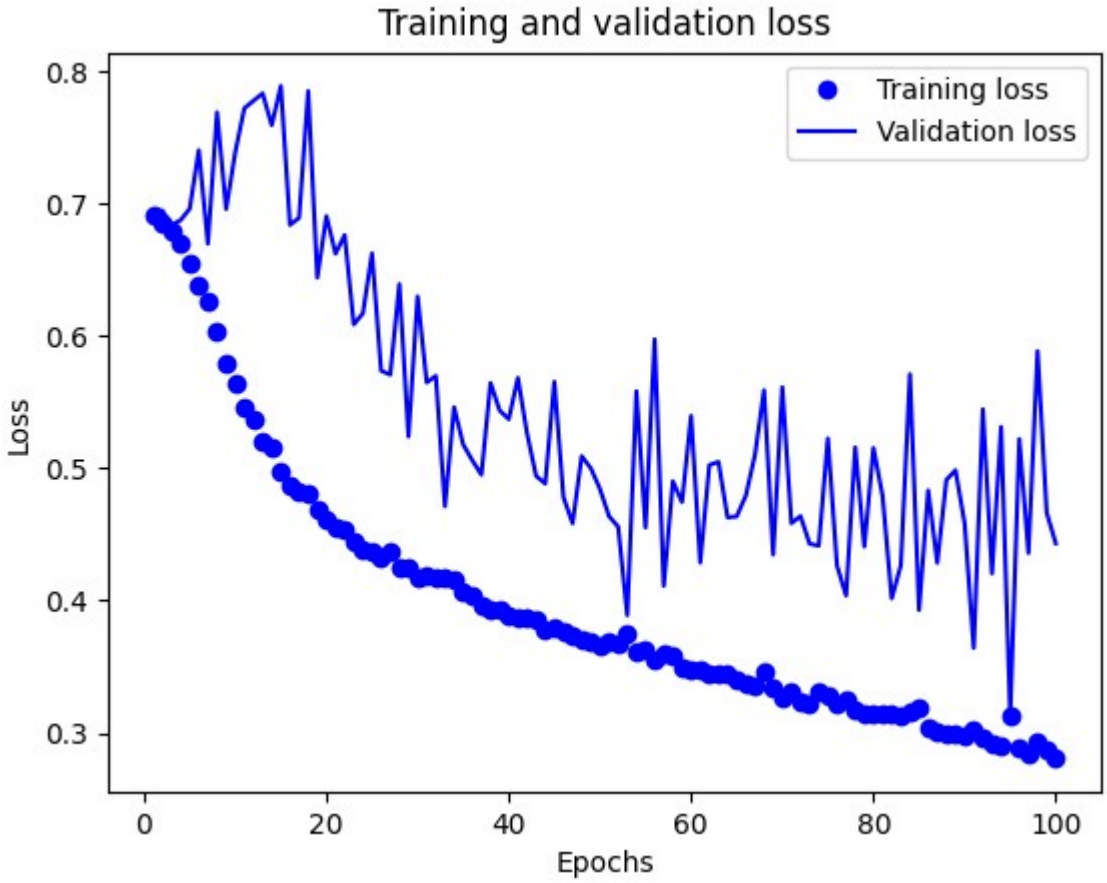

Рисунок 1 — График потерь при первоначальных данных

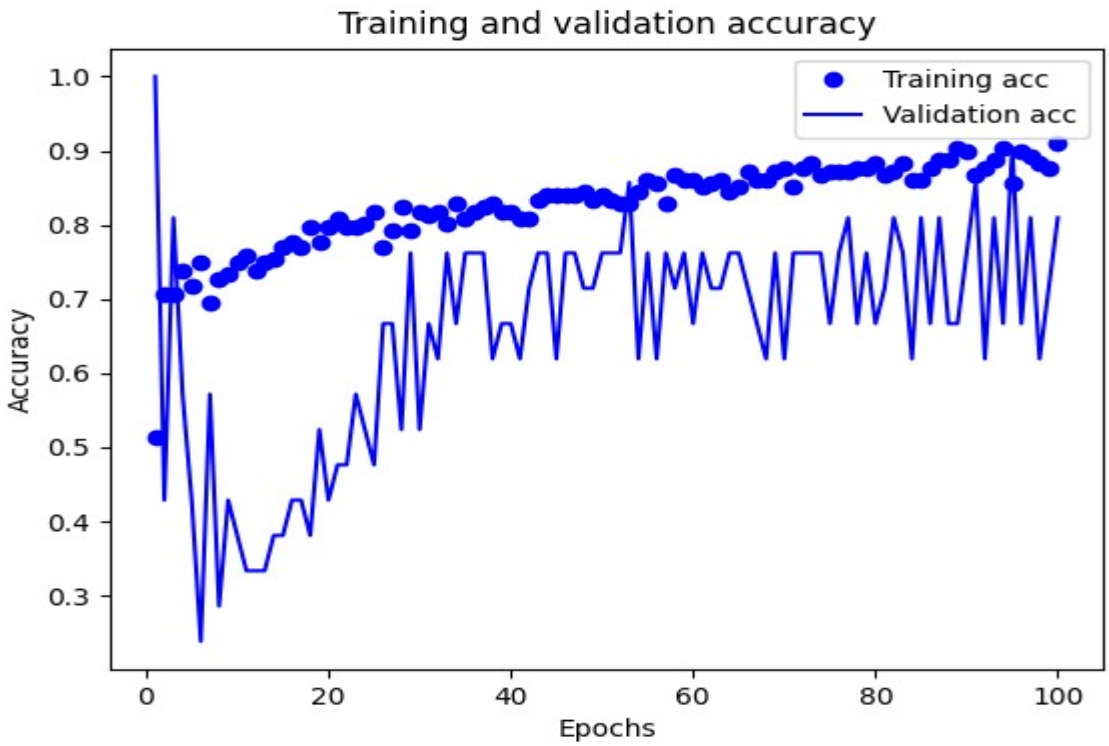

Рисунок 2 — График точности модели при первоначальных данных

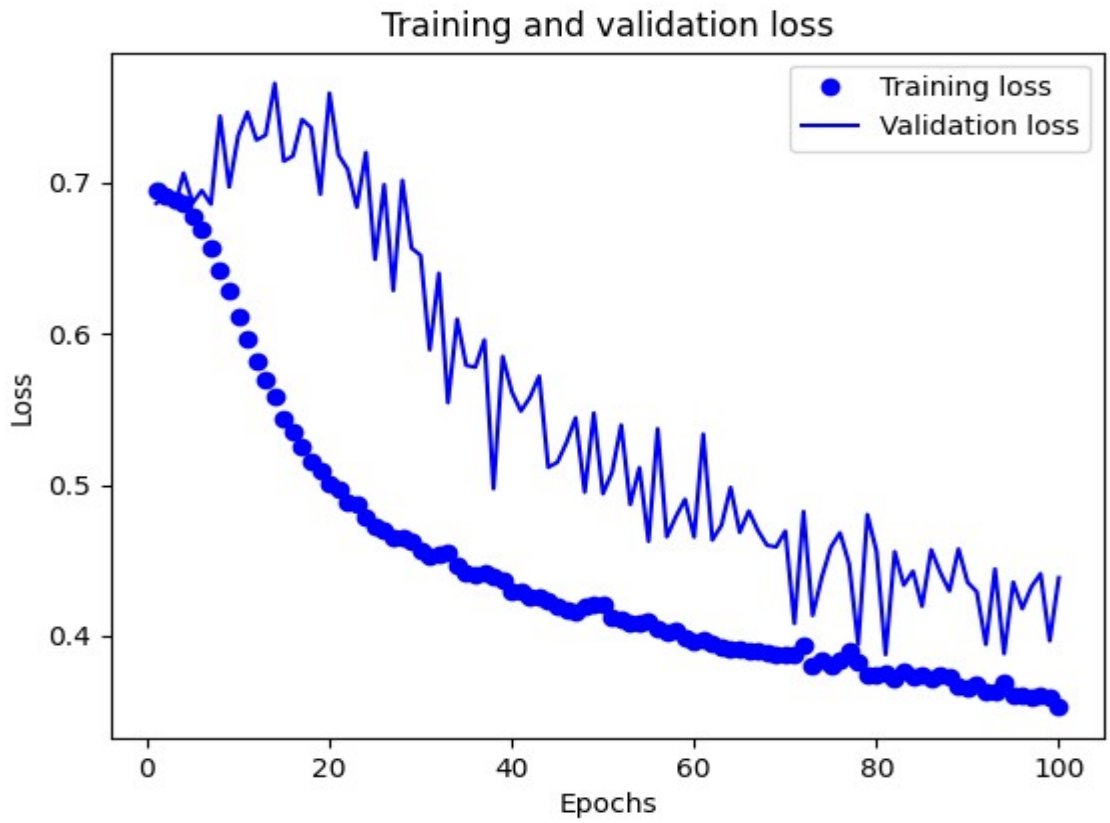

