Escuela
Politécnica
Superior

Small Vessel Detection in Seaborne Environments using Deep Learning Techniques

Bachelor's Degree in Computer Engineering

Bachelor's Thesis

Author:
Pablo Ruiz Ponce
Supervisor:
José García Rodríguez

Universitat d'Alacant
Universidad de Alicante

June 2022

Small Vessel Detection in Seaborne Environments using Deep Learning Techniques

Author
Pablo Ruiz Ponce

Supervisor
José García Rodríguez
Department of Information Technologies and Computing

Bachelor's Degree in Computer Engineering

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, June 2022

Abstract

Each day hundreds of people risk their lives on different seas and oceans all around the globe in order to run away from wars, poverty and other miseries. In this thesis, we propose the use of deep learning based object detectors to autonomously locate small vessels in changing seaborne environments for search and rescue operations using aerial images. After extensive research on actual approaches and available datasets, using the YOLOX architecture we have achieved high accuracy and real-time inference on the SeaDronesSee dataset. In order to face the high imbalance present in the dataset, the variations of the dataset and a weighted loss have been proposed and implemented. Additionally, the proposed method has been tested on unseen images from similar datasets achieving promising results and proving the robustness of the solution.

Resumen

Cada día cientos de personas arriesgan su vida en diferentes mares y océanos de todo el mundo para huir de las guerras, la pobreza y otras miserias. En este proyecto, proponemos el uso de detectores de objetos basados en deep learning para localizar, de forma autónoma, pequeñas embarcaciones en entornos marítimos cambiantes utilizando imágenes aéreas durante misiones de salvamento marítimo. Tras una extensa investigación sobre propuestas anteriores y conjuntos de datos disponibles, hemos logrado una alta precisión e inferencia en tiempo real utilizando la arquitectura YOLOX con el conjunto de datos SeaDronesSee. Para hacer frente al elevado desequilibrio presente en el conjunto de datos, se han propuesto e implementado diferentes variaciones en el conjunto de datos utilizado y el uso de una función de pérdida ponderada. Además, el método propuesto se ha probado en imágenes no vistas de otros conjuntos de datos, logrando resultados prometedores y demostrando la robustez de la propuesta.

Acknowledgements

Firstly, I would like to thank my family, Susana and my friends for being my support and not allowing me to give up during these 4 years. I would not have been able to go through it without them. Additionally, I would also like to thank José García Rodríguez for the guidance and the patience as well as all the 3D Perception Lab team for their feedback and ideas. Finally, I would like to thank all the people who have provided their help to complete this project. Thanks to Albert Mayordomo from Open Arms for sharing his knowledge and his help to start this project. Thanks to Ricardo Ribeiro for allowing me to use the data from the Seagull Dataset. And thanks to Benjamin Kiefer for the help provided using the SeaDronesSee dataset.

Por todos aquellos sueños que se ahogaron en el Mediterráneo

Contents

1. Introduction **1**
- 1.1. Overview . . . 1
- 1.2. Motivation . . . 1
- 1.3. Proposal and Goals . . . 3
- 1.4. Schedule . . . 3
  - 1.4.1. Study Viability of the Thesis: July - September . . . 3
  - 1.4.2. State of the Art and Datasets: September - November . . . 3
  - 1.4.3. First Prototypes: November - January . . . 3
  - 1.4.4. SeaDronesSee: January - March . . . 3
  - 1.4.5. Training and Testing: March - May . . . 4
  - 1.4.6. Overview . . . 4
- 1.5. Outline . . . 5

2. State of the Art **7**
- 2.1. Introduction . . . 7
- 2.2. Convolutional Neuronal Networks . . . 7
- 2.3. Object Detection . . . 9
  - 2.3.1. Two-Stage object detectors . . . 10
  - 2.3.2. One-Stage object detectors . . . 11
- 2.4. Vessel Detection . . . 11
- 2.5. Datasets . . . 12
  - 2.5.1. SAR Images . . . 12
  - 2.5.2. Satellite Images . . . 13
  - 2.5.3. Conventional Cameras . . . 13
  - 2.5.4. Seagull . . . 14
  - 2.5.5. SeaDronesSee . . . 14
  - 2.5.6. Overview . . . 15

3. Materials and Methods **17**
- 3.1. Hardware . . . 17
- 3.2. Docker . . . 18
- 3.3. Deep Learning Frameworks . . . 19
  - 3.3.1. TensorFlow . . . 19
  - 3.3.2. PyTorch . . . 19
  - 3.3.3. Framework decision . . . 20
- 3.4. Google Colaboratory . . . 21

4. Proposal **23**
4.1. Data Imbalance . . . 23
4.1.1. Data Augmentation . . . 24
4.2. Dataset Manipulation . . . 25
4.2.1. Data Cleaning . . . 25
4.2.2. Custom Datasets . . . 26
4.3. YOLOX . . . 27
4.3.1. Weighted Loss . . . 28

5. Experimentation and Results **31**
5.1. Metrics . . . 31
5.1.1. Intersection over Union . . . 31
5.1.2. Precision and Recall . . . 32
5.1.3. Average Precision . . . 33
5.2. Training . . . 34
5.3. SeaDronesSee . . . 35
5.4. Experiments on other datasets . . . 36

6. Conclusions **39**
6.1. Conclusions . . . 39
6.2. Future Work . . . 39
6.3. Publications . . . 40

Bibliography **41**

List of Acronyms **45**

A. Appendix I: Results on SeaDronesSee **47**

B. Appendix II: Results on other datasets **51**

List of Figures

1.1. Dead and missing migrants in the Mediterranean sea since 2014 2
1.2. Schedule's overview 4

2.1. Example of a CNN architecture(Lecun et al., 1998) 7
2.2. Kernel application on a CNN(Gugger & Howard, 2020) 8
2.3. Feature extraction visualization thought the layers of a CNN (Zeiler & Fergus, 2013) 9
2.4. Functioning of a two-stage object detector(Ren et al., 2015) 10
2.5. Functioning of a one-stage object detector(Redmon et al., 2015) 11
2.6. Example images extracted from the LS-SSDD-v1.0 dataset. 12
2.7. Example images extracted from the Airbus Ship Detection dataset. 13
2.8. Example images extracted from the SeaShips dataset. 13
2.9. Example images extracted from the Seagull dataset. 14
2.10. Example images extracted from the SeaDronesSee dataset. 14

3.1. Comparison between Docker and Virtual Machine functioning. 18
3.2. TensorFlow vs PyTorch statistics. 20

4.1. SeaDronesSee classes. 23
4.2. Class distribution on the SeaDronesSee training set. 24
4.3. Data augmentation examples on images(Ahmad et al., 2017) 25
4.5. Instances distribution on the SeaDronesSee versions. 27
4.6. YOLOX head architecture(Ge et al., 2021) 28

5.1. Graphical explanation of the Intersection over Union metric. 32
5.2. Graphical explanation of the Precision and Recall metrics. 33
5.3. Example of a Precision-Recall curve used to calculate the Average Precision. . 34
5.4. Inference on the SeaDronesSee validation set using the SeaDronesSee5 model 36
5.5. Predicted labels on the Seagull and Airbus Ship Detection datasets using the SeaDronesSeeBoat model 37

A.1. Model predictions on SeaDronesSee image. 47
A.2. Model predictions on SeaDronesSee images. 48
A.3. Model predictions on SeaDronesSee images. 49

B.1. Model predictions on Seagull image. 51
B.2. Model prediction on very distant vessel from Seagull images. 52
B.3. Model predictions on Airbus Ship Detection images. 53

List of Tables

2.1. Overview of the most important datasets for the vessel detection task. 15

3.1. Asimov's hardware specifications. 17

4.1. Number of instances of each class on the different versions of the original SeaDronesSee dataset. 26
4.2. Calculated weights for each class using the training set for the class loss. . . . 29

5.1. Most important YOLOX parameters and used values. 35
5.2. Model Average Precision for each class using the validation set 36
5.3. Model mAP at different IoU values and ranges using the validation set 36

1. Introduction

This first chapter introduces the Thesis. It is divided into 5 sections as follows. First, Section 1.1 synthesizes the work carried out during this Thesis. In Section 1.2 the motivation of this work is explained, followed by Section 1.3 where the proposal and the goals of this project are detailed. Thesis' timeline is presented in Section 1.4 and finally,in Section 1.5 details the structure of the Thesis are given.

1.1. Overview

This Bachelor's Thesis presents the research and experimentation carried out for the task of detecting small vessels and shipwrecked people in seaborne environments. We aim to precisely provide a real-time detection on changing conditions with a significant scale variance on the detections. In order to archive this goal, different deep learning techniques on diverse datasets have been tested. Finally, conclusions are extracted, and the future lines of work introduced.

1.2. Motivation

Each day hundreds of people risk their lives on different seas and oceans all around the globe in order to run away from wars, poverty and other miseries. Since 2014 has been recorded more than 24000 missing migrants in the Mediterranean Sea trying to reach Europe (Figure 1.1). Most vessels used are not robust enough to resist the whole crossing and, for this reason, it is crucial to locate those small vessels as fast as possible in order to provide humanitarian aid.

This thesis was born as the result of various personal projects carried out during the previous years. In the summer of 2019, I worked as a volunteer at Open Arms. Open Arms is an Non-Government Organization (NGO) that primarily provides humanitarian aid on the Mediterranean Sea to refugees who try to arrive in Europe using different kinds of vessels. The same summer I took an introductory course on deep learning provided by the university and because of that, I became more and more interested in the field of Computer Vision and Machine Learning (ML). During the followings years, I continued acquiring more understanding in this field thanks to different subjects at the university and several personal projects.

Thinking about the goals of this thesis, an interview with one member of the Open Arms crew has been made in order to understand the current approaches to locating and providing humanitarian aid. While most vessels have an Automatic Identification System (AIS) to provide real-time information about their location, this is very unusual in migrant vessels. For this reason, the common way to detect these vessels is using aerial vehicles. This task is usually made by governments and NGOs such as SeaWatch. Nevertheless, the vast extensions of the seas, make impossible the task of detecting all those vessels manually. Due to this

situation, the idea of developing a model able to automatically detect vessels using aerial images and deep learning was born.

We can conclude that as a result of my relationship with Open Arms, my knowledge in the Computer Vision and Machine Learning fields and the collaboration with the Department of Information Technologies and Computing (DTIC), this project was conceived.

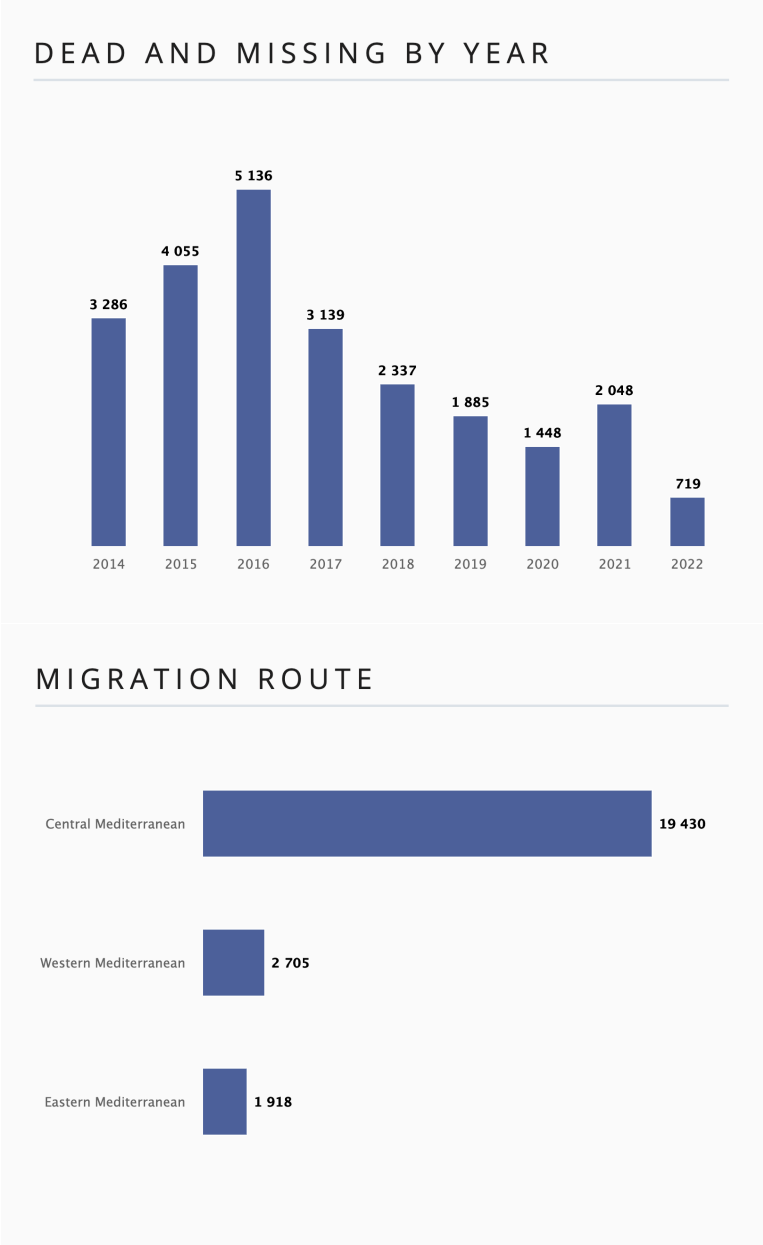


Figure 1.1: Dead and missing migrants in the Mediterranean sea since 2014[1]

[1]Source:`https://missingmigrants.iom.int/region/mediterranean`

1.3. Proposal and Goals

The proposal made in this thesis is the use of an anchor-free one-stage object detector to properly detect small vessels in changing seaborne environments using aerial imagery. In order to achieve this goal, the YOLOX architecture trained on the SeaDronesSee dataset has been proposed. Due to the noticeable imbalance found in the dataset, different approaches have been proposed to correct this problem. Among the select solutions can be found: the use of a weighted loss on the head of the model and the creation of new versions of the selected dataset. Using the previously mentioned approaches, the goal of accurately locating in real-time shipwrecked people and small boats has successfully been achieved.

