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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр. 8382 Кобенко В.П.

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.

Порядок выполнения работы.

1. Ознакомиться с задачей бинарной классификации
2. Загрузить данные
3. Создать модель ИНС в tf.Keras
4. Настроить параметры обучения
5. Обучить и оценить модель
6. Изменить модель и провести сравнение. Объяснить результаты

Требования к выполнению задания.

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.

Задача классификации – задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна.

Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование) класса, к которому относится данный объект.

Классификация объекта – номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

Функция потерь (Loss function) – измеряет точность модели во время обучения. Мы хотим минимизировать эту функцию чтоб "направить" модель в верном направлении.

Оптимизатор (Optimizer) – показывает каким образом обновляется модель на основе входных данных и функции потерь.

Метрики (Metrics) – используются для мониторинга тренировки и тестирования модели. Наш пример использует метрику accuracy равную доле правильно классифицированных изображений.

Ход работы.

Для изучения различной структуры ИНС была разработана и использована программа из приложения А.

Чтобы подготовить сеть к обучению, были настроены три параметра для этапа компиляции:

1. Функция потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении. Для задач бинарной классификации применяется функция binary crossentropy.
2. Оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь.
3. Метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

На первом слое имеем 60 нейронов, что равно количеству элементов, которые подаются на вход НС. График точности и потерь модели изображен на рис. 1 и рис. 2 соответственно.

