Comparative Analysis of YOLOv5 and YOLOv8 for Automated Fish Detection and Classification in Underwater Environments

Submitted in fulfilment

of the requirements for the degree of

Master of Science

of Rhodes University

Luxolo Kuhlane

Grahamstown, South Africa

April 14, 2024

Declaration of Authorship

I, Luxolo Kuhlane, declare that Comparative Analysis of YOLOv5 and YOLOv8 for Automated Fish Detection and Classification in Underwater Environments is my own work and that it has not been submitted, in whole or in part, for any degree in any other University. I have indicated by reference and acknowledgement the areas that are not my own work.

Abstract

The application of traditional manual techniques for fish detection and classification faces significant challenges, primarily stemming from their labour-intensive nature and limited scalability. Automating these kinds of processes through computer vision practices and machine learning techniques has emerged as a potential solution in recent years.

With the development of and increase in ease of access to new technology in recent years, the use of a deep learning object detector known as YOLO (You Only Look Once) in the detection and classification of fish has steadily become notably popular. This thesis thus explores suitable YOLO architectures for detecting and classifying fish. The YOLOv5 and YOLOv8 models were evaluated explicitly for detecting and classifying fish in underwater environments. The selection of these models was based on a literature review highlighting their success in similar applications but remains largely understudied in underwater environments. Therefore, the effectiveness of these models was evaluated through comprehensive experimentation on collected and publicly available underwater fish datasets.

In collaboration with the South African Institute of Biodiversity (SAIAB), five datasets were collected and manually annotated for labels for supervised machine learning. Moreover, two publicly available datasets were sourced for comparison to the literature. Furthermore, after determining that the smallest YOLO architectures are better suited to these imbalanced datasets, hyperparameter tuning tailored the models to the characteristics of the various underwater environments used in the research.

The popular DeepFish dataset was evaluated to establish a baseline and feasibility of these models in the understudied domain. The results demonstrated high detection accuracy for both YOLOv5 and YOLOv8. However, YOLOv8 outperformed YOLOv5, achieving 97.43% accuracy compared to 94.53%. After experiments on seven datasets, trends revealed YOLOv8's enhanced generalisation accuracy due to architectural improvements, particularly in detecting smaller fish. Overall, YOLOv8 demonstrated that it is the better fish detection and classification model on diverse data.

Acknowledgements

The authors would like to thank the authors of the publicly available datasets used in this research. This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA. The authors would like to also thank the South African Institute of Aquatic Biodiversity for providing the datasets used in this paper. The authors acknowledge that opinions, findings, conclusions, or recommendations expressed here are those of the author(s) and that none of the above-mentioned sponsors accept liability whatsoever in this regard.

I would also like to acknowledge all those who directly and indirectly supported me through my work. In particular, I would like to thank my supervisor, Prof. Dane Brown, for his support, encouragement, and advice throughout the project. Finally, I would like to thank my mother Thandeka Kuhlane, and my friends for their emotional support and encouragement throughout this project.

Contents

List of Figures

List of Tables

Listings

Glossary

AI	Artificial Intelligence
AP	Average Precision
BN	Batch Normalization
CNN	Convolutional Neural Network
COCO	Common Objects in Context (COCO dataset)
DARKNET	The Framework Used for YOLO
DL	Deep Learning
DNN	Deep Neural Network
IoU	Intersection over Union
mAP	Mean Average Precision
ML	Machine Learning
NMS	Non-Maximum Suppression
NN	Neural Network
R-CNN	Region-Based Convolutional Neural Network
SSD	Single Shot MultiBox Detector
YOLO	You Only Look Once
YOLOv5	You Only Look Once version 5
YOLOv8	You Only Look Once version 8

1

Introduction

1.1 Context of Research

Marine research has traditionally relied on manual methods for fish detection and classification that are time-consuming, costly, and susceptible to human error (Misund *et al.*, 2002). These limitations are further compounded by the inherent complexity of underwater environments (Wang *et al.*, 2018). According to Zhang *et al.* (2021), compromised clarity and intricate backgrounds significantly hinder accurate identification and data collection due to the difficulty of distinguishing fish from their surroundings.

Recent strides in computer vision technology, as expounded by researchers such as Zhang *et al.* (2020) and Bochkovskiy *et al.* (2020), represent a promising avenue for addressing challenges inherent in fish detection and classification methodologies. Incorporating computer vision techniques into this domain is a substantial opportunity to overcome limitations associated with traditional methods. This approach not only holds the potential to automate processes, mitigate human error, and optimise data collection efficiency, but it also promises enhanced accuracy.

The importance of advancing fish detection and classification lies in its potential to improve accuracy, minimise human errors, and simplify the comprehension and monitoring of fish populations. Automating this process through computer vision and object detection techniques offers a precise and more accessible means of fish detection and classification.

1.2 Motivation for this Research

The application of traditional manual techniques for fish detection and classification faces significant challenges, primarily stemming from their labour-intensive nature and limited

scalability, as discussed by (Kuswantori *et al.*, 2023). In addressing these challenges, the automation of this process through computer vision practices and machine learning techniques emerges as a potential solution. Within the field of computer vision, the pivotal role of object detection in comprehending image context and facilitating the recognition of visual entities is well-established (Zhang *et al.*, 2022).

In recent years, there has been an increase in the adoption of object detection methods for fish detection and classification. This increase is propelled by the heightened accessibility and advancements in new technologies, as discussed by (Redmon *et al.*, 2015). The efficiency and real-time processing capabilities inherent in YOLO (You Only Look Once) architectures, as underscored by Wang *et al.* (2018), position them as optimal solutions for the comprehensive analysis of images and videos derived from Baited Remote Underwater Video (BRUV).

Furthermore, a gap exists in the current body of literature regarding the assessment of the effectiveness of YOLO architectures in the detection and classification of images and videos acquired through BRUV systems (Li *et al.*, 2023). This research, conducted in collaboration with the South African Institute of Biodiversity (SAIAB) and leveraging publicly available datasets for BRUV footage, evaluates the effectiveness of YOLOv5 and YOLOv8 specifically in the context of detecting and classifying fish in BRUV footage.

1.3 Problem Statement

Accurate detection and classification of fish species in dynamic underwater environments remains a significant challenge, particularly for effective fisheries management and aquaculture. This challenge is further exacerbated by the limitations of traditional methods. The introduction of YOLO-based technology offers a practical solution, alleviating financial burdens associated with labour-intensive methods and addressing errors inherent in human-dependent processes.

Utilising YOLO's precision and efficiency is expected to yield a significant transformation.

This advancement holds promise for cost reduction, enhanced accuracy and the potential to foster sustainable fisheries and thriving aquaculture practices.

This research explores the effectiveness of an automated system designed to maximise the accuracy in detecting and classifying diverse fish species across video datasets. This research aims to accomplish fish detection and classification tasks by employing established and cutting-edge deep learning models, namely YOLOv5 and YOLOv8.

1.4 Research Question

Based on the above, the following overarching research questions can be formulated: *How does the performance of YOLOv5 and YOLOv8 compare in the automated detection and classification of fish?* This can be broken into the following sub-questions:

1. How well does YOLOv8 compare with the more established YOLOv5 algorithm at detecting and classifying fish species?
2. What challenges and limitations arise for YOLOv5 and YOLOv8 in accurately detecting and classifying fish in both complex and non-complex underwater environments?
   (a) with small fish?
   (b) with species diversity and similarity?
   (c) with variety in fish poses and orientations?

1.5 Research Objectives

This research aims to achieve the following objectives:

1. Assess the detection accuracy of YOLOv5 and YOLOv8 in automated fish species recognition using a diverse array of collected and annotated underwater fish datasets, considering real-world and intricate background settings.

2. Compare the precision, recall, and mAP (Mean Average Precision) of YOLOv5 and YOLOv8 in correctly classifying various fish species within the underwater images, especially focusing on their detection accuracy for automated fish species recognition.

3. Analyse the processing speed and efficiency of YOLOv5 and YOLOv8 in real-time fish species detection using a relatively inexpensive system designed for object detection and classification within underwater fish datasets.

4. Examine the performance of YOLOv5 and YOLOv8 in detecting and classifying both common and rare fish species, considering potential imbalances in the training data examine the performance of YOLOv5 and YOLOv8 in detecting and classifying both common rare fish species within the collected and publicly available datasets while addressing potential imbalances in the training data.

1.6 Approach

The overarching approach of this thesis is as follows. The literature assesses diverse machine-learning algorithms, object detection models, and data collection techniques. Identifying the most promising theoretical approaches is grounded in their efficacy with fish-related data and their practical applicability in field deployment.

The performance of the resultant fish detection and classification system is gauged through machine learning model metrics tailored for object detection and classification. Systematic parameter tuning is undertaken to optimise models for each experiment. Experiments are executed on acquired image and video datasets to evaluate the effectiveness of YOLOv5 and YOLOv8 in detecting and classifying various fish species.

1.7 Assumptions

The datasets used in the experiments were chosen based on the following assumptions:

- Sufficient and accurate data can be collected for fish detection, classification, and feature extraction.
- The fish in underwater environments are representative of those found in other regions and can provide insights that apply to other locations.
- The object detection models used in this research will provide accurate and reliable results for fish detection and classification and can be generalised to other species and environments.

1.8 Limitations

- The accuracy and reliability of the object detection models used in this study may be affected by the quality and quantity of available data and the choice of hyperparameters and other configuration settings.
- The environmental conditions of aquatic environments can be unpredictable, which can make data interpretation challenging.
- The models developed in this research may not apply to other aquatic species, environments, or geographical locations, and further research may be required to validate their efficacy in other contexts.
- The computational demands of the different models used in this study may be significant, which can limit their applicability in real-time fish monitoring applications or other resource-constrained environments.

- Limited Availability of Publicly Accessible Fish Datasets: Access to publicly available fish datasets is limited, and some of these datasets may not be readily convertible to the YOLO format. This scarcity of suitable datasets hinders the ability to compare and benchmark the results of this research against a broader range of existing studies and models. Researchers may face challenges in replicating and extending this work due to the lack of comprehensive, standardised fish datasets that are compatible with YOLO, thus limiting the generalisability of the findings.

1.9 Thesis Outline

The remainder of this thesis is arranged as follows: **Chapter 2:** *Concepts and Literature Review*: A broad overview of fish detection and classification is given. The concepts, algorithms, and data in related studies are also discussed and explained.

Chapter 3: *Methods and Materials*: This chapter presents and discusses the proposed approach to create an automated fish detection and classification system. This chapter also explains the implementation details for the proposed fish detection and classification system.

Chapter 4: *Experimental Setup*: This chapter details the experimental setup used in this research. It begins with appropriate metric selection, followed by the design of all experiments conducted, and finally, the breakdown of all the image datasets utilised in the experiments.

Chapters 5: *Results and Discussion*: The validation of the machine learning architecture used in the system implementation is presented, followed by the analysis of results and discussion.

Chapter 6: *Conclusion*: This chapter concludes the thesis, highlights the contributions to the research, and provides directions for future work.

2

Concepts and Literature Review

This chapter presents a broad overview of fish detection and classification systems. It details techniques utilised in related studies, providing insights crucial for the automation of fish detection and . Additionally, it provides an in-depth historical perspective of YOLO (You Only Look Once) and its evolution through various versions since its inception.

2.1 Fish Classification and Detection

In aquatic research, the importance of fish detection and classification using machine learning models is increasingly recognised (Li and Yang, 2018). Traditional methods of monitoring fish populations are often time-consuming, labour-intensive, and prone to human error (Harvey *et al.*, 2013). Automating these methods using machine learning algorithms can efficiently and accurately monitor fish populations over large areas. It can also enable researchers to gather real-time data on various fish (Zhang *et al.*, 2019).

Moreover, automated systems can be more cost-effective than traditional methods, making them more feasible to monitor and manage fish populations over larger areas in underwater environments (Bochkovskiy *et al.*, 2020). Therefore, automated fish detection and classification systems are increasingly needed to manage underwater environments efficiently and effectively. As such, researchers in the field are exploring and developing new machine-learning techniques for automated systems to enhance the understanding of different underwater environments. An object detection model named You Only Look Once (YOLO) has gained significant popularity in computer vision, and many researchers are using it in automating various object detection tasks such as fish detection and classification (Zhang *et al.*, 2020).

The YOLO object detector is a strong candidate for fish detection due to its exceptional blend of real-time efficiency and accurate object classification (Zhang *et al.*, 2020). Its unique "you only look once" approach enables the processing of entire images in a single pass, making it ideal for effective detection in underwater environments (Goodfellow *et al.*, 2014). YOLO's single-pass detection expedites the classification of multiple fish instances (Redmon *et al.*, 2015). Furthermore, its adeptness at handling diverse object sizes and shapes through anchor boxes[1] suits the wide variety of fish poses and orientations.

Moreover, YOLO's multi-class detection capabilities align well with the need to differentiate between different fish species (Goodfellow *et al.*, 2014). Leveraging on the support of an active community, readily available open-source implementations and powerful data augmentation strategies make YOLO a compelling solution for real-world fish detection and classification (Zhang *et al.*, 2021). Section 2.3 details the historical evolution and different architectures of YOLO. The following section introduces the fundamental concept of Convolutional Neural Networks (CNNs), which YOLO incorporates into its architecture.

2.2 Convolutional Neural Networks

CNN is a deep-learning neural network algorithm that utilises convolutional layers (Bengio and Lecun, 1997). The history of CNNs can be traced back to the late 20th century, with the concept of neural networks and convolutional operations gaining prominence. In the 1980s, Yann LeCun introduced the LeNet-5 architecture and demonstrated the effectiveness of convolutional layers as a feature extractor for handwritten digit classification (Bengio and Lecun, 1997). This laid the foundation for the subsequent evolution of CNNs within the field.

CNNs are structured with four fundamental layers, as documented in existing research (Bezdan and Bacanin, 2019). The layers are displayed in Figure 2.1. The network's convolutional layers employ filters to extract local features, such as edges and textures. Moreover, incorporating non-linearity through modern activation functions like Rectified

[1] Anchor boxes are a set of predefined bounding boxes of a certain height and width.

Linear Units (ReLU)[2] enhancing the network's capacity for learning intricate patterns (Krizhevsky *et al.*, 2012). Pooling layers strategically reduce spatial dimensions while preserving critical information. The resultant feature maps undergo flattening or global pooling. This renders them into fully connected layers, where learned features are consolidated such that they can be classified with a final layer.

The output layer typically uses a Softmax activation, which furnishes the final predictions contingent on the nature of the task to quantify the dissonance between predictions and actual targets. Training entails the optimisation algorithm adjusting weights via back-propagation, iteratively. Inspired by the human visual system, this architectural paradigm has demonstrated remarkable efficacy in domains such as image classification and object detection. Contributions by Bengio and Lecun (1997) and Krizhevsky *et al.* (2012) have been instrumental in establishing the widespread adoption of CNNs in computer vision.

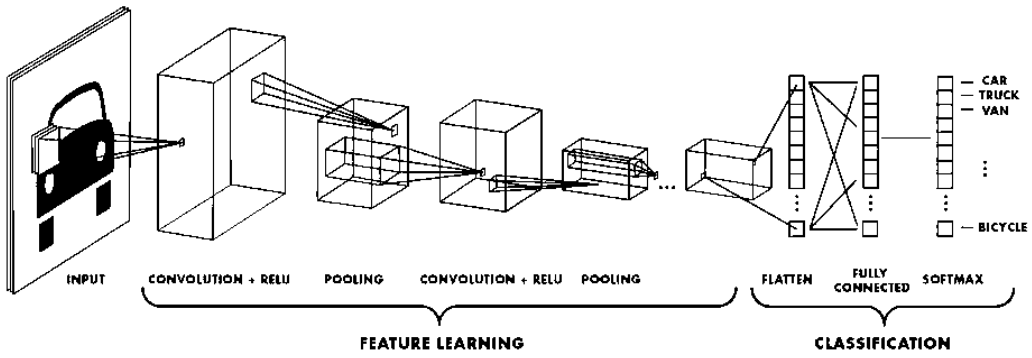


Figure 2.1: The architectural composition of a CNN, showcasing the interconnected layers (Purwono *et al.*, 2023).

Goodfellow *et al.* (2014), Zhao *et al.* (2021), Lee and Youngseop (2020) recognised the importance of combining object detection with image classification as the research in the field progressed. YOLO, introduced by (Redmon *et al.*, 2015), revolutionised object detection by proposing a unified approach that performs classification and localisation in a single pass. YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell (Cong *et al.*, 2019). In the context of YOLO, CNNs are pivotal in handling intricate object detection tasks. The CNN architecture employed

[2]ReLU is an activation function that introduces the property of non-linearity to a deep learning model and solves the vanishing gradients issue. Here's why it's so popular.

in YOLO is designed to process the entire image and predict bounding boxes and class probabilities in a single forward pass.

This approach contrasts traditional methods that often involve multi-stage pipelines for object detection. YOLO divides the input image into a grid, and each grid cell is responsible for predicting bounding boxes and class probabilities. Using CNNs in YOLO enables the model to capture spatial dependencies and hierarchical features, facilitating robust object detection across various scales and categories (Lee and Youngseop, 2020).

2.3 History on YOLO Algorithms

YOLO is a developing family of real-time object detection algorithms that have reshaped the landscape of computer vision (Redmon *et al.*, 2015). YOLO was first introduced by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous research paper titled *You Only Look Once: Unified, Real-Time Object Detection* (Redmon *et al.*, 2016). The algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals (Redmon *et al.*, 2015). Figure 2.2 presents the YOLO detection process from assigning a grid to activation maps[3] and suppressing duplicates.

The YOLO algorithm has redefined real-time object detection by leveraging CNN. Unlike its predecessor, such as the Deformable Parts Model (DPM)[4] (Ruan *et al.*, 2014), which follows a two-step process, YOLO performs detection in a single forward pass through the neural network.

Traditionally, object detection frameworks like DPM identified regions of interest in the first step and then passed them to a more robust classifier for classification in the second step. This iterative approach involved multiple passes over an image. YOLO was created

[3]Activation maps indicate the salient regions of an image for a particular prediction.

[4]DPM is a classification algorithm in which the image is segmented into several parts, each classified using specialised classifiers.

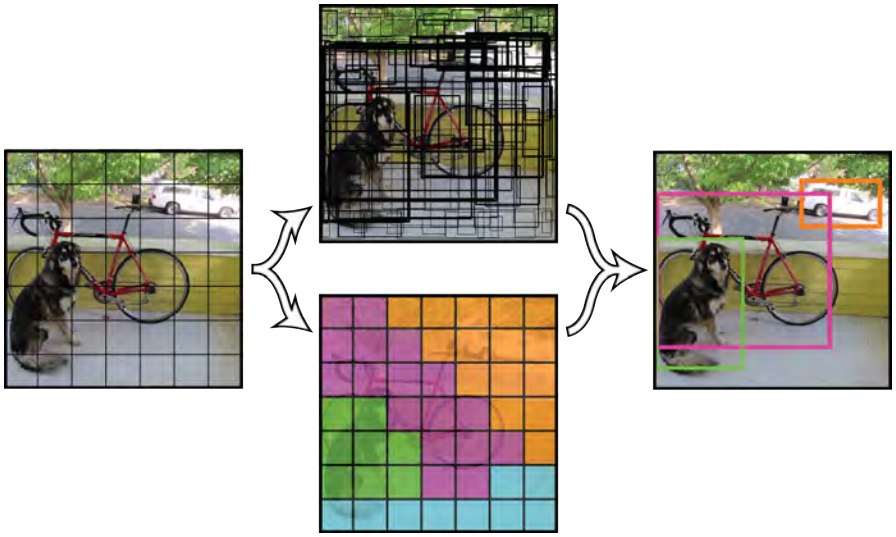


Figure 2.2: The YOLO detection system (Purwono *et al.*, 2023)..

out of the need for optimisation and sought to address this inefficiency by providing both detection and class information in a single-pass algorithm run.

In the YOLO framework, the CNN predicts the entire image in one go, estimating class probabilities for various objects. This not only streamlines the detection process but also enhances real-time capabilities. The architecture depicted in Figure 2.2 illustrates the simplicity and effectiveness of the YOLO detection system in detecting an image.

The emergence of the YOLO lineage represents a significant evolution in object detection within computer vision. The following sub-sections comprehensively explore the progression of YOLO from YOLOv1 to the latest model in the lineage YOLOv8 as of 2023, providing details on each iteration's main features, advancements, and performance improvements.

2.3.1 YOLOv1

The introduction of YOLOv1 in 2015 (Redmon *et al.*, 2015) marked a breakthrough in real-time object detection, utilizing a single neural network to predict object bounding boxes and class probabilities. Inspired by architectures like GoogLeNet (Szegedy *et al.*,

2015) and Network in Network (Lin *et al.*, 2013), YOLOv1 employed 1×1 convolutional layers to reduce feature map dimensionality, enabling efficient object detection.

YOLOv1 divides input images into an S × S grid, with each grid cell predicting B bounding boxes and C class probabilities. Bounding box predictions consist of P_c, bx, by, bh, and bw, representing confidence, center coordinates, and dimensions. The model's output is a tensor of dimensions $S \times S \times (B \times 5 + C)$, followed by non-maximum suppression for redundant detections. A comprehensive loss function balances localization, confidence, and classification errors, with λ_{coord} and λ_{noobj} regulating loss contributions.

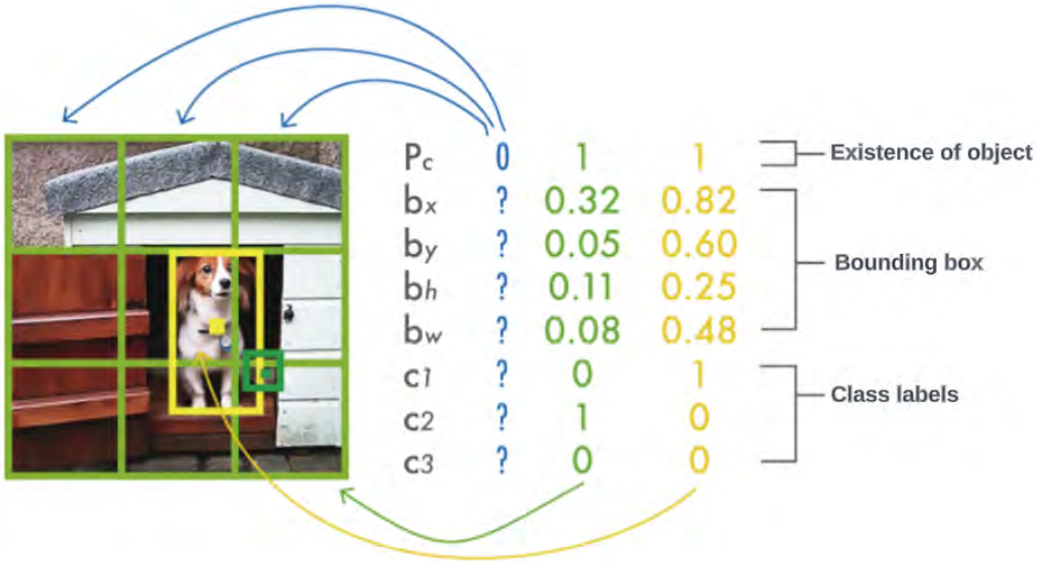


Figure 2.3: The illustration represents a simplified YOLO model featuring a three-by-three grid configuration. This model is designed for three classes, and it generates a single class prediction for each grid element, resulting in an output vector consisting of eight values (Redmon *et al.*, 2015).

The loss function includes terms for localization, confidence, and classification. Localization loss penalizes errors in bounding box predictions, with confidence loss accounting for detection accuracy. Classification loss measures class probability errors, prioritizing predictions with detected objects.

2.3.2 YOLOv2

YOLOv2, introduced by Redmon *et al.* (2016), improved upon YOLOv1 by incorporating several key enhancements. Notable improvements include batch normalization on convolutional layers for improved convergence and regularisation, and the use of a fully convolutional architecture for more efficient processing. Additionally, YOLOv2 implemented anchor boxes for bounding box prediction, enabling the model to predict multiple boxes per grid cell, and introduced dimension clusters for selecting prior boxes to balance recall and model complexity.

The model also adopted direct location prediction, predicting bounding box coordinates relative to the grid cell, and utilized finer-grained features by removing a pooling layer and introducing a pass-through layer for preserving spatial information. Multi-scale training was employed to enhance robustness to varying input sizes. With a foundational architecture based on Darknet-19, which features 19 convolutional layers and five max-pooling layers, YOLOv2 achieved significant improvements in performance, achieving an average precision (AP) of 78.6% on the PASCAL VOC2007 dataset compared to 63.4% for YOLOv1.

2.3.3 YOLOv3

Published in 2018 by Redmon and Farhadi (2018), YOLOv3 introduced significant enhancements to its predecessor. It predicted four coordinates and an objectness score for each bounding box, enabling more precise localisation. YOLOv3 utilized binary cross-entropy for classification, facilitating multi-label classification and handling datasets with overlapping labels.

The model featured a larger backbone architecture, Darknet-53, comprising 53 convolutional layers with residual connections, and incorporated spatial pyramid pooling (SPP) blocks for improved performance. Additionally, YOLOv3 adopted a multi-scale prediction approach, predicting three boxes at different scales, and utilised k-means clustering

to determine bounding box priors for anchor boxes. The multi-scale prediction architecture of YOLOv3 involved concatenating feature maps from different scales and applying up-sampling operations to align dimensions. This innovation significantly enhanced the model's ability to detect small objects and achieve more detailed bounding boxes.

2.3.4 YOLOv4

Published in April 2020 by Bochkovskiy *et al.* (2020), YOLOv4 introduced significant improvements while adhering to the YOLO philosophy of real-time, single-shot detection within the darknet framework. The model incorporated a balanced approach of *bag-of-freebies* and *bag-of-specials* to enhance both training strategies and inference accuracy.

The key changes in YOLOv4 included an enhanced architecture featuring a modified Darknet-53 backbone with CSPNet and Mish activation, alongside innovations in the neck and detection head modules. Additionally, advanced training approaches such as mosaic augmentation, DropBlock regularization, CIoU loss, and Cross mini-batch normalisation were integrated.

To enhance robustness, YOLOv4 introduced self-adversarial training (SAT), where adversarial attacks were applied to input images during training. Moreover, the model's hyperparameters were optimised using genetic algorithms, further improving training efficiency.

The architecture of YOLOv4 comprises several key modules, including CMB (Convolution + Batch Normalization + Mish activation), CBL (Convolution + Batch Normalization + Leaky ReLU), UP (up-sampling), SPP (Spatial Pyramid Pooling), and PANet (Path Aggregation Network). These modules work together to enable efficient and accurate object detection by capturing complex patterns, enhancing feature representation, and aggregating spatial information.

2.3.5 YOLOv5

YOLOv5 (Nelson and Solawetz, 2020) leverages algorithm optimisation techniques drawn from the CNN domain. These optimisations encompass adaptive learning of bounding box anchors, mosaic data augmentation, and utilisation of the cross-stage partial network (Nelson and Solawetz, 2020). Distinguishing itself from previous approaches, the YOLO model marked a pioneering advancement by seamlessly integrating the process of predicting bounding boxes and class labels within a single, end-to-end differentiable network, as explained by Nelson and Solawetz (2020).

