**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»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7383 |  | Зуев Д.В. |
| Преподаватель |  | Жукова Н. А. |

Санкт-Петербург

2020

**Цели.**

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:

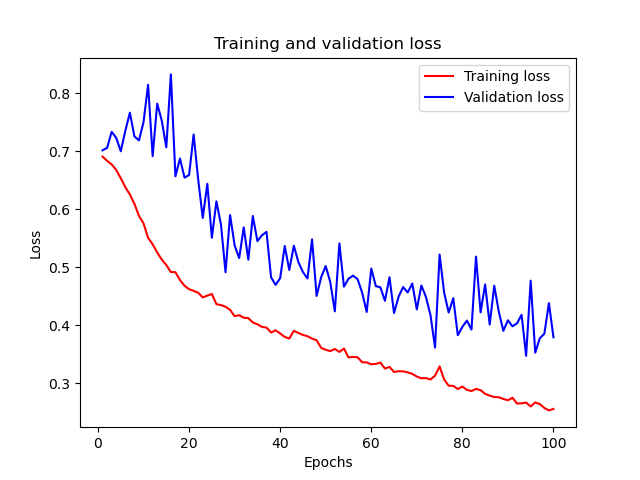
1. Изучить влияние кол-ва нейронов на слое на результат обучения модели.
2. Изучить влияние кол-ва слоев на результат обучения модели
3. Построить графики ошибки и точности в ходе обучения
4. Провести сравнение полученных сетей, объяснить результат

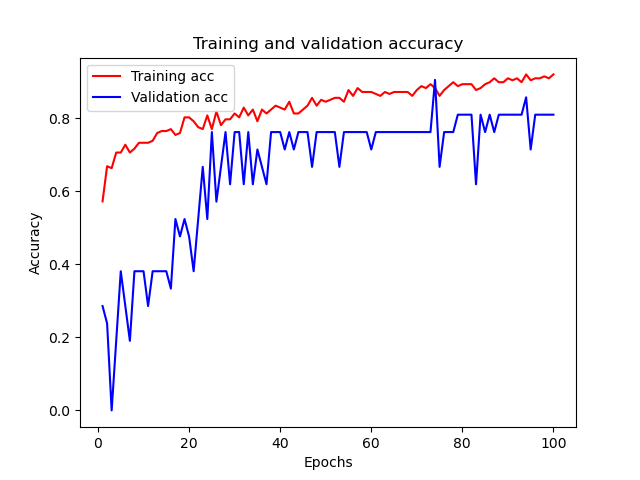
**Ход работы.**

1. Была создана и обучена модель искусственной нейронной сети в соответствии с условиями (весь код представлен в приложении А).
2. Было проведено исследование разных архитектур:

* Был уменьшен размер входного слоя в два раза.
* Был добавлен скрытый слой в архитектуру сети с 15 нейронами

Было получено 4 модели ИНС, которые нужно сравнить. Графики точности и ошибок моделей в ходе обучения представлены на рис. 1-8.

Рисунок 1 – график ошибок изначальной модели ИНС

Рисунок 2 – график точности изначальной модели ИНС

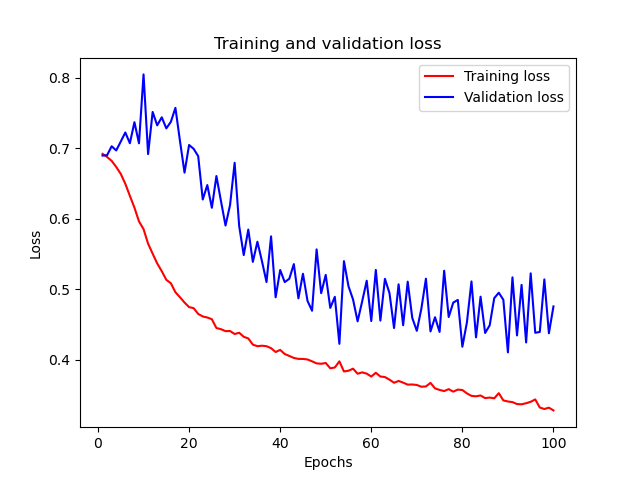


Рисунок 3 – график ошибок модели ИНС с уменьшенным количеством нейронов в входном слое

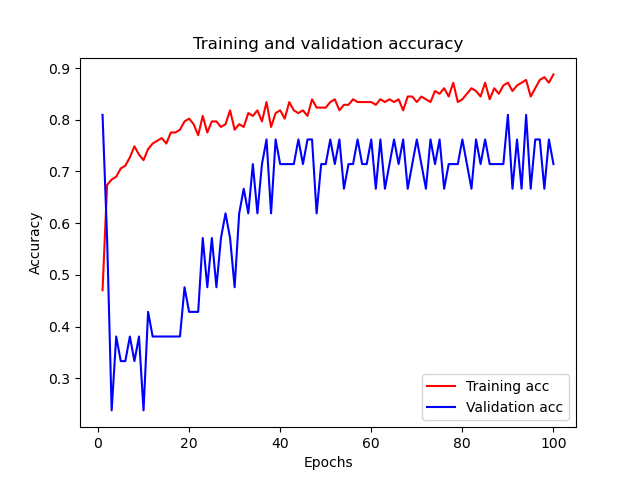


Рисунок 4 – график точности модели ИНС с уменьшенным количеством нейронов в входном слое

При уменьшении числа нейронов во входном слое немного увеличилась ошибка и уменьшилась точность обучения. Это связано с недостатком признаков, с которыми работает нейронная сеть.

