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

Студентка гр. 8382 _____ Кузина А.М.

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.

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

Требования

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

Ход работы

Данные для анализа в программу загружаются из файла sonar.csv. Данные разделяются на входные и выходные, выходные параметры переводятся в целочисленные.

Задается базовая архитектура сети – входной слой с 60ю нейронами и выходной слой с одним нейроном и функцией активации relu. После этого инициализируются параметры обучения сети – функция потерь, метрика, оптимизатор. Затем производится обучение сети в 100 эпохах батчами по 10 образцов с 10% валидационных данных.

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

Графики ошибок и точности изначальной сети представлены на рисунке 1

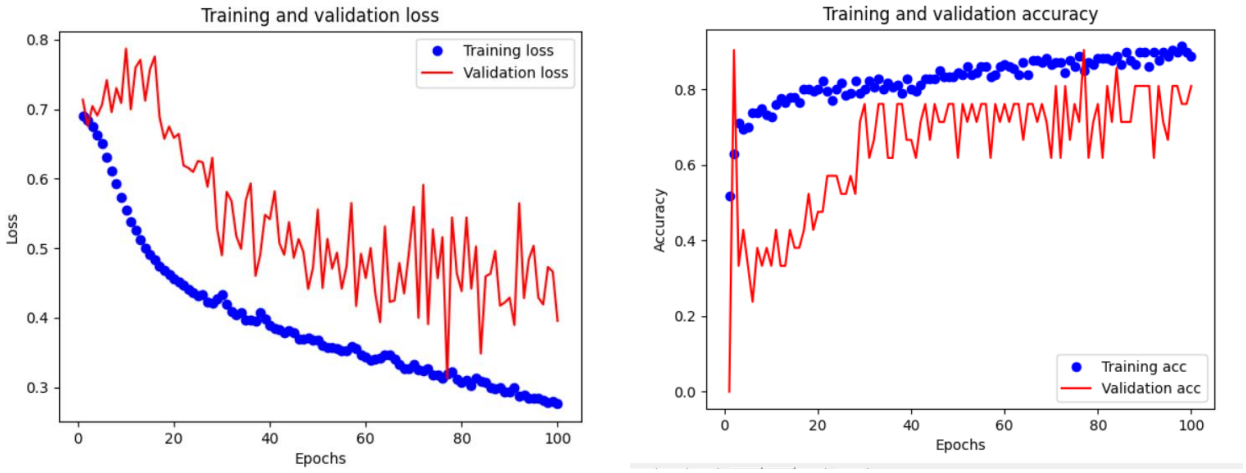

Рисунок 1 - Ошибки и точность изначальной сети

Затем количество нейронов на входном слое было уменьшено вдвое.

Сеть была обучена при данной конфигурации. На рисунке 2 представлены графики точности и ошибок сети данной сети.