1.4. Schedule

The development time of this project has been 10 months (1st July, 2021 - 31st May, 2022). All the work done during this period can be divided into 5 main phases. In each phase different task has been developed in order to achieve the goals proposed for this thesis. Once the viability of this project has been validated in the first period, this document has been written alongside the rest of the phases.

1.4.1. Study Viability of the Thesis: July - September

During this period the viability of this thesis has been studied. Firstly, an interview with a member of the NGO Open Arms has been made. During the interview, expert knowledge related to the task to perform was acquired. In addition, detailed information about the procedures and the pipeline used in real search and rescue missions in the Mediterranean sea was also acquired. Finally, the first glance at related previous work and the available datasets to determine the viability was made.

1.4.2. State of the Art and Datasets: September - November

During this period different previous approaches and datasets have been found. The main papers related to the topic were read and summarised as part of the Chapter 2. Furthermore, several datasets related to the task were found and analysed for further use. As most of those datasets are private, access requests to the creators were also submitted during this period.

1.4.3. First Prototypes: November - January

During this period the first prototypes were created. Using the available datasets at that moment, different object detector architectures were tested. The archived results were not relevant, but they were used as feedback to improve the datasets and methods.

1.4.4. SeaDronesSee: January - March

During this period the SeaDronesSee dataset was released and different tests and approaches were made in other to use it for our task. By the end of December, the dataset was released, but in order to properly use it, different modifications were made. Additionally, the initial

attempts to train this model using the YOLOX architecture were made during this period. More information related to the modifications on the dataset and the first attempts can be found in Chapters 4 and 5.

1.4.5. Training and Testing: March - May

During this period the proposed model has been trained and tested using different datasets and parameters. The proposed model has been trained using the different versions created of the SeaDronesSee using different hyper-parameters. Finally, using the datasets collected during the second period, the trained model has been tested in different environments, proving the robustness achieved.

1.4.6. Overview

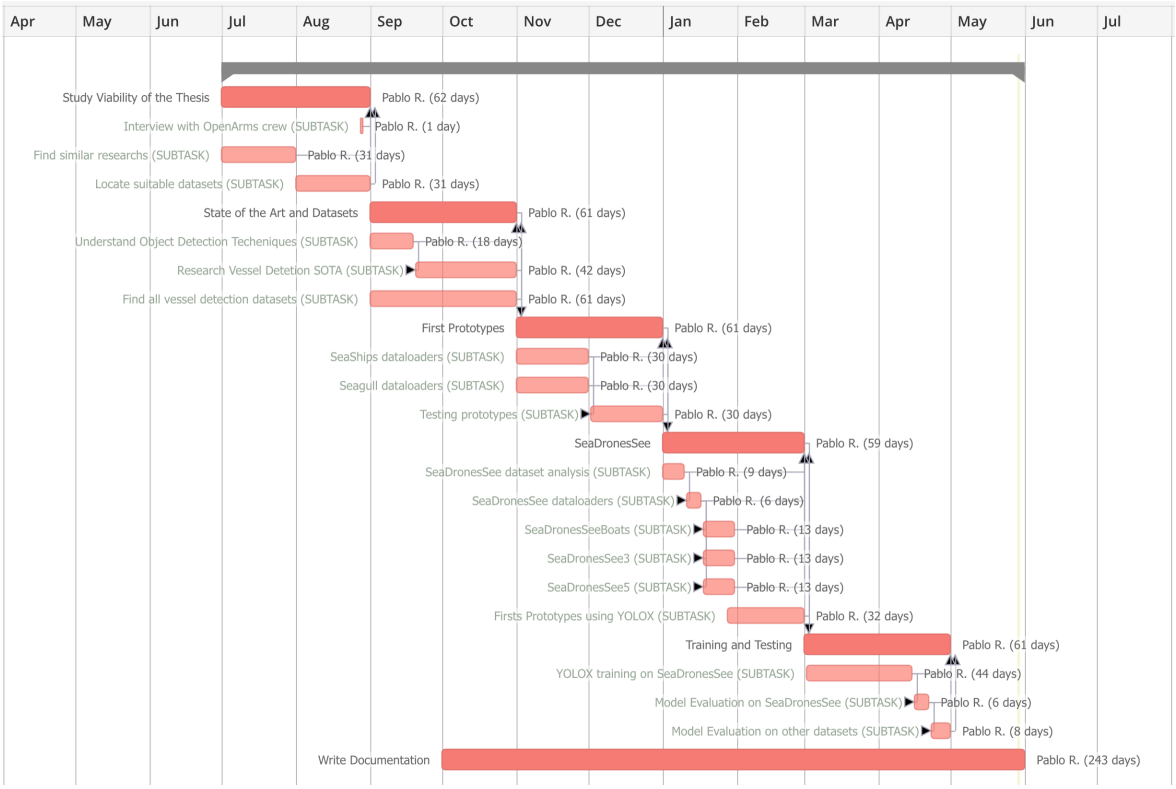


Figure 1.2: Schedule's overview

1.5. Outline

This document is structured as follows: Chapter 1 introduced the project, its motivations, and the main goals to accomplish as well as its distribution in the calendar. Then, in Chapter 2, we explain some previous concepts to understand the vessel detection task, and then we describe the state of the art in vessel detection and the available datasets. Chapter 3 describes the hardware and software tools that we will use during this thesis. In Chapter 4, we analyze and propose solutions to the class imbalance present on the dataset alongside to the selected architecture for the object detector. In Chapter 5, we describe the experiments and results obtained from the proposed methods. Finally, in Chapter 6, the main conclusions of the Thesis are laid down as well as future lines of research.

2. State of the Art

This chapter analyzes the current State of the Art techniques and datasets related to the thesis. It is divided into 5 sections as follows. First, Section 2.1 introduces the difficulties of the task to perform. In Section 2.2 a brief explanations about Convolutional Neuronal Networks is given. The different approximations to the Object Detection problem are presented in Section 2.3. Current approaches to Vessel Detection are mentioned in Section 2.4. Finally,in Section 2.5 the different datasets related to the topic of the thesis are analyzed.

2.1. Introduction

Object detection is a very well-known problem. There are several optimal solutions and architectures able to solve this task. Even for detecting vessels, multiple methods and datasets have been successfully tested. However, detecting small vessels in real-time for search and rescue operations entails certain difficulties.

2.2. Convolutional Neuronal Networks

Nowadays, almost all image related tasks in the ML field are solved using Convolutional Neural Network (CNN). They are the base for the most important classification or regression models using images. The element that gives the name to this class of networks is the Convolutional Layers. Unlike the linear layers present in the traditional Multi Layer Perceptrons, the input of this layer is a matrix instead of a vector. Using matrices as input helps keeping geometrical information present in the image.

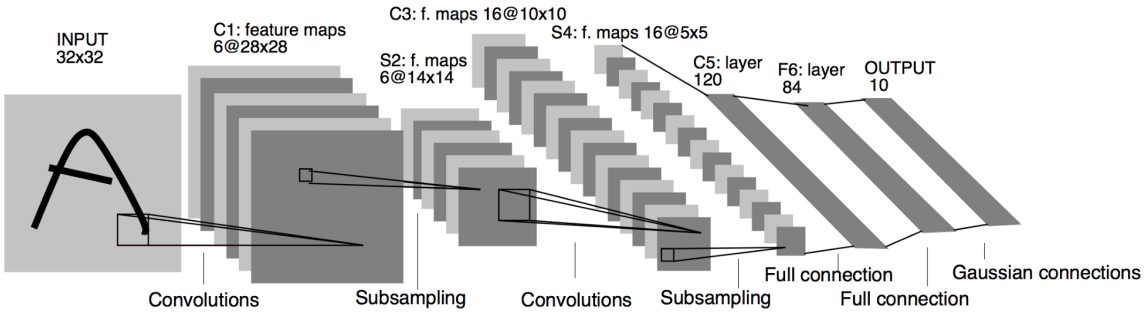


Figure 2.1: Example of a CNN architecture(Lecun et al., 1998)

Once the convolutional layer receives an input matrix, it applies a certain number of kernels to extract some features. A kernel is a square matrix with certain values. The output of this layer is the result of applying the convolution operation of the kernels alongside all the input image. This process is visually described in Figure 2.2. The values contained in the kernels are the trainable parameter that will be modified during training.

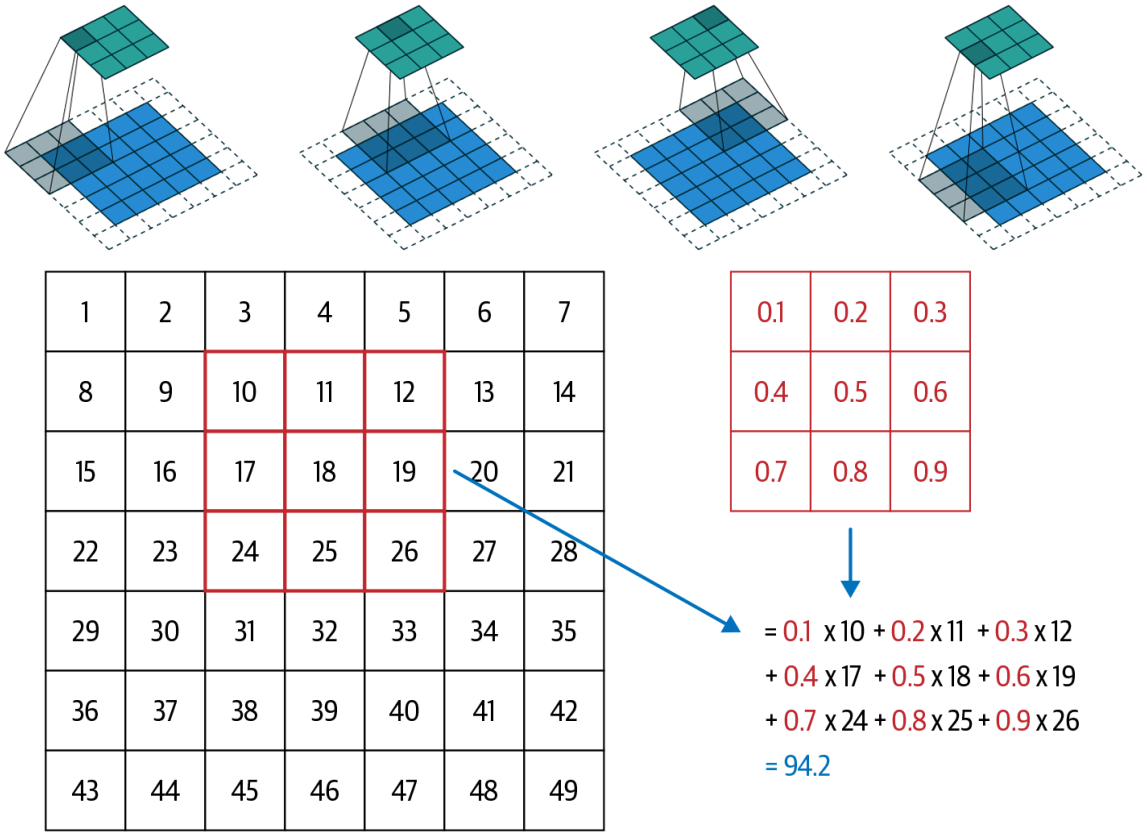


Figure 2.2: Kernel application on a CNN(Gugger & Howard, 2020)

CNN usually follows a certain structure to obtain as much information as possible from the images. After a convolutional layer, there is usually a downsampling layer and then another convolutional layer. They are different approaches to performing the downsampling but always the objective is to reduce the size of the image. Reducing the image size means that the kernels on the following convolutional layers will represent a bigger portion of the image and thus a more high-level feature. At the beginning of the network, the features extracted from the images will be low-level features such as corners or edges. After performing downsampling, the next convolutional layer will detect more high-level features such as patterns or shapes. And finally, it would be able to properly extract the important features of the image. All this process can be appreciated it in Figure 2.3.