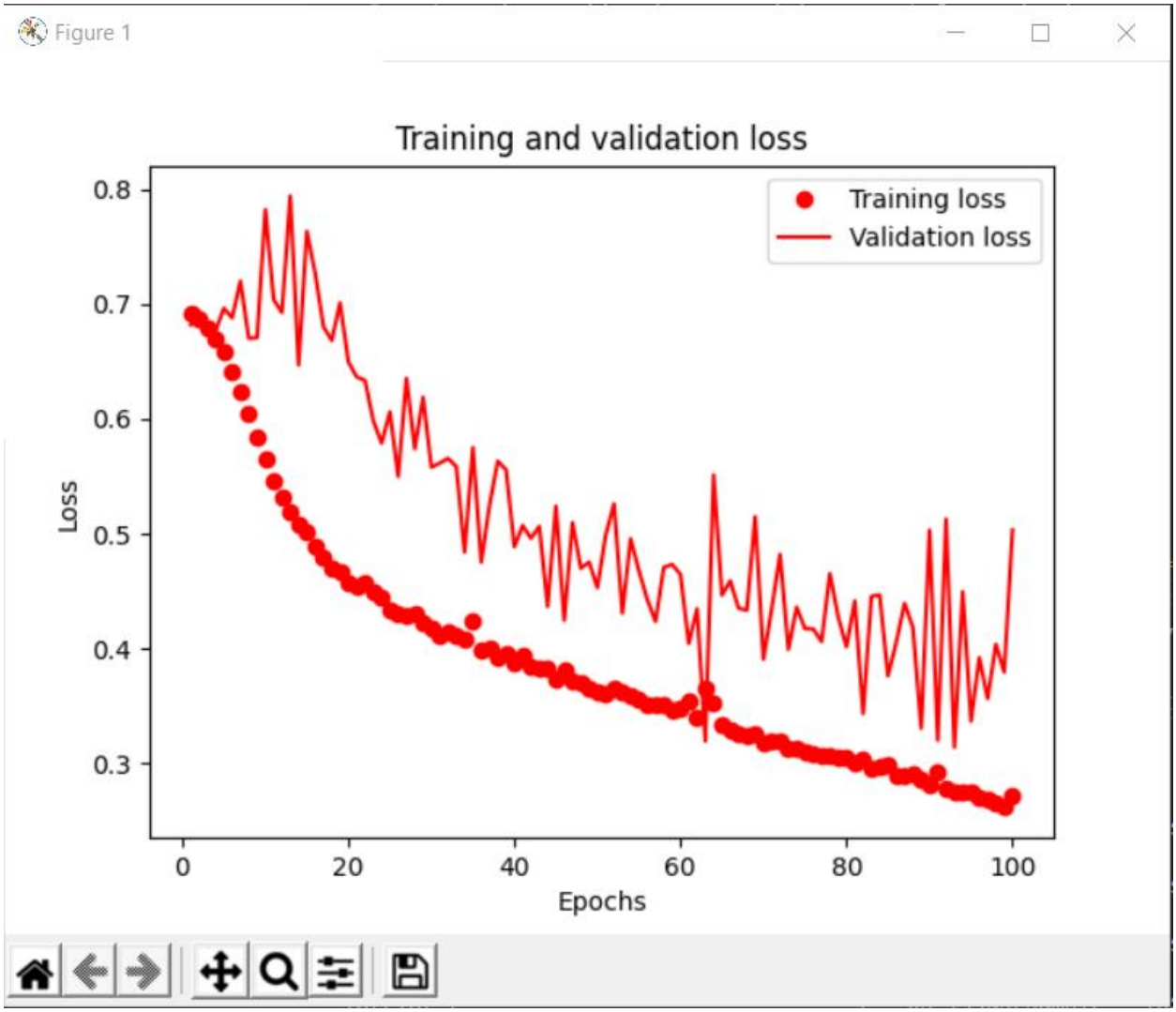

Рисунок 1 – График потерь модели при 60 нейронах

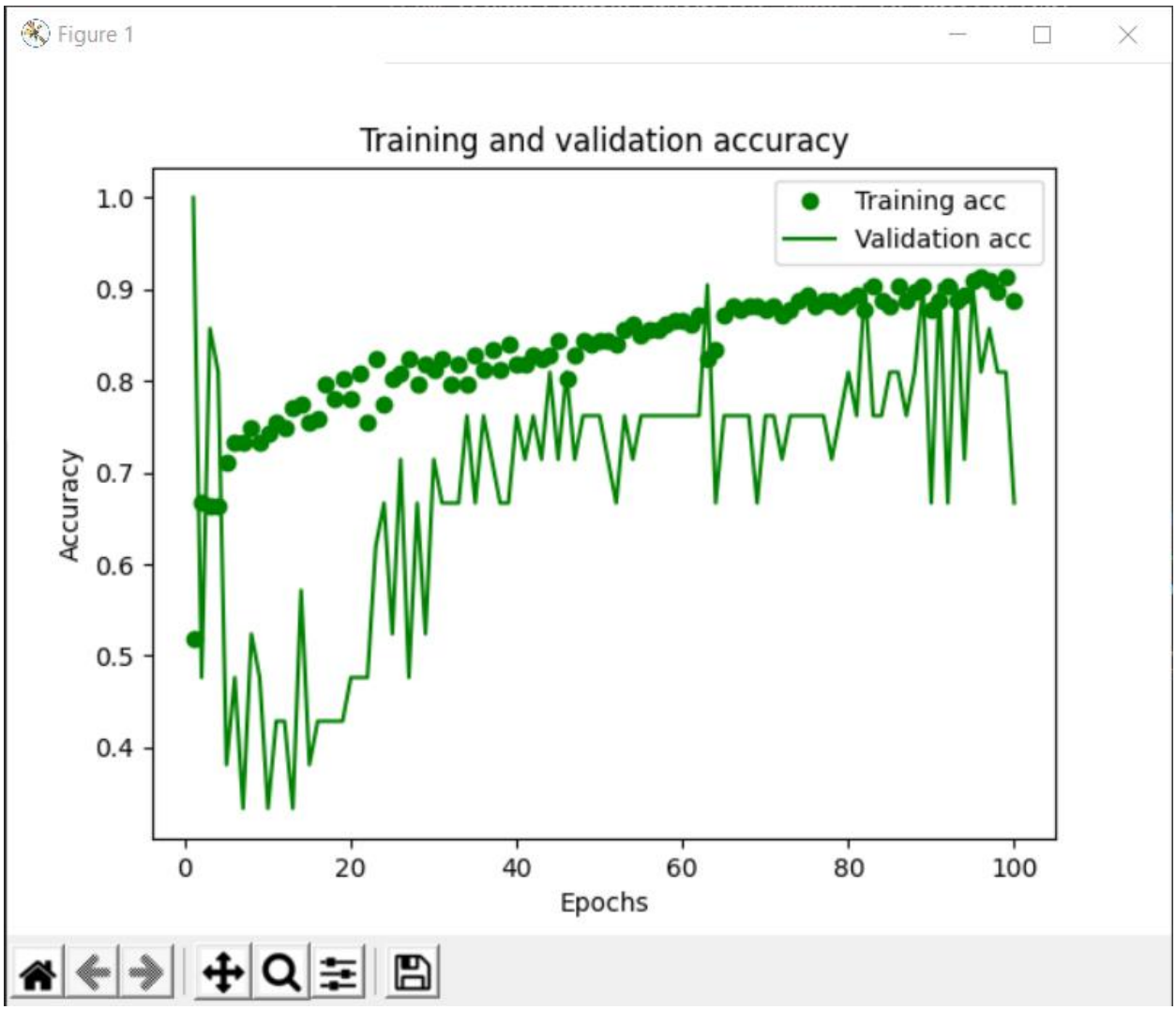

Рисунок 2 – График точности при 60 нейронах

Уменьшим размер входного слоя в два раза. Из графиков на рис. 3 и рис. 4 видно, что даже не обрабатывая на каждой итерации все элементы, можно получить результат, который не уступает в предыдущем случаем. Оба графика сходятся к таким же значениям, как и в предыдущем. Вывод, в 1 состоянии было излишнее количество нейронов в 1 слое.