Рисунок 3 — График потерь при уменьшен размер входного слоя в 2 раза

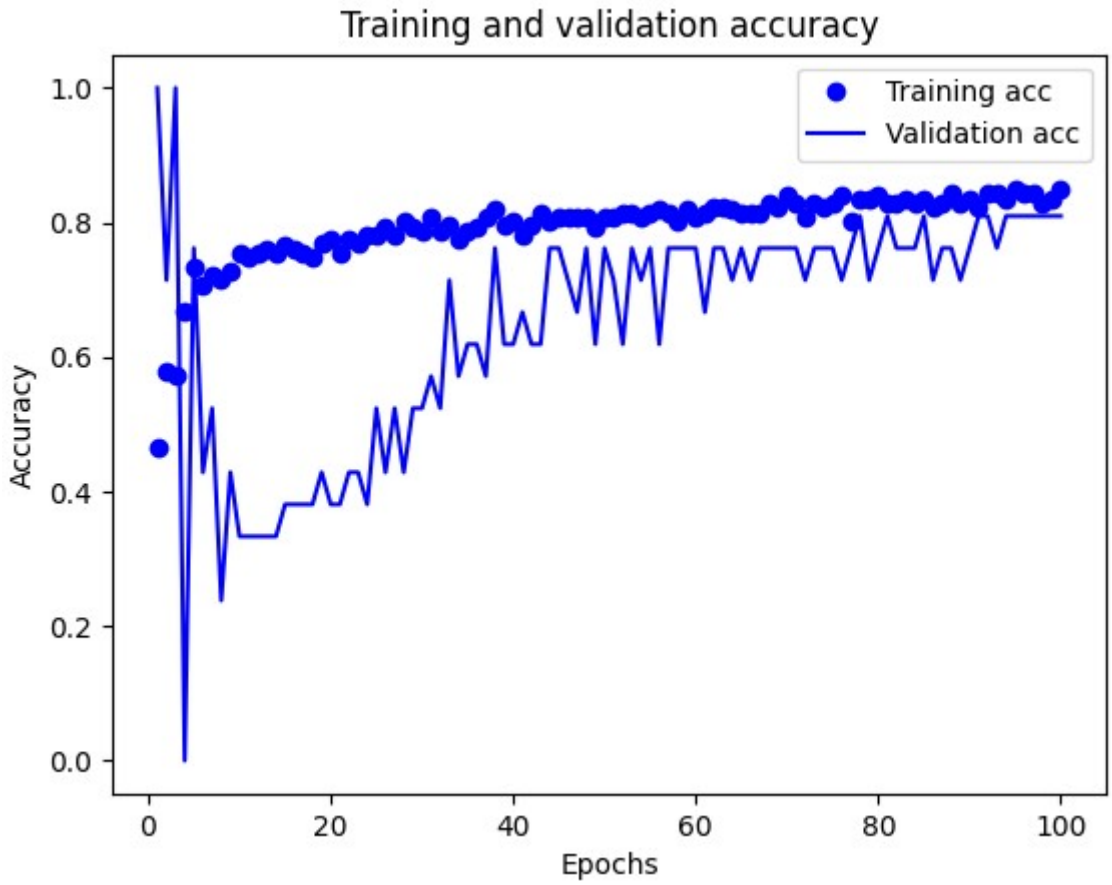

Рисунок 4 — График точности модели при уменьшен размер входного слоя в 2 раза

Заметим, что при уменьшении количества нейронов на 1 слое, был получен результат, сходимость которого примерно такая же, как в исходной модели. Поэтому установим, что в исходных данных было излишнее количество нейронов на первом слое.

Эксперимент 2.

Добавим промежуточный (скрытый) слой Dense в архитектуру сети с 15 нейронами и проанализируем результаты.

