Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Phuong Truong

Crown-of-Thorns Starfish Detection
by state-of-the-art YOLOv5

Master's Thesis
Espoo, June 6, 2022

Supervisor: Alexander Jung, Assistant Prof., Aalto University
Advisor: Huynh Vo, M.Sc. (Tech.), Aalto University

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: Phuong Truong

Title:
Crown-of-Thorns Starfish Detection by state-of-the-art YOLOv5

Date: June 6, 2022 **Pages:** 87

Major: Computer Science **Code:** SCI3042

Supervisor: Alexander Jung, Assistant Prof.

Advisor: Huynh Vo, M.Sc. (Tech.)

Crown-of-Thorns Starfish outbreaks appeared many decades ago which have threatened the overall health of the coral reefs in Australia's Great Barrier Reef. This indeed has a direct impact on the reef-associated marine organisms and severely damages the biological diversity and resilience of the habitat structure. Yet, COTS surveillance has been carried out for long but completely by human effort, which is absolutely ineffective and prone to errors. There emerges an urge to apply recent advanced technology to deploy unmanned underwater vehicles for detecting the target object and taking suitable actions accordingly. Existing challenges include but not limited to the scarcity of qualified underwater images as well as superior detection algorithms which is able to satisfy major criteria such as light-weight, high accuracy and speedy detection. There are not many papers in this specific area of research and they can't fulfill these expectations completely.

In this thesis, we propose a deep learning based model to automatically detect the COTS in order to prevent the outbreak and minimize coral mortality in the Reef. As such, we use CSIRO COTS Dataset of underwater images from the Swain Reefs region to train our model. Our goal is to recognize as many starfish as possible while keeping the accuracy high enough to ensure the reliability of the solution. We provide a comprehensive background of the problem, and an intensive literature review in this area of research. In addition, to better align with our task, we use F_2 score as the main evaluation metrics in our MS COCO-based evaluation scheme. That is, an average F_2 is computed from the results obtained at different IoU thresholds, from 0.3 to 0.8 with a step size of 0.05. In our implementation, we experiment with model architecture selection, online image augmentation, confidence score threshold calibration and hyperparameter tuning to improve the testing performance in the model inference stage. Eventually, we present our novel COTS detector as a promising solution for the stated challenge.

Keywords: Convolutional Neural Networks, COTS, CSIRO Dataset, deep learning, marine organisms, starfish, underwater images, You Only Look Once

Language: English

Acknowledgements

Aside from my profession as an analytics consultant, I have a special interest in the environment and practical applications that mitigate harmful impacts on the natural world. Never in my wildest dreams do I imagine that I would rigorously research about object detection model to protect the coral reefs from threatening species. This work cannot be done without assistance and advice. By this opportunity, I would like to express my gratitude towards all the parties involved in my master's project.

Most importantly, I am incredibly thankful to the team from Australia's national science agency CSIRO and Google for publishing the COTS Detection Dataset. Data has always been the cornerstone of machine learning and the availability of this dataset supports the community significantly in the underwater surveillance research area. Furthermore, I acknowledge Glenn Jocher from Ultranalytics for his enormous effort in building such a high-quality and open-source implementation of YOLOv5 - the detector model used in this thesis. Thanks to his splendid piece of engineering, my research programming efforts have been facilitated considerably.

Besides my strong motivation, much of the success of this thesis is attributed to the tremendous support from different professionals. Indeed, I cannot express how thankful I am to my supervisor Alex Jung and the research group. Additionally, I sincerely commend my advisor Huynh Vo, who has supported me with ardor in developing the research plan and implementation ideas. It is my great pleasure to work with all these amazing people who spend time out of their busy life to review my work with care and enthusiasm. By all means, I am wholeheartedly grateful to my family who has supported and motivated me unconditionally throughout this long and challenging journey.

Espoo, June 6, 2022

Phuong Truong

Abbreviations and Acronyms

Abbreviations

API	Application Programming Interface
AUV	Autonomous Underwater Devices
CNN	Convolutional Neural Networks
COTS	Crown-of-Thorns Starfish
CPU	Central Processing Unit
CSIRO	Commonwealth Scientific and Industrial Research Organization
CUDA	Compute Unified Device Architecture
FCN	Fully Connected Network
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GIoU	Generalized Intersection over Union
GPU	Graphical Processing Unit
HSV	Hue Saturation Value
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
MOB	Merging of Overlapping Bounding Boxes
NMS	Non-Max Suppression
OICOD	Open Image Challenge Object Detection
PANet	Path Aggregation Network
PASCAL	Pattern Analysis, Statistical Modeling, and Computational Learning
PASCAL VOC	PASCAL Visual Object Classes (challenges)
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROI	Region of Interest
RPN	Region Proposal Network
SPP	Spatial Pyramid Pooling

TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
UUV	Unmanned Underwater Vehicle
XOR	Exclusive OR

Symbols

β	Real factor β used in F score
B_i	Bounding box i
C	The smallest convex shape that encloses two bounding boxes
c	Confidence threshold
$\hat{C}_i$	Prediction confidence score of bounding box i
C_i	True confidence score of bounding box i
$\mathcal{D}$	Detection vector
ζ	Training IoU threshold
ϵ	Bounding box aggregation IoU threshold
ε	Evaluation IoU threshold
λ	Confidence weight
$\mathcal{L}$	Loss
$\mathcal{L}_{reg}$	Bounding box regression loss
$\mathcal{L}_{obj}$	Objectness loss
$\mathcal{L}_{cls}$	Classification loss, only if multiple classes exist
m	Predicted bounding box's index with the highest c from xc
$\mathcal{P}$	Precision
$\mathcal{R}$	Recall
s_i	Confidence score of predicted bounding box b_i^p
xc	Bounding box candidates with confidence score bypassing the confidence threshold c

Operators

A^B	Area of bounding box B
$B_i \cap B_j$	Intersection of bounding boxes B_i and B_j
$B_i \cup B_j$	Union of bounding boxes B_i and B_j
$\oslash$	Element-wise division

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Aquatic habitat is a complex ecosystem which includes connections and interactions among many different marine species and organisms. These interrelationships are mutually beneficial, but sometimes they can destroy each other. In reality, this is not completely a bad thing since this cancellation helps to maintain the biological balance and diversity. However, if either of the species emerges to be more dominant than the other, the mutual influence definitely becomes imbalanced and eventually either of the involved parties diminishes. This scenario has happened in the Great Barrier Reef (Australia) for many decades. This world's largest breathtakingly beautiful coral reefs have been severely endangered by a native creature to the Reef - Crown-of-Thorns Starfish (COTS). This species feeds on live coral and it is highly fecund and can easily reach pest-level under favourable conditions. According to the prediction, the next outbreak is likely to happen within a couple of coming years.

There are different surveillance and control programs conducted by humans and thus considerably prone to error and inefficient on the large scale. With the latest advances in technology, unmanned underwater vehicles (UUV) would be the most appropriate alternative for COTS surveillance. There exists limitations in human effort (health and safety issues), demanding requirements (experienced divers, ships, durable equipments), and obstacles from other conditions (hindrance of weather, watery turbidity). Indeed, the adoption of UUV can overcome all of the aforementioned challenges as it automatically collects images from the reef and performs a robust starfish detection. Hence, it can deliver an accurate, timely, and reliable assessment for the surveillance program. We believe that the true potential of UUV and its utilization are soon to be unlocked in the near future.

Given as a background context, the starfish are typically camouflaged in the underwater environment and they are potentially occluded because of

the camera altitude and angle, which makes the task even more challenging. As such, our goal in this thesis is to efficiently automate this troublesome visual search task through computer vision. Deep learning has significantly revolutionized the field of object detection recently, thus it is a promising candidate for this kind of detection task. Due to its compelling capability to learn representations from data, especially high dimensional data such as images, we are motivated to adopt deep learning to build our COTS detector in this work.

1.1 Problem Statement

We choose the latest and renowned YOLOv5 deep learning algorithm as the base model for detecting the Crown-of-Thorns Starfish, supporting this marine creature's surveillance and controlling the outbreak. Simply defined, our task is to localize and recognize this small and obscure starfish from high-resolution underwater images as many as possible within a high level of accuracy and at an acceptable speed. Specifically, we follow the typical machine learning workflow: understand the dataset through exploratory data analysis, train the model and analyze the validation results, experiment with different hyperparameters, and evaluate model performance against an unseen test dataset. By this, we make a certain number of hypotheses and conduct many iterative improvements. Motivated by an intensive literature review on the area of research, we test different adaptations to the base model, including but not limited to model architecture, online augmentations during training, and hyperparameters. Since there exists no similar works for us to benchmark the performance from, we conduct the experiments until we achieve desirable results from the model that can generalize well to our testing dataset. Thus, this desirable extent is based on our expectations and one should notice that it is definitely not any standard criterion for this problem at hand. Apart from our effort to find an optimal model, we first analyze this problem comprehensively to formulate it properly. In particular, the underwater environment poses several challenges to the image's quality. Additionally, the images are in sequence since they are derived from the video. These issues are indeed addressed properly through our work. Regarding the evaluation process, we clearly specify how our detector's performance is evaluated through a particular evaluation scheme. This ensures we leave no room for ambiguous results.

1.2 Structure of the Thesis

After the big picture, here comes the structure of this thesis:

- **Chapter 2** - an intensive literature review. It firstly glances through computer vision, object detection problems specifically, and the evolution of deep learning. The COTS outbreak and its significant consequences are discussed in great details. Different surveillance methods, including human interference and vision-based models, are presented. A strong focus is on the latter, hence a rigorous review of traditional machine learning models and deep learning models (such as Fast-CNN, novel YOLO family).

- **Chapter 3** - problem formulation. We define the COTS detection problem in a complete picture and choose an appropriate set of evaluation metrics to find the best detector.

- **Chapter 4** - major components of the COTS detector: data, model, and loss. Relevant data augmentation techniques are chosen to enhance the original dataset, rectifying the underwater environment. Important model implementation details and procedures are discussed. This chapter also describes the multi-task loss function when optimizing both bounding box regression loss and objectness loss.

- **Chapter 5** - comprehensive experimentation pipeline. Different experiments are presented regarding the choices of parameters and validation results. The best models are selected to undergo an inference and generalization assessment.

- **Chapter 6**: - evaluation and discussion. The final model is evaluated thoroughly to gain deeper insights into what have been done well and what would be the potential areas for future development.

- **Chapter 7** - conclusion. This chapter summarizes the work and discusses briefly its usability on the research area and societal impact.

Chapter 2

Literature Review

This chapter provides a review of object detection problem and key milestones in its revolutionary history. State-of-the-art object detection methods represent a series of improvements building upon prior accomplishments or failures. Therefore, it's worth glancing through the epitome of computer vision history and object detection advancements. Notably, Convolutional Neural Networks (CNN) has brought an unprecedented attention of AI practitioners back to deep learning and pushed object detection to the new frontier. Besides, this chapter provides an introductory description of the thesis research problem - the coral loss in Great Barrier Reef due to the outbreak of Crown-of-Thorns Starfish. It's worth understanding the problem thoroughly and analyzing existing papers with their proposed solutions to comprehensively evaluate and look for a better direction. Although this review highlights the more notable achievements while slicing much of the details, it highly focuses on the main architectures and algorithms in underwater object detection. The review provides both practical and theoretical importance to solve the research problem of this thesis.

2.1 Computer Vision and Object Detection

Computer vision is a sub-stream in computer science and engineering which simulates human vision and cognitive ability [21]. Among all, object detection is one of the fundamental computer vision problems. It aims to answer the question *"What objects are where in the given image?"*. Specifically, object localization and object classification tasks are done through three main stages, including region selection, feature extraction and classification. [64]. Undeniably, object detection provides a semantic and spatial understanding of images and videos to solve complex and high-level tasks such as image

classification, image captioning, object localization, object tracking, facial recognition, to name just a few [28, 36, 63]. Therefore, there exists a wide range of applications utilizing object detection, including but not limited to human interaction, robotics, autonomous driving, security, content-based image retrieval, AI-assisted medical diagnosis, autonomous retail checkout systems, video surveillance and augmented reality etc. [24, 28]. In the next section, we provide a snapshot of computer vision history over different periods: early neuron network (origin), AI winter (decline), deep neural network (prosperity). The fundamentals behind deep learning are discussed briefly together with its revolution in the context of computational resources, data availability and novel algorithms.

2.1.1 A brief history

Several neuroscientific researches have been conducted to understand human visual perception. By observing how neurons react to different stimuli, the scientists suppose that human vision is fundamentally spatially hierarchical [5]. In specific, our neurons detect simple features first (edges), then feed into the more complex visual representations (shapes, textures, patterns). This knowledge enlightens the computer scientists in using hierarchical approach to enable the computer to perceive and process images by extracting features through several layers of *"artificial neurons"*. Figure 2.1 explains how biological neuron and its operating principles inspire the idea of *"artificial neurons"*.

Early artificial neural networks could date back to 1940s and it took some decades for the method to be completely supplanted by its modern variants. In 1950s, Frank Rosenblatt [43] introduced the first neural networks model and supervised learning algorithm. The model was known as *"Perceptron"*, which classified data by using only one layer, given that the data was linearly separable. After a while, Minsky and Papert [34] (1969) discovered the limitations in the concept. *Perceptron* could not solve XOR classification problem where data clusters are not linearly separable. Indeed, it required a multi-layer perceptron with non-linear activation function to be able to extract non-linear features. Yet, the training of these multi-layer perceptron was computationally costly. [36]. Needless to say, the community have long been struggle to search for an efficient way to train these networks. Such factors contributed to the slow pace in AI research during 1970s; the period we commonly know as "AI Winter". The governments and corporations were increasingly losing their faith in the field, thus dried up the funding and investments. Artificial intelligence in general was almost relegated to the realm of science fiction. Only until mid-1980s, neural networks revived the

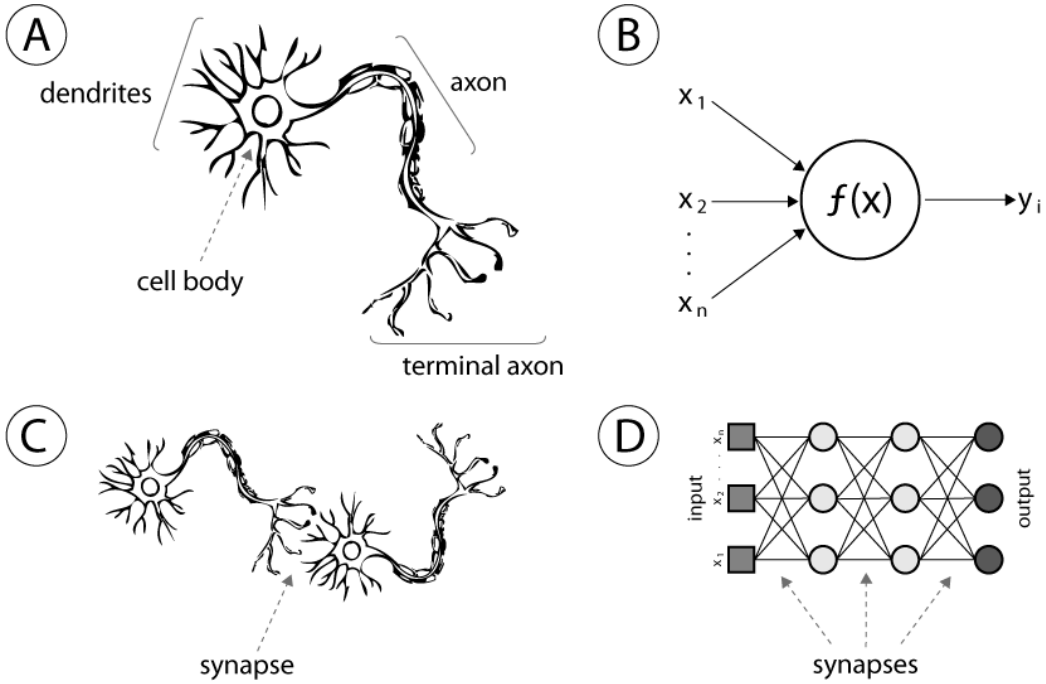

Figure 2.1: An anatomy of a biological neuron (A) and an artificial neuron (B). The dendrites and axon in the biological neuron correspond to input (x_i) and output (y_i) in the artificial neuron. Cell body, also known as nucleus, is responsible for processing input signals and firing the output signals. Similarly, input x_i are weighed separately, summed up and passed through an activation function $f(X)$ (B) to produce output y_i. Synapse connects the terminal axon to other neuron dendrites (C). Rooted by the same concept, synapses in artificial neuron interconnect neurons between different layers (D). [22, 45].

interest of the community when Rumelhart and his co-researchers [44, 63] rediscovered backpropagation algorithm and its capability in training deep multi-layer perceptrons [5, 24, 36]. Yann LeCun with his combination of convolutional neural networks and backpropagation in solving handwritten digits classification problem introduced the first successful and practical application of neural networks - an automation in reading ZIP codes on mail envelopes for United States Postal Service in the 1990s. [5].

Although neural networks just triggered a wave of interest, it was quickly sent back to oblivion in early 2000s due to the limitation in computational resources, the lack of large scale data set, the overftiting performance compared to other machine learning algorithms. Typically, kernel method with

its best known *support vector machine (SVM)*, developed and reformulated by Vladimir Vapnik and Corrinna Cortes in 1995 at Bell Labs, has become extremely popular for a long time although its former linear formulation was already proposed by Vapnik and Alexey Chervonenkis in 1963. [5]. The algorithm was backed by extensive theory and mathematical interpretation. SVM has exhibited a stunning performance on simple classification within tractable computation. Besides, logistic regression (LR) and random forests (RF) were among shallow models yet dominated in real-world applications such as spam filtering system and advertisement recommendation [24]. Neural networks with many layers, on the other hand, suffered from vanishing gradient problem and did not work well in practice. Indeed, it was considered as black boxes and could not be explained theoretically [24]. Yet, kernel functions are typically crafted manually rather than learned directly from data. Hence, SVM is a rather shallow method when dealing with either large datasets or perceptual problem (image classification as a typical example) where feature engineering is absolutely a daunting task.

Neural network still kept a long silence although it has shown potential in tackling many complex tasks. A small group of AI practitioners continued their research and started to make certain significant breakthroughs. Hinton et al. heated up the academic community with his article *"Reducing the dimensionality of data with neural networks"*, supported by Canadian Institute for Advanced Research (CIFAR). An *autoencoder* architecture was formed by an adaptive and multi-layer encoder to transform high-dimensional data to lower representation and a similar decoder network to recover the data, which worked much better than well-known PCA technique. [19]. Deep networks outperformed other shallow machine learning models and ultimately popularized the term *deep learning* among the community [24]. The first practical success of modern deep learning came in 2011 when Dan Ciresan from IDSIA (Switzerland) won two niche competitions (the ICDAR 2011 Chinese character recognition competition and the IJCNN 2011 German traffic signs recognition competition) with GPU-trained deep neural networks [5]. The watershed moment came in 2012 when Alex Krizhevsky and his group, with the advisory from Geoffey Hinton, used convolution neural networks for the first time ever to solve ImageNet LSVRC-2010 image classification challenge and achieved a top-five accuracy of 83.6%. Since 2012, deep convolutional neural networks (*convnets*) has become a state-of-the-art algorithm for most computer vision tasks and other contests such as robust reading challenge in ICDAR, Microsoft image recognition and captioning challenge (COCO) etc [24]. It has surpassed all traditional methods and brought object detection to a new frontier.

2.1.2 What makes deep learning outstanding?

Having mentioned before, deep learning - a sub-field in machine learning, has rose to prominence in early 2010s. Fundamentally, *machine learning* algorithms try to learn useful representations from the input that get us closer to the output, while *deep learning* in specific is learning by successive layers of neurons to find increasingly meaningful representations of the input data. *"Deep"* refers to deeper understanding of data through multi-stage layers of representation. In other words, deep learning expresses the ability to learn semantic, high-level and deeper features. In computer vision tasks, traditional object detection methods are adhere to handcrafted features and shallow trainable architectures [63]. Deep learning, on the other hand, automates heavy engineering-pipeline and simplifies learning workflow considerably by only single and end-to-end trainable deep-learning model [5, 54, 63].

There are two fundamentals behind deep learning's superiority, namely *Convolutional Neural Networks* and *backpropagation algorithm* [5]. Backpropagation stands for backward propagation of errors [54], which is the central algorithm in deep learning for computer vision [5]. It starts with the loss value from the last layer and works backward to compute the loss value contributed by each parameter. It helps to realize how much loss each node is responsible for and subsequently adjust the network's weights and bias in order to minimize the cost function. The proper tuning of weights ensures the model reliability by enhancing its generality. Convolution neural networks has become a representation architecture of deep learning thanks to its superiority over traditional methods.

Densely connected layer learns global patterns from all pixels of the image while convolution layer learns only local patterns in small 2D windows of the input, illustrated in Figure 2.2. After learning a certain pattern, a *convnets* can recognize that pattern in other areas of the same image (translation-invariant) or anywhere in the new images (generality and therefore reusability). A densely connected model, in contrast, has to learn again when the pattern appears in a different location, which is hugely inefficient when processing images. In addition, it requires more training examples to learn the representations that generalize well. [5]. Another key characteristic of *convnets* is its ability to learn spatial hierarchies of patterns. The first layer learns local, highly generic feature map such as visual edges, colors and textures. The higher layers combine the outputs from the first layer to form the larger patterns and extract a more abstract concept. Ultimately, the last layer learns the most complex and abstract visual concepts from the image. [5]. Figure 2.3 illustrates spatial hierarchies of patterns learnt by convolution layers.

