ICTE4.3

Vitalijs Cehs

November 2020

AALBORG UNIVERSITY
COPENHAGEN

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Semester Coordinator: Henning Olesen

Secretary: Maiken Keller

Semester:
ICTE4.3

Title:
Underwater Object Recognition with a Remotely Operated Vehicle

Project Period:
Winter Semester 2020

Semester Theme:

Advanced ICT systems

Supervisor(s):
Sokol Kosta

Project group no.:
4.3

Members
(do not write CPR.nr.):

20143513 Vitalijs Cehs

Abstract:

The aim is to develop an object detection system that would perform real-time underwater analysis to detect and identify objects of interest within the underwater environment by utilising an underwater drone BlueROV2. The system utilises a combination of Python, TensorFlow, OpenCV, YOLOv3 neural network model and BlueROV2. Testing of the prototype system is performed by a series of staged experiments.

Pages: 56
Finished:
November 30, 2020

When uploading this document to Digital Exam each group member confirms that all have participated equally in the project work and that they collectively are responsible for the content of the project report. Furthermore each group member is liable for that there is no plagiarism in the report.

Contents

List of acronyms and abbreviations

AAU Aalborg University
AI Artificial Intelligence
AP Average Precision
API Application Programming Interface
AUV Autonomous Underwater Vehicle
CNN Convolutional Neural Network
COCO Common Objects in Context
CPU Central Processing Unit
CSV Comma-Separated Values
CV Computer Vision
DNN Deep Neural Network
DSV Deep-Submergence Vehicle
FPS Frames Per Second
FRS Filter-based Restoration
GPU Graphics Processing Unit
GRU Gated Recurrent Units
HD High Definition
IoU Intersection over Union
J-EDI JAMSTEC E-Library of Deep-sea Images
JAMSTEC Japan Agency for Marine Earth Science and Technology
LSTM Long Short-Term Memory
mAP mean Average Precision
MBARI MontereyBay Aquarium Research Institute
ML Machine Learning
MLaaS Machine Learning as a Service
MS Microsoft
NMS Non-Maximum Suspension
NN Neural Network
OS Operating System
PC Personal Computer
RCNN Region based Convolutional Neural Network
ReLU Rectified Linear Unit
RNN Recurrent Neural Network
RoI Regions Of Interest
ROV Remotely Operated Vehicle
RPI Raspberry Pi
RPN Region Proposal Network
SDG Sustainable Development Goals
SDK Software Development Kit
SRS Software Requirement Specifications
SSD Single Shot Detector
SVM Support Vector Machine
TF TensorFlow
TPU Tensor Processing Unit
UDP User Datagram Protocol
UI User Interface
WWF World Wildlife Fund
XML Extensible Markup Language
YOLO You Only Look Once

1 Introduction

The Artificial Intelligence (AI) and Machine Learning (ML) fields have gained huge popularity in recent years. And it will not stop there as the advances in hardware and computing power will keep pushing the progress even further, allowing for broader industries' implementations of autonomous systems. As a matter of fact, according to Gartner, AI advancement will generate a $2.9 trillion of commercial value in 2021 [1], however apart from solely contributing to the economic value, machine learning can be a powerful tool in tackling societal and environmental problems, such as pollution and climate change [2].

As ML is a hot topic, so is a topic of sustainability. Denmark is making an effort to become a greener country, by taking part in following the 17 Sustainable Development Goals (SDG) proposed by the United Nations to achieve sustainability by 2030 [3]. Among the 17 SDG's, there is a particular one of number 14 - Life Below Water, which aims at reducing water pollution, among others, as around 8 million tons of plastic ends up in the Ocean every year (60-80% of all garbage debris). Around 80% of all marine trash originates from land and waterways connected to the Ocean, such as rivers and canals [4]. 38% of waste in the waterways are consumer-related plastics, such as drink bottles and wrappers [5].

One of the contributions to this goal has been made by GreenKayak, Denmark based non-profit, with an attempt to reduce waste volumes in Copenhagen harbour by lending kayaks for free in return to collected garbage [6]. Recently, 2020 Ren Havn event (Danish for "clean harbour") took place aiming at raising awareness and collecting trash from the Copenhagen canal, where approximately 120 kg of trash was collected by 80 volunteers in collaboration with various organisation contributing with kayaks to gather the waste(Kayak Bar, Kayak Republic, Green Kayak, GoMore and Goboat)[7].

Another example of contributing action is the initiative made by the city authority in collaboration with World Wide Fund (WWF) and canal tour boat operator Stromma to install "floating garbage bins" called Sebians [8]. These devices can suck litter into a catch bag using a submersible water pump that has a capacity of 25,000 liters per hour and can hold up to 1.5 kilos of waste. It is expected to collect around 1.8 tons of waste annually, where the most often caught objects are cigarettes butts (29% of all rubbish caught), plastic fragments (28%), food wrappers (26%), foam particles (5%), bottle caps (4%), straws (2%), and cans and plastic bags (1%) [9]. However, how can a pressing issue of water pollution be improved without hand-picking and identifying every particle of litter manually? The motivation for the project is to research the possible ways of how to contribute to the solution.

The project is tapping into the field of machine learning to explore the problem of object detection task in the underwater environment. Even though computer vision has been successfully applied underwater for inspection, monitoring and mapping, object detection has not been researched well enough [10]. The goal is to equip an underwater drone with a neural network model for object detection capabilities to identify litter objects in real-time. The idea includes experimenting on different approaches and architectures to propose a working prototype system to identify, recognize and detect objects of interest in the sub-water context.

The above stated leads to the problem space of the project that is defined in the problem formulation below. Additionally, three sub-research questions are presented to break down

the entire problem into fewer smaller areas to address the project.

1.1 Problem Formulation

"How to develop an object detection system that would perform real-time underwater analysis to detect and identify objects of interest within the underwater environment?"

The following sub questions were formed to guide the research and development process for the project and proposed prototype system.

- How can machine learning algorithms be utilized to perform underwater analysis?
- How to develop a system architecture that would employ underwater drone for the object detection task?
- How to approach the problem of computer vision in rough underwater environment?

1.2 Report Structure

In the following section the structure of the report is presented, explaining the reader what can be expected in each chapter and the flow of the project. The report starts with Introduction of the topic of artificial intelligence and machine learning, followed by the motivation for the project and problem formulation. Consecutive chapter of Methodology depicts the process model of the research and development, describing how the research was conducted. As well as the chapter explains the interconnection of the system components and the process of evaluation of the system. Subsequently, moving on to the State of the Art chapter, it describes the underwater vehicles types, outlines the research on machine learning, object detection, neural networks and existing solutions of similar systems. The following Analysis chapter narrows down the findings of the previous chapter to decide upon the choice of the algorithm most preferred for the project. As well as draws upon data preparation and requirement specifications for the system prototype. System design chapter presents the system overview and component intercommunication, followed by the Implementation chapter, where water drone assembly and model implementation are described. Finally, after experiments' and testing stages are laid out, there is a discussion on the results and future improvements, followed by the Conclusion on the problem formulation.

2 Methodology

The chapter describes the process of conducting the project, providing an overview of methodological approaches taken as well as the process behind conducting analysis and the development of the prototype. The details are summarised and described in the following process model and secondary research chapters.

2.1 Process Model

The point of departure was to establish the road-map for the project. To do so an appropriate process model had to be chosen to aid in identifying the scope of the research and the development goals for the prototype. Since the project is ought to be experimental and analytical i.e. with no user involvement it was decided to derive the requirements from the research.

Therefore, two methodologies served as an inspiration for the process model, namely Iterative Waterfall model [11] and Prototyping method [12], as it is necessary to scrutinize the best suitable system design for the task at hand (as explained in Chapter 4).

The traditional Waterfall model [13] follows a linear development sequence, where each subsequent phase is completed before the next phase begins. The phases include Requirements, Design, Implementation, Testing and Maintenance. Each phase is comprised of different tasks and activities, which cannot be altered after the phase is completed. Compared to classic Waterfall, iterative model provides room for making changes in a form of feedback paths from each phase to the preceding phase. In this case if an error was made during some stage of the project it can be easily traced back and corrected.

The approach of Prototyping suggests first, to determine the objectives, then to follow the cycle of three stages *Develop*, *Refine*, *Demonstrate* with multiple iterations through the cycle to achieve desired outcome. The final stages include testing step and implementation of the prototype system, as shown in Figure 1.

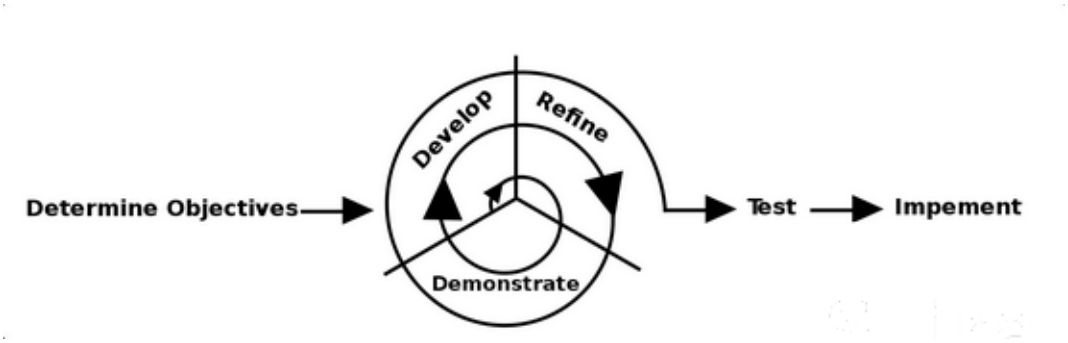


Figure 1: Prototyping method. Taken from [14]

Two methods were combined and modified to address the needs of the project associated with both prototype development and report writing. The resulting process diagram is presented in Figure 2 illustrating the flow of work.

The approach taken was to split the entire workflow into various areas this project is touching upon. The identified research areas include examination of different underwater vehicles, research on neural networks and machine learning algorithms, research on implementation frameworks and available tools, as well as different existing approaches and state of the art technologies that are already adopted.

The initial phase was to iterate between research and analysis to gather knowledge on the topics related to the project. As an outcome a set of requirements has been identified and different designs of the systems were recognized.

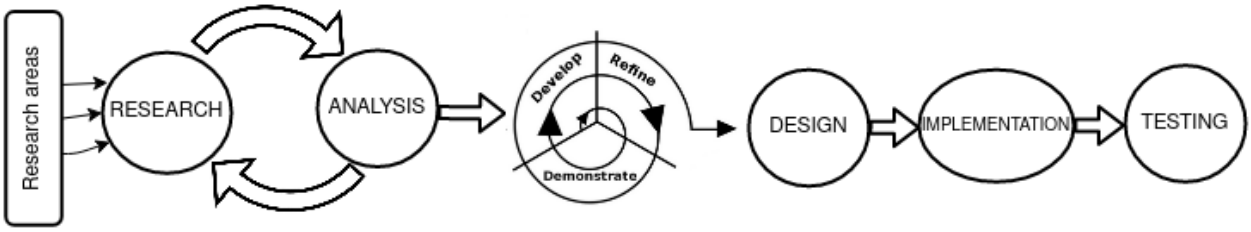


Figure 2: Overview of the Process Model

For the next phase, project development began with the Develop, Refine and Demonstrate cycle, as can be seen in Figure 2. For this step a hands on approach was carried out to learn about machine learning and different implementation frameworks. For this purpose a cloud setup was utilised to get acquainted with the flow of training, validating and observing the working principles of ML models. The following iterations included implementation of the models on a local machine and on EDGE GPU device, to conceptualize on the most suitable system design. Various iterations helped to refine the existing requirements and gather additional ones.

The next phase follows with Design, Implementation and Testing. Where first consists of the outline of the chosen design, where the overall system architecture is presented together with the system components and relationship between them. Second delves into the implementation process of the chosen framework for the prototype. Finally, third phase includes evaluation of the prototype system, with reflection on the results and potential improvements to the system.

2.2 Secondary Research

The goal of the secondary research was to gather materials associated with the problem formulation of the report and the scope of the project. In order to find relevant articles, white papers and books techniques such as keyword search and cross referencing were used. The tools that were utilised include AAU Library [15] and Google Scholar [16]. The list of keywords was composed based on the project scope and defined research questions. Exam-

ples of keywords include neural networks, computer vision, image classification and object detection.

2.2.1 Data Collection

The aim of the project was to identify the litter objects in the underwater environment with the aid of Machine Learning. For the purpose to train a model for this task relevant data sets have been identified with the research on existing solutions and by separate search on the web. It is worth to mention that the decision in the beginning of the project was not to create data from scratch but rather find a suitable data sets instead. The findings were summarized and presented in the Chapter 4.3, describing the usability of each data set as well as considerations on the relevancy to the project.

2.3 Requirements and Prioritization

To aid in the process of system development and to define the desired functionality of the prototype, Software Requirement Specifications (SRS) technique was utilised. Requirements were mainly derived from the secondary research and observations of conducted experiments. For the final solution requirements have been adjusted through various iterations of experimenting and research. The obtained requirements served as guidelines for the prototype development decisions specifically in Chapter 5 and Chapter 6, where the final system solution is proposed. Moreover, typical software development requirements fall into two categories:

- Functional - *"requirement that specifies a function that a system or system component shall perform"* [17, p. 195]
- Non-functional - *"software requirement that describes not what the software will do but how the software will do it"* [17, p. 293]

Additionally, requirements have been prioritized with the MoSCoW method [18]. It suggests to classify the set of generated requirements by importance using a four level scheme which is *Must*, *Should*, *Could* and *Won't*. The prioritization is needed due to time constraints posed on the development process, where tho focus is on developing the main functionality first to achieve a working system.

Must have are considered essential for the system and have to be included in the final prototype, otherwise if at least one is not delivered, the whole system can be considered as a failure.

Should have are considered as important as *Must* but can be postponed due to time constraints during the development making the system still viable.

Could have are considered desirable but less important than *Should*. If the development time-frame is tight, requirements from this category are considered first to be left out.

Won't have are considered not important and can be excluded from the prototype development, however, they should be addressed in later stages of development as future improvements.

Finally, a table was composed with the specified requirements that can be seen in Chapter 4.5. The table consists of four columns that show **ID** - sequence number of the requirement, **Description** - defines the requirement, **Source** - points to chapters where the requirement comes from and **MoSCoW** prioritization - importance of the requirement.

2.4 Testing and Validation

In order to finalize the system solution and evaluate its performance, testing phase should be conducted on the developed prototype. It was decided to carry out multiple testing scenarios to analyze behaviour of the system in different environmental conditions. The experiments ought to be performed in artificial setting (laboratory) and in a realistic setting in a real-world environment.

First of all, the staged experiments were carried out to generalize on the developed prototype operation and decide whether its performance is satisfactory to be executed in the real world. For this case three scenarios were introduced posing different level of constraints on the developed system. First scenario depicts general operation of the detector system without any restrictions. Meaning the system is tested above water without interference of image quality and light conditions. Second scenario includes application of slight constraints where the system is introduced to the water context and is tested in the clear water setting. Third performed experiment poses high constraints on the system where the water turbidity is high as well as low light conditions apply.

It is worth to mention that the system was tested with two different models utilised for object detection, one of which is YOLOv3 and the second is YOLOv3-tiny. Both models have been evaluated and the results compared in Chapter 7.2. Additionally, it should be said that the system did not undergo the real world experiment stage due to limitations described in Chapter 7.1.

3 State-of-The-Art

In this section main research topics are presented, which consist of three parts, namely research on Underwater Vehicles, Machine Learning and Existing Solutions. The first part describes different types of vehicles in the underwater context as well as background and purpose of their usage.

Machine Learning topic is dealing with identification of the available approaches, algorithms and tools that potentially could be used to bring intelligence to the device, to achieve the goal of detecting and identifying the objects of interest underwater.

Finally, research on existing projects is presented in order to find out how the problem of underwater object detection is dealt with, which technologies are utilized and identify the challenges that may appear in such projects.

The aim is to establish the general knowledge needed to shape the project. Together they form the base for the different parts of the project and guide the reader through the project's development flow.

3.1 Underwater vehicles

Underwater vehicles are generally classified into three categories which are DSVs, AUVs and ROVs. The choice of particular vehicle over another depends on the given mission. The tasks vary according to the mission requirements, such as maximum depth, endurance in range and time, and needed sensors; the optimal human role in command and control; available power; cost and a number of other criteria considered for the successfully fulfilling the mission goals [19, p. 18-19]. Following in the section each type of vehicles are presented, with few examples from every category. Additionally, Table 1 is provided for a quick comparison overview of the devices and a visual summary of the findings.

	DSVs	ROVs	AUVs
DEFINITION	Untethered, human-occupied, free-swimming, undersea vehicle	Tethered, self-propelled vehicle with direct real-time control	Untethered undersea vehicle, may be totally reprogrammed and equipped with decision aids to operate autonomously; or operation may be monitored and revised by control instructions transmitted by a data link.
DEPTH	Many to 1,000 m Few to 3,000 m Very few to 6,000 m One to 6,500 m	Very many to 500 m Many to 2,000 m Few to 3,000 m Few to 6,000 m One to 11,000 m	Several to 1,000 m Few to 3,000 m Very few to 6,000 m
ENDURANCE			
Time	Normally 8 hours, 24 to 72 hours max	Indefinite, depending on reliability and operator endurance	6 to 48 hours of propulsion May sit on bottom for extended periods
Range	<50 km	Limited in distance from host ship by tether	350 km demonstrated; near-term potential 1,500 km, depending on energy source
PAYLOAD	1 to 3 people, 45 to 450 kg (100 to 1,000 lb); adaptable to tools and sensors	45 to 1,590 kg (100 to 2,000 lb); adaptable to tools and sensors	11 to 45 kg (25 to 100 lb); adaptable to measuring equipment, tools, and sensors

SUPPORT			
Ship	Most DSVs require large ship support; ship size varies with DSV size	Depends on ROV size and mission requirements	Medium—depends on AUV size and mission requirements
Handling Systems	Depend on DSV size	Depends on ROV size	Similar to ROVs, depending on AUV size
Navigation Systems	Relative to seafloor or surface vessel	Relative to surface/seafloor	Seafloor and inertial navigation
STRENGTHS	Direct human observation and manipulation	Real-time feedback to operator, long endurance capability, low- cost per operating hour	Potential for automated operations, ability to operate with or without human command and without tether; minimum surface support
LIMITATIONS	Real-time feedback to controller Large size, weight, and cost due to manned requirements	Tether cable potentially limits maneuverability and range	Energy supply
	Limited mission time Potential personal hazards		Bandwidth of data link Capacity of internal recorders Limited work function complexity

Table 1: Comparative capabilities of undersea vehicles. Taken from: [19, p. 20]

DSVs or Deep Submersible Vehicles. These are an example of a conventional submarines which were dominating ocean exploration starting from 1960s. These vehicles are operated by an on board crew, typically accommodating one pilot and one or two observers. The large size of the ships, hence stability, makes them perfect for tasks such as observation, manipulation and sampling and vital tasks where human presence is required on site.