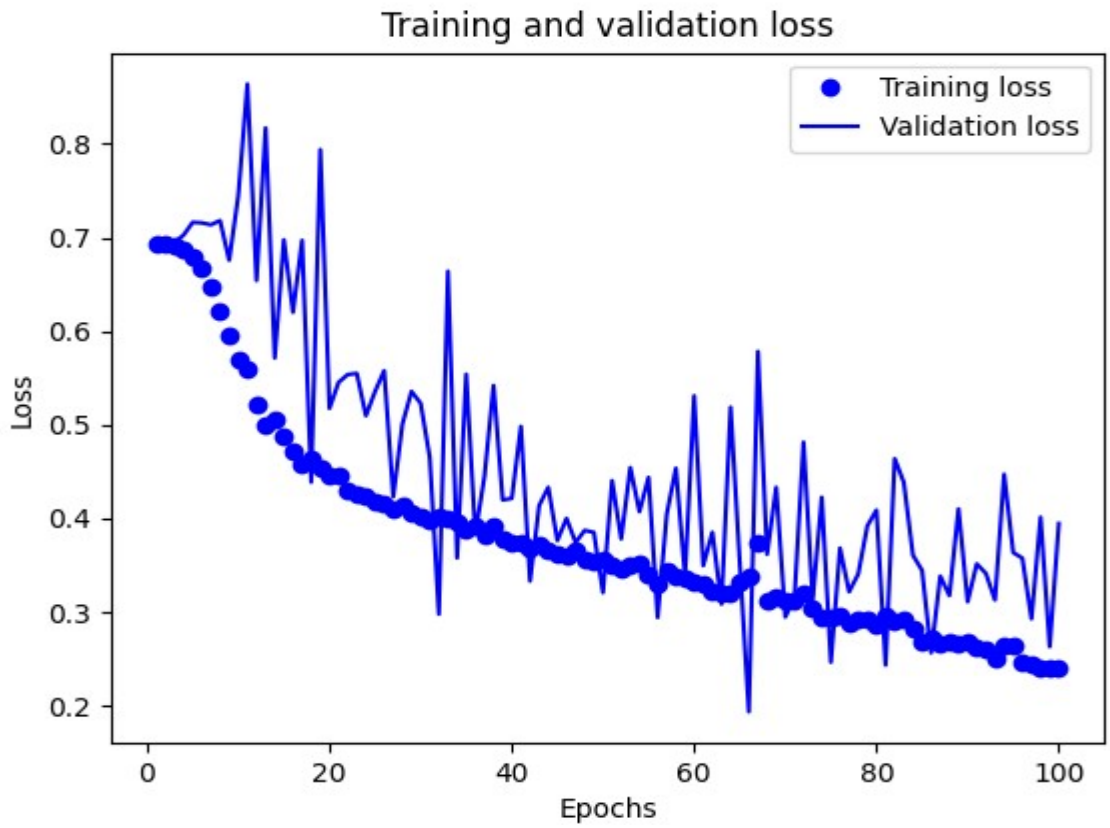

Рисунок 5 — График потерь модели при добавлении промежуточного слоя

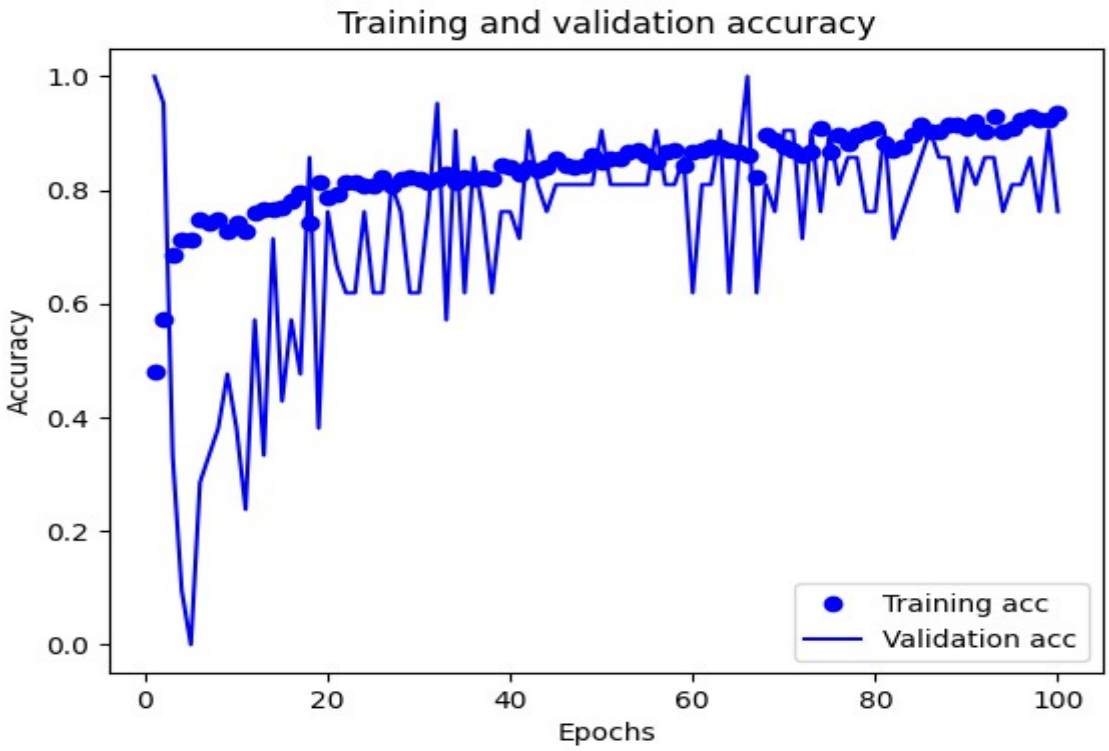

Рисунок 6 — График точности модели при добавлении промежуточного слоя

При рассмотрении графиков на рисунках 5, 6 придём к выводу что ошибка стала меньше и точность возросла.

Выводы.

В ходе выполнения лабораторной работы было изучено влияние кол-ва нейронов на слое на результат обучения модели, влияние кол-ва слоев на результат обучения модели. Также были Построены графики ошибки и точности в ходе обучения, проведено сравнение полученных сетей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import pandas
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
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)

model = Sequential()
model.add(Dense(30, input_dim=60, init="normal",
activation="relu"))
model.add(Dense(15, input_dim=60,
kernel_initializer='normal', activation='relu'))
model.add(Dense(1, init="normal", 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)

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)

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()
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()
```