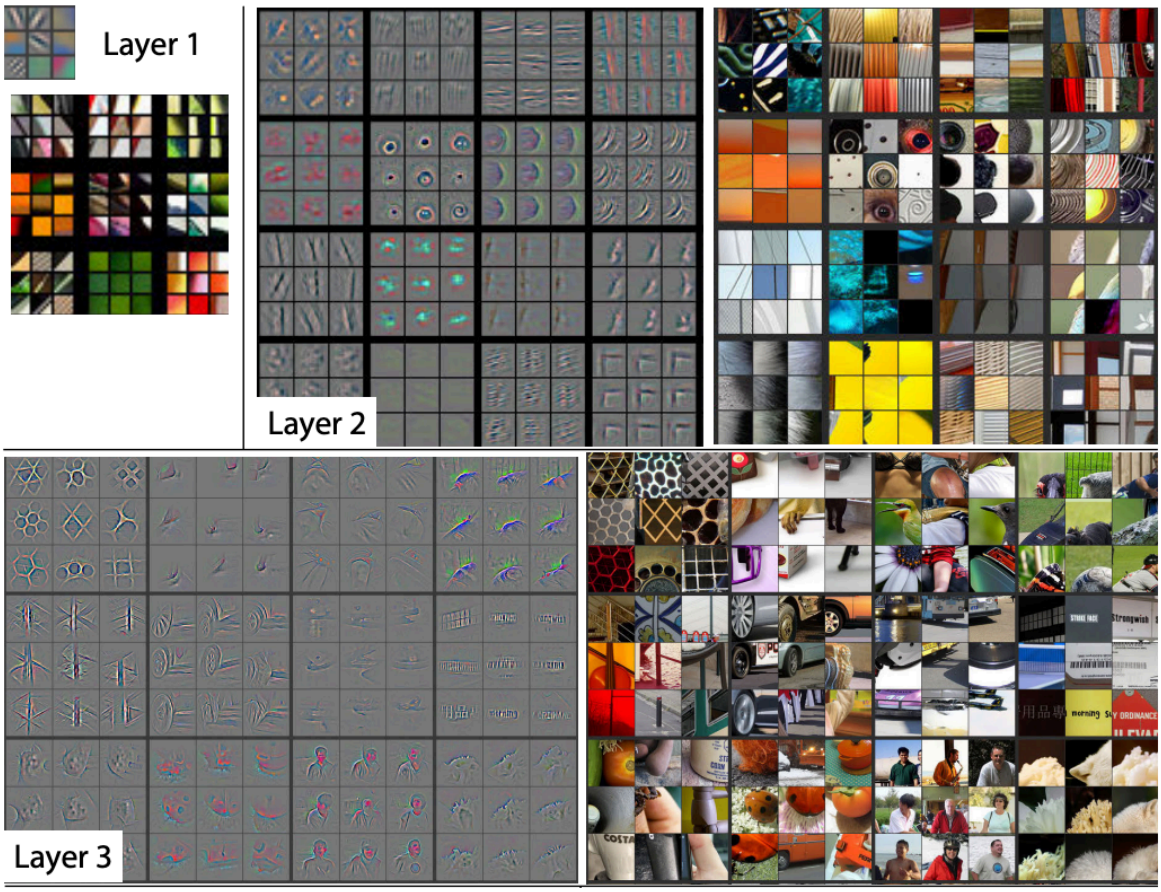


Figure 2.3: Feature extraction visualization thought the layers of a CNN (Zeiler & Fergus, 2013)

Finally, after forwarding the input images through several convolutional and downsampling layers, all the features extracted are encoded into a vector. This vector is usually used as the input of other neuronal networks. In deep learning architectures involving images, it is very common to use pre-trained backbones in order to extract features from images and then use those encoded features for other tasks such as classification or regression. A pre-trained backbone is just a CNN trained with large datasets such as Imagenet to learn how to extract the important features present on an image.

2.3. Object Detection

We understand object detection as the task of properly locating a specific object in a specific place. In the computer vision field, this task usually consists in providing the coordinates and the labels of specific objects in images. During the past decades, several approaches have been proposed to solve this task. Nowadays, with the progress archived in the ML field, this problem can be considered solved using CNNs and different deep learning architectures. Nevertheless, each year new architectures improve the actual results with faster inference times and accuracy. The actual most relevant object detectors can be classified into two

main categories: two-stage detectors and one-stage detectors.

2.3.1. Two-Stage object detectors

This approach to the object detection task was the inial one using ML. This method has two different stages, as can be inferred by its name, to solve this problem. During the first stage, the objective is to acquire all the features from the image and obtain the region proposals. Once the region proposals have been acquired, the second stage consists of the classification of these regions. After the two stages, the output of the detector is the regions with their associated labels.

The two-stage approach achieves very robust and accurate predictions. As a downside, the inference time on this kind of detector does not allow to perform real-time predictions on high frame rates. Some of the most famous architectures that implements this approach are R-CNN(Girshick et al., 2013), SPP-Net(He et al., 2014) and Faster R-CNN(Ren et al., 2015).

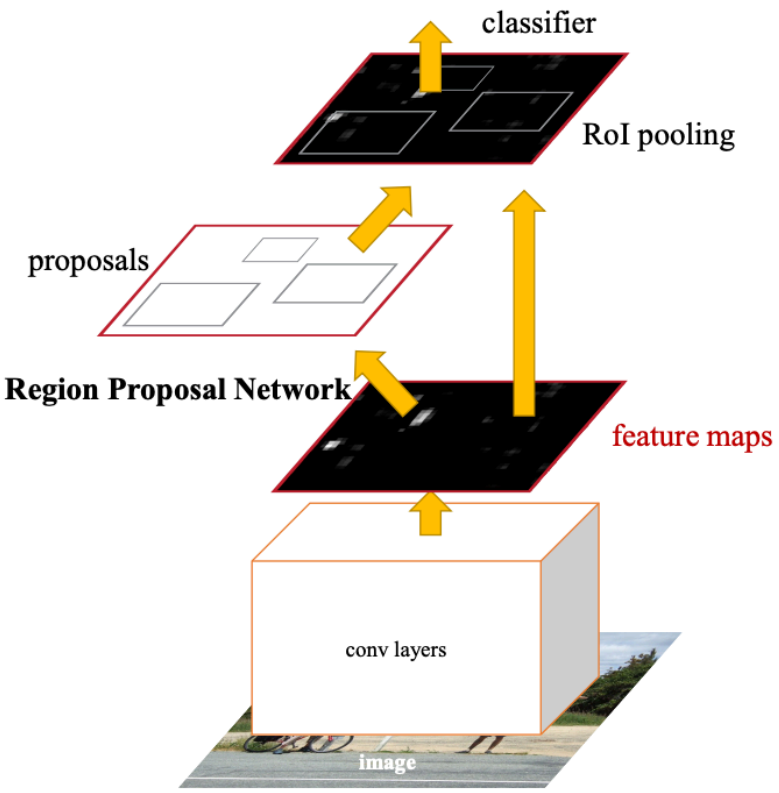


Figure 2.4: Functioning of a two-stage object detector(Ren et al., 2015)

2.3.2. One-Stage object detectors

On the other hand, one-stage detectors generate all the predictions using a single network. Among the most famous architectures of one-stage detectors, we can find the YOLO family(Redmon et al., 2015) and RetinaNet(Lin et al., 2017). The way in which the original YOLO version work is by dividing the image into grids. Then, the probability of the object on a cell of the grid is calculated and, finally, the architecture determines the bounding boxes of the objects with more probability.

Thanks to the one-stage approach, those detectors are able to make a real-time inference with much higher frame rates than two-stage detectors. As a downside, those original proposals found difficulties detecting objects with a noticeable scale variance, groups of small objects of the same class and a reduction of the accuracy compared to two-stage detectors. However, some of these difficulties have been solved with new architectures as we will see in future chapters.

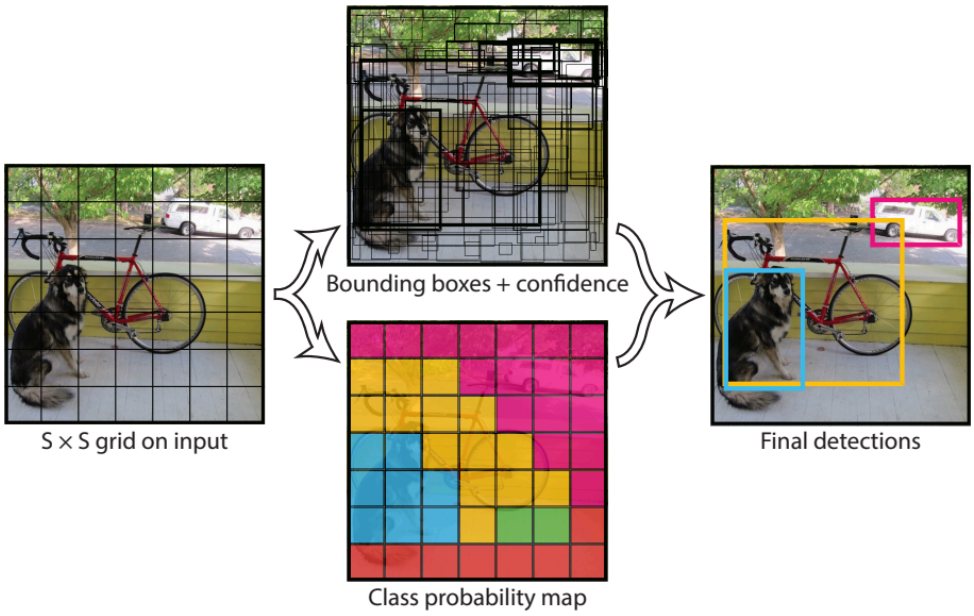


Figure 2.5: Functioning of a one-stage object detector(Redmon et al., 2015)

2.4. Vessel Detection

Detecting ships on images is being used in many applications nowadays. Some of the most common ones are: borders and harbours surveillance (Cane & Ferryman, 2018), illegal fishing detection, autonomous sailing (Chen et al., 2020) and maritime trade routes monitoring. For this task, the current state-of-the-art approaches are based on CNN object detectors. While there are some proposals that try different approximations like time-series and attention (Li

et al., 2021), the most common solutions are based on two-stage object detectors like R-CNN and Faster R-CNN and one-stage object detectors like the YOLO family. In order to deal with small size vessels, there are some proposals involving, time-series (Porto Marques et al., 2020), attention mechanisms (Dubey et al., 2021), backbone modifications (Li et al., 2021) and anchor-free detectors.

2.5. Datasets

Although the architectures proposed are very standard, the difficulty of the task is to acquire data to train those models. While it is relatively easy to find images of any vehicle, in the case of vessels the data acquisition is much more challenging. As can be seen in the following sections, the methods to obtain ship images are very expensive and, in some cases, only available for certain collectives as governments. Due to the cost of acquiring these images, most datasets are private and not valid for search and rescue duties. The majority of the available datasets are only valid for commercial uses, as they only contain big commercial vessels that can be located much more easily and from much further away. Search and rescue people and vessels that are in danger in the middle of the seas do not generate any income.

2.5.1. SAR Images

Synthetic Aperture Radar (SAR) use the echoes of electromagnetic waves to generate high-quality images of surfaces. The images acquired using this method are not affected by changing lighting or weather conditions. For this reason, these kinds of images are the most common ones for vessel detection. Among the most famous datasets of this category, we can find SSDD(Zhang et al., 2021), LS-SSDD-v1.0(Zhang et al., 2020), SAR-Ship-Dataset(Wang et al., 2019) and xView3(*xView3: Dark Vessels*, n.d.). However, SAR images are not suited for the task being solved in this thesis. The elevated cost of this device and the difficulties to obtain real-time data make its use unfeasible in real operations.

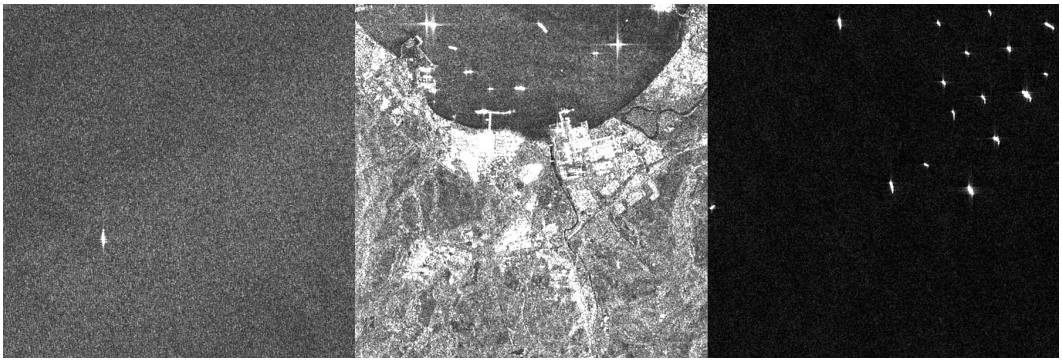


Figure 2.6: Example images extracted from the LS-SSDD-v1.0 dataset.

2.5.2. Satellite Images

These kinds of images, as their name indicates, are taken using satellites. Satellite images can be found in services like Google Maps. Although there are multiple satellites to obtain images, the number of datasets using these images is very limited. For the specific task of vessel detection, the most famous one is the one provided by Airbus for a Kaggle challenge(Airbus, n.d.). While these images could be used to train a good model, it is not practical because in real scenarios is almost impossible to obtain real-time images from those satellites.