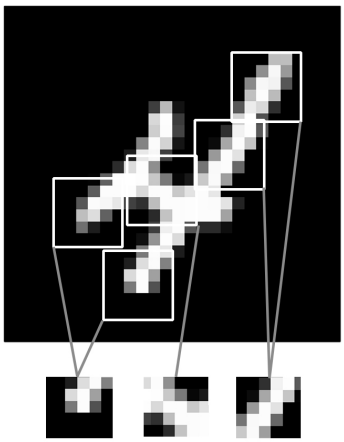

Figure 2.2: Local patterns. Figure is derived directly from *Deep Learning with Python*, Second Edition by François Chollet [6]

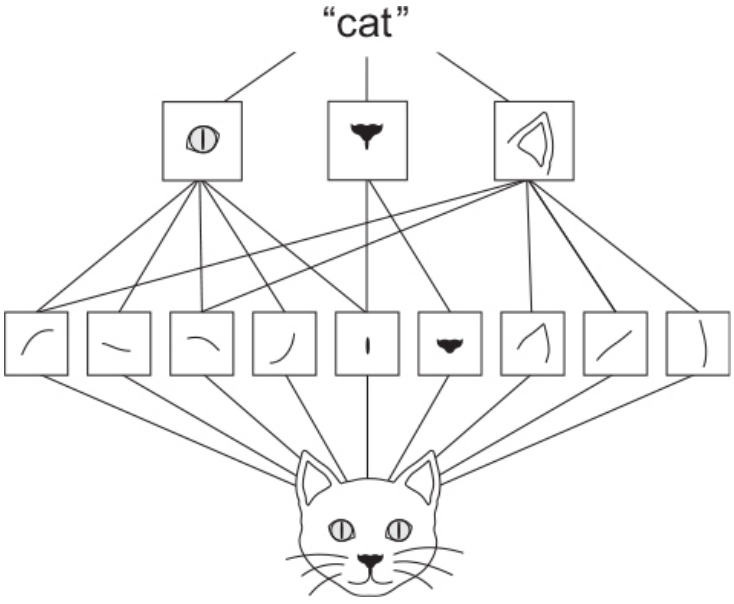

Figure 2.3: Elementary lines and textures in the first convolution layer form simple features such as eyes and ears in the second convolution. Altogether combines into high-level concept of a cat. Figure is derived directly from *Deep Learning with Python*, Second Edition by François Chollet [6]

2.1.3 Major forces behind the flourish of deep learning

Machine learning methods combine three key components, namely data, model and loss, within computationally efficient implementations to continuously adapt and search for the best hypothesis [25]. Machine learning isn't mathematics that can be solved with a pen and a piece of paper. Instead, it is engineering science with an underlying principle of *"trial and error"*, [5, 25]. Backboned by such principle, the field has grown through experimental findings, not by theory. Therefore, new ideas only become feasible when massive data and appropriate hardware are in place. According to Chollet F. (2018) [5], the three main forces which have driven the advancements in machine learning in general, and deep learning in specific are powerful computation ability (hardware), large-scale datasets, and advanced algorithms.

- **Hardware**. In the past few decades, we witnessed the bloom in the gaming industry with an emergence of the more powerful and high performance graphics chips. Throughout 2000s, technology giants such as NVIDIA and Advanced Micro Devices (AMD) were dedicated to developing fast and high performance parallel computer systems to render complex 3D scenes on screen in real time and enhance video games photo-realism. [5, 63]. In 2007, NVIDIA launched CUDA - a parallel computing platform and API for its line GPUs, which simultaneously powered the AI community considerably thereafter. A few decades ago, running a small deep learning model on a personal latop was tractable but that was no longer the case with the modern hardware. A small number of GPUs can replace a massive cluster of CPUs in highly parallellizable applications, which allows an efficient training of deep neural networks. Large companies have invested in developing increasingly specialized and special chips to train deep learning model such as NVIDIA Tesla K80 or Google TPU. Google chips have been reported 10 times faster and far more energy efficient than top-of-the-line GPUs in the market. [5]. Recently, the third generation of TPU card in 2020 represents 10,000 times more computing power than the Intel Touchstone Delta (1990). [6].

- **Datasets and benchmarks**. As we all know, deep learning algorithms learn the feature representations directly from data [28]. In other words, without massive data as raw materials, deep learning would be unable to power our intelligent machine. The internet took off allows the distribution of very large datasets to our machine learning community. [5]. The emergence of the large-annotated training data set such as ImageNet since 2007 has been attributed to the recovery of

deep learning from *"AI winter"* period [63]. Some benchmark datasets and their characteristics are provided in Table 2.1. Each was released as a publicly available set of images together with annotations (bounding boxes, instance segmentation, etc.), given in specific detection challenges (classification, localization, visual relationship etc.) [28].

- **Advanced algorithms**. Deep learning before did not prove its reliability and potential due to fairly shallow neural networks model; specifically, using only one or two layers of representations [5]. Later on, there are not only advances in the design of network structures but also in training strategies [63]. There appears many more algorithmic improvements such as better activation functions (ReLU), better weights initialisation and better optimization schemes (RMSProp and Adam). Additionally, the emergence of new techniques such as drop-out regularization, batch normalization, residual connections and depth-wise separable convolutions allows the training of thousand-layer-depth model. [5]. Overfitting problem has been relieved and the training has become far more efficient [63]. Deep neural networks (DNNs) is a robust training algorithm which acts differently from traditional approaches. Its deep architecture allows to learn complex features without a manual feature design [63]. Indeed, DNNs completely automates feature engineering and learns all features in a single-forward pass.

2.2 Crown-of-Thorns Starfish Detection

The Great Barrier Reef (the Reef) is a World Heritage Area of Outstanding Universal Value, one of the most remarkable natural gifts - the world's largest breathtakingly beautiful coral reefs. It has been symbolizing Australia's national identity for many decades. The Reef, consisting of over 3000 individual reef systems and coral cays, provides a continental-scale underwater structures and ecosystem services for an abundance of marine creatures. The healthy and functioning coral reefs underpin the integrity and biodiversity of the Reef, as well as retain the cultural and heritage values, coupled with the economic benefits [1]. Recently, the overall health of the Reef has been threatened severely by the Crown-of-Thorns Starfish outbreaks. It is worth understanding this species's biology and the stimulus behind this phenomenon, which are briefly described in the following sections. In addition, there discusses different monitoring actions to minimize coral mortality and promote recovery when the outbreaks are underway.

Dataset	Total images	Year	Classes
PASCAL VOC	VOC07: 5K images + 12.6K annotated objects	2007	20
	VOC12: 11.5K images + 27.4K annotated objects	2012	20
	Characteristics: common objects (person, animal, vehicle, indoor objects), multiple objects in one image, larger instra-class variations, close to real-world applications.		
ILSVRC	ILSVRC-13: 416K images + 401K annotated objects	2013	200
	ILSVRC-17: 477K images + 534K annotated objects	2017	200
	Characteristics: many more images and categories than VOC.		
MS COCO	MS COCO18: 123K images + 897K annotated objects	2018	80
	Characteristics: bounding box annotations + per-instance segmentation, small objects (area is smaller than 1% of the image), densely located.		
Open images	OICOD18: 1.7M images + 12.2M annotated objects	2018	500
	Characteristics: derived from Open Image V4, tasks include object detection + visual relationship detection (paired objects in a specific relation).		

Table 2.1: Typical benchmark datasets and their characteristics [28, 64]

2.2.1 COTS biology, the outbreaks and major causes

The Crown-of-Thorns Starfish *(genus Acanthaster cf. solaris)* are a natural part of the Reef ecosystem for millennia and not an introduced species [1]. They are named by the poisonous spines that cover their whole surface (body and arms). In general, the COTS have a pelagic larval duration of 10 to 40 days. This long period enables the widespread of dispersal across the seascape. Once they settle to a reef, they feed on live coral throughout their adulthood, thereby grow drastically and reach reproductive maturity within only two years. In fact, they can consume half-of-their-own-body area of coral on a daily basis and leave behind a lifeless white-scarred reef (Figure 2.4) [32].

During the spawning season, lasting from October to February, a full-grown female can produce a minimum of 30 million eggs. Simply speaking, under favourable conditions, the COTS are highly fecund and they can easily reach pest-level densities. In a positive manner, the COTS at low densities maintain the Reef's coral diversity by feeding on preferentially faster-growing corals thereby leaving room for slower-growing species to colonise. [1]. Yet, when the COTS's consumption rate exceeds the coral's growth rate, what's been known as *COTS outbreaks*, the Reef's health will be affected enormously.

The Reef is considered an environmentally sensitive and vulnerable area where several outbreaks of the COTS have occurred in the last 60 years. Four major COTS outbreaks on the Reef were documented: in the 1960s, the late 1970s, the early 1990s, and the recent outbreak detected in 2010 [7]. Unlike coral bleaching, the COTS outbreaks do not emerge as a seasonal phenomenon of an exact estimated interval. It instead initiates approximately every 10 to 20 years and the wave pervades over the Reef for at least 10 to 12 years afterwards [1].

Specifically, an outbreak undergoes two distinct phases, a primary phase with the upsurge of mature COTS in a small area and afterwards a secondary outbreak by the outspread of larvae. The mature starfish over successive generations build up a strong cluster, and with their fecund reproductive capability, they signal the primary outbreak in their own relatively small sea land. Such cluster then spawns in mass, and their larvae through the prevailing ocean currents is dispersed widely and forcefully. There comes the secondary wave of the outbreak. Observations between the initiations of successive outbreaks can approximate the timing of primary outbreaks - an average of 15 to 17-year period. Prophetically, the next outbreak is predicted to come around 2026. [3]. The historic timeline together with the total population and intensity of the outbreaks are illustrated comprehensively in part A of Figure 2.5.

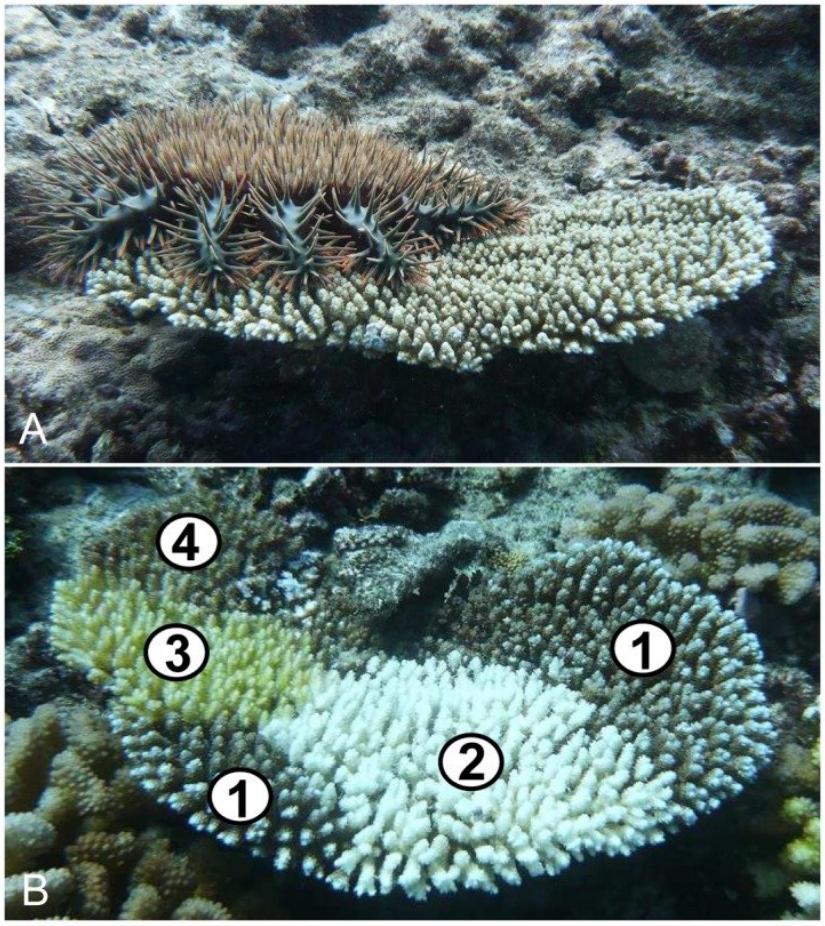

Figure 2.4: Feeding scars found on colonies upon which a COTS preys during the COTS outbreak. ©Photos Mohsen Kayal [27].
A: An observed COTS on a living coral.
B: A partially-killed coral after successive predating events by the COTS. Based on pigmented tissues, the coral's area is divided into different portions: (1) *live portion*; (2) *freshly-killed portion*(less than 1 day after the predation); (3) *recently-killed portion*, covered by algae and cyanobacteria (approximately 10 days after the predation); (4) *completely dead portion*, covered by turf algae (more than 3 weeks after the predation).

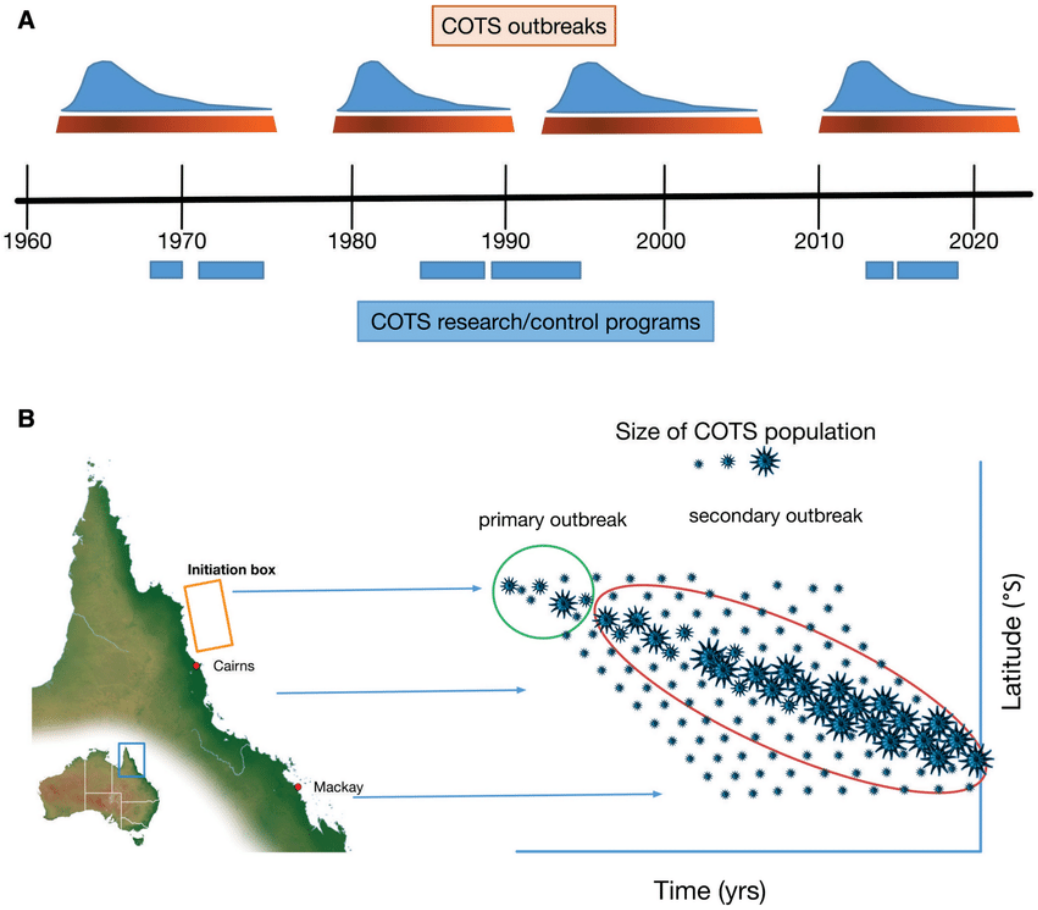

Figure 2.5: An overview of COTS outbreaks: timeline, duration, intensity, population and movement direction [3].
A: Historic timeline of the outbreaks together with their duration, the intensity and control programs/actions on the Reef:
- *Density curves* denote total population of COTS during each outbreak. An indicative timeline is projected for the latest and ongoing outbreak (2010).
- *Shaded orange bars* indicate the duration and intensity of each outbreak.
- *Blue bars* represents the duration of funding and control programs.
B: COTS populations on the Reef.
- *Initiation box*, locates mid-shelf reefs in the north-central (Wet Tropics) region where connectivity among the reefs is dense [1], denotes the starting point of the outbreaks.
- *Green circle* illustrates an estimation of COTS population in the primary stage of the outbreaks.
- *Red circle* illustrates an estimation of COTS population in the second phase of the outbreaks. Larvae spills southwards by ocean currents at a rate of 60 kilometres per year [1].

The outbreaks have narrowed the coverage of living corals substantially and weakened the habitat structure, damaged the diversity of reef-associated organisms and the overall resilience of the Reef. A multitude of factors contributing to COTS's growth and escalation is not fully concluded. Yet, after successive waves of efforts, several researchers have uncovered the main causes, namely biological traits of the COTS, environmental factors, favourably geographical conditions and human activities. Considering the environmental conditions, there is an abundance of food sources for COTS populations: high coral prey availability and land-based nutrient run-off entering the Reef lagoon after flooding events [9]. A declined number of the COTS natural predators owing to fishing pressure and poor water quality etc. is also the predominant factor; however, they can be mitigated through local management intervention. The Wet Tropics region of the Reef between Lizard Island and Cairns has relatively intensive connectivity among the reefs and hydrodynamic circulation patterns, which by all means accelerates the outbreaks. In essence, it takes years or even decades for the damaged coral reefs to recover, given that it is not exposed to other disturbances such as coral bleaching, severe tropical cyclones and flood events; otherwise, awfully persistent. [1].

2.2.2 Outbreaks monitoring

Different parties have been engaged in the effort to understand, monitor and control COTS outbreaks to an ecologically sustainable level, namely government agencies (Marine Park Authority and Queensland Government), research institutions and universities (James Cook University, University of Queensland, Central Queensland University, and Griffith University), and environmental organisations (Commonwealth Scientific and Industrial Research Organisation (CSIRO), the Australian Institute of Marine Science etc.). There exist extensive efforts to conduct targeted in-water assessment of COTS numbers by different survey techniques, listed as follows:

- **Eye on the Reef, Sightings Network reports** provide managers with an approximate number of COTS and their spatial locations observed in the Marine Park. This survey requires no survey equipment nor specialised training and the results are used to guide the more in-depth assessments.

- **Manta tow surveys** gives reef managers, marine park rangers and marine scientists the outbreak status with regards to an assessment of the reproductive-mature-COTS population and their feeding activities by assaying the visible scars. This survey requires complicated equipment and the snorkel divers are expected to undergo specialised

training. Specifically, a manta tow boat tows a trained observer and carry him at a constant speed through 100 to 1000 meters of reef habitats to collect information. This is the key method used for COTS and coral-cover surveillance. Generally effective, yet Manta tow survey has some limitations with regards to operational scalability, data resolution, reliability and traceability [32].

- **Scuba transect surveys** are particularly conducted by experienced observers, such as the scientists and reef managers, to collect thoroughly detailed assessments of COTS densities and body sizes of both juvenile and mature starfish. As the first and foremost prerequisite, the survey area (of around 100 m^2) must be pre-defined with its respective priority. Apparently, scuba transect is significantly time-intensive and it requires the most specialised knowledge and expertise. [1].

In short, current monitoring of the outbreaks on the Great Barrier Reef is conducted merely by humans and thus considerably prone to error. The survey results are limited and utterly dependent on the observer's perspective and experience. To that end, these techniques either underestimate or possibly overlook the major COTS populations when the undertaken transect is unrepresentative of the region [7]. Considering the gigantic area of the Reef, it requires enormous resources to generate a large-scale COTS distribution map, including experienced divers, ships, durable equipment and other conditions such as hindrance of weather, health and safety issues. All things considered, it calls for a cost-effective, safe and robust alternative to perform the autonomous COTS population monitoring. Such methods should involve two key capabilities: (1) autonomously collect the images from the reefs through deploying underwater imaging devices, and (2) perform robust COTS recognition [32]. Indeed, the first capability is achieved fairly easily by an Autonomous Underwater Vehicle (AUV), digital cameras and unmanned underwater vehicles (UUV) while robust texture detection of the COTS from underwater image videos is relatively an untouched area of research [7, 12]. In the long-term outlook of the Reef, AI-driven environmental survey is the potential for broad-scale surveillance of underwater habitats and it may remarkably improve the efficiency of the COTS control program by delivering accurate, timely and reliable assessments of reef-scale ecosystems. [32].

2.3 Related Work in COTS Object Detection

COTS detection is a subset of underwater object detection problems, which focuses on detecting the COTS from living corals. Historically, marine life in

general and underwater terrain-related studies have been conducted manually. There exist only a few vision-based studies in COTS detection, typically Local Binary Pattern model by Clement et al. (2005) [7] and Random Forest Classifier by Feras et al. (2015) [9]. This results from a variety of changing underwater conditions, including but not limited to turbid water state, light scarcity, messy undersea background, and diverse viewing angles (from the back, close-up and side view) which hinder the collection of high-quality underwater videos and images. Indeed, deep learning algorithms require a plethora amount of high-quality underwater videos and images to extract discriminant features and high-level abstraction. The existing databases are insufficient for training proper deep learning based COTS detector; hence a scarcity in related literature. From another angle, COTS recognition is closely related to other marine organisms (such as fish or coral) detection, which has been a relatively emerging and attractive area of study. This means that studying other research in similar domains would benefit substantially. The following section will review existing COTS detection works, together with different fish detectors built upon deep learning algorithms. Similar domain knowledge is considered under a pre-defined scope of the thesis, hence providing a solid foundation for deep learning based COTS model.

2.3.1 Local Binary Pattern (LBP)

In 2012, Clement et al. [7] laid a foundation in robust image detection of COTS by a texture-based classification procedure using Local Binary Pattern [7]. Clement and his co-researchers believed that COTS's color varied by their age, location and altitude at which the images were taken. Besides, the red wavelengths of light get absorbed much faster than the other shorter wavelengths (such as blue); for this reason, the underwater environment appears blue/green when the depth increases. To that end, color segmentation would be unreliable to recognise the starfish from survey images. In like manner, COTS are skillfully camouflaged and highly flexible; therefore, there exist different poses varying from flat, curled to partially hidden etc. [9]. In other words, shape-based matching is infeasible for this detection scheme. Instead, their thorns are particularly distinctive and recognisable; hence a potential and reliable distinguishing feature (texture).

By the Local Binary Pattern operator, 2D surface textures can be described by local spatial patterns and grey-scale contrast. These two complementary measures allow the LBP operator to label an image's pixels by comparing the threshold of neighbouring pixels with the center value and assigning a binary number as a result. The authors extended the original operator by employing *uniform patterns*, rooted in the fact that some binary

patterns appear more commonly than others. If a binary pattern contains at most two bitwise transitions (either from 0 to 1 or vice versa) when the bit pattern is traversed circularly, it is considered a uniform pattern (for example 01110000 and 11001111) ([37]. *Uniform patterns* reduces the feature vector's length and implements a simple rotation-invariant descriptor, on the whole provides sufficient information for a proper correlation when comparing with images [7, 37].

In the research paper, a total of 27 384x384 pixel textures were created from representative images, among which 12 were COTS and the rest were other marine surfaces and coral textures. When the texture database was generated, the dataset underwent an end-to-end image processing pipeline, details as below:

- **Top Hat filtering** was used to enhance images by highlighting shadowed regions and reducing major grey-scale shifts. This operation subtracted the original image from the result of performing a morphological closing operation on that original image. Specifically, a structuring element was used to erode the image and that element continues to dilate the resulting image afterward.