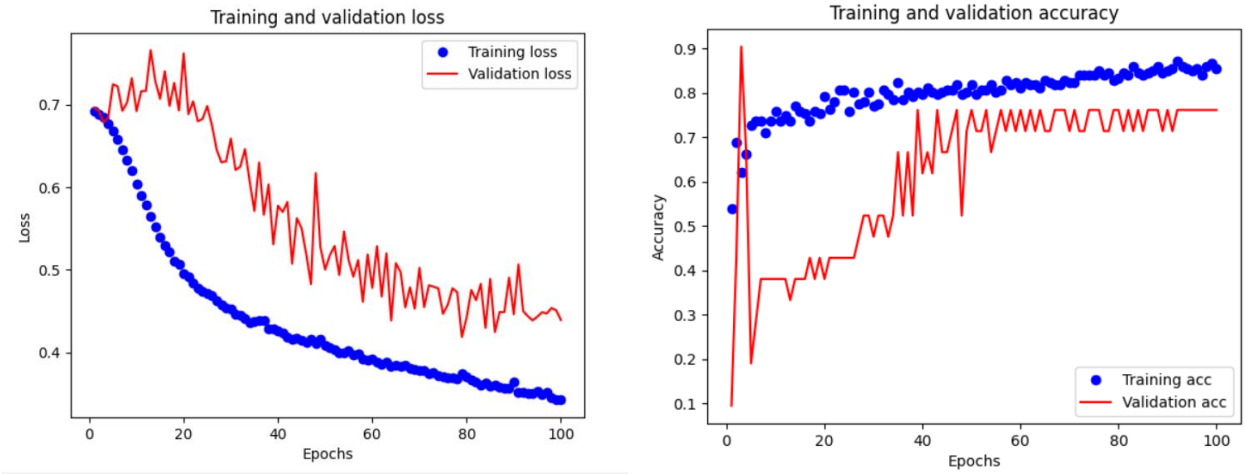

Рисунок 2 – Ошибки и точность сети, на входном слое 30 нейронов

Можно заметить, что в сравнении с изначальной конфигурацией сети, немного замедлилась скорость обучения сети, но точность обучения модели осталась на примерно таком же уровне, что просто заметить при повторных запусках программы. Следовательно во входных данных присутствует некоторая избыточность и в большем количестве нейронов на первом слое необходимости нет.

Добавим в исходную конфигурацию сети скрытый слой с 15ю нейронами и обучим данную сеть. На рисунке 3 представлены графики точности и ошибок сети с данной конфигурацией.

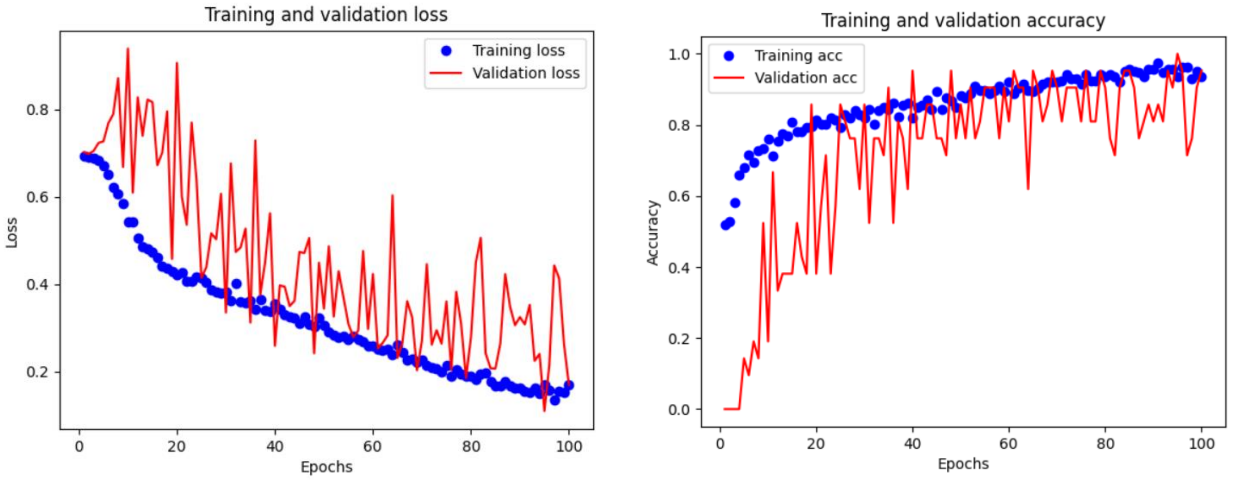

Рисунок 3 – Ошибки и точность сети, скрытый слой 15 нейронов

Можно заметить, что стала делать более точные предсказания – это связано с тем, что добавление нового слоя позволяет выявлять более сложные взаимосвязи и закономерности между входными и выходными данными. Однако при повторных запусках графики сильно различаются – иногда сеть стабильно выдает результат выше 0.8 на валидационных данных уже после 40 эпохи, но иногда даже за 100 эпох сеть на выходит выше точности в 0.7.

