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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр. 8383 _____ Бессуднов Г. И.

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.

Основные теоретические положения

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели.

Листинг 1:

```
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
```

Набор данных загружается напрямую с помощью pandas. Затем необходимо разделить атрибуты (столбцы) на 60 входных параметров (X) и 1 выходной (Y).

Листинг 2:

```
dataframe = pandas.read_csv("sonar.csv",
header=None)
dataset = dataframe.values
X = dataset[:,0:60].astype(float)
Y = dataset[:,60]
```

Выходные параметры представлены строками (“R” и “M”), которые необходимо перевести в целочисленные значения 0 и 1 соответственно. Для этого применяется LabelEncoder из scikit-learn.

Листинг 3:

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

Теперь можно задать базовую архитектуру сети.

Листинг 4:

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

Чтобы подготовить сеть к обучению, нужно настроить еще три параметра для этапа компиляции:

1.функцию 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).

Листинг 5:

```
model.compile(optimizer='adam',loss='binary crossentropy',
metrics=['accuracy'])
```

Теперь можно начинать обучение сети, для чего в случае использования библиотеки Keras достаточно вызвать метод fit сети — он пытается адаптировать (fit) модель под обучающие данные.

Листинг 6:

```
model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_split=0.1)
```

В процессе обучения отображаются четыре величины: потери сети на обучающих данных и точность сети на обучающих данных, а также потери и точность на данных, не участвовавших в обучении.

Выполнение работы

По теоретическим сведениям была построена модель №1, она представлена на рис. 1. Для отображения модели использовалась функция plot_model(). Пример графика потерь и точности модели представлен на рис. 2.

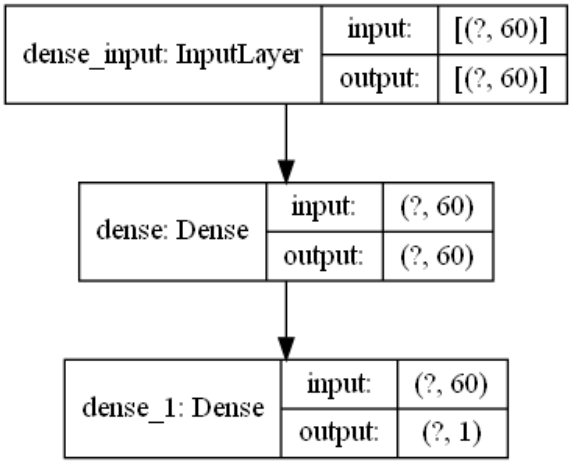

Рисунок 1: Схема модели №1

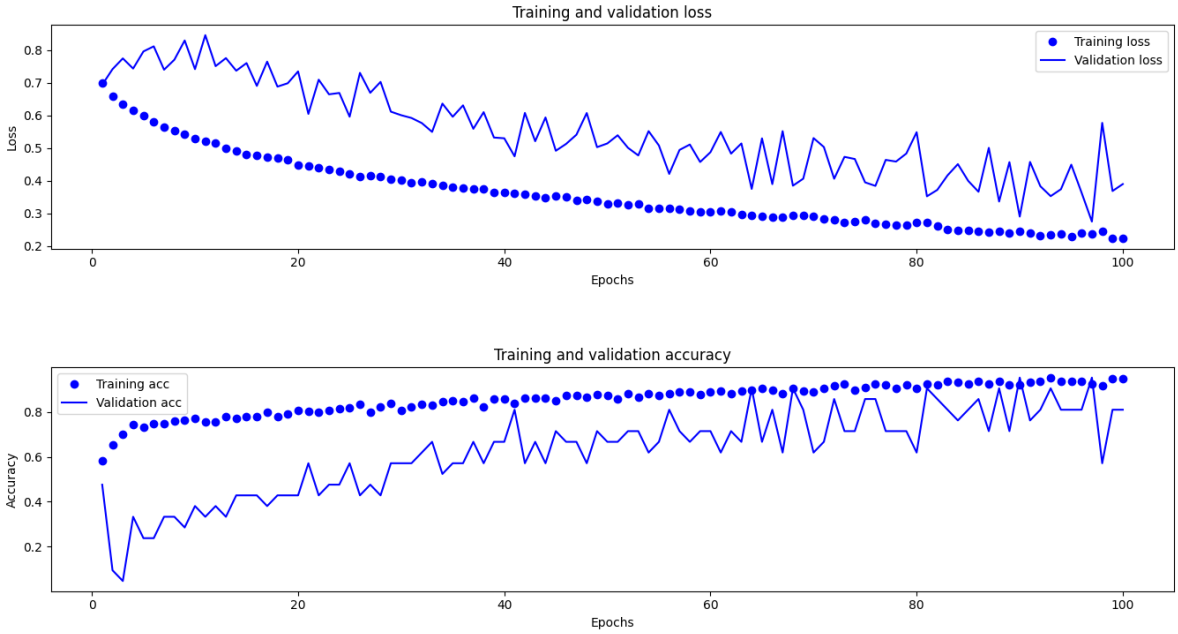

Рисунок 2: Результаты модели №1

Данная модель имеет довольно низкую точность и относительно большие потери на тестовых данных. Также график тестовых данных очень шумный и постоянно «скачет».