- **Grey-scale conversion** was done on image-level by subtracting a selected pixel from the value of each neighbouring pixel and thereafter normalizing by the addition of these differenced neighbouring pixels.

- **Local binary pattern** generated by labeling pixels with the binary numbers based on a specific threshold (a pixel was assigned 0 if its grey-scale subtracted value was less than 0 and otherwise).

- **Histograms** were created for texture comparison. First, an individual histogram was computed by a 50x50 pixel block, coupled with an overlap region of 200 pixels. This 450x450-pixel block allowed better recognition of the COTS area, especially around their legs where it could change from COTS to coral and then vice versa. This operation looped over different blocks of the image and all the histograms were summed up to create 9 bins.

- **Log-likelihood measure** was performed on image blocks to find the best matched texture (coral, sand or COTS) by comparing against the database texture.

- **Count number of detected COTS**

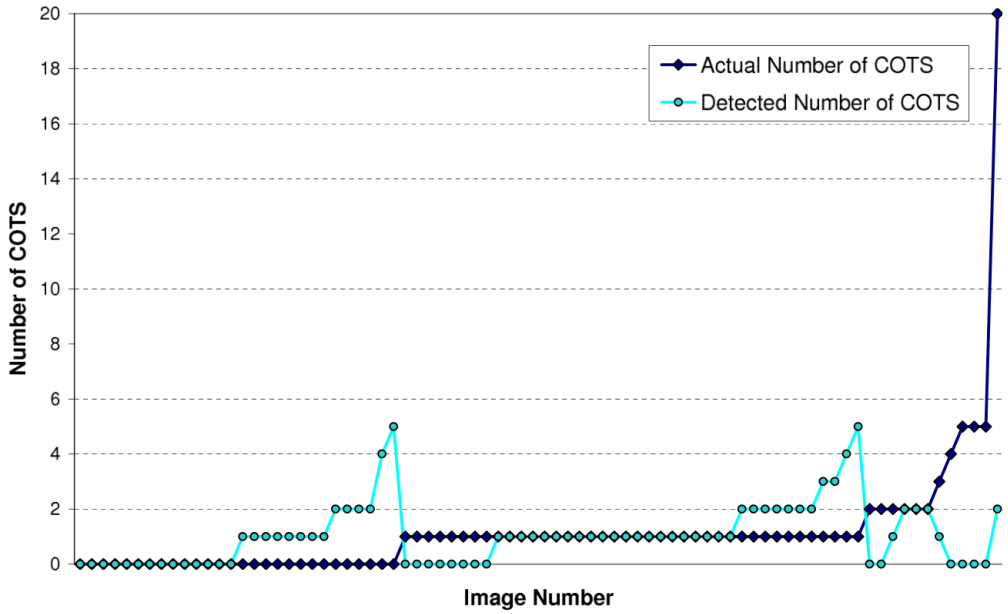

Figure 2.6: Number of COTS detected and counted for all images. Figure is derived directly from the original research [7].

Running on 80 images from the Australian Institute of Marine Science (AIMS), the algorithm accurately detected the presence and non-presence of COTS in 65% of the images, Figure 2.6. The starfish were counted correctly in 48% of the images. However, LBP falsely classified non-starfish at the rate of 31%. After analyzing false-positive cases, the authors highlighted a couple of major conditions. At higher altitudes, the COTS and its texture were relatively small compared to the overall image area; thus a higher chance of misclassification. Besides, straight thorns were the key discriminant of this technique but they failed to differentiate COTS from shelf or stag coral. That said, image quality was vital to ensure clear and sharp thorns textures. Lastly, interconnected COTS could be correctly identified; yet the counting was troublesome when the COTS were touching each other. All in all, the research not only provided a preliminary investigation of the Local Binary Pattern method for texture mapping and correlation of COTS from underwater images but also posed different challenges to open up the improvement ideas for future research. [7].

2.3.2 Random-Forest based classifier (RFC)

The second research about COTS detection and tracking was released by Feras D., Matthew D. and Peter C. in 2015. They introduced the use of a robotic system capable of COTS monitoring instead of manual interference from humans. The traditional method has proved its inefficiency while also encountering several barriers such as operational expensiveness of resources (divers and equipment), safety concerns (limited diving time), restricted daylight hours and bad sea conditions. Overcoming the challenges of the conventional method, the vision-based robotic arm extended to AUV device might present the next advancement in COTS detection and monitoring. As described in the paper, random-forest based classifier was embedded in a particle filter tracker of Baxter robot's hand moving camera (simulating an underwater vehicle), which was capable of robust COTS tracking over a complex coral reef environment. [9].

In like manner, Feras and his group saw the limitations of prior research in using shape and color as the major descriptors for COTS. They also assumed that thorns of the starfish as a texture-based feature alone was insufficient since corals structures upon which the COTS live could exhibit a relatively similar texture to the COTS. Yet, long thorns could provide strong edge information. All things considered, a feature vector was created by combining Local Binary Patterns (for texture) and Histogram of Oriented Gradient (HoG) (for edge information). The former technique is already discussed in the previous section. With the latter method, the feature was computed by firstly dividing the detecting window into equal cells, each of which had a histogram generated from a weighted vote by each pixel based on the magnitude and orientation of its intensity gradient. [9]. Regarding the samples, 3,157 images were used to train Random Forest Classifier algorithm. 30% were positive examples with full-body COTS captured or partially captured (close-up view of a single-arm). Five folds of cross-validation were performed with the best parameters from the grid-search execution.

The paper did not focus solely on detecting COTS from images but also on tracking a fixed point on their bodies to conduct the injection of biological agents. For this reason, Random Forest Classifier (recognizing scheme) was combined with Particle Filter (constant tracking while the camera moves over the scenes). Specifically, the predicted class probability from Random Forest Classifier was used as an observation measurement to weigh the particles for updating the Particle Filter. Besides, the research group chose F1-score to evaluate the performance of RFC. 0.98 F1-score was achieved for No-COTS class and 0.96 for COTS. To assess the detection system based on height and speed, a Baxter robot carried a camera by its left hand at five different speeds

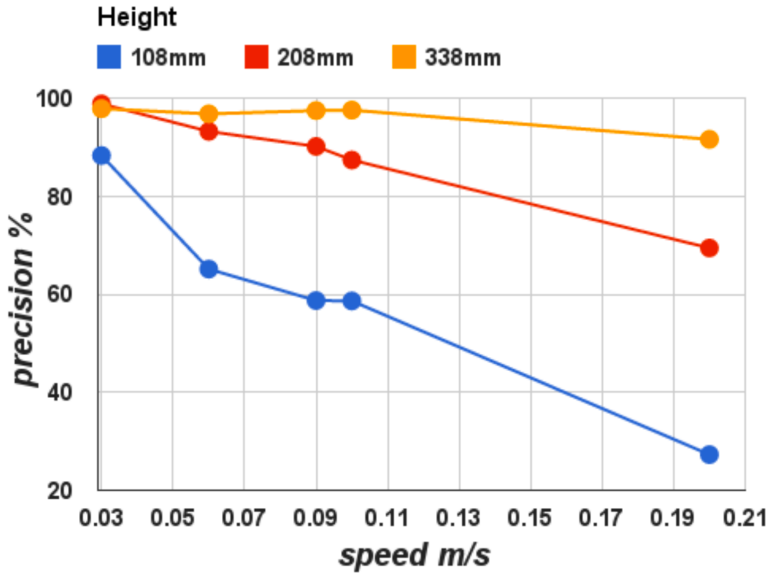

Figure 2.7: The average precision achieved by the detector at different speeds and heights. Figure is derived directly from the original research [9].

(0.03, 0.04, 0.09, 0.1 and 0.2 m/s) and at three heights (108, 208 and 338 mm above the simulated scene). This scene referred to five high-resolution images taken of COTS in different reef environments, printed as posters and laid flat on the table.

Precision was an essential metric for this evaluation since the robot would attempt to inject false positives rather than the COTS. Figure 2.7 shows the average precision resulting from 15 different combinations of speed and height over all the simulated scenes. The more accelerated the speed of the camera, the lower the precision, especially when the camera was closer to the reef. In simple terms, at lower altitudes, motion blurs and latency increases, hence shrinking the accuracy of the detector. Furthermore, as the altitude declines, the reduction of texture in the sample image would negatively influence the precision score. In conclusion, the results demonstrated the robustness of the algorithm in COTS detection, which proposes a promising future for AUV devices with robotic arms to inject and control the COTS populations to a sustainable level.

2.3.3 Deep Learning

To the best of our knowledge and exploring efforts, no research papers were found for COTS detection using deep learning algorithms. This thesis, by

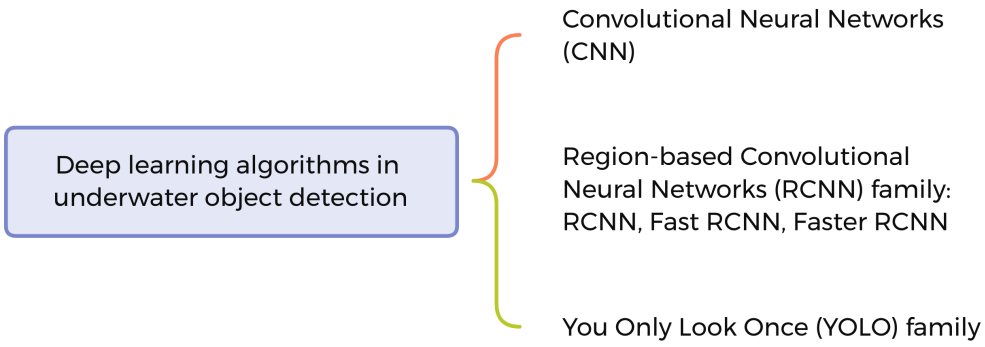

Figure 2.8: Typical deep learning algorithms in underwater object detection [12].

the time of its release, will indeed provide the officially first deep learning-based algorithm to detect COTS accurately, efficiently and in near real-time. Comparatively not much underwater surveillance marine object detection technologies are exhaustively explored due to momentous challenges like light scattering and absorption occurring underneath the sea [12]. Conventionally, underwater object detection is conducted manually or by statistical analysis and ocean model simulation. These methods are highly dependent on the availability of optical characteristics, thus demonstrating inefficiency, low accuracy and inability to process large underwater datasets. Deep learning, on the other hand, can process immense underwater data by hierarchical feature extraction through deep and big networks.

The advent of deep learning has enabled the researchers to solve many problems such as critical underwater habitat monitoring and protection, sub-aquatic species detection and recognition, emergency rescue, undersea disaster mitigation and prevention etc. [9]. That said, this literature review significantly focuses on the existing underwater object detection using deep learning techniques. Convolutional Neural Networks are the cornerstone of many current renowned algorithms. However, CNN is computationally time-consuming and inappropriate to be employed in real-time object detection. This section provides an overview of existing deep learning methods employed in underwater sub-aquatic organism detection (Figure 2.8), together with a proper comparison. The advantages and shortcomings of these methods are demonstrated, which reveals the rationales behind the chosen algorithm of this thesis - You Only Look Once (YOLO).

2.3.3.1 Convolutional Neural Networks (CNN)

Briefly mentioned in the above section, Convolution Neural Networks, also known as *convnets*, are now used universally in computer vision applications [6]. Convolution operates over *input feature maps* with *height* and *width* as spatial axes and *depth* as channel axis. For an RGB image, there are three channels: red, green, and blue while a monotonic image only poses a single-dimensional depth (gray intensity). The convolution operation extracts *patches* from the input feature map (dot product with the kernel) and applies the same transformation to all input patches. There constructs an *output feature map* which remains as a rank-3 tensor with an arbitrary depth. Specifically, different channels no longer characterise specific colors in the RGB scheme but *filters*, encoding specific aspects/traits of input data. [6]. Apart from convolution, pooling is also a basic component of *convnet*. Pooling is introduced between successive convolution layers to aggressively down-sample the feature maps. To put it differently, the spatial size of the image is reduced while the depth remains unaltered. Max pooling is the most commonly used operator which transforms local patches via a *max* tensor operation instead of a linear transformation as convolution [6, 12, 15].

CNN architecture has been employed exclusively in many research works for undersea object detection. Notably, Elawady et al. [11] used supervised CNN to classify coral on datasets from Heriot-Watt University's Atlantic Deep-Sea and University of California San Diego's Moorea Labeled Corals. His group extracted shape and texture features by computing Weber Local Descriptor (WLD), Phase Congruency (PC), and Zero Component Analysis (ZCA) Whitening. These feature-based maps were supplementary descriptors besides color channels of the coral input maps. In 2019, Suxia et. al [8] deployed CNN to detect a fish in a blurry underwater environment (data collected from the Gulf of Mexico). Their target was to embed the detector system into the Autonomous Underwater Vehicle design with all possible optimizations to enable the real-time capability. The authors employed the CNN model and back-propagated the gradient of the refined loss function to update the parameters of the network. In addition, different transformations augmented the original data to obtain more learning resources, including rotation, scale, crop and mirror symmetry. The network was simplified by the drop-out algorithm to mitigate overfitting. Above all, the experimental results were promising: high accuracy and low processing time. Thus, the proposed model could be developed further for AUV implementation. [8].

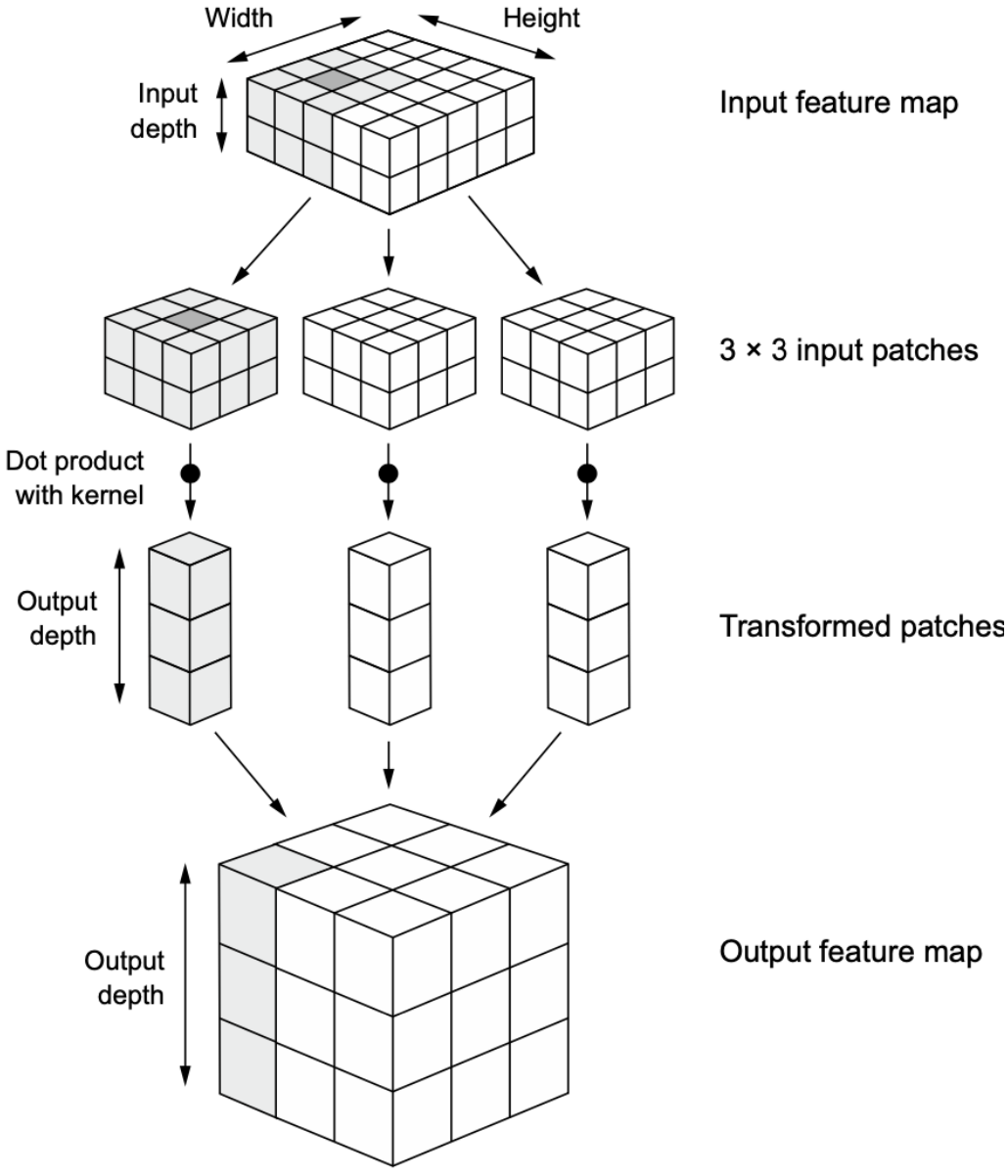

Figure 2.9: How convolution works. Figure is derived directly from *Deep Learning with Python*, Second Edition by François Chollet [6]

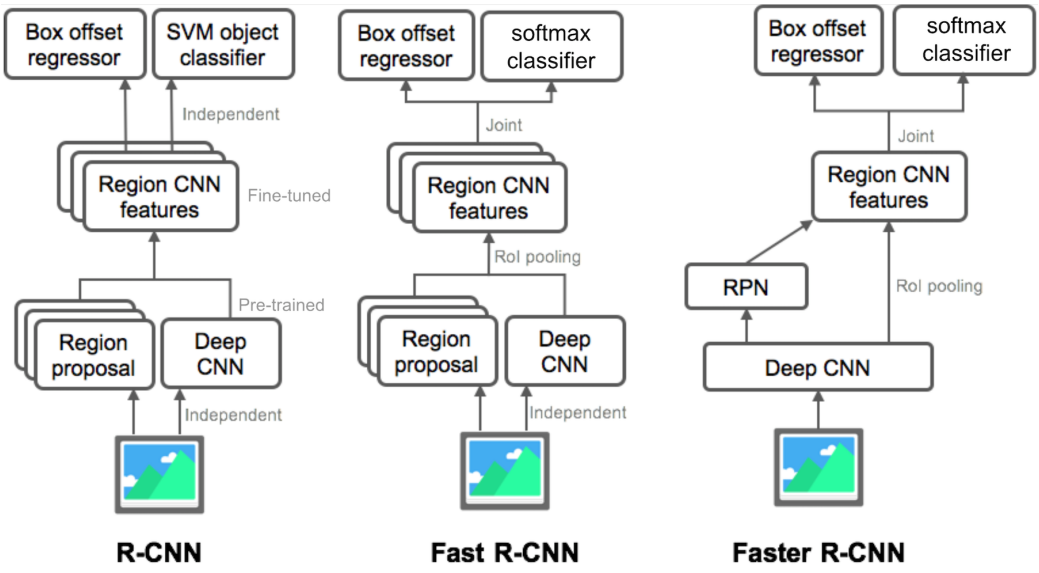

Figure 2.10: Summary of models in the R-CNN family. Figure is derived directly from the original article [59]

2.3.3.2 Region-based Convolutional Neural Networks

Regional Convolutional Neural Network (R-CNN), presented by Girshick et al. in 2014, was the combination of the CNN model with Region Proposal Network [14]. Specifically, among bounding boxes of the target object's possible appearance in the underwater images, R-CNN carried out a *selective survey* to identify a manageable number of boxes (approximate 2K candidates/image), the so-called regions or region of interest (ROI) [59]. Regions were formed based on the resemblance of texture, color, shape and size of the objects [12]. It then extracted features from each region independently using CNN operation. The features obtained by each region proposal were subsequently used to classify the target and predict the bounding box. Single-binary Support Vector Machine (SVM) was trained for each class independently to classify background and object. Following that, a linear regression model was trained to produce tighter bounding boxes for every identified object in the underwater images, thus reducing localization errors [59]. More details are put in Figure 2.10. R-CNN might be impractical for processing a massive dataset as the model workflow involves tremendous work:

- Run selective search to propose $\sim$ 2000 ROIs for every image.
- Feature extraction by CNN for every image region, which means N images * 2000 ROIs.

- Three models, namely CNN, binary SVM and linear regression are performed separately without much shared computation.

To reduce the computational time of R-CNN, Girshick published an upgraded version of R-CNN, called Fast R-CNN in 2015 [12, 36]. Remarkably, one CNN forward propagation passed over the entire image instead of 2000 runs to extract feature vectors. All the ROIs (underwater object localization) were obtained at the same time. The ROI pooling layer was then used to ensure that all the regions were of equal size. After this step, every region was fed into a Fully Connected Network (FCN) layer coupled with a softmax layer. The feature vectors were used for learning the object classifier and bounding box regressor simultaneously. Notably, Girshick introduced a multi-task loss function that considered both the classification and localization error [36]. The model branched out into two outputs: discrete probability distribution of each ROI from a SoftMax estimator and offsets prediction relative to the original bounding box for each class. [59]. The architecture of CNN allowed a multi-task learning manner. Three independent models were unified into one jointly trained framework: feature extraction from proposed regions, classification, and bounding boxes generation simultaneously. By all means, computation sharing sped up the workflow substantially. Fast R-CNN was much faster as compared to the R-CNN model; nevertheless, the improvement was still insignificant when processing an enormous dataset as the region proposals were computed separately by selective search. [12].