Advantage of DSVs are benefits from human's high-resolution, three-dimensional observation capabilities and full visual depth of the field, augmented with video cameras for close-up inspection, which is still superior to the modern observation remote sensors [19, p. 19-20].

However, human presence on board and the ship sizes increases costs of handling and support of the vehicle. Firstly, a mother ship has to be on-site to deploy and retrieve the submersible as well as provide logistical support during the mission both for the vehicle and the crew. Secondly, there is no real-time data exchange between the submarine and the base station as the observers on-board are collecting information and storing it locally during the mission (such as gathered minerals/materials, video/image footage, etc.). Acquired data becomes available after successful return and further analysis. Thirdly, battery power is an issue taking into account the ship size and human endurance, where life support for the on-board crew has to be maintained throughout the mission as well as emergency life support must exceed mission time significantly in case of any sort of failure, resulting in the reduced mission times [19, p. 20-43].

Nevertheless, as with other technological advances, the innovations in energy storage, propulsion, manipulators and composite materials is contributing to the utility of DSVs up to this day, making possible record breaking deep-dive expeditions consisting of a single crew member. One of such examples is a DSV *Limiting Factor*, Triton [20] 36000/2 model submersible which reached a maximum depth of 10,928 meters, outperforming any previous manned dive by 16 meters [21].

Autonomous Underwater Vehicles Contrary to other vehicle kinds, these type of underwater vehicles are self-guiding and self-powered with no physical connection to the operator. Unlike ROVs who have limited range of operation due to its tether, AUVs can freely glide from the surface to the depths.

Main power consumption of AUVs is to enable the thrusters to move the vehicle around, as well as powering on-board sensors and maintain the navigation and positioning functions. An additional power supply has to be provided for an AUV to power up the device at least for 24 hours in case of emergency, allowing to retrieve the device [22, p. 44-47]

Typical use sensors on an AUV is to map the ocean floor, data collection, pollution prevention, record environmental information, sense what humans have left behind, identify hazards to navigation and geological exploration. Since AUVs have faced technological challenges and applications, their development was slower comparing to ROVs. However, with the rapid evolution of battery and AUV technologies, and contribution to AUV development these devices gain a potential of new tasks and applications.

Main challenges faced by AUVs are connected with operation in underwater environments. These include communication, localization, and navigation due to the impossibility of relying on radio communications and global positioning systems due to signal propagation in such environments [23]. A notable example of an AUV is the submersible glider "Scarlet Knight", the first AUV to cross Atlantic Ocean. The device was launched from New Jersey, USA and over the course of 221 days successfully reached the coast of Baiona, Spain. The data gathered by the device contributed to the understanding of the effects of climate change on the ocean [24].

Remotely Operated Vehicles In comparison with DSVs, are connected to the *Control station* (e.g., vessel or a platform) on the surface by a tether that carries control signals, power and feedback data for the vehicle. As the name implies, these vehicles are remotely operated which significantly reduces their cost removing the need for a human life support functions, making these devices ideal for operating in dangerous and radioactive environments(e.g, mine defusing, work with radioactive components, inspection of nuclear power plants).

Typical advantages of ROVs are stronger power, stable signal transmission, convenient recovery [25, p. 173]. They are inherently suited for working for extended periods, performing local surveys, operating in high-risk areas, and passing large quantities of real-time sensor information back to a surface support vessel. However, due to the tether connected to the support craft, ROVs are subjected to mostly vertical operation and are limited in speed and distance, making them less suitable for large area surveys [19, p. 25].

As these vehicles were initially introduced in 1950s by the military, in 70s they started to gain popularity and were further developed by the industrial sector. Due to technical innovation, these devices became more available and popular for various applications. Significant part of which is scientific use such as ocean exploration and educational purposes.

ROVs range in size from light and medium weight to heavy work-class devices. Large ROVs found a niche in the offshore oil and gas industries and in the communication industry for underwater manipulation, cable burial, and inspection tasks [19, p. 21]. These type of

devices can carry much higher payload and tools, but also require heavy equipment in order to deploy the device in the water. Whereas, light and medium ROVs are mostly used for surveying, exploration and sampling tasks. These devices are typically equipped with thrusters, battery, lights, manipulators and sensors for sample collection. Additionally, they have a video camera or at least a still camera in order to transmit videos or images back to the support ship.

BlueROV2 The BlueROV2, an example of Remotely Operated Vehicle, is an affordable high-performance underwater robot with open-source electronics and software. Which is mainly used for a wide range of applications, such as research, explorations, inspections and adventuring.

The basic complication of the device consists of a ROV equipped with camera and lightning. With an HD (1080p, 30fps) wide-angle low-light camera optimized for use on the ROV. It is attached to a tilt mechanism allowing the pilot to look up and down. And a set of two or four lumen lights, providing 1500 lumens each with dimming control to brighten the underwater depths [26]. More detailed technical specifications are provided in Table 2.

Parameters	Value
Length	457 mm
Width	338 mm
Height	254 mm
Weight in Air (with Ballast)	10-11 kg
Weight in Air (without Ballast)	9-10 kg
Thrusters	T200
Maximum Rated Depth	100 m
Onboard Computer	Raspberry Pi 3B

Table 2: BlueROV2 Technical specifications. Taken from:[27]

The device connects to the *ground control station*, consisting of a PC with configured software to monitor the drone, *gamepad controller*, used to navigate the device. And a *battery* to power up the ROV, with different power supplies recommended for best performance.

ArduSub Project A fully-featured open-source solution for remotely operated vehicles and autonomous underwater vehicles. Being a part of ArduPilot [28] project it provides extensive capabilities for operating ROVs, including depth and heading hold, autonomous navigation and feedback stability control. It works with Ground Control Station and provides interoperability of software components to perform monitoring of the vehicle condition and use of assets such as simulations, log analysis tools and higher level APIs for control and management of the vehicle [29], basically serving as a "brain" of the ROV.

QGroundControl Is a user interface for operating the ROV. It is a peace of software that is installed on the Ground Control Station enabling a user of the ROV to send commands to the vehicle, receive data from the device, as well as monitor the condition of the device [30].

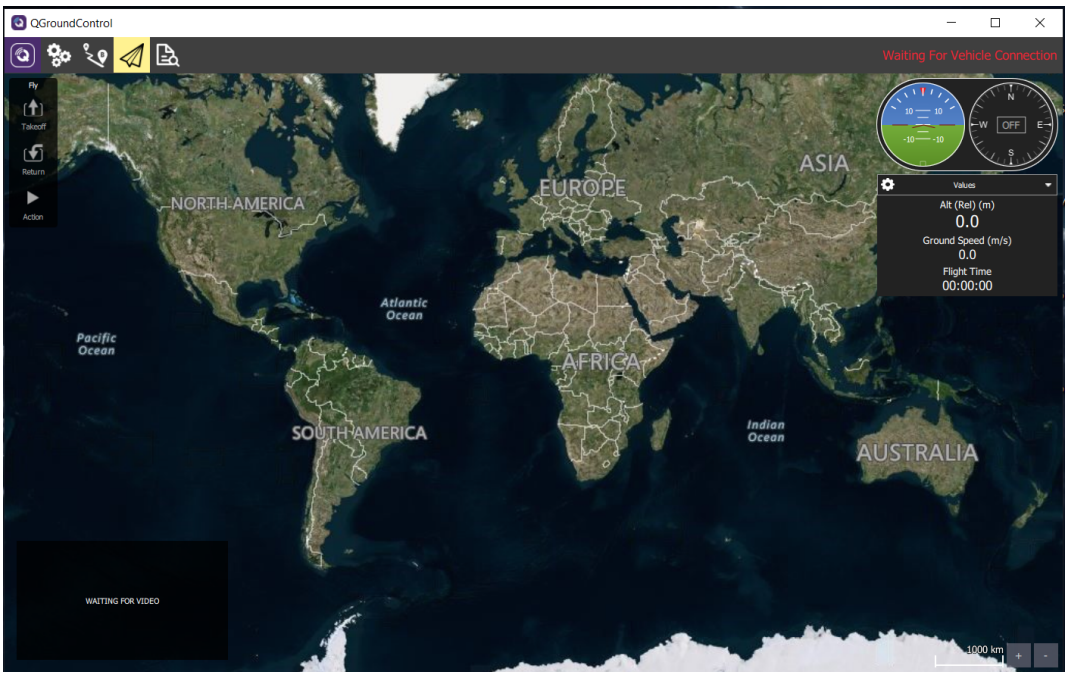


Figure 3: QGroundControl home screen. Own screenshot

Companion is a *Companion Computer* which runs software that relays communication between the autopilot and QGroundControl via an Ethernet cable. Additionally, software installed on the Companion streams HD video to control station [30]. Presented in the Figure 4 below, one can find how the components are connected and operate with each other.

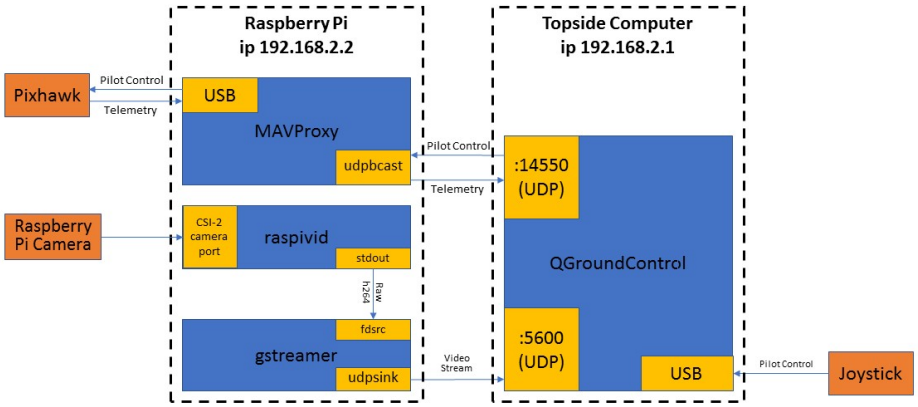


Figure 4: Diagram of the software components and their interactions. Taken from: [30]

3.2 Machine Learning overview

Machine Learning is a sub-field of Artificial Intelligence (AI), with a focus on algorithms and statistical models. It's purpose is to enable systems to perform tasks automatically without explicit instructions relying on patterns and inference. ML approach is to train a general purpose learning machine to predict the outcome by utilising a large data set [31, p. 206-208].

The typical classification falls into four broad categories, *Supervised learning*, *Semi-supervised learning*, *Unsupervised learning* and *Reinforcement learning*. The choice of implementation approach varies by the type of problem to be addressed and the desired goals.

Supervised learning is the process when a model is trained to make predictions with the data set that is well known and well defined, and the outputs are labeled. In this case the model is continuously trained until it achieves the desired level of accuracy for the prediction and can be corrected when those predictions are wrong. The resulting trained model would be used to make predictions for the new Inputs which were not present in the training data set [32, p. 20-22].

Unsupervised learning The data set used for training is unlabeled and less well defined. The goal of the model in this case is to find similarities in the training data, like patterns or structure, and to cluster the Outputs [32, p. 22-24].

Semi-supervised learning In this case the training is performed on the mixed data, utilising both labeled and unlabeled examples. Apart from making predictions the model must learn the structures to organize the data [31, p. 209-210].

Reinforcement learning Unlike supervised and unsupervised learning, this method infers optimal sequential decisions based on previous actions, providing feedback and guiding the learning algorithm. An example of reinforced learning, is an agent navigating in an environment with the presence of obstacles that have to be avoided [31, p. 210-211].

3.3 Neural networks and Deep Learning

Deep learning, also known as feature learning, uses input features and maps them to an output[33, p. 1-8]. The mapping happens in between, where multiple connected hidden layers, consisting of multiple neurons, which are processing the computations, and when combined are designated to learn the relationship between the input features and the output [34]. There is a myriad of problem types that could be solved with Deep Neural Networks (DNN), such as classification, clustering and vision. However, DNN should be designed specifically to solve a particular problem type. In its simplest form the Neural Network consigning of input Layer, single hidden layer and an output layer, can be seen in Figure 5.

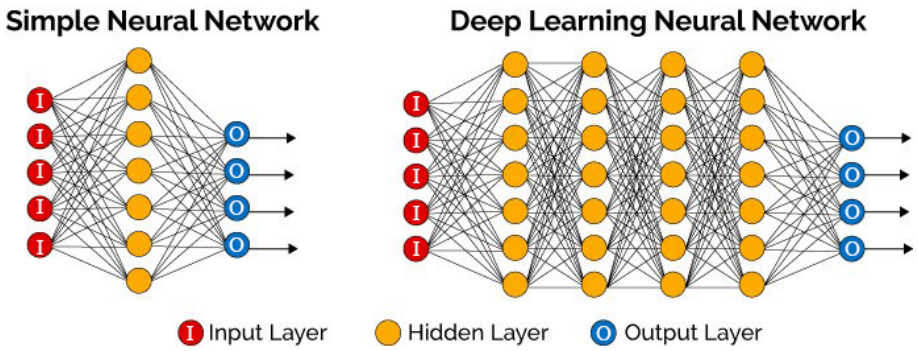


Figure 5: A simple Neural Network vs Deep Neural Network. Taken from: [35]

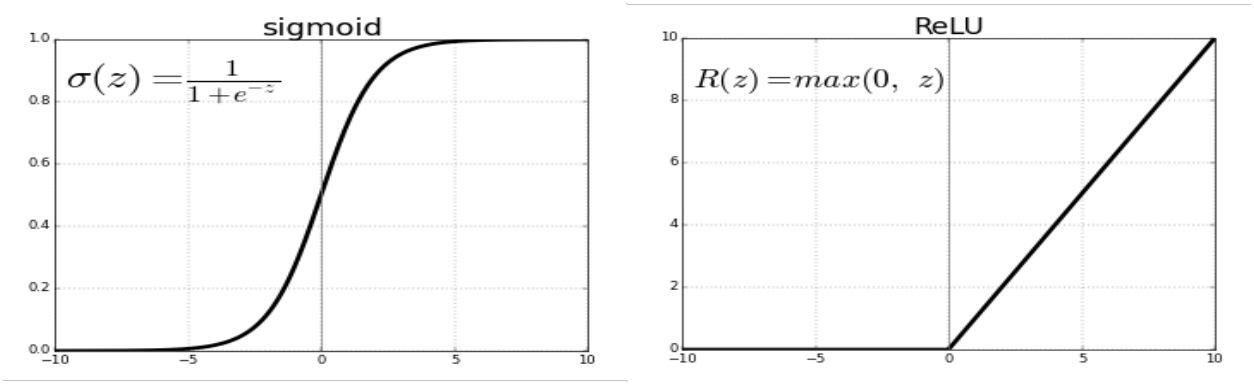
Typically, there is a number of hidden layers, each packed with multiple neurons. Each neuron has associate weights, which are summed together and passed to the activation function. There are many types of activation functions such as Sigmoid, Tanh, ReLU and Softmax, which are presented in Figure 6.

Sigmoid is an activation function used for making binary predictions. Its takes the input of a real number and returns the output in ranged between 0 and 1 [36, p. 5-6].

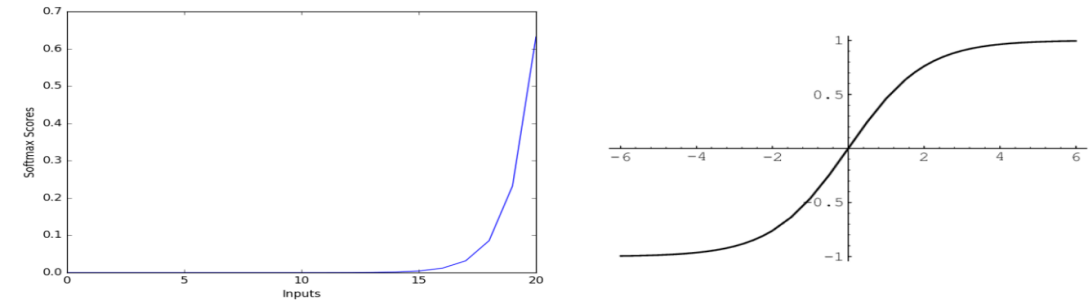
Tanh function is similar to Sigmoid, but unlike the latter it bounds the output to the range -1 to 1. It is mostly utilised when the desired output can be negative, hence the range start -1 [36, p. 7].

Rectified Linear Unit or ***ReLU*** is another type of activation function which gained popularity due to its performance. It performs a threshold operation to each input, where values below 0 are set to 0 and activates the neuron if the output for the input is 1 [36, p. 8-10].

Softmax used in the output layer of a neural network, it calculates probabilities of each class from a group of different classes, by producing an output values ranging between 0 and 1, and the sum of probabilities equals to 1 [36, p. 8].



(a) On the LEFT - Sigmoid function; On the RIGHT - ReLU function.



(b) On the LEFT Softmax function; On the RIGHT Tanh function.

Figure 6: Example of the activation function graphs. Taken from: [37]

Activation functions are considered a part of hyper-parameters within a neural network. These are parameters that define the structure of the network and the training of the algorithm [36]. Other related parameters include number of hidden layers, which can be increased until the error is adjusted(increasing accuracy). Contrary a low number of layers can lead to underfitting. Dropout, a technique to avoid underfitting, sets the probability of the value being passed to the next layer [38]. Batch size is the number of samples used to train the network for one iteration. A number of iterations is called epochs, when data is shown to the network while training. The learning rate which defines the permutations in the weights of the model [39].

3.4 CNN

As traditional NN approaches fall short in processing diversity of images, Convolutional Neural Network (CNN) prevails in image and video processing applications. CNNs are widely used for the image recognition, object detection, and tracking tasks of computer vision.