Figure 2.7: Example images extracted from the Airbus Ship Detection dataset.

2.5.3. Conventional Cameras

This kind of images are the ones that can be acquired using conventional cameras; like the ones present in our smartphones or surveillance cameras. Those images have worse resolution than the images obtained with the previous methods, but the required device to obtain the images is much cheaper. Among the datasets of this category, we can find SeaShips(Shao et al., 2018), Boat Re-ID(Spagnolo et al., 2019), Seagull(Ribeiro et al., 2019) and SeaDronesSee(Varga et al., 2021). SeaShip is a dataset composed of thousands of images of vessels captured by static surveillance cameras in different locations. Similarly, Boat Re-ID is a dataset with thousands of images obtained by a static camera located on the coast. Neither of these datasets fits the requirements of the problem to solve.

Figure 2.8: Example images extracted from the SeaShips dataset.

2.5.4. Seagull

The first dataset that fitted our proposal was the Seagull dataset. It contains 366698 images extracted from 6 different videos captured by a Unmanned Aerial Vehicle (UAV). It contains 6489 instances of boats in a maritime environment. It was a very good starting point. Nevertheless, the main problem was that all the vessels' instances and environments were very similar and the resolution of the images is quite low (256x256 pixels).

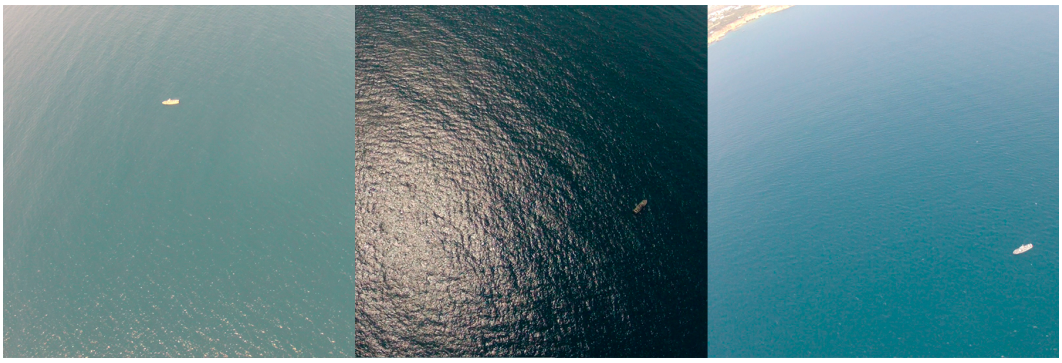


Figure 2.9: Example images extracted from the Seagull dataset.

2.5.5. SeaDronesSee

Fortunately, by the end of 2021, the SeaDronesSee dataset was released. This dataset contains 54000 images with 400000 instances with a very high resolution (3840 x 2160 to 5456 x 3632 pixels) using a UAV. The different instances were captured at different altitudes, environments and viewing angles. Apart from the object detection, this dataset allows the task of object tracking and multi-object tracking. In addition to the vessels, they are also floaters and swimmer instances on the dataset. All these features make this dataset the most suitable option.

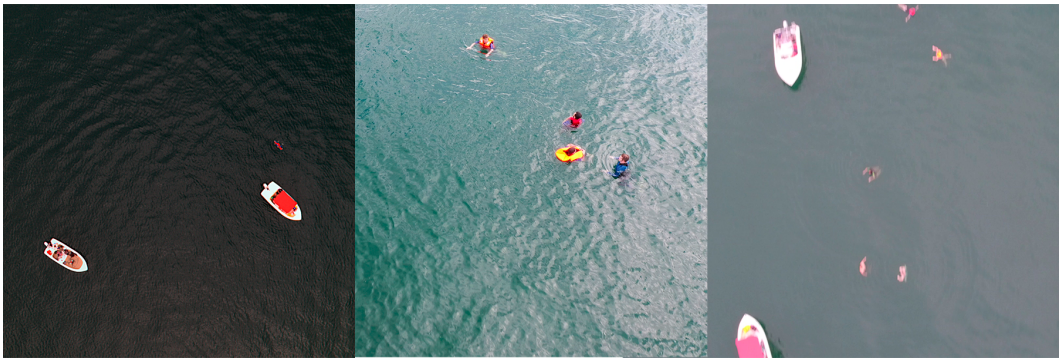


Figure 2.10: Example images extracted from the SeaDronesSee dataset.

2.5.6. Overview

In Table 2.1 we summarize the available datasets revised in this section.

Table 2.1: Overview of the most important datasets for the vessel detection task.

Dataset	Type of Images	Number of Images
SSDD	SAR	1160
LS-SSDD-v1.0	SAR	9000
SAR-Ship-Dataset	SAR	210
xView3	SAR	754
Airbus Ship Detection	Satellite	208000
SeaShips	Convetional	31455
Boat Re-ID	Conventional	5523
Seagull	Conventional	36698
SeaDronesSee	Conventional	54000

3. Materials and Methods

This chapter analyzes the main hardware and software used in this thesis. It is divided into 4 sections as follows. First, Section 3.1 introduces the hardware used to train the models. In Section 3.2 a brief explanations about the functioning of Docker and its capabilities is made. The different deep learning frameworks are compared in Section 3.3 and one is selected for being used. Finally,in Section 3.4 the cloud service Colaboratory is presented.

3.1. Hardware

Table 3.1: Asimov's hardware specifications.

	Asimov
Motherboard	ASUS X99-A Intel X99 chipset
CPU	Intel(R) Core(TM) i7-5820K 3.30GHz 6 cores 12 threads
GPU 1	NVIDIA GeForce GT730 96 CUDA cores 1024 MiB of DDR3 Video Memory
GPU 2	NVIDIA GeForce Titan X 3072 CUDA cores 12 GiB of GDDR5 Video Memory
GPU 3	NVIDIA Tesla K40c 2880 CUDA cores 12 GiB of GDDR5 Video Memory
RAM	4 × 8 GiB Kingston Hyper X DDR4 2666 MHz CL13
Storage (data)	(RAID1) Seagate Barracuda 7200rpm 3TiB SATA III HDD
Storage (OS)	Samsung 850 EVO 500GiB SATA III SSD

Training deep learning models is a quite computational and expensive task. Modern architectures have millions of trainable parameters that are constantly being updated and used in mathematical operations. Thankfully, the training time can be drastically reduced as all those operations are easily parallelizable. All modern deep learning frameworks provide implemen-

tations to parallelize most of the workload using the Compute Unified Device Architecture (CUDA) cores in the NVIDIA Graphics Processing Unit (GPU)s.

The computation server *Asimov* from the DTIC has been used for training the different models used in this thesis. With 3 NVIDIA GPUs, it provides a high-efficiency environment for all kinds of computation tasks. Thanks to the hardware provided by Asimov it has been possible to train the object detection architecture proposed in a reasonable time. Allowing to perform multiple pieces of training in order to improve the results. All the technical specifications of Asimov can be found in Table 3.1. Asimov runs Ubuntu 16.04 as Operating System and the connections to it are made via Secure Shell (SSH).

3.2. Docker

The use of the computing server *Asimov* is shared among different students and professors of the department for different projects. Any software project uses different libraries and dependencies for its proper functioning. When a server is exclusively dedicated to a project, all the dependencies and requirements can directly be installed on the computer. However, when multiple people are sharing the same resources, it is needed a way to isolate the different environments used for each specific project.

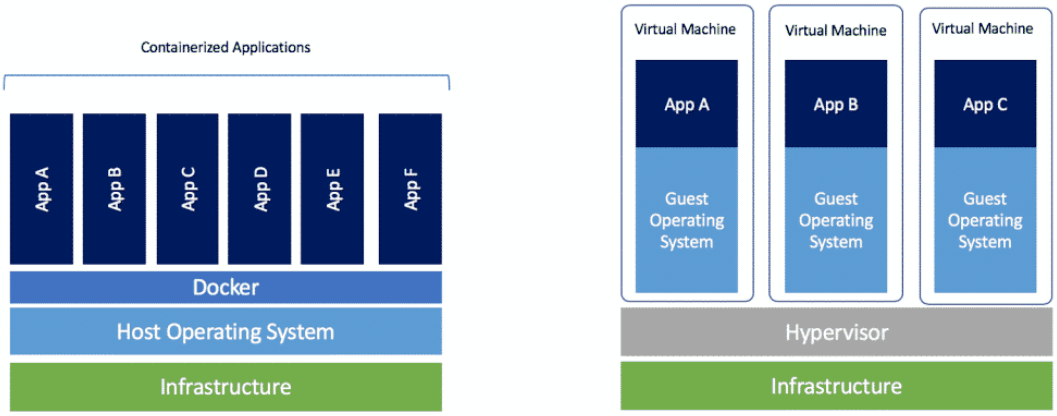


Figure 3.1: Comparison between Docker and Virtual Machine functioning[1].

Docker has been the chosen solution to virtualize and isolate the different environments. While virtual machines virtualize the whole OS, Docker provides software-level virtualization. This means that Docker uses the kernel of the host machine in order to emulate a new one. This approximation improves performance and reduces the needed storage. In addition, Docker works with the concepts of containers and images. An image can be considered as an environment with all its requirements, and a container is an instance of a given image with a current state. These concepts can be compared to the *class* and instance concepts in the Object Oriented paradigm. A custom Docker image has been made specifically for this thesis.

[1]Source: `https://www.docker.com/blog/containers-replacing-virtual-machines/`

3.3. Deep Learning Frameworks

An important decision that has to be made when developing a deep learning project is the framework that is going to be used. Several frameworks and libraries have emerged during the last decade as a consequence of the growing popularity of this field. Each one with its respective upsides and downsides. Nowadays there are still multiple options, but almost all the time the selection can be narrowed to TensorFlow or PyTorch.

3.3.1. TensorFlow

TensorFlow is Google's ML library. It provides all the features that anyone can expect in a library of this kind. The most important glsml algorithms are implemented as well as a wide range of functionalities to work with different kinds of data. It also provides hardware acceleration using GPUs and tensor processing unit (TPU)s. Additionally, TensorFlow includes Keras, which is another ML library with a higher level of abstraction. Keras is usually used for fast prototyping and as a much more beginner friendly option to start in this field.

Nevertheless, one of the strongest features of TensorFlow is the deployment functionalities. TensorFlow is by far, the most used framework to deploy models in real environments. For model server deployment you can use *TensorFlow Serving*. For using ML models in smartphones and Internet of Things (IoT) devices you can use *TensorFlow Lite*. For native web deployment, you can use TensorFlow.js. And, the most recent addition is the integration of TensorFlow models in Firebase.

Finally, one last aspect that should be considered is the community. TensorFlow has been the most popular Machine Learning framework for years. Thanks to that, there is an active massive community that is constantly creating models and improving the library. Community created models can be directly downloaded from the *TensorFlow Hub*.

3.3.2. PyTorch

PyTorch is Meta's ML library. As well as TensorFlow, it provides all the most relevant ML algorithms. It also provides hardware acceleration using GPUs and TPUs (PyTorch-XLA). In order to work with different types of data, it includes different libraries with specific functionalities. TorchVision for Computer Vision tasks, TorchText for Natural Language Processing tasks and TorchAudio for audio files. Similar to Keras, PyTorch has PyTorch Lighting, which simplifies the model definition and the training process.

Although PyTorch has deployment capabilities, the prominent use of this library is in the research field. During the last years, it has been established as the preferred option for the majority of researchers around the globe. As can be seen in Figure 3.2 the majority of the new papers and architectures are natively implemented using PyTorch. In such a fast-growing field as ML, researchers cannot wait for the TensorFlow implementation of the model that they want to use or improve.

Very similar to TensorFlow, there exists PyTorch Hub. A platform where people can upload their architecture implementations and trained models.