Fast Accurate Fish Detection and Recognition of Underwater Images with Fast R-CNN paper by Xiu et al. (2015) [30] is among the rare research using Fast R-CNN for marine organisms surveillance. Training and testing images were derived from the video dataset LifeCLEF Fish of Fish4Knowledge video repository, containing 24,872 fish images from Taiwan coral reefs. The research group built a model based on Fast R-CNN and compared the method to other detection approaches namely Deformable Part Models (DPM) and R-CNN. Fast R-CNN based-model contained five convolutional layers, an ROI pooling layer, two Fully Connected layers and two siblings layers (an FC layer underneath a softmax layer). Normalization and max-pooling layers were added after every of the fist two convolutional layers. ReLu activation function was applied to the output of every convolutional or FC layer. AlexNet was pre-trained and adapted with three major modifications: (1) the network takes two inputs: a batch of N images and a list of ROIs, and selective search generates 2000 region proposals; (2) the last pooling layer was replaced with an ROI pooling layer to pool all ROIs into fixed-size feature maps; (3) the finally FC layer was replaced with two siblings layers, one of which output softmax probabilities over 12 fish classes and a back-

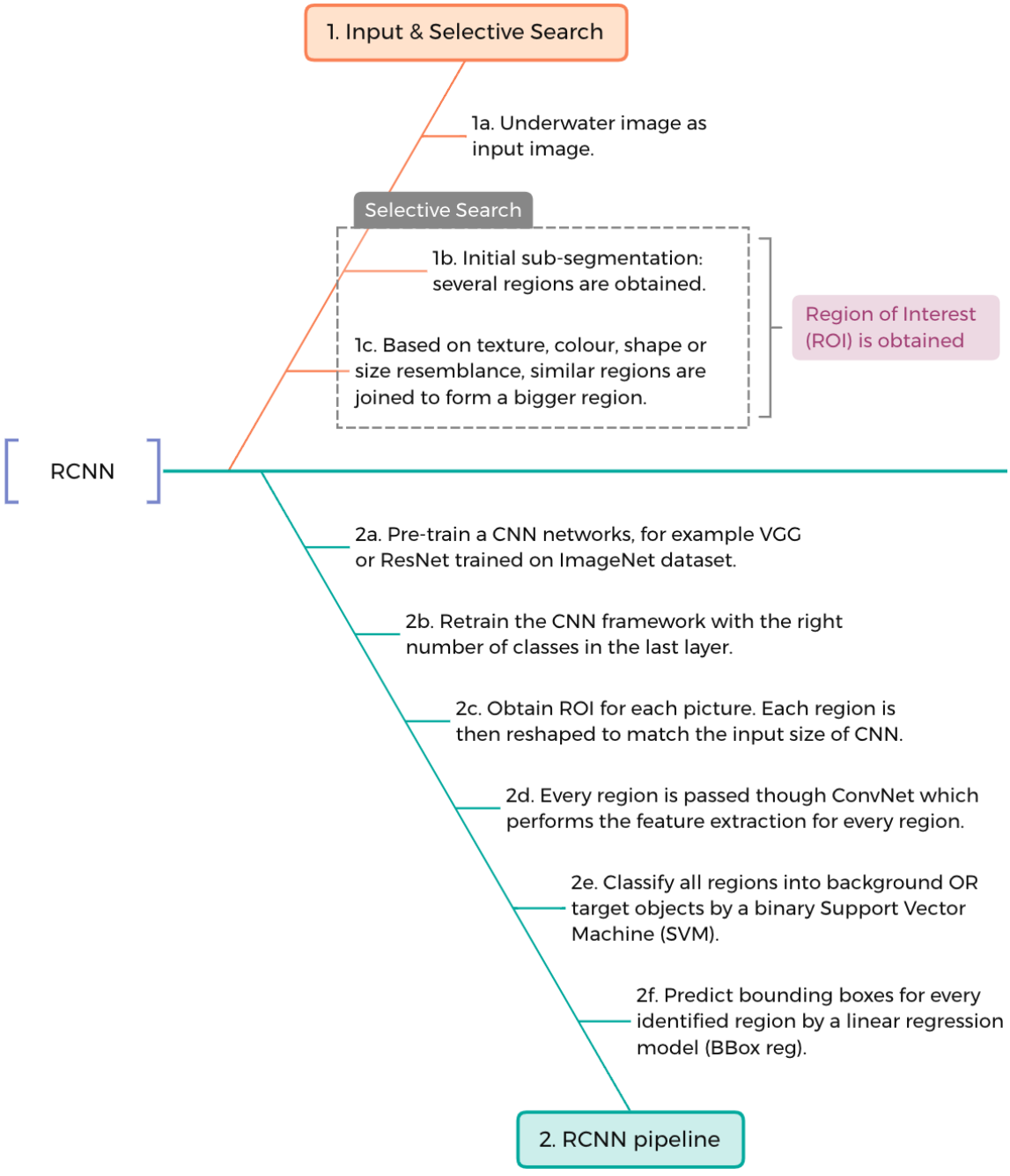

Figure 2.11: Underwater Object Detection using R-CNN [12].

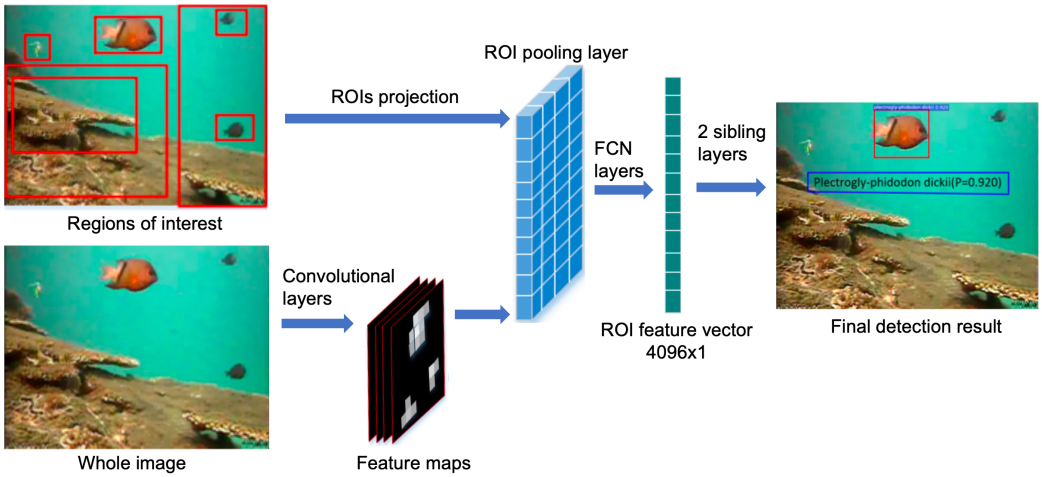

Figure 2.12: The overall architecture of automatic fish detection and recognition system using Fast R-CNN. Figure is derived directly from the original research [30].

Metrics	R-CNN	Fast R-CNN
Training time (h)	67	4
Training speedup	1x	16.75x
Testing time (second/image)	24.945	0.311
Testing speedup	1x	80.2x
mAP (%)	81.2	81.4

Table 2.2: Runtime comparison between R-CNN and Fast R-CNN [30].

ground class while the other output bounding-box coordinates for all detected fishes. A simple illustration was provided in Figure 2.12. Considering the performance, Fast R-CNN improved mean average precision (mAP) by 11.2% relative to 70.2% achieved from the DPM method and detected fish from a single image 80 times faster than R-CNN (Table 2.2). On the whole, Fast R-CNN demonstrated superior performance over the DPM method in fish detection and substantially sped up the prior version in the RCNN family. [30].

An intuitive improvement idea from Fast R-CNN architecture is to integrate the ROIs selection into the same detection network. Accordingly, Faster R-CNN (Ren et al., late 2015) - a fully-convolutional network came into reality [41]. The framework constructed a unified model, utilizing a novel Region Proposal Network (RPN) for region proposal generation and fast R-CNN for classification and bounding box regression. Compared to Fast RCNN, Faster

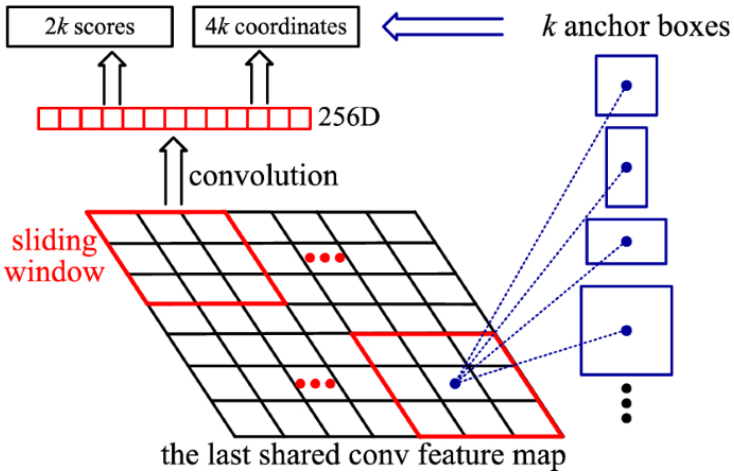

Figure 2.13: Region proposal network (RPN) by Ren et al. The research group introduced a concept of *anchor boxes* in multi-reference object detection. These reference boxes are considered per window location, which facilitates the detection of objects with different shapes and sizes [36, 41]

RCNN used Region Proposal Network to generate ROIs instead of a selective survey. The RPN fully optimized region proposal generation by using an FCN that could utilize GPU parallelization. It predicted objectness scores together with bounds for the potential object regions. [36] (Figure 2.13). Both RPN and Fast R-CNN shared the same convolutional input feature maps which were extracted from the base convolutional network (for example VGG16). Although Faster R-CNN has gained great achievements in object detection and classification lately, there still exist some shortcomings. It does not see the entire image at once but different portions of the image in sequence. That is, it requires several passes through one image to detect all objects (classes). [12].

Being inspired by the algorithmic changes in the R-CNN family, Xiu et al. (2016) [29] continued their previous work [30] in fish detection and classification using Faster-RCNN. This time, proposal generation by Region Proposal Network shared convolutional features with the following fish detection and recognition network, as depicted in Figure 2.14. Thus, the new architecture pushed the detection system toward a real-time manner. As clearly seen from Table 2.3, detection speedup was improved significantly when comparing different R-CNN-based algorithms. Besides, the faster region-based detector achieved frame rate of 9.8 fps which demonstrated its reliability in moving object detection in underwater videos. Apart from speed and computation optimisation achieved by Faster R-CNN, higher detection accuracy based on

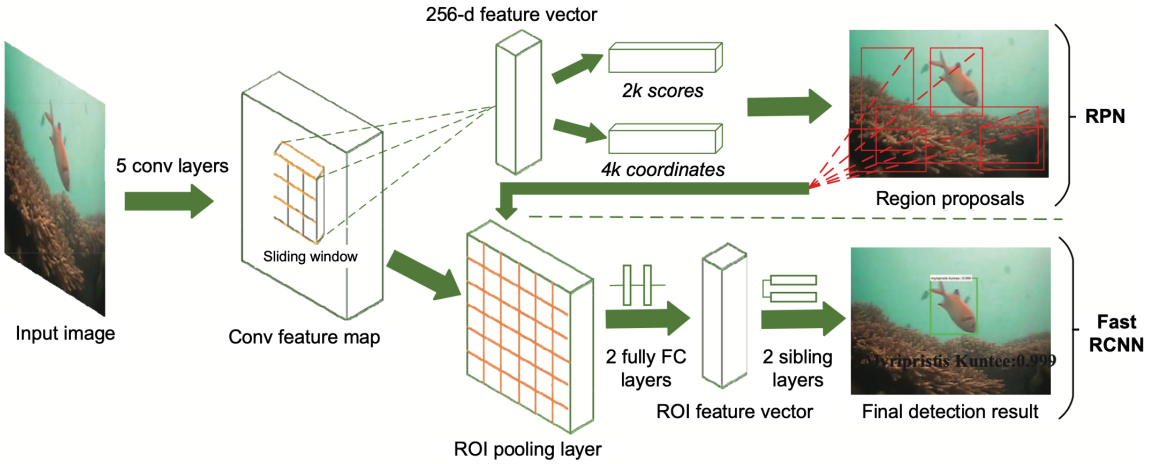

Figure 2.14: Overall architecture of Faster R-CNN fish detector by Xiu et al. [29]. Figure is derived directly from the original research.

Metrics	R-CNN	Fast R-CNN	Faster R-CNN
Detection time (s/image)	24.945	0.311	0.102
Detection speedup	1x	80.2x	244.5x
mAP (%)	81.2	81.4	82.7

Table 2.3: Runtime comparison between R-CNN, Fast R-CNN and Faster R-CNN [30].

temporal information from underwater videos would be further discussed and experimented with.

There exist other papers about marine organism detection and recognition based on Faster R-CNN. In a different manner than the above examples, Huang et al. 2019 [66] researched the efficiency of Faster R-CNN model by enhancing the underwater images against different marine turbulence. The group worked on 2000 labeled images of sea cucumber, sea urchin and scallop provided by the National Natural Science Foundation of China (NSFC). They performed data augmentation including perspective transformation, illumination synthesis and marine turbulence simulation. The model workflow goes as (1) training an end-to-end RPN by backpropagation and stochastic gradient descent (SGD), (2) using the proposal regions from step one to train a separate detection network Fast R-CNN which was initialized with VGG16 pre-trained ImageNet model, (3) sharing the detection network's convolutional layers with RPN and fine-tuning the layers unique to RPN, (4) same as step three but fine-tuned the layers unique to Fast R-CNN. The results prove the robustness of the proposed methods when compared with the orig-

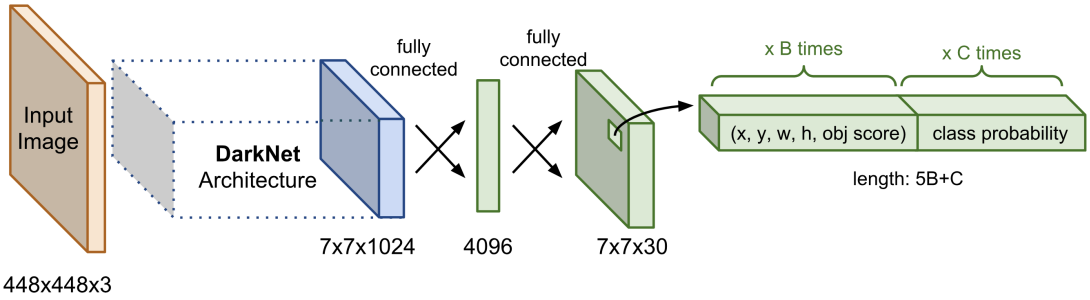

Figure 2.15: YOLO architecture. The base model is similar to GoogLeNet [53] with an inception module replaced by 1x1 and 3x3 conv layers. The prediction vector is produced by 2 FC layers over the whole convolutional feature map. Figure is derived directly from the original article [60]

inal dataset. The Faster R-CNN model showed great performance on the whole dataset (augmented plus original), which was potential for marine organism detection and recognition. [66].

2.3.3.3 You Only Look Once (YOLO)

You Only Look Once (YOLO) model by Redmon et al., 2016 [38] was the very first attempt to build a one-stage detector with real-time property [60]. YOLO does not undergo region proposal stage but it looks at the whole image at once (Figure 2.15). That is, the algorithm needs only one forward propagation via a neural network to predict bounding box coordinates and different class probabilities associated with each box (as depicted in YOLO workflow). Such unified architecture is extremely fast in performance [38]. With a single look, YOLO sees the full image and encodes contextual information about classes according to its authors, unlike region proposal method. Henceforth, YOLO detects significantly less background false-positive than Fast R-CNN [36, 38]. Through different experimental results, YOLO outperforms and proves its generalization property, implying its object representation are highly generalized. Yet, YOLO does not excel at extremely small object detection and localization, especially when they are clustered [12, 36]. So far, YOLO framework has five versions (from YOLOv1 to YOLOv5), each of which improvises upon its prior releases.

Since the initial release, YOLO model has continued to evolve and become the de factor paradigm in object detection. In 2017, Redmon and Farhadi released an enhanced version - YOLOv2/YOLO9000. YOLO9000 was a variant of YOLOv2 with the training of COCO dataset combined with the top

YOLOv1 Workflow
1. Pre-train a CNN network on image classification task.
2. Split an image into $S \times S$ cells. If an object's center falls into a cell, that cell is *"responsible"* for predicting:

- **Bounding boxes coordinates** - defined by a tuple of (center x-coord, center y-coord, width, height) $-$ (x, y, w, h) which are normalized by the image width and height, x and y are set to be offset of a cell location.
- **Confidence score** - the likelihood that the cell contains an object, defined by $Pr(object) \times IoU(pred, truth)$ where Pr = probability and IoU = Intersection over Union.
- **Probability of object class conditioned on the existence of an object in the bounding box** - defined by Pr*(object belongs to* C_i *|containing an object)*, class $C_i, i = 1, ..., K$.

3. Modify the last layer of the pre-trained CNN to output a prediction tensor of size $S \times S \times (5B + K)$.

9000 classes from ImageNet [39]. Various modifications were applied on top of YOLO architecture, most notably including:

- ***Batch normalization*** - on all convolutional layers resulted in substantial convergence improvement [12, 57, 60]. In specific, batch normalization reduced the offset of unit values in the hidden layer; hence stablizing the neural network.
- ***High-resolution input classifier*** - the base model was fine-tuned with high resolution images (size of 448×448 instead of 224×224 for about 10 epochs on dataset ImageNet) to improve the detection performance (roughly 5% increase in mAP).
- ***Convolutional anchor box detection*** - previously, fully connected layers over the whole feature map is used to predict *bounding box position*. In YOLOv2, the prediction of spatial locations and class probabilities were decoupled, the former of which was predicted in the form of *anchor boxes* by *convolutional layers*.
- ***Dimension cluster*** - YOLOv2 run *k-mean clustering* to search for priors on anchor box dimensions, rather than selected manually as in

faster R-CNN architecture. Anchor boxes generated by *k-mean clustering* led to better average Intersection over Union (IoU) scores on the condition that the priors were independent of box size [12, 60].

- ***Lightweight base model*** - Darknet-19, consisting of 19 conv and 5 max pooling layers, was adopted as the base model, thus faster processing and better performance [12].

After a number of iterative improvements, Redmon and Farhadi published YOLOv3 in 2018, which has become one of the most readable papers in computer vision. The model contained two main components: a backbone feature extractor based on Darknet53 and a detection block for bounding box localization and classification [26]. To mitigate the vanishing gradient problem of a very deep network and preserve fine-grained features for small objects, residual skip connection is applied to the model, coupled with upsampling and concatenation method [65]. YOLOv3 was improved upon the previous version with some inspirations from recent advancements in the object detection world, involving:

- ***Logistic regression for confidence scores*** - YOLOv3 predicted *confidence score* for each bounding box by *logistic regression* while prior versions used squared errors summation. Multi-label classification was performed by multiple independent logistic classifiers. Sotfmax layer was no longer used, which was helpful especially in multi-label image where all labels were not mutually exclusive (e.g. person and woman are overlapping labels). [12, 40, 60].

- ***Hybrid base model, Darknet53*** - YOLOv3 used 3×3 and 1×1 CNN layers in succession, just like the original darknet architecture, but added residual blocks. Subsequently, the network consisted of 53 CNN layers, hence the name *Darknet53*.

- ***Multi-scale prediction*** - being inspired by feature pyramid networks, YOLOv3 added several convolution layers after the base feature extractor and the last layer made a prediction at three different scales (suitable for large, medium and small objects relative to the image size), encoding boundary box, class predictions and objectness [12, 26].

The original researcher stepped away from the field after YOLOv3; however, there are emerging innovators such as Alexey Bochkovskiy (YOLOv4), Glenn Jocher (YOLOv5) and Baidu (PP-YOLO) etc. There exist different improved algorithms, not within the scope of this thesis, including YOLOX, YOLOS (You Only Look at One Sequence), Scaled-YOLOv4, YOLOR (You

Only Learn Once Representation) etc. Released in April 2020, YOLOv4 by Alexey Bochkovskiy was the next version in YOLO family without being authored by Joseph Redmon. The main focuses were speed enhancement for performing operations of a neural network and optimization for parallel computations. Notable difference from YOLOv3 was the addition of *Spatial Pyramid Pooling (SPP)* [17] component for the more accurate bounding box localization. YOLOv4 had three main stages: a feature extractor Cross-Stage-Partial-Connections (CSP) Darknet53, a neck that connected the backbone to the head with SPP and PANet path-aggregation network [33], and a head that was identical to YOLOv3 [26].

One month later, Glenn Jocher - researcher and CEO of Ultralytics LLC published YOLOv5 which was written purely in Python programming language instead of C as in the previous versions. The model was lightweight and highly suitable for embedded devices in real-time object detection. There exist four architectures in YOLOv5, namely YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x which were differentiated by the amount of feature extraction modules and convolution kernels in a specific location of the network [65]. Indeed, YOLOv5s and YOLOv5m were the preset simplified model while YOLOv5l was the benchmark and YOLOv5x was the extended model [51]. There was no official paper released by Glenn, just a public repository instead. Yet, major enhancements could be summarized as follows:

- ***PyTorch translation*** - the whole Darknet research framework was translated into PyTorch framework; which was highly integratable with IoT devices [10, 50].

- ***Data augmentation*** - the introduction of *mosaic data enhancement* has well addressed the "tiny object problem". The technique includes splicing four random pictures by arbitrary scaling, cropping and arrange them; thus exposing small targets better for feature extraction stage (more details are discussed in section 4.3).

- ***Auto-learning anchor boxes*** - Glenn developed the idea of deriving anchor boxes dimension from the distribution of bounding boxes in the custom dataset by K-means and generic learning algorithm. Thereby, YOLOv5 anchor boxes were automatically learned from our custom dataset. [50].

Regarding the architecture, YOLOv5 is constituted of three main components: backbone, neck and head (Figure 2.16). Cross Stage Partial Network (CSP) and Spatial Pyramid Pooling (SPP) are used as the backbone to extract and aggregate image features at different granularities. The model

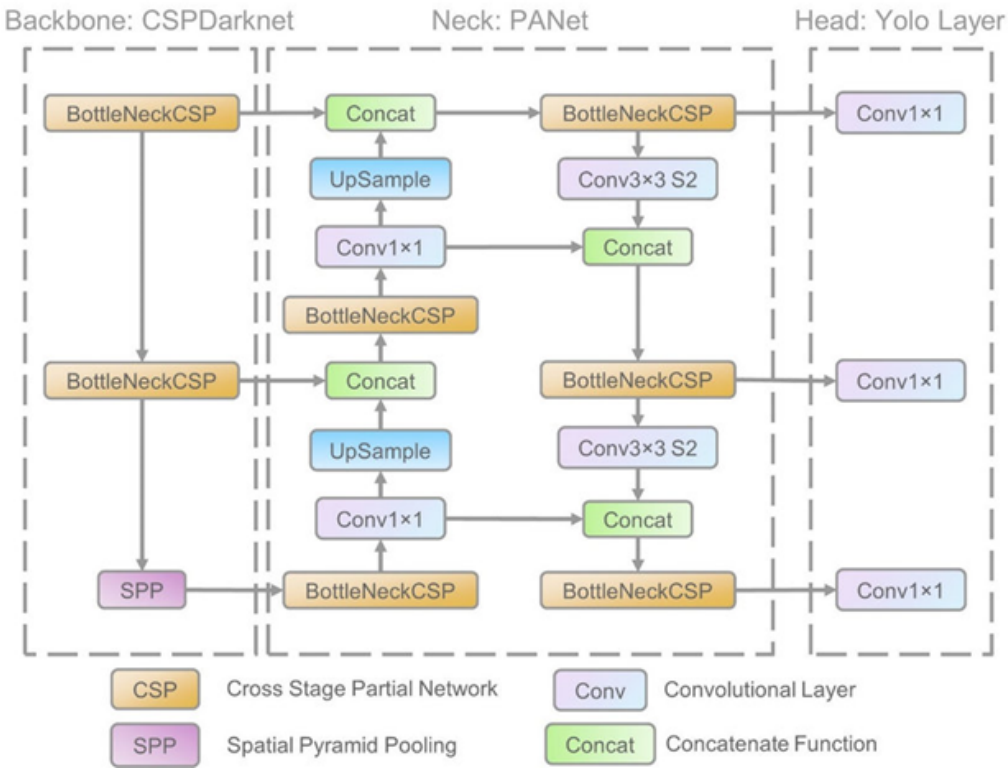

Figure 2.16: YOLOv5 architecture. Figure is derived directly from the original article [61]

adopts Feature Pyramid Network (FPN) + Path Aggregation Network (PANet) structure (similar to YOLOv4) to boost the information flow. Solid semantic features are conveyed from top to bottom while positioning features are robustly conveyed reversely up [61]. Primarily, feature pyramids enhance generalization on object scaling. In simple terms, the same objects can be identified with different sizes and scales. Lastly, the head consumes features from the neck and performs detection task. It applies the anchor boxes (learned from fitting to ground-truth bounding boxes in the custom dataset) on features and generates the output vectors with class probabilities, objectiveness score and bounding boxes dimensions.

