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

ЛАБОРАТОРНАЯ РАБОТА №2 по дисциплине «Искусственные нейронные сети» Тема: «Бинарная классификация отраженных сигналов радара»

Студентка гр. 8382 _____ Ефимова М.А.

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.

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

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.

1. Была создана и обучена модель искусственной нейронной сети (код программы представлен в приложении А).

2. Для изучения архитектуры нейронной сети при различных параметрах обучения были внесены и рассмотрены следующие изменения:

- уменьшен размер входного слоя в два раза
- добавлен скрытый слой в архитектуру сети с 15 нейронами

На рис. 1-6 представлены графики точности и ошибок 3 моделей ИНС в ходе обучения.

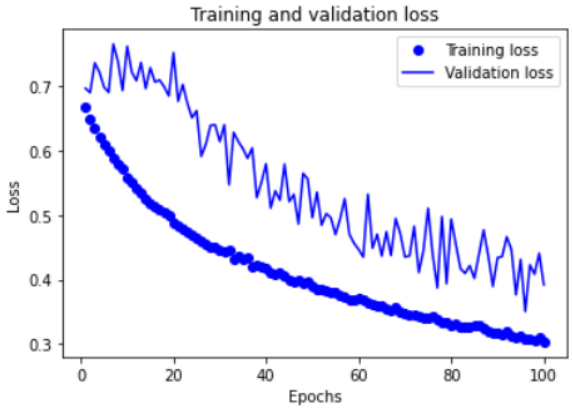

Рисунок 1 – график ошибок изначальной модели ИНС

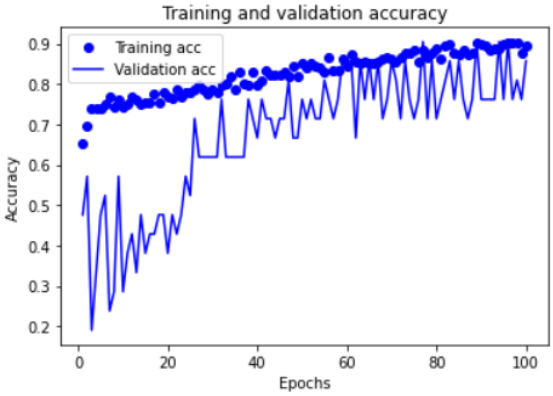

Рисунок 2 – график точности изначальной модели ИНС

Как видно из графиков, представленных на рис.3 и рис.4, при уменьшении размера входного слоя в два раза значительных улучшений или ухудшений результата не произошло, что может свидетельствовать о том, что изначальное количество нейронов было избыточным.

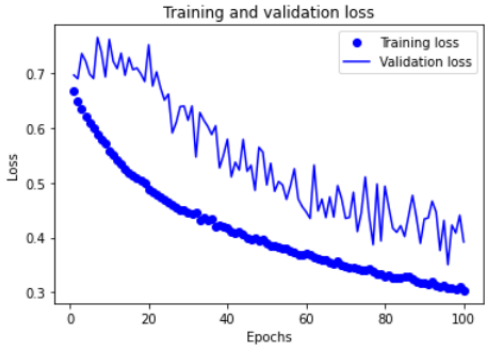

Рисунок 3 – график ошибок модели ИНС с уменьшенным количеством нейронов на входном слое

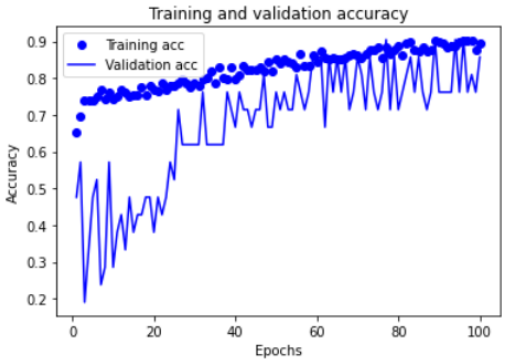

Рисунок 4 – график точности модели ИНС с уменьшенным количеством нейронов на входном слое

При добавлении промежуточного слоя с 15 нейронами результат заметно улучшился, точность стала выше, а потери меньше, что видно из графиков, представленных на рис.5 и рис.6

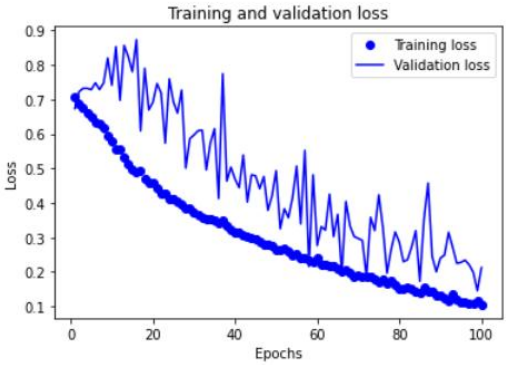

Рисунок 5 – график ошибок модели ИНС со скрытым слоем из 15 нейронов

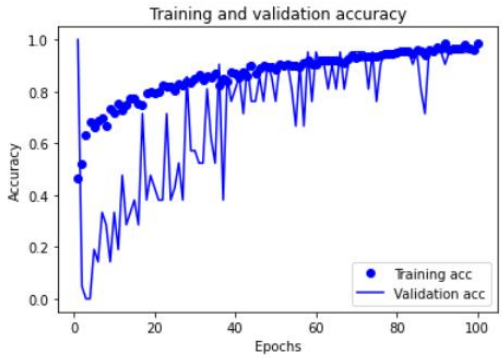

Рисунок 6 – график точности модели ИНС со скрытым слоем из 15 нейронов

Вывод.

В ходе выполнения данной работы было выявлено, что изменение количества нейронов во входном слое напрямую влияет на количество признаков, с которыми будет работать нейронная сеть. При уменьшении размера входного слоя результат не уступал предыдущему, что говорит о том, что изначально количество нейронов в 1 слое было избыточным. Было также выявлено, что нейронная сеть с несколькими слоями позволяет находить закономерности не только во входных данных, но и в их комбинации. Также, дополнительные слои позволяют ввести нелинейность

в сеть, что позволяет получать более высокую точность. При добавлении промежуточного слоя с 15 нейронами значительно выросла точность и уменьшилась ошибка, что говорит о преимуществе данной модели перед остальными.

Приложение А

```
import pandas

#model
from keras.layers import Dense
from keras.models import Sequential

#preprocessing
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

dataframe = pandas.read_csv("/content/sonar.all-data.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)

#           bias_initializer - Инициализатор вектора смещения.
model = Sequential()
model.add(Dense(30, input_dim=60, bias_initializer='normal', activation='relu'))
model.add(Dense(15, input_dim=60, bias_initializer='normal', activation='relu'))
model.add(Dense(1, bias_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)

#dict_keys(['val_loss', 'acc', 'loss', 'val_acc'])

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']

epochs = range(1, len(loss_values)+1)

# "bo" is for "blue dot"
# b is for "solid blue line"
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()  # очистка фигуры
plt.plot(epochs, acc_values, 'bo', label='Training acc')
plt.plot(epochs, val_acc_values, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```