Для следующей модели количество нейронов на входном слое равняется 30. Ее схема представлена на рис. 3, результаты — на рис. 4.

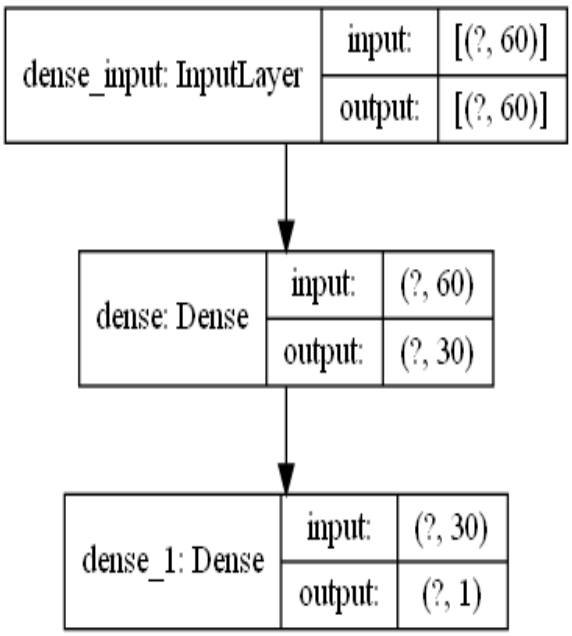

Рисунок 3: Схема модели №2

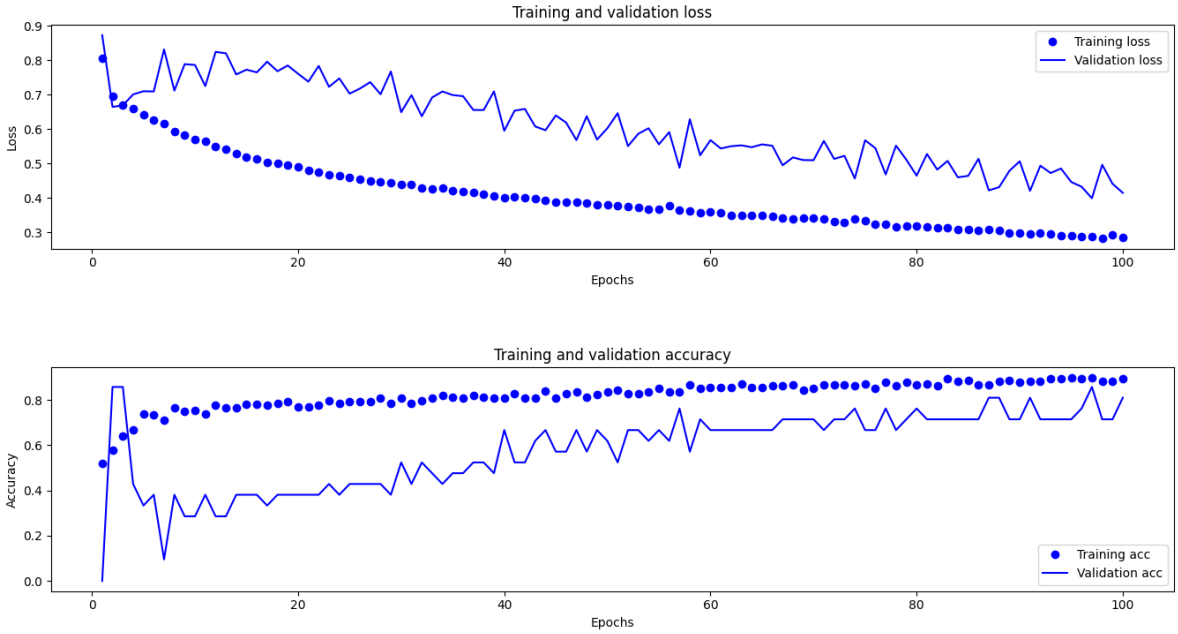

Рисунок 4: Результаты модели №2

Результат не сильно отличается первого варианта. Немного возрастает точность, но значение функции потерь остается примерно такое же.

Для модели №3 добавляется скрытый слой с 15 нейронами. Схема представлена на рис. 5, результаты — на рис. 6.