As shown in Figure 2.4 the YOLOv5 network consists of three main components: the model backbone, the model neck, and the model head. The input terminal performs the pre-processing tasks, including mosaic data augmentation (Solawetz and Francesco, 2023). Mosaic data augmentation combines four training images into one certain ratio. The model backbone is a pre-trained network that extracts rich feature representation for images. This component helps reduce the image's spatial resolution and increases feature resolution.

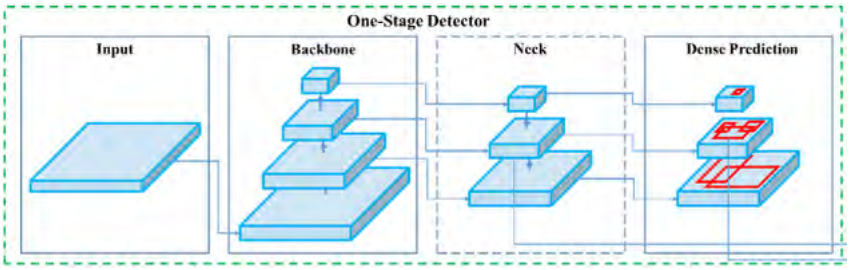


Figure 2.4: YOLOv5 single-State detector architecture (Bochkovskiy *et al.*, 2020).

The model neck is used to extract feature pyramids. This component helps the model generalise well to objects of different sizes and scales (Nelson and Solawetz, 2020). To adapt to different datasets, YOLOv5 integrates adaptive anchor frame calculation on the input to automatically set the initial anchor frame size when the dataset changes. The backbone is a CNN that aggregates and forms features. The following sections explain the backbone in more detail.

The backbone of YOLOv5 architecture plays a vital role in feature extraction from input

images, as shown in Figure 2.4. The backbone is essential because it is crucial for accurate detection. The YOLOv5 backbone architecture is based on CSPDarknet53, an enhanced version of Darknet architecture. CSPDarknet53 combines the following key components: CSPNET and Darknet-53.

CSPDarkNet53 is a neural network architecture that stems from the family of Darknet neural networks. Darknet is an open-source neural network framework written in C and CUDA, primarily developed by Joseph Redmon. DarkNet was recognised for being the framework behind the YOLO series of object detection models, including YOLOv3, widely used for real-time object detection in various applications. It introduces cross-stage connections that allow features from different stages of the network to be efficiently combined. CSPNet introduces cross-stage connections within the neural network architecture. These connections facilitate the flow of information between different stages or blocks of the network.

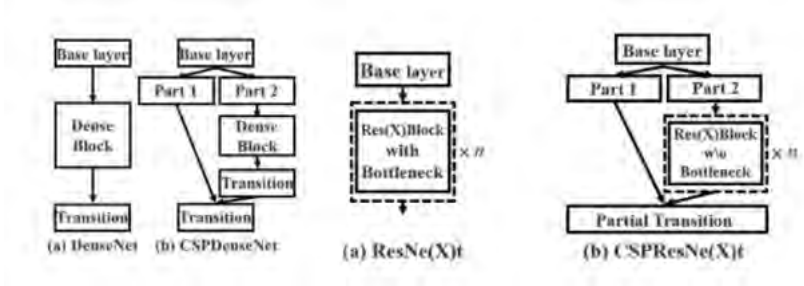


Figure 2.5: YOLOv5 cross-stage partial network architecture.

As shown in Figure 2.5, the CSP network preserves the advantage of DenseNet's feature and helps reduce the excessive amount of redundant gradient information by truncating the gradient flow (Bochkovskiy *et al.*, 2020). Applying this strategy comes with a significant advantage for YOLOv5 since it helps reduce the number of parameters and helps reduce an essential amount of computation, which leads to increasing the inference speed, a crucial parameter in real-time object detection models.

The CSPNET and Darknet-53 facilitate extracting rich hierarchical features from input images, which are then used for subsequent object detection tasks. It is a crucial component contributing to the model's ability to detect objects accurately and efficiently.

In YOLOv5, significant modifications have been made to the model's neck, explicitly addressing the Spatial Pyramid Pooling (SPP) module and integrating the BottleNeckCSP into the Path Aggregation Network (PANet). The PANet, initially employed in YOLOv4, underwent alterations in YOLOv5, incorporating the CSPNet strategy, as depicted in the architectural diagram (Figure 2.6).

The introduced SPPF block, visualised in Figure 2.6, enhances information aggregation by combining inputs to produce a fixed-length output. This configuration significantly expands the receptive field, capturing crucial contextual features while maintaining network speed. YOLOv5 optimises speed and feature extraction by adopting SPPF, a variant of the SPP block utilised in prior YOLO versions.

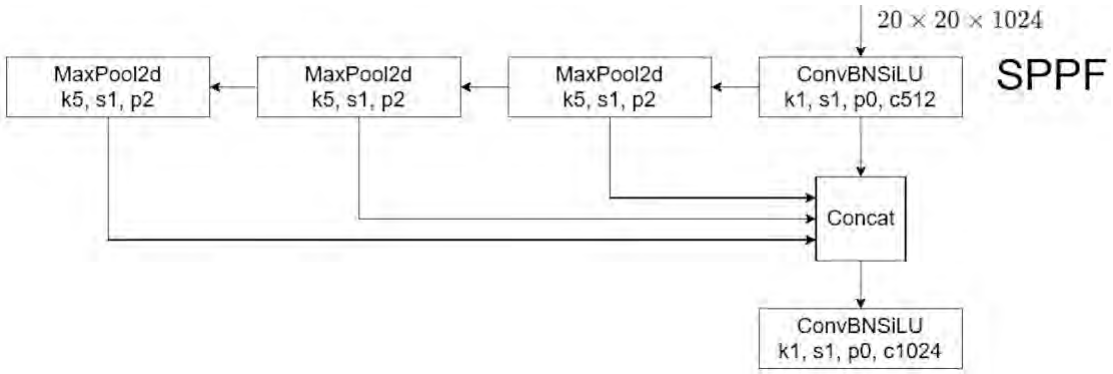


Figure 2.6: Structure of the SPPF Block.

The decision to choose YOLOv5 for fish detection is driven by its exceptional performance, real-time inference capabilities crucial for underwater environments, and widespread popularity. YOLOv5's reliability and extensive research contributions make it a powerful tool for advancing automated fish species detection and classification, with ample resources available for further refinement and customisation.

2.3.6 YOLOv6

Published in September 2022 by Liu *et al.* (2016), YOLOv6 introduced several innovations tailored for industrial applications. Departing from its predecessors, YOLOv6 adopts

an anchor-free detection approach, aiming to align with contemporary trends in object detection methods.

Key innovations of YOLOv6 include the EfficientRep backbone architecture, based on RepVGG, known for its efficiency and scalability. Task Alignment Learning (TAL) ensures label consistency across different scales and resolutions, enhancing object detection accuracy. Additionally, YOLOv6 incorporates new loss functions, self-distillation strategy, and a quantization scheme to improve performance and efficiency.

On the MS COCO dataset, YOLOv6-L achieved impressive performance with an Average Precision (AP) of 52.5% and an AP50 of 70% while maintaining an operational speed of approximately 50 frames per second (FPS) on an NVIDIA Tesla T4.

2.3.7 YOLOv7

YOLOv7, as reported by Sun *et al.* (2023), set new benchmarks in object detection, outperforming existing detectors in both speed and accuracy across a wide range of frame rates (5 FPS to 160 FPS). Trained exclusively on the MS COCO dataset without pre-trained backbones, YOLOv7 introduced architectural changes and bag-of-freebies, improving accuracy while maintaining inference speed, albeit with increased training time.

Key architectural changes in YOLOv7 include the Extended Efficient Layer Aggregation Network (E-ELAN) for efficient learning, model scaling for concatenation-based models, Planned Re-parameterised Convolution (RepConvN), coarse label assignment for auxiliary heads, and fine label assignment for lead heads. Batch Normalisation in Conv-BN-Activation and incorporation of implicit knowledge from YOLOR further enhance model performance. Additionally, YOLOv7 adopts Exponential Moving Average (EMA) as the final inference model. On the MS COCO dataset test-dev 2017, YOLOv7-E6 achieved impressive performance with an Average Precision (AP) of 55.9% and an AP50 of 73.5%, while maintaining a speed of 50 frames per second (FPS) on an NVIDIA V100.

2.3.8 YOLOv8

YOLOv8 is a new state-of-the-art deep learning model used for object detection, image classification, and instance segmentation tasks (Solawetz and Francesco, 2023). YOLOv8 was developed by Ultralytics, who also created the influential and industry-defining YOLOv5 model. YOLOv8 includes numerous architectural and developer experience changes and improvements from YOLOv5. Figure 2.7 is the model structure of YOLOv8 and how it is implemented.

YOLOv8 does not have a seminal paper by the authors, so it lacks insight into the direct research methodology and ablation studies done during its creation (Solawetz and Francesco, 2023). The YOLOv8 architecture, similar to the architecture of YOLOv5, consists of three main components: the backbone, the neck, and the head. The backbone is responsible for extracting features from the input image. It is a modified version of the CSPDarknet53 architecture, a popular backbone architecture for object detection models. The modified version of CSPDarknet53 used in YOLOv8 has additional features that improve its performance, such as cross-stage partial connections (CSP) and a Squeeze-and-Excitation (SE) block.

The neck is responsible for connecting the backbone to the head. It consists of a series of convolutional layers that down-sample the feature maps from the backbone and then up-sample them to the original resolution. This allows the head to process features at various scales, which is important for object detection. The head predicts the bounding boxes, objectness scores, and class probabilities for the objects detected in an image. It consists of a series of convolutional layers followed by a series of fully connected layers.

In addition to these main components, YOLOv8 also uses several other techniques to improve its performance, such as:

- **Mosaic Augmentation**: This technique combines four images to create a new image. This helps the model learn to detect objects in different contexts and scales.

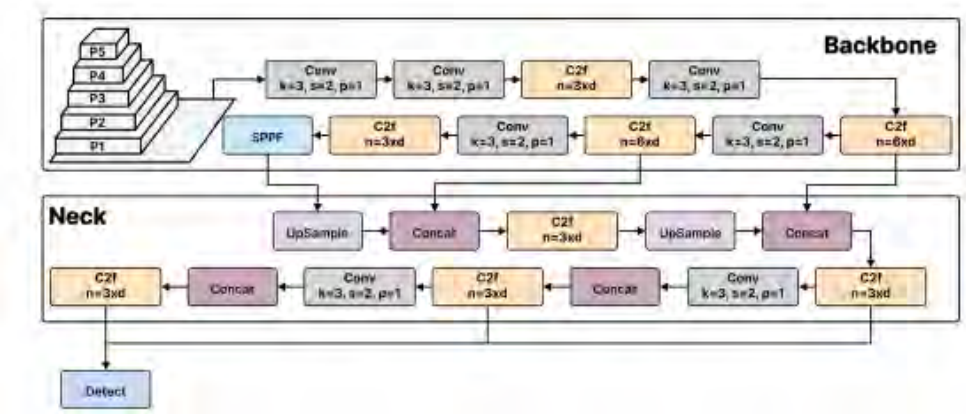


Figure 2.7: YOLOv8 architecture consists of 53 convolutional layers and employs cross-stage partial connections to improve information flow between the different layers. The head of YOLOv8 consists of multiple convolutional layers followed by a series of fully connected layers. Figure adapted from (Solawetz and Francesco, 2023).

- **Self-Attention**: This technique allows the model to learn to focus on the most important features in an image. This helps to improve the accuracy of the detection.
- **Label Smoothing**: This technique helps prevent the model from overfitting to the training data. This is done by adding noise to the labels during training.

The YOLOv8 architecture is a powerful and efficient architecture for object detection. It has been shown to achieve state-of-the-art results on various object detection benchmarks. It is a versatile, complex architecture that can be used for a wide range of object detection tasks. Fish are often small and have a lot of appearance variation, making them difficult to detect. YOLOv8 is designed to handle small objects and effectively detect fish, even in cluttered environments.

2.4 Related Studies

While object detection has succeeded in datasets encompassing generic categories, it continues to pose a challenge in underwater object detection. Within underwater settings, the quality of captured images is significantly compromised by illumination conditions, resulting in diminished visibility, decreased contrast, and colour distortion. Additionally,

the intricate underwater environment, coupled with the presence of relatively diminutive underwater entities, significantly heightens the difficulty associated with detecting submerged targets.

2.4.1 Fish Detection and Classification Systems

A novel CNN architecture comprising 22 layers has been proposed for the accurate and expeditious classification of coral reef fish within underwater images (Villon *et al.*, 2018). The training dataset utilised in this study encompassed an extensive repository of roughly over 900,000 images, which included comprehensive fish body representations, partial fish body instances, as well as environmental context factors, such as characteristics of the aquatic habitat, e.g. reef bottom or water. The CNN model could recognise fish partly concealed behind coral or other fish and was better than humans at identifying fish in unusual positions, e.g. twisted bodies. This research methodology proves to be instrumental in precisely classifying fish species within underwater images. It can potentially develop innovative video-based protocols for the cost-effective and efficient monitoring of fish biodiversity.

Duggal *et al.* (2017) developed a model that could automatically describe video content using object detection algorithms. They recognised that while video content description is a relatively simple task for humans, it presents a complex and challenging problem for computers. The system they proposed was constructed on the foundation of the YOLO object detection algorithm. Their model stood out by surpassing the performance of two other models, offering greater speed and reduced memory overhead. Utilising the YOLO object detection algorithm, which will also be applied for fish detection and counting, their research offers valuable insights into the advancement of object detection in videos. These insights have the potential to contribute to the refinement of YOLOv5 and YOLOv8 models for fish detection and classification, ultimately enhancing their ability to interpret dynamic scenes effectively.

Singh and Mukhopadhyay (2017) implemented tracking of moving objects based on optical flow was introduced. Tracking objects in complex scenes, especially determining the

contour of an object, is challenging. To address this, the researchers utilised an algorithm to calculate the velocity vector and subsequently tracked the object's contour by identifying the position of moving pixels across frames. This approach allowed them to determine the object's position and speed based on the gathered position data. Their findings demonstrated the accurate tracking of objects by the camera system. This research provides valuable insights into object tracking techniques, which could be applied to enhance detection and classification mechanisms using YOLOv5 and YOLOv8 in the context of underwater environments.

Kuhlane *et al.* (2023b) utilised YOLOv7 to detect and track bull sharks in the waters of Mozambique. Through the implementation of YOLOv7, the researchers successfully identified bull sharks, achieving a commendable outcome of 88.31% accuracy on low-resolution images and an impressive 97.42% accuracy on high-resolution images. Their research also encompassed image enhancement tools to enhance image quality and facilitate a comparative analysis of detection and tracking results between low-resolution and high-resolution images. A notable limitation of this endeavour pertains to the restricted visibility of images, which could potentially influence the robustness of the detection and tracking processes. Nevertheless, this study underscores the pivotal role of image enhancement techniques, offering promising prospects for enhancing the performance of YOLOv5 and YOLOv8 in demanding aquatic environments.

Kuhlane *et al.* (2023a) investigated the efficacy of successive YOLO model iterations, with a specific focus on YOLOv5, for detecting and classifying great white sharks in Cape Town. The research yielded significant improvements in detection accuracy and classification performance using the YOLOv5 model, demonstrating its ability to efficiently and accurately track and identify great white sharks in real-world conditions. However, the study also recognised limitations, including the need for further exploration of the model's performance in varying environmental conditions, adaptability to other shark species, and potential challenges posed by limited visibility in aquatic environments. These findings offer promising prospects for shark monitoring and conservation efforts while highlighting areas for further research and model refinement.

Kuhlane *et al.* (2023c) experimented with the YOLOv5 model for real-time detection and tracking of squids. The study showcases the YOLOv5 model's capabilities inefficiently and swiftly classifying and tracking squids in various environments, emphasising its real-time applicability. However, limitations of the research include the need for further investigation into the model's performance under different lighting and underwater conditions and its adaptability to various squid species. Additionally, the potential impact of occlusions and varying squid behaviours on tracking accuracy warrants further exploration. Addressing these limitations could lead to more comprehensive and robust applications of YOLOv5 in squid detection and tracking.

Patel *et al.* (2012) employed the then state-of-the-art YOLO9000 model to detect small fish populations in the Indian Ocean. The network was trained on a diverse set of image data extracted from images and videos captured by scientists conducting research in the Indian Ocean. The model achieved an impressive overall total recall of 93.23% in accurately classifying fish within their dataset. This suggests that newer iterations of YOLO likely exhibit even higher recall rates when compared to YOLO9000.

Rodriguez *et al.* (2015) addressed the study of biological changes in fish, particularly changes in size, using stereo systems and image processing algorithms. A primary challenge in this research was accurately estimating fish size in a pond, which could provide valuable insights into various fish-related factors. The researchers employed an image processing algorithm to detect fish based on the distance map obtained from a stereo-vision system. Their research yielded a 10% error rate in fish size estimation but demonstrated a high precision rate of 90%.

An algorithm was proposed by Nguyen *et al.* (2018) to enhance the tracking of fish movement. Their tracking difficulties included fish that appeared to be moving but were not, fish that were static, and fish that moved at different rates at different times. To address these issues, they proposed a technique combining frame difference and Gaussian mixture techniques. As it follows fish in different situations, their proposed approach outperforms Toh *et al.* (2009)'s four systems.

Wang and Xia (2023) introduced an enhanced CNN method for fish detection in under-

water images, addressing challenges posed by complex aquatic environments. The study presents a tailored CNN architecture capable of extracting discriminative features from underwater imagery, improving fish detection accuracy.

Huang *et al.* (2018) proposed fish detection using CNNs. The research outlines a novel CNN-based approach to enhance the accuracy and efficiency of fish recognition within aquatic environments. The method focuses on effectively learning relevant features from underwater imagery, thereby contributing to improved fish detection capabilities. In the context of employing YOLOv5 and YOLOv8 for fish detection and classification in underwater environments, this paper's findings resonate as they underscore the potential of CNNs to enhance the feature extraction process crucial for the accurate detection and classification of fish.

Li and Yang (2018) presented a comprehensive investigation into fish detection and classification within underwater images using CNNs. The research introduces a CNN-based methodology to effectively identify and categorise fish species within the complex underwater environment. By harnessing the capabilities of deep learning, the study aims to enhance the precision and accuracy of fish analysis. They demonstrated notable advancements in fish detection and classification accuracy, showcasing the potential of CNNs in accurately distinguishing and categorising fish species within underwater imagery.

Boom *et al.* (2012) delved into the effects of pollution and climate change on the environment. Long-term underwater environment monitoring, like other data collection methods, demands substantial labour. The researchers devised a system to monitor and identify fish using their colour and other distinguishing features. Although the system is not yet fully operational, it currently exhibits a detection and classification rate of 79.8% and a false detection rate of 11.8%. Nevertheless, the methodology of classifying fish based on their characteristics could offer valuable insights into improving the capability of YOLOv5 and YOLOv8 for fish detection and classification under diverse conditions.

The motivation for Lumaug and Neva (2018) was to rely on computer vision to count fish because manual counting is difficult. The issue with manual fish counting is that it takes a long time and causes eye strain. The researchers employed image processing

techniques (blob analysis and euclidean filtering) to automate the process of counting fish. The system had many problems with over-counting and under-counting the fish. Over-counting was caused due to lighting conditions. This paper is useful as a good basis for counting fish from the same camera position. Despite these challenges, the paper offers a foundation for automating fish counting, which could improve the fish detection and classification capabilities of YOLOv5 and YOLOv8 in underwater environments.

The research presented in this collection of studies gives a treasure trove of practical approaches and insights that can help investigate fish species detection and tracking in underwater environments using YOLOv5 and YOLOv8. These studies stress the importance of having diverse and comprehensive training data, nailing down precise object segmentation, building efficient object tracking methods, and using image enhancement tricks - all while taking into account the unique challenges of real-life underwater scenarios by drawing from and building on the lessons learned in these studies. This could mean better accuracy, smoother classification, and more reliable object recognition when dealing with complex underwater ecosystems and often tricky conditions. These improvements could open the door for new and effective ways to monitor and protect fish species.

2.4.2 Problems in Underwater Detection and Classification

This section delves into the intricate world of underwater fish detection research, highlighting the challenges and obstacles researchers encounter to identify and monitor fish populations beneath the surface.

Er *et al.* (2022) discussed the challenges, recent advances, and benchmark datasets in deep-learning-based underwater marine object detection. The authors identify four main challenges in underwater marine object detection: image quality degradation, small object detection, poor generalisation, and real-time detection. They then review recent advances in deep learning-based underwater marine object detection, including image enhancement and restoration techniques, deep learning architectures, data augmentation techniques, and lightweight deep learning models. The authors also review the most extensively used

benchmark datasets for underwater marine object detection. The authors suggest that future research in deep learning-based underwater marine object detection should focus on developing more robust and efficient deep learning models, as well as collecting more diverse and high-quality training datasets.

Zhang *et al.* (2023) encountered several challenges in developing a robust fish detection model for real-world underwater environments. One challenge was the poor image quality of underwater images, which can be degraded by light attenuation, scattering, and absorption, as well as by suspended particles and noise. To address this challenge, the authors used various image enhancement techniques and trained their model on a large dataset of underwater images. Another challenge was the camouflage of fish, making them difficult to distinguish from the background. The authors used a data augmentation technique called random colour jittering to help the model learn to detect fish with various camouflage patterns. The authors also addressed occlusion challenges, dense biological distribution, and dynamic background. They used a technique called non-maximum suppression (NMS) to remove overlapping bounding boxes, anchor boxes to detect fish of different sizes and configurations, and a technique called temporal information aggregation to improve the model's ability to detect fish in dynamic environments.

Kuswantori *et al.* (2023) developed an optimised YOLO algorithm for fish detection and classification in an automatic sorting system. The authors identified the following challenges in developing such a system: fish often move quickly and erratically, fish can be occluded by other fish or objects in the environment, fish can vary in size, shape, and colour, and the system must operate in real-time. To address these challenges, the authors proposed several improvements to the original YOLO algorithm, including a new loss function, anchor box selection mechanism, and data augmentation scheme. The authors evaluated their proposed algorithm on a real-world dataset of fish images and achieved state-of-the-art performance. In addition to the above challenges, the authors also discussed the need for a large and diverse dataset of fish images to train the model, a computationally efficient model that can run on real-time hardware, and a model that is robust to environmental variations. The authors proposed the following solutions to these challenges: a data augmentation scheme to increase the diversity of the training dataset

artificially, a lightweight backbone network for the model, and training the model on a large and diverse dataset of fish images collected in different environments.

In recent research on underwater fish detection and classification, a trio of pivotal studies has shed light on the challenges and solutions in improving the performance of YOLOv5 and YOLOv8 models. Er *et al.* (2022) investigated the difficulties in underwater marine object detection, emphasising image quality degradation, small object detection, poor generalisation, and real-time processing as primary obstacles. They explore advancements in deep learning, including image enhancement and data augmentation techniques, underlining the importance of diverse training datasets. Zhang *et al.* (2023) tackled poor image quality issues and fish camouflage using image enhancement and data augmentation. Their approach encompassed techniques like non-maximum suppression and anchor boxes to address occlusion and dynamic backgrounds. In a separate endeavour, Kuswantori *et al.* (2023) devised an optimised YOLO algorithm for fish detection in real-time sorting systems, focusing on the challenges of fish movement, occlusion, size, and shape variations. Their improvements included a novel loss function and data augmentation scheme, achieving state-of-the-art performance. Additionally, they stressed the need for a diverse dataset and computational efficiency in model design. Collectively, these studies contribute valuable insights for enhancing fish detection and classification in underwater environments with YOLO models.

2.4.3 Public Datasets

Garcia-D'Urso *et al.* (2022) introduced a new dataset of images of fish. The dataset contains pixel-wise (mask) labelled specimens, species information, and different size measurements. The authors evaluated the performance of two different methods of fish instance segmentation and size estimation of the DeepFish dataset. They used Mask R-CNN and PointNet++. The finding was that Mask R-CNN achieved a mean intersection over union (IoU) of 72.6% for fish instance segmentation and a mean absolute error of 1.6 cm for fish size estimation. PointNet++ achieves a mean IoU of 69.7% for fish instance segmentation and a mean absolute error of 1.8 cm for fish size estimation. The authors conclude

that the DeepFish dataset is valuable for developing fish instance segmentation and size estimation methods. Mask R-CNN is a promising approach for these tasks.

Liang and Song (2023) proposed a lightweight marine biological target detection algorithm based on the YOLOv5 object detection framework. The algorithm is designed to be efficient and accurate for detecting marine biological targets in underwater images. The algorithm was evaluated on a dataset of underwater images containing marine biological targets. The results showed that the algorithm achieved a mean average precision (mAP) of 85.7% (Liang and Song, 2023).

This is comparable to the performance of other state-of-the-art marine biological target detection algorithms. The tool used in the paper is the YOLOv5 object detection framework. The YOLOv5 framework is a fast and accurate object detection framework well-suited for marine biological target detection. DeepFish is a versatile benchmarking dataset for comparing YOLOv5 and YOLOv8 on fish detection.

Recognising the inherent complexities of the DeepFish dataset, the outcomes of the experiments by Liang and Song (2023) will contribute significantly to a comprehensive understanding of the strengths and weaknesses exhibited by these models. The paper by Fisher *et al.* (2016) evaluated the performance of the YOLOv2 object detection framework on the Fish4Knowledge dataset, which contains over 100,000 images of fish from coral reefs. The authors found that the YOLOv2 model achieved a mAP of 65%. They also found that the performance of the YOLOv2 model could be improved by using a larger and more diverse dataset.

The findings by Fisher *et al.* (2016) suggest that YOLOv2 is a promising approach for fish detection but struggles with smaller fish. The model's performance can be further improved by using a larger and more diverse dataset and newer methods. In the context of this research, Fisher's dataset will be used to evaluate two contemporary object detection frameworks, namely YOLOv5 and YOLOv8. The hypothesis posits that YOLOv5 and YOLOv8 are expected to exhibit superior performance compared to YOLOv2 when applied to the *Fish4Knowledge* dataset. Additionally, the proposed research will conduct

experiments involving various hyperparameters to enhance the models' performance and achieve high accuracies on several public and collected datasets.

2.5 Summary

This chapter provides an overview of the existing literature to gauge the current state of fish detection and classification. It details the significance of accurate detection, emphasising the potential for automation through the application of deep learning. Such models are important for accurate fish detection and classification, particularly in challenging underwater environments.

The history of YOLO was explored in detail, including the different versions of YOLO throughout the years. According to the literature, the YOLO model emerges as the most advanced and recommended choice for fish detection and classification systems. However, the older model, YOLOv2 and YOLOv3 have been widely adopted in related literature. Considering the literature findings, it is suggested that exploring newer YOLO architecture could lead to performance and accuracy enhancements.