The approach it takes to analyze densely populated images is to model smaller pieces of

information and combine them using deep networks across multiple layers. There are three types of layers in CNNs, which are:

- Convolutional Layer
- Pooling Layer
- Fully connected layer

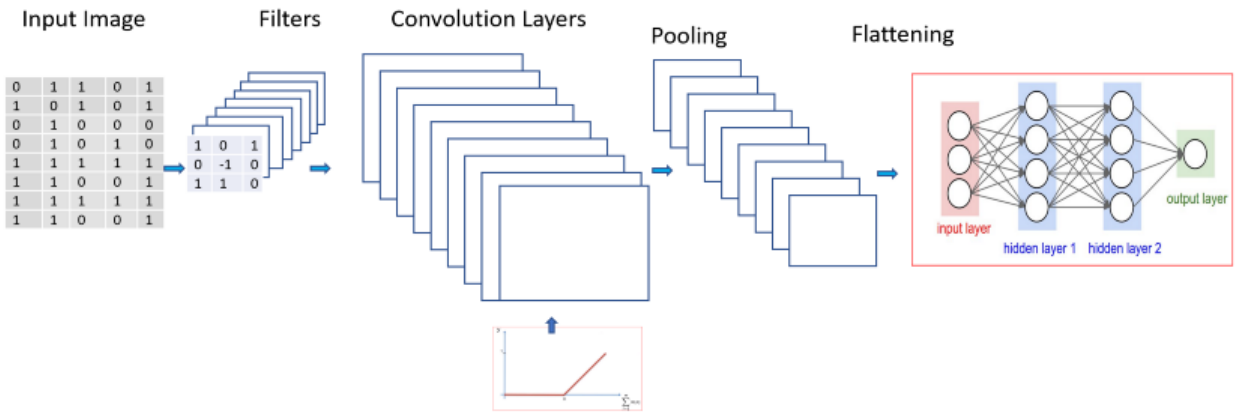


Figure 7: Example of Full CNN. Taken from: [40]

Convolutional and pooling layers aid in feature extraction and final classification is done by the fully connected layer. The layers in CNN's are set to three channels, hence the set dimensions height, width and depth. Neurons of one hidden layer are not fully connected, but are only connected to a part of neurons of other layer.

Convolutional Layer produces feature map by applying a fixed size filter to the entire image, adding stride and zero padding. The resulting matrix is passed to the activation function [41, p. 64]. **Pooling Layer** is responsible for dimensionality reduction, decreasing the parameter count, optimizing the feature map size to ensure key information about the image is kept [41, p. 65]. In **Fully Connected Layer** neurons have full connections to all the activations in the previous layer as well as full connections to the output layer, where desired number of outpus is generated [38, p. 63-64].

3.5 RNN

Recurrent Neural Networks is mainly used for sequential data and natural language processing, in applications such as machine translation, image captioning and sentiment analysis [42, p. 273-278]. Its special feature is in internal memory, which can remember key details about the input, thus increasing accuracy of the next prediction. Unlike feed-forward neural networks, such as CNN, information in RNN is time stamped(remembers position of the

sequence) and fed cyclically in a loop, using current input and the learnings from the inputs received previously, when making a decision, as shown in Figure 8. Meaning input weights are attached to both present and past inputs of the network, allowing to provide an output for each timestamp[41, p. 66-68].

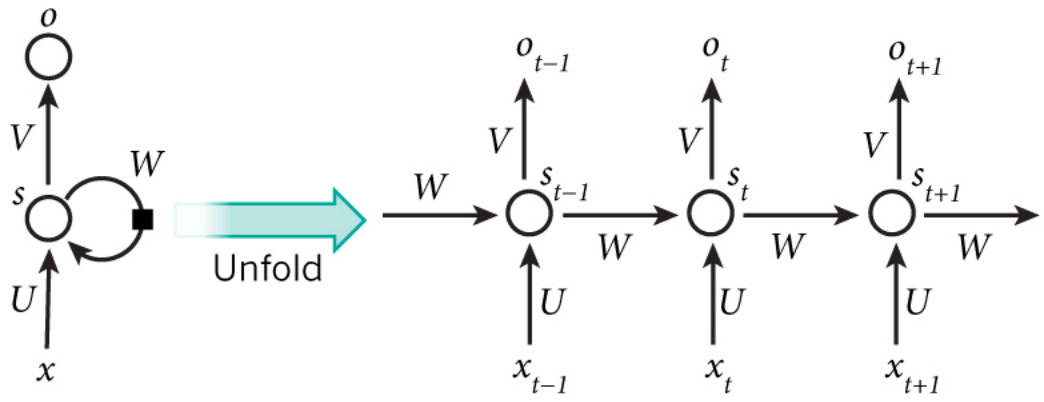


Figure 8: Example of a simple RNN with time layered representation. Taken from:[43]

However, there are practical issues in training a recurrent neural network, such as the vanishing and exploding gradient problems [42, p. 274]. To overcome these shortcomings Long Short-Term Memory (LSTM) is employed stretching the classic RNN's short-term memory and Gated Recurrent Units (GRU) which decide on importance of data and whether it should be retained [42, p. 292-297].

3.6 Computer vision

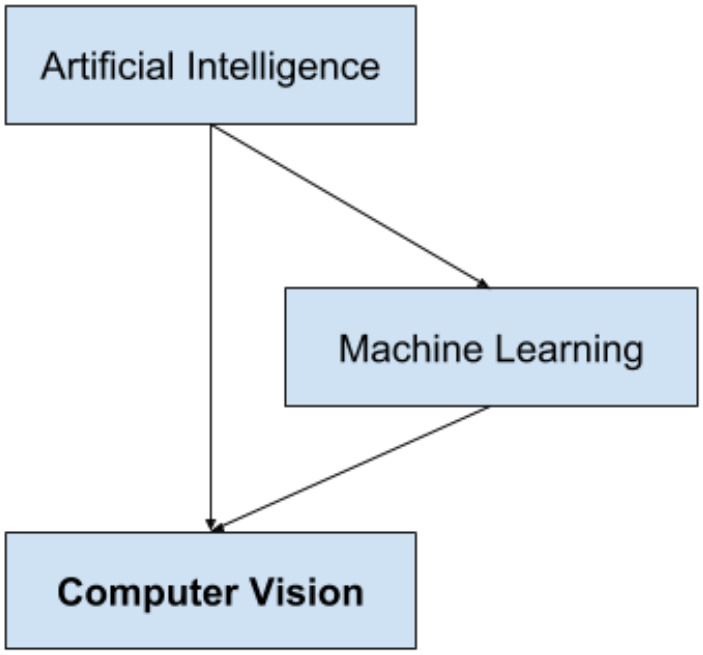


Figure 9: Relationship of topics. Taken from:[44]

Even though CV is field of study that has been around for years, it is still a developing area as there are various challenges related to making computers understand the context of what is shown to them. Complexity of visual perception persists in variety of orientations that an object may have, light conditions, occlusion from other objects etc. and the abundance of meaningful information that might be extracted for a certain purpose [45, p.15-17].

Being a broad area of study Computer Vision has many applications, often specializing in a particular task or domain. Typical tasks of CV include such examples[46, p. 3-10] as:

- Identify features present in the image, such as edge detection.
- Classify objects present in the image, assigning label to an image(person, furniture item, etc.)
- Classify images with localization, recognize an object together with the location of the object in the bounding box.
- Object detection, identifying all the objects present in an image together with their locations
- Semantic segmentation, detecting different objects present in the picture at the pixel level, resulting with regions of different classes of objects.

3.6.1 Object detection

Vision based object detection has had major attention in recent years due to its' prospects of transforming the industries (such as an introduction of autonomous cars), thus putting a lot of pressure and resources in research and development for such systems to be robust[47]. Object Detection algorithms function as a combination of image classification and object localization, allowing for semantic understanding of real world images and videos. Its' goal is to identify all objects present in the image with labels and bounding boxes. Object detection is divided into two types: classification problem based, such as RCNN, Fast RCNN, Faster RCNN, and regression problem based, like Yolo and SSD.

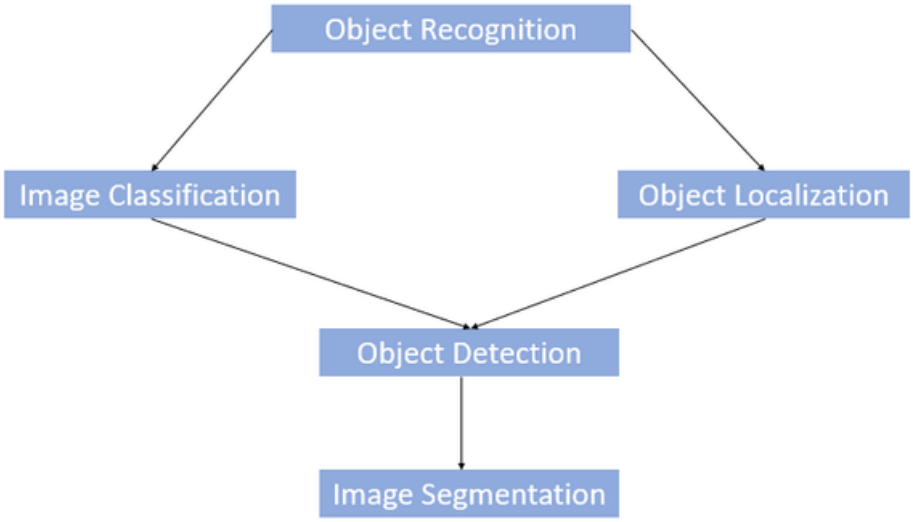


Figure 10: Overview of Object Detection field. Taken from:[48]

3.6.2 Region-based Convolutional Neural Network

CNN had a classification problem due to using the Exhaustive Search technique, where a sliding window of different scales is applied on the image to make region proposals. As it is slow and computationally expensive to run multiple generated patches through entire network. The approach taken by RCNN employs the object proposal algorithm Selective Search, which reduces the number of produced Regions of Interest to around 2000, which are then fed into CNN producing a feature vector for each proposed region. The final classification of an object is done by *Support Vector Machine* (SVM) and localization with bounding box *regression*. For final output, to all scored regions Non-Maximum suspension is applied to eliminate duplicates and propose the final detection candidate[49].

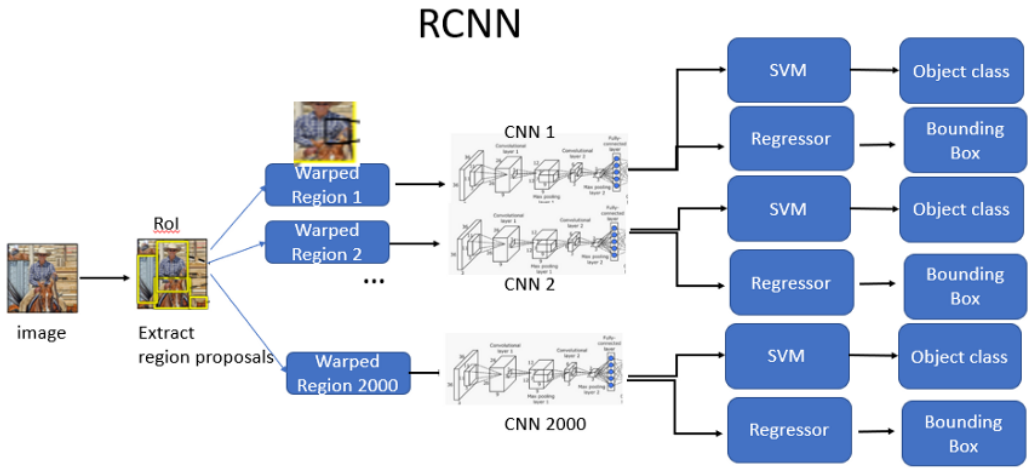


Figure 11: Architecture of RCNN. Taken from:[40]

Non-Maximum suspension checks the overlapping regions computing IoU, rejecting the ones lower then the set threshold and ensures an object is detected only once. *Intersection over Union* (IoU) compares two bounding boxes, ground truth and the one predicted by algorithm resulting with a IoU value, that shows overlap of the two boxes.

Although, the original RCNN algorithm is computationally slow and expensive, there are newer architectures that improve the performance, such as Fast RCNN[50] and Faster RCNN[51]. Each next version is addressing the shortcomings of the previous approach.

3.6.3 You Only Look Once

YOLO family is a series of end-to-end deep learning models designed for fast object detection[52]. YOLO is general purpose detector that detects various objects simultaneously. It is considered one of the most popular algorithms, due to fast performance architecture. Base YOLO network processes images in real time at 45 fps, and a fast version with smaller architecture performs at 155 fps on a GPU[53]

The object detection in YOLO is re-framed as a regression problem, thus changing the approach compared to more recent methods[53]. There is no regional proposal step like in R-CNN. Instead, the entire image is split into a grid of set size. Taking into account the contextual background information, for each grid cell the network predicts fixed amount of bounding boxes along with their confidence and class probabilities from full image in one evaluation. As YOLO sees the entire image during training and testing, it implicitly encodes contextual information about classes as well as their appearance and the presented background. Finally, non-max suppression algorithm removes the duplicate overlapping bounding boxes and generates final prediction for each class[53]. There are different versions of YOLO family of networks, such as YOLO v2[54] and YOLO v3[55], YOLO v4[56], which differ mainly in performance and accuracy increases.

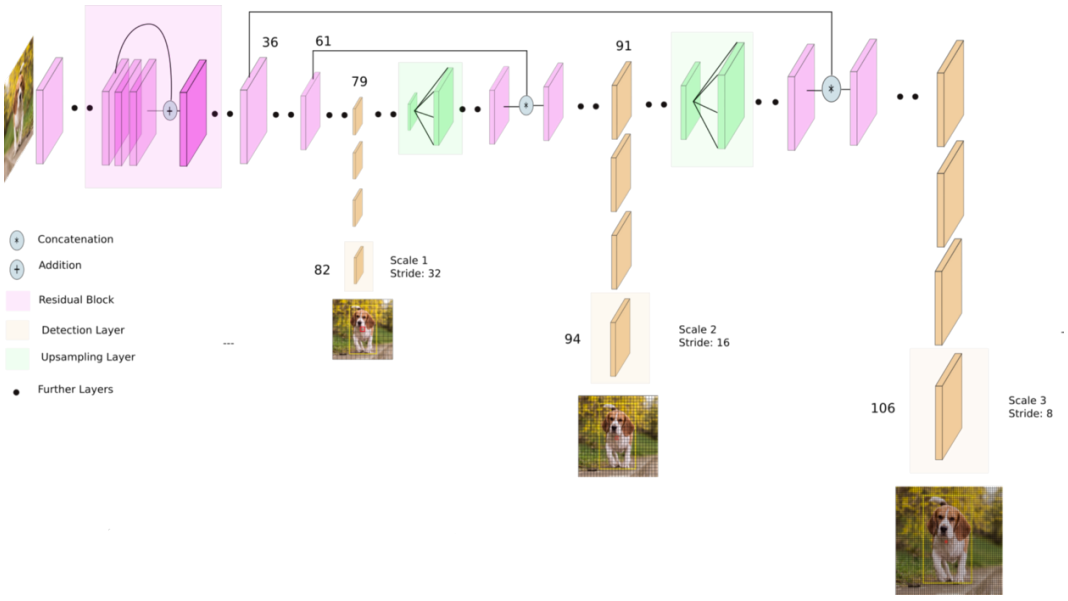


Figure 12: YoloV3 network architecture. Taken from:[57]

3.6.4 Single Shot Multibox Detection

The SSD approach is based on feed-forward convolutional network, which produces a set of bounding boxes with fixed size as well as confidence scores for object classes present in the boxes[58].

In SSD Regional Proposal Network is eliminated. The dropped accuracy is compensated with *multi-scale features* and *default bounding boxes*. SSD object detection is composed of two parts, where one extracts feature map and other applies convolutional filters to detect objects[59]. For the feature extractor a default convolutional network is used called VGG-16. Then convolutional layers are added to create the grids used for the classifications/predictions. More specifically, allowing to extract features on different scales and progressively decrease the size of the input to each subsequent layer. By combining predictions from multiple feature maps with various resolutions helps to identify small and large objects. Similar to YOLO, the predicted objects have multiple bounding boxes per object, which are compared in the final step with ground truth bounding boxes. The final step is to apply non-maximum suppression to produce the final detection[58].

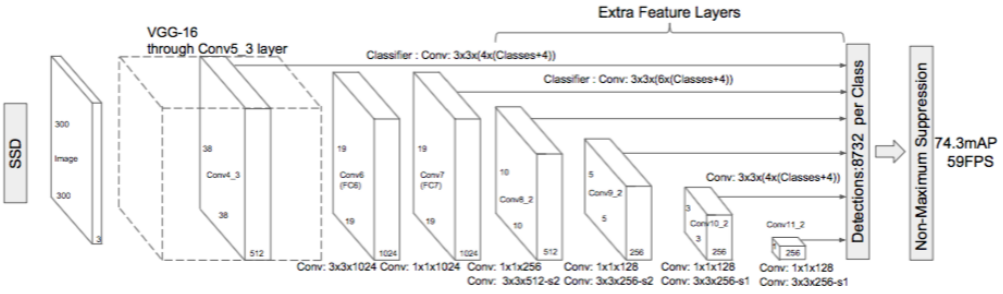


Figure 13: Architecture of SSD. Taken from:[60]

3.7 Machine learning Tools

For the purpose of developing the prototype a Machine Learning model has to be selected. There are many open-source libraries available with large communities and great deal of documentation. A summary of most relevant existing tools and frameworks is presented bellow, identifying key features appropriate for the development stage.

PyTorch is an open source machine learning end-to-end framework developed and maintained by Facebook. Popular among academic research. The ecosystem of tools and libraries around the framework enables supporting development in areas from computer vision to reinforcement learning. PyTorch includes distributed training, supporting asynchronous execution; Dynamic development, allowing manipulation with computational graphs on the go; Integration with various cloud providers. With the release of PyTorch 1.5, new and updated APIs enable C++ frontend API parity with Python[61].

- Is the preferred tool for academic research.
- Provides a low level API.
- Provides access to pretrained models, allowing to modify and expand.
- Allows to create custom expressions.

TensorFlow is an open-source, end-to-end platform, used to build and deploy machine learning models. It provides a library of functions that allow to build architectures in much less time[62]. TensorFlow models can be created and deployed on various platforms, such as Node.js, browser, mobile and EDGE devices(Android, iOS, RPI). At the time of writing this report, the stable available version of TensorFlow was 2.2.

Two abstraction levels are provided, the Sequential API and Subclass API. Where former provides a more user-friendly approach for inexperienced users allowing to build a model by putting together building blocks. Whereas, latter provides a more advanced approach for experienced users to develop the models, allowing to create custom layers, activation functions, and training loops[63].

- Enables creation of machine learning systems and neural network models.
- Enables fast and distributed training.
- Provides different level APIs.
- Provides a variety of pretrained models.
- Provides extensive system customization.

Keras is a deep learning API written in Python. It was developed with a focus on user experience, fast prototyping and modularity. Keras is built on top of TensorFlow 2.0 taking full advantage of its platform deployment capabilities. Its models can be exported to JavaScript to run in the browser as well as to TF Lite becoming available on Android, iOS, and embedded devices[64]. Keras offers consistent and simple APIs, while minimizing the number of actions user needs to perform, it also provides clear and actionable feedback on user error encounters. Keras offers high customization in the form of modules, allowing to tailor building blocks(such as layers, optimizers, activation functions) of the model according to own needs[64].