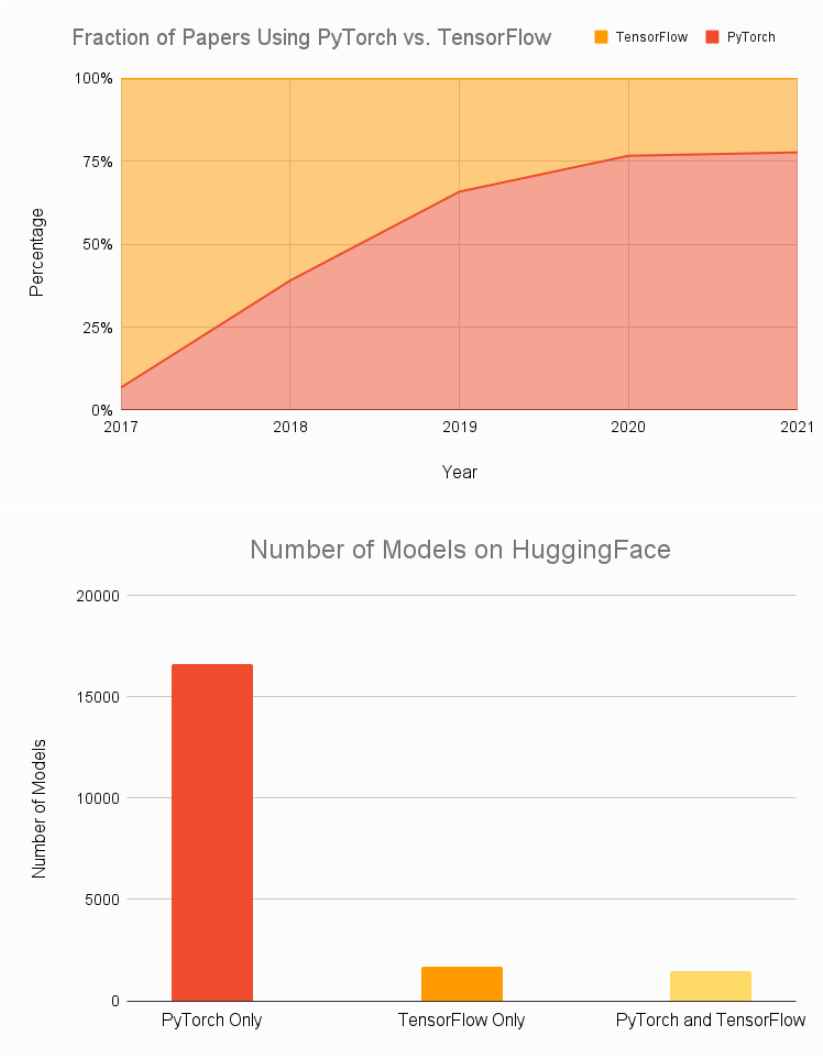


Figure 3.2: TensorFlow vs PyTorch statistics[2].

3.3.3. Framework decision

Once all the factors to choose a framework have been considered, it is time to make a decision. At first, it may seem that the best option might be TensorFlow. If the idea is to deploy the proposed object detector in an aerial vehicle, TensorFlow Lite looks like the most suitable option for this task. However, as explained in the previous section, the cutting-edge architectures are implemented in PyTorch. This is the exact case that we have to face with the architecture that will be proposed in the next chapter. It is not viable to migrate such a complex architecture into TensorFlow Lite. For all these reasons, the selected framework that is going to be used during this thesis is PyTorch.

[2]Source: `https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/`

3.4. Google Colaboratory

Colaboratory is a cloud-based application developed by Google. It allows you to execute Python code like a Jupyter notebook, but on the browser and using Google machines as computational power. It also allows hardware accelerations using GPUs and TPUs and complete integration with your Google Drive storage. It comes with the most used Deep Learning libraries pre-installed; being the perfect solution for rapidly and easily developing prototypes. Nevertheless is not recommended for training big projects due to the time usage limits and the limited hardware resources.

4. Proposal

This chapter analyzes the proposal made to solve the problem presented by this thesis. It is divided into 3 sections as follows. First, Section 4.1 introduces the imbalance problem present in the dataset. In Section 4.2 different modifications to the dataset are presented to equilibrate the number of instances. Finally,in Section 4.3 the proposed architecture and its modifications is analyzed.

4.1. Data Imbalance

After taking into account all the possible datasets, the decision of using SeaDronesSee has been made. Once a dataset is selected, it is important to carefully analyze and understand the data that is on it. SeaDronesSee contains more than 400000 instances of 6 different classes. All the information related to the images, the instances and the metadata is stored in JSON files with the COCO(Lin et al., 2014) dataset structure. The dataset is divided in training, validation and test sets to properly train the models.

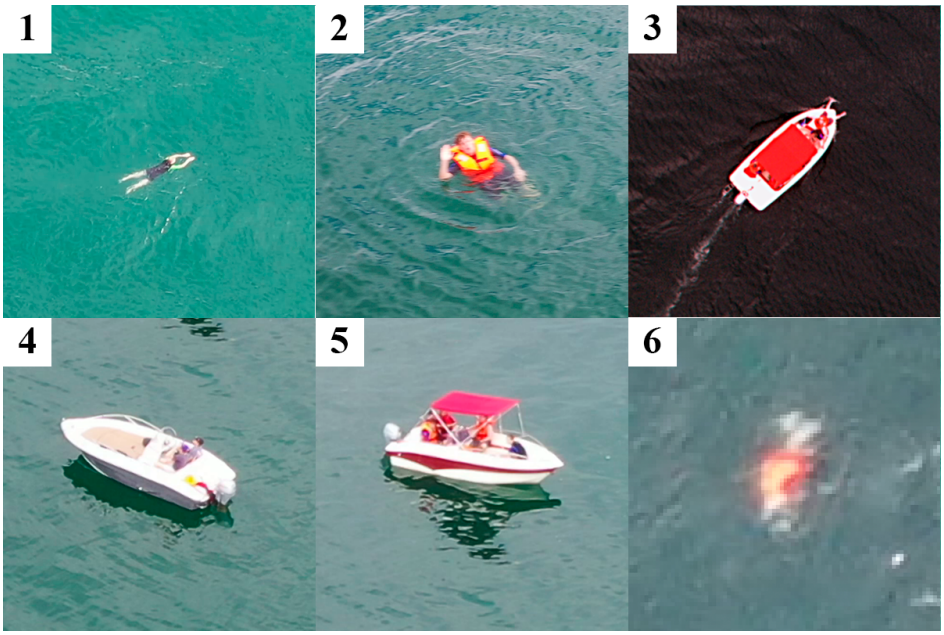


Figure 4.1: SeaDronesSee classes: (1) swimmer, (2) floater, (3) boat, (4) swimmer*[1], (5) floater*, (6) life jacket.

[1]The * represents that this instance is inside a boat

The first noticeable problem when analyzing the dataset is the number of instances per class. Looking at figure 4.2, it is crystal clear that the dataset is tremendously unbalanced. This is a significant problem when training deep learning models. If there is no balance between the number of instances per class, the model is always going to be biased towards the majority classes. This is a very common issue in object detection and classification problems, and there are multiple approaches to mitigate it.

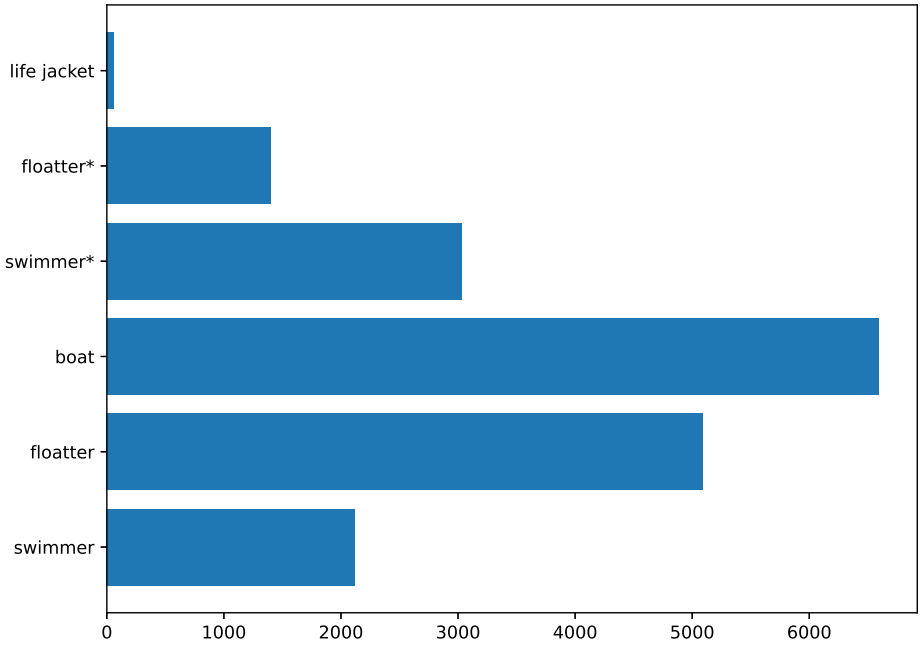


Figure 4.2: Class distribution on the SeaDronesSee training set.

4.1.1. Data Augmentation

If it is possible, the best option is always to populate the dataset with more new instances of the minority classes. The problem is that, in most cases, it is almost impossible to capture new data. This is our particular case. We don't have the means to capture new aerial images of vessels at the sea. When there are no options to acquire new data, the most common way to equilibrate the datasets is to generate new data using the available one. This method is known as data augmentation, and it is very common in tasks that require images, even when there is no imbalance(Perez & Wang, 2017). Applied to images, this method usually consists in performing some modifications to the images like rotations, flips, contrasts and color corrections. There are many transformations that can be applied, and the selection will always depend on the dataset that is being used.

While this technique may seem the solution, it was rapidly discarded due to the images of the dataset. Almost all the images on SeaDronesSee contain multiple objects to detect. The problem is that the instances of the minority classes always appear with instances of the majority classes. If we increase the number of instances of life jackets, the number of

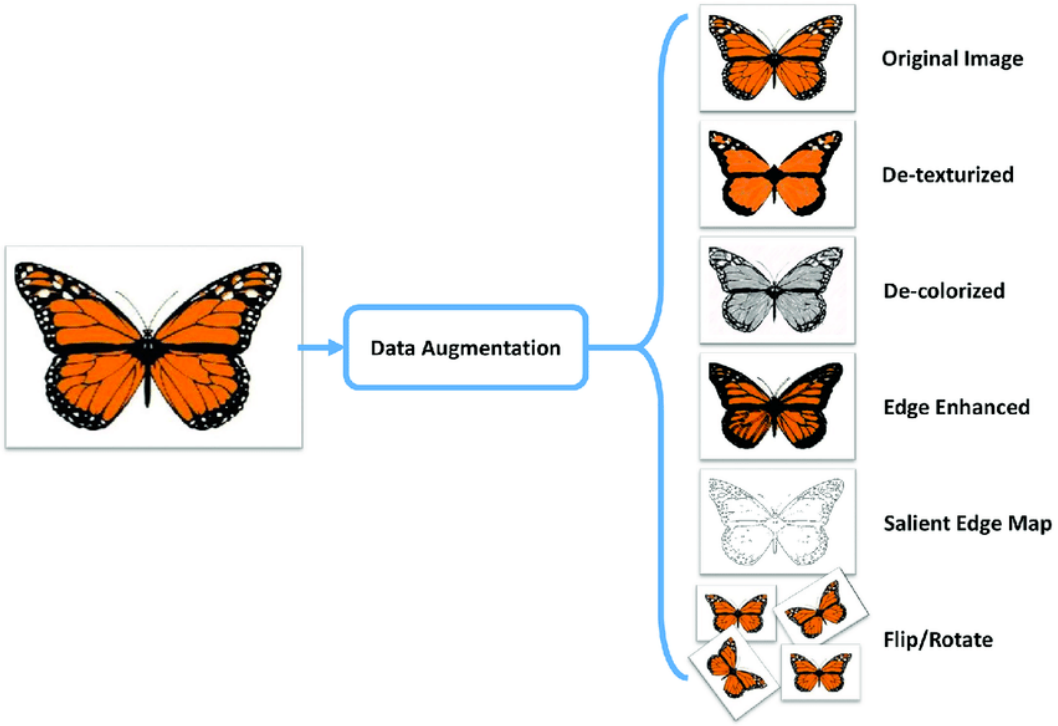


Figure 4.3: Data augmentation examples on images(Ahmad et al., 2017)

instances of boats and floaters will increase too. Nevertheless, the are other techniques that can be applied to solve the imbalance problem. There will be explained in the following sections.

4.2. Dataset Manipulation

Once the problem has been identified, one of the first ideas to solve it was the modification of the dataset structure. In section 4.2.2 we will dive more into the different selected approaches. In addition, other minor changes have been required in order to properly use the SeaDronesSee dataset.

4.2.1. Data Cleaning

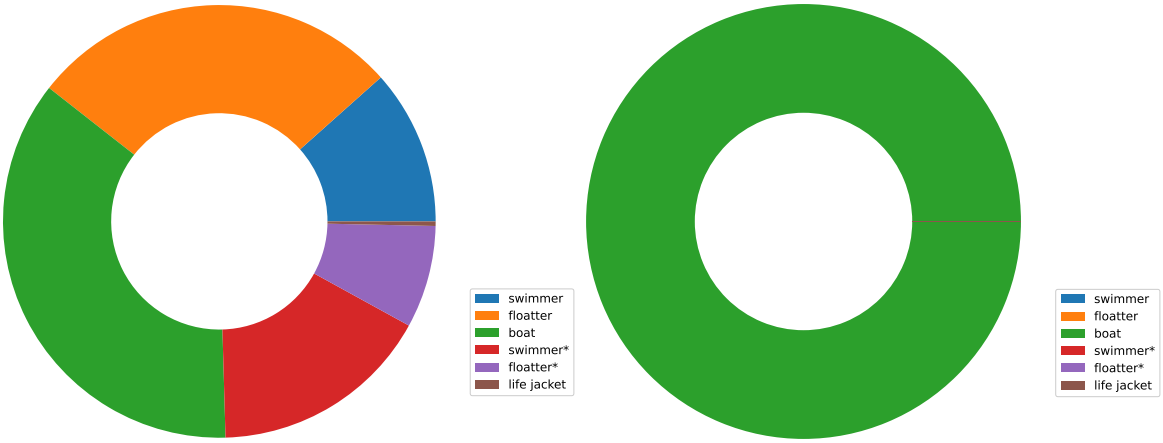
As this dataset is very new, we found some minor issues that needed to be fixed to properly use it. First of all, the *iscrowd* attribute was missing on the annotation files; generating errors on the model data loaders. This attribute determines if the recognized object is a group of objects or not. The other significant error found was the missing images. There were annotations pointing to image ids that didn't exist. In order to solve this problem, the annotations that were causing those problems were deleted.

