SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I9P104

Volume 13 Issue 9, 45-73, September 2026

Original Article

Advancing Underwater Object Detection: A Comparative Study of YOLOv5s, YOLOv8s, and YOLOv11s with Bias Analysis

Milind Shah[1], Sandipkumar R Panchal[2]

[1]*Computer Engineering, Dr. Subhash University, Junagadh, Gujarat, India.*
[2]*School of Engineering and Technology, Dr. Subhash University, Junagadh, Gujarat, India.*

[1]*Corresponding Authors : milindshahcomputer@gmail.com*

Received: 06 August 2026 Revised: 05 September 2026 Accepted: 11 September 2026 Published: 29 September 2026

Abstract *- Environmental distortions and dataset-induced bias are challenges for underwater object detection due to the impact on model generalization. This research paper focuses on bias in the RUOD dataset and compares the performance of YOLOv5s, YOLOv8s, and YOLOv11s with 9,800 training images. Precision, Recall, and mean Average Precision (mAP) were used to perform a comprehensive analysis. The comparative results demonstrate that YOLOv11s has the highest performance when evaluated by mAP@0.5 (0.8721), followed by YOLOv8s and YOLOv5s, with slight differences among the models. The bias analysis of the training and validation sets showed a significant class imbalance, with the dominant classes being fish (17.5%) and echinus (15.15%), and the underrepresented classes being turtle (5.44%) and jellyfish (3.62%). Class-wise evaluation, however, suggests that the accuracy of detecting classes does not directly relate to the frequency of the classes, as there are several classes with low representation that are detected with high accuracy. The results show that there is a complex interplay between the bias in the datasets and the performance of the models, and that architectural improvements are not enough to entirely compensate for the bias effects. This research work gives insights into bias-aware evaluation to enhance underwater object detection systems.*

Keywords *- Underwater object detection, YOLO, Deep learning, Object detection, Dataset imbalance, Performance evaluation, Computer vision.*

1. Introduction

During the last twenty years, object detection has demonstrated remarkable advancements [1-4]. The major drawback of object detectors is that they are vulnerable to the problem of spatial bias, and specifically, they work poorly when it comes to detecting objects towards the edges of the image. Spatial bias has long been missing suitable methods of measurement and identification, and little is known of its origins and the extent to which it is present [5]. Regrettably, the detectors in reality cannot work equally well throughout the spatial zones, but they often exhibit a clear performance degradation along the image edge. Such constraints are made even worse in domain-specific problems like underwater object detection, whereby additional biases are brought about by the environmental conditions.

It is a phenomenon is known as spatial bias, but it is expected in object detection and somehow has long been overlooked by the detection community. Leaving the issue aside could lead to severe safety risks and the probability of huge losses (property damage). As an illustration, a fire detector can be effective in identifying fire in the central region and become ineffective in identifying fire in the extreme regions of the image. This type of fire alarm system is not reliable because the central zone of the lens will only take a small percentage of the image. The spatial robustness of Convolutional Neural Networks (CNNs) [6-10] has recently arrived, due to the understanding of how small image changes (e.g., color jitter or translation) affect the accuracy of image classification in the direction of that much desired translation invariance. The classifier has it been found that the classifier can make completely different predictions when the same object is presented at a different spatial position [9, 10].

Figure 1 shows a series of concepts of spatial bias in object detection organized around a core concept, which breaks down into its definition, affected image regions, CNN-based empirical evidence, domain-specific amplification, and the existing measurement gap - all of which explain why spatial bias is an important yet historically underaddressed challenge in the detection literature.