- Provides high level API with a user friendly interface
- Enables fast prototyping
- Optimized to work with smaller data sets.
- Usable on a variety of back-end systems
- Supports customization of various layers(dropout, pooling)

3.7.1 On the Cloud

There is a number of online Machine Learning services, also known as MLaaS, where big companies provide data, computational power, etc. These services speed up the process of training models or suggest pre-trained models;

Google AutoML Vision There are two solutions provided by particular vendor, AutoML Vision[65] and AutoML Video Intelligence. The main difference is in data used for the model training. The first one allows training based on images, whereas the second enables utilisation of videos. The platform supports Single-label classification, Multi-label classification and Object detection tasks. Similarly for video input Classification and Object tracking is supported. It is possible to utilize labeled videos, providing an associate CSV, or unlabeled videos while utilizing the proposed labeling tool.

The whole process consists of 5 steps, import, label(filter), train, evaluate, and use/deploy.

For the import, various labeled pictures have to be uploaded. The supported formats are available such as JPG, PNG, GIF, BMP, ICO, and ZIP. In case cloud storage is used,

it is required to have a CSV file with image paths. Optionally, users can specify Train, Validation and Test split.

After data has been uploaded, it can be modified by adding new labels or manipulating the images/videos(relabel, remove, edit). For the training process the desired hardware has to be selected, defining model optimization(High accuracy vs Fats prediction) and node budgets(depends on image count).

The platform provides a comprehensive overview of the model, showing precision and recall scores, as well as adjustable confidence threshold level and confusion matrixes for the model evaluation.

Additionally, the platform provides several deployment options, where it is possible to export the model as a TF Lite(for EDGE & mobile), TensorFlow.js(browser & Node.js), Core ML(iOS devices), Coral(TPU-based EDGE) and a Docker container(convenient packaged interconnection).

In case of Video Intelligence three types of the results are provided. Labels for the new input video, time period when object is present in the video and a confidence score of the prediction.

Azure Iot EDGE Developed by Microsoft, Azure IoT EDGE is a fully managed service built on Azure IoT Hub. It allows to deploy workloads such as Artificial Intelligence, Azure or other services, custom code to run on EDGE devices via standard containers[66].

The offered architecture consists of three components: IoT EDGE modules which are containers running Azure or 3rd party services or custom code deployed on the device locally. IoT EDGE runtime manages the modules deployed to each device and cloud-based interface, which helps to manage and remotely monitor IoT EDGE-enabled devices.

Additional features:

- Container managements system
- Zero-touch provisioning with edge devices
- Supports various SDKs (in C, C#, Node, Python, and Java)
- Tools for module development(coding, testing, debugging)

AWS SageMaker is another Machine learning service provided by Amazon[67]. They offered various tools to work with algorithms and models for Computer vision, Natural language processing, speech recognition, as well as image and videos. They provide services to create, train and deploy machine learning models in the cloud as well as on embedded systems and edge-devices.

3.8 Existing Solutions and Research papers

In the following chapter a research on existing solutions has been summarized. The most relevant articles have been reviewed and presented for the readers' convenience. As the

focus of the report is on image recognition in a video stream, the research has been narrowed down to services and principles satisfying aforementioned task. The papers are classified into categories, namely object recognition approaches with single data-set, OR approach with multiple data-sets and comparison of various state-of-the-art algorithms for object detection. For readers convenience a summary of research papers is presented in Table 3 below.

Research papers	Obj. Recog	Img. Class	Obj. Detec	Performance Evaluation	Dataset	Data manipulation	Neural Network
Applying Machine Learning Using Custom Trained CNNs on Subsea Object Detection and Classification		x	x	YoloV2	COCO +custom		CNN
Robotic Detection of Marine Litter Using Deep Visual Detection Models			x	YoloV2 Tiny-Yolo Faster R-CNN SSD	J-EDI +custom	Deep learning	CNN
Underwater Object Recognition Based on Deep Encoding-Decoding Network	x			AlexNet	Fish4Knowledge	Data augmentation: horizontal mirroring, crop, downsampling Transfer learning	CNN
Man-Made Object Recognition From Underwater Optical Images Using Deep Learning And Transfer Learning	x			AlexNet	Amazon	Transfer learning Simulated images: manipulations with color	CNN
Transferring Deep Knowledge for object recognition in Low-quality underwater videos	x	x	x	Transfer learning Model evaluation	Fish4Knowledge	Transfer learning Horizontal mirroring Crop Subsampling Affine transformation	DNN
Investigation of Vision-based Underwater Object Detection with Multiple Datasets	x	x	x	Image processing	Garda Portofino Soller	Pose estimation	

Reveal of Domain Effect: How Visual Restoration Contributes to Object Detection in Aquatic Scenes	x		x	Image quality vs Detection performance;	URPC2018	Cross-domain data augmentation	CNN
Robust Underwater Object Detection with Autonomous Underwater Vehicle: A Comprehensive Study			x	Summary Models & Algorithms	Various Datasets examined		DNN/ CNN
Research on underwater object recognition based on YOLOv3	x	x		YoloV3 Faster R-CNN	URPC	Adjusted training configurations	CNN
PENGUIN: Aquatic Plastic Pollution Sensing using AUVs	x	x	x	Plastic type detection			

Table 3: Summary of research papers on existing solutions.

1.Applying Machine Learning Using Custom Trained Convolutional Neural Networks on Sub-sea Object Detection and Classification The aim of the presented white-paper was to create three different data sets which were used for training the chosen algorithm, implement a convolutional neural network and compare the resulting data sets on the same benchmark. The created data sets were divided into *above water*, *below water* and *hybrid*, where the last is a combination of the other two [68, p. 29].

The discussion held in the paper describes difficulties of detection and classification of sub-sea objects, as it ought to be more error prone and complex in underwater environment comparing to above water. This is true due to few challenges such as availability of the operational domain training data and rough environmental conditions where device is operating. The paper provides an overview of the evolution of image recognition and object classification and some simple methods formerly used. Main problem these methods faced in previous years was due to low computational power and availability of data sets for analysis.

A breakthrough came in 2012 when ImageNet held a competition where a number of approaches were presented to correctly classify objects in images [68, p. 5]. The most accurate approach was presented by [69] as referenced in[68, p. 3] with the use of convolutional neural network to classify objects in images with a single label.

Further development in the field led to detecting several objects at once, also taking into account their position. Regional - Convolutional Neural Network (R-CNN) was a step forward, outperforming other algorithms by 30%. Drawback however was in its processing speed which took around 13 secs on a powerful GPU. Its successors, Fast R-CNN[50] (2015), Faster R-CNN[51] (2017), and Mask R-CNN[70] (2020) reached performance of 7 frames per second (FPS) with mAP of 73.2% on a powerful GPU[68, p. 6].

Alternative approach arrived with YOLO algorithm (2015) followed by YOLOv2 which yielded 78.6% mAP at 40 FPS, and YOLOv3 which had little improvement in accuracy. Latter two algorithms are considered state-of-the-art in real-time object detection because of their increased accuracy and higher speed.

Finally, authors present the methods chosen for implementation of their detectors(YOLOv2) and go into details comparing and evaluating the performance of the trained neural network. It is worth mentioning that images were manually labeled and used for training selected algorithms, with resulted Average Precision scores of 36%, 77%, and 92% was achieved for each data set, *above water*, *below water* and *hybrid* respectively [68, p. 57]

Outcome The report provided a nice base knowledge of deep learning approaches and the development of image recognition techniques over the years. As authors have chosen a specific algorithm to train a neural network, the discussion on the working principles of the algorithm was beneficial, it was also worth to get an insight on acquisition of the used data set (including the work behind it, data preparation, labeling and annotating the data) and further evaluation of the results achieved by implemented detectors with these data.

2.Robotic Detection of Marine Litter Using Deep Visual Detection Models The authors of the paper are dealing with the problem of pollution with plastic debris in underwater environments. They believe that AUVs equipped with the detection and

removal mechanisms could contribute to the solution of the problem. Therefore, they are investigating the plausibility of real-time detection of underwater trash using deep-learning visual detection, while examining the performance of currently available methods.

The challenges identified for the task consist of *Light conditions in shallow waters*; *Changing turbidity of the water*; *Variety of debris - and Condition*; *Dissolution of the trash/materials*, all of this implies the complexity of the problem. To restrain the problem authors focus on single type of material chosen for the purpose - plastic.

The goal of the paper is to evaluate the accuracy and performance of a number of deep-learning algorithms (YOLOv2, Tiny-YOLO, Faster RCNN and SSD). Additionally, evaluate different training techniques on a particular network (YOLOv2) and to produce a unique dataset for training the deep visual detection models for the task [71, p. 2].

First step to fulfill the purpose of detection of underwater trash with deep-learned appearance model a large annotated type-specific dataset is required. Authors have identified two possible data sources: where one is video annotation database provided by The Monterey Bay Aquarium Research Institute (MBARI) who have conducted a 22-year survey on trash accumulated across the sea bed off seaboard of the United States of America [72] and second, made by Japan Agency for Marine Earth Science and Technology(JAMSTEC), who has made a dataset of deep sea debris available online as part of the larger J-EDI (JAMSTEC E-Library of Deep-sea Images) dataset. It contains type-specific debris data in a form of short video clips, dating back to 1982[73]. Second data set was chosen and processed resulting with a final training dataset to be composed of 5,720 images with resolution of 480x320 pixels. For evaluation of the model 820 images were collected and annotated, its worth to mention that the testing set was carefully selected in order to be challenging for detectors, so as to provide realistic evaluation of performance of the detectors in real life cases[71, p. 5]

In terms of evaluation, three different hardware types where used to indicate how the models could be expected to perform, which are: offline manner - GPU(Nvidia 1080), real-time on robotic platform - embedded GPU(Nvidia Jetson TX2) and lower power robotic platform - CPU(Intel i3-6100U).

The results provided in the white paper point out that Faster RCNN stands out in terms of accuracy, but falters in inference time. YOLOV2 shows a good balance of accuracy and speed, while SSD provides the best inference times on CPU. In case performance is the primary consideration, however, Tiny-YOLO outpaces all other algorithms significantly on the Jetson TX2.

The paper is concluded with considerations that it is plausible to use visual deep learning models in real-time for the purpose of collecting trash debris.

3.Underwater Object Recognition Based on Deep Encoding-Decoding Network Authors are proposing a new deep learning architecture with convolutional encoding and decoding features for object recognition task. The proposed framework takes a pre-trained convolutional model, AlexNet trained on ImageNet task, transfers the knowledge of the two first layers to the model for underwater detection task. The proposed encoding-decoding is used to finetune the features. The resulting convolution-deconvolution network [74, p. 4] is employed for image classification.

As for dataset, the Fish4Knowledge project was used[75] which contains 23 classes of fishes manually labeled by marine biologists. As an additional contribution from the authors served data augmentation, to increase amount of training data. Following the reference of [76] they have applied a horizontal mirror, crop, down-sampling, and affine transformation. This was done to simulate the movement of the target object and its appearance in the camera, taking into account its distance, movement directions and appearance angle.

The resulting experiments showed positive outcomes, where high accuracy of the model was achieved. This implies that the approach and modifications proposed by the authors have had positive impact on the overall model performance.

4.Man-Made Object Recognition from Underwater Optical Images Using Deep Learning And Transfer Learning Another similar example can be seen in the work made by [77] . They discuss the challenges of optical object identification in an underwater environment, where poor image quality is the main influencing factor affecting optical image analysis. Additionally, there is a lack of large-scaled labeled data for training deep learning models with underwater images, which is one of the prerequisites for estimation of parameters when training deep learning models. Therefore they present a development approach of a novel recognition system for man-made objects from underwater optical images with the help of deep learning and transfer learning. They hypothesise that it is possible to recognize underwater man-made objects using pretrained model with labeled in-air images of man-made objects. Therefore, during the training phase the in-air large-scale dataset of labeled images is combined with unlabeled underwater objects. The resulting framework can successfully classify the underwater man-made objects with robustness[77, p. 1853] They combine two approaches of color transfer method[78] and turbidity simulation[79] to imitate the underwater appearance on in-air images. Following the theory of transfer learning[77], the input of labeled in-air images is used as source domain and unlabeled underwater images are assigned as target domain. The proposed framework consists of 5 convolutional layers and 3 fully connected layers. Classifiers are implemented by fully connected layers at the end of the network. The feature vector generated by the last fully connected layer is processed by the softmax function, while 28 the vector of probabilities represents the final prediction results of the categories[77, p. 1853]. The dataset used for the experiments was taken from amazon.com[67] containing 2817 images of man-made objects, with 31 categories consisting of 36 to 100 images per category, where presented objects have irregular forms and shapes and are captured from different angles. Three experiments were conducted in comparison to the AlexNet (CNN) where 1st showed an accuracy of 55.7% when training data from the same domain were used. However, when training data and test data were generated by different imaging systems the performance of AlexNet dropped significantly, 17.33% and 38.50% for the 2nd and 3rd experiment respectively. The accuracy increase from experiment 2 to experiment 3 is justified by the ability of the proposed framework to transfer the knowledge learned from the source domain to the target domain, that is, from the in-air images to underwater images[77, p. 1855]. The results indicate that introducing transfer learning to a CNN model eases up the need to explicitly collect and annotate underwater images for training the model.

5.Transferring Deep Knowledge for Object Recognition in Low-quality Underwater Videos The authors of the paper are looking into the field of underwater ecosystems exploration. They are trying to solve the problem of moving object detection underwater with the help of Deep CNN, transfer learning and data augmentation to achieve a real-time fish detection and tracking framework from video monitoring systems of AUVs. The challenges in this context are the following: Traditional interest point detectors fall short and produce weak descriptors due to low illumination in the environments and thus a low contrast background; the object of tracking may appear to be in different shapes from various camera angles due to its free moving environment; underwater videos are typically of low resolution and low saturation resulting in a limited discriminative information of objects to be recognized from the videos; there is an insufficient amount of context data available, needed for training of data-hungry CNNs for a specific task of underwater recognition. The proposed framework attempts to overcome the aforementioned challenges by utilizing data augmentation, prior knowledge and transfer learning for a deep CNN model. The idea is to utilize a well trained deep CNN and transfer it to new tasks (UnderWater-CNN) in different domain (Fish4Knowledge). By fine tuning the parameters for target domain and performing data manipulation on the training set it is possible to design a real time recognition system for underwater videos. Data augmentation in this case means the performed manipulations, such as horizontal mirroring, crop, sub-sampling 29 and affine transformation, with the target domain data set, to increase the amount of limited training data and as a result to improve the models performance. On top of that, due to different operational domains, the CNN model learned from ImageNet cannot be used as a feature extractor for underwater videos/environment. So the authors propose to recognize the knowledge from the source domain as prior values for the parameters of the target CNN model[76, p. 8]. The problem of moving objects in the video is dealt with the help of background modeling approaches, most typical example is to take a sequence of frames comparing them and identify where significant change happens. The ViBe[76, p. 10] - background subtraction method used as the object detection algorithm. A problem with ViBe is it may produce false positive patches in the video feed but this issue is handled by the classification model of UW-CNN, which was enhanced with additional categories to distinguish the objects, such as stones and corals.

6.Investigation of Vision-based Underwater Object Detection with Multiple Datasets The authors of this paper similarly investigate the prospects of vision-based object detection algorithms[10] in underwater environments. They use multiple data sets of images made underwater by different research groups on various depths. They have presented a new algorithmic sequence, based on salient color uniformity and sharp shapes of the objects. Highlighted challenges for computer vision are depletion of light under water and challenging operating conditions. Object detection approaches based on feature extraction, such as feature constellation methods were proved to be unreliable in underwater conditions.

7.Reveal of Domain Effect: How Visual Restoration Contributes to Object Detection in Aquatic Scenes The papers investigates an issues within visual restoration and object detection as the essentials of underwater robot perception. Visual restoration is

suggested to address the issues and improve domain specific video quality for subsequent image processing of aquatic scenes. The contribution from the paper is summarized accordingly: at first quality-diverse data domain is constructed with filter-based restoration(FRS) and GAN-based restoration(GAN-RS which is a learning based scheme), both for training and testing; following with the analysis of single stage detectors such as SSD, RetinaNet, RefineDet and DRN, and their performance on different data domains, both within-domain and cross-domain analysis; following with the real-world experiments of performance of online object detection[80, p. 2]. Underwater Robotic Picking Contest 2018 (URPC2018) was used as the data source for the project. Containing 2,901 aquatic images for training and 800 samples for testing, with four categories(labeling), such as "trepang", "echinus", "shell", and "starfish". From this set 3 data domains were composed: 1) original data set with test set and train set; 2) train-F and test-F sets, where all samples were processed by FRS; 3) All samples restored by GAN-RS, producing train-G and test-G data sets. Additionally, a mix of all training sets was composed. Furthermore, degenerated visual samples were restored in datasets F and G. The results of the experiments showed that accuracy decreases with the rise of restoration intensity as well as false positive results increase with the improvement of domain quality, thus recall efficiency is steadily reduced with increasing restoration intensity[80, p. 6]. Another highlight of the results is the notion that low-quality samples mixed with high-quality cannot be learned properly[80, p. 7]. However, visual restoration seems to be of an importance to tackle the problem of domain shift.

8.Robust Underwater Object Detection with Autonomous Underwater Vehicle: A Comprehensive Study The paper dives into investigation of problems affiliated with underwater challenges posed onto object detection task carried out by AUVs. Therefore they introduce a study on various underwater object detection techniques, comparing conventional methods and existing deep learning approaches, while defining the requirements for the solution to be real time and robust while maintaining reasonable accuracy[81].

The paper follows with comparison of trending underwater detection methods described in detail, with an in depth discussion of each approach presenting advantages and disadvantages. Moreover, authors are investigating previously utilised frameworks along with conducted experiment results and utilised data sets.

The paper is beneficial for the project as it lays out a setting for computer vision, that gives a good starting point for further research and provides a comprehensive overview of methods for object detection as well as a brief look on the possible problems associated with the data sets and solutions on how to address them.

9.Research On Underwater Object Recognition Based On YOLOv3 The paper describes a comparison of two state-of-the-art algorithms, Faster R-CNN and YOLOv3 to experiment and evaluate their performance on underwater image dataset. The focus of the paper is on locating and identifying sea cucumbers, sea urchins and scallop. The results show the performance increase of YOLOv3 in comparison to Faster R-CNN, which is indicated by higher mean Average Precision (mAP) being 6.4% and higher recall rate (Recall) of 13.9% and increased detection speed(FPS) of 12 FPS compared to Faster R-CNN[82].