YOLO family is suitable for processing video in real-time and highly reliable for detecting fish in severely noise, high attenuation of lights and hazy underwater images [26, 65]. There appears an increasingly more amount of studies focusing on an end to end underwater object detection models which address specific problems of marine organisms in complex living habitats. Remarkably, Sung et al. [52] (2017) employed YOLO model to detect and

classify fish using 829 underwater images. They achieved 93% classification precision on 100 images of fish species and 16.7 frames per second of fish detection [12, 52]. YOLO model has indeed outperformed other fish detectors using sliding window algorithm and classifier trained with a histogram of oriented gradient features and SVM. Jalah et al. (2020) [23] proposed a unified approach to detect and classify fish from underwater videos (LifeCLEF 2015 benchmark from Fish4Knowledge repository and a dataset collected by The University of Western Australia (UWA). Originally, YOLO model was employed to capture static and clearly visible fish. To enable the detection of freely moving fish or skillfully camouflaged fish, the authors used temporal information acquired via Gaussian mixture models and optical flow. This unified approach achieved high F-scores of 95.47% and 91.2%, and accuracy of 91.64% and 79.8% for each dataset respectively. The results have become benchmark performance for these datasets for the time being. [23].

Among many qualified studies, a recent publication *Automated detection, classification and counting of fish in fish passages with deep learning* by Kandimalla et al. (2022) is worth noticing. The authors applied YOLO algorithm which accurately detected and classified eight species of fish from a public high-resolution DIDSON dataset (video from both cameras and imaging sonar) captured from the Ocqueoc River in Michigan, USA. This study contributed to the Ocean Aware project, funded by Canada's Ocean Supercluster which aimed to build commercialized solutions for tracking fish health without the use of traditional tags. The researchers used convolution weights pre-trained from ImageNet dataset with 80 classes. The model was adapted accordingly to the specific parameters such as the number of output classes, batch size, and image dimensions etc. Additionally, image augmentation during training was performed, which enhanced the result substantially. The highest mAP achieved by YOLOv3 is 0.73 which is 0.11 higher than Mask R-CNN counterpart at an IoU threshold of 0.4. Besides, YOLOv3 was 3x faster than the other method in terms of processing image frames per second. These experimental results demonstrated great feasibility for deploying YOLO model into embedded devices which requires real-time processing. [26].

In 2021, Wang et al. published a paper comparing four sizes of YOLOv5 and also together with other detection algorithms in detecting underwater objects (holothurian, echinus, scallop, starfish and waterweeds) from the Underwater Robot Professional Contest (URPC^2) dataset [58]. The experimental results showed that YOLOv5 achieved the highest accuracy and efficiency compared to other two-stage detectors. The ranking was YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x in ascending order regarding mAP and in reverse order in terms of efficiency (obviously, the larger size, the more

parameters). Under a similar context, Zhong et al. (2022) [65] recently published a comparison of YOLOv3, YOLOv5, and YOLOR models in detecting fish and turtle in the coral reef. YOLOv5 indeed was able to achieve high precision with the most efficiency (the average execution time was only 13.3ms, nearly half of the others'). By this, YOLOv5 algorithm is truly promising for developing real-time underwater object detection.

This literature review has extensively covered the following areas: the outbreak of COTS in the Great Barrier Reef and its severe consequences, current methods to prevent the outbreak, existing studies of COTS detection based on underwater images, the feasibility of building a deep learning-based detector rooting from similar domain knowledge (fish and coral detection). Particularly, we focused on works related to deep learning methods which would be the main theme of this thesis. By this, we accumulate sufficient background knowledge of current technologies for underwater object detection problem. Apparently, the hybrid approaches (the ones with architectures similar to R-CNN) may be daunting in performance optimisation as region proposals and classification take place in two completely separate stages [36]. On the contrary, this thesis mainly focuses on an end-to-end deep learning based detection framework - YOLO. We are confident enough with a comprehensive understanding and specialized expertise of various techniques to build a reliable COTS detector based on YOLOv5. CSIRO Crown-of-Thorns Starfish Detection Dataset by Lin et al. [32] is a suitable testbed for constructing and evaluating YOLO based detector. All things considered, we can now proceed to problem formulation, methodology, implementation and evaluation phases of this thesis in the upcoming chapters.

Chapter 3

Problem Formulation

The literature review has given an intensive understanding of the current object detection problem, which involves only one explicit foreground class of interest. Different recent approaches, reviewed in Section 2.3, shed light on the research area, albeit the scarcity of underwater images. Now, it's time to discuss our thesis research problem in great details. This chapter covers the problem-solution frame and it is organized into three main sections. Firstly, we provide background of the dataset used to build the COTS detector. Some findings and observations from the exploratory data analysis (EDA) are presented to give a thorough understanding of the given input images. On top of that, comparing this dataset against the benchmark datasets (reviewed in Section 2.1.3) to extract explicit characteristics helps formulate the problem much more decently. Next, we introduce the set of main evaluation metrics, mostly and reliably used in object detection, to rank different model performances and select the final model. Lastly, an evaluation scheme is described in brief to show how the predictions are matched against the real objects and thereafter compute the important evaluation metrics.

3.1 Data set Description

For the research purpose, we use the *CSIRO Crown-of-Thorns Starfish (COTS) Detection Dataset* by Jiajun Lin et al. [32] to tackle the coral loss on the Great Barrier Reef due to the COTS outbreaks. The data set was collected by COTS Control teams from the Australian agencies, in collaboration with domain experts and marine scientists. The main method used to collect these underwater images was "Manta Tow". A snorkel-diver with a camera is attached to the bottom of a manta-tow boat moving at a speed of fewer than five knots and stopping every 200-meter transect. This gives the

snorkel-diver two minutes to observe variables, assess the underwater habitat virtually, and record the results back to a data sheet. We call this process "surveys". Indeed, the camera captures an oblique view of the reef and the distance between the diver and the reef changes dynamically when the diver moves around and explores the area. It can be as closed as just 10 centimeters or substantially as far as 10 meters. The setting of the camera is 24 frames per second at 3840x2160 resolution. The videos are captured continuously and processed manually afterward. Specifically, the periods of no activity between transects are cut and removed. The data set was collected by GoPro Hero 9 cameras in only one day in October 2021 at five different areas in the Swain Reefs region of the Great Barrier Reef. Undoubtedly, there exist variations in the coral habitat, lighting, visibility, depth, distance from the bottom, and viewpoint. [32].

The data set contains many sequences of underwater images. Among 35K collected images, 23,500 images are publicly released for research purposes. Each image contains mostly zero or one or even more than one starfish. Thousands of individual COTS are visible, and they are annotated by the bounding boxes around each COTS. The raw images have been annotated by expert annotators, with the help of a pre-trained COTS detection model and successfully undergone the quality assurance process. Specifically, only $4,919$ images contain the starfish, and we choose this particular number of images to train the model. Most frames have 1 to 3 annotations, a few outlier frames contain more than 10 bounding boxes and one image exists the maximum of 18 visible COTS.

Being compared to some conventional object detection benchmark datasets (PASCAL VOC, ILSVRC, MS-COCO), this collection of underwater images differs considerably in some manners. Firstly, there is only one class which is the Crown-of-Thorns Starfish while there are 20 classes in PASCAL VOC07 and VOC12, 200 classes of visual objects in ImageNet (ILSVRC) and 80 categories in MS-COCO-17 [64]. Secondly, the camera records sequences of images; therefore, COTS are framed in a sequence-based fashion (Figure 3.1). Multiple images of the same COTS are taken when the survey boat passes by. However, the detection is carried out individually, which is similar to the MS-COCO challenge. Thirdly, the starfish are good at hiding themselves; therefore, their visibility can be partly captured. Besides, multiple COTS are detected in the same image, sometimes overlapping annotations.

As shown in Figure 3.2, the Crown-of-Thorns Starfish are considerably small. On the average, the object's dimensions are approximately 50x50 pixels, which in general presents merely 0.25% of the whole image (size of 1280x720). Undoubtedly, such tiny size poses an astounding challenge to object detection and localization tasks. This is a similar problem that AI

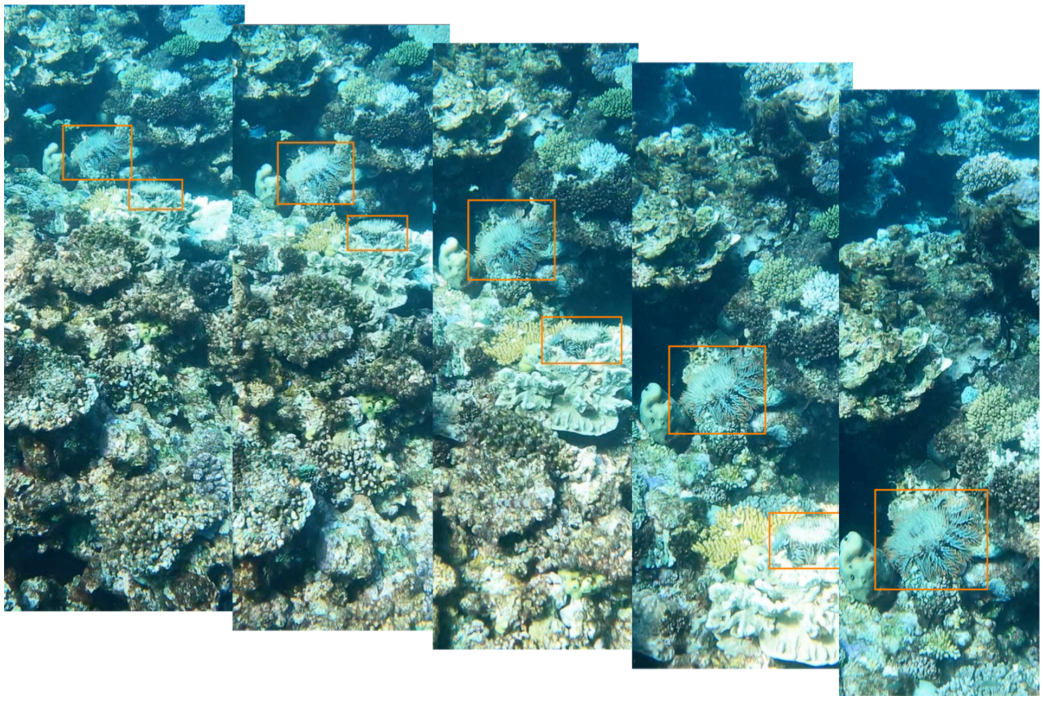

Figure 3.1: Detected COTS are annotated by bounding boxes in the image sequence [32]

practitioners encounter in the MS-COCO challenge. The data set contains common objects in their natural habitats whose area are less than 1% of the image and they are densely located in the frame [64]. Indeed, to accurately locate the COTS, the region sample size needs to be scaled substantially.

We decided to only use the images with the existence of COTS for training. Nearly 80% of the published CSIRO data set does not contain COTS annotations. Therefore, there is no point to waste computing resources and effort on background images. We split $4,919$ annotated images into a training set of size 4242 (86.2%) and a validation set of size 606 (12.3%) to approach generalization ability. The input images are pre-processed and augmented during training while the original testing images are kept untouched. This, indeed, ensures the overall performance of the model and its reliability.

3.2 Evaluation Metrics

In simple terms, a *metric* is some numerical value which can be measured in a reproducible manner, demonstrating a specific correlation with the ultimate

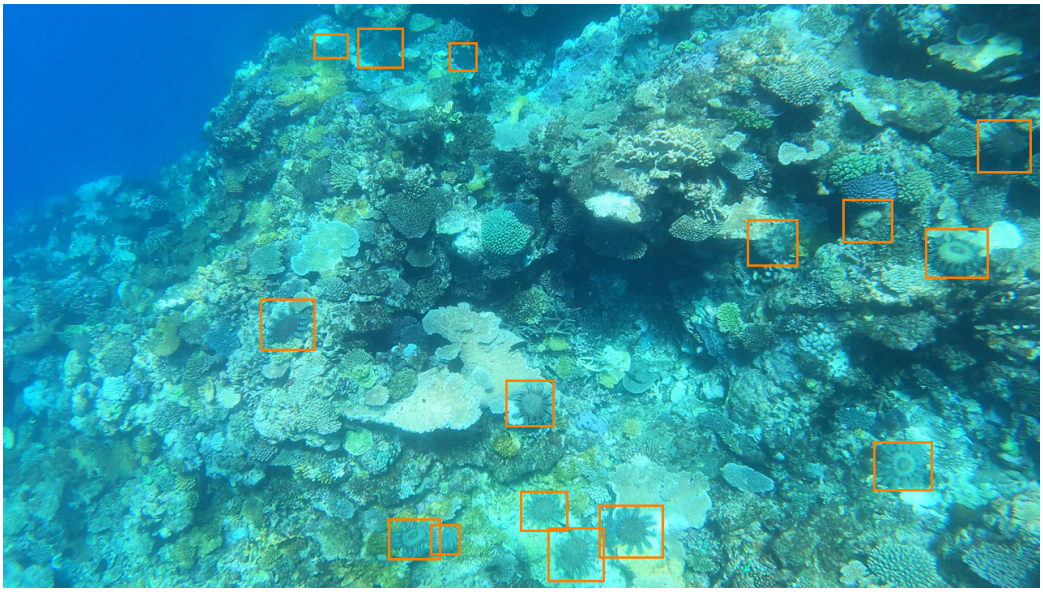

Figure 3.2: An image is annotated with the bounding box coordinates given from the original dataset. This image contains many COTS, 15 have been detected (video 2, sequence 22,643, video frame 5,777, sequence frame 414)

objective of the problem, thus providing significant insights into the model performance. In COTS detection, we do *binary classification*, either starfish or non-starfish (it can be background, coral, or other marine organisms). As such, ground truth instances are referred to as a positive example for starfish presence and as a negative example for the latter scenario accordingly. The prediction falls under a specific category of *true positive* (TP), *false positive* (FP), *true negative* (TN), or *false negative* (FN). Particularly, *true positive* indicates that the detected image region actually contains COTS. Likewise, *true negative* accurately recognizes no starfish in the predicted bounding box. In an adverse manner, *false positive* raises a false alarm when wrongly identifying the starfish in background/other object regions and *false negative* is the outcome when the model incorrectly predicts COTS presence as negative. From a naive perspective, the ultimate objective is to detect *most* of the Crown-of-Thorns Starfish presented in the given underwater images while omitting the *least* number of COTS. Such ambiguous objectives should be quantified through a set of appropriate metrics, defined as follows:

- **Precision**: This metric attempts to understand what proportion of positive identifications are actually correct. For instance, the precision of 79% indicates that among all identified starfish, 79% of them are

truly correct. The precision is defined as the percentage of correctly labeled items belonging to the positive class (COTS presence) divided by the total number of predicted positive instances, calculated as below:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

- **Recall**: Intuitively, recall captures the percentage of positive ground truth labels which are identified correctly. Specifically, a recall of 49% tells that the model correctly classifies 49% of all genuine COTS. The metric is calculated by dividing the number of correctly labeled COTS by the total number of actual COTS in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

- F_2 **score**: It would be preferable to have a single metric for performance assessment [36]. Precision and recall are in tension, which means that precision improvement typically reduces recall and vice versa. By this, combining these metrics can generate a *harmonic average* F_1. For our problem, omitting positive instances leads to severer consequences, as clearly stated in section 2.2.1. F_β, a more general F score, uses a positive real factor β to weigh recall β times more heavily than precision. Thereby, F_2 score is chosen to emphasize recall 2 times over precision. It makes sense to tolerate only a few false positives, meaning very few starfish are missed. In essence, precision and recall are provided as complementary comparisons while the average F_2 at different intersection over union (IoU) thresholds is the main metric to select the best model for this thesis.

$$F_\beta = (1 + \beta^2) \times \frac{(Precision \times Recall)}{(\beta^2 \times Precision) + Recall}, \text{ with } \beta = 2 \tag{3.3}$$

3.3 Evaluation Schemes

In order to compute the above-mentioned metrics for performance assessment, an evaluation scheme is essential, as inspired by Pasi's work [36]. It determines the whole computational process of calculating these numerical results given the object detector's outputs and the ground truth labels. First and foremost, the predicted bounding boxes must be individually matched against the ground truth bounding boxes. This is the so-called *ground truth matching* process [36]. The logic is provided in Algorithm 1. The predictions

coupled with corresponding confidence scores s and ground truth bounding boxes for a single image are taken as input. The algorithm greedily matches each prediction with a single ground truth based on a specific IoU threshold ε. In case of many predictions over one ground truth bounding box, the predicted box with the highest confidence score will be matched first. After matching, the ground truth is taken out from consideration; therefore, the remaining same predictions will be regarded as false positives. Simply put, this comprehensively incentivizes the elimination of duplicate predictions. Subsequently, the results encode the true positive $\mathcal{B}^p_{TP}$ and false positive $\mathcal{B}^p_{FP}$ which are input into the calculation of the required evaluation metrics thereafter. In this thesis, we apply an adapted MS COCO evaluation scheme which computes the metrics over different IoU threshold ε and outputs the average F_2 scores, being described as pseudo-code in Algorithm 2.

Algorithm 1 Prediction & ground truth matching for a single class [36]

Input:

$\mathcal{B}^p = \{(b^p_i, s_i)\}^N_{i=1}$	N predicted bounding boxes with confidence score s_i for from image I
$\mathcal{B}^g = \{b^g_j\}^M_{j=1}$	M ground-truth bounding boxes from image I
ε	Evaluation IoU threshold

Output: $\mathcal{Y} \in \{0,1\}^N$
N-length binary vector indicating true positive and false negative for each prediction b^i_p

function MatchBoxes($\mathcal{B}^p, \mathcal{B}^g, \varepsilon$)
  $\mathcal{Y} \leftarrow \emptyset$
  Sort $\mathcal{B}^p$ in descending order based on confidence score s
  for $i \leftarrow 1, \ldots, N$ **do** ▷ iterate over predictions
    $t \leftarrow IoU(b^p_i, \mathcal{B}^g)$
    $k \leftarrow argmax(t)$ ▷ find index of GT box with max IoU in t
    if $t_k \geq \varepsilon$ **then**
      $\mathcal{B}^g \leftarrow \mathcal{B}^g - b^k_g$
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{1\}$ ▷ matched, add one TP
    else
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{0\}$ ▷ nothing matched, add one FP
  return $\mathcal{Y}$

Algorithm 2 Detection results evaluation based on MS COCO scheme [36, 55]

Input:

$\mathcal{I} = \{(\mathcal{B}^p, \mathcal{B}^g)_k\}_{k=1}^K$	K pairs of bounding boxes, each pair of which corresponds to image I_k
iou_ε	Evaluation IoU threshold vector, there are n ε values in range of $[\varepsilon_{min}, \varepsilon_{max}]$ by a specific step_size, normally 0.05

Output:

$\mathcal{P}$	Average Precision
$\mathcal{R}$	Average Recall
F_2	Average F_2 scores

function EVALUATE($\mathcal{I}, iou_\varepsilon$)
  $m \leftarrow \emptyset$
  for ε in iou_ε **do** ▷ iterate different IoU ε threshold
    $\mathcal{E} \leftarrow \emptyset$
    for all $(\mathcal{B}^p, \mathcal{B}^g)$ in $\mathcal{I}$ **do**
      Sort $\mathcal{B}^p$ in descending order by confidence score s
      $\mathcal{E} = \mathcal{E} \cup$ MATCHBOXES($\mathcal{B}^p, \mathcal{B}^g, \varepsilon$)
    Sort $\mathcal{E}$ in descending order by confidence score s
    $\mathcal{E}'_T =$ BITWISEAND($\mathcal{E}, 1$)
    $\mathcal{E}'_F =$ BITWISEAND($\mathcal{E}, 0$)
    $\mathcal{E}_T =$ CUMULATIVESUM($\mathcal{E}'_T$)
    $\mathcal{E}_F =$ CUMULATIVESUM($\mathcal{E}'_F$)
    $\mathcal{P}_\varepsilon = \mathcal{E}_T \oslash max(\mathcal{E}_T + \mathcal{E}_F)$ ▷ element-wise division
    $\mathcal{R}_\varepsilon = \mathcal{E}_T \oslash \sum_{k=1}^K \mid \mathcal{B}_k^g \mid$
    $F_2(\varepsilon) =$ COMPUTEFBETA($\mathcal{P}_\varepsilon, \mathcal{R}_\varepsilon$)
    $m_1 \leftarrow m_1 \cup \mathcal{P}_\varepsilon$
    $m_2 \leftarrow m_2 \cup \mathcal{R}_\varepsilon$
    $m_3 \leftarrow m_3 \cup F_2(\varepsilon)$
  $\mathcal{P} = average(m_1)$ ▷ calculate average precision
  $\mathcal{R} = average(m_2)$ ▷ calculate average recall
  $F_2 = average(m_3)$ ▷ calculate average F_2

  return $\mathcal{P}, \mathcal{R}, F_2$

Chapter 4

Methods

In the previous chapter, we presented a comprehensive problem formulation for this thesis. Accordingly, this chapter focuses on describing our approach toward the COTS detection built on top of the famous single-shot object detection - YOLOv5 algorithm. As such, here we demonstrate our adaptations and small modifications to the standard YOLOv5 detector. Those changes make our model better align with the dataset and serve our purpose of robustly detecting the Crown-of-Thorns Starfish. As YOLOv5 was published as an open-source *GitHub repository*, without any official paper regarding detailed development. For this reason, we are trying to give more insights regarding different aspects, particularly loss function and bounding box aggregation. We firstly describe the multi-task losses in details; then present our ideas on initializing the network weights and image-processing during the training process, and finally define our approach to result postprocessing.