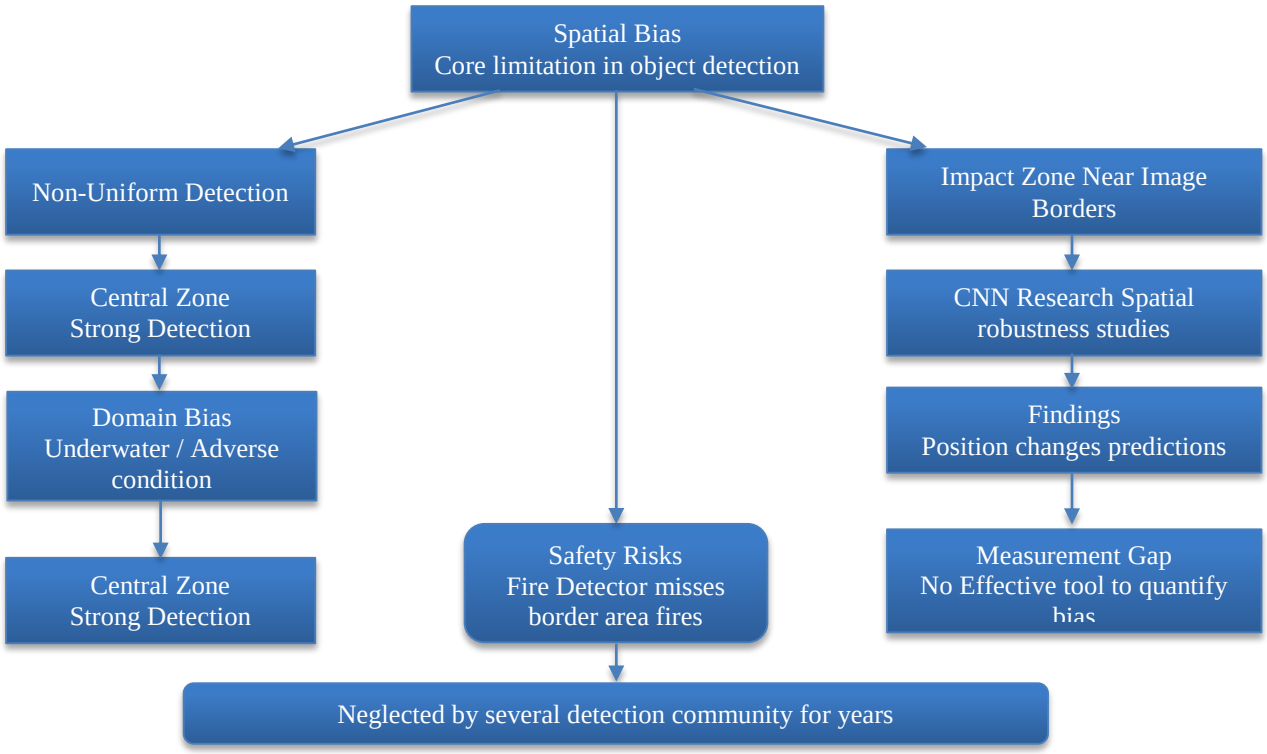


Fig. 1 Conceptual overview of spatial bias in object detection

Nevertheless, the problem of insufficient methods for measuring and detecting spatial bias has been an open issue over the years. The conventional assessment, i.e., the AP metric, quantifies the overall image zone detection performance, and it offers no information to guide the spatial strength of detectors; one can barely know where and by how much the performance is compromised. Thus, an evaluation protocol is especially critical since it will have a chance to increase a deeper insight into the spatial bias and provide a means to develop the methodology. Although there have been considerable advances in reducing inaccuracy and improving the performance of object detection machine learning models, other types of bias, including those related to the presence of data imbalance, environmental conditions, and machine design, have not been extensively studied. This is of special concern to the underwater object detection problem, in which phenomena like light attenuation, color distortion, and limited annotated data impose other biases on models and thus influence their performance. Previous studies have indicated that biases, class imbalances, and disparities exist in computer vision datasets, especially in areas like autonomous driving, where they can cause unfair object classification as well as wrong predictions of Vulnerable Road Users (VRUs) [11], [12]. The presented findings point to the larger role of dataset-driven biases, which will tend to be even more severe in more demanding conditions like underwater object detection.

In most object detection datasets, classes are highly imbalanced; that is, some classes have many more instances than others [13-16]. The result of this imbalance is that it may give rise to models that lead to a biased identification of dominant classes and the inability to identify less frequent but critical classes [17-19]. These biases may often be passed on to the training data, resulting in skewed class detection performance [20, 21]. In the underwater object detection task, the problem is even more prominent in the context of insufficiently annotated data and asymmetric distribution of categories.

The implications of dataset bias and class imbalance cannot be ignored, as they may have a severe impact on the reliability and safety of perception systems utilized in the real world. In adverse scenarios, such as underwater situations, it is particularly important to ensure robustness and fairness, and a failure to detect can lead to significant operational risk.

Figure 2 demonstrates that bias in detecting ML objects has three sources of origin: data imbalance, environmental conditions, and model design. These converge in the context of underwater detection, where the attenuation of light, distortion of color, and limited annotated data contribute to the further amplification of each source. This compounded effect has two crucial consequences, namely, class imbalance effects (when rare classes are missed and bias is inherited from the training data) and operational risks (such as unreliable detection and safety failures). Both results justify the fact that just models are required, which can be robust enough to work in adverse conditions of the real world.