The paper goes into in-depth discussion on Faster R-CNN and YOLOv3 working principles, underlying architecture and experimental results, providing an overview and comparison of the two methods. The key takeaway is that YOLOv3 algorithm has outperformed opposing Faster R-CNN in terms of mAP and Recall and the detection speed can meet the real-time performance requirements. Whereas algorithms accuracy should be improved by utilising image enhancement techniques.

4 Analysis

The following chapter is evaluating the information gathered in Chapter 3. First, three possible system architectures are considered with evaluation of which approach is more suitable for real-time object detection. Second the available object detection methods and algorithms presented in Chapter 3.6.2 and Chapter 3.6.3 are compared, deciding which is more suitable for the system. Third, the availability of data sets and their usage are evaluated. Finally, data preparation is described, required for the preparation step of the model.

4.1 System architecture

The following chapter presents the consideration between different architectures, that are possible for the system. There are four major steps that need to be considered to prepare a machine learning model: processing input data, training a deep learning model, storing the model and deploying the model, where the most computationally expensive is the training part. However, diverse approaches can be taken, there are some benefits and drawbacks in each of the methods.

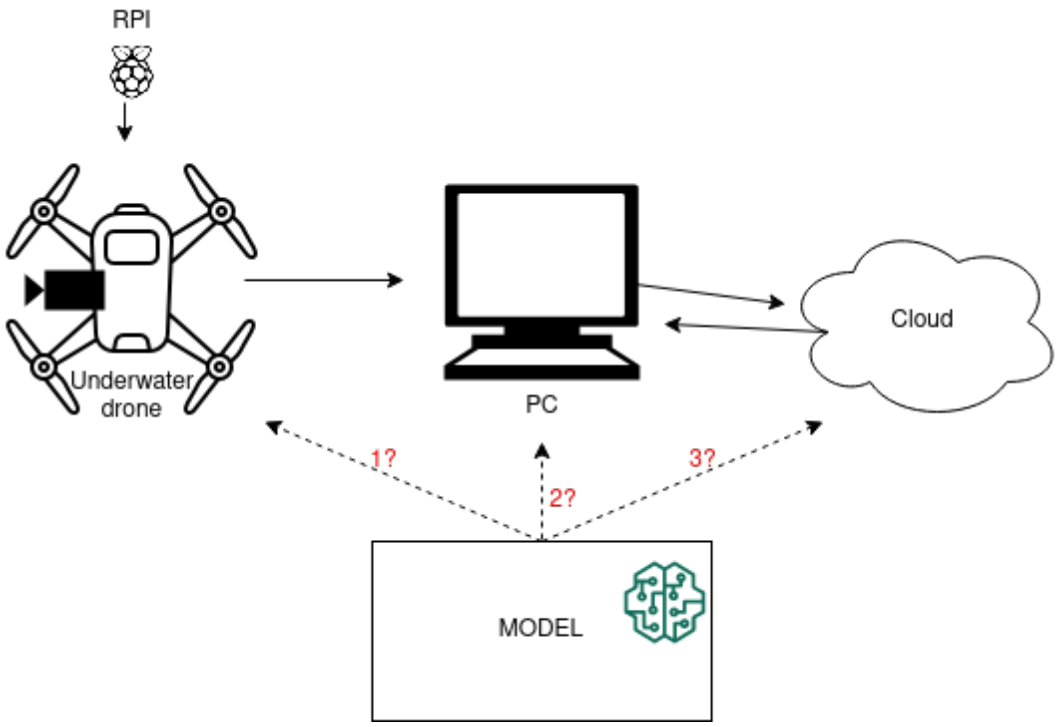


Figure 14: Architecture proposals for the system.

1. EDGE setup with a drone In this case the model is deployed on the GPU accelerator device attached to the drone, where all the processing and computation happens within the device. The drones' video feed will be used directly for the input to the object detection model, which in turn will produce final detection.

One of the main problems with particular approach is in the hardware limitations. In this case with Jetson NANO (provided by AAU), the device where model is deployed is not powerful enough for retraining the model, it has to be trained using other means and later redeployed on the device every time additional training is conducted. Moreover, this approach is highly dependent on the performance capabilities [83] of the accelerator device which in turn might not meet the processing requirements needed for the optimal performance of the model.

In this case, it is required to upgrade the hardware to improve performance, which is not always possible considering the limited physical space on such devices. However, it is possible to increase performance with a hardware accelerators, that are made specifically to perform functions more efficiently than it is possible to achieve with software. Tensor Processing Unit (TPU) is an example of a hardware accelerator[84] designed specifically to accelerate artificial intelligence applications, such as machine learning, NN's and vision tasks. Benefit of particular approach is that it is possible to achieve real-time detection as the whole system is unified and the processing is done by internal components without employing external resources.

2. Local Machine implementation: One of the options is to deploy the model locally on the computer device, that is going to host the model and run the object detection. In this case, training process and evaluation is performed on the device itself, as well as processing for the object detector is dependent on the device's hardware. The device running on CPU will not have a significant performance for the task, as neural networks require major computational resources. Instead, a GPU should be used to amplify the speed of the model, to achieve higher frame rates[85]. Additionally, following local setup allows to choose the most suitable framework for the development, instead of, for example, being locked with a particular choice that the cloud vendor has made available. Moreover, local implementation does not limit with the choice of the models as it can be an end-to-end method, or a black box algorithm.

3. Cloud setup implementation: Another approach is to make the system architecture cloud-based. Meaning the entire process will happen on the cloud platform, including training, processing input and storing the model. The deployment of the model is also possible at the cloud but it results in a web based object detection application. The benefits of particular approach are for example, vast infrastructure of cloud vendors. The provided services aid in the complete process of model preparation, including evaluation and monitoring of the model. As well as provide computational resources, allowing significantly reduce training time by selecting high-end hardware virtualization. However, using such services and hardware is usually adds up in cost, it is still a good option for prototyping while choosing pay-as-you-go[86] option.

The drawback of deploying a model in the cloud means that the input needs to be transferred over the network for processing, which would result in processing overhead, such as computation time and/or bandwidth, which is already a major disadvantage. Additionally, potential problem appears in the ability to freely modify the algorithm, where not all cloud vendors allow to tune the underlying algorithms.

The above presented possibilities are all suitable for the task of object detection, the

specific choice however, depends on the particular application at hand. As the goal of this project is to build a neural network for underwater object detection, as stated in Chapter 1.1, the local approach shall be taken due to freedom of choosing appropriate algorithms and tools for the task, as well as ability to tune the algorithms.

4.2 Object detection methods

The following chapter describes the consideration between different algorithms for object detection task, identified in the Section 3, Chapter 3.2. The reviewed algorithms are elaborated on to determine the most suitable choice for the project.

Faster RCNN

As defined in Chapter 3.6.2 the R-CNN is computation and time consuming to train due to 2000 region proposals for a single image and additionally, spending 47 seconds for processing each test image[51]. Therefore, particular method is not going to be considered for the project, along with its successor Fast R-CNN. Instead, Faster R-CNN will be evaluated whether it suits the project.

The architecture of Faster R-CNN can be appreciated in Figure 15. Faster R-CNN is essentially RPN and Fast R-CNN. An input image is fed to the convolutional network to produce feature map. Instead of Selective Search applied to feature map to predict region proposals, RPN is used. The predicted region proposals come out different sizes and are reshaped by RoI pooling layer, which in turn further classifies the image in proposed region as well as predicts offset values for bounding boxes[87]

Ultimately, Faster R-CNN achieves 5 FPS on a GPU and state of the art accuracy of 73.2% mAP on PASCAL VOC 2007 and 70.4% mAP on 2012[88, p. 2]. Despite the high indicators on accuracy, the frame rate is not suitable for real time application, thus alternative methods should be assessed[51].

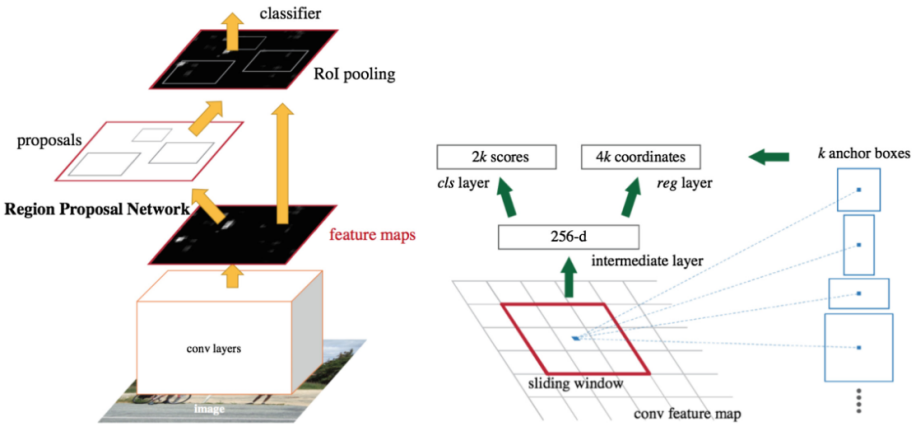


Figure 15: Left: Region Proposal Network. Right: Architecture of Faster R-CNN. Taken from:[51]

YOLOv3

Contrary to Faster R-CNN, YOLO excels in speed at the cost of accuracy. As mentioned in Chapter 3.6.3, base model is processing images at 45 FPS and Fast YOLO can reach 155 FPS. One of the benefits of YOLO, as a single shot detector, it looks at entire image once, as a result generating fewer false positive opposed to region proposal methods[88]. One of limitations for this algorithm is that each grid cell predicts two bounding boxes and can have only one class, this puts strong spatial constrains on bounding box predictions, thus confining amount of nearby objects the model can predict[53]. Additionally, the model struggles to generalize objects with new and/or unseen aspect ratios and configurations, because it learns the predictions from bounding box data[53]. It is worth to mention, that region based algorithms use *mean Average Precision* (mAP) score to compare performance, but Single shots use *Average Precision* AP. The difference is that mAP is just averaged values of different class AP's. The performance of YOLOv3 can be seen in Figure 16.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 16: YOLOv3 Performance comparison with state-of-the-art detectors. Taken from:[55]

SSD

Single Shot Multibox Detector is addressing problems present in YOLO by incorporating anchors adopted in Multi-Box method, multi-scale representation and RPN[88]. Similar to YOLO, SSD is trained end-to-end, but contrary to YOLO, ground truth data in SSD is assigned to specific outputs in the fixed set of detector outputs. Unlike YOLO, SSD predicts different feature maps of different scales and separates predictors for different aspect ratios. The scale is handled by predicting bounding boxes after multiple convolutional layers, since each layer operates at a distinct scale various size objects can be detected.

In addition to being an object detector that can be integrated into systems needing detection components, the network also classifies the detected objects[58].

The network has significant improvement in speed for high-accuracy detection (59 FPS with mAP 74.3% on VOC2007test, vs. Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4%)[58].

All things considered, the suitable choices for the system would be to use either YOLO or SSD, where in case of YOLO, detection operates faster but with lower accuracy and SSD

is slightly behind in speed but excels in accuracy.

4.3 Data preparation

For the purpose of training and validating the machine learning algorithm a data set has to be prepared. It has to be constructed in a way for the model to understand the semantics of underwater environment as well as learn the object of interest to be detected. In the following chapter a summary of various open-source data sets identified in the Chapter 3.8 has been composed, while explaining their purpose and usability for the project.

Fish4Knowledge is an entire project framework which focuses mostly, among other things on: methods for capturing, storing and analyzing massive amounts of domain specific data. The open source data set they provide is manually labeled and consists of 23 classes of objects, being different species of fish[89]. The data set consists of 17 videos, where each is 10 minutes long with resolution of 320x240 and 25-bit color depth at a frame rate of 5 fps[90]. The consisting data include presence of domain specific features, such as dynamic background, high water turbidity, low contrast and crowded scenes. However, the ground truth is only available for a specific set of frames which are the most populated by objects of interest. The ground truth data is available in XML format with listed objects for each frame and their contour information with coordinates. Even though data set is focused on fish species and movement, it is of a particular interest for the project due to the reasons of availability of domain specific features which may help to train the model for the underwater object detection purpose. Additional appealing option is the XML formatted labels, that reduce the work needed to construct a custom data set from scratch.

JAMSTEC E-library of Deep-sea Images Also being publicly available, this database is a collection of deep-see videos and images of type-specific marine debris data, offering around 38 thousand hours of videos and 1.5 million pictures. The data have been accumulated by various dive surveys and are dating back to 1982[73]. Useful point is that the data is classified by shape and material, that could be valuable for this project. Additionally, there are types of seabed sediments defined for the data that could contribute to the context learning. However, the data is not labeled[73].

Amazon data set The particular data set is a composition of all products sold by Amazon[91]. Different type of data sets offered, consisting out of images, review data and ratings of items, there are also smaller subsets for experimentation for each product category. The appealing point in the data set is that the present objects are captured from different angles, as identified in the Chapter 3.8 However, due to the nature of the data being mostly text information, this data set was not considered for the purpose of this project.

COCO[92]

The focus of the data set is on object recognition in the context of scene understanding, hence the name Common Objects in Context. The data set contains 80 object classes, with 328 thousand images with 2.5 million labeled instances. Data set is suitable for image recognition/classification, object detection, segmentation, and captioning[93]. It is one of the most popular data sets for training deep learning applications, due to being well documented and segmenting images into categories and object, while additionally providing machine-

readable context captions and tags. Particular data set could be potentially used for this project as a starting point, due to its properties.

ImageNet It is a large data set of images organized similar to WordNet hierarchy. Based on concept known as *synset*, meaning synonym set, it is utilised to provide around 1000 images for each synset. Images in the data set are human-annotated and quality-controlled. The total amount of images is around 14 million, 1 million of which is with bounding box annotations and 1.2 million have SIFT features[94]. The data set is a good starting point for the prototype system, due to its size and providing a well maintained annotations.

PASCAL VOC Being a set of data sets, used in Pascal VOC challenge, it provides standardised image data sets for object class recognition as well as a common set of tools for accessing the data sets and annotations. Enables evaluation and comparison of different methods. Often used to benchmark implementations of different models[95].

Open Images Data set V6 Possibly the largest image data set with object location annotations. The data set contains around 9 million images with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives. Additionally, there are 16 million bounding boxes for 600 object classes over 1.9 million images[96]

For the data preparation step, it is essential to define what data will be used for training. It is considered a common practice to use pretrained models and retrain them using images for a specific domain at hand[97, p. 8] However, there is not always a prepared data set available and in such case, a custom data set has to be made. Moreover, both images and videos can be used for training the model. In both cases, the data has to be annotated. Meaning the objects in the images have to be identified and noted, so that the computer model understands and recognizes the object during training. Some available data sets already have this part covered. However, in case of custom objects to be used, they have to be annotated manually or using available labeling tools, identifying position(bounding boxes) and the label(object class). The annotation comes in a separate file, usually XML or CSV, containing the reference metadata of the object, such as position and label.

Based on the evaluation above, it can be concluded that there are available data sets for the purpose of object detection in underwater environment. Even though they are not specifically designed for the task at hand, there are some good options for base model such as *Open Images*, *ImageNet* and *COCO* which can be used in combination with *J-EDI* and *Fish4Knowledge* to understand the semantic context of the environment.

4.4 Evaluation metrics

To perform an evaluation of the model different evaluation metrics should be considered. These include Precision, Recall, True Positives and False Positives, Intersection Over Union (IoU) and Average Precision (AP), to name a few. In the following Chapter these concepts will be considered for general understanding and future evaluation of the model.

Intersection over Union measure is the constituent principle for most metrics, therefore

it should be looked at first. The definition of Intersection Over Union (IoU) is an overlap area between the ground truth and predicted bounding boxes divided by the area of union between them. Prediction is considered to be satisfactory when the resulting score is higher than 0.5 threshold[98]. This metric determines the amount of correctly made prediction and the amount of false positive results.

- True Positives are the correct detections where IoU is greater or equals to the 0.5 threshold.
- False Positives refer to erroneous detections where the IoU is bellow the threshold.
- False Negatives refer to an undetected ground truth bounding box.
- True Negatives are not considered for object detection task as there are infinite amount of bounding boxes in an image that should not be detected.

Furthermore, the assessment of object detection methods is based on the relationship of Precision and Recall, where Precision is defined as *"the ability of a model to identify only relevant objects*[99, p. 3] and Recall *is the ability of a model to find all relevant cases*[99, p. 3]

The relationship between the two can be viewed as a trade-off, while improving one metric the other decreases and vice versa. It is important to find right balance for the given task, whether it is high recall so the detector finds all ground-truth objects or high precision that only relevant objects are identified[100].

The Average Precision (AP) is another metric often used in object detection contests, such as [93]. It is defined as area under the precision-recall curve, and is calculated by averaging over multiple IoU values. Such approach yields better localization for object detectors [93] In turn, mean Average Precision (mAP) is calculated average for all AP values over all classes[101, p. 158-162].

The mentioned metrics aid in the process of analyzing the performance of the models and serve as a tool to compare them They are used not only at inference time but also in the training phase, to analyze how well the model underwent the training cycle. For example, tuning the precision-recall values on the trained model helps to achieve the desired outcome from the model depending on the task, however, a good object detector is considered when its threshold values vary while precision and recall stay high. For this project acknowledged important metrics that should be used include IoU, precision, recall, AP and mAP.

4.5 Requirement specification

In order to build a system prototype and have a basic working functionality defined for the future chapters of design and implementation, requirements have been composed according to description in Chapter 2.3. The requirements are stemming from literature review, Chapter 3.8 and Chapter 4. Typically the number of requirements for similar projects has to be much greater, but due to limited resources and project size it was decided to limit the amount.

Jointly functional and nonfunctional requirements form a list of (17) entries, as seen in Tables 4 & 5. Requirements that are the most important for the working prototype are presented in the chapter below.

ID	Description	Source	MoSCoW
RQ-1	The system must be able to use video feed from the drone as input for object detector.	Section 1.1	Must
RQ-2	The system must be able to detect object of interest from the input.	Section 1.1; 3.1	Must
RQ-3	The system must be able to detect objects within range of defined classes.	Section 3.8.5; 3.8.9	Must
RQ-4	The system must show the confidence score of the identified objects.	Section 3.8	Must
RQ-5	The system must draw bounding boxes for each detected object.	Section 3.8	Must
RQ-6	The performance of the system should be able to meet real time requirements.	Section 3.8.8; 3.8.9	Should
RQ-7	The system should show FPS of the incoming stream.	Section 3.8	Should
RQ-8	The system should identify multiple objects from the input.	Section 3.6.1	Should
RQ-9	The system should be able to perform in rough underwater conditions.	Section 3.6; 3.8.3; 3.8.5	Should
RQ-10	The system should detect objects in different orientations.	Section 3.6; 3.8.2	Should
RQ-11	The system could have possibility to save and store frames of identified objects.	Section 3.8 Exepriment observation	Could
RQ-12	The system won't detect objects partially buried in the ground.	Section 3.8.2	Won't

Table 4: Functional requirements of the system along with their source, description and prioritization.