4.1 Loss Function

Among key components for solving a machine learning problem, the choice of loss function plays an indispensable role [25, 36]. As defined previously, an end-to-end COTS detector is selected, which allows to perform multiple tasks simultaneously. Therefore, a multi-task loss for object selection is absolutely a preferable choice. Many researchers have applied this concept, notably the one proposed by Girshick (2015) [13]. Multi-task loss sums up two separate loss functions, one from classification and the other from regression tasks; holistically enabling joint optimization [36]. In YOLOv5, the loss function includes three components: bounding box regression loss, objectness loss (also known as confidence loss), and classification loss [51]. It is defined

through the following formula:

$$\mathcal{L}_{total} = \mathcal{L}_{obj} + \mathcal{L}_{reg} + \mathcal{L}_{cls} \tag{4.1}$$

Given that COTS detection is a single-class problem, only one class exists (0 denotes COTS class). Hence, $\mathcal{L}_{cls}$ is omitted.

4.1.1 Objectness loss

The first component in the formula (4.1) is the *objectness loss*. Objectness concept refers to the existence of the object in an image. Objectness score for each bounding box is predicted using logistic regression. One ground truth object is assigned only one bounding box prior. Therefore, the non-best predicted bounding box priors (the one satisfying the overlapping threshold but not being the most matching) will be ignored. These boxes incur no coordinate nor classification losses, only objectness one. [40]. For the grid containing no object, the confidence score equals 0. IoU ratio between the predicted box and the true counterpart is assigned to replace that zero confidence score, instead. In short, *objectness loss* reflects the difference in the predicted "objectness" versus the ground truth's one. The loss is adapted from binary-cross entropy loss, defined as follows:

$$\begin{aligned}\mathcal{L}_{obj} = &- \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{i,j}^{obj} \left[\hat{C}_i^j log(C_i^j) + (1 - \hat{C}_i^j) log(1 - C_i^j)\right] \\ &- \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{i,j}^{noobj} \left[\hat{C}_i^j log(C_i^j) + (1 - C_i^j) log(1 - \hat{C}_i^j)\right]\end{aligned} \tag{4.2}$$

- s^2: the number of grids
- B: the number of bounding boxes in each grid
- $I_{i,j}^{obj}$ equals 1 when an object exists in the bounding box, else 0. $I_{i,j}^{noobj}$, likewise but reverse.
- $\hat{C}_i^j$: prediction confidence of the j-th bounding box in the i-th grid
- C_i^j: true confidence of the j-th bounding box in the i-th grid
- λ_{noobj}: confidence weight when no object exists in the bounding box

4.1.2 Bounding box regression loss

In object localization, Intersection over Union (IoU) is the de facto standard to measure the similarity between two bounding boxes. The overlapping area (intersection) is divided by the total area (union) of the boxes; that is the greater the IoU, the more matching the boxes are (Figure 4.1). IoU's range is between $[0, 1]$. If two boxes do not overlap, the IoU equals 0. Under such a scenario, the metric indeed does not give any information regarding their positions, reflecting if the bounding boxes are in vicinity or further away from each other [42]. By this, Generalized IoU (GIoU) would be a better substitute as it inherits major properties from IoU while rectifying the stated weakness. Formula 4.3 defines GIoU, where C is the smallest convex shape that encloses both bounding boxes B_1 and B_2. The area of C excluding B_1 and B_2 is divided by the total area of C. This ratio is subsequently subtracted from the IoU. The value range of GIoU is $[-1, 1]$ and the higher the metric, the closer the bounding boxes.

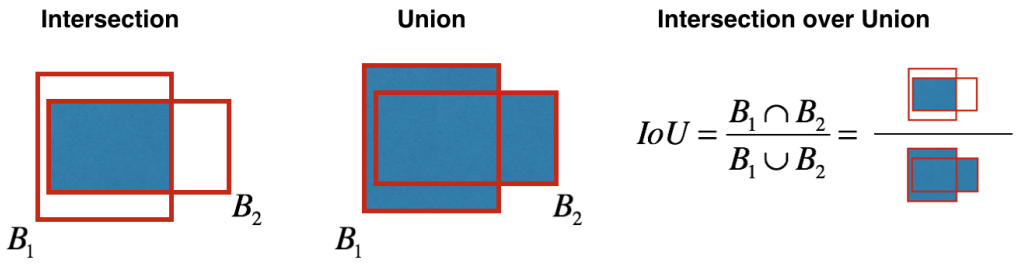

Figure 4.1: IoU illustration by Zafar et al. [20]

$$\text{GIoU} = \frac{|\, B_1 \cap B_2 \,|}{|\, B_1 \cup B_2 \,|} - \frac{|\, C \setminus (B_1 \cup B_2) \,|}{|\, C \,|} = \mathbf{IoU} - \frac{|\, C \setminus (B_1 \cup B_2) \,|}{|\, C \,|} \tag{4.3}$$

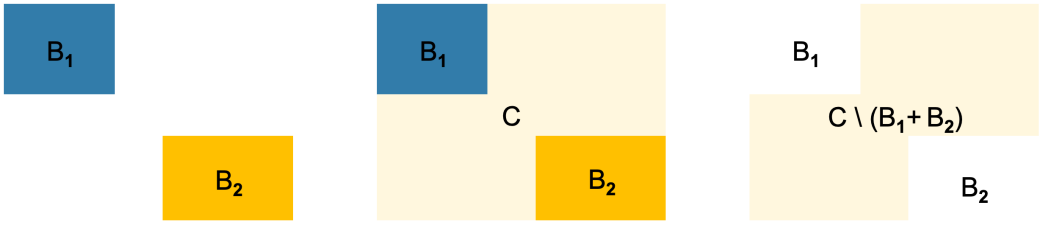

Figure 4.2: GIoU illustration in non-overlapping context

Regarding the second component of the formula (4.1), the difference between the predicted bounding box properties and the real counterparts is

measured, representing the bounding box regression loss. Generalized intersection over union (GIoU) loss is considered the loss function for bounding box regression. We are directly using GIoU metric as a loss to optimize deep neural network-based object detectors. IoU has zero gradient for non-overlapping cases, which potentially influences the training process and convergence rate. GIoU, on the other hand, has a gradient in all contexts; hence eliminating the vanishing gradients in non-overlapping cases. The regression loss is calculated as $\mathcal{L}_{GIoU} = 1 - GIoU$, expressed in full form:

$$\mathcal{L}_{reg} = \mathcal{L}_{GIoU} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{i,j}^{obj} \left[1 - IoU + \frac{A^c - U}{A^c} \right] \tag{4.4}$$

- s^2: the number of grids
- B: the number of bounding boxes in each grid
- $I_{i,j}^{obj}$ equals 1 when an object exists in the bounding box, otherwise 0
- IoU: intersection over union between the predicted bounding box and the ground truth frame
- A^c: the area of the smallest enclosing box C
- U: the total area of the predicted bounding box and the real counterpart, excluding their overlapping area

4.2 Transfer Learning

Being compared with different benchmark datasets, CSIRO Crown-of-Thorns Starfish Detection Dataset is relatively small. Indeed, training a proper deep learning model requires a sizable amount of input data. *Transfer learning* appears to be an effective technique to mitigate this deficiency of training data. Specifically, transfer learning is a paradigm of pre-training a deep network on a large dataset and using those trained weights and features to initialize the training on a much smaller and typically task-specific dataset [36]. The main idea is to learn low-level features in the early layers of the CNN, including but not limited to edges, shapes, corners, and intensity etc. which do not appear to be dataset nor problem-specific. By this, these features are not necessarily relearned entirely but are transferred from another computer vision task. Experimental results from Wang et al. (2020) [56] demonstrates the improvement in mAP by 4% using transfer learning when detecting objects using underwater robots.

For a successful learning transfer of pre-trained features to the current object detection task of interest; the relationship between the source and the CSIRO COTS dataset matters [28]. In addition, many empirical experiments have shown that transfer learning can speed up model training convergence given a sufficiently large dataset and prevent deep models from overfitting to a small dataset [16, 36, 47]. Objects365 has 365 categories from 2 million images with 30 million bounding boxes. The 365-category dataset covers also classes defined in the PASCAL VOC and the COCO benchmarks. [46]. This dataset focuses on object detection more than image classification task. Indeed, it eliminates those images which only serve the latter purpose, in a similar manner as the COCO dataset. There is a large number of images and a diversity in object categories, particularly those belonging to marine animals which perfectly meets our preference. In this thesis, we test CNN backbones for our detector which is pre-trained on the Objects365 dataset. As said, we do fine-tuning for the same CNN on our starfish dataset, continuing from a pre-trained weight initialization from Objects365-based training. Besides, we also train our model from scratch as an alternative, albeit more training time and resources are required.

4.3 Data Augmentation

As discussed, to achieve a desirable generality, deep learning requires a large amount of labeled training data. In fact, a similar external dataset seems not to exist. Underwater data is relatively scarce and under-qualified, even corrupted. Apart from transfer learning, data augmentation is another common method to mitigate data shortage while eliminating the laborious effort of manual data collection and labeling. Indeed, the benefit is two-fold: artificially inflate the dataset with new examples generated from the existing one on-the-fly, and facilitate feature extraction with the more useful information. A downside of introducing more data into the model is a significant increase in the training hours [67]. To compile an optimal training dataset for the model to train efficiently, different data augmentation techniques should be chosen properly, tested, and well evaluated. From a practical perspective, some manual observations from the original dataset could be helpful for the selection of data augmentation techniques.

4.3.1 Dataset observations

The underwater environment poses a great challenge to detect marine species. There are different unfavourable conditions of light such as light scattering,

uneven illumination in addition to water absorption of light, and massive impurities in the water. Thus, the underwater videos and images normally have low contrast, poor visibility, and substantially blue-green colour. This is not an exception for our CSIRO COTS dataset. As expected, all of the images are masked with the blue color, some intensively. Only when the camera is moving toward the starfish within a distance of less than 1 meter can we spot them clearly without being overshadowed by the blue color. Plainly speaking, the blue color at high intensities has darkened the images. Not only should color correction be applied, but also highlight adjustment. Around 10% of the images show noisiness by many white blur spots, giving them a sensation of being obscured with a thin layer of blurriness on top.

As mentioned in the dataset description, the videos were recorded within only one day in five different areas of the Reef; hence countless variations of water conditions and coral habitats. Apart from the main properties of the image (illumination and color), observations from the objects themselves are worth discussing. The dataset includes three main videos, each of which contains many sequences. Each sequence, in specific, is recorded in a relatively similar fashion. The diver is gradually approaching the coral reef; therefore, the early frames do not contain much useful information. COTS look extremely tiny and partially obscure, angled from bottom-up or top-down. The visible COTS in these frames have no definite and clear shapes. The unique thing that characterises the COTS is their distinguishable thorny legs. As the diver gets closer, the angle becomes more direct and top-down; hence well-shaped COTS visibility. Moreover, the water becomes more transparent and the images in this position of the sequence have the highest quality. The starfish are still visible at the end of the sequence; however, they are moving out of the frame little by little. Some reach the edges of the image eventually. On the whole, many images contain the COTS with approximately 40-60% of body exposure. In these circumstances, rotation and flipping techniques may be helpful. By all means, these images should undergo an enhancement process before being input into the training model. This not only retains the information of the target object but also removes interference information caused by the background and other sea animals.

4.3.2 Augmentation

We apply an online and random augmentation strategy to training images only, keeping validation and test data untouched. The transformations proceed on the fly during the training process in this "online" fashion rather than being generated beforehand. With YOLOv5, each training batch undergoes an augmentation process through a *data loader* which includes three

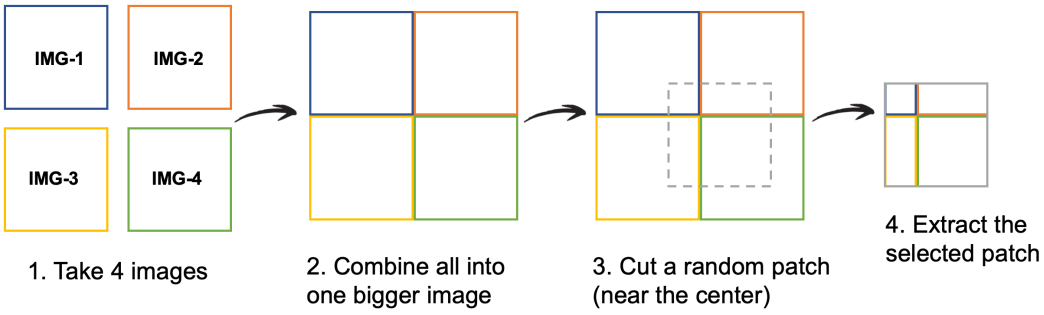

Figure 4.3: Different steps in mosaic augmentation [48]

main transformations: geometric scaling, color-space adjustments, and mosaic augmentation [50]. Mosaic augmentation is a cutting-edge technique which combines an original image with other three random images into four tiles by a random ratio. The model does not learn information from only one original image but tries to localize objects in different corners formed by different contexts. The images are never presented twice in the same way. The technique is shortly described in Figure 4.3. Indeed, the applicability of this method is proven through many experiments on different data sets, especially useful for the COCO object detection benchmark. Holistically, mosaic is exceptionally robust to address the problem of "small object detection".

Different geometric augmentation techniques are applied in our implementation, listed with details in Table 4.1. By these, new sample instances are generated by the combination of techniques such as randomly flipping the image by the y-axis; scaling the object along the x-axis by a specified factor (thus modifying the size of the original image); shifting the image around (random translation); rotating the image by a random degree; or shearing the image along the horizontal axis to produce a parallelogram-like shape. To visualize these ideas more specifically, we provide the image after transformations as in Figure 4.4.

Operations	**Possible Values**	**Probabilities**
(a) Rotation (+/- degree)	[-45, 0, 45]	[0.25, 0.5, 0.25]
(b) Translation (+/- fraction)	[-0.1, 0, 0.1]	[0.25, 0.5, 0.25]
(c) Scale (+/- gain)	[0.8, 1, 1.2]	[0.25, 0.5, 0.25]
(d) Shear (+/- gain)	[-0.1, 0, 0.1]	[0.25, 0.5, 0.25]
(d) Horizontal Flip	[true, false]	[0.5, 0.5]

Table 4.1: Geometric Transformation

Regarding color-space transformation, some techniques have been tried

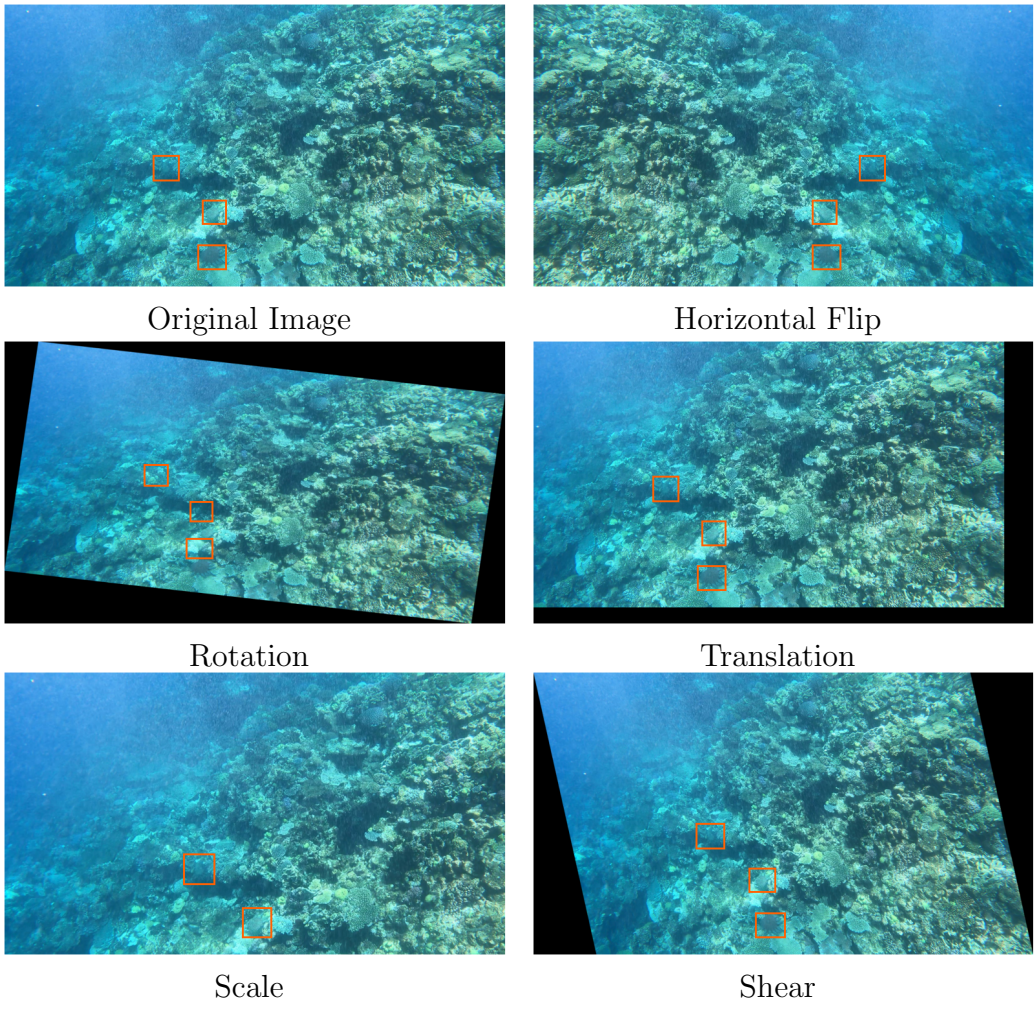

Figure 4.4: Geometric transformations applied to an image *video_1_9374.jpg*

and selectively chosen. Gamma correction is used to either encode or decode luminance. In our training dataset, it helps mostly brighten our underwater images. As observed in section 4.3.1, some of our images suffer from a foggy layer. Thus, Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied to improve the visibility level [62]. Regarding HSV, the hue of the color is captured by the hue channel while the saturation controls the colorfulness and lastly, the brightness is parameterized by a value channel. Adjusting these three channels jointly will generate a variety of color spaces. The transformations are illustrated individually in Figure 4.5.

Some advanced techniques which belong to *oversampling* strategy have been applied. New training instances are generated by mixing existing images; hence allowing the model to learn from a wider array of contexts [49].

Operations	**Possible values**	**Probabilities**
HSV-Hue (+/- fraction)	[-0.015, 0, 0.015]	[0.25, 0.5, 0.25]
HSV-Saturation (+/- fraction)	[-0.7, 0, 0.7]	[0.25, 0.5, 0.25]
HSV-Value (+/- fraction)	[-0.4, 0, 0.4]	[0.25, 0.5, 0.25]
CLAHE	clip_limit = 2, tile_grid_size = (8, 18)	0.25
Gamma Correction	gamma = 1.3	0.5

Table 4.2: Color-space Transformation

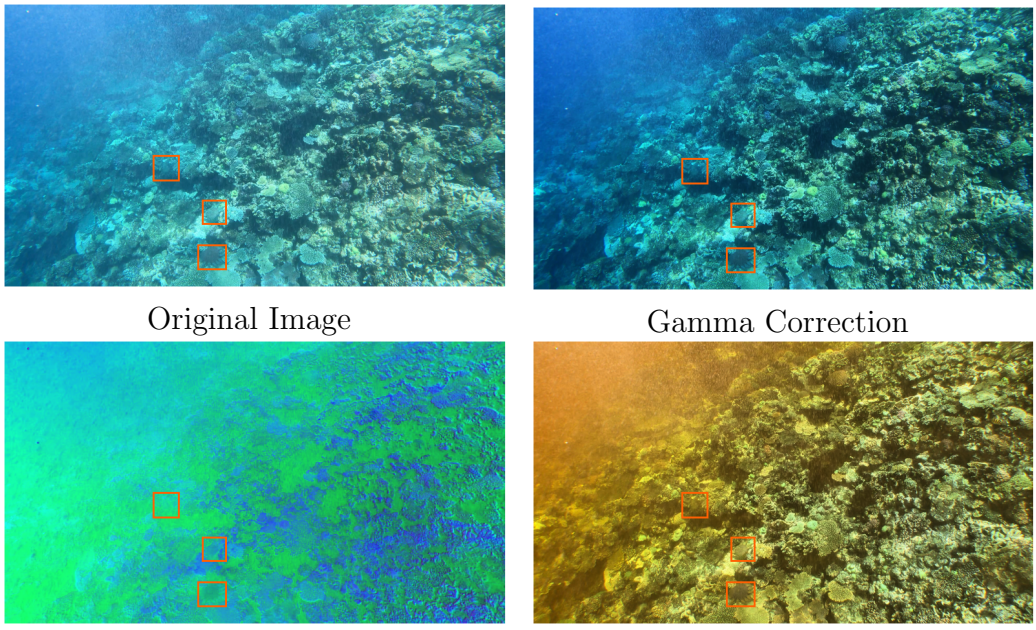

Figure 4.5: Colorspace transformations applied to an image *video_1_9374.jpg*

Mixup augmentation is a technique that generates a weighted combination of random pairs of images from the training data. In our implementation, this method is applied with a probability of 0.1. An illustration is provided in Figure 4.6. Briefly discussed above, *Mosaic* is native to YOLOv5 model and its probability parameter is set to 1. Simply put, the mosaic data loader randomly places four pictures into the final frame and apply the above augmentations (rotation, gained scaling, shear, color space scaling, etc.) on the fly. By different mixtures of techniques in randomness, we give examples of some synthesized images during training as in Figure 4.7. COTS class is denoted with 0, coupled with the annotated bounding boxes.

