|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Специального машиностроения**

КАФЕДРА **СМ11 «Подводные роботы и аппараты»**

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Андреев Евгений Викторович

*фамилия, имя, отчество*

Группа СМ11-31М

Тип практики **Научно-исследовательская работа**

Название предприятия **НУК СМ МГТУ им. Н.Э. Баумана**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Андреев Е. В.

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Макашов А. А. *подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2021 г.*

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра **«Подводные роботы и аппараты»** **(СМ11)**

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

**(производственной практики)**

на предприятии **НУК СМ МГТУ им. Н.Э. Баумана**

Студент Андреев Евгений Викторович, СМ11-31М

(фамилия, имя, отчество; индекс группы)

**Тема научно-исследовательской работы:**

Использование системы видеопозиционирования в АНПА

**Дата выдачи задания « » сентября 2020 г.**

**Руководитель НИР**   **/** Макашов А. А.

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /** Андреев Е. В.

(подпись, дата) (Фамилия И.О.)

РЕФЕРАТ

Отчёт на 33 стр., 3 ч., Рисунок 19 рис., 7 источников, 1 таблица.

ИСПОЛЬЗОВАНИЕ СИСТЕМЫ ВИДЕОПОЗИЦИОНИРОВАНИЯ В АНПА

Перечень ключевых слов: АНПА, видеокамера, донная зарядная станция, распознавание образов, свёрточная нейронная сеть, полётная траектория.

Целью данной работы является исследование возможности использования системы видеопозиционирования для наведения, обхода и стыковки с донной зарядной станцией.

В процессе работы был проведён сбор и систематизация информации о различных типах нейронных сетей, их архитектуре и способах подготовки набора данных для обучения. Были исследованы современные программные инструменты для тренировки нейронных сетей и контроля процесса обучения. Описаны техники отключения произвольных нейронов (dropout) и искусственное наполнение данных (data augmentation) для борьбы с переобучением (т.н. overfitting).

Предложена траектория обхода донной зарядной станции в плоскости горизонта без потери станции из видимости. Результаты исследования проверены на математической модели в пакете Matlab Simulink.

В приложениях приведены исходные коды на языке Python для обучения искусственной нейронной сети и расчёта точек траектории.

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc58164207)

[ВВЕДЕНИЕ 6](#_Toc58164208)

[1 Исходные данные 8](#_Toc58164209)

[2 Глубокая нейронная сеть 10](#_Toc58164210)

[2.1 Выбор архитектуры и типа сети 11](#_Toc58164211)

[2.2 Подготовка набора данных для обучения 15](#_Toc58164212)

[2.3 Метрики качества и борьба с переобучением 16](#_Toc58164213)

[3 Построение траектории обхода и стыковки 20](#_Toc58164214)

[3.1 Траектория обхода донной станции 20](#_Toc58164215)

[3.2 Траектория стыковки с донной станцией 23](#_Toc58164216)

[ЗАКЛЮЧЕНИЕ 25](#_Toc58164217)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc58164218)

[Приложение А. Листинг кода обучения нейронной сети 27](#_Toc58164219)

[Приложение Б. Программа для расчёта координат точек траектории 32](#_Toc58164220)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете о НИР применяются следующие сокращения и обозначения:

CNN – Convolutional Neural Net;

АНПА – Автономный необитаемый подводный аппарат;

ВК – Видеокамера;

ДЗС – Донная зарядная станция;

ПО – Программное обеспечение;

СНС – свёрточная нейронная сеть;

СК – система координат.

# ВВЕДЕНИЕ

В настоящее время всё большее применение находят автономные необитаемые подводные аппараты (АНПА). К видам технических работ, осуществляемых аппаратами, можно отнести обследование трубопроводов и кабелей, проверку точности карт, фото- и видеосъёмку, в том числе маршрутную, осмотр опор эстакад и платформ и много другое.

Целесообразно использовать уже имеющееся на АНПА оборудование для повышения точности определения пространственного положения. Видеосистемы, в отличие от инерциальных, определяют позиционные координаты аппарата и не подвержены накоплению ошибки с течением времени. В этой связи актуальной задачей является разработка методов позиционирования подводного аппарата по данным видеосистемы.

Исследование проводилось применительно к задаче стыковки АНПА с донной зарядной станцией. Для наведения предлагается использовать предварительно обученную свёрточную нейронную сеть для распознавания передней грани и остальных граней донной зарядной станции (ДЗС), а также каскадный классификатор Хаара для обнаружения стыковочного узла, помеченного маркерами специального вида.

Предполагается, что аппарат оснащён всеми необходимыми измерителями параметров движения, вопросы маневрирования в данной работе не рассматриваются. Предварительное наведение на большой дистанции осуществляется с помощью гидроакустического маяка станции.

Возможность использования каскадного детектора и различных опорных маркеров, таких как AruCo, Pi-Tag, CCTag и активного светодиодного маркера для позиционирования АНПА у донного объекта, а также оценка положения аппарата по 4-м точкам исследовалась в предыдущих работах [1, 2].

Цель работы – исследование системы видеопозиционирования АНПА.

Задачи:

* сбор и систематизация информации по подготовке данных и обучению глубокой нейронной сети;
* исследование методов борьбы с переобучением;
* обзор методов построения полётных траекторий;
* расчёт точек первого участка траектории для обхода типа "звёздочка";
* обоснование и расчёт конечного участка траектории для осуществления плавной стыковки;
* моделирование предложенного варианта обхода ДЗС для плоской задачи;
* представление рекомендаций по улучшению точности и стабильности детектирования.

# 1 Исходные данные

В качестве объекта исследования за основу был принят аппарат МТ-2010, разработанный в ИПМТ ДВО РАН [3] для МЧС РФ. Внешний вид АНПА показан на рисунке Рисунок 1.

|  |  |
| --- | --- |
|  |  |

Рисунок 1 – Внешний вид МТ-2010

АНПА используется для обследования небольших районов по широкому спектру исследований. Конфигурация его ДРК включает четыре кормовых маршевых движителя и один вертикальный подруливающий. При этом маневрирование по курсу и дифференту осуществляется за счёт разности скоростей вращения маршевых движителей, а погружение происходит за счёт вертикального подруливающего движителя. Этим достигается высокая маневренность аппарата. В качестве упрощения раскладку движителей примем горизонтальной, а условия видимости во время стыковки считаем идеальными. Выберем горизонтальный угол зрения камеры 65º. Будем решать плоскую задачу. Тактико-технические характеристики приведены в таблице Таблица 1.