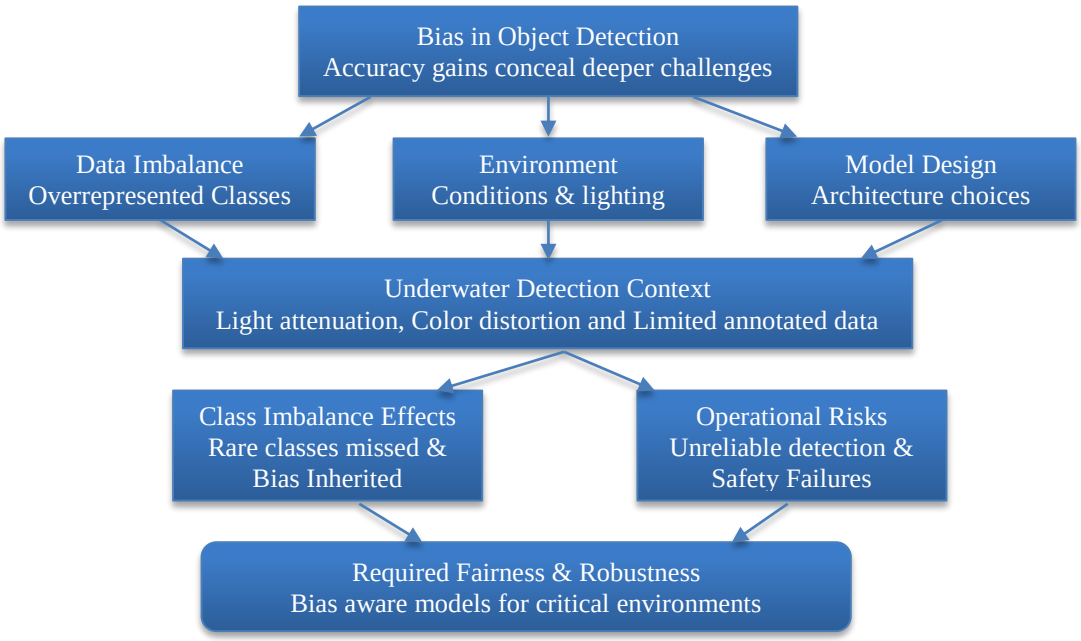


Fig. 2 Sources and consequences of bias in underwater ML object detection

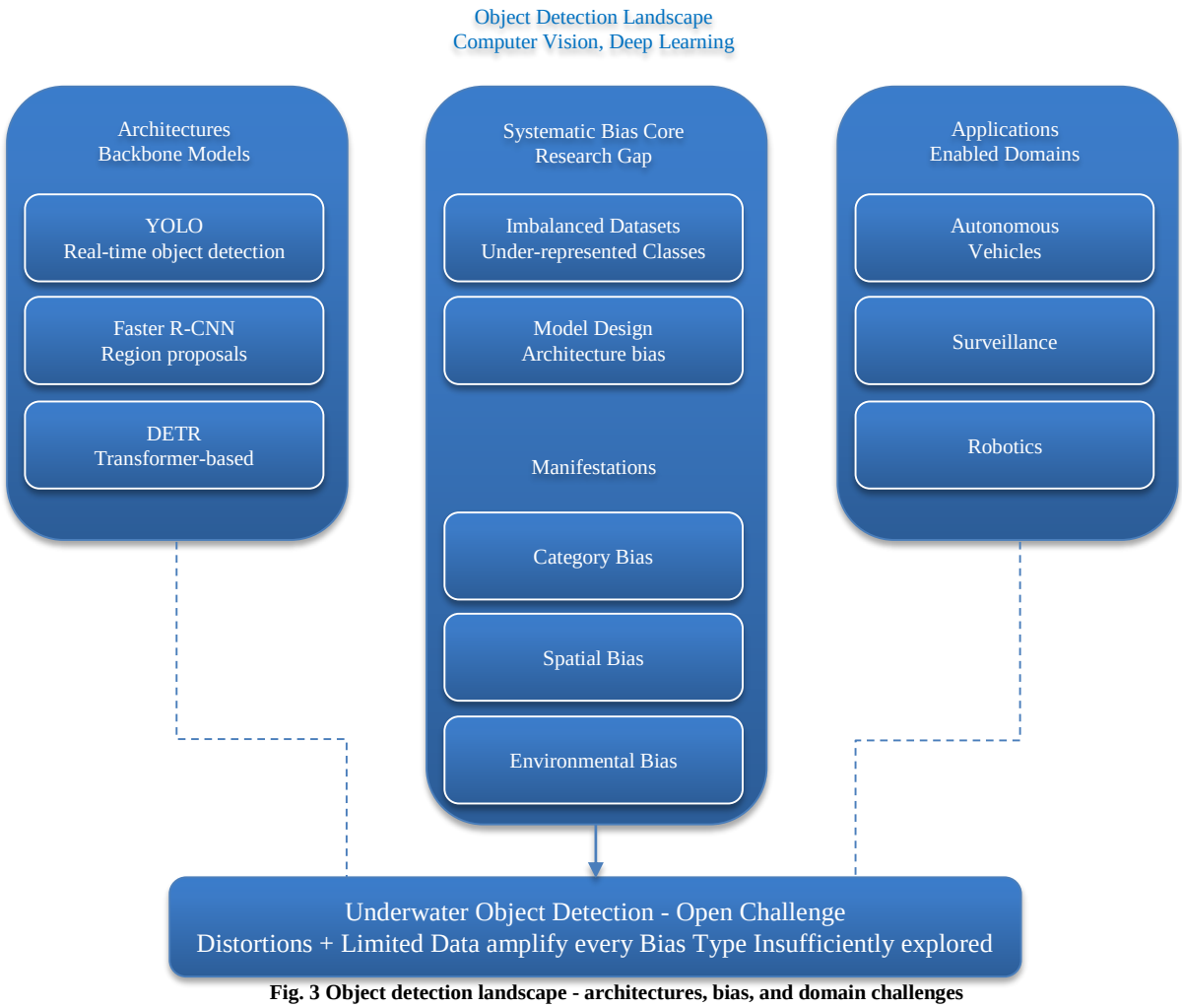


Fig. 3 Object detection landscape - architectures, bias, and domain challenges

1.1. Bias in Object Detection: Recent Advances and Challenges

Object detection is an essential computer vision problem that has made great steps with the help of deep learning systems such as YOLO, Faster R-CNN, and DETR, and now finds use in many areas. Although these have been made, object detection systems tend to be biased, as they are characterized by systematic performance differences across object categories, spatial locations, and environmental conditions. These biases are normally due to unbalanced datasets and natural properties of models. Recent studies have only started to discuss robustness and generalization, but bias is an area that has not been studied well, especially in domain-specific problem solutions like underwater object detection, where environmental distortions and limited data indeed make the task even more difficult [22-24].

Figure 3 illustrates how three large deep learning architectures (YOLO, Faster R-CNN, DETR) regulate object detection in various fields such as autonomous vehicles and robotics, but systematically cause biases based on skewed datasets and architecture. These biases are presented as category, spatial, and environmental performance gaps. The most important open challenge is underwater object detection, in which all bias types are combined, and where the distortion of the environment and low-frequency data increase the bias.

1.2. Primary Sources of Bias in Underwater Object Detection

There are various sources of bias in underwater object detection algorithms, which have a great impact on the detection performance. Such biases are caused by environmental factors, the methodology of data collection, and the limitations of the algorithms, which result in discrepancies in the outcomes of the various underwater conditions.

The environmental factors are also important since light scattering and absorption cause optical distortions, whereas unequal lighting conditions deteriorate the quality of the image and enhance noise, complicating the process of object recognition [25, 26]. Besides this, biases in data collection, such as domain shift, which is caused by changes in points of acquisition, equipment, or time, may lead to varying training and deployment conditions, thus decreasing the model's generalization [27]. This is further enhanced by the fact that there is limited diversity in the datasets since most datasets do not cover the entire spectrum of underwater conditions [26].