4.2.2. Custom Datasets

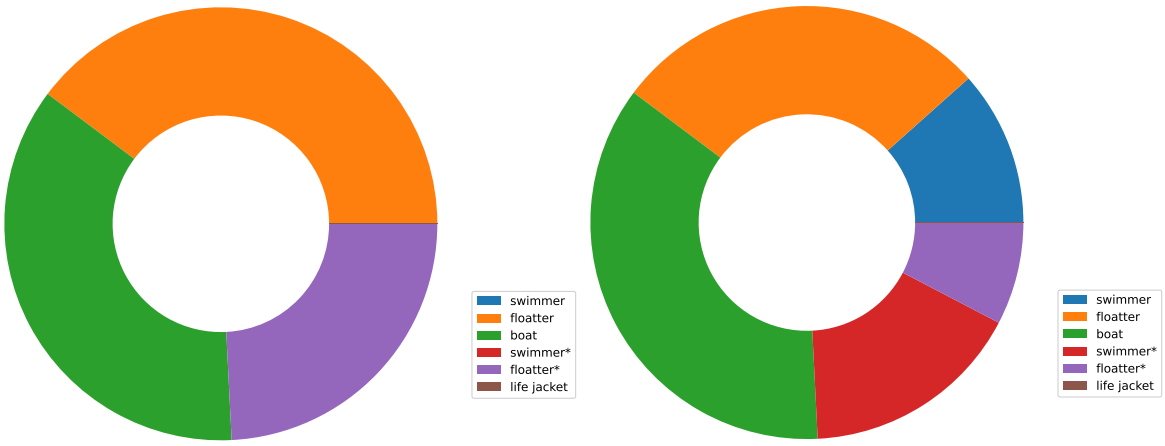
Once the dataset was ready to use, one of the approaches to equilibrate the dataset instances was the creation of different datasets, merging similar classes, in order to create evenly distributed datasets as can be seen in Figure 4.5. As the main objective was to detect vessels, the first version only contained labels of the type *boat*. For the second version, 2 more classes were added. In addition to the boats, the *floater* and the *floater in a boat* classes were included. All the people on the water were included in the *floater* class (including the life jacket), while the instances of people inside the boat were merged into the *floater in a boat* class. Lastly, in order to test our model with the benchmark provided by the creators of the dataset, a third version was created. This version is almost identical to the original dataset. The only difference between the two is that the life jacket class was removed and those labels were included in the floaters class. While this change provides much more balanced data, the accuracy lost due to this change is minimal.

Table 4.1: Number of instances of each class on the different versions of the original SeaDronesSee dataset.

Dataset	swimmer	floater	boat	swimmer*	floater*	life jacket
SeaDronesSee	2120	5092	6593	3027	1401	62
SeaDronesSeeBoats			6593			
SeaDronesSee3		7274	6593		4428	
SeaDronesSee5	2120	5154	6593	3027	1401	



(a) SeaDronesSee **(b)** SeaDronesSeeBoats



(c) DronesSee3 **(d)** SeaDronesSee5

Figure 4.5: Instances distribution on the SeaDronesSee versions.

4.3. YOLOX

Among all the possible architectures to perform object detection, the chosen one was YOLOX Ge et al. (2021). This detector is part of the YOLO family, which provide excellent results and fast inference becoming the most suitable choice. In fact, the State of the Art model for this dataset was YOLOv5 Jocher et al. (2022). Nevertheless, the choice of YOLOX over other models was based on a specific problem inherit form the task to perform. While the rest of the versions are anchor-based detectors, YOLOX is an anchor-free detector. Anchor-based detectors have proven to be an optimal approach for object detection tasks. However, this kind of detector has problems with objects candidates with large scale variations. This scenario is very common in UAV and aerial images Yang et al. (2020), as the camera is not in a fixed position or altitude. Taking into account that the dataset and objective of this project is to provide robust real-time inference on aerial images, using an anchor-free detector seemed the best choice.

The anchor-free detection is not the unique change in this architecture in comparison to the previous versions. As the YOLOv4(Bochkovskiy et al., 2020) and YOLOv5 architectures are very optimized for the anchor-based detection, YOLOX takes YOLOv3(Redmon & Farhadi, 2018) as starting point. The other major change used in this architecture is the decoupled head. In all the previous versions of YOLO, the head of the architecture is coupled. This means that the same part of the head was in charge of the classification and the location task. This can produce a reduction in the model's performance. In order to address this problem, the head of YOLOX is decoupled, using one part for the classification problem and the other for the regression task. This is not exclusive, as other object detectors already use this approach, but it was the first time in the YOLO family.

Like the majority of deep learning architectures, YOLOX has different versions of different sizes. The larger the model, the better results; but also higher inference times. For this particular task, the most suitable option is a balance between accuracy and inference time. We need to provide real-time inference using limited resources.

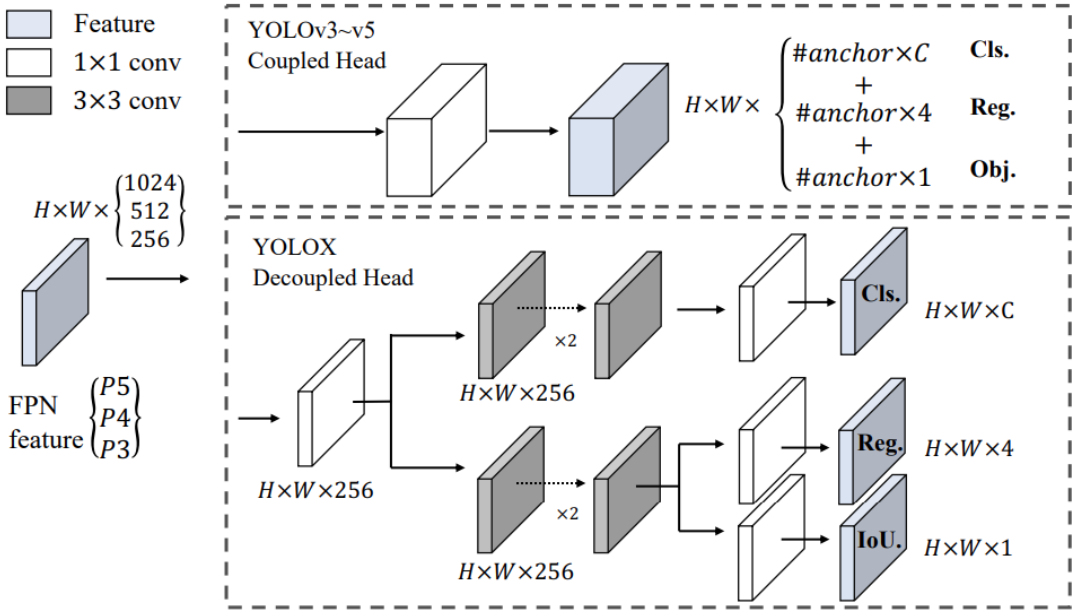


Figure 4.6: YOLOX head architecture(Ge et al., 2021)

4.3.1. Weighted Loss

Once the architecture for the object detectors has been chosen, additional modifications have been made to provide further help in the class imbalance problem. Aiming to mitigate this problem, the class loss function on the head of the architecture has been modified. A ponderated loss has been used depending on the class predicted. This means that, depending on the expected class for a certain region, the loss is going to be multiplied by a certain factor. There are different ways to calculate this factor, but always is going to be greater when a minority class is expected. The original loss function used in the model is a BCEWithLogitsLoss.

$$\ell_c(x, y) = L_c = \{l_{1,c}, \dots, l_{N,c}\}^\top \tag{4.1}$$

$$l_{n,c} = -w_{n,c}[p_c y_{n,c} \cdot \log \sigma(x_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - \sigma(x_{n,c}))] \tag{4.2}$$

In this case, the parameter p_c has been used to ponderate the weight. The p_c parameter is a tensor with a length equal to the number of classes. The weights for each class have been calculated by dividing the number of instances of this class by the number of instances that are not from this class.

$$\frac{Number\ of\ Positive\ Samples}{Number\ of\ Negative\ Samples} \tag{4.3}$$

Table 4.2: Calculated weights for each class using the training set for the class loss.

Dataset	swimmer	floater	boat	swimmer*	floater*	life jacket
SeaDronesSee	7.629	2.592	1.774	5.043	12.058	294.080
SeaDronesSee3		1.515	1.774		3.131	
SeaDronesSee5	7.629	2.549	1.774	5.043	12.058	

5. Experimentation and Results

This chapter analyzes the experimentation process and the results obtained. It is divided into 4 sections as follows. First, Section 5.1 introduces the metrics used in Object Detection tasks. In Section 5.2 a brief explanation on the training parameters and some results obtained is made. The results obtained during the evaluation of the models on the different SeaDronesSee datasets is analyzed in Section 5.3. Finally,in Section 5.4 the results obtained on other datasets are evaluated. All the code and experiments of this thesis can be found on GitHub[1][2]

5.1. Metrics

Before training our models, it is important to understand the metrics that are going to be used. A metric is a numeric value that provides understandable information to the user. They will provide us with valuable information to evaluate the different trainings that are going to be performed. Depending on the task, there are different metrics available. Nevertheless, before explaining the different metrics that have been used, there are some previous concepts that must be clear:

- **True Positive (TP):** Correct detection made by the model
- **False Positive (FP):** Incorrect detection made by the model
- **False Negative (FN):** A ground truth missed (not detected) by the model
- **True Negative (TN):** Background region correctly not detected by the model

5.1.1. Intersection over Union

The first and the most important metric used in object detection task is the Intersection over Union (IoU). This metric determines the degree of overlap between the predictions of our model and the ground truth bounding boxes determined by the dataset. This metric produces a value between 0 and 1 determined by this formula:

$$IoU = \frac{area(groundTruth \cap prediction)}{area(groundTruth \cup prediction)} \tag{5.1}$$

Although its importance, this metric is rarely used by itself. It is commonly used as a threshold to determine whether a prediction is valid or not. Using a threashold α a TP can

[1]YOLOX modified architecture and parameters used: `https://github.com/pabloruizp/YOLOX`
[2]Custom datasets and scripts: `https://github.com/pabloruizp/TFG`

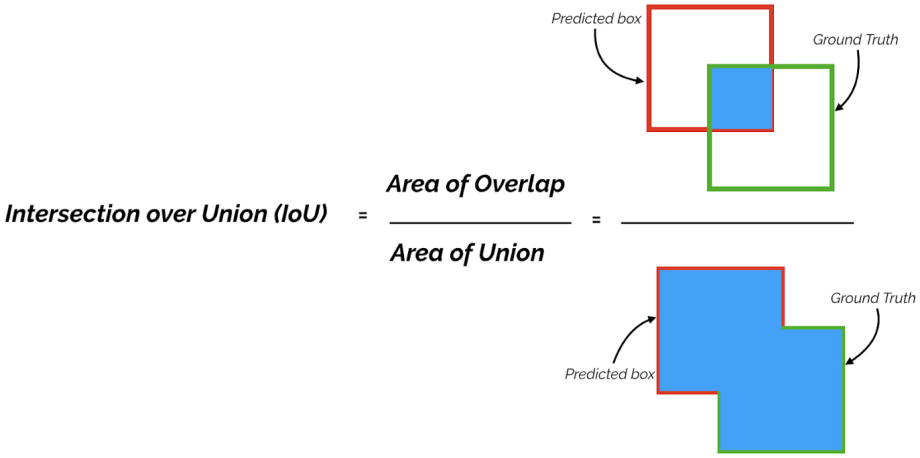


Figure 5.1: Graphical explanation of the Intersection over Union metric[3]

be considered when $IoU(groudTruth, prediction) \geq \alpha$ and a FP can be considered when $IoU(groudTruth, prediction) < \alpha$.

5.1.2. Precision and Recall

Precision can be defined as the exactness of the model to detect relevant objects. Among all the detections made by the model, the ratio of correct detections.The formula to calculate the Precision is:

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \tag{5.2}$$

On the other hand, Recall can be defined as the capacity of the model detection ground truths. The ratio of ground truths defined in the dataset that te model has been able to detect. The formula to calculate the Recall is:

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ groundTruths} \tag{5.3}$$

Those metrics are usually used alongside an IoU threshold. They can even be used with an IoU range. This means that the computed metric is the average of this metric calculated for a range of thresholds with an increment.