Таблица 1 – Тактико-технические характеристики МТ-2010

|  |  |
| --- | --- |
| Параметр | Значение |
| Максимальная рабочая глубина, м | 3000 |
| Вес, кг | 300 |
| Габариты, м | ∅0,45 × 3,0 |
| Скорость, м/с | 0-2,5 |
| Автономность, ч (пробег ~ 100 км); | 20 |
| Энергетика: емкость батареи литий-ионных аккумуляторов, кВт·ч | 2,6 |
| Угол зрения камеры в воде, º | 65 |
| Дальность видимости (идеальные условия), м | 10 |

Донную зарядную станцию будем считать усечённой четырёхугольной пирамидой с верхним и нижним основаниями 1 и 2 м соответственно. Примем допуски по углу входа АНПА в отверстие стыковочного узла равными ±10º.

На рисунке Рисунок 2 приведена последовательность операций, на которые можно разбить исследуемую задачу стыковки.

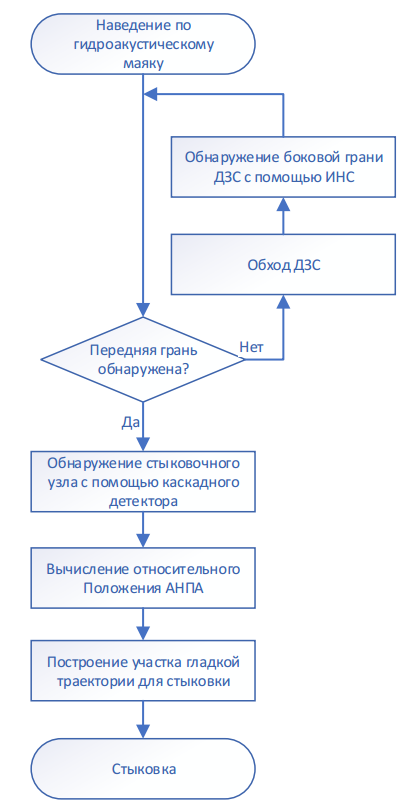


Рисунок 2 – Диаграмма последовательности операций

# 2 Глубокая нейронная сеть

Глубокое обучение — совокупность методов машинного обучения, основанных на обучении представлениям, а не специализированным алгоритмам под конкретные задачи. Машинное обучение используется в областях, не имеющих детерминированного алгоритма решения, таких как распознавание образов в компьютерном зрении.

Отметим, что существует несколько типов задач, связанных с распознаванием образов. Это

* классификация,
* детектирование и
* сегментация.

В задачах классификации требуется определить, присутствует ли на изображении искомый объект или нет. Для каждого изображения определяется вероятность его принадлежности к каждому из выходных классов. При детектировании дополнительно указывается описывающий прямоугольник (т.н. bounding box), показывающий границы объекта на изображении. Сегментация, как показано на рисунке Рисунок 3, призвана определить принадлежность каждого пикселя на изображении к определённому классу.

|  |  |
| --- | --- |
|  |  |

Рисунок 3 – Слева – детектирование, справа – сегментация

В данной научной работе будем решать задачу классификации, в дальнейшем под термином "распознавание" будем иметь в виду именно её.

## 2.1 Выбор архитектуры и типа сети

Полносвязная сеть в задачах распознавания основывается на значениях конкретных пикселей. Для сети типа перцептрона, впервые использовавшейся для распознавания рукописных цифр, два изображения, показанные на рисунке Рисунок 4, абсолютно различны.

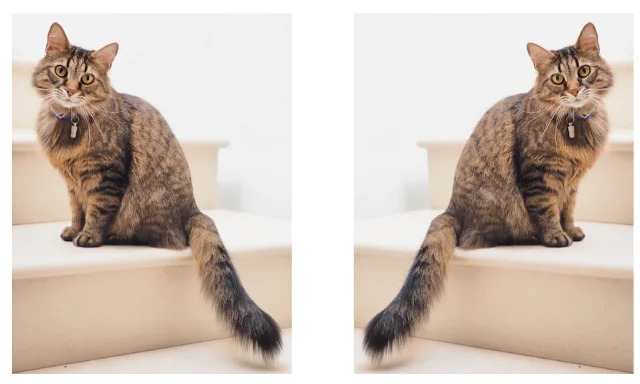


Рисунок 4 – Идентичные с точки зрения человека изображения

Поэтому для задач распознавания образов и классификации изображений используются свёрточные нейронные сети (СНС), способные автоматически выделять существенные признаки на изображении аналогично каскадному классификатору Хаара. СНС основаны на комбинации слоёв свёртки (convolution) и выборки (pooling), как показано на рисунке Рисунок 5. Операция свёртки строит карту для каждого автоматически выбранного признака, а последующая выборка позволяет эффективно уменьшать размерность данных. Последний слой всегда полносвязный, количество выходных нейронов равно количеству детектируемых классов объектов.

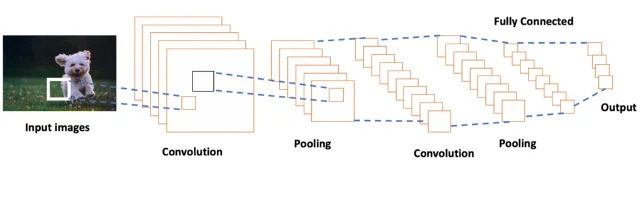


Рисунок 5 – Обобщённая структура свёрточной нейросети

Прежде чем приступить к обучению необходимо определиться с типом и конфигурацией сети. За основу была взята СНС по классификации цветов [4], проектирование и обучение которой показано в официальном примере на странице фреймворка TensorFlow [5]. Модель имеет простейшую структуру, представляющую собой последовательный набор слоёв. Архитектура сети подходит для решаемой задачи, поскольку:

* данная сеть осуществляет распознавание образов на цветных изображениях;
* СНС способна успешно классифицировать 5 видов цветов с точностью до 89% (результат невысокий, поскольку в учебных целях в наборе данных добавлено значительное количество "выбросов");
* подобная точность достигнута на наборе данных в 5 тысяч изображений, а значит, обучение не займёт слишком много времени.

Отметим, что в данной работе в качестве прототипа используется лишь модель, представляющая собой комбинацию слоёв и их функций активации, заданные параметры оптимизатора и начальные значения весов, а не предобученный граф вычислений. Предобученный граф может понадобиться в случае дообучения имеющейся СНС на распознавание дополнительных классов объектов.

Для готовой модели помимо пути в файловой системе к каталогу с изображениями необходимо задать размер входных данных. В используемом прототипе СНС производит масштабирование любого изображения до 180 пикселей по ширине и высоте.

Чем меньше ширина и высота, тем быстрее происходят вычисления в процессе принятия решения нейросетью (т.н. inference), что важно в случае работы в режиме реального времени. Уменьшим фотографию ДЗС в графическом редакторе (см. рисунок Рисунок 6) до масштаба, когда человек ещё может распознать объект на ней.