Addressing challenges in underwater detection and classification, the chapter highlights the most common issue of poor image quality and inadequate lighting due to varying water conditions. Moreover, the exploration of publicly available datasets for underwater detection and classification showed the challenges encountered in each dataset. Some datasets are not accessible to the public, while others present compatibility issues with YOLO models.

3

Methods and Materials

This chapter presents the methodological approach and successive stages involved in developing an automated fish detection and classification system utilising two YOLO models: YOLOv5 and YOLOv8. Furthermore, this chapter explains the system's infrastructure, covering both hardware and software aspects of the research.

3.1 Methodology Overview

The proposed system's methodology focuses on selecting the most suitable YOLO model architecture for optimal performance in detecting and classifying fish in underwater environments provided in each dataset. Hyperparameter tuning is also carried out to optimise tasks trained on both models. Figure 3.1 presents the proposed fish detection and classification system. The overview outlines the workflow across multiple stages of the system.

The primary stage involves the process of data collection. The datasets used in this research were a total of seven datasets, with two publicly available and five collected datasets. The collected datasets were custom-annotated in collaboration with the SAIAB. The next process involved data construction, converting the datasets into YOLO formats.

Following the conversion, the research conducted a YOLO architecture selection process. The process involved selecting the most suitable YOLOv5 and YOLOv8 versions for this research. Proceeding with the architecture selection phase, the system's core functionality was executed by employing both model architectures to detect and classify fish within the datasets.

Following this operational phase, a comprehensive evaluation was conducted utilising established object detection and classification metrics, namely mean Average Precision

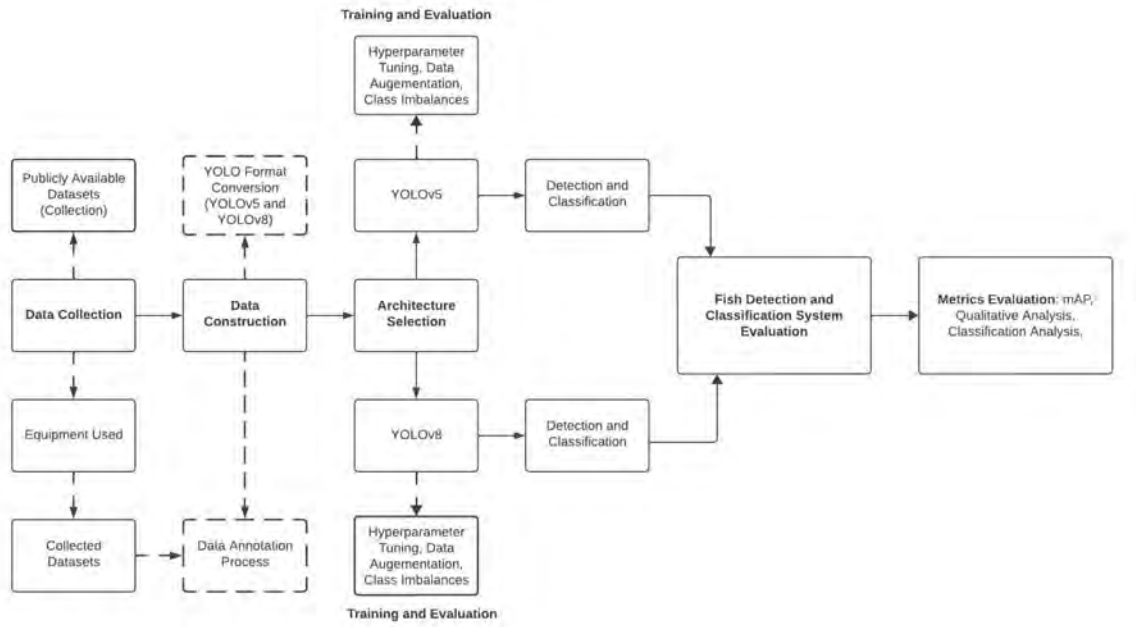


Figure 3.1: Overview of the proposed fish detection and classification system implemented.

(mAP), classification analysis, class-specific classification and qualitative assessments through visual inspection. The following sections provide in-depth details on the methodological approaches.

3.2 Data Collection

This section provides a comprehensive account of data collection and preparation procedures undertaken for the research on automated fish species detection and classification in underwater environments. The image samples for each dataset are shown in Appendix A for the interested reader.

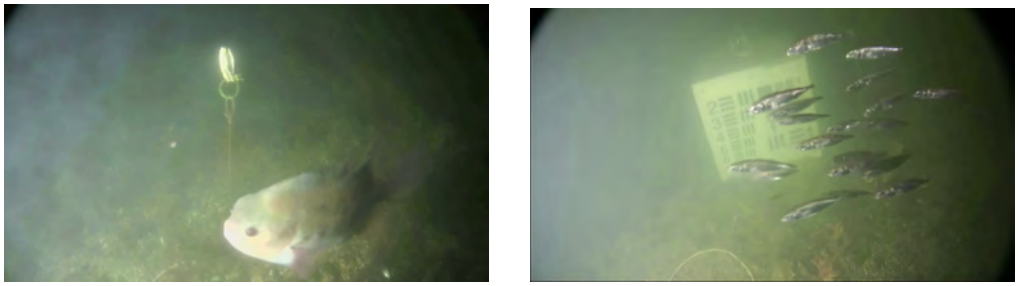
3.2.1 Public Datasets

The following subsections specify the attributes of the datasets, namely the fish datasets, geographic regions, and acquisition methods.

3.2.1.1 Brackish Dataset

The Brackish dataset is an open-source underwater dataset created in 2019 by Pedersen *et al.* (2019). The annotated version of this dataset, tailored for object detection, was sourced from Zhang *et al.* (2023). This dataset has gained recognition in the research community as one of the first underwater detection datasets captured outside tropical water environments.

The dataset comprises recordings from nine meters beneath the water's surface in brackish environments, totalling 89 videos. The videos are annotated with bounding boxes that categorise objects into six classes: *fish*, *crab*, *shrimp*, *starfish*, *small fish*, and *jellyfish*.



(a) fish **(b)** small fish

Figure 3.2: Image samples from the Brackish Dataset (Pedersen *et al.*, 2019).

3.2.1.2 DeepFish Dataset

Twenty habitats from remote coastal marine environments of tropical Australia were sampled to collect data as shown in Figure 3.3. The underwater video recordings were obtained by deploying cameras affixed to metal frames, which were lowered over the side of a research vessel to capture footage of various fish species along the coast of Hinchinbrook, a region in North Eastern Australia (Saleh *et al.*, 2020).

3.2.2 Custom-Annotated Datasets

The following annotated datasets were collected and custom-annotated in collaboration with SAIAB for this research. The proposed datasets are Pondoland, Richards Bay,

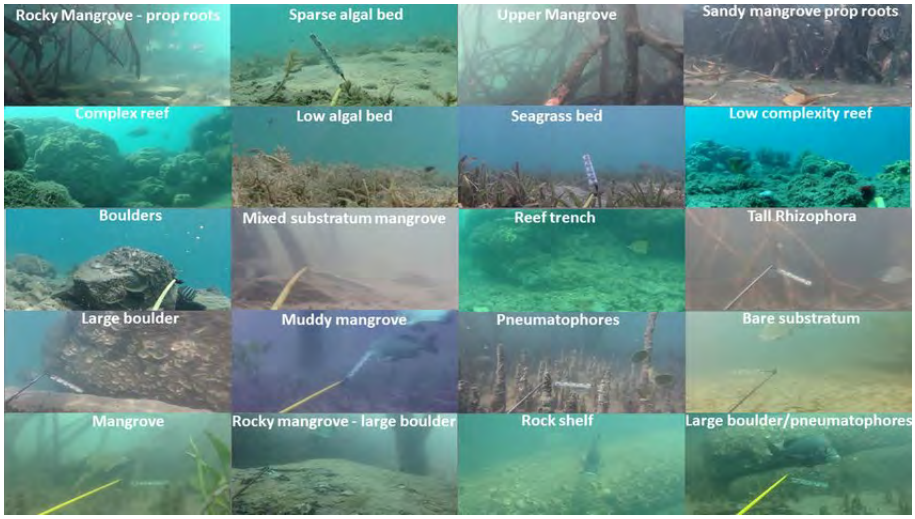


Figure 3.3: Image samples from DeepFish Dataset (Saleh *et al.*, 2020).

Tanganika, 2013 Aldabra, and 2020 Aldabra datasets containing diverse fish species and various underwater environments. The following subsections first describe the equipment used within the dataset, followed by a description of the datasets. The image and annotation counts breakdown for these datasets are detailed in Section 4.2. Note that it was not possible for marine biologists experts to name all fish classes due to the nature of the relatively unexplored data and regions where they were captured. In these cases, generic numbered class names were given.

3.2.2.1 Equipment

The equipment utilised to capture the fish species within the datasets consists of Baited Remote Underwater Video (BRUV) systems, which captured most of the fish datasets, accompanied by GoPro Hero 3 cameras. This combination of technology enables the comprehensive documentation of a wide range of fish species across various aquatic environments. Detailed descriptions of the specific ecological settings are provided in Section 3.2.1.

This choice is attributed to the effectiveness of BRUV in non-invasively capturing fish images in their natural habitats (Nalmpanti *et al.*, 2023). BRUV configurations typically involve placing cameras near baited stations, enabling researchers to record the presence

and behaviour of fish species attracted to the baited area.

3.2.2.2 Pondoland Dataset

The Pondoland dataset consists of diverse fish species from Pondoland, a natural region in Eastern Cape Province. This province is situated in South Africa, as shown in Figure 3.4. The dataset contains the following four classes: *Slinger. C. Puniceus*, *Diplodus Hottentotus*, *Pisces-1*, and *Pisces-2*.

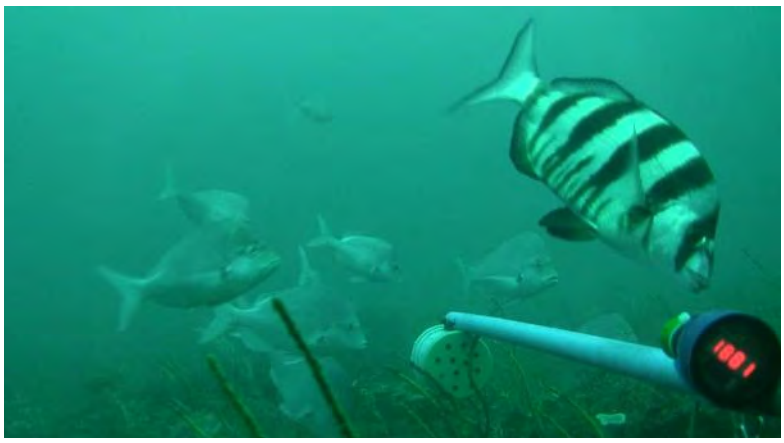


Figure 3.4: Image samples from Pondoland Dataset.

3.2.2.3 Richards Bay Dataset

The Richards Bay dataset contains a single fish species from Richards Bay, a city in KwaZulu Natal in South Africa, as shown in Figure 3.5. The dataset contains the following class: *Fish*.

3.2.2.4 Tanganyika Dataset

The Tanganyika dataset contains diverse fish species from Lake Tanganyika found in Zambia, as shown in Figure 3.6. The dataset contains three classes: *Fish-1*, *Fish-2*, and *Fish-3*. The breakdown of image and annotation counts for this dataset are detailed in Section 4.2.2.3.

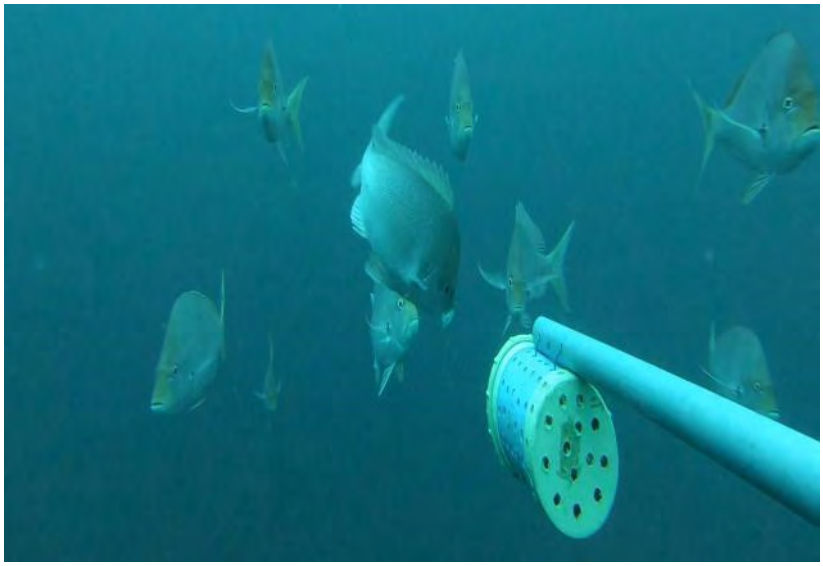


Figure 3.5: Image samples from Richards Bay Dataset.

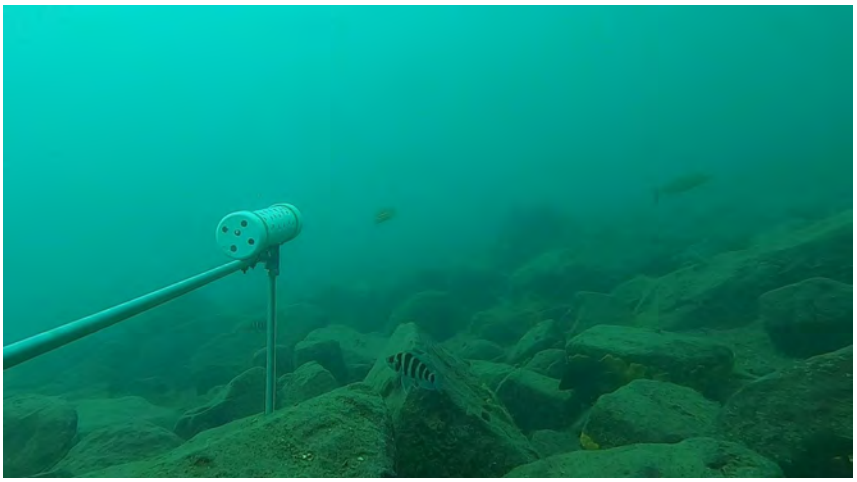


Figure 3.6: Image samples from Tanganyika Dataset.

3.2.2.5 2013 Aldabra Dataset

The 2013 Aldabra dataset contains diverse fish species from Aldabra Islands, a group of islands situated in the Indian Ocean, as shown in Figure 3.7. The dataset contains the following class: *Fish*.

Figure 3.7: Image samples from 2013 Aldabra Dataset.

3.2.2.6 2020 Aldabra Dataset

The 2020 Aldabra dataset contains diverse fish species from Aldabra taken in 2020 in the same location as 3.2.2.5. The datasets contain the following classes: *Caranx Melampygus*, *Odonous Niger*, *Carcharhinus amblyrhynchos*, *Carangoides*, *Naso Brevirostris*, *Epinephelus Tukula*, and *Labroides dimidiatus*.

Figure 3.8: Image samples from 2020 Aldabra Dataset.

3.2.3 Data Annotation Process

The annotation of the process was conducted using an annotation software called Roboflow. Bounding boxes were drawn around each fish instance presented within the images. These bounding boxes act as ground-truth labels, providing precise information regarding the location and dimensions of each fish within the images. The annotation process is initiated with an unannotated image followed by a selection of the bounding box tool, symbolised by a rectangle icon within the tool's ribbon.

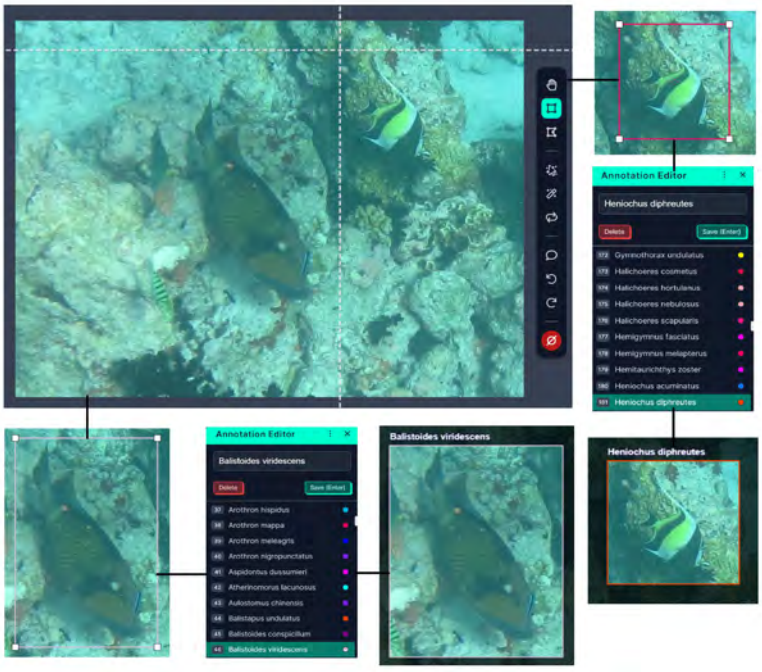


Figure 3.9: Visualisation of the annotation process: utilising bounding boxes with Roboflow.

3.2.4 Data Structure

Following the custom annotation process, the datasets were organised into a YOLO format data structure, adhering to a systematic structure. This structure's core is the `data.yaml` file, which contains the dataset configuration details. Accompanying these files are README documents offering comprehensive details on the dataset and Roboflow-related insights. Within the datasets, three main directories stand out: `test`, `train`,

and `valid`.

```
    test
        images
        labels
    train
        images
        labels
    valid
        images
        labels
data.yaml
README.dataset.txt
README.roboflow.txt

```

Listing 3.1: YOLO-formatted data structure.

The `test` directory holds image and label sub-directories for testing purposes, while `train`and `valid` replicate this organisation, serving as repositories for training and validation data, respectively. This structured dataset layout facilitates model training, testing, and validation activities.

3.3 YOLO Architecture Selection

Two popular YOLO architectures, YOLOv5 and YOLOv8 were considered for evaluation on underwater fish datasets for detection and classification. The two architectures for the evaluation were selected by investigating and comparing each architecture's discriminative ability to detect and classify fish in underwater environments.

3.3.1 YOLOv5 and YOLOv8 Selection

The selection process for fish detection and classification is motivated by two primary factors: the representation of YOLOv5 in fish detection-related publications. YOLOv5 is a prominently used YOLO model for fish detection and classification, with published

studies demonstrating its effectiveness in various underwater environments (Li *et al.*, 2023, Xing *et al.*, 2023, Wang and Xia, 2023). These research papers highlight YOLOv5's suitability for fish detection and classification tasks.

YOLOv8, the successor of YOLOv5, inherits the strengths of its predecessor while introducing novel advancements tailored for enhanced object detection performance (Wang *et al.*, 2023). Its anchor-free design, a key differentiator, eliminates the need for predefined anchor boxes, leading to a more precise object localisation, particularly beneficial for small objects such as smaller fish (Hindarto, 2023).

Despite their promising capabilities, both models have not been extensively evaluated for fish detection and classification tasks compared to other object detection models such as CNN Li *et al.* (2015). This gap in research presents an opportunity to systematically assess their performance on a diverse range of fish species and challenging underwater environments.

To address this gap, this research comprehensively evaluates both models on different datasets. In the study conducted by Gasparovic *et al.* (2023), the authors compared the performance of five different YOLO versions, namely YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8.

The findings in the study showed that YOLOv5 and YOLOv8 outperformed the other models by achieving competitive mAP scores of 0.96 and 0.97, respectively. The study also emphasised that despite newer versions such as YOLOv6, YOLOv7 and YOLOv8, the previous model, YOLOv5, exhibited better performance detecting fish within the utilised dataset. The datasets employed in the study were acquired through a remotely operated vehicle (ROV) comprising 2415 images.

However, a limitation of the research lies in its reliance on a single dataset, which did not offer a conclusive assessment of the various detection and precision scores associated with each model in underwater environments.

3.3.2 YOLOv5 Versions

The distinctive choice of YOLOv5s is evident in its prominent position within the provided in Table 3.1. Notably, despite its comparatively smaller model size, YOLOv5s had high metrics performance, with an Average Precision (AP) of 36.8 for both the validation and test datasets, complemented by a competitive AP at 50%. This selection is highlighted by the model's remarkable processing efficiency, executing computations at an impressive 2.2ms and achieving a Frames Per Second (FPS) of 455 on the v100 platform.

Table 3.1: YOLOv5 pre-trained model checkpoints on the different YOLOv5 versions.

Model	**size**	**AP**$^{\text{Val}}$	**AP**$^{\text{test}}$	**AP**^{50}	**Speed**$^{\text{v100}}$	**FPS**$^{\text{v100}}$	**params**
YOLOv5s	640	36.8	36.8	55.6	2.2ms	455	7.3M
YOLOv5m	640	44.5	44.5	63.1	2.9ms	345	21.4M
YOLOv5l	640	48.1	48.1	66.4	3.8ms	264	47.0M
YOLOv5x	640	50.1	50.1	68.7	6.0ms	167	87.7M
YOLOv5x + TAA	832	51.9	51.9	69.6	24.9ms	40	87.7M

YOLOv5s aligns with the research objectives, emphasising the consideration of computational constraints. The application of parameters, maintaining a modest count of 7.3M, highlights YOLOv5's capabilities within the research context.

3.3.3 YOLOv8 Versions

The research emphasises practical considerations and efficiency in the selection of YOLOv8n outlined in Table 3.2. YOLOv8n is chosen for its compact model size and reduced computational demands compared to alternative models. It features 640×640 pixels, an mAP of 37.3 on the validation set, and a short CPU processing speed of 80.4ms.

Table 3.2: YOLOv8 pre-trained model checkpoints on the different YOLOv5 versions.

Model	**Size**$^{\text{pixels}}$	**mAP**$^{\text{Val}}$	**Speed CPU**$^{\text{ms}}$	**Speed**$^{\text{A100}}$	params
YOLOv8n	640	37.3	80.4	0.99	3.2
YOLOv8s	640	44.9	128.4	1.20	11.2
YOLOv8m	640	50.2	234.7	1.83	25.9
YOLOv8l	640	52.9	375.2	2.39	43.7
YOLOv8x	640	53.9	479.1	3.53	68.2

YOLOv5s aligns with the research objectives, emphasising the consideration of computational constraints. The application of parameters, maintaining a modest count of 7.3M, highlights YOLOv5's capabilities within the research context.

The hardware and software infrastructure supporting these computations is outlined in Section 3.4 of the research, providing a comprehensive understanding of the computational environment for this research. The selected architectures for YOLOv5 and YOLOv8 are discussed in detail in Section 3.3 and provide a thorough understanding of their underlying design principles and modifications.

3.3.4 YOLOv5s

This section outlines the implementation of YOLOv5s for fish detection and classification, specifically highlighting the architecture process from input to output. The input to the YOLOv5s architecture is a single underwater image of fish. The image is resized to a dimension of 640 × 640 pixels to ensure compatibility with the model's internal processing as shown in Figure 3.10.

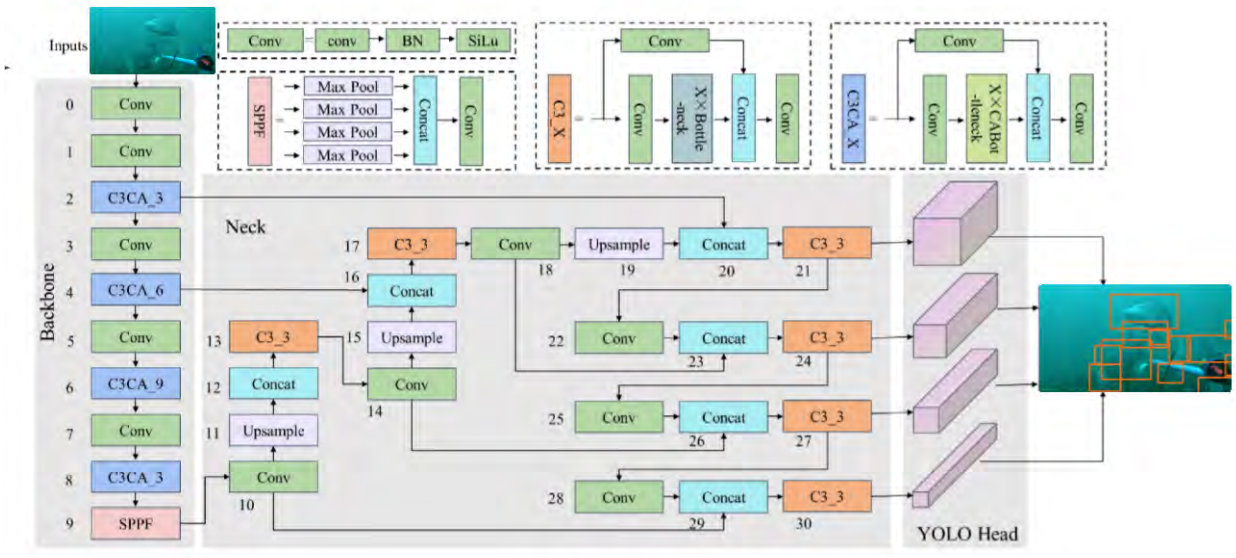


Figure 3.10: The network structure of YOLOv5s.

3.3.5 YOLOv8n

This section outlines the implementation of YOLOv8n for fish detection and classification, specifically highlighting the architecture process from input to output. Images are processed through the YOLOv8n network in a forward-pass manner. First, the image is normalised and resized to 640 × 640 pixels. The image is then passed through the backbone, which extracts features from the image. The features from the backbone are then passed through the neck, which fuses the feature maps and improves the flow of information as shown in Figure 3.11.