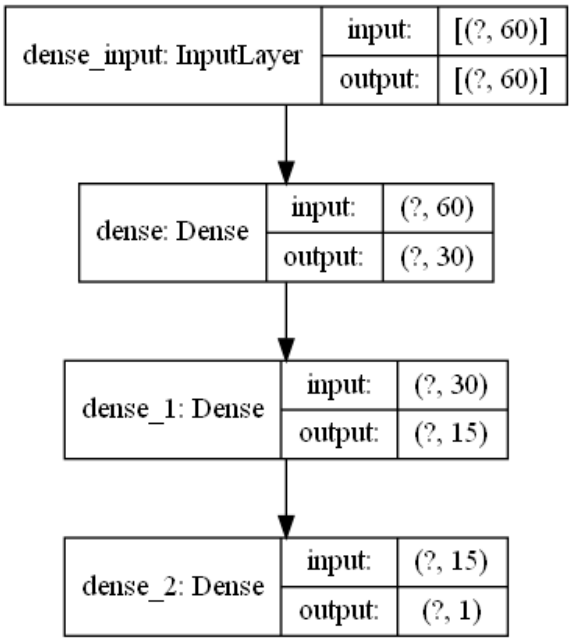

Рисунок 5: Схема модели №3

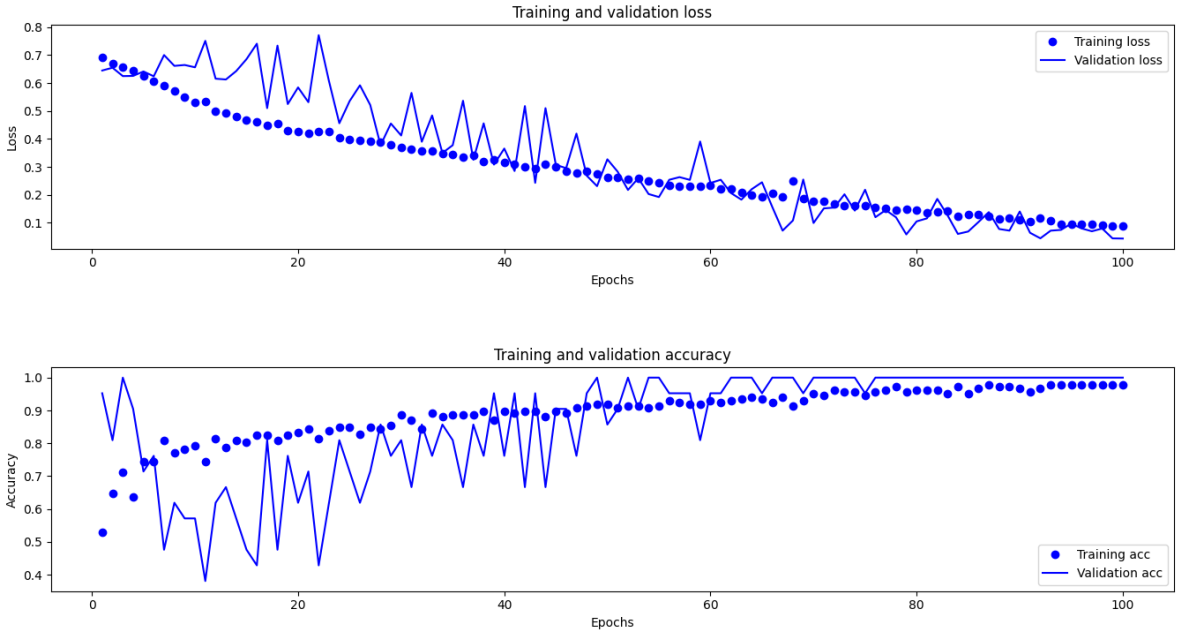

Рисунок 6: Результаты модели №3

По сравнению с другими вариантами эта модель делает значительный прогресс. Модель показывает высокую точность и малое значение функции потерь. Хотя графики тестовых значений по-прежнему остаются очень шумными, особенно на первых 50-ти эпохах, но результат приемлим.

Проведем еще один тест. Для следующей модели установим число нейронов на входном слое — 60, а на скрытом - 15. Схема модели №4 представлена на рис. 7, результаты — на рис. 8.