[3]Source: `https://miro.medium.com/0*-8wov7TFINqw0xAY`

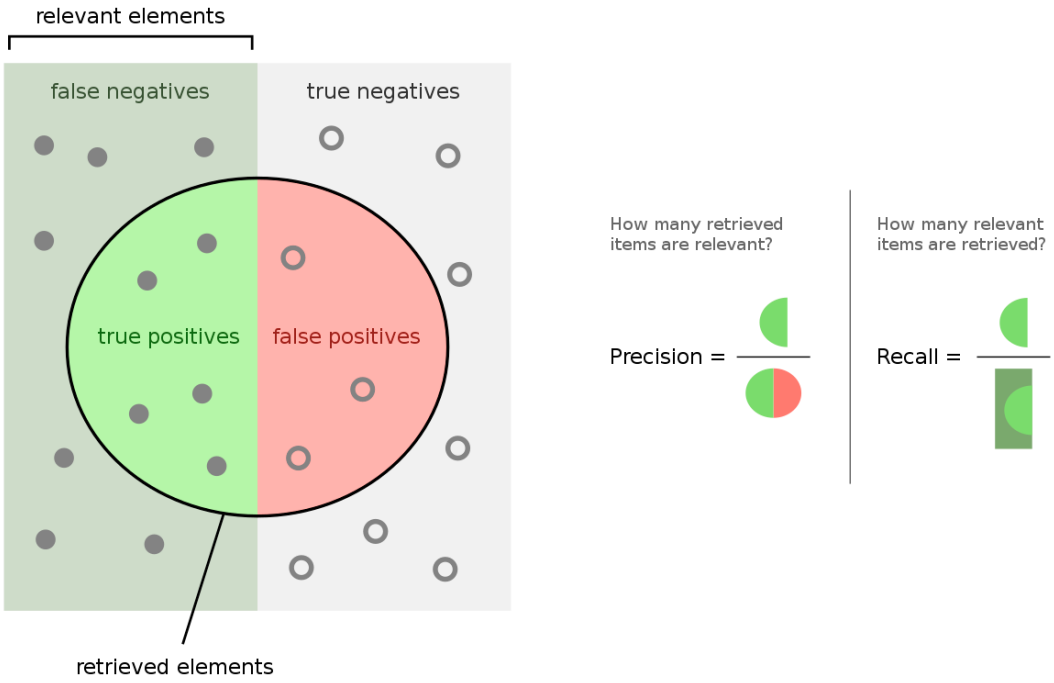


Figure 5.2: Graphical explanation of the Precision and Recall metrics[4]

5.1.3. Average Precision

A perfect model has a Precision and a Recall of 1. This means that all the predictions made by the model are all the ground truths. Object detection models produce predictions alongside confidence in them. This parameter is used as a threshold to filter predictions with a low level of confidence. The lower this threshold, the higher the number of FPs produced by the model. This can be translated as low precision and high recall. The recision-Recall curve (PR) is a plot of those metrics with different confidence thresholds. The Average Precision (AP) is the area under the PR curve. Its formula is:

$$\int_0^1 p(r)\, dr \tag{5.4}$$

This metric is calculated for each class that our model can detect. It is calculated using a fixed IoU threshold value o range. Another derived metric that can be calculated is the mean Average Precision (mAP). This is calculated as the mean AP of all the classes.

[4]Source: `https://en.wikipedia.org/wiki/File:Precisionrecall.svg`

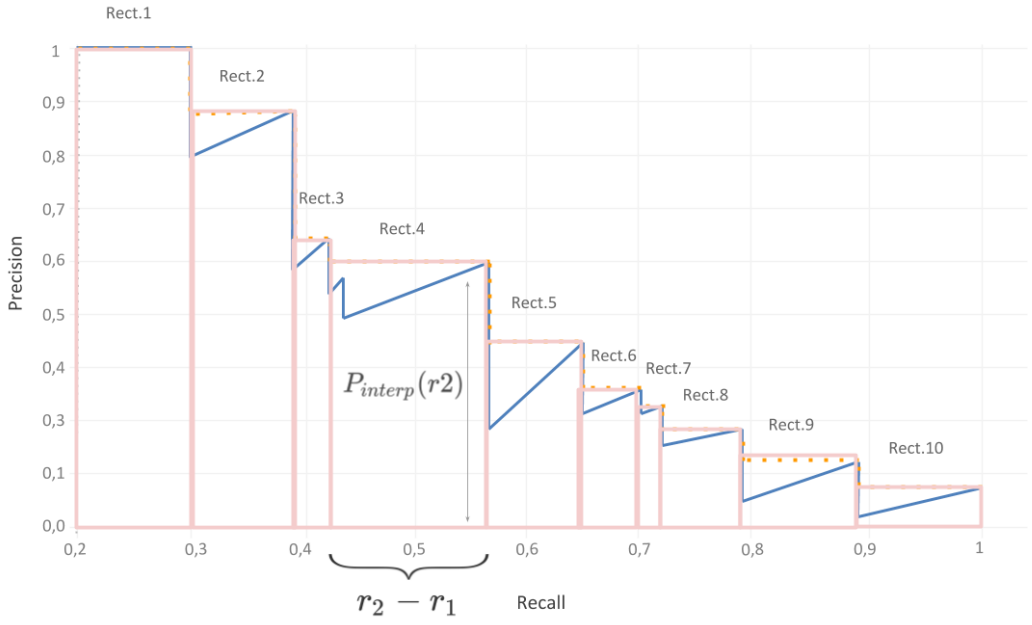


Figure 5.3: Example of a Precision-Recall curve used to calculate the Average Precision.[5]

5.2. Training

In order to train an ML model, there are several considerations that must be made. In big architectures such as YOLOX, the usually are several parameters that can be adjusted in order to personalize some characteristics related to the architecture and the training loop. The most important ones and the values that we have finally used can be found in Table 5.1.

During the experimentation phase of this thesis, several configurations have been tested. At first, the model depth and width are determined by the model selected. In our particular case, we have decided to use the YOLOX-S version as a perfect balance between performance and speed. The pre-trained weights provided by the creators as been used as the start point for training. The most noticeable improvement testing parameters was using a bigger input size. As the resolution of the images at SeaDronesSee is quite big and the objects are quite small, increasing the input size of the models prevents some of the loss of information during the resize of the input images. On the other hand, increasing other parameters such as the number of epochs without data augmentation only generates overfitting.

Other parameters such as the batch size and the number of epochs have been tested. Due to the hardware limitations and the high input size, the maximum batch size achieved has been 8. Regarding the number of epochs and using the hardware mentioned in Chapter 3, the average time per 100 epochs is 1 day. Different experiments have been made and, with the available data, from epoch 200 onwards there are no noticeable improvements.

[5]Source: `https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html`

Table 5.1: Most important YOLOX parameters and used values.

Parameter	Description	Value
num_classes	Number of classes the model can detect	5
depth	Factor of model depth	0.33
width	Factor of model width	0.50
input_size	(height, width) of the input images	(960, 960)
test_size	Output image size during evaluation/test	(960, 960)
warmup_epochs	Epoch number used for warmup	5
max_epoch	Max training epoch	200
warmup_lr	Minimum learning rate during warmup	0
basic_lr_per_img	Learning rate for one image.	0.01 / 64
no_aug_epochs	Number of epochs without data augmentation	20
weight_decay	Weight decay of optimizer	5e-4
mosaic_prob	Probability of applying mosaic augmentation	1
mixup_prob	Probability of applying mixup augmentation	1
flip_prob	Probability of applying flip augmentation	0.5
degrees	Rotation angle range	10.0
mosaic_scale	Scale applied to mosaic augmentation	(0.1, 2)
mixup_scale	Scale applied to mixup augmentation	(0.5, 1.5)

5.3. SeaDronesSee

Now that our models have been successfully trained is time to evaluate their capabilities. The first evaluation process has been made using the validation sets of the datasets used for training. The validation sets of the different versions of SeaDronesSee contain images that the models have never seen before. The results obtained can be seen in Figure 5.4 and Tables 5.2 and 5.3.High quality predictions can be found on Appendix A.

The results obtained show great results on the vessel detection task, which was the principal objective of this thesis. On the other hand, the proposed methods find some difficulties locating people, especially inside vessels. This situation can not be considered an important issue. In real world scenarios, not being able to detect people inside a vessel is not a significant problem as long as the model is able to detect the vessel itself. Using the NVIDIA Tesla K80 as inference device, we have achieved real-time inference at 20 frames per second.

Figure 5.4: Inference on the SeaDronesSee validation set using the SeaDronesSee5 model

Table 5.2: Model Average Precision for each class using the validation set

Model Name	swimmer	floater	boat	swimmer*	floater*	life jacket
SeaDronesSeeBoats			73.310			
SeaDronesSee3		42.378	73.939		11.332	
SeaDronesSee5	38.556	40.475	73.695	12.247	13.925	

Table 5.3: Model mAP at different IoU values and ranges using the validation set

Model Name	mAP@50:0.05:95	mAP@50	mAP@75
SeaDronesSeeBoats	0.733	0.957	0.879
SeaDronesSee3	0.425	0.725	0.430
SeaDronesSee5	0.358	0.666	0.328

5.4. Experiments on other datasets

The trained models have also been tested on other datasets to evaluate their robustness. On datasets such as LS-SSDD-v1.0 and SeaShips, the models are unable to detect any object correctly due to the visible differences to the training datasets. Nevertheless, on datasets such as Seagull and Airbus Ship Detection, the models provide accurate predictions. As the models

Figure 5.5: Predicted labels on the Seagull and Airbus Ship Detection datasets using the SeaDronesSeeBoat model

have not been trained using those datasets, several false positives are produced during the inference due to slight differences in the environment. The results obtained demonstrate the robustness of the proposed method on different seabornne environments using aerial images (Figure 5.5). High quality predictions can be found on Appendix B.

6. Conclusions

This chapter comments the conclusions obtained with the results obtained in this thesis. It is divided into 3 sections as follows. First, in Section 6.1 a conclusion about the results obtained is made. Different proposals to improve the results obtained are presented in Section 6.2. Finally,in Section 6.3 the publications derived from this thesis are mentioned.

6.1. Conclusions

As a result of this thesis, we have obtained a robust model able to accurately detect small vessels and shipwrecked people in different seaborne environments. This model has been successfully tested on several datasets achieving very good results.

In order to achieve those results, several steps have been needed. First of all, an extensive research on the actual methods to detect objects in seaborne environments has been made. We have also maintained conversations with different NGOs related to this task to better understand the problem and the current approaches. Once the problem and the actual solutions were understood, we explored the available datasets that could fit us for this task. After the extensive search and evaluation, the best-suited dataset, and the selected option, was SeaDronesSee. Nevertheless, due to the imbalance present in this dataset, different solutions have been proposed and tested to solve this problem. The selected solution has been the creation of different versions of the original dataset and the use of a weighted loss on the head of the model. Finally, the anchor-free object detector YOLOX has been trained using the selected dataset, the proposed methods and the selected hyper-parameters.

Although the achieved results can be considered as very good, there is always room for improvement. The model has problems due to the reflections on the water created by the sun. In addition, as the data available to test the model is very limited, there are several situations that could not be evaluated. Images with different meteorological situations or reduced light conditions are not available in the actual datasets.

6.2. Future Work

In order to solve the previously mentioned problems, we have to determine future lines of work aiming to generate an even more robust model. Due to the experimentation that has been done in this thesis, we consider that the best and most optimal way to improve the model's performance is by improving the datasets. Although SeaDronesSee has meant a significant improvement with respect to the previously available datasets, there are still situations that are not represented enough.

Thanks to all the gathered datasets and the connections with the different NGOs, a possible option to improve the actual dataset used for training is mixing images from different sources. Using real images for training our models is always the best option. However, the significant

changes between the images from different sources might be harmful to the training of our model.

If this option does not provide the expected results, another proposed solution is the synthetic generation of more data. This approach has already been successfully tested by the creators of the dataset(Kiefer et al., 2021). Using this technique we will be able to easily generate a great volume of data, perfectly labelled in different environments. The difficulty of this approach consists in the generation of realist simulations that could be transferred to real world scenarios.

6.3. Publications

As a result of this thesis and project, we have published our proposal and results in the 17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022). This publication is *Small Vessel Detection in Changing Seaborne Environments using Anchor-Free Detectors on Aerial Images*.