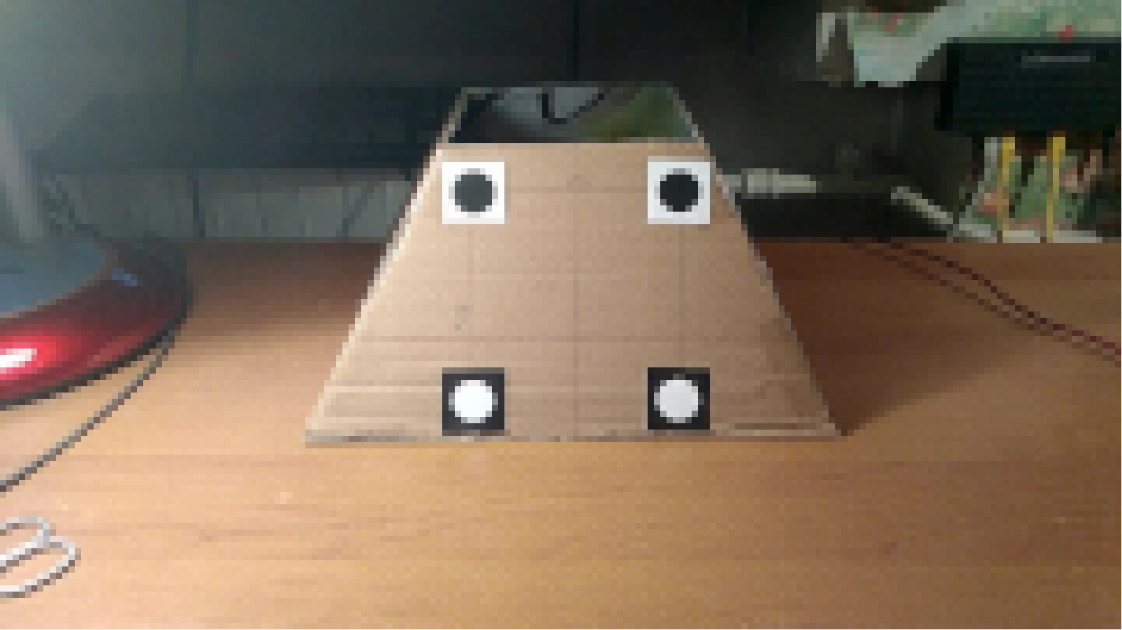


Рисунок 6 – Изображение ДЗС, уменьшенное до разрешения 160 на 90 течек

Итоговая конфигурация нейросети, а также размерности входных данных каждого слоя и количество тренируемых параметров показаны на рисунке Рисунок 7.

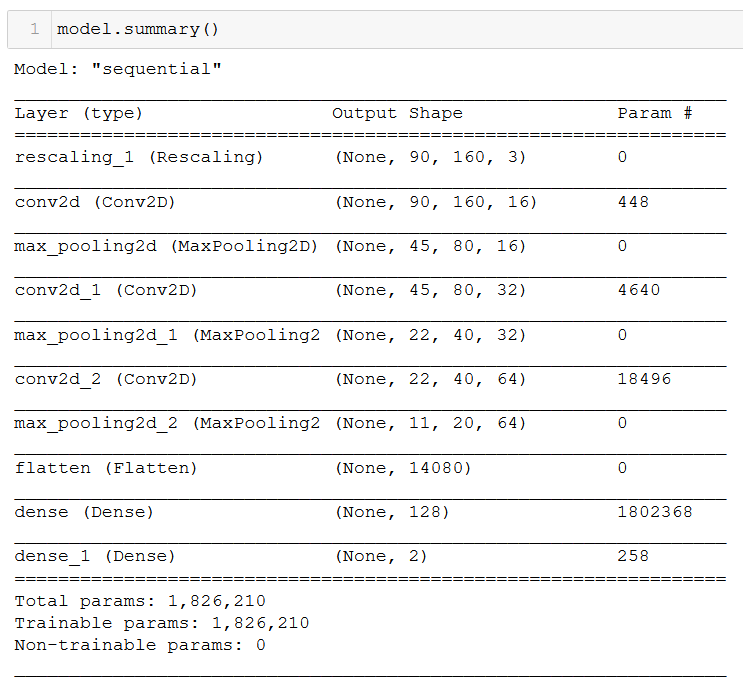


Рисунок 7 – Размерности входных данных каждого слоя

Граф вычислений, полученный с помощью инструмента визуализации обучения TensorBoard [6], показан на рисунке Рисунок 8.

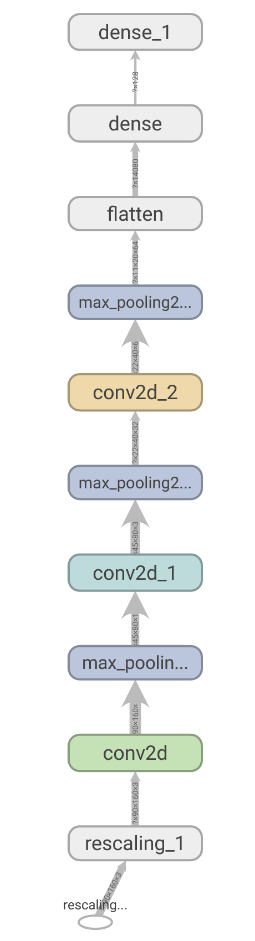


Рисунок 8 – Граф вычислений обучаемой нейронной сети

## 2.2 Подготовка набора данных для обучения

Подготовка набора данных для обучения нейросети существенно проще, чем в случае каскадного детектора. Как было указано выше, для классификации, в отличие от детектирования, изображения не нуждаются в предварительной разметке. Действуя аналогично [2], заснимем тестовые видео с объектами распознавания, затем извлечём отдельные кадры с помощью бесплатной утилиты “Free Video to Jpeg Converter”. Подобный подход позволит существенно сэкономить время, а также одновременно учесть все негативные эффекты, такие как размытия, расфокусировка, возникающие во время движения реального робота.

В отличие от каскадного детектора, нейросети не нужен набор данных, не содержащих объекты распознавания. Данный класс можно создать программно самостоятельно, указав пороговое значение достоверности, например, 85%, при распознавании на уже обученном графе вычислений. Таким образом, изображения, набравшие вероятность принадлежности менее 85% к любому классу, автоматически относятся к фоновым.

## 2.3 Метрики качества и борьба с переобучением

Запустим обучение СНС как показано на рисунке Рисунок 9.