Bias is also caused by algorithmic limitations. Algorithms that only use single-modal data, like only optical or acoustic measurements, can do well in complicated environments where one of the modalities is not reliable [28]. In addition, detectors can miss certain objects and produce inaccurate performance due to the difficulty in extracting features, especially when the size of the object is small or when the objects are hidden in shadows or have low contrast [29]. To prevent these, new strategies have considered multi-modal fusion, including the use of optical and acoustic data, to enhance robustness and minimize bias in operating under a wide range of underwater conditions.

1.3. Impact of Underwater Environmental Complexity on Detection Bias

Underwater conditions are highly variable in visibility, illumination, and the dynamics of a scene, which further increases detection bias in an object detection system. Light attenuation, scattering, and distortions caused by movement are some factors that deteriorate visual properties and add discrepancies among data samples [29, 30]. Consequently, models that are trained under certain conditions are not likely to work effectively in environments that they have not seen or are not represented sufficiently, which results in biased detection results.

Along with the problem of environmental degradation, there are other issues like camouflage, low contrast, and similarity with the background that make it hard to differentiate objects from the background, which can be addressed through algorithms. These do not only lower detection accuracy but also bias object categories, which only increases bias in model predictions. Moreover, these problems are worsened by the use of single-modal data and the existence of domain changes between training and deployment environments, which constrain the strength and the generalization of current approaches [27, 28].

The latest developments in multi-modal solutions and adaptive learning techniques can provide possible solutions to overcome these difficulties, but there is still an open question on how to successfully deal with bias in highly dynamic underwater space.

1.4. Research Gap

Even with these advances, there are still a amount of research gaps. Previous works explored various deep-learning and YOLO based frameworks for underwater object detection, but as performance comparisons are often made with different training conditions, pre-processing techniques, input resolution, or evaluation settings, it can be challenging to conclude whether the experimental performance improvement is due to the architecture or the experimental setup.

Furthermore, the comparative performances of lightweight models such as YOLOv5s, YOLOv8s, and YOLOv11s have not been thoroughly explored with a common experimental setup on the RUOD benchmark. In particular, there is a need for a systematic analysis, taking not only the complete detection accuracy into account but also the behavior of the converging sets, the qualitative detection performance, the class-wise confusion, and the impact of class imbalance.

To fill this gap, in this study, the YOLO series of models, namely YOLOv5s, YOLOv8s, and YOLOv11s, were evaluated in a controlled comparative manner on the RUOD dataset for underwater object detection. To achieve a fair comparison of all three architectures, the same set of preprocessing steps, training conditions, input resolution, and evaluation protocol were used. Besides traditional detection metrics, the research compares the training convergence, representative outputs of detection, confusion matrix, and class imbalance in the dataset to offer a thorough evaluation of model behaviour. The study also examines the general trend of performance gains in the newer YOLO architecture to determine whether these gains are sufficient across different aspects of the evaluation or just the aggregate mAP.

1.5. Research Objectives

The objective of this research is to systematically study the effects of environmental degradation, class imbalance, and spatial and light variations on the performance of underwater object detection models, and analyze the cause of the bias. It also aims to make a controlled comparison of representative YOLO-based detectors, look into the most impactful sources of detection errors, and analyze their performance in generalization to underwater conditions. The study aims to develop an assessment framework that is robust, fair, and reliable under the influence of biases and to propose strategies for mitigating the negative effects of bias in underwater object detection systems.

1.6. Paper Contribution

This paper presents a comprehensive investigation of bias in underwater object detection, combining theoretical analysis with experimental validation to better understand its sources, impact, and mitigation strategies. The main contributions of this research are summarized as follows:

- This paper systematically frames bias in underwater object detection by linking modern detection advances with underwater-specific environmental challenges.
- It presents a structured review of underwater object detection methods, spanning traditional feature-based techniques and deep learning approaches.
- It identifies and organizes the major sources of object detection bias, including spatial, lighting, class-imbalance, and object-relativity biases.
- It explains how underwater conditions such as turbidity, low contrast, and illumination variation amplify detection bias and degrade generalization.
- It develops a comparative experimental methodology using representative YOLO-based models and consistent evaluation protocols.
- It reports experimental findings that expose architecture-specific sensitivity to underwater bias through quantitative, qualitative, and ablation-based analysis.
- It discusses key limitations and future research directions for building fairer and more robust underwater detection systems.

1.7. Paper Organization

The paper is divided into 9 sections, and each section describes the corresponding research topics, experiments, and findings in the domain of object detection based bias detection and mitigation in deep learning approaches. Figure 4 presents the structure of the research, which is organized into parts and subsections to present, evaluate, and implement bias in underwater object detection.

