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. 7382 _____ Токарев А.П.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Реализовать классификацию между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей.

60входныхзначенийпоказывают силу отражаемого сигнала под определеннымуглом.Входныеданные нормализованы инаходятсяв 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 входе обучения
4. Провести сравнение полученных сетей, объяснить результат

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.

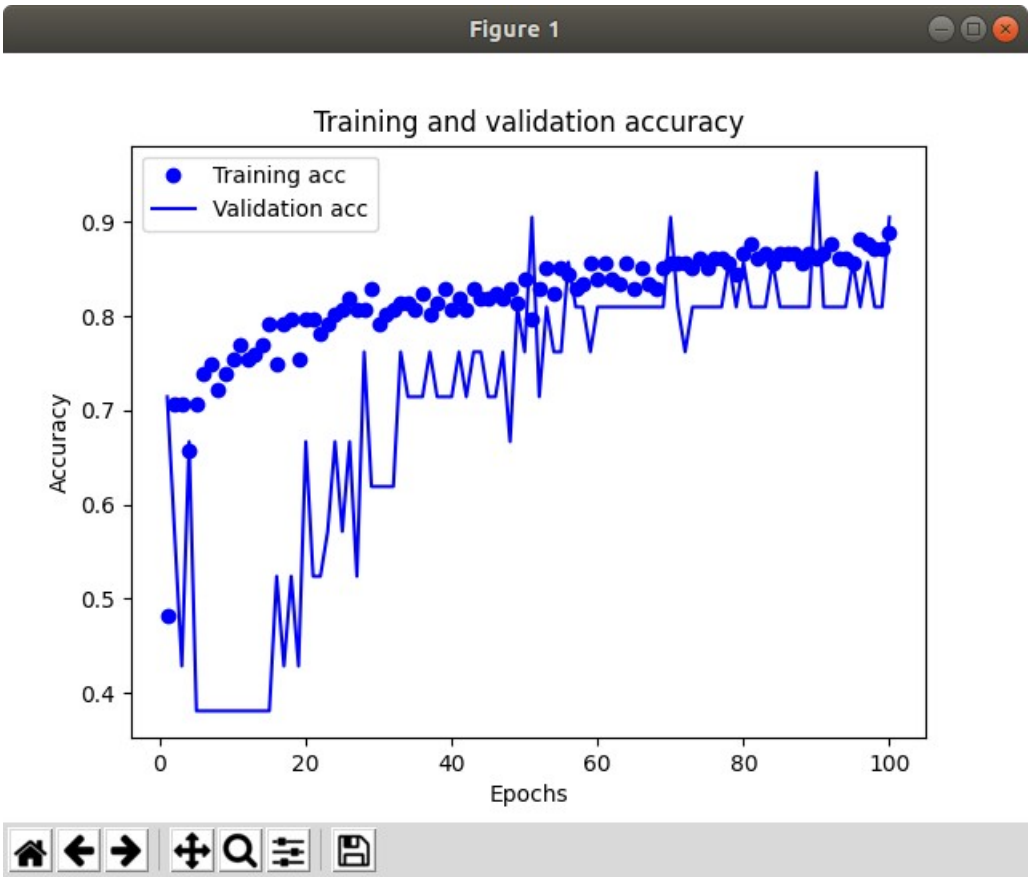

Рисунок 1 – График точности модели при 60 нейронах

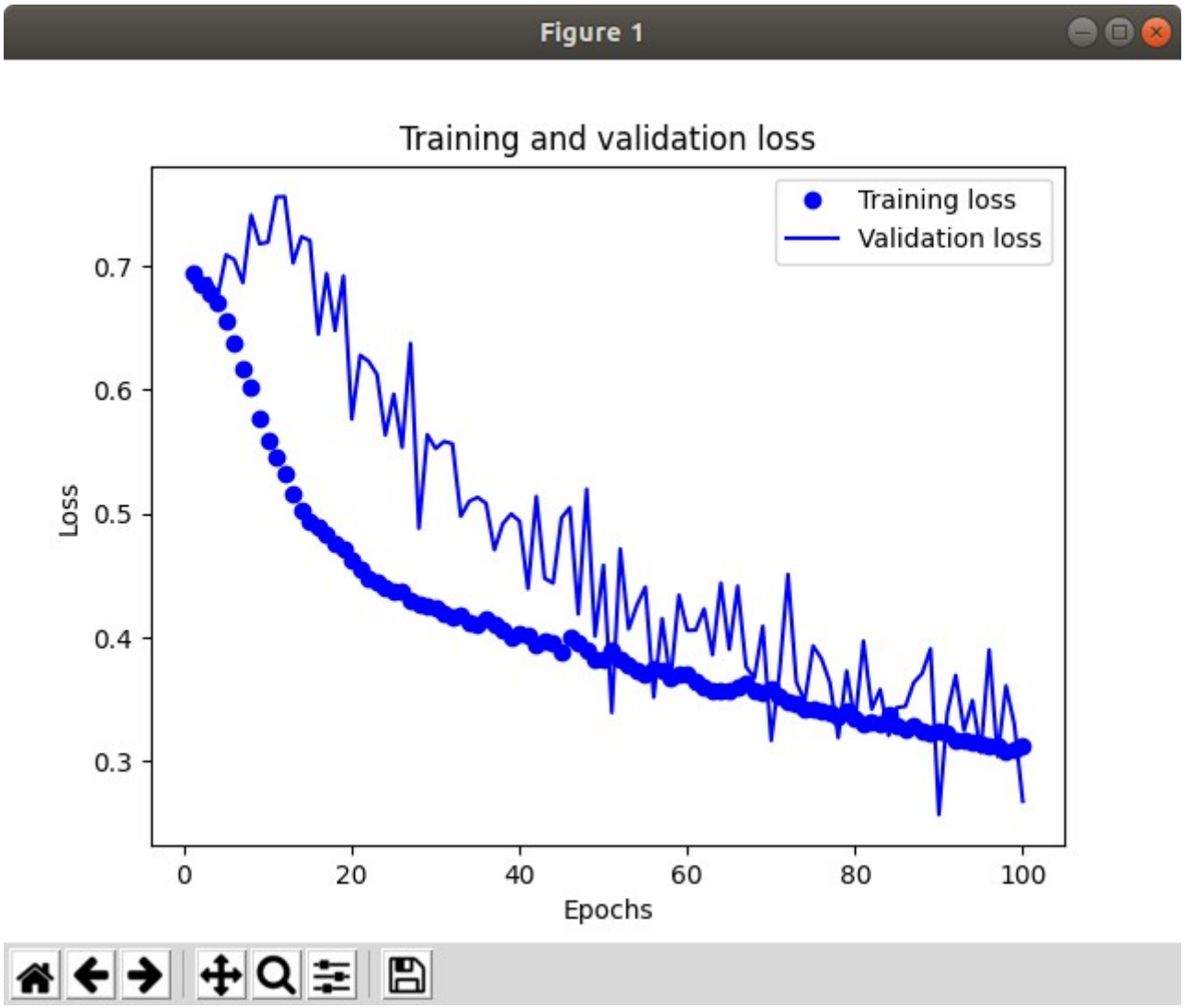

Рисунок 2 – График потерь при 60 нейронах

Добавим скрытый слой в 61 нейрон из расчёта Nск >= Nвх + Nвых