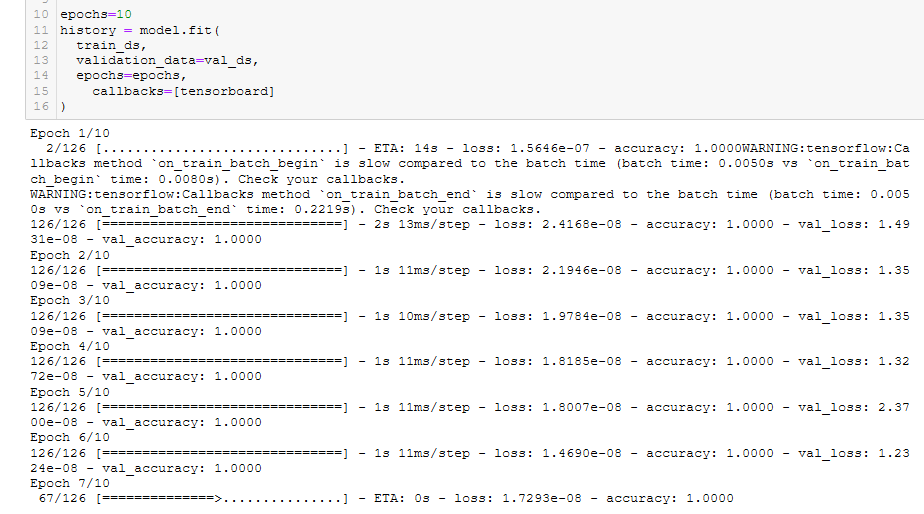


Рисунок 9 – Процесс обучения нейронной сети в среде Jupyter Notebook

Сформированных набор данных был предварительно поделён на обучающий и тестовый в соотношении 80/20. Обучение производится методом градиентного спуска [], основная цель в процессе обучения – минимизировать функцию потерь. Всего задано 10 эпох обучения. На рисунке Рисунок 9 показаны графики функции точности и функции потерь на обучающем и тестовом наборах соответственно.

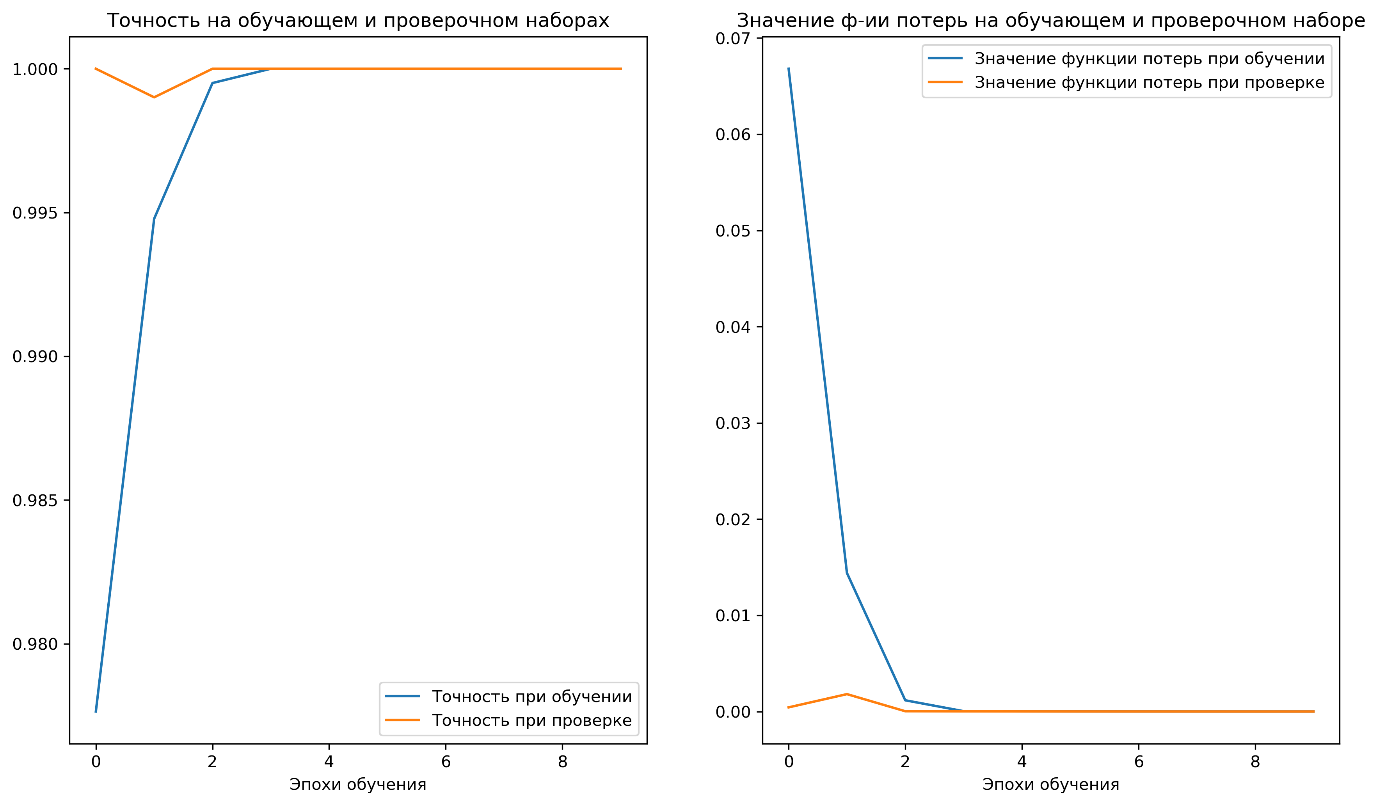


Рисунок 10 – Полученные графики точности и функции потерь на обучающим и проверочном наборах