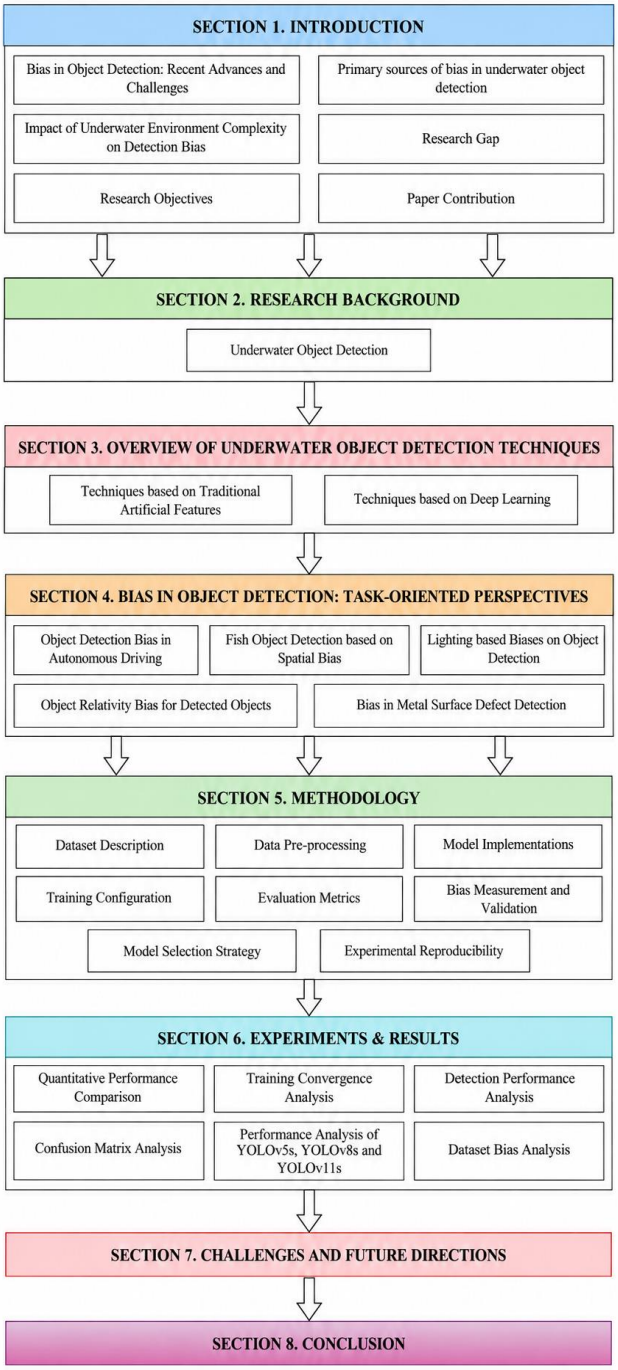


Fig. 4 Paper workflow and organization

2. Research Background

2.1. Underwater Object Detection

In recent years, the field of computer vision that is focused on object identification, has been much talked about. Deep learning's efficiency has led to a variety of deep learning models for object detection. There are generally two kinds of deep learning object detectors: one-stage and two-stage [31, 32]. In two-stage detectors, the fine-tuning process is a single-stage process for the object detection duty.

Single-stage object detection algorithms offer the benefit of making the location and category of the object as the output of a single forward pass, which is not needed in two-stage algorithms. This results in quicker detection time. Larger-scale feature maps are generated by SSD [2, 33] that predict the location and type of an object, which enables object detection of various sizes. To solve the problem of unbalanced positive and negative examples in single-stage object detection, the RetinaNet [34] algorithm was proposed to adopt the focal loss, which has a significant influence on the detection accuracy. The YOLO family of algorithms, such as YOLO-v2, YOLO-v3, and YOLO-v4, is a popular family of fast object detection algorithms [35-37].

For object detection operations, the position detection area is first determined in two stages, and then the target area is determined in the second stage. One way to improve the learning process for creating candidate areas is with the help of the Region Proposal Network (RPN), which is proposed by Faster R-CNN [38]. Cascade RCNN [39] cascades multiple Faster RCNNs on top of each other, allowing it to progressively refine the target location and target detection. Object Underwater Detection (UOD) is one of the objectives in underwater image recognition, which aims to detect both the type and location of objects in underwater images, similar to other object detection [40]. Underwater images are, however, limited in their visibility due to high noise, fuzzy edges, poor contrast, and color variation in UOD activities.