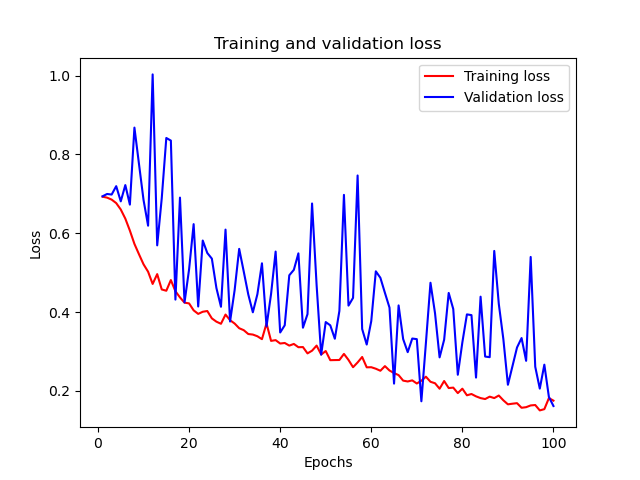


Рисунок 5 – график ошибок модели ИНС с скрытым слоем

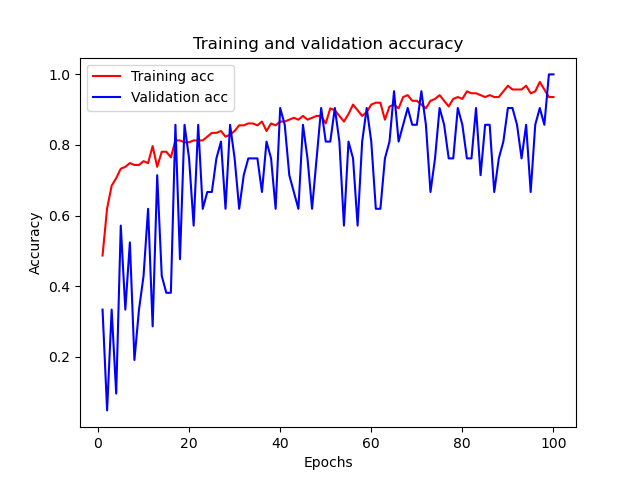


Рисунок 6 – график точности модели ИНС с скрытым слоем

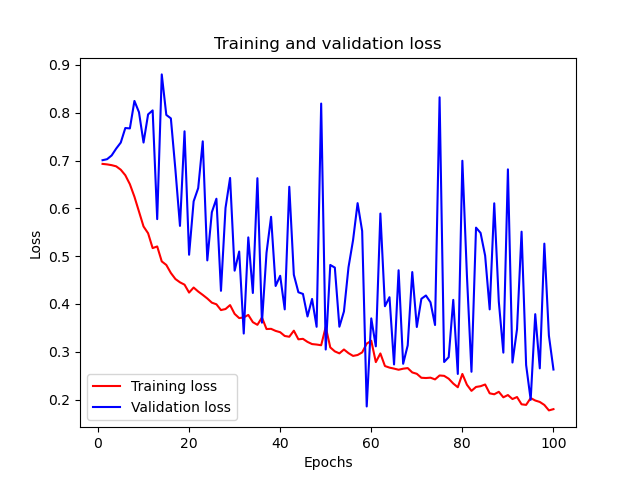


Рисунок 7 – график ошибок модели ИНС с уменьшенным количеством нейронов в входном слое и скрытым слоем

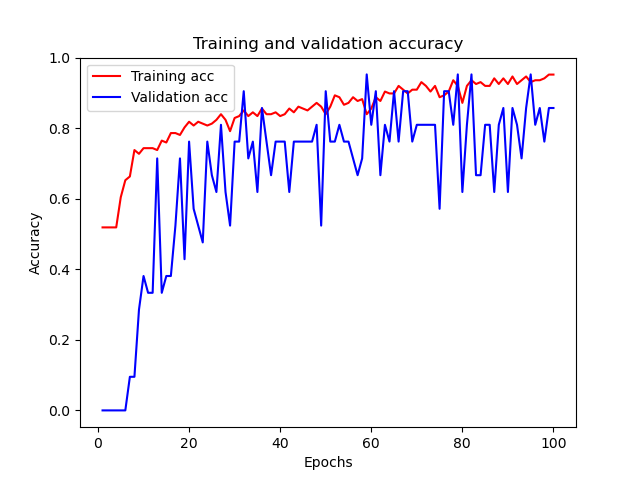


Рисунок 8 – график точности модели ИНС с уменьшенным количеством нейронов в входном слое и скрытым слоем

При добавлении скрытого слоя появляется переобучение сети. На графике это можно увидеть, когда кривая ошибок на тестовых данных находится сильно выше кривой ошибок на обучающих данных, а с кривой точности, наоборот. Это связано с избыточной сложностью модели.

**Вывод.**

В ходе выполнения данной лабораторной работы было изучено влияние количества нейронов во входном слое, количества слоев на результат обучения модели. В результате было выяснено, что при уменьшении количества нейронов во входном слое может уменьшиться и точность работы нейронной сети из-за недостатка признаков, так же, что при увеличении числа скрытых слоев может появиться эффект переобучения сети, в связи с избыточной сложностью архитектуры модели.

**ПРИЛОЖЕНИЕ А**

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

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['accuracy']

val\_acc = H.history['val\_accuracy']

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

print(len(loss))

plt.plot(epochs, loss, 'r', 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()

plt.clf()

plt.plot(epochs, acc, 'r', 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()