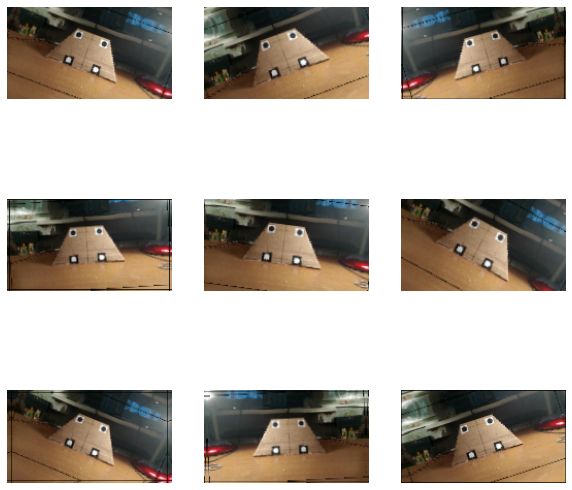


Рисунок 11 – Искусственное наполнение данных

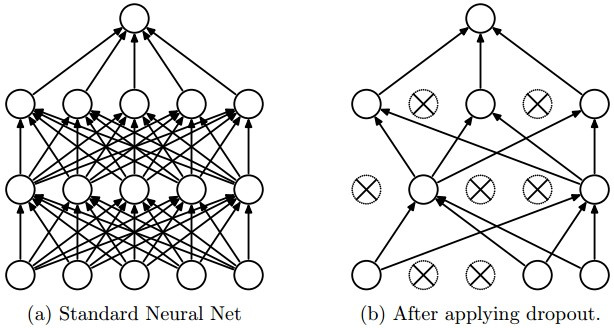


Рисунок 12 – Принцип действия dropout

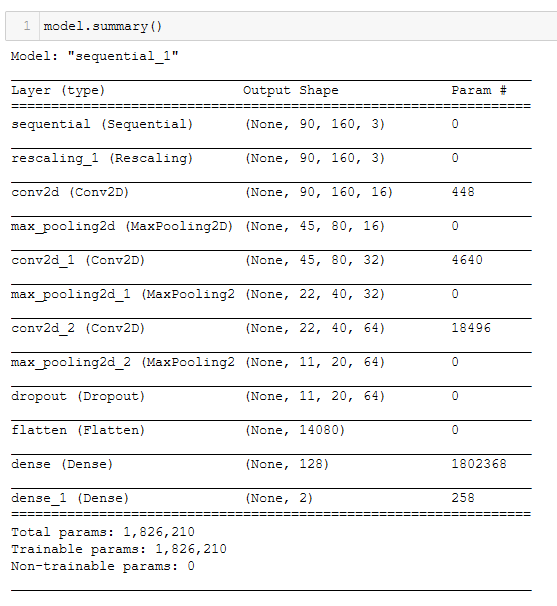


Рисунок 13 – Параметры модели с новым слоем Dropout

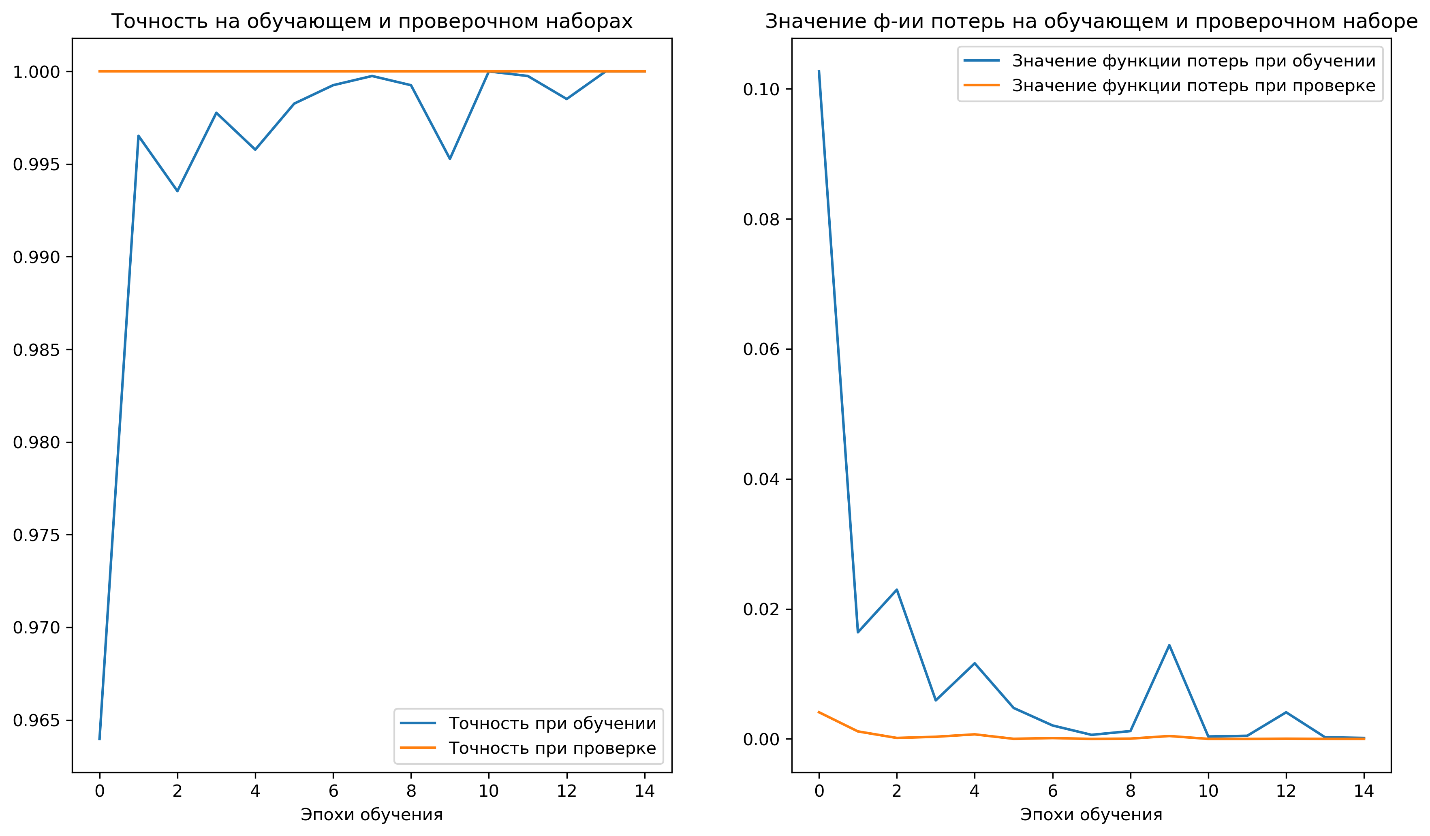


Рисунок 14 – Итоговые графики функции обучения и функции потерь на обучающем и тестовом наборах

# 3 Построение траектории обхода и стыковки

## 3.1 Траектория обхода донной станции

Введём систему координат OZX, связанную с центром нижнего основания ДЗС, как показано на рисунке Рисунок 13, а также определим граничные окружности перемещения центра масс и носа аппарата исходя из условия дальности видимости, приведённого в пункте 1.