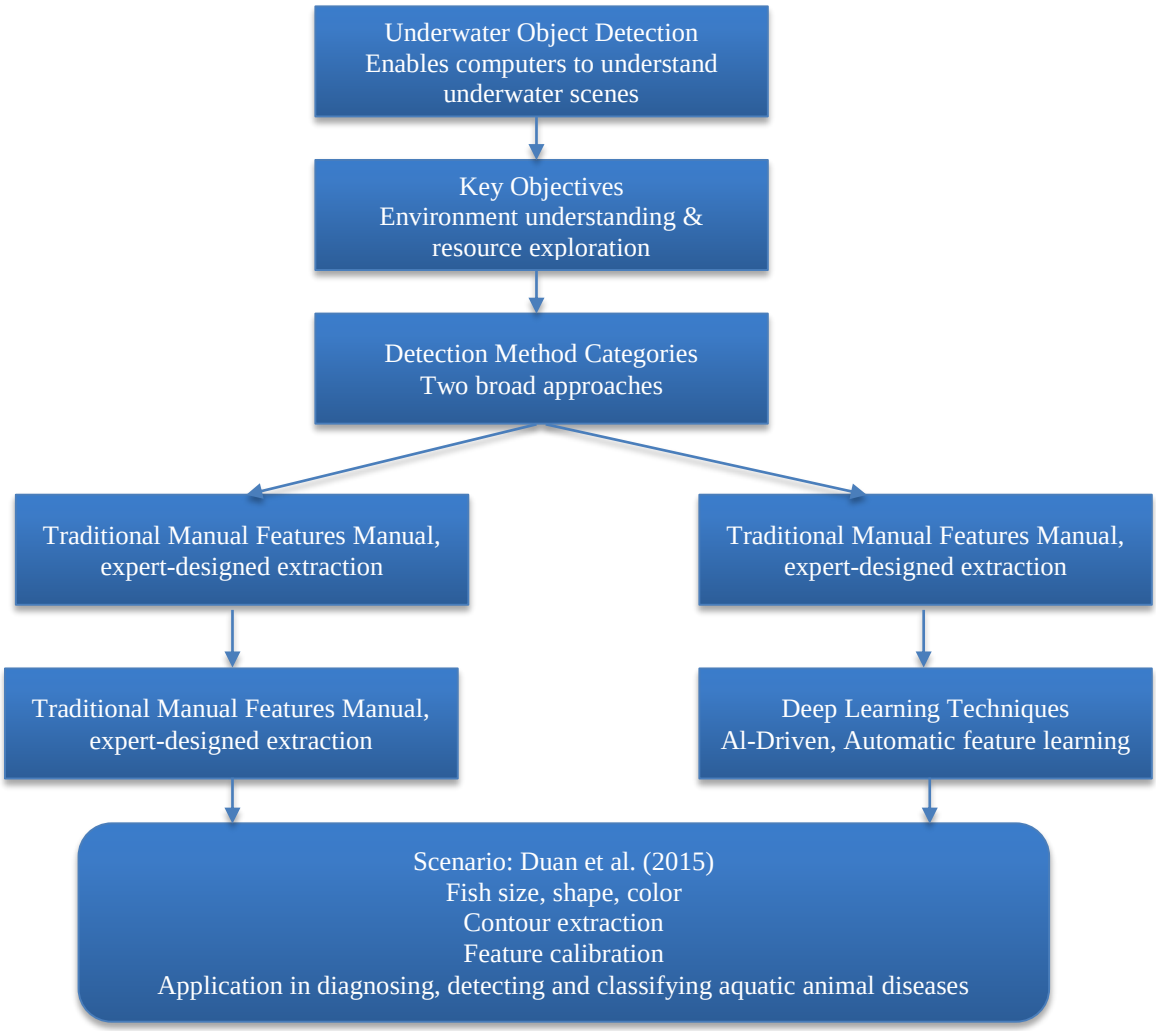


Fig. 5 Hierarchical overview of underwater object detection

3. Overview of Underwater Object Detection Techniques

In underwater computer vision and image processing, object detection is the key goal in order to allow computers to make sense of underwater scenes. His potentiality is not only important in the study of the underwater environment but also in the development of the underwater environment, with its immense importance in the exploration and exploitation of resources. Over the past few years, advancements in underwater object detection research have undergone significant change, as compared to the use of conventional manual features to the adoption of deep learning methods. To begin with, conventional manual features were mostly applied during the initial stages of research. These, however, had severe constraints when used in real-life underwater conditions. Moreover, the majority of detection algorithms for underwater object detection are based on the manual design of feature extraction, an activity that also requires professional skills and difficult algorithm debugging. But this method has only limited generalizability and a low level of detection, thus, it has not been developed in related areas.

The discipline of underwater object detection has recently put Artificial Intelligence (AI) technology into scope, because of the efforts of academics at universities and other research institutions. Numerous approaches have been developed in this space can be grouped into two classes: traditional feature-based methods and deep learning. For example, Duan et al. [41] reviewed all the recent advances in fish size, shape, color, and others using computer vision. These have included image capture, contour extraction, feature calibration, and computation, which may be accomplished using computer vision to analyze, identify, and classify aquatic animal diseases. Based on the above, this component primarily focuses on the underwater objects detection method based on deep learning analysis and classical artificial characteristics, and brings in the input of numerous field experts [42].

Figure 5 shows a five-level pyramid that leads down to the general field of underwater object detection by its major objectives, categories of detection methods, and two simultaneous branches of techniques - the traditional manual features and the deep learning approach. Duan et al. (2015) can be used as a research scenario since it presents a practical implementation with the analysis of fish attributes, including size, shape, color, contour extraction, feature calibration, and classification of aquatic diseases.

3.1. Techniques based on Traditional Artificial Features

Feature extraction approaches that are manually developed, and standard classification algorithms, are the cornerstones of underwater object detection techniques now in use. Typical approaches include sensing techniques like sonar and optical imaging, with subsequent handcrafted feature extraction like texture, shape, color, and motion patterns. On top of these features, traditional machine learning algorithms are applied to accomplish object recognition tasks.

Among these, the features based on texture are important in characterizing the surface properties of imagery from underwater scenes. For example, Shi et al. used a Gray-Level Co-occurrence Matrix with a Support Vector Machine to automatically identify the outline of underwater cages. This method involves calculating statistical texture information from grayscale images and then using this information to accurately extract structural edges in complex underwater scenes.

Although they work well in certain applications, they suffer from the drawback of depending on handmade features, which can be time-consuming and less effective in dynamic settings. Underwater object detection has been greatly improved by deep learning, including the use of Convolutional Neural Networks (CNNs), which can learn features automatically and enhance detection accuracy.

3.2. Techniques based on Deep Learning

The quick evolution of deep learning algorithms in the past several years has allowed underwater object detection to make incredible advances. The ability of Convolutional Neural Networks (CNNs) to automatically learn relevant characteristics in complicated underwater images is largely responsible for their rapid rise to popularity. As a result, these approaches generally achieve higher accuracy and robustness in detection and recognition tasks compared to traditional, handcrafted feature-based methods. There are a number of studies that have added to these improvements. Ge et al. introduced a single-stage underwater detection framework using anchor-based feature structure and dual feature enhancement, for instance. They combine several backbone networks to better account for the contextual information and enable multi-scale object detection. In a similar fashion, Lei et al. modified the YOLOv5 algorithm for underwater environments with a twin-transformer backbone, an improved multi-scale feature fusion module, and a new confidence loss module for better detection performance in challenging environments.

These developments have come despite the fact that there is still a shortage of labeled underwater image data. To overcome this, researchers have been looking at different measures to enhance model performance. A good example is Zeng et al., who used adversarial learning techniques in combination with Faster R-CNN to generate extra training samples and enhance detection capability by training both together. The techniques proposed by Zurowietz and Nattkemper are another successful method called Unsupervised Knowledge Transfer (UnKnoT) using data augmentation, namely scale transfer, which is used for replicating training samples and deriving similar object classes between different image sets.