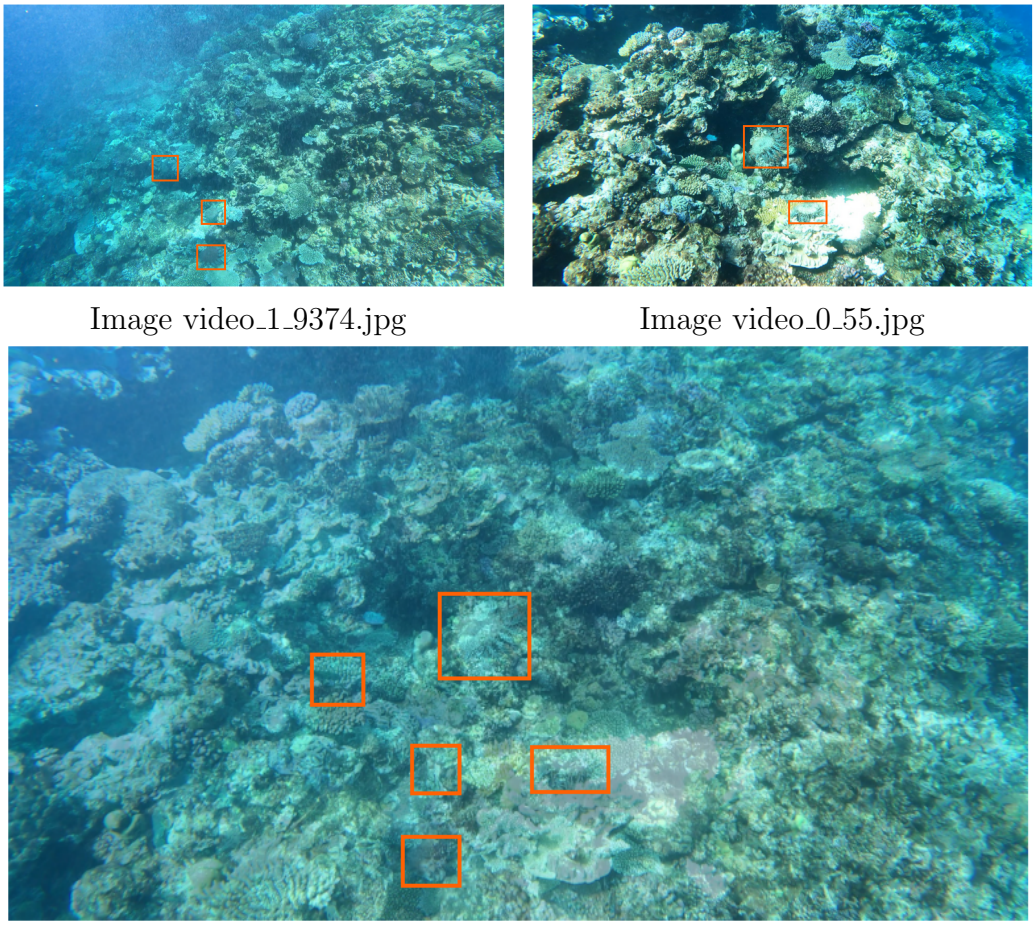

Image video_1_9374.jpg

Image video_0_55.jpg

Mixup

Figure 4.6: Mixup combines images *video_0_55.jpg* and *video_1_9374.jpg*

4.4 Bounding Box Aggregation (BBA)

You Only Look Once algorithm divides an image into an $s \times s$ grid where each cell predicts several bounding boxes, clustering around an object. Although some anchors exceed the predetermined IoU threshold ξ to be considered positive instances, a single unambiguous bounding box per object is preferable in most cases. Obviously, we need to perform a bounding box aggregation to eliminate duplicate predictions which are typically counted as false positives during the ground-truth matching assessment. In our implementation, Non-Max Suppression (NMS) is performed as a postprocessing step of the algorithm. Indeed, Non-Max Suppression is a traditional and greedy aggregation method used in many computer vision tasks. It enforces a strict

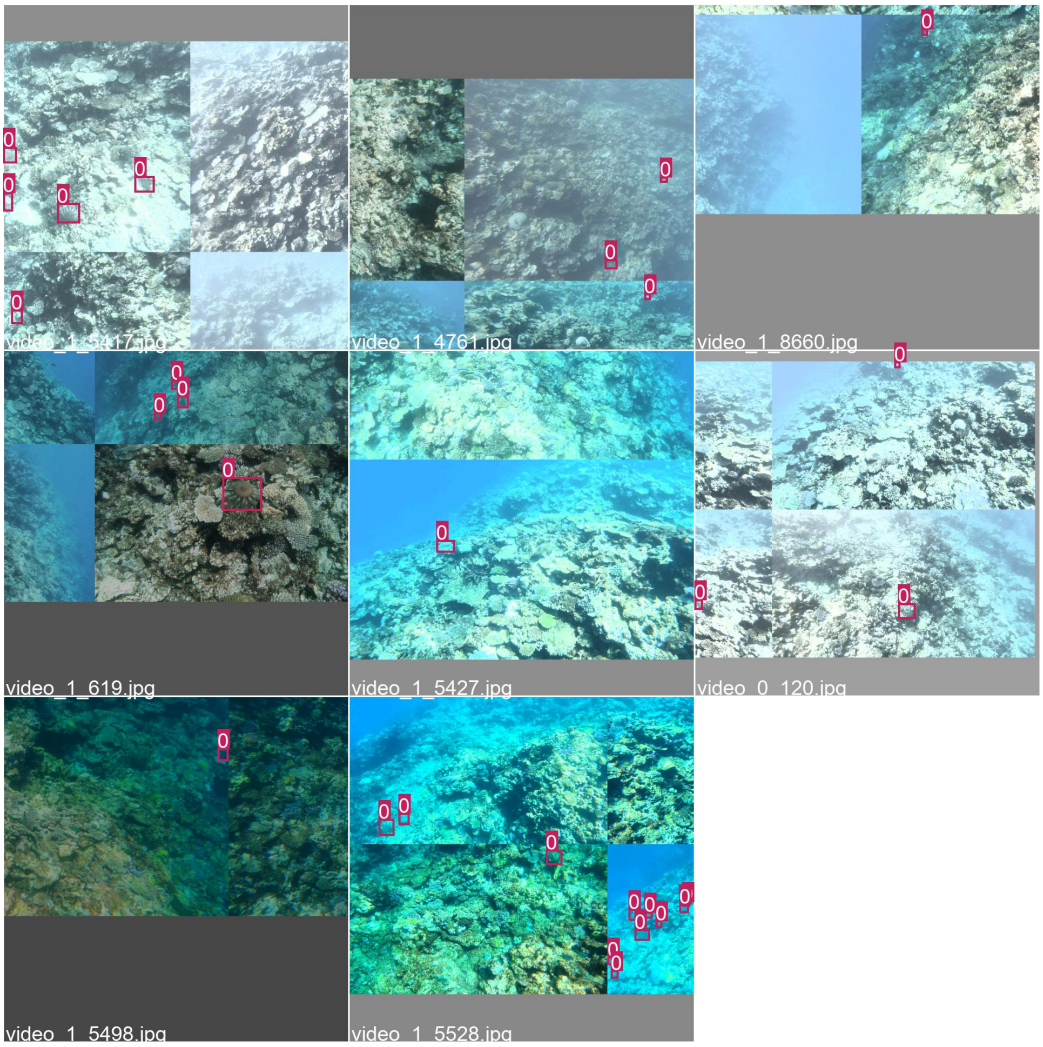

Figure 4.7: An example of an augmented data during training with certain techniques applied randomly

alignment and a unique mapping between the predicted bounding box and the real counterpart. [36].

From a practical perspective, our COTS detector will futuristically be embedded into underwater-surveillance devices which automatically navigate toward the starfish and possibly remove them from the coral. For this reason, the application requires a fine-grained delineation of individual objects. We aim to crudely yet surely detect as many COTS as possible. By this, we do not choose the Merging of Overlapping Bounding Boxes (MOB) method since it merges all overlapping and dense candidate boxes into a single *enclosing box* [36]. This method somewhat sacrifices our desired performance. As

for precise object localization, NMS may increase false-negative rate in case there are many small, densely-located and heavily overlapping objects [18]. Since the objects are occluded to a certain extent by the forefront ones, their bounding boxes are suppressed by a specific overlapping threshold during evaluation phase [4]. Hence, this degrades the object detection performance in general. Yet, this is not the case for our CSIRO dataset since the COTS are not densely-packed and overlapping, as discussed in Section 3.1.

In this thesis, we employ the Non-Max Suppression algorithm as the main bounding box aggregation method with an IoU threshold of 0.45 for validation during training, and with 0.6 for the evaluation of model performance on testing data. This threshold defines the minimum IoU value to suppress the bounding box predictions, keeping only the most reliable predictions without any duplication. When being compared to the predicted frame with the highest confidence score, among the remaining predictions, some will be removed when they individually exceed the overlapping area ratio. The algorithm is described as follows:

Algorithm 3 Bounding Box Aggregration & NMS for a single class

Input:
$\mathcal{B}^p = \{(b_i^p, s_i)\}_{i=1}^N$ | N predicted bounding boxes, confidence score s_i
c | Confidence threshold
ϵ | Bounding box aggregation IoU threshold
Output: Detection vector $\mathcal{D}$

Require: $0 \leq c \leq 1$, $0 \leq \epsilon \leq 1$
function NonMaxSuppression($\mathcal{B}^p, c, \epsilon$)
  $\mathcal{D} \leftarrow \emptyset$
  $xc \leftarrow \mathcal{B}^p > c$
  while $xc \neq \emptyset$ **do**
    $m \leftarrow$ argmax $\{s\}$ ▷ confidence score s of all predicted BB in xc
    $\mathcal{D} \leftarrow \mathcal{D} \cup (b_m^p, s_m)$
    $xc \leftarrow xc - (b_m^p, s_m)$

    for $b_i^p \in xc$ **do**
      if $iou(b_m^p, b_i^p) \geq \epsilon$ **then**
        $xc \leftarrow xc - (b_i^p, s_i)$
  return $\mathcal{D}$

Chapter 5

Experiments and Results

In this chapter, we thoroughly describe important implementation details regarding different facets of the training and validation process. Our goal is to provide a comprehensive yet concise overview of the implementation pipeline. Specifically, the parameter values mentioned in the following sections are tested and analyzed comprehensively before being used in the majority of trials. Only the most relevant details from the experimental spaces are provided to keep the focus on prominent aspects. After hundreds of trials, we have achieved satisfying performances; hence conducting model inference thereafter. We discuss this testing phase together with the outcomes in the form of tables and figures to vividly illustrate the final model selection.

5.1 Training the Model

This section provides the most crucial implementation details of the training and evaluation of our Crown-of-Thorns Starfish detector. Regarding the infrastructure, our solution is based on the *YOLOv5 framework* developed purely in Python by Glenn Jocher. Besides, we use Google Colab Pro+ as the main service for our computational hardware. For most experiments, we have Nvidia Tesla P100-PCIE-16GB, 16281MiB, 8 CPUs core, and 51.0 GB of RAM. Such hardware and software environments are sufficient for our training purposes, which ensures the rapid and frequent experimentation. We use *Weights & Biases* as the centralized experiment management and tracking which can log trained parameters as well as the results of key metrics.

Before considering any changes to the base model YOLOv5, we first train with the default settings to establish a performance baseline and spot areas for improvement. After each epoch, we run inference on a fixed and separate validation dataset, derived from the original CSIRO COTS Dataset by 12.3%.

The average F_2 score from different IoU thresholds (0.3 to 0.8, step_size = 0.05) as our main evaluation metric is calculated in accordance with our adapted MS-COCO based evaluation scheme, reviewed thoroughly in section 3.3. The validation F_2 result is used to determine and save the best version of the model. The training parameters are listed as follows:

- **Epoch** (one complete pass of the entire training data, engaged in online preprocessing, augmentation, and shuffle). We initially trained for 50 epochs; however, the model encountered overfitting mostly after 20^{th} epochs. Therefore, we adjusted the number of epochs accordingly and additionally enabled *early stopping* when the validation performance was no longer improved. The epoch with the highest average F_2 score will be considered the *best epoch* whose weights are ultimately saved as the outcome of the training.

- **Batch size**. The model was trained with a first batch size of 8, then evolved to 16 and further. However, due to the computing resource constraints, it was devastating to train a complicated model with a larger batch. On that note, we chose the largest batch size that our hardware allowed and it indeed yielded stable and high results after the evaluation.

- **Weights**. We initialized the CNN backbone with weights pretrained on the Objects365 dataset (briefly discuss in *Transfer Learning section*). Specifically, this backbone is the best epoch from training YOLOv5m on the Objects365 dataset and it is available in the YOLOv5 official repository. Moreover, the CNN backbone weights were not frozen to enable adaptation to our custom dataset during training. In fact, we also froze the backbone just for the sake of random experiment; however, the model performance was significantly worse. As another alternative, we also trained the model from scratch given that the time and resources were expensively consumed. As it happened, we initiated training with different YOLOv5 frameworks, naming just the notable ones, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5l6 and YOLOv5x.

- **Input image size**. This controls the resizing operation for an image. We first tried with the default image size of 640; however, the performance was not as profound as high-resolution images. After several experiments, the image size of 1280 indeed is the most suitable alternative; yet longer run-time and highly resource-demanding.

- **Hyperparameters**. There are different types of parameters that control the whole learning process of the deep learning model. We use the

default training settings of the underlying YOLOv5 framework with the SGD optimizer, an initial learning rate of 0.01 as well as other optimizer parameters such as momentum, warm-up epoch, weight decays, etc. In particular, *momentum* replaces the gradient with an aggregate of gradients in the gradient descent algorithm. *Weight-decay* is a regularization technique which shrinks the weights during the back-propagation process by adding a penalty term to the cost function of the network. [35].

- **NMS IoU threshold**. For Non-Max Suppression, we set the BBA IoU threshold $\epsilon = 0.45$ for all candidate models coupled with confidence score threshold $c = 0.001$. This choice of the confidence score is for the valid mAP values.

- **Anchor Box**. In YOLOv5, anchor boxes automatically learn from the custom training dataset [50]. Simply put, AutoAnchor algorithm runs anchor verification and generation before the training starts. Apart from our dataset, it will determine whether the original anchor is a good fit or if an adjustment is required based on the input image size. Indeed, the four Feature Pyramid Network (FPN) layers from the YOLOv5 detector (from P_3 to P_6) are individually associated with a specific anchor box. Each pyramid level is responsible for detecting the objects according to their model scales (s - small, m - medium, l - large, x - xlarge).

Discussed briefly above, we want to provide a deeper understanding about our experiments with different hyperparameters. As we're mostly aware, hyperparameters are not trainable during the machine learning optimization process, but they indeed tune this process to find the best model architecture [36]. We first used the hyperparameters from the base model and then tuned them to have some benchmark on the training performance. Then we ran a finite grid search (due to our limitation in computing resources) over a hyperparameter space to automatically find the best ones. Besides optimization hyperparameters as listed above, we experimented with different settings for data augmentation regarding geometric, color-space, and advanced transformations. By all means, finding a suitable set of hyperparameters is considerably far from trivial. Therefore, we only presented the most notable hyperparameters which were considered significant contributors to the overall performance, not a dedicated list of all experiments.

5.2 Experimental Results

In this section, we summarize key findings from our experimental results and provide the rationales behind the best model choice. One should notice that our experiments are incremental in nature where the subsequent trial is built upon the previous training outcomes. We also tune the default parameters to make sure that the chosen architecture would be superior to the rest. We run many random experiments to search for new ideas; however, only the most relevant observations are presented in the findings. Undoubtedly, our goal is to provide insights into the most essential properties of our optimal model rather than being redundant. Besides, the allocation of GPU differs from time to time; thus we can't provide an accurate speed comparison. We endeavor to present the outcomes in a logical manner rather than in the chronological order of the experiments. The performances are selectively provided in Table 5.1

We applied transfer learning to initiate the training model with pretrained weights directly from the YOLOv5m Objects365 and the results were profound. The training duration is just 1/3 when being compared to training from scratch run time, under the same settings of hyperparameters. Additionally, the computational resources allowed to train with a larger batch size (the maximum of 9 when we used auto-batch)[1] with the former technique than the latter of 5. For the sake of experiment, we locked the weights of YOLOv5m Objects365's backbone (12 layers) and not surprisingly, the results inclined us to favor this attempt. The results were much lower than in other experiments while the objectness loss was approximately three times higher. Hence, this implies that some domain adaptation is absolutely necessary when training a custom dataset.

Training from scratch is our alternative method to compare with the pretrained weights initialization. We started with the simplest model YOLOv5s, especially YOLOv5s6 (with an additional large object output layer P6). With this architecture, we gained a fairly average result which was considered a performance baseline. YOLOv5l6 was chosen since it showed the best alignment with our object detection problem despite the long training hours. There was a harmonic balance between precision and recall achieved by YOLOv5l6 model. Not stopping there, we also tried the most complicated architecture - YOLOv5x. As introduced before, YOLOv5x is an extended model of the YOLOv5 series. It requires substantial computing resources (CUDA memory) as it has more parameters. During the experimental phase, YOLOv5x shows a poorer performance than YOLOv5l and the calculation speed is

[1]The maximum that could fit into current GPU memory

	Validation set results (%)		
Experiment	Precision	Recall	F_2
Default settings to set performance baseline, image size = 640			
(a) Objects365, pretrained	74.8	42.2	46.2
(b) Objects365, freeze backbone	30.9	38.3	36.5
(c) YOLOv5s, default	34.8	50.3	46.2
(d) YOLOv5s6, default	35.3	42.6	40.9
(e) YOLOv5m, default	44.9	46.8	46.4
(f) YOLOv5m6, default	40.2	46.6	45.1
(g) YOLOv5l, default	57.7	46.3	48.2
(h) YOLOv5l6, default	**42.6**	**50.1**	**48.4**
(i) YOLOv5x, default	62.2	41.5	44.5
The best tuned models, image size = 1280			
(j) Objects365, pretrained, HSV	75.7	75.2	75.3
(k) Objects365, pretrained, no HSV	**81.5**	**86.4**	**85.4**
(l) YOLOv5l6, HSV	74	75.9	75.6
(m) YOLOv5l6, no HSV	81	85.3	84.4

Table 5.1: The experiments on different model architectures. (a) training initiated with pretrained weights from Objects365 YOLOv5m, default settings (b) training initiated with pretrained weights from Objects365 YOLOv5m with the frozen backbone of 12 layers, default settings. (c) to (i) training from scratch with YOLOv5 models of different sizes: small, medium, large and extended-large. (d), (f), (h) are YOLOv5 P6 models. (j) to (m) are the best models with optimally tuned hyperparameters. (j) and (l) have Hue Saturation Value transformation while (k) and (m) don't. One should notice that not all the experiments results are presented albeit our huge effort in random experiments.

significantly slow. Unambiguously, this extended model does not meet the expectation of accurate detection nor satisfy the lightweight pursuit of UUV. All aspects considered, YOLOv5l6 is the most cost-effective implementation yet bringing about outstanding outcomes.

The training is done with different image input sizes, starting from 640. After all, we find out that the size of 1920 generates the best result without compromising the whole computing resources. A higher resolution image drastically improves the performance, which is totally explainable due to the increased pixel information. As clarified in Section 3.1, 4,242 images are used for training while the validation set contains 606 images. As the images are extracted from the video so they belong to specific video sequences. By this,

the images in sequence are almost identical to a certain extent, depending on the diver's speed of moving. On that note, our split ensures that no training sequence is leaked into the validation; hence a reliable evaluation of the model performance. Besides, we consider the recommendation from YOLOv5 documentation of training with 0-10% background images to produce the best results. Yet, 0% background works best in reality. For the testing dataset, we combined the rest of 71 images in which COTS are presented and the other 50 images in the same video sequence but without any starfish (background images/ negative instances).

Regarding the image augmentation, different combinations of techniques have been tested and analyzed for the best outcomes. The geometric transformations significantly improve the performance of our COTS detector. A notable observation is that removing rotation of 45 degrees would lower recall by 7% while precision remained mostly unchanged. This is somewhat considerable as we place a strong emphasis on the recall metrics. On the other hand, some techniques in the color-space conversion are less preferable. Brightness enhancement helps significantly while HSV is deemed detrimental. We decide to remove HSV transformations eventually. Regarding the advanced techniques, mosaic has proven its superiority by strong empirical evidence, the training without mosaic in similar settings degrades the model performance (F_2 score) by 5%. Undeniably, mosaic is really a compelling property of the YOLOv5 algorithm.

5.3 Model Inference

We finally have the best models from the experimental spaces after iterative experiments. Hence, the next step is to assess the models' performance and their generalization ability by running model inference on the testing dataset. Indeed, we choose two models which have almost equivalent performance for the testing phase. Additionally, we infer the model with different modes: with/without Test Time Augmentation (TTA). According to YOLOv5 documentation from Ultralytics, TTA enhances mAP and Recall considerably during testing and inference. In specific, TTA upscales the image size by 30% coupled with horizontal flipping and the images are processed at three different resolutions. The outputs are merged before the non-maximum suppression postprocessing stage [2]. Indeed, inference with TTA takes 2-3x more times than normal evaluation. Due to the large amount of small starfish in the dataset, our model may benefit from higher resolution image [50]. To achieve the best inference results, we detect the COTS at 1280 resolution in the same manner as in the training process.

	With TTA, results (%)			**Without TTA, results (%)**		
Experiment	Precision	Recall	F_2	Precision	Recall	F_2
(k), $c = 0.2$	70.6	80	77.9	85.7	88.9	88.2
(k), $c = 0.25$	90.9	80	82	**92.8**	**91.7**	**91.9**
(m), $c = 0.25$	76.1	81.4	80.3	77.8	0.8	79.5
(m), $c = 0.3$	84.2	86.5	86	83.3	86.2	85.6

Table 5.2: Model inference results with/without Test Time Augmentation, image_size = 1280, NMS IoU threshold $\epsilon = 0.6$, NMS confidence threshold c for (k) and (m) - the best models depicted in Table 5.1.