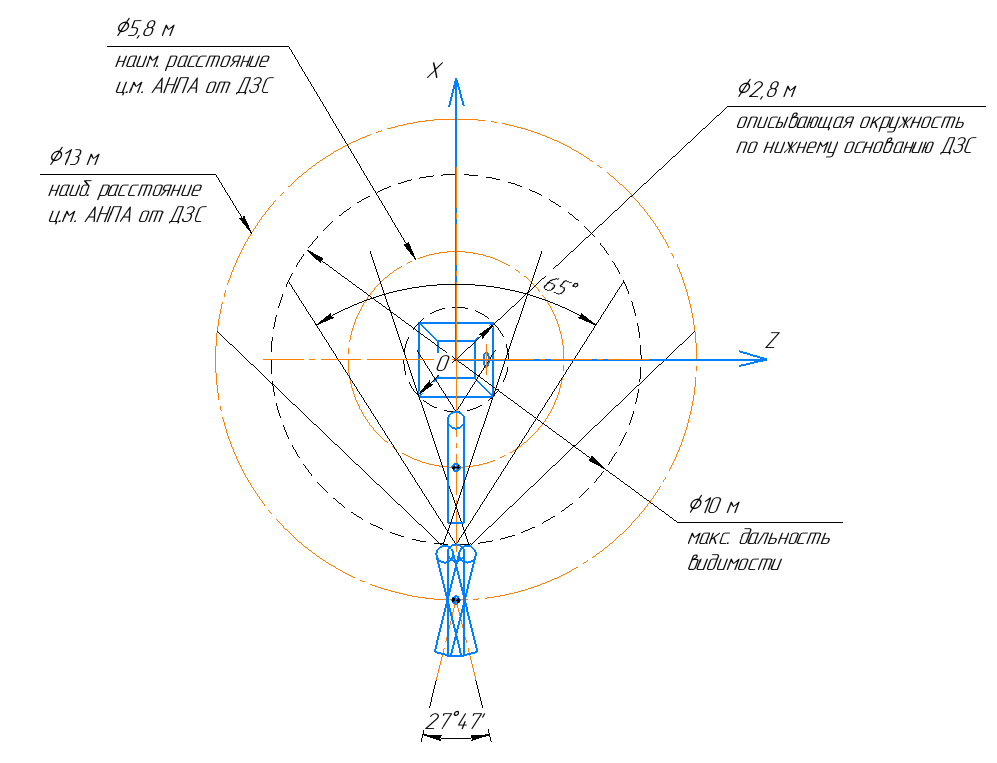


Рисунок 15 – Введённая местная система координат OZX, связанная с ДЗС

Изначально предполагалось использовать полётные траектории, описанные, например, в [7, 8]. Пример полётной траектории, задающей разворот, показан на рисунке Рисунок 14.

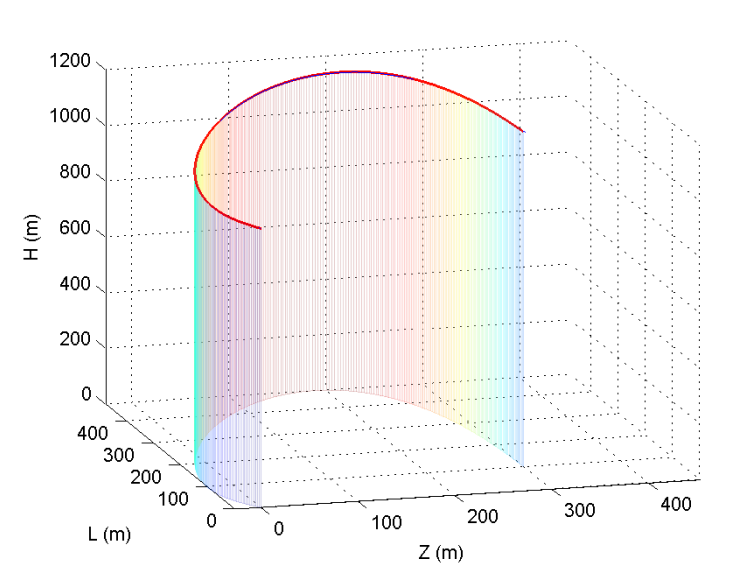


Рисунок 16 – Пример полётной траектории для осуществления разворота

Однако в случае подобного разворота произойдёт потеря ДЗС из видимости. К тому же, нет возможности заранее рассчитать полный угол разворота, поскольку при монокулярном зрении одновременно видны лишь 2 грани станции, 2 оставшиеся – скрыты.

Следовательно, сформулируем задачу следующим образом: необходимо совершить обход ДЗС на угол не менее 180º, теряя её из области видимости как можно реже и на как можно меньшие промежутки времени.

Из рисунка Рисунок 13 видно, что наибольший угол поворота по курсу с учётом отдаления АНПА составляет примерно 27º. Предложим траекторию как показано на рисунке Рисунок 15. Предполагается осуществлять смещение на 35º52' сектор. Вблизи ДЗС не будет помещаться в область видимости видеокамеры, но на максимальной дистанции маневрирование построено так, чтобы не терять её из кадра. Сценарий на языке python для расчёта координат точек в зависимости от заданных сектора смещения и диаметров окружностей, ограничивающих перемещение центра масс АНПА приведена в приложении А.