4. Bias in Object Detection: Task-Oriented Perspectives

Figure 6 represents a taxonomy of bias in object detection systems based on task-specific situations and operational circumstances. It points out five sources of bias: (1) bias in autonomous driving cases, (2) spatial bias in fish object detection, (3) bias in light-based cases, (4) object relativity bias, and (5) bias in metal surface defect detection. These categories are discussed in detail as explained below.

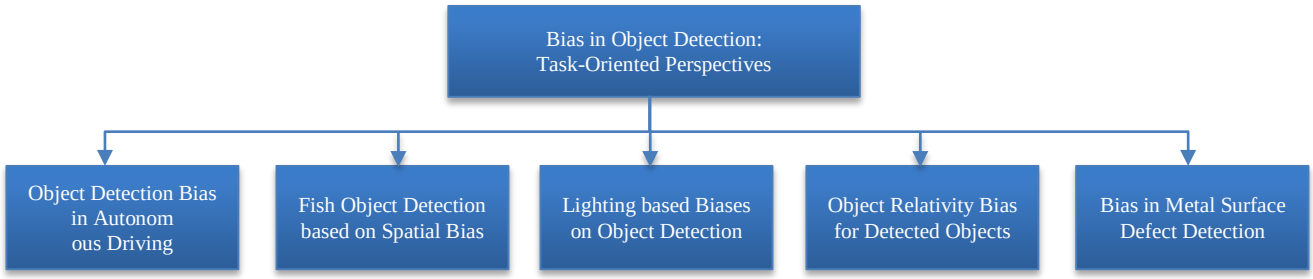


Fig. 6 Bias-related applications

4.1. Object Detection Bias in Autonomous Driving

In Reference [50], Dewant Katare et al. explore the current state-of-the-art techniques for object detection and classification in AI systems, which are evaluated using metrics like the selectivity score and cosine similarity. This study is especially geared towards collaborative processing and weight-based model updates for perception tasks in vehicular edge environments. This research examines various input class combinations, input class viewpoint variations, and data variety to see how these affect the model's behavior. The results indicated that methods such as invariance, cosine similarity, and selectivity scores are effective in measuring the level of bias during training. This will be a step forward toward more balanced and reliable AI models using vehicle edge technologies.

Moreover, the research proposes strategies to detect and identify bias in AI systems in real-world situations, such as driverless cars. Experiments carried out using a biased car dataset emphasize the role of training and testing data that are diverse and well-distributed. The diversity helps to improve generalisation and minimize algorithmic bias. To check for bias detection, the model is tested with different levels of data diversity with the help of the key indicators that are used during the training stage like: Selectivity score and Cosine similarity. Also, a system trained on the nuScenes dataset is able to detect various objects, including pedestrians and vehicles, to show applicability. Future research can investigate edge-cloud distributed machine learning frameworks, particularly for more vulnerable and underrepresented road users, such as pedestrians and cyclists. These classes tend to be under-represented when compared with the dominant classes like "vehicles" or "traffic signs". The emergence of autonomous systems will require a combination of cosine similarity, selectivity score, and invariance metrics to gain deeper insights into bias in neural networks. The analysis of these metrics at both the layer and block level can increase the interpretability and generalization and reduce bias of AI models, when compared to similar measurements for other object categories such as pedestrians, cyclists, or motorcyclists.

4.2. Fish Object Detection based on Spatial Bias

This article provides a dataset using multiple platforms, such as field imagery and digital resources, relevant to target detection of fishes. The 1038 images in 16 categories include common fish species such as carp and hairtail. Images of much more valuable fish species, such as grouper, can also be seen. Optimization and refinement of images further enhanced the quality of data closer to real-world applications and hence feature learning. In addition, extensive data prioritization, including classification, cleaning, and annotation, was carried out to ensure accurate and reliable support during neural network training.

Based on this dataset, a more accurate detection framework, referred to as BIAS-YOLO, is developed using the YOLOv5 architecture. The original Spatial Pyramid Pooling (SPP) module is enhanced by modifying its internal information-transfer mechanism, enabling improved learning and integration of multi-scale feature representations. Furthermore, a spatial bias mechanism is introduced, which dynamically adapts to the spatial arrangement of convolutional features through inversion and convolution operations on the feature map. This mechanism enables the network to capture global contextual information from convolutional layers more effectively. The redesigned SPP module is also facilitated by a lightweight spatial bias mechanism that ensures maximum utilization of multi-scale information.

The experimental results using a proposed dataset show a difference of 3.1% in mAP@0.5 between the baseline YOLOv5 model and BIAS-YOLO. To verify our proposed approach, we conducted multiple ablation tests, with the primary measure of success being our mAP@0.5. According to the results, BIAS-YOLO always achieves the best performance results across the board and leads the original

YOLOv5 network by 3.1% in performance. Another sign of the strength of the approach is that when omitting the SBB module, it is the only one that is able to deliver the best performance as measured by the mAP@0.5. Although the SPPF model contains the fewest parameters due to its compact architecture, the additional parameters introduced by the proposed approach are justified by the substantial improvement in detection performance. Also, the proposed network has better performance in terms of GFLOPS, which demonstrates its computational efficiency [51].

4.3. Lighting-based Biases on Object Detection

This paper examines how lighting-related biases affect the work of the YOLOv8 object detection model. Carefully varying the brightness levels of the objects and transferring them into randomly generated background images allowed us to get a measure of the effect of brightness on detection accuracy with our controlled imagery generating method. To investigate the impact of various light conditions, the entire data set was created from 192,000 images across 40 different brightness levels. There was a significant effect of brightness on accuracy ($p < 0.0001$), and as brightness increased, the more accurate the model. These findings suggest the presence of non-linear lighting effects and the need to consider lighting consequences to increase the quality of the object-detecting AI-based systems.