As observed, no obvious overfitting patterns are visible in the table. The best model obtains a relatively high precision as well as recall which implies that the model can calibrate properly with the testing images. This result is given in the condition of a confidence score $c = 0.25$. The low the confidence score results in low precision since the model accepts more predictions as positive instances, most of which contribute to false positives. Indeed, the precision increases nearly 20% when the c changes from 0.2 to 0.25 in the model (k) with TTA mode. To some extent, the precision metric is clearly correlated with the confidence threshold. Although our goal is to crudely predict as many COTS as possible, high precision is indispensable for model reliability. Regarding TTA, it is obvious that the inference without augmentation is preferable. In practice, we select $c = 0.25$ for the pretrained model to highlight the precision of 92.8% while keeping a commendable recall of 91.7%. We provide the inference results for a single image and for a sequence of images in Figure 5.1 and Figure 5.2 accordingly. Eventually, training with pretrained weights from YOLOv5m Objects365 yields the best results. The model generalizes well to our testing dataset and detect the object at speedy rate (1.7ms NMS/ image).

video_2_7999.jpg Predicted video_2_7999.jpg

Figure 5.1: Model inference on a random test image *video_2_7999.jpg*

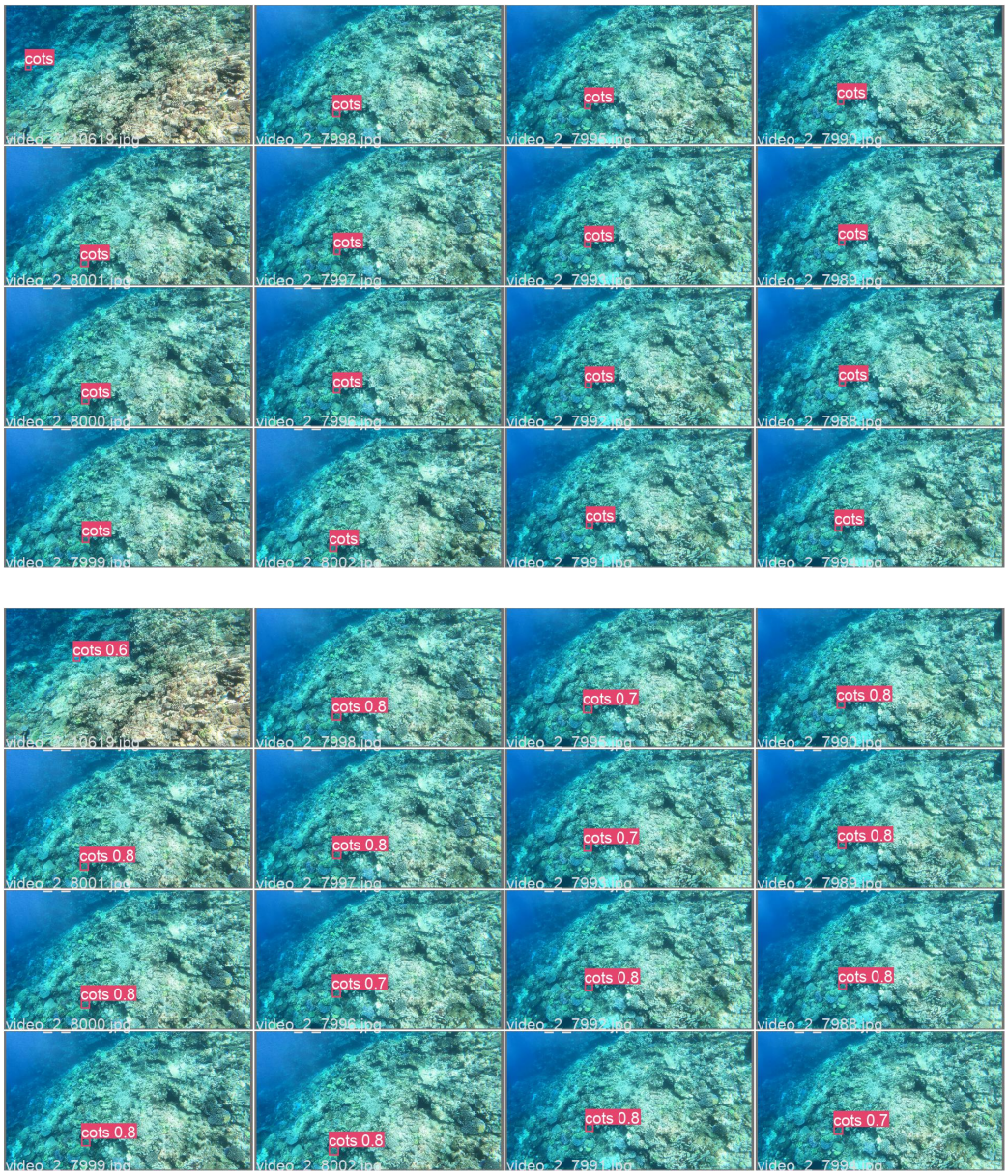

Figure 5.2: Model inference on a video sequence. The upper image contains the original images with their ground-truth labels. The lower image contains the predictions (bounding boxes and confidence scores for the COTS).

Chapter 6

Evaluation

Given our experimental results, this chapter will provide deeper insights into the evaluation of our starfish detector. This part is given as a self-reflection on our dedicated effort (*what has been done well*) as well as improvements ideas for future development. The evaluations are taken against different aspects such as the CSIRO dataset, best practices when training YOLOv5 model from official documentation, loss and evaluation schemes, etc. Thus, we can derive potential enhancements regarding COTS detection task specifically, or even marine organisms detection on the whole.

6.1 Model Evaluation

Our COTS detector has fulfilled its mission to a satisfying extent. The selected model can recognize many of the Crown-of-Thorn Starfish in the given underwater images and also localize them quite correctly. The results from the testing dataset have proven its generality when "seeing" new images for the first time. Although we place a strong emphasis on recall to make sure that we miss the least amount of this seabed creature for mitigating the outbreak possibility. Furthermore, we also attempt to ensure the accuracy of the model by not compromising the precision too much.

Benchmarking the implementation documentation from Ultralytics - the owner of this state-of-the-art YOLOv5 algorithm, we are moderately confident with our model reliability. As suggested, there should be more than $1,500$ images per class with an approximate $10,000$ labeled objects. Our dataset indeed contains $4,919$ detected COTS images with $11,898$ annotated instances. Variation in the images is utterly vital for achieving representative ability. The CSIRO dataset meets this requirement since the underwater videos are captured from different watery conditions at different times of the

day, under various light exposures, and through diverse camera angles, etc. As discussed in Chapter 3, this dataset underwent an AI-assisted annotation and passed a rigorous quality assurance test before being published. Therefore, label consistency is somehow ensured in the same manner as label accuracy. Simply put, no positive instances should miss the labels or be enclosed by loose bounding boxes.

Regarding the selection of key hyperparameters, at first, we tried with a large batch size (e.g. 20); however, this number was infeasible when we trained a more complicated architecture such as YOLOv5l6 and with high-resolution images. For this reason, we chose the largest batch size that our hardware would allow for (9 is the maximum for the pretrained and 5 for YOLOv5l6, the input image size is 1280). We acknowledged that a small batch size might produce poor batch norm statistics. Yet, the training and evaluation results have demonstrated persuasive evidences of stable performances and successful generality. Introducing augmentation on the fly during training also mitigates the risk of overfitting. We observed the influences of different augmentation techniques on model performance either separately or in combination. Through these findings, the hyperparameters were evolved accordingly to justify their inclusions in the augmentation strategy. All things considered, we have obtained desirable results against our initial expectations for this task.

Nevertheless, the underwater images are in sequence, and each image almost replicates its previous frame, the use of mosaic augmentation with the probability of 1 enhances the overall performance substantially. That is, every image is unique since it is a random and centered crop from a combination of four images. Therefore, the same image never appears twice. This compelling characteristic of mosaic indeed alleviates the pain of data shortage and duplication. In addition, transfer learning from YOLOv5m on the Objects365 dataset is truly a critical cornerstone of our implementation. The pretrained-weight initialization does not only compensate for our small dataset but also enables a cost-effective implementation. The training and evaluation time in total is just about one-third, compared to the run time of training from scratch. Moreover, less computing resources are required, hence the pretrained model allows training with the larger batch size. Without freezing the model's backbone, this model learns the data representations efficiently and performs the best detection so far.

6.2 Discussion

At this point, we have gained certain insightful observations and findings from our implementations. By this, we recognize different areas for further improvements. These aspects belong to either outside of our research scope or limitations in the implementations.

With regards to image augmentation, there are other useful techniques that can be integrated directly through the Albumtations library. However, we only applied the most frequently used approaches referenced from some existing works. With the limited availability of computing resources, we only performed simple tuning for our predefined data augmentation scheme. As stated before, the underwater environment is truly special in the sense of blue/green color due to light absorption. We have experimented with different color-space transformations to rectify the underwater images; however they surprisingly worsen the performance. Apart from *gamma correction*, we need further analysis on the color correction to somehow boost the overall model performance. To that end, color correction for underwater images and a thorough augmentation strategy on the whole for this specific task are among potential development direction.

Finding an external dataset is another potential alternative; however, it requires a considerable amount of work. Synchronizing many sources of images is far from trivial from both technical and theoretical perspectives. Indeed, we fail to discover any similar dataset but we believe that more qualified datasets will be published soon in the urgent of COTS surveillance. Due to such uniqueness, the CSIRO COTS Dataset motivates us to utilize it to the fullest. From a data-centric point of view, there exist various approaches to process this dataset which open boundless development possibilities.

Implementation with pretrained weights can also be experimented further by other benchmark datasets. In our case, Objects365 has the most similarity with our custom dataset; therefore, being chosen for the trial. Researching or implementing other alternatives is not our main priority; hence out of scope. Regarding evaluation metrics, we choose F_2 score as the main criteria for model evaluation and selection as recall is more essential for our COTS detector. However, in native YOLOv5, widely used average precision (AP) is the evaluation choice. The fitness of the model during the training and evaluation phase is defined by mAP@0.5[1] and mAP0.5-0.95[2] with the ratio of 0.1 and 0.9 accordingly. Thus, using this evaluation scheme can be a worthwhile endeavor.

[1]Average mAP at IoU threshold 0.5

[2]Average mAP over different IoU thresholds, from 0.5 to 0.95, step 0.05

In our research, we don't use Focal Loss for loss computation. Specifically, *Focal loss* is a variant of cross-entropy loss that addresses the extreme imbalance between foreground and background classes which may partially exist in our COTS detection problem. One-stage object detectors in general evaluate approximately $10^4 - 10^5$ candidate locations per image; only a few of which contain the class. By this, easily classified background examples incur a non-trivial loss which overwhelms the infrequent observations and dominates the gradient [31, 36]. Focal loss down-weighs easy examples by adding a modulating factor to the conventional cross-entropy loss, hence focusing the model on hard negatives. Much severe imbalance happens in the case of tiny objects and large images. From our perspectives, such is not our scenario; hence not within our consideration. To summarize, we only recommend the most relevant and straight-forward directions for further improvements. Under our assumption, this list of improvement ideas is not conclusive yet and there are many other possibilities yet to be discussed.

Chapter 7

Conclusions

In this thesis, we developed a modern detector based on deep learning in COTS surveillance for maintaining the biological diversity of the Great Barrier Reef. We have tackled different challenges presented in the CSIRO COTS dataset, including small object size, low visibility due to the starfish's camouflage ability, varying coral habitat, lighting, sea depth, and variations in the camera perspective, to name just a few. These aspects truly simulate the real-world scenario in underwater object detection tasks. Yet, our COTS detector is able to extract object representations directly from the underwater images, and it remains robust to the high intra-class variation presented in the given dataset. Furthermore, images in video sequences are diversified through mosaic augmentation techniques during processing. Hence more useful and relevant information about the starfish can be extracted. Since this dataset was released recently, there have not been any published papers yet. It is hard for us to benchmark our performance and evaluate the superiority of our model. However, we believe that our work contributes significantly to this area of research and sets a milestone for using deep learning to detect this seabed creature.

On the whole, the model performance is not the only criteria to rank our effort but other outstanding facets. The author has conducted an intensive literature review in addition to a comprehensive picture of the problem and its major causes. Thereafter, the problem is framed in a decent manner that aligned with the goal of the project. By this, a particular evaluation scheme is defined to ensure fairness and credibility of model assessment. This indeed has been ignored in many research works, albeit its significance. The lack of proper evaluation definition leaves room for ambiguous results. Regarding the implementations, we did not experiment with all possibilities; however, we attempt to cover most of the essential aspects such as transfer learning, rich data augmentation, hyperparameters tuning, model calibra-

tion, etc. Yet, our effort is by no means conclusive. That is, we provide a foundation work for further developments built upon. Real-world applicability is necessary, which is outside this academic exercise.

In closing, our model has obtained a satisfying performance. It is able to generate an automatic, fast, and reliable COTS detection. Indeed, the accuracy and speed need improving further to ensure an adequate quality for being embedded into underwater object devices. Furthermore, the generalization ability of our COTS detector should be further verified against different conditions of the undersea environment. This is among our recommendations for future research direction. Considering the usability of our model, it is likely to suit other applications and problems in the same domain such as detecting other marine organisms. Looking forward, more fine-grained vision-based detectors with novel algorithms are expected to emerge in the near future.

Bibliography

[1] Crown-of-thorns starfish strategic management framework, July 2020. https://nla.gov.au/nla.obj-2866007885/view. Accessed 22.4.2022.

[2] Arie, L. G. The practical guide for object detection with YOLOv5 algorithm, March 2022.

[3] Babcock, R., Plaganyi, E., Condie, S., Westcott, D., Fletcher, C., Bonin, M., and Cameron, D. Suppressing the next crown-of-thorns outbreak on the Great Barrier Reef. *Coral Reefs 39* (October 2020).

[4] Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. Soft-NMS – Improving object detection with one line of code, 2017. https://arxiv.org/abs/1704.04503.

[5] Chollet, F. *Deep Learning with Python.* Manning Publications Co., Shelter Island, NY 11964, 2018, ch. 1, pp. 3–66.

[6] Chollet, F. *Deep Learning with Python*, second ed. Manning Publications Co., Shelter Island, NY 11964, 2021, ch. 8, pp. 201–238.

[7] Clement, R., Dunbabin, M., and Wyeth, G. Toward robust image detection of crown-of-thorns starfish for autonomous population monitoring. In *Proceedings of the 2005 Australasian Conference on Robotics and Automation*, C. Sammut, Ed. Australian Robotics and Automation Association, Australia, 2005, pp. 1–8. https://eprints.qut.edu.au/32830/.

[8] Cui, S., Zhou, Y., Wang, Y., Zhai, L., and Minutolo, A. Fish detection using deep learning. *Appl. Comp. Intell. Soft Comput. 2020* (January 2020).

[9] Dayoub, F., Dunbabin, M., and Corke, P. Robotic detection and tracking of crown-of-thorns starfish. In *2015 IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS) (2015), pp. 1921–1928.

[10] Do, T. Evolution of Yolo algorithm and Yolov5: The state-of-the-art object detention algorithm. Bachelor's thesis, Oulu University of Applied Sciences, 2021.

[11] Elawady, M. Sparse coral classification using deep convolutional neural networks, 2015. `https://arxiv.org/abs/1511.09067`.

[12] Fayaz, S., Parah, S., and Qureshi, G. Underwater object detection: architectures and algorithms - a comprehensive review. *Multimedia Tools and Applications* (March 2022).

[13] Girshick, R. Fast R-CNN. *CoRR abs/1504.08083* (2015).

[14] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[15] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G., Cai, J., and Chen, T. Recent advances in convolutional neural networks, 2015.

[16] He, K., Girshick, R., and DollÃ¡r, P. Rethinking imagenet pre-training, 2018.

[17] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 37*, 9 (2015), 1904–1916.

[18] He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X. Bounding box regression with uncertainty for accurate object detection, 2018.

[19] Hinton, G. E., and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (July 2006), 504–507.

[20] Iffat, Z., Giounona, T., Richard, B., Nimesh, P., and Leonardo, A. *Hands-on Convolutional Neural Networks with Tensorflow*. Packt Publishing, Shelter Island, NY 11964, August 2018, ch. 1, pp. 157–200.

[21] Ikeuchi, K., Ed. *Computer Vision, A Reference Guide*. Springer, 2021.

[22] Ivan, L. The essence of artificial neural networks. Medium, 2016.

[23] Jalal, A., Salman, A., Mian, A., Shortis, M., and Shafait, F. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecological Informatics 57* (2020), 101088.

[24] Jiang, X., Hadid, A., Pang, Y., Granger, E., and Feng, X., Eds. *Deep Learning in Object Detection and Recognition*. Springer Nature Singapore Pte Ltd., 2019, ch. 1 and 2, pp. 1–54.

[25] Jung, A. Machine learning: The basics, 2018. https://doi.org/10.48550/arxiv.1805.05052.

[26] Kandimalla, V., Richard, M., Smith, F., Quirion, J., Torgo, L., and Whidden, C. Automated detection, classification and counting of fish in fish passages with deep learning. *Frontiers in Marine Science 8* (2022). https://www.frontiersin.org/article/10.3389/fmars.2021.823173. Accessed 4 May 2022.

[27] Kayal, M., Vercelloni, J., Lison de Loma, T., Bosserelle, P., Chancerelle, Y., Geoffroy, S., Stievenart, C., Michonneau, F., Penin, L., Planes, S., and Adjeroud, M. Predator crown-of-thorns starfish (acanthaster planci) outbreak, mass mortality of corals, and cascading effects on reef fish and benthic communities. *PloS one 7* (10 2012), e47363.

[28] Li, L., Wanli, O., Xiaogang, W., Paul, F., Jie, C., Xinwang, L., and Matti, P. Deep learning for generic object detection: A survey, 2018. https://arxiv.org/abs/1809.02165.

[29] Li, X., Shang, M., Hao, J., and Yang, Z. Accelerating fish detection and recognition by sharing CNNs with objectness learning. In *OCEANS 2016 - Shanghai* (2016), pp. 1–5.

[30] Li, X., Shang, M., Qin, H., and Chen, L. Fast accurate fish detection and recognition of underwater images with fast R-CNN. In *OCEANS 2015 - MTS/IEEE Washington* (2015), pp. 1–5.

[31] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. Focal loss for dense object detection, 2017.

[32] Liu, J., Kusy, B., Marchant, R., Do, B., Merz, T., Crosswell, J., Steven, A., Heaney, N., von Richter, K., Tychsen-Smith, L., Ahmedt-Aristizabal, D., Armin, M. A., Carlin, G., Babcock, R., Moghadam, P., Smith, D., Davis, T., Moujahid, K. E., Wicke, M., and Malpani, M. The CSIRO Crown-of-Thorn starfish detection dataset, 2021.

[33] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. Path aggregation network for instance segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 8759–8768.

[34] Minsky, M., and Papert, S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[35] Munawar, M. R. How do hyperparameters of YOLOv5 work?, February 2022. `https://chr043416.medium.com/how-hyperparameters-of-yolov5-works-ec4d25f311a2`. Accessed 31 May 2022.

[36] Pasi, P. AIR: Aerial Inspection RetinaNet for Land Search and Rescue Missions. Master's thesis, Aalto University. School of Science, 2022. `http://urn.fi/URN:NBN:fi:aalto-202202061749`.

[37] Pietikäinen, M. Local Binary Pattern. Scholarpedia, 2010. `http://www.scholarpedia.org/article/Local_Binary_Patterns`. Accessed 26 April 2022.

[38] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection, 2015.

[39] Redmon, J., and Farhadi, A. Yolo9000: Better, faster, stronger, 2016. `https://arxiv.org/abs/1612.08242`. Accessed 4 May 2022.

[40] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement, 2018. `https://arxiv.org/abs/1804.02767`. Accessed 4 May 2022.

[41] Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39* (June 2016).

[42] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression, 2019.

[43] Rosenblatt, F. The perceptron - a perceiving and recognizing automaton. Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[44] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323*, 6088 (1986), 533–536. http://www.nature.com/articles/323533a0.

[45] Saurabh, M. What is the relation between artificial and biological neuron. Medium, 2020.

[46] Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., and Sun, J. Objects365: A large-scale, high-quality dataset for object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 8429–8438.

[47] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2014.

[48] Singh, A. Data augmentation for object detection, 2020. https://www.kaggle.com/code/ankursingh12/data-augmentation-for-object-detection/notebook. Accessed 26 May 2022.

[49] Solawetz, J. Data augmentation in YOLOv4, May 2020. https://blog.roboflow.com/yolov4-data-augmentation/. Accessed 23 May 2022.

[50] Solawetz, J. Yolov5 new version explained, June 2020. https://blog.roboflow.com/yolov5-improvements-and-evaluation/#a-development-history-of-yolov5. Accessed 12 May 2022.

[51] Song, Q., Li, S., Bai, Q., Yang, J., Zhang, X., Li, Z., and Duan, Z. Object detection method for grasping robot based on improved yolov5. *Micromachines 12*, 11 (2021). https://www.mdpi.com/2072-666X/12/11/1273.

[52] Sung, M., Yu, S.-C., and Girdhar, Y. Vision based real-time fish detection using convolutional neural network. In *OCEANS 2017 - Aberdeen* (2017), pp. 1–6.

[53] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1–9.

[54] Szeliski, R. *Computer Vision, Algorithms and Application*, second ed. Springer, 2022.

[55] Tsung-Yi, L., Michael, M., Serge, B., Lubomir, B., Ross, G., James, H., Pietro, P., Deva, R., Lawrence, Z. C., and Piotr, D. Microsoft COCO: Common objects in context, 2014.

[56] Wang, C.-C., and Samani, H. Object detection using transfer learning for underwater robot. In *2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS)* (2020), pp. 1–4.

[57] Wang, C.-C., Samani, H., and Yang, C.-Y. Object detection with deep learning for underwater environment. In *2019 4th International Conference on Information Technology Research (ICITR)* (2019), pp. 1–6.

[58] Wang, H., Sun, S., Wu, X., Li, L., Zhang, H., Li, M., and Ren, P. A YOLOv5 baseline for underwater object detection. In *OCEANS 2021: San Diego, Porto* (2021), pp. 1–4.

[59] Weng, L. Object detection for dummies part 3: R-cnn family. *lilianweng.github.io* (2017). `https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/`. Accessed 29 April 2022.

[60] Weng, L. Object detection part 4: Fast detection models. *lilianweng.github.io* (2018). `https://lilianweng.github.io/posts/2018-12-27-object-recognition-part-4/`. Accessed 4 May 2022.

[61] Xu, R., Lin, H., Lu, K., Cao, L., and Liu, Y. A forest fire detection system based on ensemble learning. *Forests 12* (February 2021), 217.

[62] Yadav, G., Maheshwari, S., and Agarwal, A. Contrast limited adaptive histogram equalization based enhancement for real time video system. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2014), pp. 2392–2397.

[63] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. Object detection with deep learning: A review, 2018. `https://arxiv.org/abs/1807.05511`.

[64] Zhengxia, Z., Zhenwei, S., Yuhong, G., and Jieping, Y. Object detection in 20 years: A survey, 2019. `https://arxiv.org/abs/1905.05055`.

[65] Zhong, J., Li, M., Qin, J., Cui, Y., Yang, K., and Zhang, H. Real-time marine animal detection using yolo-based deep learning networks in the coral reef ecosystem. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLVI-3/W1–2022* (2022), 301–306.

[66] Zhou, H., Huang, H., Yang, X., Zhang, L., and Qi, L. Faster R-CNN for marine organism detection and recognition using data augmentation. In *Proceedings of the International Conference on Video and Image Processing* (New York, NY, USA, 2017), ICVIP 2017, Association for Computing Machinery, pp. 56–62.

[67] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J., and Le, Q. V. Learning data augmentation strategies for object detection, 2019.