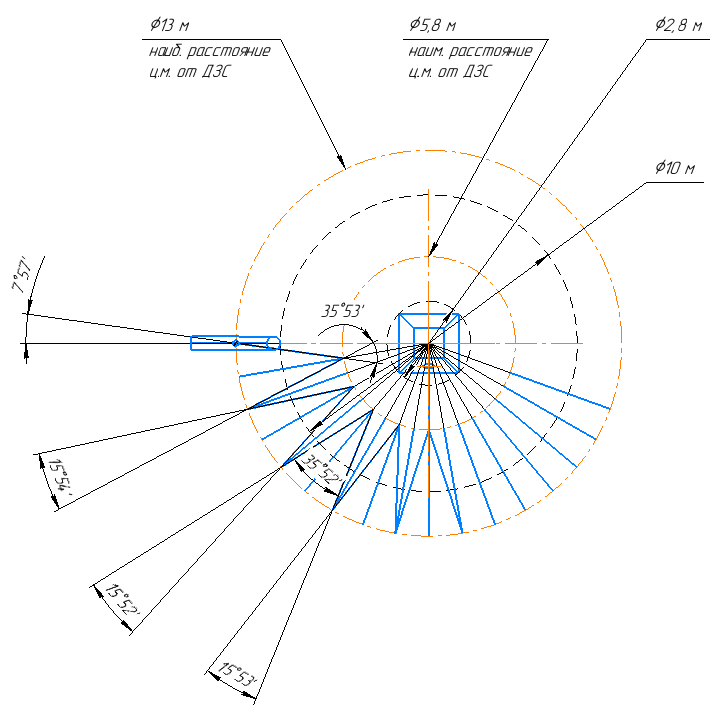


Рисунок 17 – Траектория для обхода ДЗС

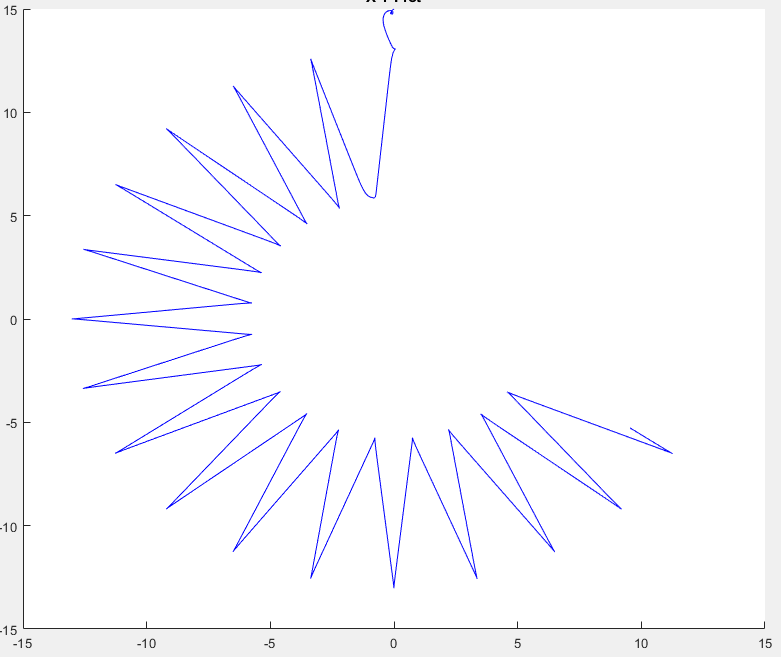


Рисунок 18 – Траектория движения АНПА по расчётам математической модели

## 3.2 Траектория стыковки с донной станцией

В работе [7] отмечалось, что реализация движения вдоль заданного пути в пространстве может осуществляться в рамках путевой или траекторной стабилизации. Эти задачи различаются тем, что при путевой стабилизации не контролируется время движения по траектории, а само движение планируется, как правило, с постоянной скоростью.

Поскольку в данной задаче требований ко времени осуществления манёвра не предъявляется, будем решать задачу путевой стабилизации как более простую.

Предположим, что стыковочный узел находится на положительной полуоси OZ, как показано на рисунке Рисунок 13. Выберем класс кривых второго порядка вида

,

поскольку они имеют асимптоту в виде горизонтальной оси и обеспечат плавную стыковку. На рисунке Рисунок 17 показано семейство c областью определения *x ≥ 1*кривых для различных значений параметра *k*.

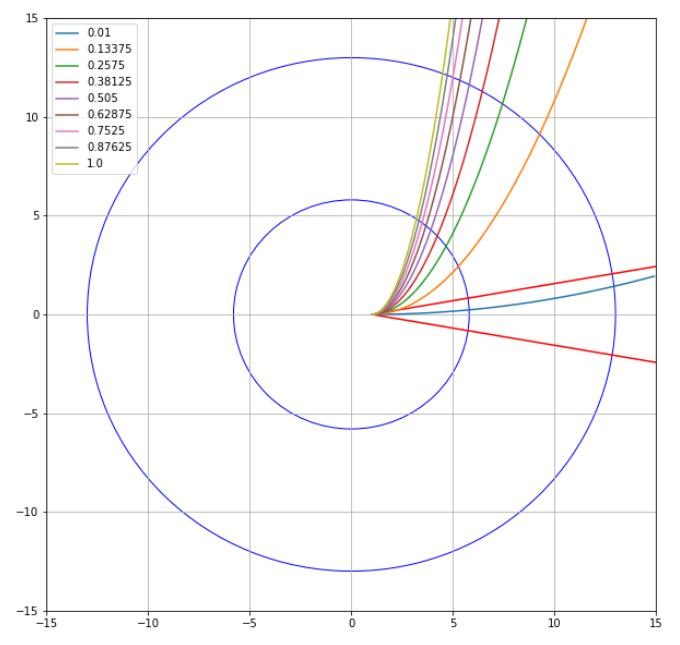


Рисунок 19 – Кривые второго порядка с различными коэффициентами при старшей степени. Красным показан сектор ±10º, обозначающий допуски входа в стыковочный узел

# ЗАКЛЮЧЕНИЕ

В результате проведённого исследования была исследована возможность применения

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Андреев Е.В. Разработка и исследование метода позиционирования подводного аппарата у донного объекта с использованием маркеров специального вида: НИР/ Андреев Евгений Викторович. – Москва, 2019. – 40 с.
2. Андреев Е.В. Использование каскадного детектора для построения системы позиционирования подводного аппарата: НИР/ Андреев Евгений Викторович. – Москва, 2019. – 68 с.
3. Борейко А.А., В.Е. Горнак, С.В. Мальцева, Ю.В. Матвиенко, Д.Н. Михайлов. Малогабаритный многофункциональный автономный необитаемый подводный аппарат «МТ-2010». Подводные исследования и робототехника, №2. С. 37. 2011.
4. Страница примера классификации изображений цветов с официального сайта фреймворка TensorFlow [электронный ресурс]. URL: <https://www.tensorflow.org/tutorials/images/classification> (дата обращения: 01.11.2020).
5. Официальный сайт фреймворка TensorFlow [электронный ресурс]. URL: <https://www.tensorflow.org/> (дата обращения: 01.11.2020).
6. Репозиторий инструмента визуализации процесса обучения нейронной сети TensorBoard [электронный ресурс]. URL: https://github.com/tensorflow/tensorboard (дата обращения: 01.11.2020).
7. Ткачёв С.Б., Крищенко А.П., Канатников А.Н. Автоматическая генерация пространственных траекторий БПЛА и синтез управлений. Математика и математическое моделирование. МГТУ им. Н.Э. Баумана. 2015. №1. С. 1 – 17.
8. Yasmina Bestaoui. 3D flyable curves for an autonomous aircraft. AIP Conference Proceedings 1432. 2012. № 132.

# Приложение А. Листинг кода обучения нейронной сети

#!/usr/bin/env python

# coding: utf-8

import pathlib

import datetime, os

import matplotlib.pyplot as plt

import numpy as np

import os

import PIL

import tensorflow as tf

gpus = tf.config.experimental.list\_physical\_devices('GPU')

if gpus:

try:

for gpu in gpus:

tf.config.experimental.set\_memory\_growth(gpu, True)

except RuntimeError as e:

print(e)

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential

from tensorflow.keras.callbacks import TensorBoard

tensorboard = TensorBoard(log\_dir='./logs', histogram\_freq=0,

write\_graph=True, write\_images=False)

print(tf.\_\_version\_\_)

# ## Explore the dataset

data\_dir = pathlib.Path(r'C:\Users\corsair\.keras\datasets\img\_source\_root')

image\_count = len(list(data\_dir.glob('\*/\*.jpg')))

print(image\_count)

roses = list(data\_dir.glob('zdun/\*'))

PIL.Image.open(str(roses[0]))

# # Load using keras.preprocessing

# ## Create a dataset

# Define some parameters for the loader:

batch\_size = 32

img\_height = 90 #64

img\_width = 160 #128

# It's good practice to use a validation split when developing your model. Let's use 80% of the images for training, and 20% for validation.