Рассмотрим конфигурацию сети, где на входном слое будет 30 нейронов, и также будет скрытый слой с 15ю нейронами. На рисунке 4 представлены графики точности и ошибок сети.

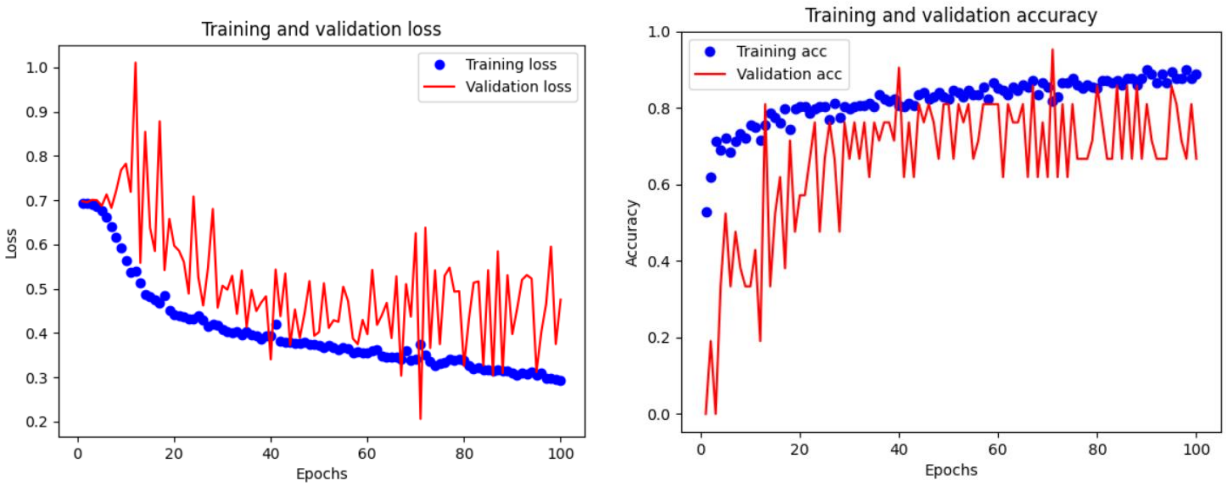

Рисунок 4 – Ошибки и точность сети, два слоя по 40 нейронов

Можно заметить, что в сравнении с предыдущей конфигурацией в данном случае сеть обучается хуже, чем сеть с предыдущей конфигурацией. Также примерно после 70 эпохи на валидационном множестве ошибки растут, а точность падает. Это свидетельствует о переобучении сети – нахождении несуществующих взаимосвязей.

Сравнивая все рассмотренные модели, наилучшие результаты показала модель с входным слоем с 60 нейронами, скрытым слоем с 15 нейронами и выходным слоем с одним нейроном.

Выводы

В ходе лабораторной работы было изучено влияние количества нейронов на слое, количества слоев на скорость и качество обучения. Было выявлено, что количество нейронов на слое влияет на количество признаков, рассматриваемых сетью. Также то, что количество слоев сети влияет на выявление более сложных зависимостей между входными и выходными данными. Также то, что при неудачной конфигурации сети, сеть может переобучиться – найти несуществующие соотношения во входных и выходных данных. Была создана, настроена и обучена нейронная сеть, обеспечивающая бинарную классификацию между камнями и металлическими цилиндрами.

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

# Загрузка данных
dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

# Переход от текстовых меток к категориальному вектору
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)

# Задание архитектуры сети
model = Sequential()
model.add(Dense(30, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(3, activation='softmax'))

# Инициализация параметров обучения
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])

# Обучение
history = model.fit(X, dummy_y, epochs=200, batch_size=8, validation_split=0.1)

#Получение ошибки и точности в процессе обучения
loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.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, '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, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```