Bibliography

Ahmad, J., Muhammad, K., & Baik, S. (2017, 08). Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. *PLOS ONE*, *12*, e0183838. doi: 10.1371/journal.pone.0183838

Airbus. (n.d.). *Airbus ship detection challenge.* Retrieved from `https://www.kaggle.com/competitions/airbus-ship-detection/data`

Bochkovskiy, A., Wang, C., & Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, *abs/2004.10934*. Retrieved from `https://arxiv.org/abs/2004.10934`

Cane, T., & Ferryman, J. (2018). Evaluating deep semantic segmentation networks for object detection in maritime surveillance. In *2018 15th ieee international conference on advanced video and signal based surveillance (avss)* (p. 1-6). doi: 10.1109/AVSS.2018.8639077

Chen, Z., Chen, D., Zhang, Y., Cheng, X., Zhang, M., & Wu, C. (2020). Deep learning for autonomous ship-oriented small ship detection. *Safety Science*, *130*, 104812. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0925753520302095` doi: https://doi.org/10.1016/j.ssci.2020.104812

Dubey, S., Olimov, F., Rafique, M. A., & Jeon, M. (2021, 11). Improving Small Objects Detection using Transformer.
doi: 10.36227/techrxiv.16921000.v2

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: exceeding YOLO series in 2021. *CoRR*, *abs/2107.08430*. Retrieved from `https://arxiv.org/abs/2107.08430`

Girshick, R. B., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, *abs/1311.2524*. Retrieved from `http://arxiv.org/abs/1311.2524`

Gugger, S., & Howard, J. (2020). *Deep learning for coders with fastai and PyTorch.* Sebastopol, CA: O'Reilly Media.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, *abs/1406.4729*. Retrieved from `http://arxiv.org/abs/1406.4729`

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., … Minh, M. T. (2022, February). *ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.6222936` doi: 10.5281/zenodo.6222936

Kiefer, B., Ott, D., & Zell, A. (2021). Leveraging synthetic data in object detection on unmanned aerial vehicles. *CoRR*, *abs/2112.12252*. Retrieved from `https://arxiv.org/abs/2112.12252`

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

Li, H., Deng, L., Yang, C., Liu, J., & Gu, Z. (2021). Enhanced yolo v3 tiny network for real-time ship detection from visual image. *IEEE Access*, *9*, 16692-16706. doi: 10.1109/ACCESS.2021.3053956

Lin, T., Goyal, P., Girshick, R. B., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, *abs/1708.02002*. Retrieved from `http://arxiv.org/abs/1708.02002`

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., … Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, *abs/1405.0312*. Retrieved from `http://arxiv.org/abs/1405.0312`

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *CoRR*, *abs/1712.04621*. Retrieved from `http://arxiv.org/abs/1712.04621`

Porto Marques, T., Albu, A., O'Hara, P., Serra Sogas, N., Morrow, B., Mcwhinnie, L., & Canessa, R. (2020, 11). Size-invariant detection of marine vessels from visual time series.. doi: 10.1109/WACV48630.2021.00049

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, *abs/1506.02640*. Retrieved from `http://arxiv.org/abs/1506.02640`

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, *abs/1804.02767*. Retrieved from `http://arxiv.org/abs/1804.02767`

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, *abs/1506.01497*. Retrieved from `http://arxiv.org/abs/1506.01497`

Ribeiro, R., Cruz, G., Matos, J., & Bernardino, A. (2019). A data set for airborne maritime surveillance environments. *IEEE Transactions on Circuits and Systems for Video Technology*, *29*(9), 2720-2732. doi: 10.1109/TCSVT.2017.2775524

Shao, Z., Wu, W., Wang, Z., Du, W., & Li, C. (2018, 08). Seaships: A large-scale precisely-annotated dataset for ship detection. *IEEE Transactions on Multimedia*, *20*, 1-1. doi: 10.1109/TMM.2018.2865686

Spagnolo, P., Filieri, F., Distante, C., Mazzeo, P. L., & D'Ambrosio, P. (2019). A new annotated dataset for boat detection and re-identification. In *2019 16th ieee international conference on advanced video and signal based surveillance (avss)* (p. 1-7). doi: 10.1109/AVSS.2019.8909831

Varga, L. A., Kiefer, B., Messmer, M., & Zell, A. (2021). Seadronessee: A maritime benchmark for detecting humans in open water. *CoRR*, *abs/2105.01922*. Retrieved from https://arxiv.org/abs/2105.01922

Wang, Y., Wang, C., Zhang, H., Dong, Y., & Wei, S. (2019). A sar dataset of ship detection for deep learning under complex backgrounds. *Remote Sensing*, *11*(7). Retrieved from http://www.mdpi.com/2072-4292/11/7/765 doi: 10.3390/rs11070765

xview3: Dark vessels. (n.d.). Retrieved from https://iuu.xview.us/dataset

Yang, J., Xie, X., Shi, G., & Yang, W. (2020). A feature-enhanced anchor-free network for uav vehicle detection. *Remote Sensing*, *12*(17). Retrieved from https://www.mdpi.com/2072-4292/12/17/2729 doi: 10.3390/rs12172729

Zeiler, M. D., & Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, *abs/1311.2901*. Retrieved from http://arxiv.org/abs/1311.2901

Zhang, T., Zhang, X., Ke, X., Zhan, X., Shi, J., Wei, S., … Kumar, D. (2020). Ls-ssdd-v1.0: A deep learning dataset dedicated to small ship detection from large-scale sentinel-1 sar images. *Remote Sensing*, *12*(18). Retrieved from https://www.mdpi.com/2072-4292/12/18/2997 doi: 10.3390/rs12182997

Zhang, T., Zhang, X., Li, J., Xu, X., Wang, B., Zhan, X., … Wei, S. (2021). Sar ship detection dataset (ssdd): Official release and comprehensive data analysis [Journal Article]. *Remote Sensing*, *13*(18). doi: 10.3390/rs13183690

List of Acronyms

AIS	Automatic Identification System.
AP	Average Precision.
CNN	Convolutional Neural Network.
CUDA	Compute Unified Device Architecture.
DTIC	Department of Information Technologies and Computing.
GPU	Graphics Processing Unit.
IoT	Internet of Things.
IoU	Intersection over Union.
mAP	mean Average Precision.
ML	Machine Learning.
NGO	Non-Government Organization.
PR	recision-Recall curve.
SAR	Synthetic Aperture Radar.
SSH	Secure Shell.
TPU	tensor processing unit.
UAV	Unmanned Aerial Vehicle.

A. Appendix I: Results on SeaDronesSee

Figure A.1: Model predictions on SeaDronesSee image.

Figure A.2: Model predictions on SeaDronesSee images.

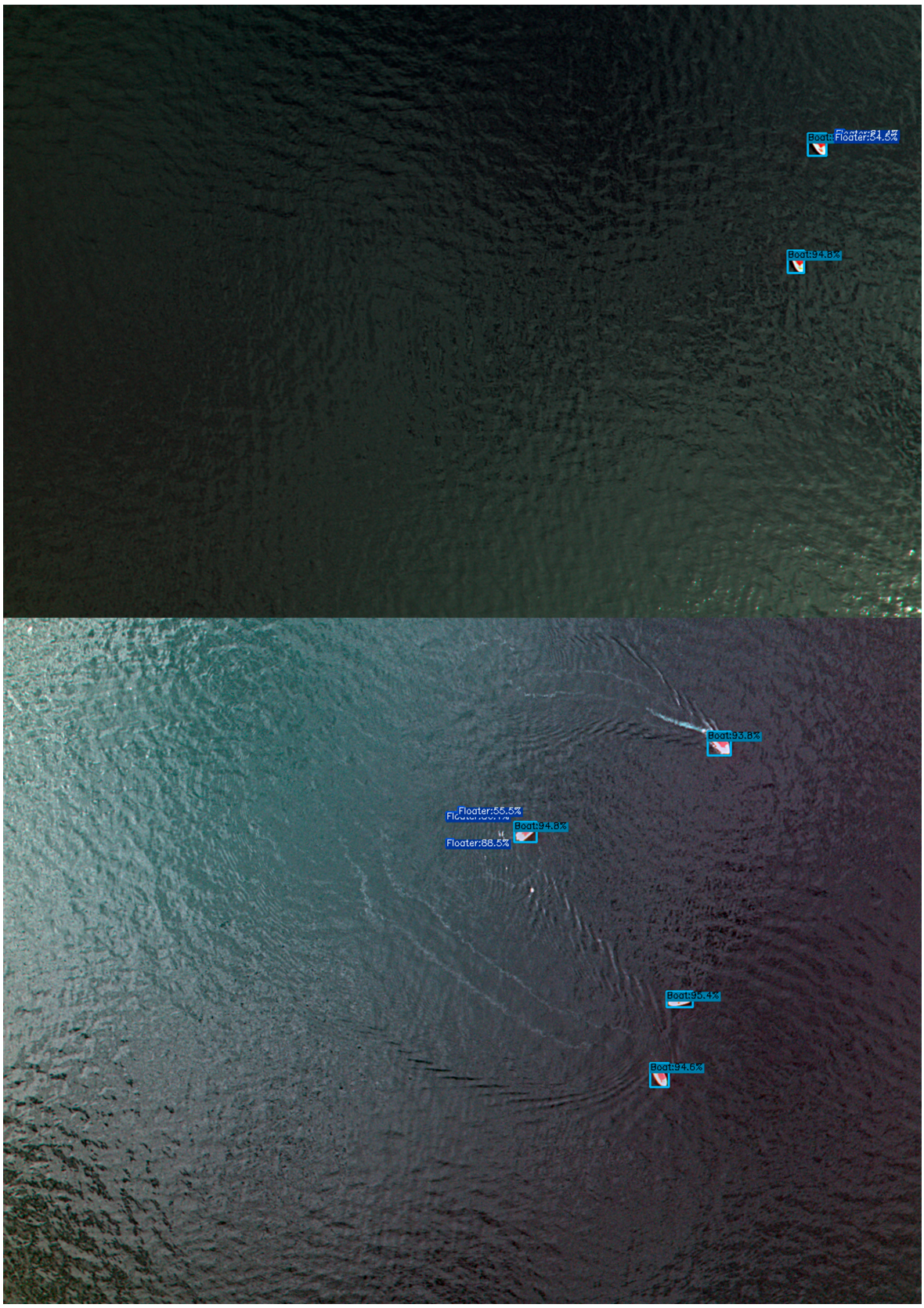


Figure A.3: Model predictions on SeaDronesSee images.

B. Appendix II: Results on other datasets

Figure B.1: Model predictions on Seagull image.

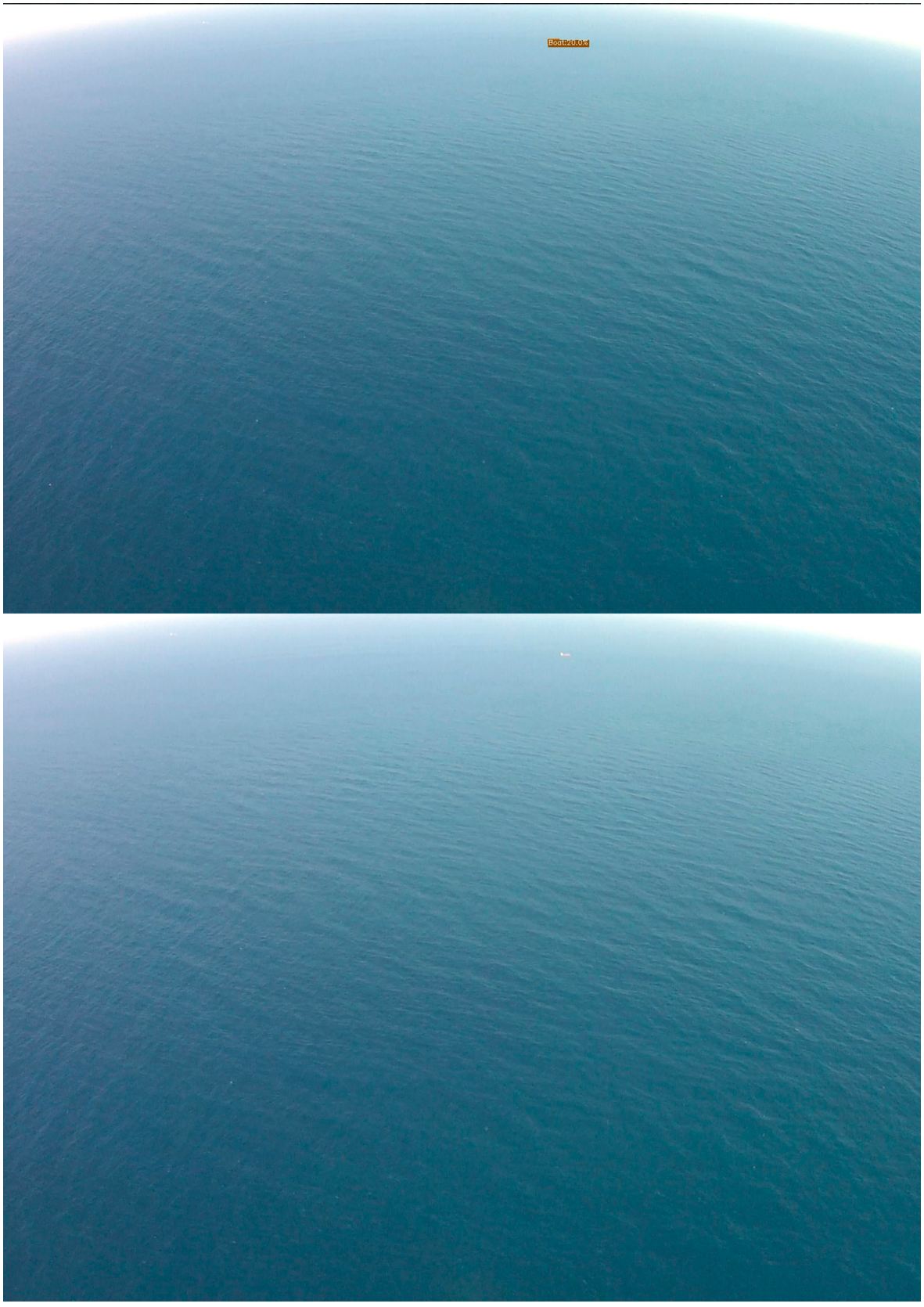


Figure B.2: Model prediction on very distant vessel from Seagull images.

Figure B.3: Model predictions on Airbus Ship Detection images.