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

data\_dir,

validation\_split=0.2,

subset="training",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size)

val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

data\_dir,

validation\_split=0.2,

subset="validation",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size)

# You can find the class names in the `class\_names` attribute on these datasets. These correspond to the directory names in alphabetical order.

class\_names = train\_ds.class\_names

print(class\_names)

# ## Visualize the data

# Here are the first 9 images from the training dataset.

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))

for images, labels in train\_ds.take(1):

for i in range(9):

ax = plt.subplot(3, 3, i + 1)

plt.imshow(images[i].numpy().astype("uint8"))

plt.title(class\_names[labels[i]])

plt.axis("off")

for image\_batch, labels\_batch in train\_ds:

print(image\_batch.shape)

print(labels\_batch.shape)

break

# ## Configure the dataset for performance

AUTOTUNE = tf.data.experimental.AUTOTUNE

train\_ds = train\_ds.cache().shuffle(1000).prefetch(buffer\_size=AUTOTUNE)

val\_ds = val\_ds.cache().prefetch(buffer\_size=AUTOTUNE)

# ## Standardize the data

# The RGB channel values are in the `[0, 255]` range. This is not ideal for a neural network; in general you should seek to make your input values small. Here, you will standardize values to be in the `[0, 1]` range by using a Rescaling layer.

normalization\_layer = layers.experimental.preprocessing.Rescaling(1./255)

# Note: The Keras Preprocessing utilities and layers introduced in this section are currently experimental and may change.

# There are two ways to use this layer. You can apply it to the dataset by calling map:

normalized\_ds = train\_ds.map(lambda x, y: (normalization\_layer(x), y))

image\_batch, labels\_batch = next(iter(normalized\_ds))

first\_image = image\_batch[0]

# Notice the pixels values are now in `[0,1]`.

print(np.min(first\_image), np.max(first\_image))

# ## Data augmentation

data\_augmentation = keras.Sequential(

[

layers.experimental.preprocessing.RandomFlip("horizontal",

input\_shape=(img\_height,

img\_width,

3)),

layers.experimental.preprocessing.RandomRotation(0.1),

layers.experimental.preprocessing.RandomZoom(0.1),

]

)

# Let's visualize what a few augmented examples look like by applying data augmentation to the same image several times:

plt.figure(figsize=(10, 10))

for images, \_ in train\_ds.take(1):

for i in range(9):

augmented\_images = data\_augmentation(images)

ax = plt.subplot(3, 3, i + 1)

plt.imshow(augmented\_images[0].numpy().astype("uint8"))

plt.axis("off")

# ## Dropout

num\_classes = 2

model = Sequential([

data\_augmentation,

layers.experimental.preprocessing.Rescaling(1./255),

layers.Conv2D(16, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(32, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(64, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Dropout(0.2),

layers.Flatten(),

layers.Dense(128, activation='relu'),

layers.Dense(num\_classes)

])

# ## Compile the model

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

# ## Model summary

model.summary()

# ## Train the model

epochs = 15

history = model.fit(

train\_ds,

validation\_data=val\_ds,

epochs=epochs

)

# ## Visualize training results

acc = history.history['accuracy']

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

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(epochs)

plt.figure(figsize=(14, 8), dpi=300)

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Точность при обучении') #) label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Точность при проверке')# label='Validation Accuracy')

plt.legend(loc='lower right')

plt.xlabel('Эпохи обучения')

plt.title('Точность на обучающем и проверочном наборах') #'Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Значение функции потерь при обучении')# label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Значение функции потерь при проверке')# label='Validation Loss')

plt.legend(loc='upper right')

plt.xlabel('Эпохи обучения')

plt.title('Значение ф-ии потерь на обучающем и проверочном наборе')# 'Training and Validation Loss')

plt.show()

# ## Predict on new data

img\_path = 'E:/University/11sem/nirs/NeuralNet/pyramid0639.jpg'

img = keras.preprocessing.image.load\_img(

img\_path, target\_size=(img\_height, img\_width)

)

img\_array = keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0) # Create a batch

predictions = model.predict(img\_array)

score = tf.nn.softmax(predictions[0])

print(

"This image most likely belongs to {} with a {:.2f} percent confidence."

.format(class\_names[np.argmax(score)], 100 \* np.max(score))

)

model.save('docking\_model')

# Приложение Б. Программа для расчёта координат точек траектории

import cmath

import math

from matplotlib import pyplot as plt

# Диаметры, ограничивающие перемещение центра масс

D\_mass\_center\_max = 13

D\_mass\_center\_min = 5.8

angular\_delta = 15

angle = 90

def real\_to\_math\_yaw(yaw):

"""returns yaw in Degrees"""

math\_yaw = -1 # 360 - yaw

yaw %= 360

if yaw >= 0 and yaw <= 90:

math\_yaw = 90 - yaw

elif yaw > 90 and yaw < 180:

delta = yaw - 90

math\_yaw = 360 - delta

elif yaw >= 180 and yaw < 270:

delta = yaw - 180

math\_yaw = 270 - delta

else:

delta = yaw - 270

math\_yaw = 180 - delta

return math\_yaw

def real\_to\_math\_yaw\_2(yaw):

yaw %= 360

return (360 + 90 - yaw) % 360

x\_outer = [] # Z AUV

y\_outer = [] # X AUV

x\_inner = []

y\_inner = []

N = 360 // angular\_delta

for i in range(N):

# в полярных координатах

p1 = cmath.rect(D\_mass\_center\_max, math.radians(angle))

p2 = cmath.rect(D\_mass\_center\_min, math.radians(angle + angular\_delta/2))

x\_outer.append(p1.real)

y\_outer.append(p1.imag)

x\_inner.append(p2.real)

y\_inner.append(p2.imag)

#print(i, round(x\_outer[i], 2), round(y\_outer[i], 2), round((angle % 360) , 3), sep='\t')

angle += angular\_delta

print(len(x\_outer), len(y\_outer))

x\_final = []

y\_final = []

direction = []

for i in range(len(x\_outer)):

x\_final.append(x\_outer[i])

x\_final.append(x\_inner[i])

direction.append("foreward")

y\_final.append(y\_outer[i])

y\_final.append(y\_inner[i])

direction.append("backward")

from pprint import pprint

#for elem in zip(x\_final, y\_final, direction):

# print(round(elem[0], 2), round(elem[1], 2), elem[2][0] , sep='\t')

plt.figure(figsize=(8, 8))

plt.plot(x\_final, y\_final, color='c')