ID	Description	Source	MoSCoW
RQ-1	The system must be able to provide detection feedback to the user.	Section 3.8	Must
RQ-2	Data augmentation techniques should be applied on the training data to improve performance of models.	Section 3.1; 3.8.3; 3.8.7	Should
RQ-3	The model should be trained utilising transfer learning method.	Section 3.8.4; 3.8.5	Should
RQ-4	The model should be fine-tuned for bottle detection task.	Section 3.8.3; 3.8.4	Should
RQ-5	The system should operate with at least 20 FPS.	Section 3.8.9 Experiment observation	Should

Table 5: Nonfunctional requirements of the system.

5 System Design

The chapter presents the proposed design of the system, where underwater drone is equipped with the object detection model. The decision making process was based on the requirements, specified in Chapter 4.2 as well as Chapter 4.5. First, the system overview is presented with an architectural diagram, followed by the explanation of the system components and their interconnection.

5.1 System overview

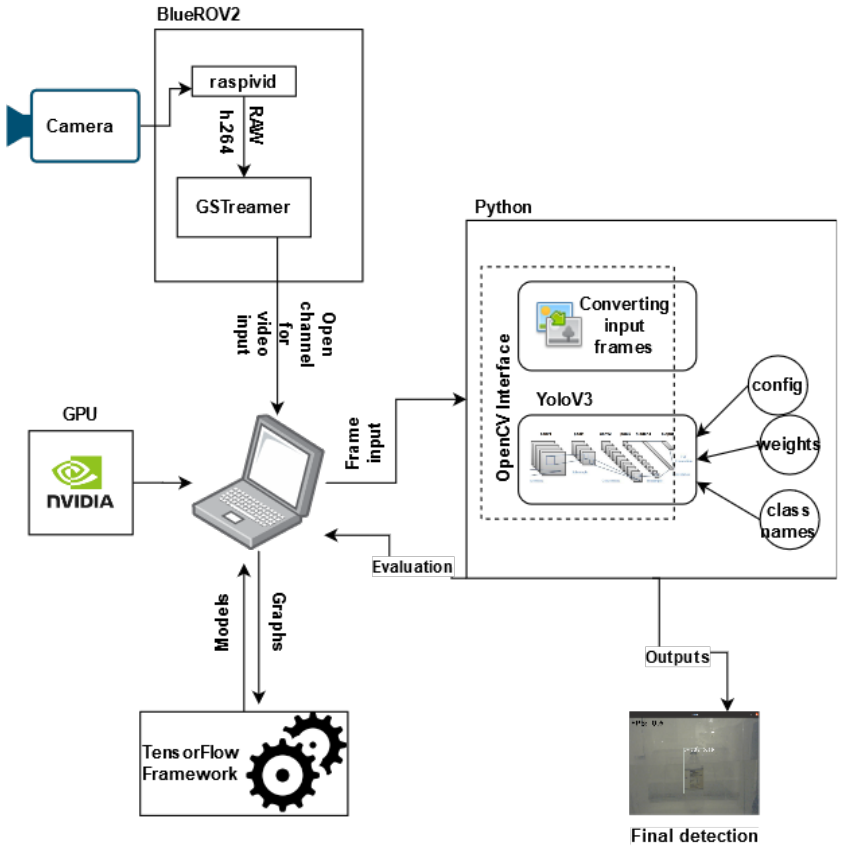


Figure 17: Overview of the system. Diagram contains picture from: [71]

The proposed system overview can be seen in the Figure 17. The whole system can be logically divided into three units, namely *Underwater drone* as an input source for the model, *Machine Learning framework* which performs the object detection on the input and the *Mediator* that handles the connection for the data flow. The system is going to be implemented locally on the PC, which acts as a middle link between the units. The drone is going to serve as an observer and will be responsible for the input to the machine learning algorithm. While being equipped with a camera and connected to the control station PC

with a teather, the drone will be submerged into the water. With the stream continuously running, the drone will transfer the feed to the control station PC.

The other important unit is the Machine Learning Framework. It provides neural network model for the object detection task and a toolbox that enables evaluation of the model.

Finally, the Mediator is the connection link between the entities. First, it hosts connection to the drone as well as runs a Python script where the model is deployed. Additionally, PC provides the system with computational power coming from the connected GPU to speed up the processing of the object detection.

The Python script is composed of two major parts, where first is managing the incoming video stream and second is running the object detection model, that applies object recognition on the video feed from the drone, showing the detection results.

5.2 Components and interconnection

Various components are included in the system to interconnect together. The drone is running various libraries to manage communication among the drones components as well as communicate with the control station PC. For the projects interest only relevant ones will be described. The main task for the drone is to perform visual observation, for that it utilises a mounted camera connected to the Raspberry PI inside the drone. The RPI is running a *raspivid*, a tool for capturing video with a camera module. The output from the tool is a raw footage that is encoded into h.264 format and sent over to another enclosed library - Gstreamer. It receives the footage, packs it and sends over a *udpsink* to the control station PC and to ground control station software, detailed architecture of the connection can be seen in Figure 4. Additionally, the split connection settings that enable second stream to control station PC can be viewed in Figure 19.

In turn, the first part of the script on control station PC, is performing receiver operations. It captures the stream from the Gstreamer and utilising OpenCV library transforms the stream into a *numpy* array.

The second part of the script performs the necessary configuration to perform object detection. The procedure includes defining the input source, loading of the model with necessary configuration files and processing of the input before final results are displayed. In the following Chapter 6 the working principles are described in detail.

6 Implementation

The chapter presents the implementation process behind the proposed system prototype. First, the development of the prototype was divided into different stages. First, it was necessary to assemble the drone and connect it with the control station. Second, was the implementation of a state-of-the-art object detector YOLOv3 with OpenCV in Python. Third, where the whole prototype system was assembled. The implementation decisions for every stage are summarized and described, together with screenshots of the corresponding code examples and building process.

6.1 Building the drone

The drone comes in a form of disassembled parts and has to be built from scratch. The process of assembly is relatively straightforward. It consists of 2 major parts: building the device and setting up the software. There are various guides on the web which cover the process. [REF: https://bluerobotics.com/learn/bluerov2-assembly/]

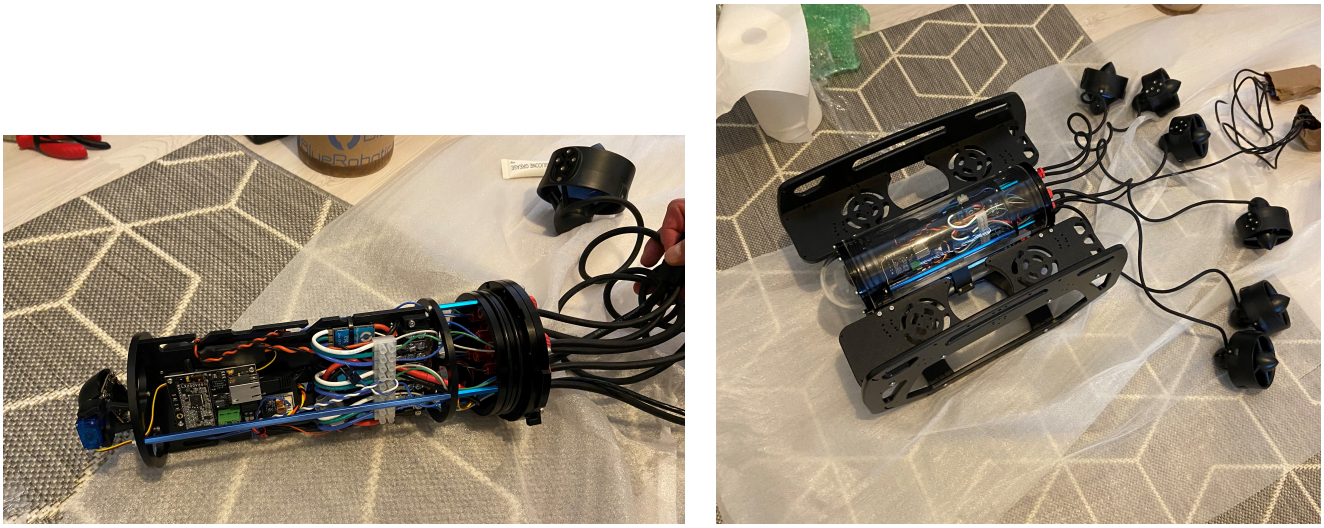


Figure 18: Left:Electronics enclosure. Right: Enclosure mounted on the frame with thrusters

The device construction starts with assembling the frame, and attaching a watertight container(battery enclosure), where in the future the power unit will be situated. Following, is to attach cabling for thrusters, lights, power supply and tether cable to the electronics enclosure capsule. It is worth to mention that drones produced from 2019 are shipped with preconnected microprocessors. The final step consists of mounting the electronics enclosure onto the frame, together with all the connected thrusters, lights and thether. Finally, fairings and buoyancy can be attached to the frame along with ballast, making the drone ready for the dive.

Software setup is also covered extensively by the official producer and can be familiarized with at: [102]. In order to operate the ROV, QGroundControl has to be installed on the topside computer enabling communication with the drone. Moreover, QGroundControl is a UI tool for managing, monitoring and operating the ROV. On the first setup, the drones sensors have to be calibrated with the software and the thruster configuration has to be adjusted.

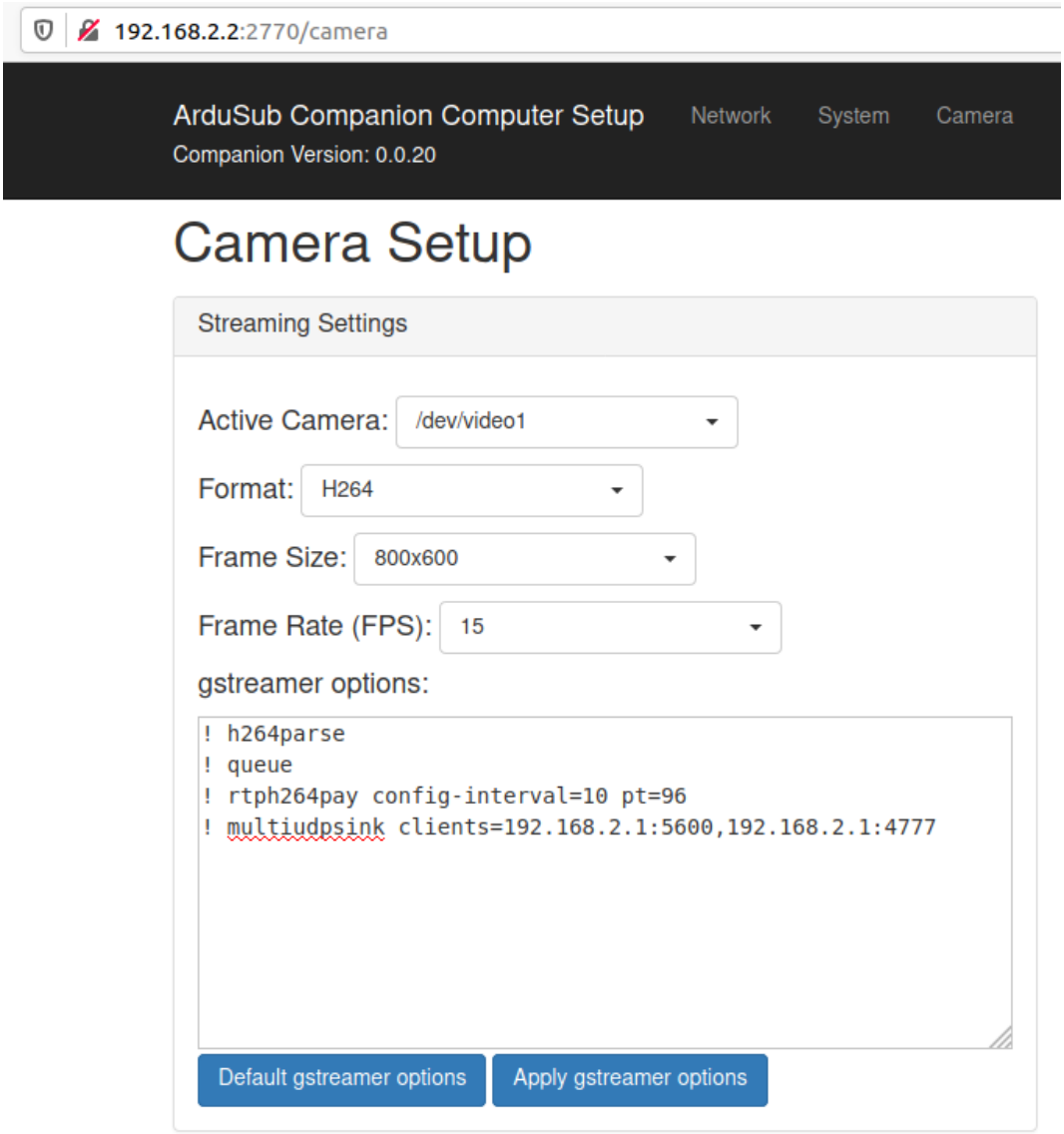


Figure 19: Drone's Camera settings view.

To run the object detection software on the drones' camera feed, the settings for the camera have to be adjusted. By default there is a single video feed channel coming from the drone to the topside computer and it is reserved for QGroundControll software. It is required to split the channel to be able to use it for object detection. For that Gstreamer

options have to be updated. Instead of using the default *udpsink* to specific IP and port address, *multiudpsink* is defined with additional IP and port, as can be seen in Figure 19.

6.2 Implementing the model

To build the system prototype it was decided to use the official YoloV3 model pretrained on the COCO data set[103]. The implementation was done in Python, utilising OpenCV library for video processing and frame manipulation. A script was composed particularly for object detection task which was initially performed on webcam feed. The code begins with importing the necessary libraries. Such as *numpy* for array manipulations and *cv2*, an open source library for computer vision and machine learning.

First the models parameters are initialised with weights, configurations and class names. The pretrained weights used for the prototype have been trained on MS COCO data set. The configuration of the model contains the entire information about the setup of the neural network, such as batch size, learning rates, activation functions, number of hidden layers and configuration of each particular layer, to name a few. The class names file contributes as name implies with classes that the model is trained on and can identify.

```
133 if __name__ == '__main__':
134     # Create the video object
135     # Add port= if is necessary to use a different one
136     video = Video(port=4777)
137 
138     net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
139     classes = []
140     with open('coco.names', 'r') as f:
141         classes = f.read().splitlines()
142 
```

Figure 20: Code sniped for network initialization

The next part is to read the input from the video feed and process each frame. It is done through *blobFromImage* function which converts the input to blobs for the neural network. It scales the image pixel value and resizes the image, without cropping, to a certain size of 416x416. Mean substraction is not performed, hence the zero range, as seen in the Figure 21. The resulting blobs are set as an input to the network.

```
156         height, width, _ = frame.shape
157         frame_id += 1
158         blob = cv2.dnn.blobFromImage(frame, 1 / 255, (416, 416), (0, 0, 0), swapRB=True, crop=False)
159         # Used to set input from the blob to network
160         net.setInput(blob)
```

Figure 21: Code snippet of blob function

In the next step, in Figure 22, the last layers of the network are identified. For that purpose the *getUnconnectedOutLayers* function is utilised that returns the names of the unconnected layers, which are the last output layers. Then the *forward* function is utilised to get the output from the identified output layers.

```
162         # Here getting outputlayers names and parsing them in net.forward func.
163         output_layers_names = net.getUnconnectedOutLayersNames()
164         layerOutputs = net.forward(output_layers_names)
165
```

Figure 22: Code snippet of assigning output layers

The following *for* loop shown in Figure 23 is processing the candidate objects, resulting with arrays with number of classes and 5 elements, where 4 represent the center coordinates of the bounding box and its height and width, and the last element shows the confidence of a bounding box enclosing an object. Box with the highest confidence score gets assigned to the corresponding class. The corresponding class_id's, bounding boxes and confidence scores are appended to the arrays.

```
171         for output in layerOutputs:
172             for detection in output:
173                 scores = detection[5:]
174                 class_id = np.argmax(scores)
175                 confidence = scores[class_id]
176                 if confidence > 0.5:
177                     center_x = int(detection[0] * width)
178                     center_y = int(detection[1] * height)
179                     w = int(detection[2] * width)
180                     h = int(detection[3] * height)
181
182                     # possitions for the upper left corner of imgs
183                     x = int(center_x - w / 2)
184                     y = int(center_y - h / 2)
185
186                     boxes.append([x, y, w, h])
187                     confidences.append((float(confidence)))
188                     class_ids.append(class_id)
189
```

Figure 23: Code snippet of a candidate calculation

The accumulated possible predictions are cut down by the confidence threshold comparison of *NMSBoxes* function, in Figure 24. Meaning, when the confidence of a bounding box is less then the set threshold, the box is dropped and not considered for prediction, reducing number of overlapping boxes. With lower threshold there is less accuracy on de-

tecting overlapping objects, with threshold to high, multiple bounding boxes appear for a single object.

```
190         # ususally >1 bounding boxes per object, here we keep only the most certain boxe
191         # print(len(boxes))
192         indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
193         # print(indexes.flatten())
```

Figure 24: Code snippet of NMS filtering

```
198     if len(indexes) > 0:
199         for i in indexes.flatten():
200             x, y, w, h = boxes[i]
201             label = str(classes[class_ids[i]])
202             confidence = str(round(confidences[i], 2))
203             color = colors[i]
204             cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
205             cv2.putText(frame, label + " " + confidence, (x, y + 20), font, 2, (255, 255, 255), 2)
```

Figure 25: Code snippet of visualise final detection

The final *'if'* statement, in Figure 25, checks if there are any objects identified, by comparing the lenght of *indexes* variable. The flatten function transforms the initial 3 channels into a single channel and assigns a label to the object identified with highest confidence score and an associate correctly assigned bounding box. Methods *cv2.rectangle* draws the bounding box on the frame and *cv2.outText* places the label and confidence of the object detected. In addition, in Figure 26, frames per second are calculated and displayed within a frame. The script is running in a loop that will break the transmission of the frames when an assigned key will be pressed.

```
206             #FPS output
207             elapsed_time = time.time() - starting_time
208             fps = frame_id / elapsed_time
209             cv2.putText(frame, "FPS: " + str(round(fps, 2)), (10, 50), font, 3, (0, 0, 0), 3)
```

Figure 26: Code snippet of FPS visualisation

6.3 Final Assembly

Next step was to utilise the video feed from the drone's camera and apply object detection. For this purpose a Python script was utilised which employs the Gstreamer library. It is used by the *Raspberry PI* on the drone to interface the video frames from the camera, encode into h.264 format and send over the UDP socket to the topside computer's software.