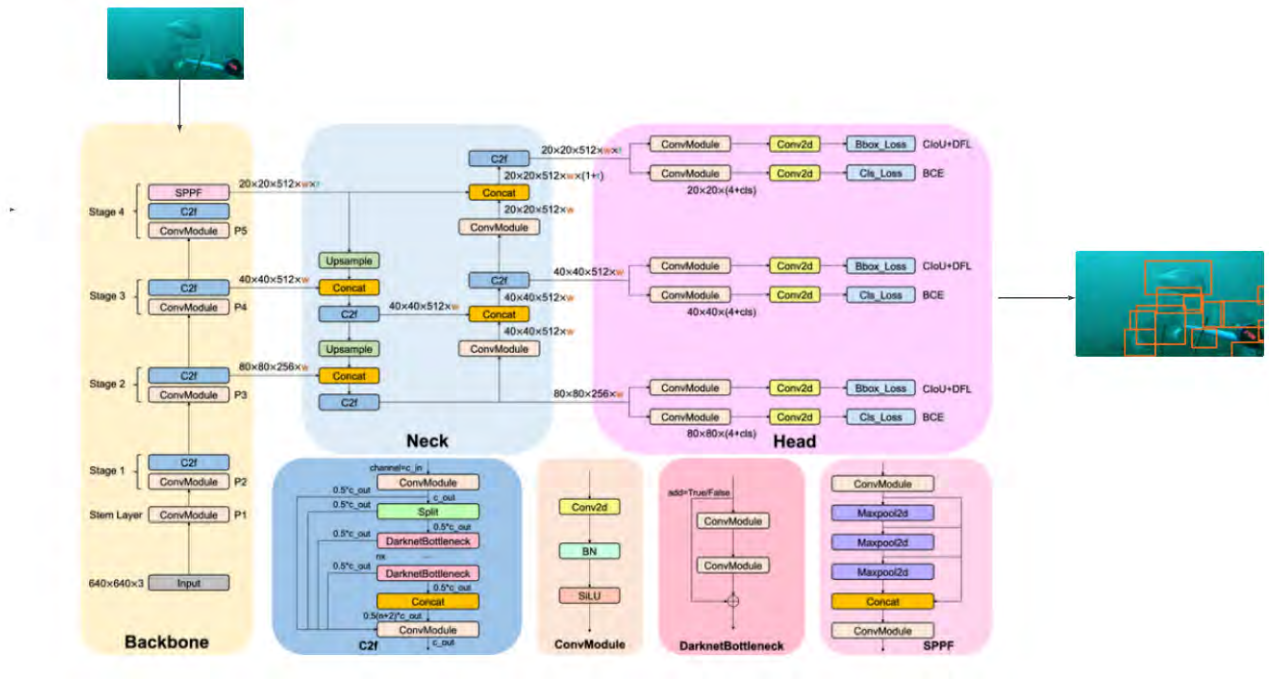


Figure 3.11: The network structure of YOLOv8n

The system infrastructure for this research consisted of hardware and software components necessary to train and evaluate the YOLOv8n architecture for fish detection and classification. In Section 3.4, the hardware and software components are outlined in detail.

3.4 System Infrastructure

The building and testing of the system's machine learning models required various hardware and software components as follows.

3.4.1 Hardware

A dedicated machine-learning server was created for use during this research. The machine learning server components included:

- **CPU**: 12th Gen Intel Core i7-12700K
- **RAM**: 64GB
- **GPU**: NVidia GTX 1080Ti

3.4.2 Software

The software employed in constructing the system was selected based on its provision by the South African Institute of Aquatic Biodiversity (SAIAB) for research purposes. The following software was used to build, train and test the system:

- Ubuntu 22.04.2 LTS
- Python 3.9
- TensorFlow 2.13.0-rc2
- Keras-Tuner 1.0.4

3.5 Model Improvements

This section is dedicated to refining the selected YOLOv5 and YOLOv8 models, primarily focusing on enhancing their performance in object detection in underwater environments. It encompasses several crucial aspects of model improvement, including data augmentation techniques to augment the training dataset and hyperparameter tuning to optimise model settings.

3.5.1 Data Augmentation

In this data augmentation example in Listing 3.2, a combination of techniques is applied to augment the original images for YOLOv5 and YOLOv8 models used in object detection tasks. The process begins by highlighting the bounding box areas in the original images, represented by specific colours. A transformation is introduced to rotate the images by 45°, simulating variations in fish orientations and underwater conditions. This augmented dataset, characterised by rotated images with highlighted bounding boxes, enhances the models' ability to generalise and detect fish species from diverse angles and under different environmental circumstances.

```
1
2  image_points = np.copy(image)
3  colors = [(0, 255, 0), (128, 128, 255)]
4  for bb, color in zip(bbs.bounding_boxes, colors):
5      image_points[bb.y1_int:bb.y2_int:4, bb.x1_int:bb.x2_int:4] = color
6
7  # rotate the image with the highlighted bounding box areas
8  rot = iaa.Affine(rotate=45)
9  image_points_aug, bbs_aug = rot(image=image_points, bounding_boxes=bbs)
10
11 # visualize
12 side_by_side = np.hstack([
13     bbs.draw_on_image(image_points, size=2),
14     bbs_aug.draw_on_image(image_points_aug, size=2)
15 ])
```

Listing 3.2: Data augmentation layers.

The visualisation showcases both the original and augmented bounding box representations, aiding in understanding the impact of data augmentation on model performance. This approach is valuable for improving the models' accuracy and adaptability when detecting fish in underwater environments, ultimately contributing to the research's overarching objectives.

3.5.2 Hyperparameter Tuning

The selection of hyperparameters for evaluating YOLOv5 and YOLOv8 on fish detection and classification is not only guided by domain-specific considerations but also constrained by software limitations and compatibility. These constraints encompass factors such as the capabilities of the chosen optimisation algorithms, the support for specific hyperparameter settings within the training framework, and the computational resources available for experimentation.

By navigating these software restraints, the chosen hyperparameters strike a balance between optimization effectiveness and practical feasibility, ensuring that the models can be trained efficiently within the constraints of the available software infrastructure. This pragmatic approach underscores the importance of aligning hyperparameter choices with both domain requirements and software limitations to achieve optimal performance in real-world applications of object detection and classification tasks, such as fish detection.

In underwater fish detection using YOLOv5 and YOLOv8 models, optimising hyperparameters is important in achieving accurate and robust object detection results. The following hyperparameters were selected using Random Search to enhance the models' capabilities in detecting and classifying fish species under challenging underwater conditions. These parameters serve as the baseline for subsequent experiments in Section 5.1.1, guiding the overall configuration of both models hyperparameters for training.

```
1         lr0: 0.0001  # initial learning rate (SGD=1E-2, Adam=1E-3)
2         lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
3         momentum: 0.937  # SGD momentum
4         weight_decay: 0.0003  # optimiser weight decay 3e-4
5         warmup_epochs: 3.0  # warmup epochs (fractions ok)
```

```
6         warmup_momentum: 0.8  # warmup initial momentum
7         warmup_bias_lr: 0.1  # warmup initial bias lr
8         optimiser: SGD # optimser
9         box: 0.05  # box loss gain
10        cls: 0.5  # cls loss gain
11        cls_pw: 1.0  # cls BCELoss positive_weight
12        obj: 1.0  # obj loss gain (scale with pixels)
13        obj_pw: 1.0  # obj BCELoss positive_weight
14        iou_t: 0.20  # IoU training threshold
15        anchor_t: 4.0  # anchor-multiple threshold
16        anchors: 3  # anchors per output layer (0 to ignore)
17        fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
18        hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
19        hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
20        hsv_v: 0.4  # image HSV-Value augmentation (fraction)
21        degrees: 0.0  # image rotation (+/- deg)
22        translate: 0.1  # image translation (+/- fraction)
23        scale: 0.5  # image scale (+/- gain)
24
```

Listing 3.3: Hyperparameter tuning with random research.

1. **`lr0` (Initial Learning Rate):** This parameter sets the initial learning rate for the optimiser. In this case, it is configured to 0.0001, and the comment notes that for stochastic gradient descent (SGD), the typical value is 1E-2, while for Adam, it is 1E-3.

2. **`lrf` (Final OneCycleLR Learning Rate):** It defines the final learning rate for the OneCycleLR learning rate schedule, calculated as the product of `lr0` and `lrf`. OneCycleLR is a technique that adjusts the learning rate during training to enhance convergence and generalisation.

3. **`momentum`:** Specifies the momentum for the SGD optimiser. A value of 0.937 indicates a high momentum setting, which helps accelerate training and improves the convergence of the model.

4. **`weight_decay`:** This parameter controls the strength of weight decay regularisation in the optimiser. A value of 0.0001 (1e-4) is set to prevent overfitting by penalising large weights.

5. **`optimiser:`** Specifies the optimiser used for training. In this case, Stochastic Gradient Descent (SGD) is employed.

The choice of Stochastic Gradient Descent (SGD) as the optimiser for fish detection and classification in underwater environments is driven by its robustness to noisy data and the ability to fine-tune hyperparameters manually. Koushik and Hayashi (2016), the authors demonstrate that SGD does outperform AdamW on small datasets. They attribute this to SGD's superior generalisation performance and ability to avoid getting stuck in local minima. In underwater conditions, where images can be affected by factors like water turbidity and varying lighting, SGD's stochastic nature enables the model to handle diverse data patterns.

Manual control over the learning rate allows for adaptability to the dynamic underwater environment, and the momentum term enhances stability during training, which is crucial for navigating complex optimisation landscapes. Furthermore, the controlled weight decay in SGD prevents overfitting, a critical consideration in scenarios with limited labelled data. Overall, the historical success of SGD in deep learning tasks and its versatility make it a reasonable and effective choice for fish detection in challenging underwater conditions.

These baseline hyperparameters are carefully chosen to provide a starting point for experimentation. The values are not defaults but are selected based on domain knowledge and empirical observations to suit the specific challenges of underwater fish detection. They are used consistently across experiments in Chapter 5 to ensure a fair and meaningful comparison of different model configurations and strategies.

3.6 Summary

The chapter details dataset collection and processing procedures, including data sources, equipment, dataset description, system infrastructure and YOLO architecture selection. Furthermore, detailing techniques such as data augmentation and hyperparameter tuning

were explored. These YOLO architectural refinements aim to enhance generalisation, with data augmentation applied both within the network and directly to the data.

4

Experimental Setup

This chapter details the experimental setup used in this research. It begins with the appropriate metric selection, followed by the design of all experiments conducted, and finally, the breakdown of all the image datasets utilised in the experiments.

4.1 Measuring Model Performance

Multiple metrics are available for evaluating the performance of a given machine-learning model. The metrics used to assess the performance of YOLOv5s and YOLOv8n are accuracy, precision, recall, F1-score, Mean Average Precision (mAP), and Intersection Over Union (IoU).

4.1.1 Accuracy

Accuracy is a fundamental metric in machine learning and is the favoured metric for comparing the performance of machine learning models in the majority of fish-related research papers available (Boom *et al.*, 2012, Zhao *et al.*, 2021, Albawi *et al.*, 2017, He *et al.*, 2016a, Lee and Youngseop, 2020, Bochkovskiy *et al.*, 2020, Singh and Mukhopadhyay, 2017, Huang *et al.*, 2018).

$$\text{Accuracy} = \frac{\text{number of correct classifications}}{\text{total number of classifications attempted}}$$

Accuracy gives a good reflection of the performance of a given model if the test data used is balanced. This means it consists of an equal number of samples belonging to each target class. However, since each test data entry is randomly chosen, this is often not guaranteed. The F1-score will thus also be reported to cater to random test data.

4.1.2 F1-score

The F1-score aims to provide an accurate reflection of the performance of a given model. It is calculated as the harmonic mean of the model's precision and recall. Precision is the measure of the correctness of all positive classifications per class:

$$\text{Precision } = \frac{\text{number of correct classifications of a given class}}{\text{total number of classifications into that class}}$$

or

$$\text{Precision } = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Contrary to precision, the recall metric counts how many classifications of a particular class were found out of all the samples for that class:

$$\text{Recall } = \frac{\text{number of correct classifications of a given class}}{\text{total number of samples for that class}}$$

or

$$\text{Recall } = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Finally, the F1-score combines the precision and recall of the model by using the following calculations:

$$F - \text{ score } = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}.$$

Precision, recall, and consequently, the F1-score are computed for each class individually. These class-specific scores are amalgamated using macro averaging. In this approach, the

distinct F1-scores obtained for each class are aggregated without prior weighting. This method ensures an overall F1-score that equally considers all classes, regardless of their frequencies or sizes, better capturing the natural setting of underwater life.

4.1.3 Mean Average Precision – mAP

Mean Average Precision (mAP) is an important metric in object detection, serving as a comprehensive measure of a model's performance across various object classes. Based on precision and recall analyses, mAP consolidates metrics to assess an algorithm's accuracy in classifying and categorising objects within images (Haruna *et al.*, 2017). While precision provides valuable insights, it may not capture the complete performance picture, especially when dealing with predictions of varying confidence levels.

Addressing this limitation, Average Precision (AP) considers the precision-recall trade-off at different confidence thresholds. The AP is derived from the area under the precision-recall curve, presenting a nuanced evaluation. Elevated AP values signify superior performance, indicating a model's adeptness at maintaining a delicate balance between precision and recall. In scenarios involving multiple object classes, as is common in object detection tasks, the role of mAP becomes paramount. Calculating the average AP across all classes, mAP offers a consolidated measure that encapsulates a model's overall effectiveness. This proves particularly critical in real-world applications where accurate detection across diverse categories is imperative.

The significance of mAP is underscored by its establishment as a standard benchmark, notably in datasets such as COCO (Common Objects in Context). Models achieving notable mAP scores on these datasets showcase their robustness and reliability in navigating complex scenarios, solidifying mAP's standing as a fundamental metric in advancing the field of computer vision and object detection.

4.1.4 Precision-Recall Curve

The precision-recall curve is a graphical representation that illustrates the trade-off between precision and recall for different thresholds in binary classification or information retrieval tasks. It is a valuable tool for evaluating and comparing the performance of machine learning models, particularly in scenarios where class imbalances exist or where different levels of confidence are required.

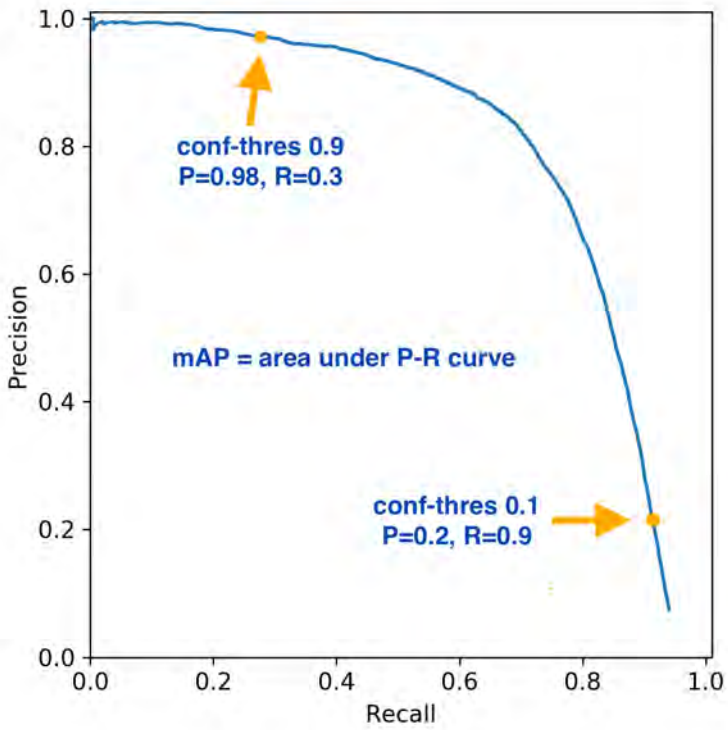


Figure 4.1: Representation of the Precision-Recall Curve

The Precision-Recall Curve is a vital tool for assessing the performance of binary classification models, especially in situations with class imbalances. It offers insights into how the model's precision and recall change as the classification threshold is adjusted, helping to make informed decisions about model selection and threshold tuning.

4.1.5 Intersection over Union (IoU)

IoU is an important concept in computer vision and object detection. It measures the degree of overlap between two bounding boxes, typically used to represent the location and extent of objects in an image. IoU is particularly useful for evaluating the accuracy of object detection algorithms by quantifying the similarity between the predicted bounding box and the ground trust bounding box.

IoU is calculated as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

The Area of Overlap is the region where the predicted bounding box and the ground truth bounding box intersect. The Area of Union is the total surface cover d by the predicted and ground truth bounding boxes.

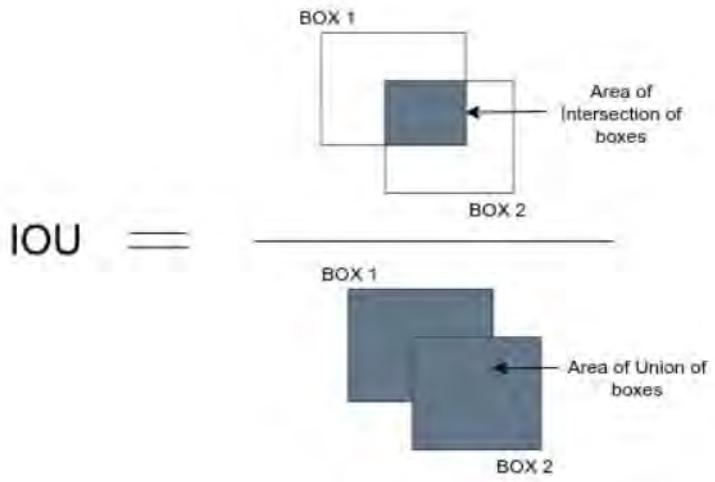


Figure 4.2: Diagrammatic representation of the formula to calculate IOU

IoU values range from 0 to 1, where a higher value indicates a greater overlap and a value of 0 indicates a lesser overlap. Important in this research, the default IoU threshold for calculating the mAP will be set to 0.5, commonly denoted as mAP_0.5.

4.2 Description of Datasets

The datasets acquired for this research are detailed in Section 4.2.1 to 4.2.2. All images are converted to the .png format and resized to 640×640, and a 70:10:20 split on the data was used for training, validation, and testing for all experiments used in this research.

In the following Table 4.1 contains the sources for all datasets utilised in this research.

Table 4.1: Sources for all datasets utilised in this research.

Dataset Sources
Brackish Dataset
DeepFish Dataset
Pondoland Dataset
Richards Bay Dataset
Tanganyika Dataset
2013 Aldabra Dataset
2020 Aldabra Dataset

4.2.1 Public Datasets

Two publicly available fish image datasets were acquired for Experiments 1.1 to 1.2.

4.2.1.1 Brackish Dataset

The annotations include six classes: *crab*, *fish*, *jellyfish*, *shrimp*, *small_fish*, and *starfish*. The dataset in Table 4.2 contains image and annotation counts for this dataset.

Table 4.2: Classwise image and annotation counts of the Brackish dataset.

Class	Images	Annotations
crab	1467	6538
fish	1467	3241
jellyfish	1467	637
shrimp	1467	548
small_fish	1467	9556
starfish	1467	5093

4.2.1.2 DeepFish Dataset

The annotations include one generic class: *Fish*, and contain 4504 images and 15463 annotation counts. Table 4.3 contains image and annotation counts for this dataset.

Table 4.3: Classwise image and annotation counts of the Brackish dataset.

Class	Images	Annotations
Fish	4505	15463

4.2.2 Collected Datasets

Five-collected fish image datasets were acquired for Experiments 2.1 to 2.5. The datasets were collected and custom-annotated in collaboration with SAIAB for this research.

4.2.2.1 Pondoland Dataset

The annotations include four classes: *Slinger. C. Puniceus*, *Diplodus Hottentotus*, *Pisces-1*, and *Pisces-2*. Table 4.4 contains image and annotation counts for this dataset.

Table 4.4: Classwise image and annotation counts of the Pondoland dataset.

Class	Images	Annotations
Slinger. C. Puniceus	650	11420
Diplodus Hottentotus	650	475
Pisces-1	650	468
Pisces-2	650	353

4.2.2.2 Richards Bay Dataset

The annotations include one generic class: *Fish*. The dataset below in Table 4.5 contains image and annotation counts.

Table 4.5: Classwise image and annotation counts of the Richards Bay dataset.

Class	Images	Annotations
Fish	4580	20472

4.2.2.3 Tanganyika Dataset

The annotations include three classes: *Fish-1*, *Fish-2*, and *Fish-3*. The dataset is listed below in Table 4.6 containing image and annotation counts.

Table 4.6: Classwise image and annotation counts of the Tanganyika dataset.

Class	Images	Annotations
Fish-1	3184	9734
Fish-2	1215	4569
Fish-3	1045	4486

4.2.2.4 2013 Aldabra Dataset

The annotations include one generic class: *Fish*. The dataset is listed below in Table 4.7 containing image and annotation counts.

Table 4.7: Classwise image and annotation counts of the 2013 Aldabra dataset.

Class	Images	Annotations
Fish	1786	3619

4.2.2.5 2020 Aldabra Dataset

The annotations include seven classes: *Labroides dimidiatus*, *Carangoides*, *Caranx melampygus*, *Epinephelus Tukula*, *Carcharhinus amblyrhynchos*, *Naso Brevirostris* and *Odonous niger* as shown in Table 4.8.

4.3 Experiments – Public Datasets

The following experiments were created to satisfy the research objectives in Section 1.5.

Table 4.8: Classwise image and annotation counts of the 2020 Aldabra dataset.

Class	Images	Annotations
Labroides dimidiatus	517	654
Carangoides	1250	3572
Caranx melampygus	8761	87215
Epinephelus Tukula	967	1010
Carcharhinus amblyrhynchos	3531	4645
Naso Brevirostris	1185	1754
Odonous niger	7640	60510

4.3.1 Brackish Species Dataset Experiment

The two models were evaluated on the Brackish dataset to evaluate the effectiveness of their detection and classification of the different classes in the dataset provided.

- Experiment 1.1 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the Brackish dataset.

The results for the experiments above are shown in Section 5.2.1.

4.3.2 DeepFish Dataset Experimemnt

The two models were evaluated on the DeepFish dataset.

- Experiment 1.2 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the DeepFish dataset.

The results for the experiments above are shown in Section 5.2.2.

4.4 Experiments – Collected Datasets

The following experiments were created to satisfy the research objectives in Section 1.5.

4.4.1 Pondoland Dataset Experiment

The two models were evaluated on the Pondoland dataset.

- Experiment 2.1 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the Pondoland dataset.

The results for the experiments above are shown in Section 5.3.1.

4.4.2 Richards Bay Dataset Experiment

The two models were evaluated on the Richards Bay dataset.

- Experiment 2.2 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the Richards Bay dataset.

The results for the experiments above are shown in Section 5.3.2.

4.4.3 Tanganyika Dataset Experiment

The two models were evaluated on the Tanganyika dataset.

- Experiment 2.3 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the Tanganyika dataset.

The results for the experiments above are shown in Section 5.3.3.

4.4.4 2013 Aldabra Dataset Experiment

The two models were evaluated on the 2013 Aldabra dataset.

- Experiment 2.4 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the 2013 Aldabra dataset.

The results for the experiments above are shown in Section 5.3.4.

4.4.5 2020 Aldabra Dataset Experiment

The two models were evaluated on the 2020 Aldabra dataset.

- Experiment 2.5 – Compares the performance of YOLOv5s and YOLOv8n when trained and tuned on the 2020 Aldabra dataset.

The results for the experiments above are shown in Section 5.3.5.

4.5 Summary

This chapter detailed the metrics chosen for model performance measurement: accuracy, F1-score, precision, and recall. It listed the two YOLO model evaluation experiments. Furthermore, it provided the composition of the various selected datasets and their information about the class, image count, and annotation count. The next chapter uses this experimental setup to conduct experiments on YOLOv5 and YOLOv8.

5

Results and Discussion

This chapter presents the outcomes of a comparative analysis conducted between YOLOv5s and YOLOv8n in detecting and classifying fish across the different datasets. It evaluated the performance of both models, comparing their strengths, limitations, and overall efficiency in detecting and classifying fish.

5.1 Preliminary Results

The following section presents the outcomes of the hyperparameter tuning conducted in preparation for experiments on YOLOv5s and YOLOv8n, along with experimental results and conclusions drawn from the analysis.

5.1.1 Hyperparameter Tuning

The selected hyperparameters for YOLOv5s and YOLOv8n architectures are summarised in Table 5.1. The hyperparameters were tuned using Random Search; this extensive training process was conducted on the Pondoland dataset. The dataset was selected as the base dataset because it contains the most complex backgrounds compared to other datasets.

The selected hyperparameters differ from the default settings and were selected for fish detection and classification training. The chosen hyperparameters for the training configuration include a batch size of 16, which was capped at 16 due to the availability of VRAM for this research, with early stopping implemented after 30 and 20 Epochs for both models, respectively. The training process is set to run for a total of 150 epochs for both models, utilising pre-trained weights from YOLOv5s.pt and YOLOv8n.pt models.

The initial learning rate (lr0) is established at 0.05 and 0.01 for both models, gradually decreasing to a final learning rate (lrf) of 0.1 and 0.01 through a cosine annealing and step decay learning rate schedule. During hyperparameter tuning, it was observed that YOLOv8 learned the data quicker, so the learning rate was reduced and was lower than YOLOv5. The chosen learning rate for YOLOv5s was the best learning compared to other learning rates, which caused a lot of fluctuation during the initial start of training.

A momentum of 0.942 and 0.951 is employed for optimisation using the SGD optimiser. Both models train with a warm-up period of 3 and 2 epochs, and a weight decay of 0.0003 and 0.0001 is applied. The carefully chosen hyperparameters aim to facilitate effective training and optimal performance for object detection tasks using YOLOv5s and YOLOv8n architectures. The accuracy-epoch training curve in Figure 5.1 is included to show that using the optimised hyperparameters yields similar training results but with noticeable differences during the earlier epochs.

Table 5.1: Best hyperparameters determined with Ray Tune to evaluate YOLOv5s and YOLOv8 architectures.

Hyperparameter	**YOLOv5s**	**YOLOv8n**
Batch Size	16	16
Early Stopping	30	20
Epochs	150	150
Model	YOLOv5s.pt	YOLOv8n.pt
Learning Rate (lr0)	0.05	0.01
Learning Rate (lrf)	0.1	0.01
Learning Rate Schedule	Cosine Annealing	Step Decay
Momentum	0.9	0.9
Optimiser	SGD	SGD
Pre-Trained	True	True
Warm-up Epochs	3	2
Weight Decay	0.0003	0.0001

The models were trained from scratch to determine the best epochs, as shown in Figure 5.1. The models did not improve in accuracy after 100 epochs. Furthermore, graphs show that YOLOv8n yielded a more erratic learning curve than YOLOv5s. This is attributed to the learning rate selected for YOLOv8 and how it learnt the data quicker than YOLOv5.