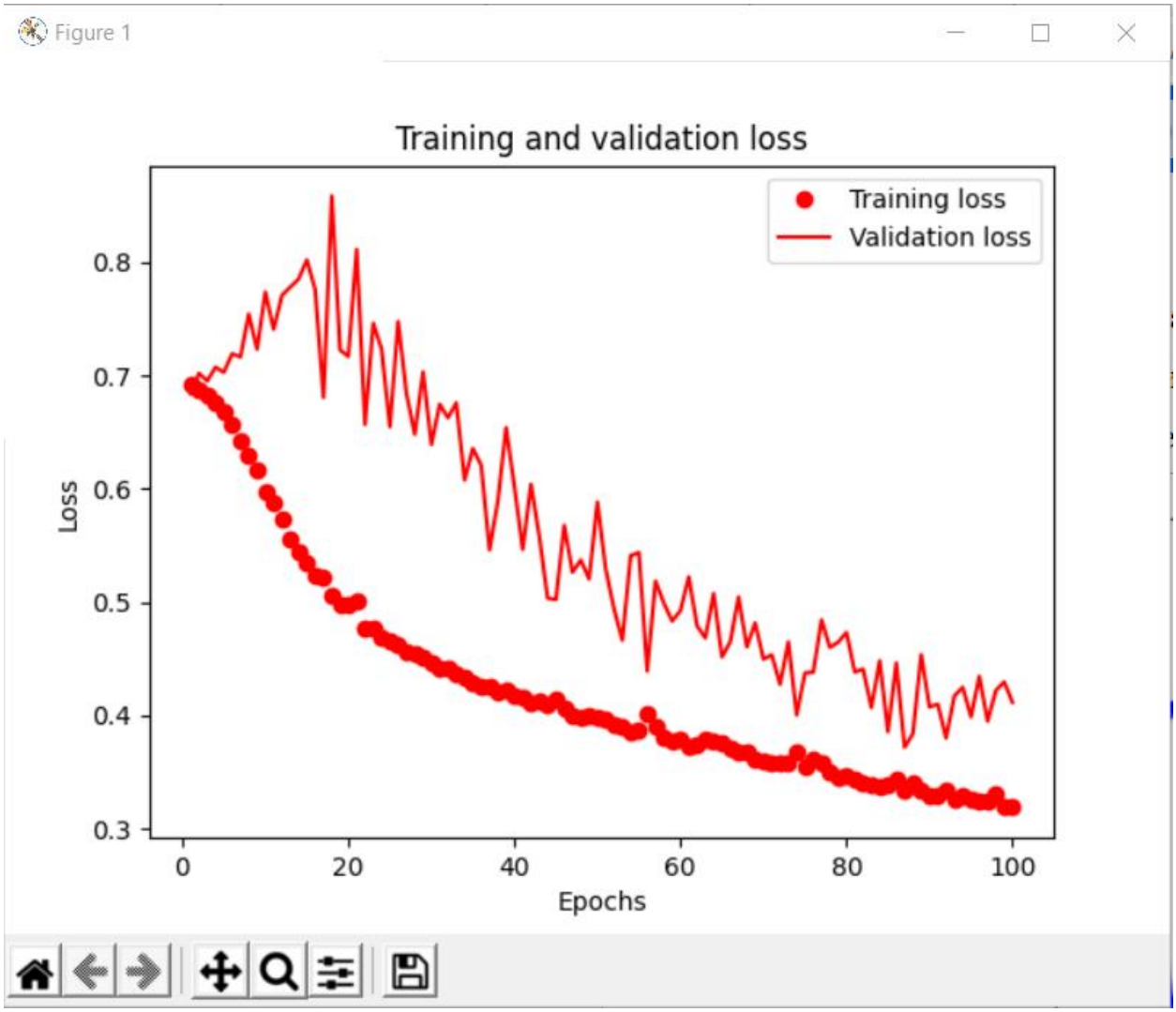

Рисунок 3 – График потерь модели при 30 нейронах

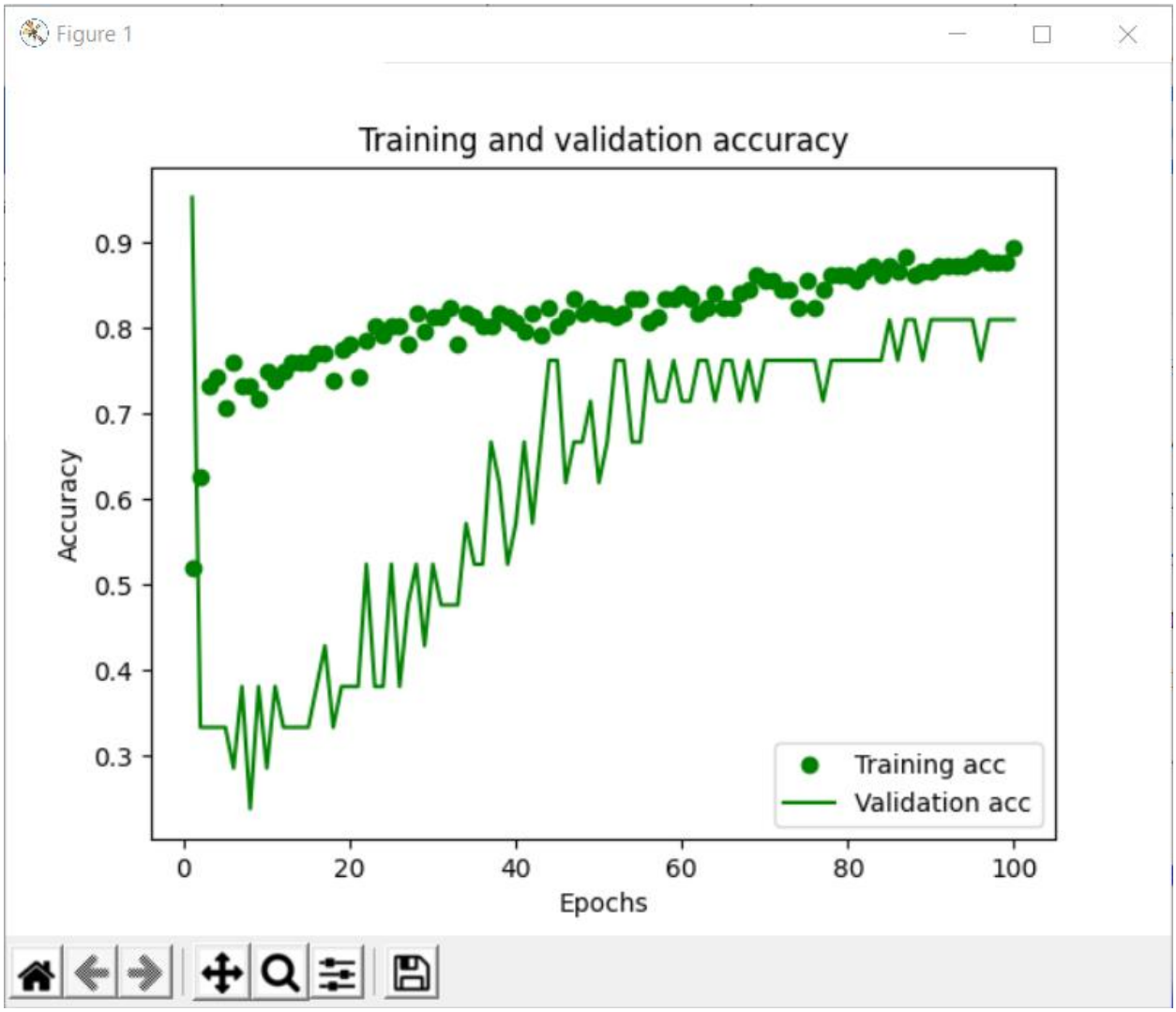

Рисунок 4 – График точности при 30 нейронах

Добавили промежуточный (скрытый) слой Dense в архитектуру сети с 15 нейронами. Из графиков на рис. 5 и рис. 6 видно, что точность увеличилась и ошибка стала меньше. Т.е. добавляя второй слой мы начали рассматривать еще и комбинации изначальных признаков (сигналов), что и дало выигрыш по сравнению со 2 состоянием.