A problem was encountered while performing the implementation on Windows PC. The Gstreamer library used for operating streaming media on from the drone, required special installation in order to work properly. More specifically, the compatibility of versions for different tools and libraries was required. A particular version of the Gstreamer library is

compatible with particular version of Python. On top of that, another SDK of specific version was required for Gstreamer to correctly interoperate with Pythons development environment. Thus, a decision was made to switch to a different OS, like Linux, which ought to be more user friendly.

The script was reconstructed on Ubuntu 20.04 operating system and after successful configuration of the needed dependencies, Gstreamer library was operating correctly. From there it was possible to prepare the script to produce a video stream from the drone. As a base, a Python script was used, provided in the documentation of the drones software[29] , therefore its not going to be described in detail.

```python
@staticmethod
def gst_to_opencv(sample):
    """Transform byte array into np array
    Args:
        sample (TYPE): Description
    Returns:
        TYPE: Description
    """
    buf = sample.get_buffer()
    caps = sample.get_caps()
    array = np.ndarray(
        (
            caps.get_structure(0).get_value('height'),
            caps.get_structure(0).get_value('width'),
            3
        ),
        buffer=buf.extract_dup(0, buf.get_size()), dtype=np.uint8)
    return array
```

```python
        self.start_gst(
            [
                self.video_source,
                self.video_codec,
                self.video_decode,
                self.video_sink_conf
            ])

        self.video_sink.connect('new-sample', self.callback)

    def callback(self, sink):
        sample = sink.emit('pull-sample')
        new_frame = self.gst_to_opencv(sample)
        self._frame = new_frame

        return Gst.FlowReturn.OK
```

Figure 27: Left: Gst to OpenCV convertion function. Right: Gst 'start' and 'callback' functions

Figure 28: Code snippet of Gst streaming script

In general, the script defines video capture class constructor for BlueROV. The main idea is that the stream from the drone is encoded, packed and sent by Gstreamer library and needs to be converted to OpenCV in order to manipulate the frames for object detection. The receiver script is responsible to initialize pipeline and sink where video will be buffered. Next the frames are received, decoded and finally converted to OpenCV.

In order to run object detection model on the drones video feed the scripts were combined together. The sender script had to be updated to capture the pipeline prepared for object detector. The *Video* class is modified to point to video source, defined by the *port=4777* variable, as shown in Figure 20. Additionally, the input frame sizes are adjusted for the object detection mode used, then the model is initialized as described in Chapter 6.2 Implementing the model.

7 Testing and Evaluation

To evaluate the performance of the system prototype the testing of object detector in three separate experimental settings is described in the following chapter. Additionally, a discussion on identified implications is provided. Finally, the chapter is concluded with possible future improvements identified during the various stages of the project.

7.1 Experiments and Results

In order to test the performance and evaluate the system, three experiments were conducted. Even though the model did not undergo the training step on a custom data set, the performance of the a generic model pretrained on COCO data set will be evaluated instead.

Limitations: There were some limitations in the testing step of the project. First issue appeared with the topside computer. The dead battery issue and lack of access of power source for the laptop prevented from performing experiments in an real world environment. Secondly, the experiment was expected to be performed in the university's premise, however, due to Covid-19 virus threat it was not possible to utilise provided facilities. Additionally, the lack of necessary equipment to submerge the drone into the water have led to replicating the set up is a staged environment. Thirdly, since the object detector is deployed on a local machine with only CPU support, the observed models inference did not meet real time performance. To achieve better results a high-end GPU hardware needs to be utilised. In the light of the issues described the experiment is going be performed in a staged environment.

The used equipment includes a transparent plastic tank(dimensions - 56x39x42cm/65l) filled with water. The drone is positioned right in front of the water tank. And finally an objects of interest(plastic bottles) that are going to be shown to the camera for detection.

The hardware utilised for the experiments includes a Lenovo Thinkpad T570 running Intel(R) Core i7-6600U CPU@2.6GHz processor, and a BlueROV2 underwater drone.

Two models have been observed, YOLOv3 and Tiny-YOLOv3. The difference between two is the latter is a more lightweight model, specifically designed to run with devices with constrained computational resources, such as mobile devices, or microprocessors. However, the speed of the model compromises accuracy significantly as was observed during the experiments. The benchmark of YOLO model presented in Figure 29, shows the comparison of performance for state-of-the-art object detection models on a high-end hardware, described in [53, p. 1], for comparison.

Performance on the COCO Dataset

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

Figure 29: YOLO benchmark on COCO dataset. Taken from:[103]

First test is made above water, where the generic model should perform the best as it was trained on a data set of common objects in context as described in Chapter 4.3, thus not being subjected to the constrains of underwater environment. In this case, it is expected for the system to perform exceedingly compared to the following experiments.

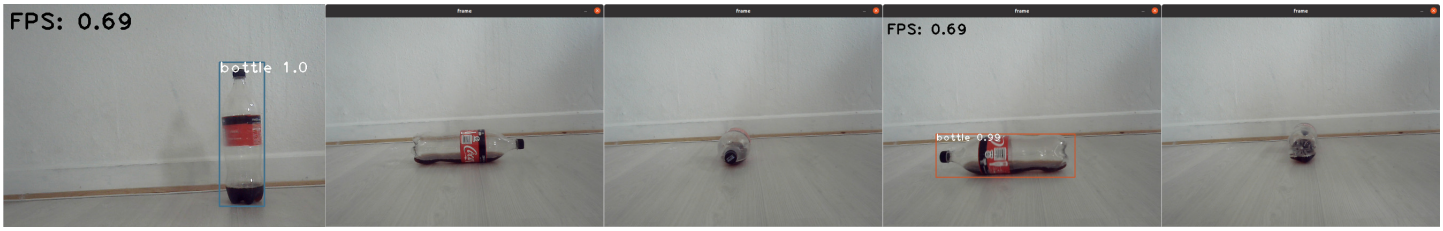


Figure 30: Single object positions for test detection

Few variants of the setup have been done. First, placing a single objects in front of the camera for detection, while changing its position, distance and perspective.

In the first experiment, the object is detected with higher accuracy in the upright positions. It is interesting to note that the object being in horizontal position is only detected when facing left, as seen in Figure 30. While the object was not positioned vertically, detector struggled to identify the bottle. The observed results point out that there was insufficient data of object orientations when training was performed.

Similar results are produced when multiple objects are placed for detection. Even though three out of four bottles were identified with adequate accuracy, the produced results shown in Figure 31a indicate the struggle of YOLO detector to distinguish multiple

(a) Multiple object positions for test detection

(b) False positive detection

Figure 31: Experiment 1: Above water detection

overlapping objects, where the bounding boxes are wrongly positioned for the two horizontally placed object. Moreover, in a multiple object setting various false positive results are produced, shown in the Figure 31b. Some individual features of the bottle object are recognized by detector as a different class assigning wrong label. It is assumed that such result occurs due to the context nature of YOLO detector as described in Chapter 3.6.3. Even though there were errors in bounding box localization's and object recognition, the detection model was able to successfully identify the object of interest in most cases.

Second experiment is performed with the clear water setting. In this case the performance of the system will be examined in regards to the minor environment constrains, such as slight water turbidity and object position, so that it has lose movement within the environment. The Figure 32 shows a screenshot from the drones camera where the bottle is being placed into the tank and gets detected with 70% precision and around 0.6 FPS. It can be observed that while the person is submerging the object into the tank, the detector does not recognize the hand of the person and does not produce a label for it. The screenshot on the right, in Figure 33, shows the submerged bottle object being detected with high precision of 95% at 0.94 FPS on a CPU. Here the water tank was intentionally placed at a distance from the camera, so that the transparent object merges with the background and becomes barely visible, in order to impose additional complications for the detector. Even though the object is recognized with high precision the localization of the object is erroneous as only the sticker on the bottle is detected but the edges of the bottle itself are not recognized. This reinforces the notion of known problem of object detectors struggling to distinguish edges [38][104][68] of objects in challenging environment.

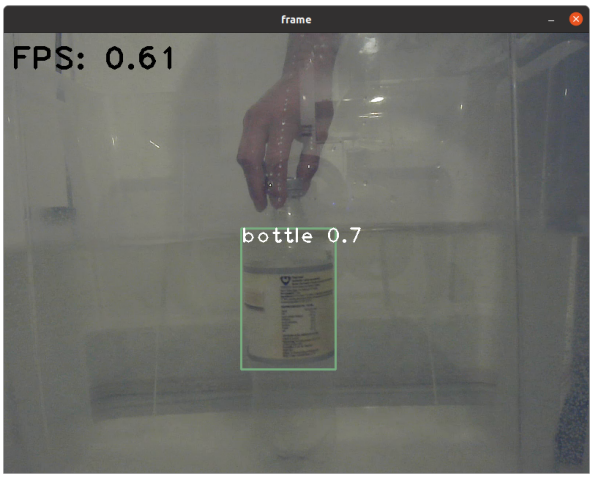


Figure 32: Experiment 2. Placing the object

Figure 33: Experiment 2. Object detection

The third experiment is performed as closely as possible to the real environmental constraints, such as low light conditions and high water turbidity. To achieve such constraints, as close as possible to real life context, water tank was filled with murky water.

It is worth to note, that as experiment proceeded, it became apparent that when drone placed directly underwater detector would operate differently. Here, however, its' strong headlights needed for low light conditions reflected from the container affecting the purity of the experiment.

Nonetheless, the results of the experiment, presented in Figure 34, point out localization problems, where among three screenshots only the one on the right has more accurately identified bounding boxes, covering most of the objects' surface. In the second screenshot, position of the bottle impacted the recognition and only one out of two objects was detected successfully. It is also worth to note that even though two objects are tightly positioned together, in the picture on the right, both are detected, though with overlapping bounding boxes.

Additionally, among three presented results, objects were recognized with relatively high accuracy 84%, 96%, 99% and 97% respectively, while maintaining FPS at 0.7.

Figure 34: Dirty water experiment

7.2 Discussion

Model Training Implications

While defining the requirements for the system prototype it was mentioned that *"The system must be able to detect objects within range of defined classes"* (in Chapter 4.5, Requirement number #3), which was limited in this project case to be a single class *Bottle*. However, the model was not limited to only that class intentionally. For the sake of the experiment to be more real-life like, more diverse cases should be identified, like in Figure 31b.

Meaning, that specific context of litter pollution is very diverse and complex. There are different classes of garbage present, as mentioned in Chapter 1, such as bags, bottle caps, cans, cigarette butts and unusual objects as refrigerator[7]. Not to mention the variety of the same litter type: even a simple object as bottle can have meridian shapes, forms, colors, etc. The model should be able to recognize this heterogeneity.

Therefore it gives an insight for the future improvement on how to address model training phase. It is purposelessly for this context(underwater litter pollution) to gather the data set with a particular object, instead a general purpose model proves to be more advantageous, while being supplemented with diverse data from different contexts, angles and domains in order for the model to exceedingly perform in real world scenario.

YOLOv3 vs YOLOv3-tiny Implications

Each experiment was conducted in two iterations, where two models have been observed, namely YOLOv3-416(called "base" further on) and YOLOv3-tiny. The idea was to compare the working capacity and performance of both models, as the tiny version should operate much faster, while compromising 40% in mAP score in comparison with the base model, as identified in benchmark results shown in Figure 29.

In fact, during the first experiment the base model did not exceed the speed of 1.3 FPS while exhibiting acceptable accuracy on the hardware defined in previous Chapter 7.1. However, the tiny version maintained the speed of 8 FPS though the accuracy drop was noticeable, the model performed decently.

Nevertheless, during the second experiment when the first constraints were introduced the tiny version of the model experienced a substantial performance drop. It was determined that the accuracy of tiny model was significantly lower compared against the base model. Unfortunately, the model could not recognize the object at all, therefore it was decided to drop the tiny version and continue the experiments with only the base model.

7.3 Future improvements

The following chapter presents the potential improvements and additions to the system identified during various stages of the project development. The enhancements presented refer to separate components of the prototype as well as the processes of their preparation.

7.3.1 Data preparation and utilization of data augmentation techniques

One of potential improvements to the system that should be considered is a thorough data preparation step. As it was observed in the experimental phase, as well as pointed out by identified research the object detection is strongly dependent on the data utilised for training. The problem in question is not only the sufficiency of the data, but also its quality and proper application. As seen in Chapter 3.8 utilization of various data augmentation techniques (such as Horizontal mirroring, Affine transformation, Sub-sampling, etc) pointed out by [74],[77],[76],[80] improves detection capabilities greatly and thus should be incorporated into the system, especially while dealing with complex environments such as underwater domain.

7.3.2 NN Model training with modern techniques

A substantial part of the system involves the choice of the underlying model and the training process of it. Firstly, it was identified to be a good practice to chose a pre-trained general purpose model as a base model and apply novel learning techniques such as transfer learning and deep learning. The suggested approach is identified to contribute to the certain task the model is ought to solve as well as increasing its performance for the given task. Approach identified in [71],[74], [77],[76], suggests to incorporate a number of interconnected NNs to raise the accuracy and precision of the identification and detection.

7.3.3 Functional performance

Since the proposed system is only a prototype with limited basic functionality, including top of the line components to the system would increase its efficiency. The proposed system is utilising a CPU for inference that is not designated to perform complex computations, thus not being able to meet the real-time requirements. This could be addressed by incorporating a powerful GPU accelerated hardware that would drastically boost the computational speed, therefore increasing the performance of the detector.

The model used for the system prototype was not trained specifically for the task at hand, as mentioned in earlier Chapters. Therefore, this area would be one of the points of improvement to the system. By utilising a number of data sets from different domains, such as underwater data for context, data set of man made materials for diversity in conjunction with the techniques mentioned previously it would contribute to the improved performance of the NN.

Additional, improvement to the system prototype would be incorporation of the evaluation functionality. It enables the assessment of the network performance during training process as well as compares the operation to the benchmark results.

8 Conclusion

The project carried out to investigate the possibilities of underwater object detection and an attempt to create a system prototype to perform underwater analysis of debris in real-time. The main problem formulation that guided the flow of the project goes as following:

"How to develop an object detection system that would perform real-time underwater analysis to detect and identify objects of interest within the underwater environment?"

Additionally, to the main research question, three sub-questions were introduced to outline the project scope and set up the scene for the prototype development. The answers to each research question jointly provide a comprehensive answer to the main problem formulation.

- How can machine learning algorithms be utilized to perform underwater analysis?

The project has succeeded to utilize the machine learning algorithm to perform the underwater analysis, namely the object detection task to identify litter objects in underwater setting. The proposed system utilised YOLO model and was able to classify and localize single as well as multiple objects from the input, while providing satisfactory accuracy of the predictions. Even though, the conducted experiments showed the potential of the system to perform the object detection task underwater, the system failed to meet the real time requirement due to hardware limitations. Additionally, the system was not tested in real world environment.

- How to develop a system architecture that would employ underwater drone for the object detection task?

To develop a system architecture for such solution it was required to apply knowledge from different areas, such as Machine Learning, Neural Networks, implementation frameworks and tools, existing solutions and state of the art technologies as well as Computer Vision field. Additionally, three possible solutions were identified on how to incorporate an underwater drone BlueROV2 into the system prototype, which are cloud, edge and local setups. All three solutions are possible in practice to build a satisfactory prototype, while some approaches have drawbacks and pose different limitations, the local implementation was chosen as a candidate solution due to freedom of choice of algorithms, frameworks and possibility to tune the algorithm according to the posed problem.

- How to approach the problem of computer vision in rough underwater environment?

To address the challenges of computer vision in rough underwater environment a research on existing solutions was conducted, where various approaches to the problem were acknowledged such as data augmentation, transfer learning and deep learning. Various methods were identified for image processing and enhancement such as Horizontal mirroring and

down-sampling. However, a novel approach was not proposed as it was out of scope of the project.

In conclusion, it can be said that the main research question was answered successfully with the aid of the sub-questions. The project concluded with a partially working prototype system that although did not meet real time performance capabilities and was not tested in real world setting nevertheless showed satisfactory performance in the laboratory setting.

References

[1] Gartner, *Gartner says AI augmentation will create $2.9 Trillion of business value in 2021*, 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-08-05-gartner-says-ai-augmentation-will-create-2point9-trillion-of-business-value-in-2021.

[2] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, A. Luccioni, T. Maharaj, E. D. Sherwin, S. Karthik Mukkavilli, K. P. Kording, C. Gomes, A. Y. Ng, D. Hassabis, J. C. Platt, F. Creutzig, J. Chayes, and Y. Bengio, *Tackling climate change with machine learning*, 2019. [Online]. Available: www.climateinformatics.org.

[3] Denmark.dk, *Sustainability in Denmark — The pioneer in promoting sustainability*, 2020. [Online]. Available: https://denmark.dk/innovation-and-design/sustainability.

[4] *Components — Trash Pollution — Ocean Health Index*, 2020. [Online]. Available: http://www.oceanhealthindex.org/methodology/components/trash-pollution%20http://www.oceanhealthindex.org/Components/Trash_Pollution.

[5] *This Danish scheme is offering free kayak rides ... for picking up trash — World Economic Forum*. [Online]. Available: https://www.weforum.org/agenda/2019/06/danish-scheme-free-kayak-rides-for-picking-up-trash/.

[6] *GREENKAYAK - KAYAK REPUBLIC*. [Online]. Available: https://kayakrepublic.dk/diverse/greenkayak/.

[7] *The Why – Collecting trash in Copenhagen and presenting WHY PLASTIC? - Ren Havn 2020*, Aug. 2020. [Online]. Available: https://www.thewhy.dk/news/collecting-trash-in-copenhagen-and-presenting-why-plastic-ren-havn-2020.

[8] "Save the Baltic Sea STIFLED SEAS," [Online]. Available: https://www.stromma.com/en-dk/copenhagen/about-us/environmental-policy/wwf-baltic-sea/.

[9] *Seabins clean Nyhavn canal area in Copenhagen — TheMayor.EU*, Dec. 2019. [Online]. Available: https://www.themayor.eu/en/seabins-clean-nyhavn-canal-area-in-copenhagen.