However, YOLOv8 achieved higher accuracies at a far earlier epoch than YOLOv5s and

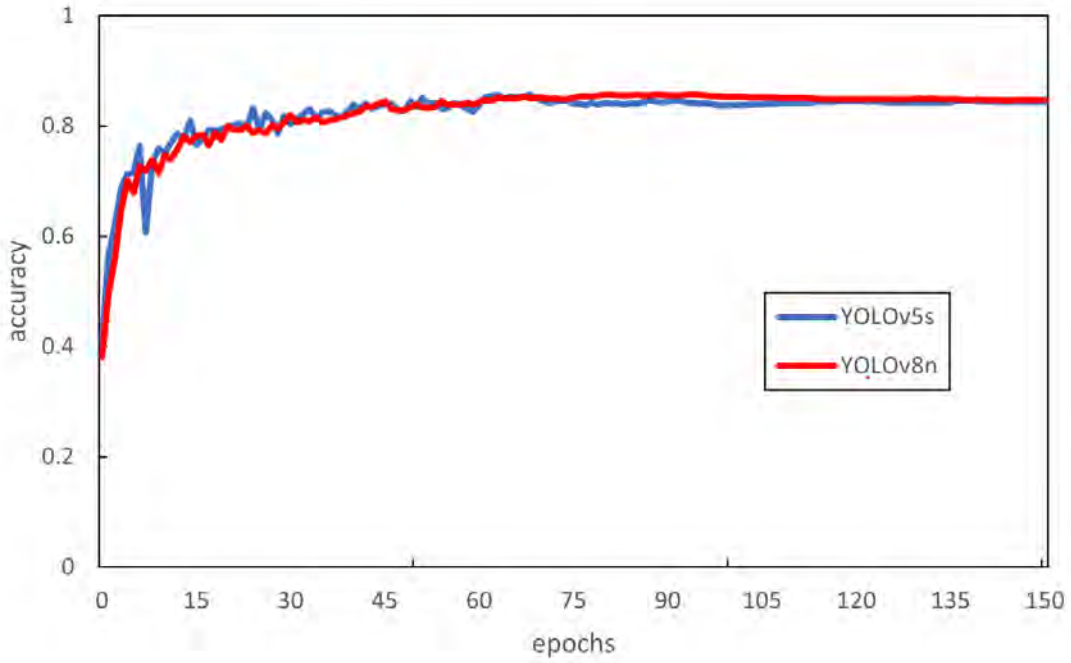


Figure 5.1: Validation accuracy of YOLOv5s and YOLOv8 models when trained on the Pondoland dataset.

obtained a higher maximum validation accuracy overall. The hyperparameter tuning was performed to produce an optimised model for the relevant data it was trained on, depending on the experiment. The resulting models were not fine-tuned but constructed from scratch. Due to time constraints, the optimal hyperparameters determined during preliminary experimentation were applied to all subsequent experiments in this research.

5.2 Experiment 1 – Public Datasets

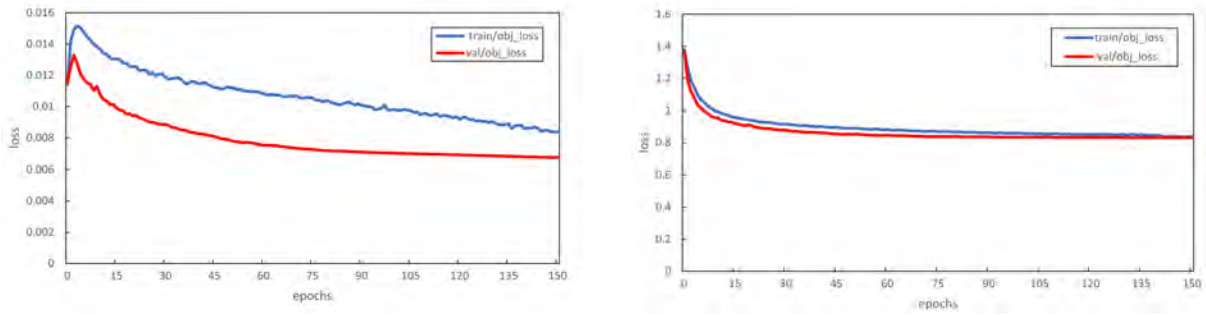
This section evaluates YOLOv5s and YOLOv8n on publicly available datasets to validate their effectiveness in fish detection and classification tasks. The subsections of this experiment present the results obtained during training and inference to analyse model fitting and offer a comparative analysis with findings from other research on the same datasets.

5.2.1 Experiment 1.1 – Brackish – Static Background

Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.2.1.1 Loss Curves

Figure 5.2a and 5.2b depict loss curves for YOLOv5s and YOLOv8n models, respectively. YOLOv8n exhibits early overfitting, where the validation set stops improving around Epoch 140. This means that in the initial epochs, the models quickly learn to perform well on the training data but struggle to predict bounding boxes on validation data. As a result, a noticeable performance gap exists between the training and validation set early in training.



(a) YOLOv5s **(b)** YOLOv8

Figure 5.2: Training and Validation Object Loss Curves for YOLOv5s and YOLOv8 on the Brackish dataset.

In the subsequent phase of Experiment 1.1, a comparative evaluation of mAP curves is conducted.

5.2.1.2 Detection mAP Curve

Figure 5.3 compares the mAP scores on the validation set over the same 150 Epochs during training. YOLOv8n exhibited strong initial performance due to quicker training convergence. It reached its highest mAP_0.5 of 0.986 at an earlier epoch than YOLOv5s. In

contrast, YOLOv5s achieved its best mAP_0.5 of 0.9648 at a later epoch than YOLOv8n. Notably, both models demonstrated consistent training performance, with only minor fluctuations in mAP_0.5 during the early stages of training directly after the initial 10 epochs. This is attributed to the SGD optimiser being used in this research.

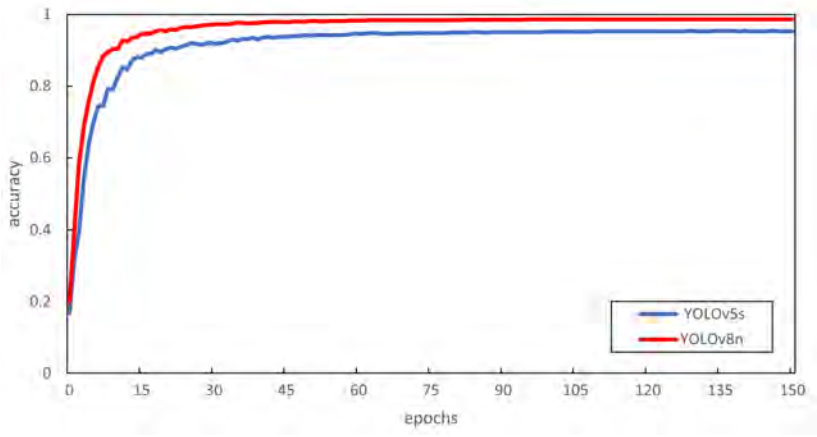
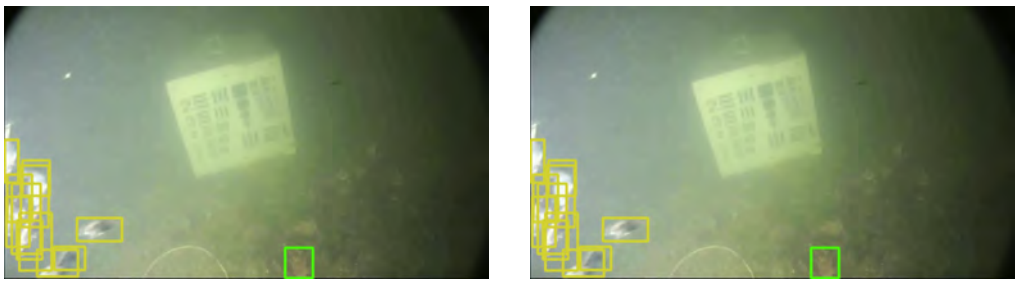


Figure 5.3: mAP_0.5 accuracy of YOLOv5s and YOLOv8n models on the Brackish dataset.

5.2.1.3 Detection Qualitative Analysis

The inference results of object detection on the unseen Brackish test dataset are presented visually in Figure 5.4a and 5.4b. The results presented in the figures exhibit accurate object detection for both YOLOv5s and YOLOv8n on the Brackish dataset at an IOU threshold of 0.5.



(a) YOLOv5s **(b)** YOLOv8

Figure 5.4: Detection results for YOLOv5s and YOLOv8 on the Brackish dataset.

As shown in the visual representations, the models had overlapping bounding boxes on the *small_fish* and *crab* classes depicted by the yellow and green bounding boxes, respectively.

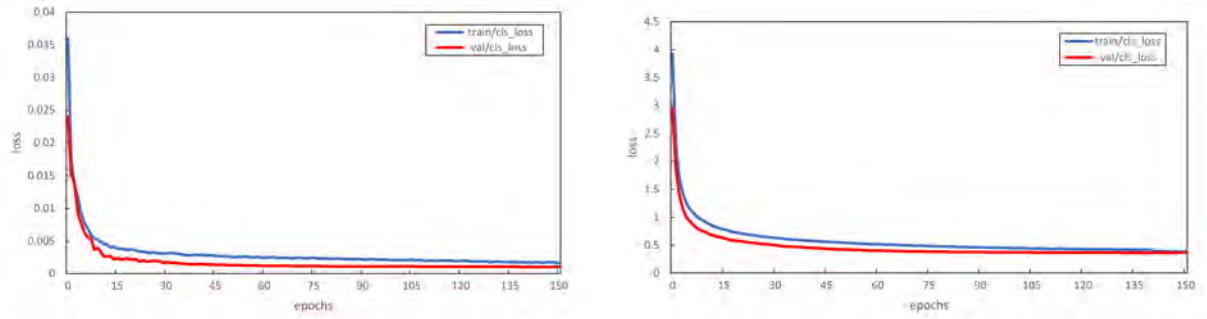
This could be attributed to many factors, including the image quality and the size of the fish within the image. Another factor could be the IoU threshold of 0.5, which was not high enough to suppress duplicate detections. Overall, the models performed well at detecting the various classes within the dataset, as shown in the next Section 5.2.1.4.

5.2.1.4 Class-Specific Classification

Table 5.2 presents the predicted multi-class accuracies for the same YOLOv5s and YOLOv8n models used during fish detection experiments and on the same Brackish dataset. Notably, both models achieve reasonably high accuracies across most classes, with YOLOv8 exhibiting slightly better results on average. However, YOLOv8n significantly outperforms YOLOv5s on smaller object classes such as *small_fish*.

Table 5.2: Predicted class F1-scores for YOLOv5s and YOLOv8n on the Brackish dataset.

Class	Images	Instances	YOLOv5s	YOLOv8n
crab	1467	1189	0.98	0.98
fish	1467	322	0.98	0.99
jellyfish	1467	55	0.98	0.99
shrimp	1467	76	0.99	0.98
small_fish	1467	1148	0.80	0.94
starfish	1467	791	0.96	0.99



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.5: Training and validation class loss curves for YOLOv5s and YOLOv8n on the Brackish dataset.

As shown in Figure 5.5a and 5.5b, YOLOv5s demonstrates a consistent decrease in class loss throughout the training in contrast to YOLOv8n exhibiting a consistent decrease

in the class loss until it reaches Epoch 140. Due to time and technical constraints, the training for both models does not provide a conclusive analysis of the class loss curve.

5.2.1.5 Classification Analysis

Table 5.3 presents the performance of evaluation metrics for YOLOv5s on the Brackish dataset. The results show better performance across various classes. With a precision score of 1.00, YOLOv5s demonstrates perfect accuracy in classifying successfully detected objects, extending across all classes, including *crab*, *fish*, *jellyfish*, *shrimp*, *_fish*, and *starfish*. Furthermore, the model exhibits a commendable recall, capturing between 80% to 99% of actual instances across different classes. The F1-Scores reflect the model's ability to maintain a strong overall performance, balancing precision and recall effectively.

Table 5.4 presents the evaluation metrics for YOLOv8n on the Brackish dataset. YOLOv8 also achieved a perfect precision score of 1.00. Notably, the model produces higher recall values than YOLOv5s, accurately capturing between 94% and 99% of instances across all classes. The F1-Scores summarise YOLOv8n's robust overall performance, specifically due to the improved recall and accuracy of the *shrimp* and *small_fish* classes.

Table 5.3: Performance of evaluation metrics for YOLOv5s on the Brackish dataset.

Class	Precision	Recall	F1-Score
crab	1.00	0.99	0.95
fish	1.00	0.99	0.96
jellyfish	1.00	0.98	0.88
shrimp	1.00	0.91	0.84
small_fish	1.00	80	0.86
starfish	1.00	0.96	0.96

In summary of the detection and classification results, both YOLOv5s and YOLOv8n exhibit exceptional performance on the Brackish dataset. After demonstrating excellent detection results, both models consistently achieve perfect precision and high recall scores across all six classes, showcasing their ability to identify fish as objects accurately. The F1-Scores further validate their robust overall performance. Finally, YOLOv8n performed

Table 5.4: Performance of evaluation metrics for YOLOv8n on the Brackish dataset.

Class	Precision	Recall	F1-Score
crab	1.00	0.99	0.84
fish	1.00	0.99	0.84
jellyfish	1.00	0.98	0.91
shrimp	1.00	0.94	0.98
small_fish	1.00	0.99	0.98
starfish	1.00	0.98	0.99

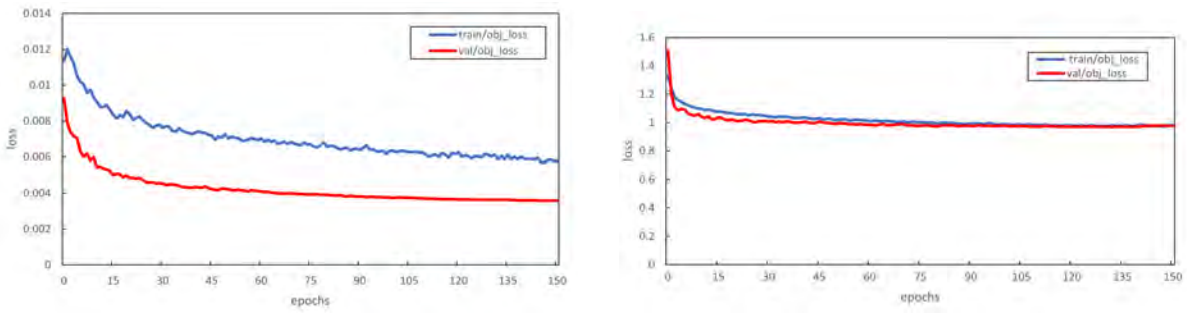
better than YOLOv5s at detecting *small_fish*, suggesting that YOLOv8n is better at both detecting and classifying smaller fish than YOLOv5s on this particular dataset.

5.2.2 Experiment 1.2 – DeepFish – Static Background

Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.2.2.1 Loss Curves

Figure 5.6a and Figure 5.6b represent the loss curves for YOLOv5s and YOLOv8. The models exhibit early overfitting, where the validation stopped improving around Epoch 35.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.6: Training and validation object loss curves for YOLOv5s and YOLOv8n on the DeepFish dataset.

In the subsequent phase of Experiment 1.2, a detailed examination of the mAP curves was conducted to establish a comprehensive comparison between the YOLOv5s and YOLOv8n models.

5.2.2.2 Detection mAP Curve

In the comparative analysis of the models trained over 150 epochs, both achieved competitive mAP scores. YOLOv8n exhibited strong initial performance, surpassing YOLOv5s, eventually reaching its highest mAP_0.5 score of 0.974 compared to YOLOv5s highest mAP_0.5 score of 0.945.

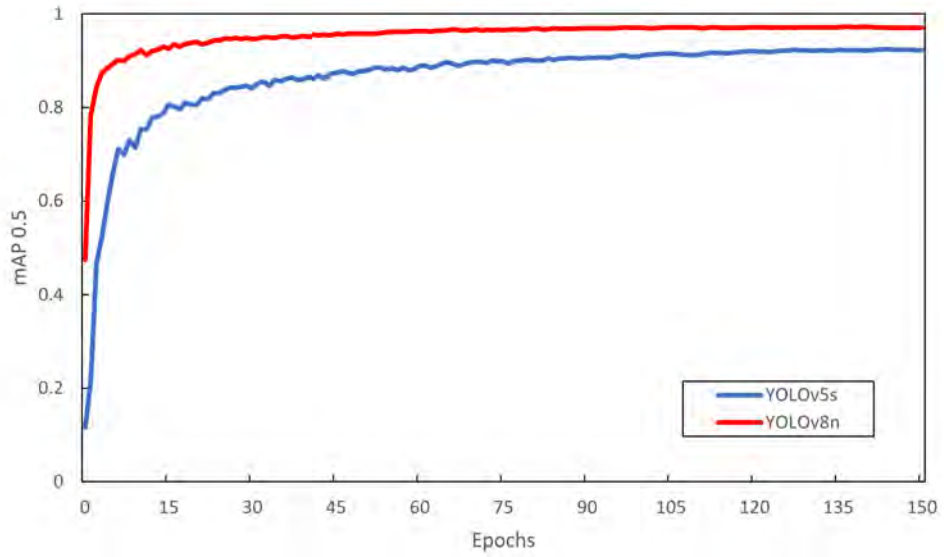


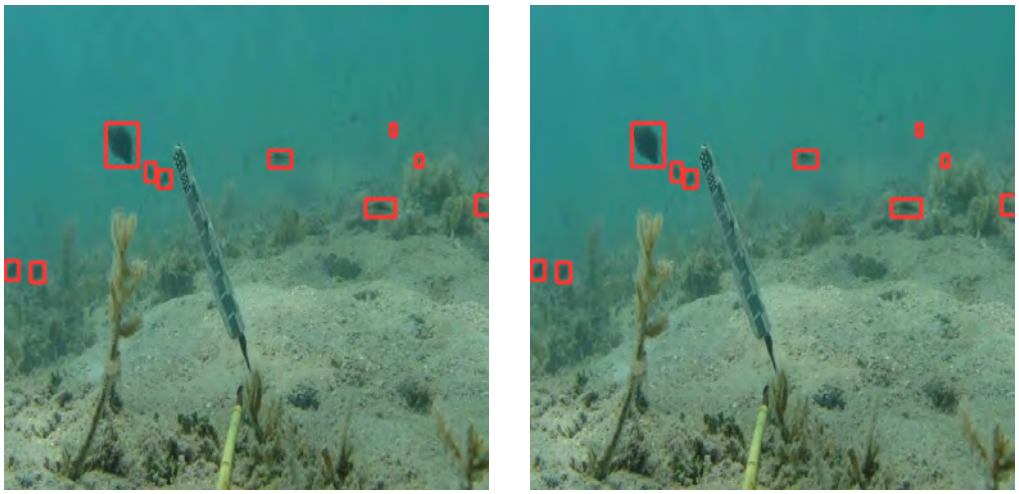
Figure 5.7: mAP_0.5 accuracy of YOLOv5s and YOLOv8nn models on the DeepFish dataset.

In the subsequent Section 5.2.2.4, the experiments evaluated the individual class accuracy across various classes in the DeepFish dataset.

5.2.2.3 Detection Qualitative Analysis

Figure 5.8a and 5.8b show that the models accurately detected the class within the dataset. As shown, both models could detect the very small fish in the background, which human

eyes could have missed. This demonstrates both models' ability to detect even very small fish which are difficult to detect.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.8: Detection results for YOLOv5s and YOLOv8n on the DeepFish dataset.

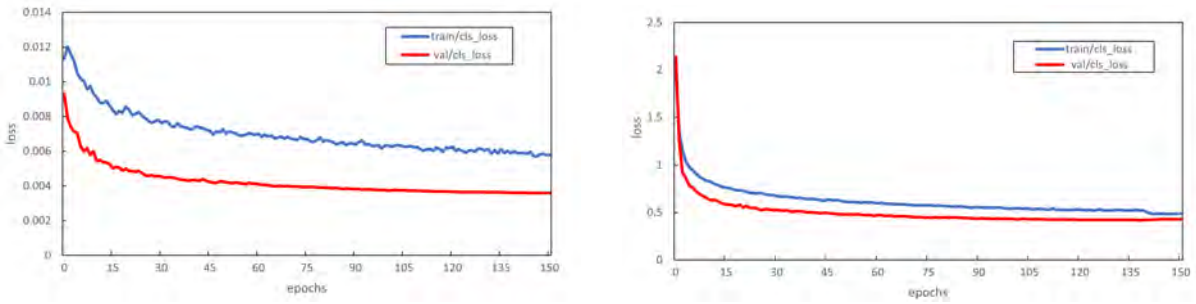
5.2.2.4 Class-Specific Classification

Table 5.5 presents the predicted class accuracies on various classes within the DeepFish dataset. Notably, both models achieve reasonably high accuracies on the class within the DeepFish dataset. YOLOv8n obtained a class accuracy of 0.97 compared to the class accuracy of 0.92 obtained by YOLOv5s.

Table 5.5: Predicted class for YOLOv5s and YOLOv8n on the DeepFish dataset.

Class	Images	Instances	YOLOv5s	YOLOv8nn
Fish	4505	31189	0.92	0.97

YOLOv8n consistently outperformed YOLOv5s across all images within the dataset, as indicated by the mAP curve in Section 5.2.2.2. Following the evaluation of individual class detection, the research assessed class loss during both training and validation for YOLOv5s and YOLOv8n.



(a) YOLOv5 **(b)** YOLOv8n

Figure 5.9: Training and validation class loss curves for YOLOv5s and YOLOv8n on the DeepFish dataset.

5.2.2.5 Classification Analysis

Table 5.6 and 5.7 present the evaluation metrics for YOLOv5s and YOLOv8n on the DeepFish dataset.

The evaluation metrics for YOLOv5s show a precision score of 1.00, indicating perfect accuracy in classifying fish instances. The recall value of 0.95 demonstrates that YOLOv5s successfully captures 95% of actual fish instances. The F1-Score of 0.84 reflects a solid overall performance, considering both precision and recall.

Table 5.6: Performance of evaluation metrics for YOLOv5s on the DeepFish dataset.

Class	Precision	Recall	F1-Score
Fish	1.00	0.95	0.84

Table 5.7: Performance of evaluation metrics for YOLOv8n on the DeepFish dataset.

Class	Precision	Recall	F1-Score
Fish	1.00	0.99	0.90

The evaluation metrics for YOLOv8n reveal a precision score of 1.00, denoting precise fish classifications. YOLOv8n excels in recall with a rate of 0.99, indicating its ability to identify 99% of fish instances correctly. The F1-Score of 0.90 reflects a strong overall performance. Both YOLOv5s and YOLOv8n exhibit remarkable performance on the DeepFish dataset, with YOLOv8n demonstrating a slight advantage in terms of recall and overall performance for the *Fish* class.

In the following Section 5.2.3, a comprehensive discussion of the presented results in the context of object detection on the Brackish and DeepFish datasets will be conducted, accompanied by a comparative analysis of findings reported in related research papers.

5.2.3 Experiment 1 Discussion

The results of Experiment 1.1 demonstrate the effectiveness of both YOLOv5s and YOLOv8n for object detection in underwater environments. Both models achieve a high mAP_0.5 score on the Brackish dataset, with YOLOv8n outperforming YOLOv5s on certain classes, such as shrimp and *small_fish*. The strong performance of both YOLOv5s and YOLOv8n on the Brackish dataset can be attributed to several factors.

First, these models are based on deep learning architectures trained on large datasets of natural images, enabling them to learn robust features for object detection (Selçuk and Serif, 2023). Second, both models employ efficient feature extraction and detection algorithms, which allows them to achieve real-time performance on resource-constrained devices (Zhang *et al.*, 2023).

The results of Experiment 1.1 and the comparison of YOLOv5s and YOLOv8n with the models used in the Brackish paper by (Zhang *et al.*, 2021) demonstrate that deep learning-based object detection models are a promising approach for underwater object detection. As shown in Table 5.8, the proposed YOLOv5s and YOLOv8n models outperform YOLOv4 and Tiny YOLOv4 on the Brackish dataset, both in terms of overall mAP and performance on individual classes.

YOLOv5s and YOLOv8n achieve mAPs of 0.96 and 0.98, respectively, compared to 0.93 and 0.80 for YOLOv4 and Tiny YOLOv4, respectively. Additionally, YOLOv5s and YOLOv8n outperform YOLOv4 and Tiny YOLOv4 on all individual classes, except *starfish*, where YOLOv4 and YOLOv8n achieve similar performance.

YOLOv5s and YOLOv8n incorporate several architectural improvements over YOLOv4, such as a more efficient feature extraction backbone and a new head network for object

Table 5.8: YOLOv5s and YOLOv8n models were utilised, while YOLOv4 and Tiny YOLOv4 were included for comparison against Zhang *et al.* (2021)'s research. The table presents the overall mAP scores and accuracy results for individual classes across these models.

Model	mAP	crab	fish	jellyfish	shrimp	small_fish	starfish
YOLOv4	0.93	0.91	0.98	0.96	0.94	0.83	0.95
Tiny YOLOv4	0.80	0.67	0.95	0.78	0.83	0.61	0.94
YOLOv5s	0.96	0.98	0.98	0.98	0.99	0.80	0.96
YOLOv8n	0.98	0.98	0.99	0.99	0.98	0.94	0.99

detection (Sharma *et al.*, 2023). These architectural improvements allow YOLOv5s and YOLOv8n to learn more complex and informative features from the input data, which leads to improved performance on object detection tasks. The results of Experiment 1.2 provide valuable insights into the performance of YOLOv5s and YOLOv8n models on the DeepFish dataset.

Similar to Experiment 1.1, both models exhibit early overfitting, which is often a challenge in object detection tasks with limited training data. However, YOLOv8n demonstrates a faster convergence and superior mAP_0.5 scores, indicating its potential to achieve accurate object detection earlier in the training process. The evaluation metrics also reinforce the models' accuracy, with both achieving high precision and recall, emphasising their robust overall performance. The visualised detection results affirm the effectiveness of both YOLOv5s and YOLOv8n in accurately detecting objects within the DeepFish dataset, validating their practical utility.

In Table 5.9, the research findings indicate that YOLOv3 and YOLOv4, as employed in (Muksit *et al.*, 2022), achieve a mAP score of 0.96 and 0.96, respectively. Comparing these results to the outcomes of this research, where YOLOv5s achieves an mAP of 0.94, and YOLOv8n reaches an mAP of 0.97, a discernible difference becomes apparent. These variations in mAP scores can be attributed to multiple factors and highlight the dynamic nature of object detection models. The evolution of YOLO architecture from YOLOv3 to YOLOv5s and YOLOv8n introduces substantial optimisations (Sharma *et al.*, 2023, Bochkovskiy *et al.*, 2020). YOLOv5s and YOLOv8n place a particular emphasis on optimising the trade-off between speed and accuracy (He *et al.*, 2023). These models strive to maintain real-time capabilities while still achieving competitive accuracies as

indicated in the results produced (Zhang *et al.*, 2023).

Table 5.9: Object detection accuracy on the DeepFish Dataset: A comparison of YOLOv5s, YOLOv8n, YOLOv4, and YOLOv3

Model	mAP	Precision	Recall	F1-Score
YOLOv3	0.96	0.94	0.94	0.94
YOLOv4	0.96	0.95	0.93	0.94
YOLOv5s	0.94	1.00	0.95	0.98
YOLOv8n	0.97	1.00	0.99	0.98

YOLOv5s and YOLOv8n are both object detection algorithms that are more accurate and efficient than their predecessors, YOLOv3 and YOLOv4 (Zhang *et al.*, 2023). This is partly due to their use of new backbone architectures, such as CSPDarknet53 and SPP, which can extract features from multiple scales and sizes of objects (Mahasin and Dewi, 2022). Moreover, YOLOv5s and YOLOv8n use data augmentation, multi-scale training, SimCLR, MixUp, and CutMix to improve their training performance, and NMS and PA to improve their inference performance. As a result, YOLOv5s and YOLOv8n are both promising candidates for real-time object detection applications.