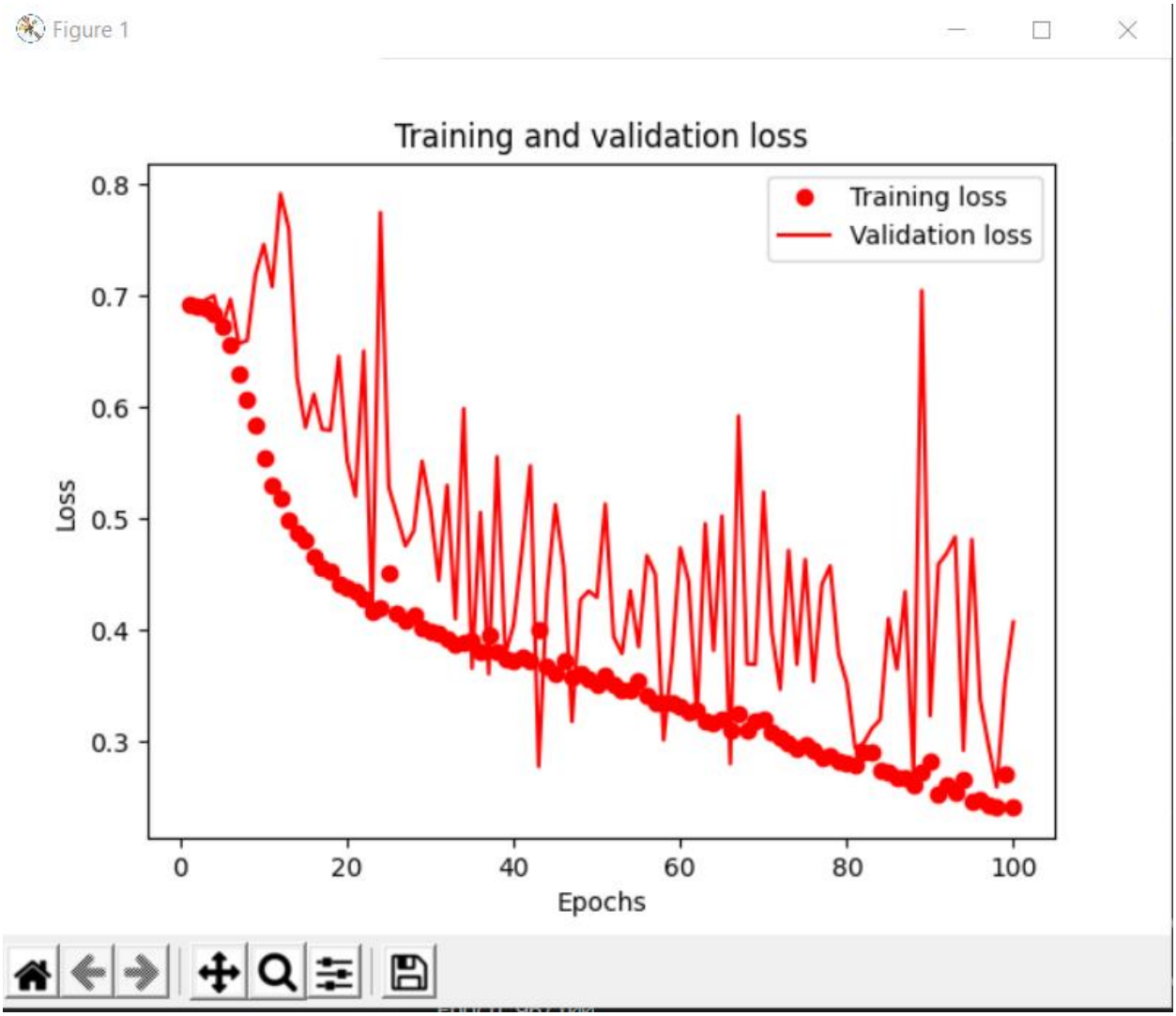

Рисунок 5 – График потерь модели при 30 нейронах на 1 слое и 15 нейронах на 2 слое