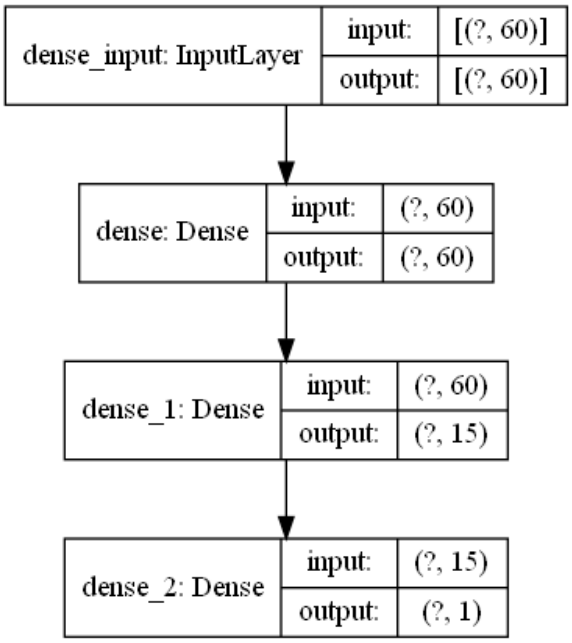

Рисунок 7: Схема модели №4

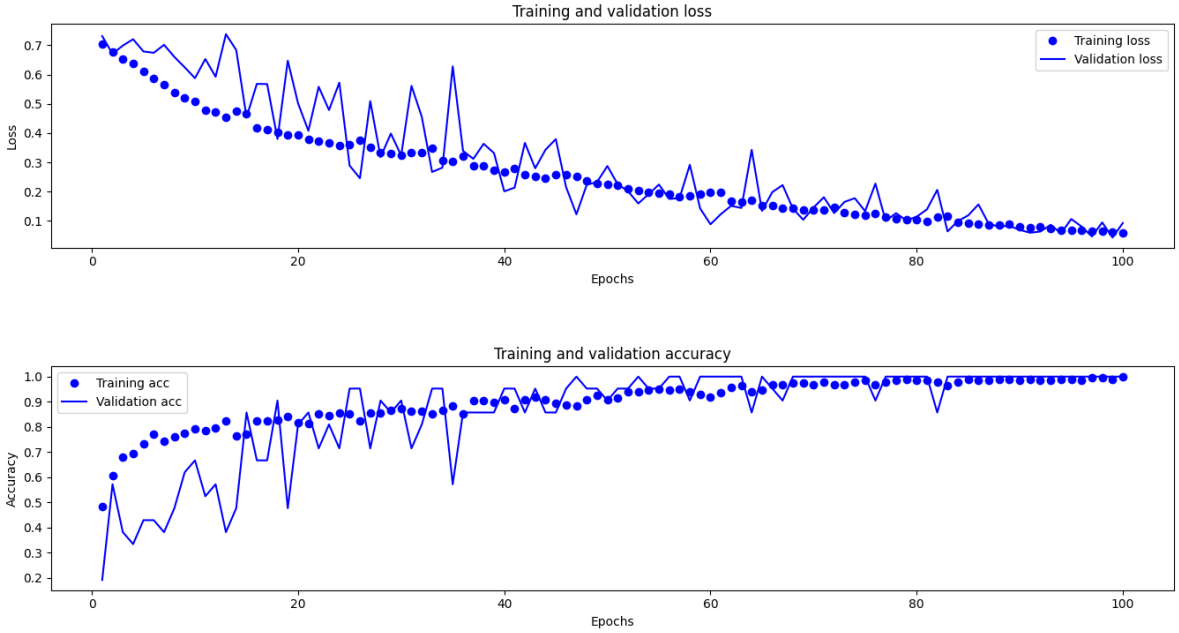

Рисунок 8: Результаты модели №4

Данная модель, также как и предыдущая, показывает хорошие результаты точности и малые значения функции потерь. Отличие может быть в том, что точность данной модели быстрее доходит до своего максимального значения и график выглядит менее шумным после 40-ой эпохи.

Выводы

Была реализована искусственная нейронная сеть для классификации между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей. В ходе работы проведено исследование, показывающее зависимость точности и потерь от количества слоев и нейронов на слое. Можно сделать вывод о том, что модели №3 и №4 показали лучшие результаты.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

```
from pathlib import Path

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

#Загрузка данных
path = Path("sonar.csv")
dataframe = pandas.read_csv(path.absolute(), header=None)
dataset = dataframe.values
X = dataset[:, 0:60].astype(float)
Y = dataset[:, 60]

# Категориальный вектор
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)

# Построение модели
model = Sequential()
model.add(Dense(60, input_dim=60, activation="relu"))
model.add(Dense(15, activation="relu"))
model.add(Dense(1, activation="sigmoid"))

# Параметры обучения
model.compile(optimizer="adam", loss="binary_crossentropy",
              metrics=["accuracy"])

# Обучение моедли
H = model.fit(X, encoded_Y, epochs=100, batch_size=10,
                 validation_split=0.1, verbose=0)
```

```
#Получение ошибки и точности в процессе обучения
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.subplot(2, 1, 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()

#Построение графика точности
plt.subplot(2, 1, 2)
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.tight_layout()
plt.show()

#печать модели
plot_model(model, to_file="model_4.png", show_shapes = True)
```