One key factor contributing to these variations in mAP scores is the architectural evolution of YOLO models. Over time, YOLO architectures have undergone substantial optimisations, such as transitioning from YOLOv3 to YOLOv5s and YOLOv8n. These architectural enhancements include improvements in network depth, feature extraction, and object detection techniques (Bochkovskiy *et al.*, 2020, Magalhães and Peixoto, 2019, Atik *et al.*, 2022, Hussain, 2023). YOLOv5s, for instance, features a deeper and more complex model architecture that allows it to capture intricate details and patterns within underwater images more effectively. This results in YOLOv5s's exceptional precision, with a precision score of 1.00, and competitive recall, achieving 95% recall (Redmon and Farhadi, 2018).

The results indicate that the architectural advancements in YOLO models, coupled with dataset differences, have improved the performance of YOLOv5s and YOLOv8n in fish detection, achieving mAP scores of 0.94 and 0.97, respectively. These disparities emphasise the adaptability of YOLO models to specific use cases and underscore the influence of architectural enhancements in achieving higher detection accuracy. Section 5.3 analyses

experiments on various collected datasets using YOLOv5s and YOLOv8n.

5.3 Experiment 2 – Collected Datasets

This section evaluates the models on collected datasets to validate their effectiveness in fish detection and classification tasks. The experiments in this section present the results obtained during training and inference to analyse model fitting and offer a comparative analysis.

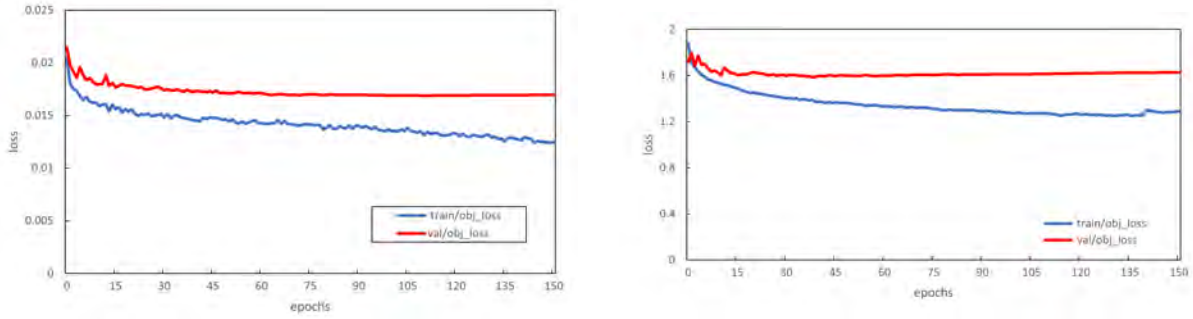
5.3.1 Experiment 2.1 – Pondoland – Static Background

Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.3.1.1 Loss Curves

Figure 5.10a and 5.10b, respectively. Both models exhibit early overfitting, where validation stops improving around Epoch 65. YOLOv5s validation stabilises and no longer improves after 65 epochs. YOLOv8n's validation stabilises at an earlier epoch before 65 epochs and stops improving. This means that in the initial epochs, the models quickly learn to perform on training data but struggle to generalise to unseen data due to the limited training data.

In the subsequent phase of Experiment 2.1, a formal assessment of the mAP curve performance is conducted to compare the performance of YOLOv5s and YOLOv8n models. This analysis aims to objectively evaluate and contrast the accuracy and precision of both models in object tasks. An assessment of the mAP curve performed is conducted in Section 5.3.1.2 for this experiment.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.10: Training and validation object loss curve for YOLOv5s and YOLOv8n on the Pondoland dataset.

5.3.1.2 Detection mAP Curve

YOLOv5s exhibits a strong initial start, reaching its highest mAP_0.5 of 0.812 at an earlier epoch. YOLOv8n reaches its highest mAP_0.5 of 0.802 at a later epoch than YOLOv5s. Both models exhibit minor fluctuations throughout the earlier stages of training and later remain consistent and closer to each other, with YOLOv5s being above YOLOv8n in terms of accuracy scores throughout the training process.

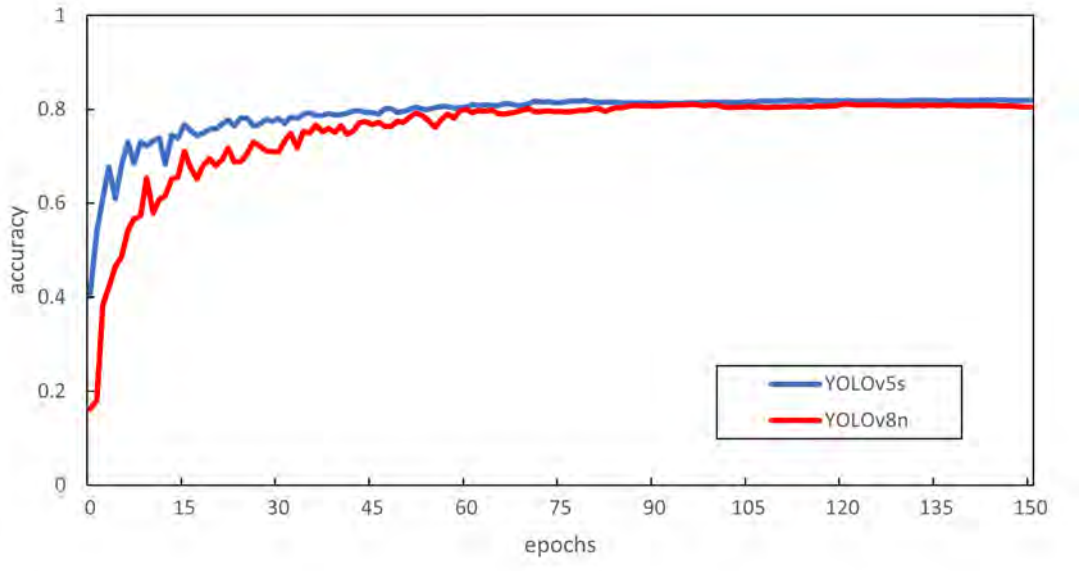
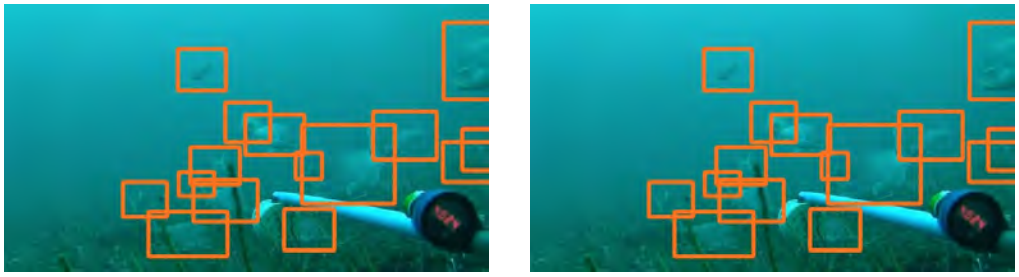


Figure 5.11: mAP_0.5 accuracy of YOLOv5ss and YOLOv8nn models on the Pondoland dataset.

5.3.1.3 Detection Qualitative Analysis

Similar to Section 5.2.1.3, Figure 5.12a and Figure 5.12b have overlapping bounding boxes and the bounding boxes also have a gap between the class. The main factors that could have contributed to this are the limited sample size of the images in the datasets and the IoU of 0.5, which was not high enough to suppress duplicate detections.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.12: Detection results for YOLOv5s and YOLOv8n on the Pondoland dataset.

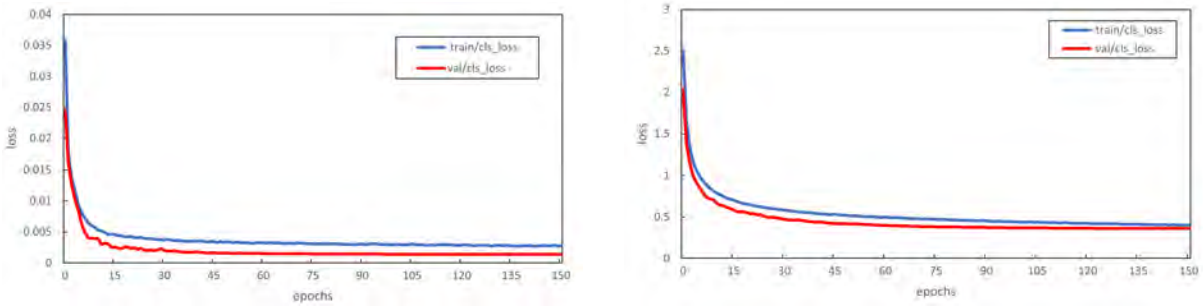
5.3.1.4 Class-Specific Classification

Both models achieve reasonably high accuracies across most classes, with YOLOv5s exhibiting slightly better performance on average. YOLOv5s consistently outperforms YOLOv8n throughout all 5 classes, demonstrating its effectiveness in object detection across a diverse range of classes in the Pondoland dataset.

For the following classes, *Slinger. C. Puniceus* and *Diplodus Hottentotus*, YOLOv5s perform better than YOLOv8n at detecting the classes. However, the models struggled to detect *Pisces-1* and *Pisces-2* this is due to the classes being underrepresented and not appearing as often as the other classes within the dataset.

Table 5.10: Predicted class for YOLOv5s and YOLOv8n on the Pondoland dataset.

Class	Images	Instances	YOLOv5s	YOLOv8nn
Slinger. C. Puniceus	650	2382	0.89	0.86
Diplodus Hottentotus	650	1241	0.84	0.82
Pisces-1	650	125	0.63	0.67
Pisces-2	650	112	0.65	0.65



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.13: Training and validation class loss curves for YOLOv5s and YOLOv8n on the Pondoland dataset.

Following the evaluation of classes within the Pondoland dataset, the next phase of Experiment 2.1 involves assessing the Precision, Recall, and F1 scores.

5.3.1.5 Classification Analysis

YOLOv5s demonstrate better precision, recall, and F1-Score across the different classes. The *Slinger. C. Puniceus* and *Diplodus Hottentotus* classes exhibit precision scores of 0.80 and 0.80, respectively, with corresponding recall values of 0.89 and 0.90.

Table 5.11: Performance of evaluation metrics for YOLOv5s on the Pondoland dataset.

Class	Precision	Recall	F1-Score
Slinger. C. Puniceus	0.80	0.89	0.79
Diplodus Hottentotus	0.80	0.90	0.80
Pisces-1	0.68	0.81	0.84
Pisces-2	0.64	0.79	0.84

Table 5.12: Performance of evaluation metrics for YOLOv8n on the Pondoland dataset.

Class	Precision	Recall	F1-Score
Slinger. C. Puniceus	0.80	0.91	0.81
Diplodus Hottentotus	0.84	0.96	0.82
Pisces-1	0.71	0.84	0.84
Pisces-2	0.69	0.82	0.84

YOLOv8n demonstrates competitive performance across the object classes. Notably, the *Slinger. C. Puniceus* class showcases a precision score of 0.80 and a recall of 0.91. The

Diplodus Hottentotus class exhibits impressive precision (0.84) and recall (0.96) scores. The *Pisces-1* and *Pisces-2* classes also demonstrate reasonable performance with precision scores of 0.71 and 0.69 and recall values of 0.84 and 0.82, respectively.

In summary, these results provide insights into the comparative performance of YOLOv5s and YOLOv8n on the Pondoland dataset, highlighting their respective strengths and weaknesses across different object classes. Both YOLOv5s and YOLOv8n exhibit remarkable performance on the Pondoland dataset, with YOLOv8n demonstrating a slight advantage in terms of recall and overall performance.

5.3.2 Experiment 2.2 – Richards Bay – Semi-Static Background

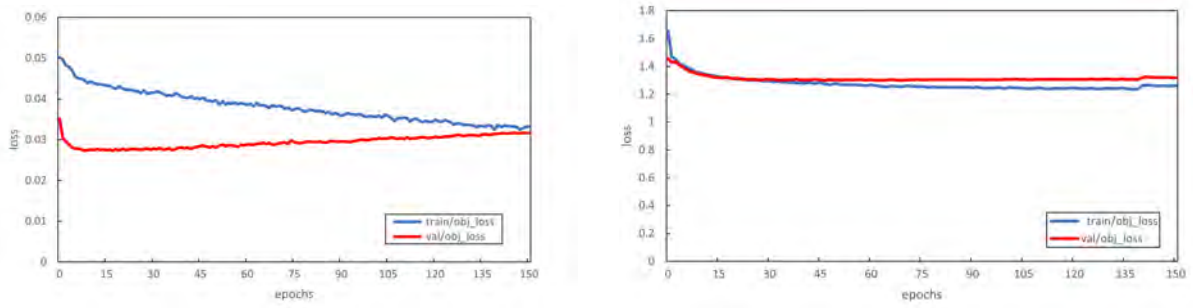
Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.3.2.1 Loss Curves

YOLOv5s exhibits overfitting throughout the training process and begins to converge in the later epochs, whereas YOLOv8n convergence in earlier epochs and exhibits overfitting closer to the later epochs and experiences far less overfitting compared to YOLOv5s. As a result, there is a noticeable gap between the training and validation sets early in training for YOLOv5s.

5.3.2.2 Detection mAP Curve

YOLOv5s exhibited a strong initial performance, surpassing YOLOv8n, but as training progresses, YOLOv8n eventually outperformed YOLOv5s, reaching its highest mAP_-0.5 of 0.939 in later epochs and YOLOv5s achieving the highest mAP_0.5 of 0.921 in an earlier epoch. Both models exhibit consistent fluctuations throughout the training process, as shown in Figure 5.15. YOLOv5s exhibits more minor fluctuations throughout the training process compared to YOLOv8n.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.14: Training and validation object loss curves for YOLOv5s and YOLOv8n on the Pondoland dataset.

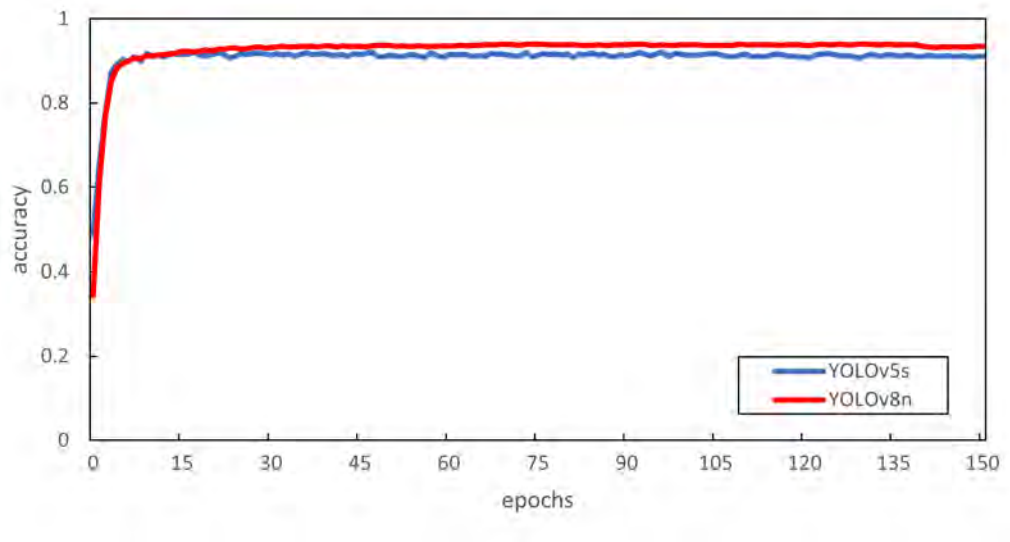
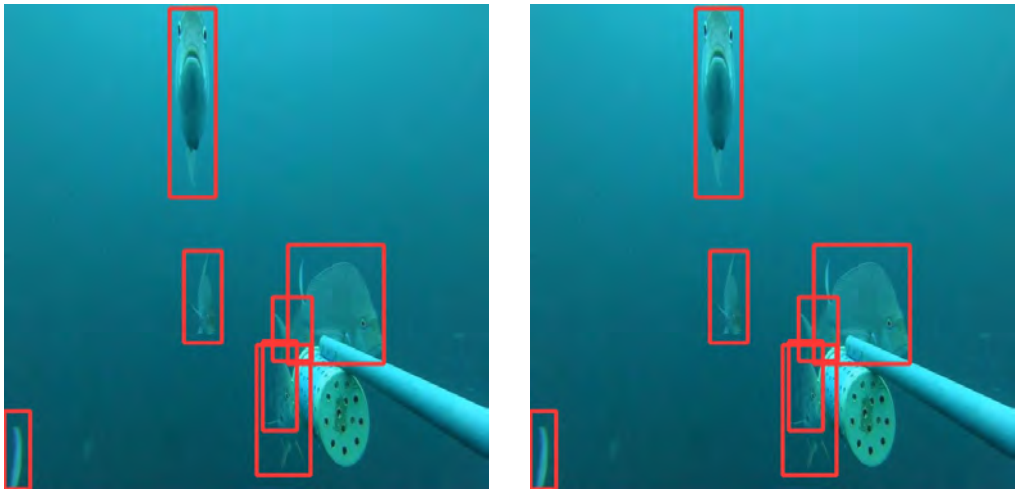


Figure 5.15: mAP_0.5 accuracy of YOLOv5s and YOLOv8nn models on the Aldabra dataset.

5.3.2.3 Detection Qualitative Analysis

Figure 5.16a and 5.16b show that both models accurately detected the fish within the dataset with an overlapped bounding box between the BRUV camera and the class. The models detected the side of the BRUV camera as a *Fish* class; this could have resulted because of the limited images within the dataset. The models did not have enough data to train with to learn the features effectively similar to Pondoland. Overall, the models detected the class accurately with small gaps between the objects compared to the Pondoland dataset, denoting a good IoU.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.16: Detection results for YOLOv5s and YOLOv8n on the Richards Bay dataset.

5.3.2.4 Class-Specific Classification

Both models achieve reasonably high accuracies. YOLOv8n obtained a class accuracy of 0.93 compared to YOLOv5s, achieving a class accuracy of 0.92. YOLOv8n consistently outperformed all images within the dataset as indicated by the high-class accuracy score as indicated by the mAP curve in Section 5.3.2.2 with YOLOv8n outperforming YOLOv5s throughout the training process.

Table 5.13: Predicted class for YOLOv5s and YOLOv8n on the Richards Bay dataset.

Class	Images	Instances	YOLOv5s	YOLOv8nn
Fish	4580	2382	0.92	0.93

5.3.2.5 Classification Analysis

The evaluation metrics of YOLOv5s show a precision score of 1.00, indicating perfect accuracy in classifying fish instances. The recall value of 0.96 demonstrates that YOLOv5s successfully captures 96% of actual *Fish* instances. The F1-Score of 0.91 reflects a strong overall performance, considering both precision and recall.

The evaluation metrics of YOLOv8n reveal a precision score of 1.00, denoting precise fish classifications. YOLOv8n excels in recall with a rate of 0.99, indicating its ability

Table 5.14: Performance of evaluation metrics for YOLOv5s on the Richards Bay dataset.

Class	Precision	Recall	F1-Score
Fish	1.00	0.96	0.91

Table 5.15: Performance of evaluation metrics for YOLOv8n on the Richards Bay dataset.

Class	Precision	Recall	F1-Score
Fish	1.00	0.99	0.91

to identify 99% of fish instances correctly. The F1-Score of 0.91 reflects a strong overall performance.

Both models exhibit remarkable performance on the Richards Bay dataset, with YOLOv8n demonstrating a slight advantage in recall and overall performance for the Fish class.

5.3.3 Experiment 2.3 – Tanganyika – Static Background

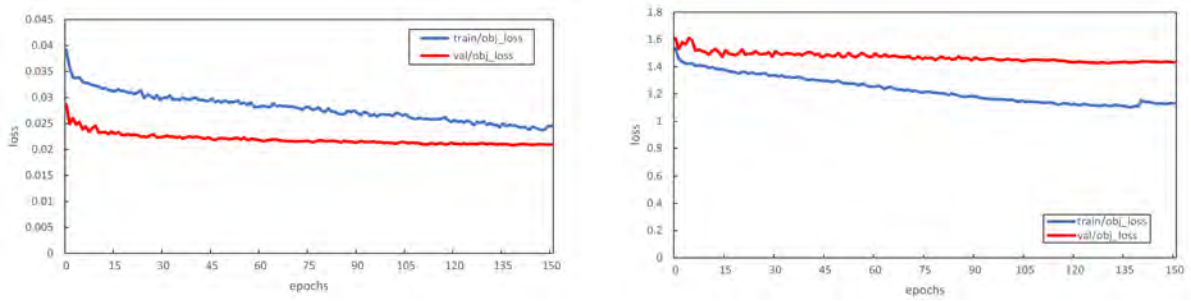
Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.3.3.1 Loss Curves

Both models exhibited early overfitting, where validation stops improving around Epoch 65. YOLOv5s train and validation stabilise and no longer decrease after Epoch 50. This means that in the initial epochs, the models quickly learn to perform well on training data but struggle to generalise to unseen validation data due to the limited training data.

5.3.3.2 Detection mAP Curve

YOLOv5s exhibited strong initial performance, surpassing YOLOv8n. YOLOv5s reached its highest mAP_0.5 of 0.911 at an earlier epoch than YOLOv8n. In contrast, YOLOv8n



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.17: Training and Validation Object Loss Curves for YOLOv5s and YOLOv8n on the Tanganyika dataset.

achieved its best mAP_0.5 score of 0.878 at a later epoch than YOLOv5s. Notably, both models demonstrated consistent performance, with minor fluctuations in mAP_0.5 throughout the training process, highlighting their robustness in object detection on tasks.

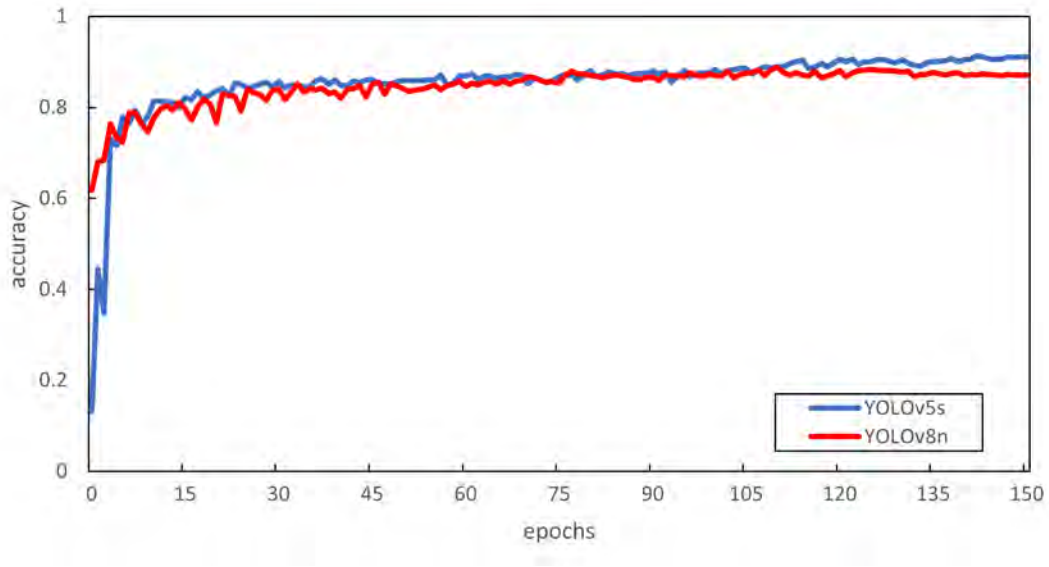


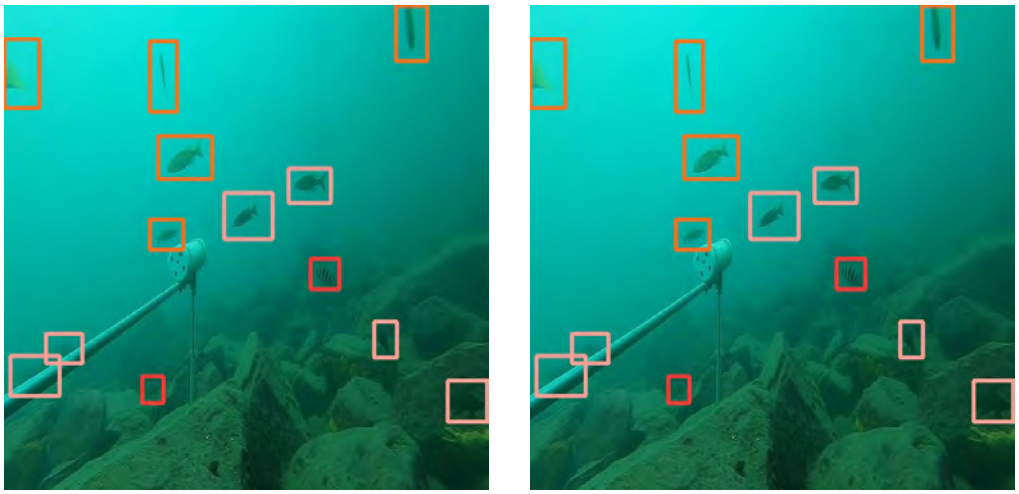
Figure 5.18: mAP_0.5 accuracy of YOLOv5s and YOLOv8n models on the Tanganyika dataset

5.3.3.3 Detection Qualitative Analysis

In Figure 5.19a and Figure 5.19b, the classes were detected accurately by both models with small gaps between the objects. The small gaps denote a good IoU-accurate object detection model. The models were able to detect the fish which were positioned behind

the BRUV accurately. As mentioned in Section 5.2.1.3, the models can detect smaller fish which human eyes could have missed during manual counting.

The background of the images has a slight bit of murk on it, but that did not deter the models from being able to detect the classes within the dataset accurately. This shows that the models have a strong ability to detect very small and hidden fish within images.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.19: Detection results for YOLOv5s and YOLOv8n on the Tanganyika dataset.

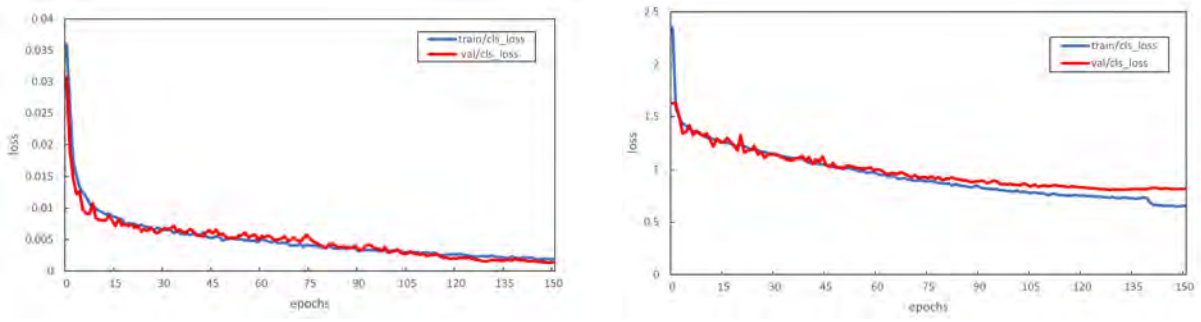
5.3.3.4 Class-Specific Classification

YOLOv5s consistently outperforms YOLOv8n throughout all 3 classes, demonstrating its effectiveness in object detection across a diverse range of classes in the Tanganyika dataset.