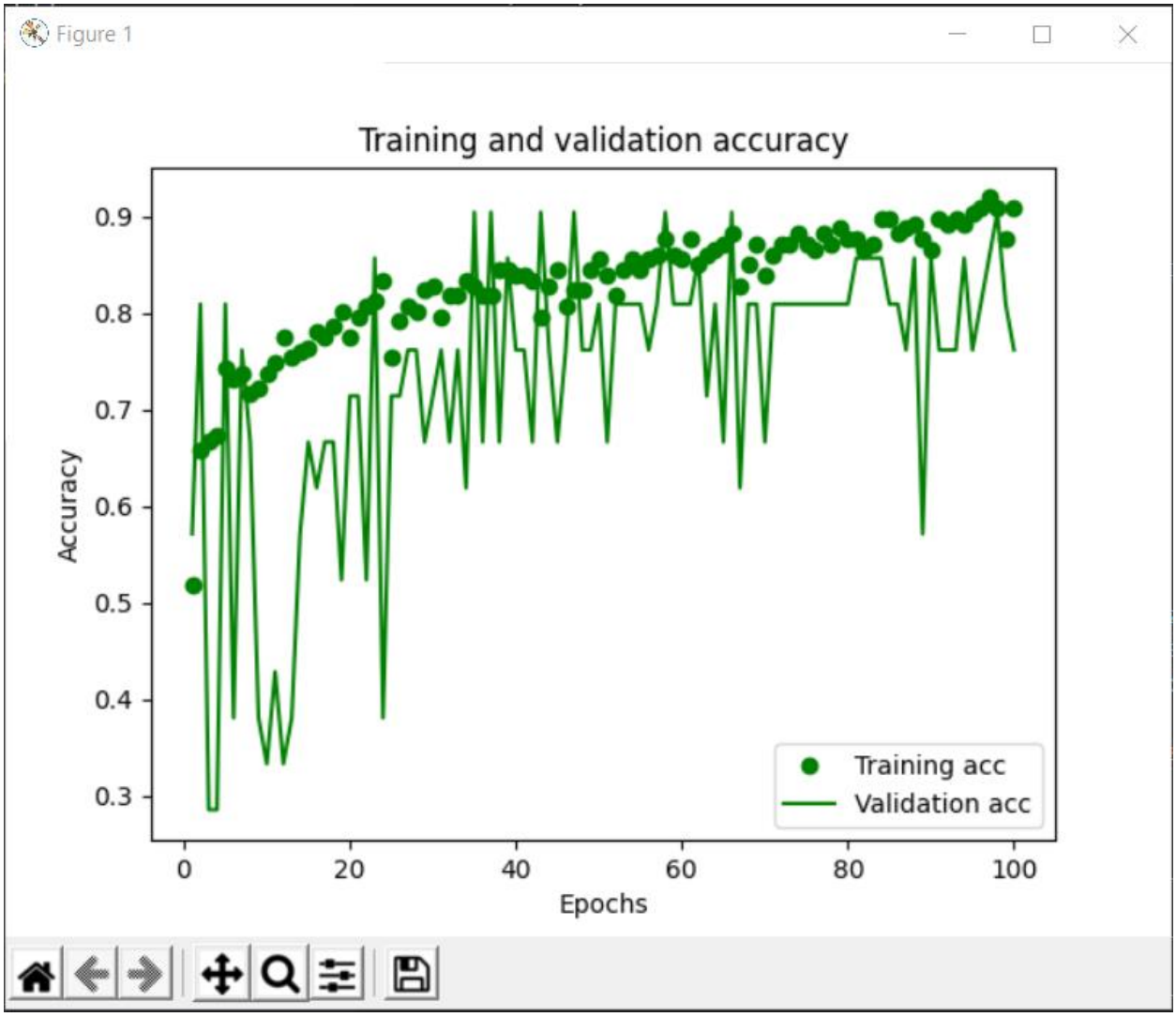

Рисунок 6 – График точности при 30 нейронах на 1 слое и 15 нейронах на 2 слое

Выводы.

В ходе работы было выявлено, что изменение количества нейронов во входном слое напрямую влияет на количество признаков, с которыми будет работать нейронная сеть. Так, сеть с 30 и 60 нейронами в 1 слое показала одинаковые результаты, следовательно не обрабатывая на каждой итерации все элементы результат остается неизменным. Добавляя второй слой мы начали рассматривать еще и комбинации изначальных признаков, что и дало выигрыш по сравнению со 2 состоянием.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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


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, kernel_initializer='normal',
activation='relu'))
model.add(Dense(15, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='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['acc']
val_acc = H.history['val_acc']
epochs = range(1, len(loss) + 1)
# Построение графика ошибки
plt.plot(epochs, loss, 'ro', label='Training loss')
plt.plot(epochs, val_loss, 'r', 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, 'go', label='Training acc')
plt.plot(epochs, val_acc, 'g', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```