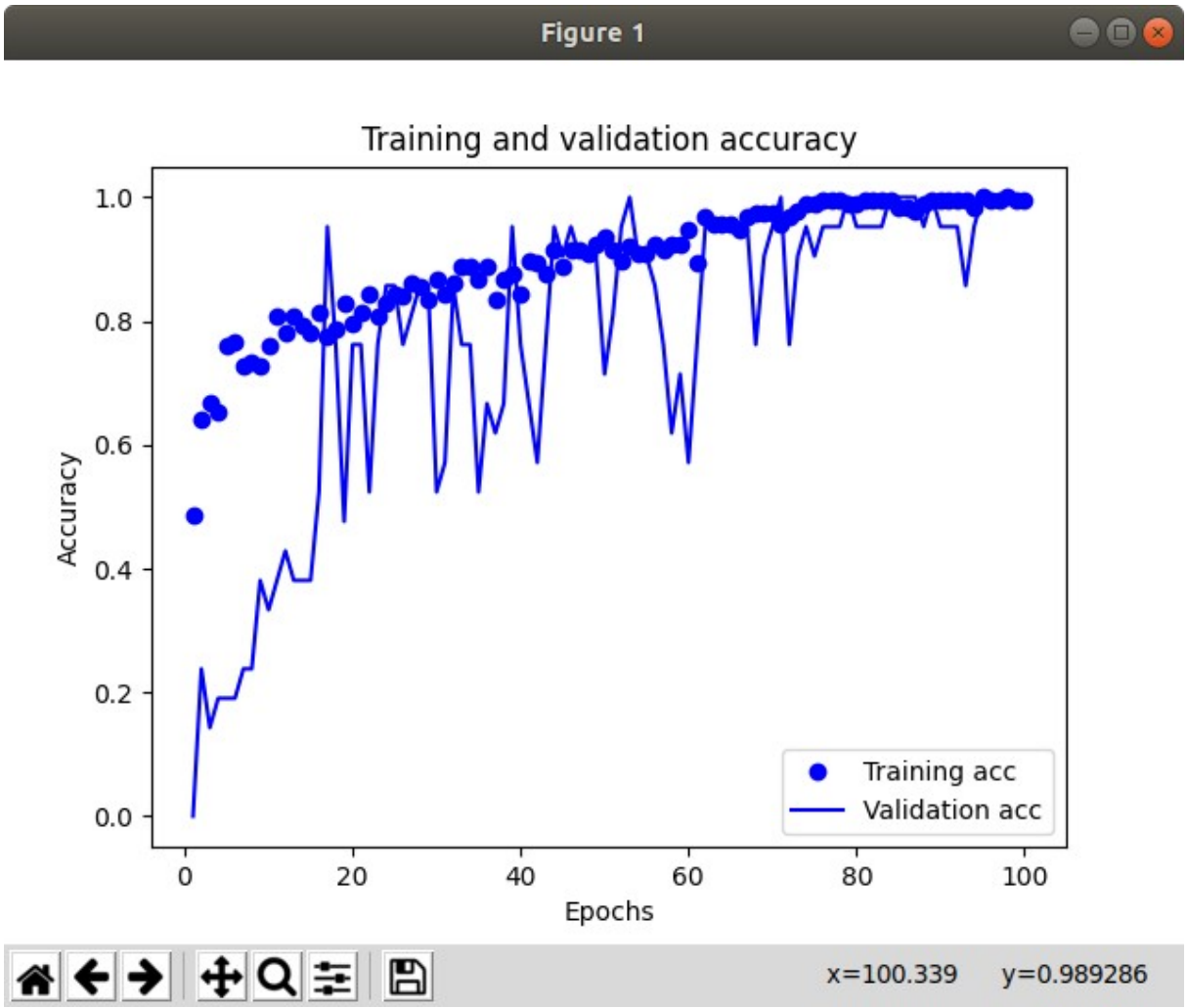

Рисунок 3 – График точности модели при доп слое

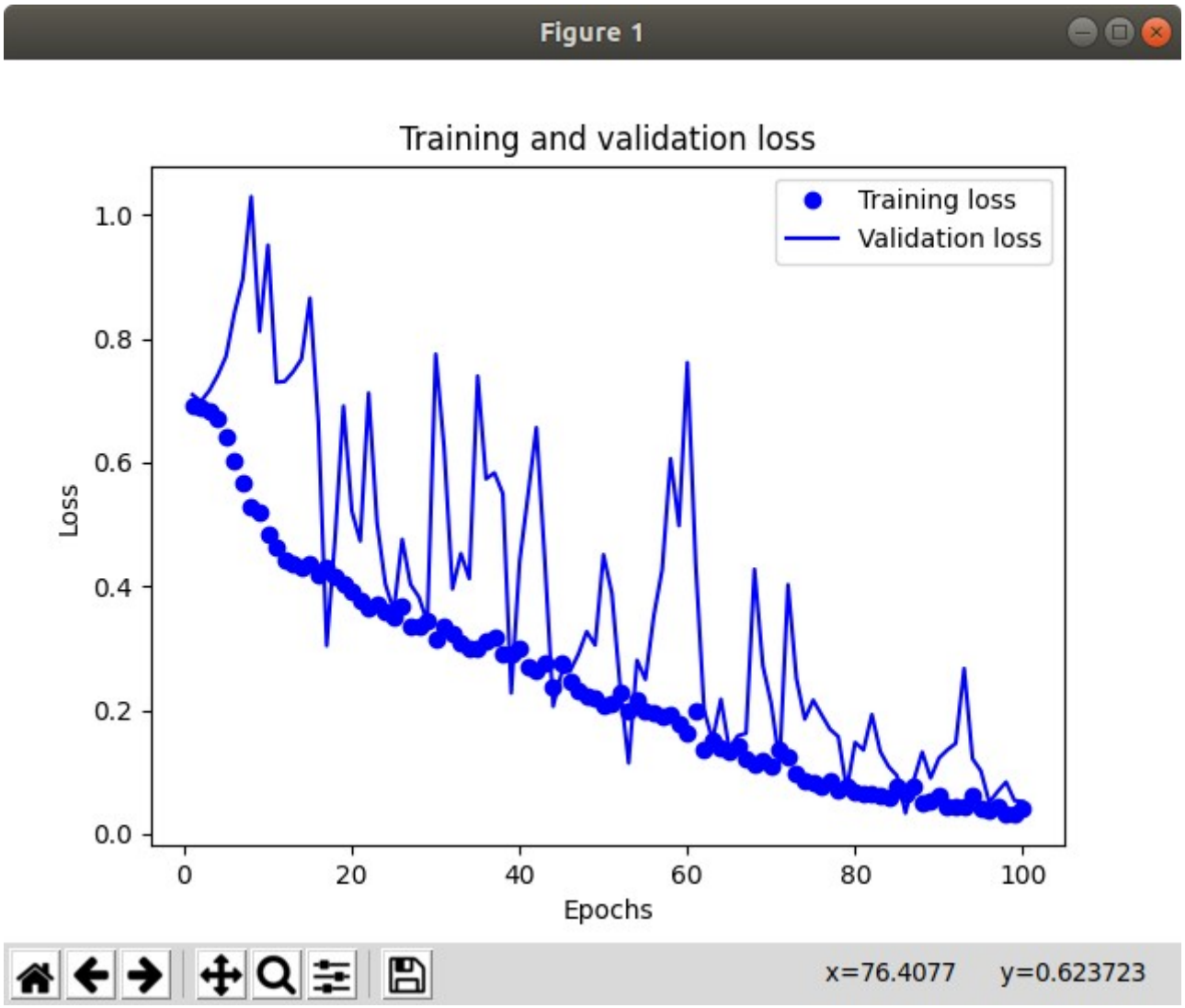

Рисунок 4 – График потерь при доп слое

Отлично, получили точность в 1.0

Выводы.

Вывод: Добавляя второй слой мы начали рассматривать комбинации изначальных признаков, что значительно улучшило точность модели, изменять количество нейронов на слоях — нецелесообразно, так как мы уже получили идеальную точность.

Приложение А.

Исходный код программы.

```
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(60, init='normal', activation='relu'))
model.add(Dense(61, init='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)

# 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()
```