Table 5.16: Predicted classes for YOLOv5s and YOLOv8n on the Tanganyika dataset.

Class	Images	Instances	YOLOv5s	YOLOv8nn
Fish-1	3184	444	0.93	0.92
Fish-2	1215	1022	0.87	0.85
Fish-3	1045	363	0.94	0.91

Following the evaluation of individual class detection, the research assesses class loss training and validation on both models as displayed in Figure 5.20a and 5.20b, respectively.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.20: Training and validation class loss curves for YOLOv5s and YOLOv8n on the Tanganyika dataset.

The class loss curves demonstrate a consistent decrease in class loss throughout the training process, with minor fluctuations exhibited for both models. Overall, both models achieve low-class loss values throughout the training process.

5.3.3.5 Classification Analysis

YOLOv5s demonstrates higher precision and recall for the *Fish-1*, *Fish-2*, and *Fish-3* classes. Precision scores of 1.00 indicate that all identified instances are accurate for these classes, while recall values hover around 0.95 to 1.00, indicating successful detection of a vast majority of instances. F1-Scores are relatively high, ranging from 0.84 to 0.92, reflecting robust model performance.

Table 5.17: Performance of evaluation metrics for YOLOv5s on the Tanganyika dataset.

Class	Precision	Recall	F1-Score
Fish-1	1.00	1.00	0.89
Fish-2	1.00	0.98	0.92
Fish-3	1.00	1.00	0.87

Table 5.18: Performance of evaluation metrics for YOLOv8n on the Tanganyika dataset.

Class	Precision	Recall	F1-Score
Fish-1	1.00	0.95	0.85
Fish-2	1.00	0.97	0.88
Fish-2	1.00	0.94	0.84

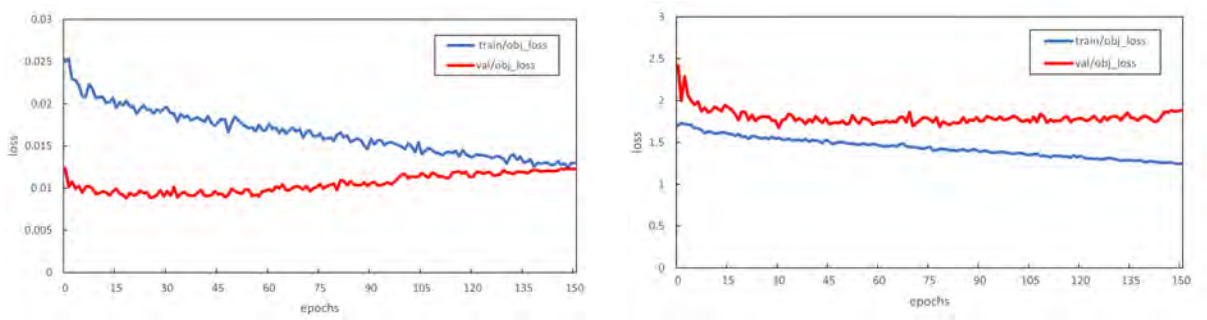
YOLOv8n showcases remarkable precision for the same classes, with all *Fish-1*, *Fish-2*, and *Fish-3* instances being accurate. Although recall values are slightly lower, varying from 0.94 to 0.97, they still indicate strong performance in detecting the majority of instances. F1-Scores for YOLOv8n range from 0.84 to 0.88, reinforcing the model's overall effectiveness in object detection on the Tanganyika dataset. These results collectively highlight the strong performance of both YOLOv5s and YOLOv8n on this dataset, with slight variations in recall and F1-Score.

5.3.4 Experiment 2.4 – 2013 Aldabra – Complex Background

Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.3.4.1 Loss Curves

Both models exhibit early overfitting, where the validation set stops improving around Epoch 120. This means that in the initial epochs, the models quickly learn to perform well on the training data but struggle to generalise to unseen validation data due to the limited training data. As a result, there is a noticeable performance gap between the training and validation sets early in training.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.21: Training and validation object loss curve for YOLOv5s and YOLOv8n on the 2013 Aldabra dataset.

5.3.4.2 Detection mAP Curve

YOLOv5s exhibited a strong initial performance, surpassing YOLOv8n, but as the training progressed, YOLOv8n eventually outperformed YOLOv5s, reaching its highest mAP_-0.5 score of 0.734 in later epochs.

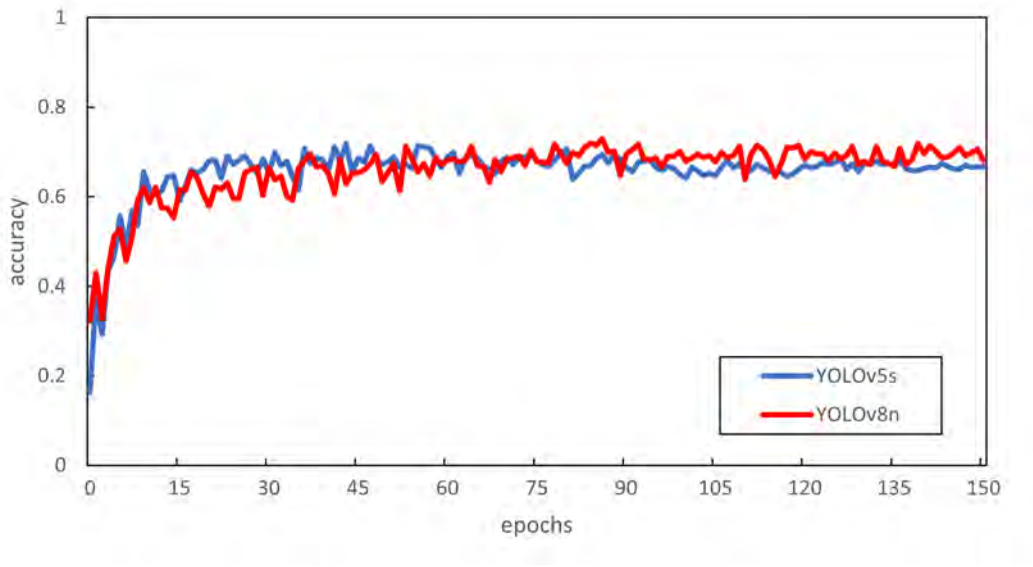


Figure 5.22: mAP_0.5 accuracy of YOLOv5s and YOLOv8nn models on the Aldabra dataset.

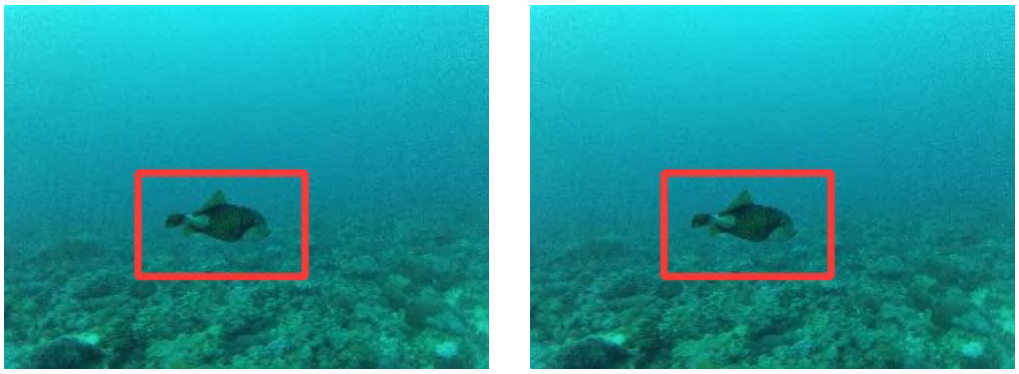
Both models exhibited consistent fluctuations throughout the training process, as shown in Figure, and both models have consistent overfitting.

5.3.4.3 Detection Qualitative Analysis

Figure 5.23a and 5.23b show that both models could detect the class within the images. However, the bounding boxes have a huge gap between the objects. The huge gap could be attributed to the low IoU used due to time and technical constraints, but overall, the models performed well. Another factor is that this dataset contained a few images, and the images within the dataset had many unclear images and varying image sizes.

This results in the models struggling to train with this dataset as attributed in results obtained from the class loss curve in Section 5.3.4.5. The datasets also contained a varying

degree of different backgrounds that contained a lot of distractions. Overall, the models could detect the classes with a gap between the bounding box and the class object in some images.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.23: Detection results for YOLOv5s and YOLOv8n on the Richards Bay dataset.

5.3.4.4 Class-Specific Classification

Both models achieve reasonable accuracies. YOLOv8n obtained a class accuracy of 0.714 compared to YOLOv5s, achieving a class accuracy of 0.712. YOLOv8n consistently outperformed YOLOv5s across all images within the dataset, as indicated by the mAP curve in Section 5.3.4.2.

Table 5.19: Predicted class for YOLOv5s and YOLOv8n on the 2013 Aldabra dataset.

Class	Images	Instances	YOLOv5s	YOLOv8n
Fish	1786	3189	0.71	0.71

5.3.4.5 Classification Analysis

In this analysis of object detection, both models achieved remarkably high precision scores of 1.00 for the *Fish* class. However, both models' recall and F1-Score metrics remain consistent, with recall around 0.86 and F1-Scores ranging from 0.69 to 0.70.

These results indicate strong and consistent performance in classifying the Fish class on this specific dataset for both YOLOv5s and YOLOv8n.

Table 5.20: Performance of evaluation metrics for YOLOv5s on the 2013 Aldabra dataset.

Class	Precision	Recall	F1-Score
Fish	1.00	0.86	0.70

Table 5.21: Performance of evaluation metrics for YOLOv8n on the 2013 Aldabra dataset.

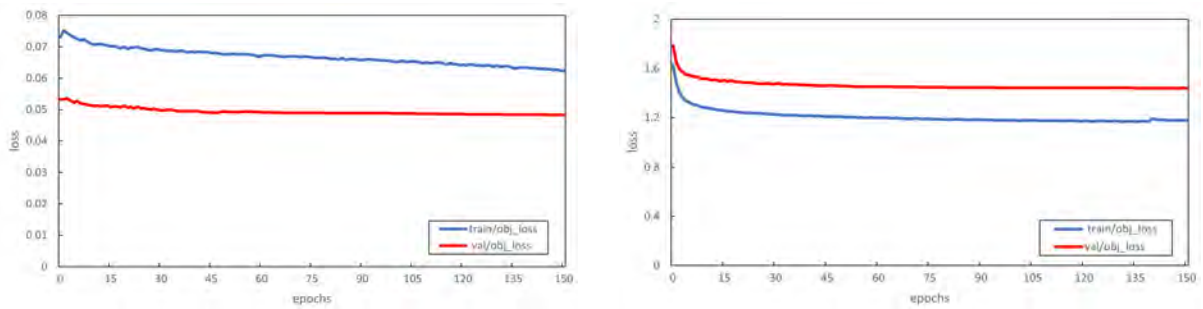
Class	Precision	Recall	F1-Score
Fish	1.00	0.86	0.69

5.3.5 Experiment 2.5 – 2020 Aldabra – Complex Background

Loss curves with mAP and visualisation are used to evaluate fish detection, followed by classification accuracies that measure the model's ability per class.

5.3.5.1 Loss Curves

Both models exhibit early overfitting, where validation stops improving around Epoch 45 for both models. YOLOv5s validation stabilises and no longer improves after 65 epochs, whereas YOLOv8n's validation stabilises at an earlier epoch and stops improving. This means that in the initial epochs, the models quickly learn to perform on training data but struggle with unseen data due to the limited training data.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.24: Training and validation object loss curves for YOLOv5s and YOLOv8n on the 2020 Aldabra dataset.

5.3.5.2 Detection mAP Curve

YOLOv5s exhibits a strong initial start, reaching its highest mAP_0.5 of 0.84 at an earlier epoch. YOLOv8n reaches its highest mAP_0.5 score of 0.85 at a later epoch than YOLOv5s. Both models exhibit minor fluctuations throughout the earlier stages of training and remain closer to each other throughout the later epochs in the training process.

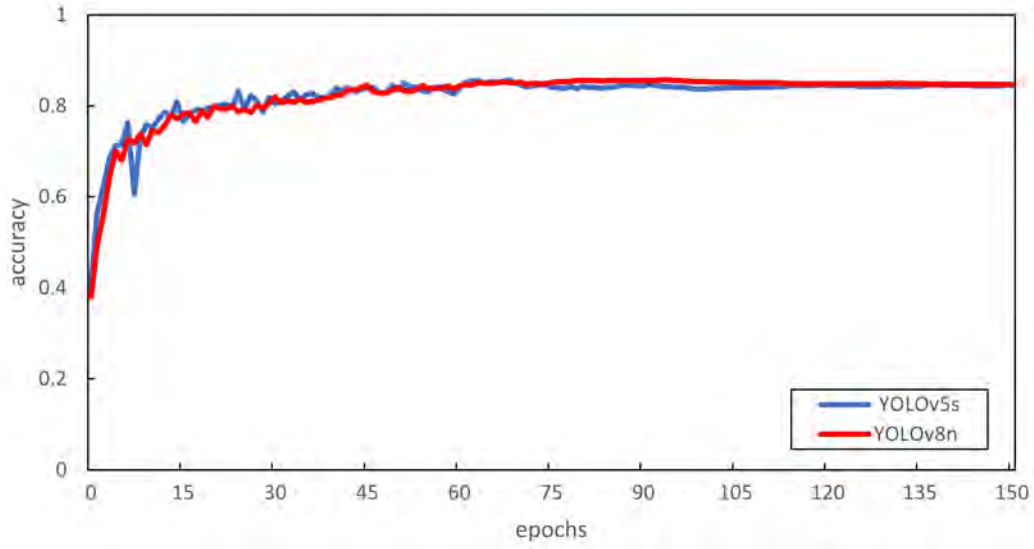


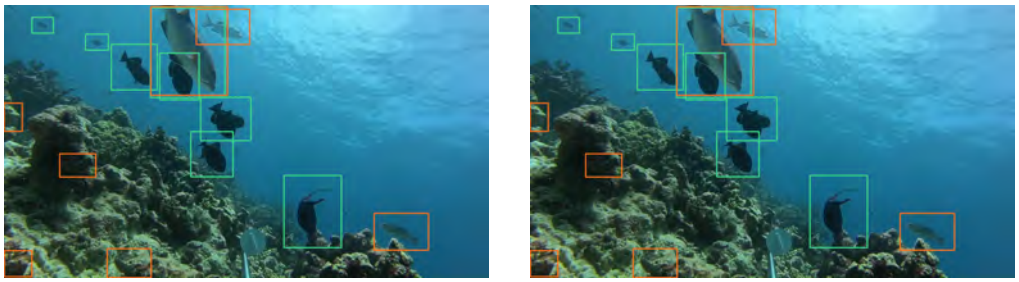
Figure 5.25: mAP_0.5 accuracy of YOLOv5s and YOLOv8nn models on the 2020 Aldabra dataset

5.3.5.3 Detection Qualitative Analysis

In Figures 5.26a and 5.26b, the models demonstrated accurate class detection; however, there was a noticeable gap between the object class and the bounding box. Notably, the models faced challenges detecting fish hidden within dark areas, similar to the difficulties encountered with hidden fish in the Tanganyika dataset. The obscured fish in this dataset, hidden within rocks, led to the models struggling and producing inaccurate bounding boxes.

The dataset presented challenges, including significant background distractions and a notable class imbalance. These factors adversely affected the models' performance, par-

ticularly in detecting individual classes. Despite these obstacles, the models managed to detect some classes with reasonable accuracy with a few minor incorrect detections.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.26: Detection results for YOLOv5s and YOLOv8n on the DeepFish dataset.

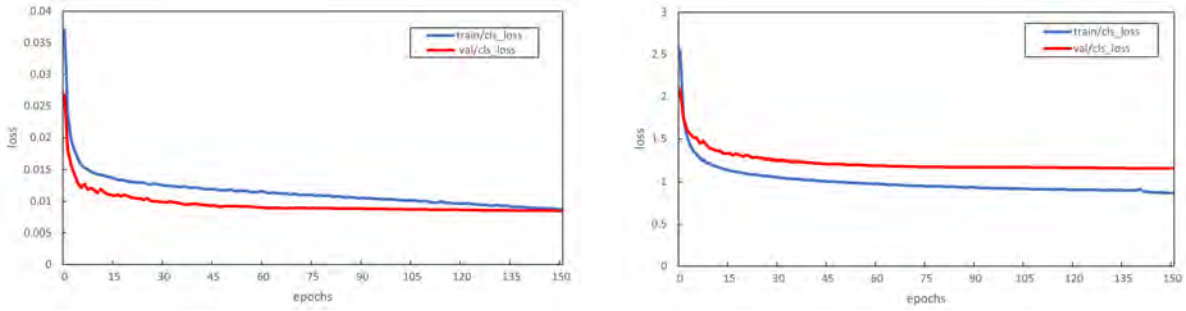
5.3.5.4 Class-Specific Classification

Differences emerge in class recognition scores between the models. For example, the *Caranx melampygus* class attains a classification score of 0.98 under YOLOv5s, which is further elevated to an impressive 0.99 with YOLOv8n, showcasing the latter's superior proficiency in accurate class detection.

Table 5.22: Predicted classes for YOLOv5s and YOLOv8n on the 2020 Aldabra dataset.

Class	**Images**	**Instances**	**YOLOv5s**	**YOLOv8nn**
Labroides dimidiatus	517	87	0.63	0.65
Carangoides	1250	257	0.88	0.92
Caranx melampygus	8761	7582	0.98	0.99
Epinephelus Tukula	967	57	0.92	0.92
Carcharhinus amblyrhynchos	3531	689	0.86	0.85
Naso Brevirostris	1185	177	0.75	0.77
Odonus niger	911	7640	0.99	0.99

The class loss curves demonstrate a consistent decrease in class loss throughout the training process, with minor fluctuations exhibited for both models. Overall, both models achieve low-class loss values throughout the training process. Also, there is a huge different in accuracy between the represented class *Caranx melampygus* and the least represented class *Labroides dimidiatus*. Both achieved 0.63 and 0.65 for *Labroides dimidiatus* while both models performed better with 0.98 and 0.99 for *Caranx melampygus*.



(a) YOLOv5s **(b)** YOLOv8n

Figure 5.27: Training and Validation Class Loss Curves for YOLOv5s and YOLOv8n on the Tanganyika dataset.

5.3.5.5 Classification Analysis

YOLOv5s demonstrated commendable precision and recall for most classes, with high F1-Scores. *Carangoides* and *Epinephelus Tukula* notably stood out with perfect precision scores, and *Caranx melampygus* exhibited remarkable precision and recall.

YOLOv8n, on the other hand, maintained similarly strong precision, recall, and F1-Scores, consistently exceeding 0.88 across most classes. *Caranx melampygus* and *Odonus niger* achieved near-perfect precision and recall, reaffirming their exceptional detection capabilities.

Table 5.23: Performance of evaluation metrics for YOLOv5s on the 2020 Aldabra dataset.

Class	Precision	Recall	F1-Score
Labroides dimidiatus	0.84	0.84	0.87
Carangoides	1.00	0.95	0.88
Caranx melampygus	1.00	0.93	0.98
Epinephelus Tukula	1.00	0.96	0.92
Carcharhinus amblyrhynchos	1.00	689	0.86
Naso Brevirostris	0.88	0.81	0.83
Odonus niger	1.00	5946	0.99

In the following Section 5.3.6, a comprehensive discussion of the presented results in the context of object detection on the Brackish and DeepFish datasets will be conducted, accompanied by a comparative analysis of findings reported in related research papers.

Table 5.24: Performance of evaluation metrics for YOLOv8n on the 2020 Aldabra dataset.

Class	Precision	Recall	F1-Score
Labroides dimidiatus	0.86	0.87	0.87
Carangoides	1.00	0.96	0.88
Caranx melampygus	1.00	1.00	0.99
Epinephelus Tukula	1.00	1.00	0.95
Carcharhinus amblyrhynchos	1.00	0.99	0.91
Naso Brevirostris	0.92	0.94	0.87
Odonus niger	1.00	1.00	0.99

5.3.6 Experiment 2 Discussion

In Experiment 2.1, the performance of YOLOv5s and YOLOv8n models on the Pondoland dataset was comprehensively assessed, with the models trained using specific hyperparameters as outlined in Table 5.1. The analysis revealed several noteworthy findings. Both YOLOv5s and YOLOv8n exhibited early overfitting, with validation loss plateauing around Epoch 65, a common occurrence when training deep learning models with limited data (Saini and Susan, 2023). While YOLOv5s achieved its highest mAP_0.5 earlier than YOLOv8n, both models demonstrated competitive mAP scores, with YOLOv5s maintaining a slight advantage in accuracy. This indicates that YOLOv5s may adapt more swiftly to the dataset, but the difference in mAP scores is relatively small, highlighting the strong overall performance of both models.

Class-specific analysis revealed that both models achieved high accuracies, with YOLOv5s generally outperforming YOLOv8n. However, underrepresented classes in the dataset, such as *Pisces-1* and *Pisces-2*, presented challenges for both models, emphasising the importance of dataset diversity. Furthermore, both YOLOv5s and YOLOv8n demonstrated competitive precision, recall, and F1-Scores, with YOLOv8n showing a slight advantage in the recall, potentially attributable to architectural differences or specific hyperparameter choices. Detection visualisations confirmed accurate object localisation and classification for both models. These results collectively underscore the capability of YOLOv5s and YOLOv8n in object detection on the Pondoland dataset while emphasising the importance of dataset diversity and hyperparameter optimisation to enhance their performance

(Sportelli *et al.*, 2023, Bhujel *et al.*, 2021, Khan *et al.*, 2022).

In Experiment 2.2, the comprehensive evaluation of YOLOv5s and YOLOv8n on the Richards Bay dataset yielded valuable insights. The utilisation of specific hyperparameters in Table 5.1 guided the evaluation. Both models displayed signs of overfitting, although they exhibited distinct patterns at different stages of training. YOLOv5s experienced early overfitting, leading to a pronounced gap between training and validation losses in later epochs. In contrast, YOLOv8n demonstrated a more resilient performance, converging at an earlier stage and experiencing milder overfitting in the later phases. These divergent behaviours suggest that variations in generalisation capabilities may be attributed to architectural distinctions (Gillani *et al.*, 2022). The subsequent analysis of the mAP curve provided insights into the dynamic performance of the models.

YOLOv5s ultimately achieved a higher mAP_0.5 score of 0.939, while YOLOv5s peaked at 0.921 in an earlier epoch. Both models exhibited fluctuations during training, with YOLOv5s displaying more stability and fewer minor fluctuations than YOLOv8n. This emphasises the adaptability of YOLOv8n over time and underscores the importance of closely monitoring performance throughout the training process (Hindarto, 2023). The assessment of class-specific accuracy revealed that both models excelled in accurately classifying fish instances, with YOLOv8n consistently outperforming YOLOv5s.

This trend aligns with the overall performance trends observed in the mAP curve, emphasising the significance of evaluating model performance on individual object classes due to potential variations in complexity. The evaluation metrics further confirmed exceptional precision, recall, and F1-Scores for the Fish class for both models, with YOLOv8n slightly outperforming YOLOv5s in the recall. These results collectively emphasise the robust object detection capabilities of both models, with YOLOv8n holding a slight advantage in terms of recall and overall performance, likely attributed to architectural improvements (Zhang *et al.*, 2023, Saini and Susan, 2023, Telaumbanua *et al.*, 2023).

In Experiment 2.3, YOLOv5s and YOLOv8n were assessed on the 2022 Tanganyika dataset. Both models displayed early signs of overfitting, with validation loss stabilising around Epoch 65. It's noteworthy that YOLOv5's training and validation loss appeared

to stabilise earlier, around Epoch 50. This behaviour suggests that, like in the previous experiments, the models rapidly learned to perform well on the training data but encountered difficulties in generalising to unseen validation data, emphasising the significance of addressing overfitting in future experiments.

The analysis of mAP curves revealed that YOLOv5s initially outperformed YOLOv8n on the Tanganyika dataset. YOLOv5s achieved its highest mAP_0.5 score at an earlier epoch compared to YOLOv8n. Conversely, YOLOv8n showed a slower but consistent rise in mAP_0.5, ultimately surpassing YOLOv5s. Both models demonstrated robust and competitive performance throughout training, with some variations in their learning dynamics. This performance contrast could be attributed to the dataset-specific characteristics of the 2022 Tanganyika dataset. The class-specific analysis of detection results indicated that YOLOv5s consistently outperformed YOLOv8n across all three classes in the Tanganyika dataset. This finding suggests that YOLOv5s excels in recognising a diverse range of object classes, reinforcing its versatility in object detection tasks. Class loss curves for both models demonstrated consistent decreases with minor fluctuations, further underscoring their effectiveness in class-specific training.

Evaluation metrics emphasised the exceptional precision and recall of YOLOv5s and YOLOv8n for the *Fish-1*, *Fish-2*, and *Fish-3* classes. Both models maintained precision scores of 1.00, indicating highly accurate detections, while recall values ranged from 0.95 to 1.00, suggesting the successful classification of most instances. F1-Scores remained high, indicating overall strong performance in object detection on the Tanganyika dataset. These results collectively highlight the efficacy of both models on this specific dataset, with YOLOv5s showcasing an edge in class-specific accuracy.

Visualisations of object detection further confirmed the models' competence in accurately localising and classifying objects within the Tanganyika dataset, with no noticeable discrepancies between YOLOv5s and YOLOv8n. The results suggest that both models are proficient in recognising object instances in different poses, scales, and conditions. In conclusion, the performance of YOLOv5s and YOLOv8n on the 2022 Tanganyika dataset reveals their robustness and adaptability to varying dataset characteristics, positioning

them as strong candidates for diverse object detection applications.

Experiment 2.4 evaluated YOLOv5s and YOLOv8n models on the 2013 Aldabra dataset. Both models exhibited early signs of overfitting in the training phase, with validation loss plateauing around Epoch 120. The analysis of the mAP curves demonstrated performance fluctuations, with YOLOv5s initially outperforming YOLOv8n but eventually being surpassed by YOLOv8n. This dynamic shift in performance from Experiment 2.2 to 2.3 is attributed to the unique dataset characteristics of the 2013 Aldabra dataset, which contains more complex object instances, distinct environmental conditions, and variations in object poses and sizes compared to the Richards Bay dataset.

Class-specific accuracy assessments highlighted both models' proficiency in classifying *Fish* instances, with YOLOv8n consistently outperforming YOLOv5s. This difference in class accuracy suggests that YOLOv8n may have a more robust ability to distinguish between fine-grained variations within the *Fish* class, which could be due to architectural variances or model optimisations specific to YOLOv8n. The evaluation metrics reinforced these findings, showing remarkable precision and recall scores, with both models achieving strong overall performance. However, the consistently high recall values for both models hint at features and patterns reliably detected by both YOLOv5s and YOLOv8n, underscoring their proficiency in fish instance recognition. Object detection visualisations further confirmed their robustness in accurately localising and classifying objects, with YOLOv8n exhibiting a slight advantage over YOLOv5s' recall and overall performance.