[10] D. L. Rizzini, F. Kallasi, F. Oleari, and S. Caselli, "Investigation of vision-based underwater object detection with multiple datasets," *International Journal of Advanced Robotic Systems*, vol. 12, Jun. 2015, ISSN: 17298814. DOI: 10.5772/60526.

[11] M. Samadi Gharajeh, "Waterative Model: an Integration of the Waterfall and Iterative Software Development Paradigms," *Database Systems Journal*, vol. X, pp. 75–81, 2019.

[12] D. Y. Kim, "A design methodology using prototyping based on the digital-physical models in the architectural design process," *Sustainability (Switzerland)*, vol. 11, no. 16, Aug. 2019, ISSN: 20711050. DOI: 10.3390/su11164416.

[13] A. Butterfield and G. E. Ngondi, Eds., *A Dictionary of Computer Science*. Oxford University Press, Jan. 2016, ISBN: 9780199688975. DOI: 10.1093/acref/9780199688975.001.0001. [Online]. Available: https://www-oxfordreference-com.zorac.aub.aau.dk/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-5781?rskey=KUdxtk&result=6321%20https://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975%20http://www.oxfordr.

[14] J. Volchko, *Prototyping Methodology: Steps on How to Use It Correctly*, Jun. 2017. [Online]. Available: https://www.lumitex.com/blog/prototyping-methodology.

[15] *Databases and suppliers - Aalborg University Library (AUB) - Aalborg University (AAU)*. [Online]. Available: https://www.en.aub.aau.dk/find-material/databases.

[16] *Google Scholar*, 2020. [Online]. Available: https://scholar.google.com/.

[17] D. P. F. Möller, "Systems and Software Engineering," in *Animal Genetics*, 5, vol. 39, Sep. 2016, pp. 235–305. DOI: 10.1007/978-3-319-25178-3{_}6. [Online]. Available: https://www.iso.org/standard/71952.html.

[18] J. Pérez, *Karl Wiegers Joy Beatty Software Requirements*. 2013, p. 672, ISBN: 978-0-7356-7966-5. [Online]. Available: https://books.google.dk/books/about/Software_Requirements.html?id=nbpCAwAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false%20https://www.academia.edu/31604577/Karl_Wiegers_Joy_Beatty_Software_Requirements.

[19] N. R. Council, *Undersea Vehicles and National Needs*. National Academies Press, Nov. 1996. DOI: 10.17226/5069.

[20] Triton, *Triton Submarines*. [Online]. Available: https://tritonsubs.com/products-services/all-subs/triton-36000-2/%20http://tritonsubs.com/products-services/all-subs/.

[21] *Deepest Ever Submarine Dive Made by Five Deeps Expedition*, 2019. [Online]. Available: https://www.maritime-executive.com/article/deepest-ever-submarine-dive-made-by-five-deeps-expedition.

[22] Germanischer Lloyd Aktiengesellschaft, *Rules for Classification and Construction Ship Technology- Underwater Technology - Unmanned Submersibles (ROV, AUV) and Underwater Working Machines*, 2009. [Online]. Available: www.gl-group.com%20http://www.gl-group.com/infoServices/rules/pdfs/gl_i-5-3_e.pdf.

[23] S. Watson, D. A. Duecker, and K. Groves, *Localisation of unmanned underwater vehicles (UUVs) in complex and confined environments: A review*, Oct. 2020. DOI: 10.3390/s20216203. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6203.

[24] Smithsonian Ocean Team, *Underwater Robots Explore the Ocean*, Oct. 2010. [Online]. Available: https://ocean.si.edu/planet-ocean/tides-currents/underwater-robots-explore-ocean.

[25] J. Chen, "Research on Development Status and Trend of Underwater Vehicle Technology," Tech. Rep., 2019.

[26] *BlueROV2 - Affordable and Capable Underwater ROV*. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/.

[27] *Blue Robotics Tech Spec*. [Online]. Available: https://www.bluerobotics.com/store/.

[28] ArduPilot, *Open Source Drone Software. Versatile, Trusted, Open. Ardupilot*, 2019. [Online]. Available: https://ardupilot.org/%20http://www.ardupilot.org/.

[29] *ArduSub · Overview · GitBook*. [Online]. Available: http://www.ardusub.com/%20https://www.ardusub.com/.

[30] "ArduSub_Software_in_the_loop," [Online]. Available: http://www.ardusub.com/introduction/required-software.html%20http://www.ardusub.com/images/software-components.jpg.

[31] O. Simeone, *A brief introduction to machine learning for engineers*, 2018. DOI: 10.1561/2000000102.

[32] T. O. Ayodele, "Types of Machine Learning Algorithms," *New Advances in Machine Learning*, pp. 19–49, 2010, ISSN: 978-953-307-034-6. [Online]. Available: www.intechopen.com.

[33] I. G. Courville, Y. Bengio, and Aaron, "Introduction to Deep Learning 1. What is Deep Learning?" *Nature*, vol. 29, no. 7553, pp. 1–73, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[34] A. I. Georgevici and M. Terblanche, "Neural networks and deep learning: a brief introduction," *Intensive Care Medicine*, vol. 45, no. 5, pp. 712–714, 2019, ISSN: 14321238. DOI: 10.1007/s00134-019-05537-w. [Online]. Available: https://doi.org/10.1007/s00134-019-05537-w.

[35] F. Vázquez, *Deep Learning made easy with Deep Cognition – Becoming Human: Artificial Intelligence Magazine*, Dec. 2017. [Online]. Available: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351.

[36] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation functions: Comparison of trends in practice and research for deep learning*, 2018.

[37] H. Sharma, *Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning*, Jan. 2017. [Online]. Available: https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e.

[38] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016, ISSN: 18728286. DOI: 10.1016/j.neucom.2015.09.116.

[39] A. Aghaebrahimian and M. Cieliebak, "Hyperparameter tuning for deep learning in natural language processing," in *CEUR Workshop Proceedings*, vol. 2458, 2019.

[40] F. R-cnn, "Computer Vision — A journey from CNN to Mask R-CNN and YOLO -Part 1," pp. 1–18, Jul. 2020. [Online]. Available: https://towardsdatascience.com/computer-vision-a-journey-from-cnn-to-mask-r-cnn-and-yolo-1d141eba6e04.

[41] S. Gollapudi, *Learn Computer Vision Using OpenCV*. 2019, ISBN: 9781484242605. DOI: 10.1007/978-1-4842-4261-2.

[42] R. E. Neapolitan, "Neural Networks and Deep Learning," in *Artificial Intelligence*, Chapman and Hall/CRC, 2018, pp. 389–411, ISBN: 9783319944623. DOI: 10.1201/b22400-15. [Online]. Available: https://www.taylorfrancis.com/books/9781351384391/chapters/10.1201/b22400-15.

[43] WildML, "Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs," *Wildml.Com*, pp. 1–7, Sep. 2015. [Online]. Available: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[44] J. Brownlee, *A Gentle Introduction to Computer Vision*, Jul. 2019. [Online]. Available: https://machinelearningmastery.com/what-is-computer-vision/.

[45] S. J. D. Prince, *Computer vision: models, learning and inference*. Feb. 2012. DOI: 10.1016/S0091-6749(04)00596-2. [Online]. Available: http://www.computervisionmodels.com.

[46] D. Huttenlocher, "Computer vision," *Computer Science Handbook, Second Edition*, pp. 43–1, 2004, ISSN: 1340-5551. DOI: 10.4324/9780429042522-10.

[47] *Object detection technology: The next big thing — HPE*, May 2019. [Online]. Available: https://www.hpe.com/us/en/insights/articles/object-detection-technology-the-next-big-thing-1905.html.

[48] Pawangfg, *Object Detection vs Object Recognition vs Image Segmentation*, 2020. [Online]. Available: https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/.

[49] *13.8. Region-based CNNs (R-CNNs) — Dive into Deep Learning 0.15.1 documentation*. [Online]. Available: https://d2l.ai/chapter_computer-vision/rcnn.html.

[50] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, 2015, pp. 1440–1448, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.169. [Online]. Available: https://github.com/rbgirshick/.

[51] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017, ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. [Online]. Available: https://github.com/%20https://papers.nips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[52] D. Robertson, *7 Object Detection with R-CNN, SSD, and YOLO - Deep Learning for Vision Systems MEAP V08 livebook*. [Online]. Available: https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-7/v-7/325%20https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-7/v-7/7.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, University of Washington, vol. 2016-Decem, Washington, May 2016, pp. 779–788, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.91. [Online]. Available: http://pjreddie.com/yolo/.

[54] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 6517–6525, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.690. [Online]. Available: http://pjreddie.com/yolo9000/.

[55] ——, *YOLOv3: An incremental improvement*, Apr. 2018. [Online]. Available: http://arxiv.org/abs/1804.02767.

[56] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*, 2020. [Online]. Available: https://github.com/AlexeyAB/darknet..

[57] A. Kathuria, *What's new in YOLO v3?. A review of the YOLO v3 object... — by Ayoosh Kathuria — Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.

[58] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, Dec. 2016, pp. 21–37, ISBN: 9783319464473. DOI: 10.1007/978-3-319-46448-0{_}2. [Online]. Available: https://github.com/weiliu89/caffe/tree/ssd.

[59] Jonathan Hui, *SSD object detection: Single Shot MultiBox Detector for real-time processing*, 2018. [Online]. Available: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06%20https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.

[60] *SSD - Papers*, 2019. [Online]. Available: https://papers.readthedocs.io/en/latest/imagedetection/ssd/#ssd.

[61] *PyTorch*. [Online]. Available: https://pytorch.org/.

[62] *TensorFlow*, 2020. [Online]. Available: https://www.tensorflow.org/.

[63] Google, *TensorFlow Core — Machine Learning for Beginners and Experts*, 2020. [Online]. Available: https://www.tensorflow.org/overview%20https://www.tensorflow.org/overview/.

[64] F. Chollet and O. &, *Keras: the Python deep learning API*, 2020. [Online]. Available: https://keras.io/.

[65] *Cloud Auto Ml Custom Machine Learning Models — Google Cloud*. [Online]. Available: https://cloud.google.com/automl.

[66] Oktaviani.J, "Azure IoT Edge Documentation," *Sereal Untuk*, vol. 51, no. 1, p. 51, 2018, ISSN: 1098-6596. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-edge/#iotedge-overvie%20https://azure.microsoft.com/de-de/services/iot-edge/%20https://azure.microsoft.com/en-us/services/iot-edge/.

[67] A. Mishra, "Amazon SageMaker," in *Machine Learning in the AWS Cloud*, 2019, pp. 353–385. DOI: 10.1002/9781119556749.ch16. [Online]. Available: https://aws.amazon.com/sagemaker/.

[68] K. W. Breistein, "Applying Machine Learning Using Custom Trained Convolutional Neural Networks on Subsea Object Detection and Classification," Ph.D. dissertation, Norwegian University of Science and Technology, 2019, p. 82.

[69] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017, ISSN: 15577317. DOI: 10.1145/3065386. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386.

[70] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020, ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2844175. [Online]. Available: https://github.com/.

[71] M. Fulton, J. Hong, M. J. Islam, J. Sattar, J. Islam, and J. Sattar, "Robotic detection of marine litter using deep visual detection models," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 5752–5758, Sep. 2019, ISSN: 10504729. DOI: 10.1109/ICRA.2019.8793975. [Online]. Available: https://arxiv.org/pdf/1804.01079.pdf.

[72] K. Schlining, S. von Thun, L. Kuhnz, B. Schlining, L. Lundsten, N. Jacobsen Stout, L. Chaney, and J. Connor, "Debris in the deep: Using a 22-year video annotation database to survey marine litter in Monterey Canyon, central California, USA," *Deep-Sea Research Part I: Oceanographic Research Papers*, vol. 79, pp. 96–105, 2013, ISSN: 09670637. DOI: 10.1016/j.dsr.2013.05.006. [Online]. Available: http://dx.doi.org/10.1016/j.dsr.2013.05.006.

[73] *About — JAMSTEC E-library of Deep-sea Images*. [Online]. Available: https://www.godac.jamstec.go.jp/jedi/e/about_site.html.

[74] X. Wang, J. Ouyang, D. Li, and G. Zhang, "Underwater Object Recognition Based on Deep Encoding-Decoding Network," *Journal of Ocean University of China*, vol. 18, no. 2, pp. 376–382, 2019, ISSN: 19935021. DOI: 10.1007/s11802-019-3858-x.

[75] B. J. Boom, P. X. Huang, J. He, and R. B. Fisher, "Supporting ground-truth annotation of image datasets using clustering," in *Proceedings - International Conference on Pattern Recognition*, 2012, pp. 1542–1545, ISBN: 9784990644109.

[76] X. Sun, J. Shi, L. Liu, J. Dong, C. Plant, X. Wang, and H. Zhou, "Transferring deep knowledge for object recognition in Low-quality underwater videos," *Neurocomputing*, vol. 275, pp. 897–908, Jan. 2018, ISSN: 18728286. DOI: 10.1016/j.neucom.2017.09.044.

[77] X. Yu, X. Xing, H. Zheng, X. Fu, Y. Huang, and X. Ding, "Man-Made Object Recognition from Underwater Optical Images Using Deep Learning and Transfer Learning," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, 2018, pp. 1852–1856, ISBN: 9781538646588. DOI: 10.1109/ICASSP.2018.8461549.

[78] R. M. Nguyen, S. J. Kim, and M. S. Brown, "Illuminant Aware Gamut-Based Color Transfer," *Computer Graphics Forum*, vol. 33, no. 7, pp. 319–328, Oct. 2014, ISSN: 14678659. DOI: 10.1111/cgf.12500. [Online]. Available: http://doi.wiley.com/10.1111/cgf.12500.

[79] Y. Schechner and N. Karpel, "Clear underwater vision," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, IEEE, 2004, pp. 536–543, ISBN: 0-7695-2158-4. DOI: 10.1109/CVPR.2004.1315078. [Online]. Available: http://ieeexplore.ieee.org/document/1315078/.

[80] X. Chen, Y. Lu, Z. Wu, J. Yu, and L. Wen, *Reveal of domain effect: How visual restoration contributes to object detection in aquatic scenes*, Apr. 2020. [Online]. Available: http://en.cnurpc.org/.

[81] D. Gomes, A. F. Saifuddin Saif, and D. Nandi, "Robust underwater object detection with autonomous underwater vehicle: A comprehensive study," in *ACM International Conference Proceeding Series*, New York, NY, USA: ACM, Jan. 2020, pp. 1–10, ISBN: 9781450377782. DOI: 10.1145/3377049.3377052. [Online]. Available: https://dl.acm.org/doi/10.1145/3377049.3377052.

[82] H. Yang, P. Liu, Y. Z. Hu, and J. N. Fu, "Research on underwater object recognition based on YOLOv3," *Microsystem Technologies*, pp. 1–8, Jan. 2020, ISSN: 14321858. DOI: 10.1007/s00542-019-04694-8. [Online]. Available: https://link.springer.com/article/10.1007/s00542-019-04694-8.

[83] *Jetson Modules Specifications Overview*, 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules.

[84] *Products — Coral*. [Online]. Available: https://coral.ai/products/#prototyping-products%20https://coral.ai/products/.

[85] *What's the Difference Between a CPU vs a GPU? — NVIDIA Blog*, Dec. 2009. [Online]. Available: https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/.

[86] aws, *AWS Pricing*, 2020. [Online]. Available: https://aws.amazon.com/pricing/%20https://calculator.aws/#/addService%0Ahttps://docs.aws.amazon.com/pricing-calculator/latest/userguide/aws-pc.pdf%0Ahttps://calculator.aws/#/estimate%0Ahttps://aws.amazon.com/pricing/.

[87] M. Rohith, S. Sai, and S. Veeravalli, "OBJECT DETECTION AND IDENTIFICATION A Project Report," no. November, pp. 1–44, 2019. [Online]. Available: https://www.researchgate.net/publication/337464355_OBJECT_DETECTION_AND_IDENTIFICATION_A_Project_Report.

[88] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, *Object Detection with Deep Learning: A Review*, Apr. 2019. DOI: 10.1109/TNNLS.2018.2876865.

[89] J. Chen-Burger, R. B. Fisher, D. Giordano, L. Hardman, and F.-P. Lin, *Fish4Knowledge Homepage*, 2010. [Online]. Available: https://groups.inf.ed.ac.uk/f4k/%20http://groups.inf.ed.ac.uk/f4k/.

[90] *PeRCeiVe Labs*, 2019. [Online]. Available: http://www.perceivelab.com/datasets%20http://www.perceivelab.com/dataset/Fish%20Detection%20and%20Tracking.

[91] *Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more*. [Online]. Available: https://www.amazon.com/%20http://www.amazon.com/.

[92] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, Springer Verlag, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1{_}48.

[93] COCO Consortium, *COCO - Common Objects in Context*, 2016. [Online]. Available: https://cocodataset.org/#home%20http://mscoco.org/dataset/#detections-leaderboard.

[94] *ImageNet*. [Online]. Available: http://www.image-net.org/about-stats.

[95] *The PASCAL Visual Object Classes Homepage*, 2012. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/.

[96] *Open Images V6 - Description*. [Online]. Available: https://storage.googleapis.com/openimages/web/factsfigures.html.

[97] C. Brown, *Advances in Computer Vision*, K. Arai and S. Kapoor, Eds., ser. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2014, vol. 943, ISBN: 978-3-030-17794-2. DOI: 10.4324/9781315802145. [Online]. Available: http://link.springer.com/10.1007/978-3-030-17795-9.

[98] H. Rezatofighi, N. Tsoi, and J. Gwak, "Generalized Intersection over Union," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 658–666. [Online]. Available: https://giou.stanford.edu/.

[99] R. Padilla, S. L. Netto, and E. A. Da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," *International Conference on Systems, Signals, and Image Processing*, vol. 2020-July, no. July, pp. 237–242, 2020, ISSN: 21578702. DOI: 10.1109/IWSSIP48289.2020.9145130.

[100] Google Developers, *Classification: Precision and Recall — Machine Learning Crash Course*, 2020. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[101] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press, Apr. 2009, ISBN: 9780521865715. [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf%20http://www.informationretrieval.org/%20http://search.ebscohost.com/login.aspx?direct=true&db=cat01619a&AN=up.975861&site=eds-live.

[102] *BlueROV2 Software, Computer, and Joystick Setup Instructions*. [Online]. Available: https://bluerobotics.com/learn/bluerov2-software-setup/.

[103] J. Redmon, *Darknet: Open source neural networks*, 2013. [Online]. Available: https://pjreddie.com/darknet/.

[104] M. Skaldebo, "Visual Aided Deep Learning Applications for Underwater Operations," no. November, 2019.