In Experiment 2.5, YOLOv5s and YOLOv8n were assessed on the 2020 Aldabra dataset. Both models displayed early signs of overfitting, with validation loss stabilising around Epoch 45 for both models. YOLOv5s exhibited earlier stabilisation in validation loss than YOLOv8n, but both models encountered similar challenges in generalising to unseen data due to limited training data, a trend consistent with previous experiments. The analysis of mAP curves revealed that YOLOv5s initially outperformed YOLOv8n on the Aldabra dataset, achieving its highest mAP_0.5 at an earlier epoch. YOLOv8n, however, reached its highest mAP_0.5 score at a later epoch, ultimately closing the gap between the two models. Both models exhibited stable performance throughout the training process,

suggesting their robustness in object detection tasks.

Class-specific analysis of detection results demonstrated some distinctions between YOLOv5s and YOLOv8n. Notably, YOLOv8n showcased superior proficiency in accurate class detection, particularly with the *Caranx melampygus* class, achieving an impressive recognition score. While both models showed consistency in class loss reduction, the results pointed to the strength of YOLOv8n in recognising specific classes within the Aldabra dataset. Evaluation metrics for individual classes reinforced the remarkable capability model's decision-making process and recall values for most classes. The results showcased the versatility of YOLOv5s and YOLOv8n in object detection and classification on the Aldabra dataset. Visualisations of object detection demonstrated accurate localisation and classification of objects by both models, indicating high precision and agreement with ground truth annotations.

Complex backgrounds, characterised by elements of clutter, intricate visual distractions, and variations in object scale and positioning, present challenges to deep learning models (Ren *et al.*, 2015), such as those utilised in the research. These complex backgrounds may lead to feature interference, as there could be features within the background that bear similarity to the objects of interest, thereby complicating accurate object differentiation. Furthermore, the presence of contextual information introduced by distinct environmental backgrounds may influence the decision-making process of the model. This observation aligns with the insights derived from the paper by (He *et al.*, 2016b), which highlights the importance of comprehending the nuances of environmental backgrounds and their potential impact on the accuracy of object detection models.

In all experiments, YOLOv5s and YOLOv8n consistently demonstrated robust object detection performance. YOLOv5s initially outperformed YOLOv8n, but both models achieved high mAP scores and excelled in class-specific recognition with minor dataset-specific variations. While YOLOv5s showed strong generalisation, YOLOv8n excelled in class recognition. These findings highlight the adaptability of both YOLO architectures for object detection, with the choice between them depending on dataset priorities and objectives.

5.4 Summary

This chapter presents and discusses the results obtained from multiple experiments, along with the validation of system parameters. The experiments conducted on the Brackish, DeepFish, Tanganyika, and Aldabra datasets revealed the effectiveness of both YOLOv5s and YOLOv8n in underwater object detection. These models consistently achieved high mAP scores, underscoring their accuracy in recognising objects within diverse underwater environments. YOLOv8n, in particular, exhibited superior performance in specific classes, making it well-suited for particular dataset requirements.

A comparison with previous models and research papers highlighted the substantial advancements introduced in YOLOv5s and YOLOv8n, resulting in improved mAP scores. Despite early overfitting tendencies typical in limited data deep learning training, these models demonstrated robust object detection capabilities. YOLOv5s initially outperformed YOLOv8n, but YOLOv8n, in most cases, caught up with YOLOv5s and outperformed their model, indicating their potential for accurate detection. Class-specific analysis and visualisations confirmed their precision in object classification.

In summary, both models proved effective for underwater object detection, with YOLOv5s excelling in generalisation and YOLOv8n consistently demonstrating class recognition capabilities. These findings provided valuable insights for selecting the most suitable model for various underwater object detection scenarios.

6

Conclusion and Future Work

This chapter concludes the thesis, highlights the contributions made towards the research, and provides directions for future work.

6.1 Conclusion

This research represents an advancement in underwater object detection, focusing on the detection and classification of fish. The main objective was to thoroughly understand and evaluate the use of YOLOv5 and YOLOv8 architectures, and in doing so, the research assessed their practical effectiveness in addressing detection and classification in challenging underwater environments.

The study commenced with a thorough analysis of existing literature exploring detection and classification strategies for underwater scenarios. This groundwork aimed to address common challenges encountered in these environments. Subsequently, an overview of YOLO's architecture and its evolving iterations through the years provided a framework for selecting the most suitable YOLOv5 and YOLOv8 versions for the research. Considering computational limitations and the nature of the chosen datasets, the smallest versions of both models were employed for use.

The research used seven datasets, encompassing a combination of publicly available datasets and custom-created datasets with detailed annotations. These datasets facilitated the comprehensive evaluation of YOLOv5 and YOLOv8 performance across various underwater scenarios represented within each dataset. Hyperparameter tuning, a crucial process for optimising model performance, was conducted using the Pondoland dataset due to its inherently complex background compared to the other datasets used in this research.

The findings reveal the individual strengths of YOLOv5 and YOLOv8 and signify a paradigm shift in underwater object detection. The substantial improvements in mAP scores demonstrate the impact of architectural enhancements and dataset-specific adaptations. These models are characterised by exceptional mAP scores, and robust class recognition capabilities.

The experiments showcased the effectiveness of YOLOv5 and YOLOv8 in underwater object detection, with notable achievements on datasets such as Brackish and DeepFish. Both models achieved high mAP scores, with YOLOv8 performing better in specific classes in diverse datasets like Brackish. Their success can be attributed to their advanced deep learning architectures, trained on comprehensive natural image datasets, enabling them to extract robust features for object detection.

Furthermore, these models' efficient detection and classification algorithms contribute to their real-time performance, making them well-suited for resource-constrained devices. Comparative analyses against prior models highlight the ongoing evolution of YOLO architectures, with YOLOv5 and YOLOv8 surpassing YOLOv4 and Tiny YOLOv4, particularly in overall mAP and class-specific accuracy on the same datasets.

The adaptability of YOLOv5 and YOLOv8 was evident across diverse datasets, including Pondoland, Richards Bay, Tanganyika, 2013 Aldabra, and 2020 Aldabra. While YOLOv5 demonstrated strong early adaptability and generalisation, YOLOv8 exhibited better class-specific classification, especially in datasets with fine-grained variations. However, challenges such as early signs of overfitting and the need for careful hyperparameter tuning were identified, emphasising the importance of addressing these issues for further model refinement.

The observed variations in performance across datasets underscore the nature of object detection tasks and highlight the need to consider dataset characteristics in model selection carefully. Despite these challenges, the consistent demonstration of robust object detection capabilities by both YOLOv5 and YOLOv8 positions them as reliable choices for diverse underwater object detection applications.

Furthermore, these models' efficient detection and classification algorithms contribute to their real-time performance, making them well-suited for resource-constrained devices. Comparative analysis against prior models highlights the ongoing evolution of YOLO architectures, with YOLOv5 and YOLOv8 surpassing YOLOv3, YOLOv4 and Tiny YOLOv4, particularly in overall mAP and class-specific accuracy on the same datasets.

In conclusion, this research provides valuable insights into the strengths, challenges, and comparative performance of YOLOv5 and YOLOv8 in underwater object detection. By contextualising the research's significance through comparisons with prior models and studies, it emphasises the advancements introduced by YOLOv5 and YOLOv8, establishing them as powerful tools and innovative solutions in the challenging domain of underwater object detection. The new releases of YOLO versions do not guarantee increased performance and accuracy but may require more or newer hardware resources.

6.2 Contributions

This study has achieved six objectives, each yielding distinct contributions that collectively enhance the knowledge and resources available in underwater fish detection. Firstly, an effort was dedicated to collecting and annotating datasets featuring underwater fish, encompassing a rich diversity of real-world scenarios and intricate background settings. This contribution is instrumental as it adds valuable data to the existing pool and provides researchers with well-annotated datasets essential for training and validating object detection models in underwater environments.

Building on this, the second objective involved the collection of publicly available datasets featuring underwater fish, further enriching the available resources. This contribution enhances the accessibility of datasets, fostering collaboration and allowing researchers to benchmark their methodologies against a broader spectrum of underwater scenarios. The public availability of these datasets is a tangible contribution to the research community, facilitating advancements in underwater fish detection beyond the scope of this study.

The third objective focused on designing and creating a cost-effective system for object detection and classifying within underwater fish datasets. This system not only aids in achieving the study's goals but also represents a methodological contribution. By providing a blueprint for an economically viable setup, the study contributes to the broader accessibility of research tools, potentially opening avenues for researchers with limited resources to engage in similar studies.

Moving forward, the fourth and fifth objectives involved a comprehensive evaluation of two state-of-the-art architectures, YOLOv5 and YOLOv8, for underwater fish detection. This addresses the need to assess the applicability of these architectures in underwater environments and contributes valuable insights into their relative performances. The quantitative measure of mAP and the nuanced analysis of precision and recall metrics offer a comprehensive understanding of these architectures' capabilities, providing researchers with informed choices for their specific underwater fish detection applications.

The culmination of these efforts in the sixth objective, evaluating the capability of YOLOv5 and YOLOv8 to classify different fish species within underwater images accurately, solidifies the study's impact. The specific focus on species recognition is crucial in ecological studies and fisheries management. The study, by assessing and providing insights into the classification capabilities of these architectures, contributes not only to the field of computer vision but also to the broader ecological and conservation applications of underwater fish detection.

In conclusion, the research objectives exhibit seamless interconnection where each object complements and builds upon the other to form a comprehensive and valuable set of contributions. Ranging from dataset enrichment to methodological advancements and architectural evaluations, these collective contributions serve to enhance the comprehension of underwater fish detection. The research effectively addresses the questions outlined in Section 1.4, thereby establishing a solid foundation for future research endeavours in this domain.

6.3 Future Work

Possible future work includes the following additional objectives:

1. Collect appropriate datasets and, thus, create systems for other underwater fish datasets, such as different water conditions from different environments worldwide.
2. Explore techniques to improve the detection of small objects, a critical aspect for applications like medical imaging, where the objects of interest are often very tiny.
3. Explore methods for enhancing multi-object classifying capabilities in object detection, focusing on maintaining object identity over time, which is critical for applications like autonomous driving and surveillance.
4. Expand the collected underwater fish datasets to provide a large training sample size, producing higher system accuracy.
5. Effectively combine appropriate models to produce one top-performing system for fish species detection and classifying instead of a single low-performing model.
6. Develop an application for the system for automatic fish detection that can be attached to an underwater camera for real-time detection.

References

Albawi, S., Mohammed, T. A., and Al-Zawi, S. Understanding of a convolutional neural network. *Institute of Electrical and Electronics Engineers*, 8:659, 2017.

Atik, M. E., Duran, Z., and ÖZGÜNLÜK, R. Comparison of YOLO Versions for Object Detection from Aerial Images. *International Journal of Environment and Geoinformatics*, 9:87–93, 06 2022. doi:10.30897/ijegeo.1010741.

Bengio, Y. and Lecun, Y. Convolutional networks for images, speech, and time-series. *Rm 4G332, AT&T Bell Laboratories, 101 Crawfords Corner Road*, 11 1997.

Bezdan, T. and Bacanin, N. Convolutional neural network layers and architectures. *Institute of Electrical and Electronics Engineers*, pages 445–451, 01 2019. doi:10.15308/Sinteza-2019-445-451.

Bhujel, A., Arulmozhi, E., Moon, B.-E., and Kim, H.-T. Deep-learning-based automatic monitoring of pigs' physico-temporal activities at different greenhouse gas concentrations. *Animals*, 11, 10 2021. doi:10.3390/ani11113089.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-y. YOLOv4: Optimal Speed and Accuracy of Object Detection. *Springer Link*, pages 1–2, 04 2020.

Boom, B., Haung, P., Beyand, C., Spampinato, C., Palazzo, S., He, J., Beauxis-Assalet, E., Lin, S., Chou, H., Nadarajan, G., Chen-Burger, J., Van Ossenbruggen, J., Giodano, D., and Hardman, D. Long-term underwater camera surveillance for monitoring and analysis of fish population. *International Conference on Computational Intelligence and Software Engineering*, 1:10–20, 2012.

Cong, A., Duong, L.-D., Huynh, H., Long, N., and Tran, N. A Model for Real-Time Traffic Signs Recognition Based on the YOLO Algorithm – A Case Study Using Vietnamese Traffic Signs, pages 104–116. Future Data and Security Engineering, 11 2019. ISBN 978-3-030-35652-1. doi:10.1007/978-3-030-35653-8_8.

Duggal, S., Manik, S., and Ghai, M. Amalgamation of video description and multiple object localization using single deep learning model. *Proceedings of the 9th International Conference on Signal Processing Systems*, 1:109–115, 2017.

Er, M., Zhang, Y., and Gao, W. Research Challenges, Recent Advances and Benchmark Datasets in Deep-Learning-Based Underwater Marine Object Detection: A Review. *TechRxiv*, 11 2022. doi:10.36227/techrxiv.19350389.v3.

Fisher, R., Shao, K.-T., and Chen-Burger, J. Overview of the Fish4Knowledge Project. *Springer Link*, pages 1–17, 03 2016. doi:10.1007/978-3-319-30208-9_1.

Garcia-D'Urso, N., Galán Cuenca, A., Pérez-Sánchez, P., Climent i Pérez, P., Guilló, A., Azorin-Lopez, J., Saval-Calvo, M., Guillén-Nieto, J., and Soler-Capdepón, G. The deepfish computer vision dataset for fish instance segmentation, classification, and size estimation. *Scientific Data*, 9:287, 06 2022. doi: 10.1038/s41597-022-01416-0.

Gasparovic, B., Mausa, G., Rukavina, J., and Lerga, J. Evaluating YOLOV5, YOLOV6, YOLOV7, and YOLOV8 in Underwater Environment: Is There Real Improvement? *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, 11 2023. doi:10.23919/SpliTech58164.2023.10193505.

Gillani, I., Munawar, M., Talha, M., Azhar, S., Mashkoor, Y., uddin, M., and Zafar, U. YOLOv5, YOLO-x, YOLO-r, YOLOv7 Performance Comparison: A Survey. *Springer*, pages 17–28, 09 2022. doi:10.5121/csit.2022.121602.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014. doi:10.1145/3422622.

Haruna, K., Ismail, M. A., Damiasih, D., Sutopo, J., and Herawan, T. A collaborative approach for research paper recommender system. *PLOS ONE*, 12:e0184516, 10 2017. doi:10.1371/journal.pone.0184516.

Harvey, E., Mclean, D., Frusher, S., Haywood, M., Newman, S., and Williams, A. The use of BRUVs as a tool for assessing marine fisheries and ecosystems: a review of the hurdles and potential. The University of Western Australia, 05 2013. ISBN 978-1-74052-265-6.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Institute of Electrical and Electronics Engineers*, 8:656–658, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *Institute of Electrical and Electronics Engineers*, pages 770–778, 06 2016b. doi:10.1109/CVPR.2016.90.

He, Y., Su, Y., Wang, X., Yu, J., and Luo, Y. An improved method MSS-YOLOv5 for object detection with balancing speed-accuracy. *Frontiers in Physics*, 10:1101923, 01 2023. doi:10.3389/fphy.2022.1101923.

Hindarto, D. Exploring YOLOv8 Pretrain for Real-Time Detection of Indonesian Native Fish Species. *Sinkron*, 8:2776–2785, 10 2023. doi:10.33395/sinkron.v8i4.13100.

Huang, Z., Lu, C., and Haung, L. A CNN Based Method for Fish Detection. *Springer Link*, 02 2018.

Hussain, M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines and Tooling*, 11:677, 06 2023. doi:10.3390/machines11070677.

Khan, M. G., Saeed, M., Zulfiqar, A., Ghadi, Y., and Adnan, M. A novel deep learning based anpr pipeline for vehicle access control. *IEEE Access*, 10:1 – 1, 06 2022. doi:10.1109/ACCESS.2022.3183101.

Koushik, J. and Hayashi, H. Improving Stochastic Gradient Descent with Feedback. *Springer*, 11 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi:10.1145/3065386.

Kuhlane, L., Brown, D., and Boby, A. Exploring the Incremental Improvements of YOLOv5 on Tracking and Identifying Great White Sharks in Cape Town. *Springer*, pages 1455–1473, 08 2023a. doi:10.1007/978-3-031-37963-5_98.

Kuhlane, L., Brown, D., and Boby, A. Exploring The Incremental Improvements of YOLOv7 on Bull Sharks in Mozambique. *Lecture Notes Series*, 506, 02 2023b.

Kuhlane, L., Brown, D., and Marais, M. Real- Time Detecting and Tracking of Squids Using YOLOv5. *Springer*, pages 1–5, 08 2023c. doi:10.1109/icABCD59051.2023.10220521.

Kuswantori, A., Suesut, T., Tangsrirat, W., Schleining, G., and Nunak, N. Fish Detection and Classification for Automatic Sorting System with an Optimized YOLO Algorithm. *Applied Sciences*, 13(6):3812, 2023.

Lee, Y.-H. and Youngseop, K. Comparison of CNN and YOLO for Object Detection. *Journal of the Semiconductor & Display Technology*, 19:85–92, 2020.

Li, L., Shi, G., and Jiang, T. Fish detection method based on improved YOLOv5. *Aquaculture International*, 31:1–18, 03 2023. doi:10.1007/s10499-023-01095-7.

Li, R. and Yang, J. Improved YOLOv2 Object Detection Model. *Springer*, pages 1–6, 05 2018. doi:10.1109/ICMCS.2018.8525895.

Li, X., Shang, M., Qin, H., and Chen, L. Fast accurate fish detection and recognition of underwater images with Fast R-CNN. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1–5. 10 2015. doi: 10.23919/OCEANS.2015.7404464.

Liang, H. and Song, T. Lightweight marine biological target detection algorithm based on YOLOv5. *Frontiers in Marine Science*, 10, 2023. ISSN 2296-7745. doi: 10.3389/fmars.2023.1219155.
URL https://www.frontiersin.org/articles/10.3389/fmars.2023.1219155

Lin, M., Chen, Q., and Yan, S. Network In Network. *Springer*, 12 2013.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. *Computer Vision and Pattern Recognition*, 2016.

Lumaug, R. G. and Neva, M. Fish tracking and counting using image processing. *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, pages 1–4, 2018.

Magalhães, R. and Peixoto, H. Object recognition using convolutional neural networks. *Springer*, 11 2019. doi:10.5772/intechopen.89726.

Mahasin, M. and Dewi, I. Comparison of CSPDarkNet53, CSPResNeXt-50, and EfficientNet-B0 Backbones on YOLOv4 as Object Detector. *International Journal of Engineering, Science and Information Technology*, 2:64–72, 09 2022. doi:10.52088/ijesty.v2i3.291.

Misund, O., Kolding, J., and Fréon, P. Fish capture devices in industrial and artisanal fisheries and their influence on management. *Handbook of Fish Biology and Fisheries, Vol II*, pages 13–36, 01 2002.

Muksit, A. A., Hasan, F., Emon, M. F. H. B., Haque, M. R., Anwary, A. R., and Shatabda, S. YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment. *Elsevier*, 12:1545, 10 2022. doi:https://doi.org/10.1016/j.ecoinf.2022.101847.

Nalmpanti, M., Chrysafi, A., Meeuwig, J. J., and Tsikiras, A. C. Monitoring marine fishes using underwater video techniques in the mediterranean sea. *Springer Link*, 30(2):145–162, 2023. doi:10.12345/mar_ecol_res.2020.12345.

Nelson, J. and Solawetz, J. Responding to the Controversy about YOLOv5. *Roboflow*, 09 2020.
URL https://blog.roboflow.com/yolov4-versus-yolov5/

Nguyen, N., Huynh, K., Vo, N., and Van Pham, N. Fish detection and movement tracking. *International Conference on Computational Intelligence and Software Engineering*, 1:1–4, 2018.

Patel, C., Patel, A., and Patel, D. Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*, 55:50–56, 10 2012. doi:10.5120/8794-2784.

Pedersen, M., Haurum, J., Gade, R., Moeslund, T., and Madsen, N. Detection of marine animals in a new underwater dataset with varying visibility. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2019.

Purwono, P., Ma'arif, A., Rahmaniar, W., Imam, H., Fathurrahman, H. I. K., Frisky, Z., and Haq, Q. M. U. Understanding of Convolutional Neural Network (CNN): A Review. *Institute of Electrical and Electronics Engineers*, 2:739–748, 01 2023. doi:10.31763/ijrcs.v2i4.888.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXivLabs: experimental projects with community collaborators*, 2:10–20, 2016.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.02640, 2015.
URL http://arxiv.org/abs/1506.02640

Redmon, J. and Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv*, 04 2018.

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, 06 2015. doi:10.1109/TPAMI.2016.2577031.

Rodriguez, A., Rico-Daiz, A., Rabunal, J., Puetras, J., and Pena, L. Fish monitoring and sizing using computer vision. *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 1908:419–428, 2015.

Ruan, Z., Wang, G., Lin, X., Xue, J.-H., and Jiang, Y. Deformable Part-Based Model Transfer for Object Detection. *IEICE Transactions on Information and Systems*, E97.D:1394–1397, 05 2014. doi:10.1587/transinf.E97.D.1394.

Saini, M. and Susan, S. Tackling class imbalance in computer vision: a contemporary review. *Artificial Intelligence Review*, 56:1–57, 07 2023. doi:10.1007/s10462-023-10557-6.

Saleh, A., Laradji, I. H., Konovalov, D. A., Bradley, M., Vazquez, D., and Sheaves, M. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Scientific Reports*, 10(1):14671, 2020. doi:https://doi.org/10.1038/s41598-020-71639-x.

Selçuk, B. and Serif, T. A comparison of yolov5 and yolov8 in the context of mobile ui detection. *Springer*, pages 161–174, 08 2023. doi:10.1007/978-3-031-39764-6_11.

Sharma, N., Baral, S., Paing, M., and Chawuthai, R. Parking time violation tracking using yolov8 and tracking algorithms. *Sensors*, 23:5843, 06 2023. doi:10.3390/s23135843.

Singh, S. and Mukhopadhyay, S. Detection of moving objects based on enhancement of optical flow. *Optik - International Journal for Light and Electron Optics*, 145, 07 2017. doi:10.1016/j.ijleo.2017.07.040.

Solawetz, J. and Francesco, J. What is YOLOv8? The Ultimate Guide. *Roboflow*, pages 1–2, 01 2023.

Sportelli, M., Apolo, O. E., Fontanelli, M., Frasconi, C., Raffaelli, M., Peruzzi, A., and Perez-Ruiz, M. Evaluation of YOLO Object Detectors for Weed Detection in Different Turfgrass Scenarios. *Applied Sciences*, 13:8502, 07 2023. doi:10.3390/app13148502.

Sun, T., Chen, H., Liu, H., Lou, H., and Duan, X. HPS-YOLOv7: A High Precision Small Object Detection Algorithm. *Springer*, 04 2023. doi:10.21203/rs.3.rs-2813484/v1.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi:10.1109/CVPR.2015.7298594.

Telaumbanua, A., Larosa, T., Pratama, P., Fauza, R., and Husein, A. Vehicle detection and identification using computer vision technology with the utilization of the yolov8 deep learning method. *sinkron*, 8:2150–2157, 10 2023. doi:10.33395/sinkron.v8i4.12787.

Toh, Y., Ng, T., and Liew, B. Automated Fish Counting Using Image Processing. *International Conference on Computational Intelligence and Software Engineering*, 1:1–5, 2009.

Villon, S., Mouillot, D., Chaumont, M., Darling, E., Subsol, G., Claverie, T., and Villéger, S. A deep learning algorithm for accurate and fast identification of coral reef fishes in underwater videos. *PeerJ Preprints*, page e26818v1, 03 2018. doi:10.7287/peerj.preprints.26818v1.

Wang, H. and Xia, N. Underwater Object Detection Method Based on Improved Faster RCNN. *Springer*, 02 2023.

Wang, R., Liang, F., Mou, X., Chen, L., Yu, X., Peng, Z., and Chen, H. Development of an Improved YOLOv7-Based Model for Detecting Defects on Strip Steel Surfaces. *Coatings*, 13:536, 03 2023. doi:10.3390/coatings13030536.

Wang, X., Liu, C., Zhang, W., Wang, Y., Zhao, H., Yang, S., and Liu, X. Underwater fish detection and recognition using single-shot multibox detector. *Sensors*, 18(3):701, 2018.

Xing, B., Sun, M., Ding, M., and Han, C. Fish Sonar Image Recognition Algorithm Based on Improved YOLOv5. *Aquaculture International*, 09 2023. doi:10.21203/rs.3.rs-3369498/v1.

Zhang, J., Zhang, J., Zhou, K., Zhang, Y., Chen, H., and Yan, X. An Improved YOLOv5-Based Underwater Object-Detection Framework. *Sensors*, 23:3693, 04 2023. doi:10.3390/s23073693.

Zhang, M., Xu, S., Song, W., He, Q., and Wei, Q. Lightweight Underwater Object Detection Based on YOLOv4 and Multi-Scale Attentional Feature Fusion. *Remote Sensing*, 13:4706, 11 2021. doi:10.3390/rs13224706.

Zhang, S., Zhen, A., and Stevenson, R. GAN Based Image Deblurring Using Dark Channel Prior. *Electronic Imaging*, 2019, 01 2019.

Zhang, T., Li, S., Li, T., and Wang, L. Real-Time Fish Detection and Tracking for Underwater Robots Using Mask R-CNN. *Sensors*, 20(19):5433, 2020.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. ByteTrack: Multi-object Tracking by Associating Every Detection Box. *Springer Link*, pages 1–21, 10 2022.

Zhao, K., Dong, W., Liu, C., and Xue, Y. Fish detection and tracking in underwater videos based on YOLO algorithm. *Measurement*, 183:109–119, 2021.

A

Datasets

The complete code and limited data are available at the Github Repository[1]. An alternative location for the data is: Google Drive[2]. The complete set of data is available on request via luxolokuhlane123@gmail.com or d.brown@ru.ac.za email addresses.

The rest of this appendix contains samples of the data used for experiments.

A.1 Brackish Dataset

Figure A.1: Samples of images in the Brackish dataset.

[1]https://github.com/luxolokuhlane/ObjectDetection
[2]https://drive.google.com/drive/folders/0ABeM-H8VkVlSUk9PVA

A.2 DeepFish Dataset

Figure A.2: Samples of images in the DeepFish dataset.

A.3 Pondoland Dataset

Figure A.3: Samples of images in the Pondoland dataset.

A.4 Richards Bay Dataset

Figure A.4: Samples of images in the Richards Bay dataset.

A.5 Tanganika Dataset

Figure A.5: Samples of images in the Tanganika dataset.

A.6 2013 Aldabra Dataset

Figure A.6: Samples of images in the 2013 Aldabra dataset.

A.7 2020 Aldabra Dataset

Figure A.7: Samples of images in the 2020 Aldabra dataset.