Furthermore, the paper shows that brightness is also an important image property in terms of detection performance, and hence, brightness normalization may be a useful detection technique to improve the robustness of the detection models under various lighting conditions. This work can make systematic predictions on the model at varying light intensities to give insight into how light affects the accuracy of detection, and propose practical solutions to handle these lighting biases. In general, this study can help achieve the larger goal of producing more robust and trustworthy AI systems that can consistently perform well in highly challenging and real-world situations. Future research can also expand the analysis to other environmental conditions, including glare, contrast, and color differences, and justify the suggested approaches in real-life uncontrolled environments [52].

4.4. Object Relativity Bias for Detected Objects

Any Computer Vision task requires Object detection. In order to detect objects efficiently and effectively, there are a number of object detection algorithms. Objects in the real world that are smaller may combine to make larger objects, and those that are large may make certain compositions of larger objects. The term "Object Relativity Bias" is used to emphasize the importance of the positional relationships between smaller and larger objects during detection. For example, an isolated wheel can generally be identified as a wheel based solely on its visual characteristics. However, when two wheels are observed horizontally connected by an iron frame, accompanied by a seat, pedals, and a handlebar, the surrounding spatial relationships provide additional contextual information that supports the identification of the wheels as components of a bicycle. Such relationships among objects can facilitate object detection and improve detection confidence. Therefore, this correlation is referred to as "Object Relativity Bias."

Object Relativity Bias (ORB) is a method that attempts to increase the level of confidence of the detected objects by taking into consideration their spatial relations in the image. The main concept is that objects are more likely to be observed in the same location in reference to each other. As an example, when objects A and B are detected with a confidence score of x, and the co-occurrence of the two usually forms or implies the presence of object C, then the probability of A and B being correctly identified should be high when this relationship is observed. Based on this principle, relational heuristics are incorporated into existing object detection systems. These heuristics consider both the positional structure and compositional structure between the detected objects, hence improving the overall prediction confidence. This approach can also be used to evaluate the performance of object detection models. Firstly, objects are recognized in a complete image. After that, all identified regions are cropped with their bounding box, and each of them is processed separately by the same detection algorithm. It is anticipated that the overall contextual information in the original image will produce higher confidence scores than isolated detections.

Object Relativity Bias (ORB) is a process intended to maximize the confidence ratings of objects that have been detected by object detection models. Experiments as part of this research were implemented using Google Colab, a cloud computing platform built by Google Research that supports the Python programming language with limited computing resources and is open source. Its implementation is based on machine learning systems, specifically TensorFlow, which can be easily deployed in the Colab environment. To detect an object, the model used is Faster R-CNN with an InceptionResNet V2 backbone, which has been trained on the Google Open Images dataset. Faster R-CNN is a popular object detection system widely used. It includes a pre-trained convolutional neural network to extract features, which allows it to effectively address and handle complex computer vision tasks with high accuracy.

The Faster R-CNN architecture was used to perform the experiments using the InceptionResNet-v2 model for object detection. First, the individual input images were run through the system to obtain detected objects, their bounding boxes, and confidence scores. Only the detections with a confidence score greater than 0.5 were counted to make the analysis process simple. The detected objects were then cropped out of the original image according to the bounding boxes and then independently processed using the same model. The aim was to compare the confidence of detecting objects in isolation and

in their original composition in context. On the contrary, the findings did not indicate a uniform reduction in the confidence of isolated objects. In other instances, objects like bicycle wheels, footwear, and helmets were recognized better when processed separately, and then, in other instances, there was better confidence when such objects were included in a composite image. These observations indicate that the effects of object relativity can be subject to variations depending on the type of objects and the dependence on context [54].

4.5. Bias in Metal Surface Defect Detection

High performance and efficiency are important factors that have greatly enhanced the accuracy of detecting defects in metal surfaces using deep learning models, specifically YOLO-based models. Still, the problems of generalizability, fairness, and bias are not well studied and, in many cases, people overestimate their predictions. In industries, class imbalance issues and the existence of single-class and multi-class defects also make model training more complicated and may add bias to the detection results.

The complexity of datasets and strategies of evaluation are explored in recent research that addresses these concerns while being fairness-conscious. In particular, the co-occurrence impact analysis reveals the need to create better datasets and training schemes due to multiple defects occurring in a single image that affect detection performance. New measures of fairness, such as Disparate Impact Ratio (DIR) and Predictive Parity Difference (PPD), have been adapted to be useful in illustrating the disparity between defect classes in industrial quality control applications.

Additionally, the wrong representation of certain defects can introduce a non-uniform bias in the models to favor certain classes and cause false positives. This problem highlights the importance of balanced datasets, data augmentation, and resampling methods that guarantee a fair representation of all the classes of defects. Moreover, by measuring the performance of a class in terms of precision, recall, and F1-score, bias in the performance of the various defects can be evaluated.

Overall, these approaches highlight the need for using training and assessment systems that are more bias-aware to boost the robustness, fairness, and generalizability of deep learning systems in real-world deployment in industry for defect detection [55].

5. Methodology

Figure 7 shows the methodology for developing and evaluating YOLO-based object detection models. It starts with the use of the RUOD dataset, followed by the pre-processing step where the images are resized to 512 × 512 pixels and labeled in YOLO format for uniformity and compatibility. Then, different YOLO models (YOLOv5s, YOLOv8s, and YOLOv11s) are trained to be compared. Their training is done using the same hyperparameters (batch size of 32, epochs of 100, learning rate of 0.01, and MuSGD optimizer) to ensure that they are fairly compared. The trained models are then evaluated by using conventional object detection metrics, such as mean Average Precision (mAP), precision, recall, and F1-score. Lastly, bias analysis is performed to evaluate the model's fairness and robustness to different data distributions.

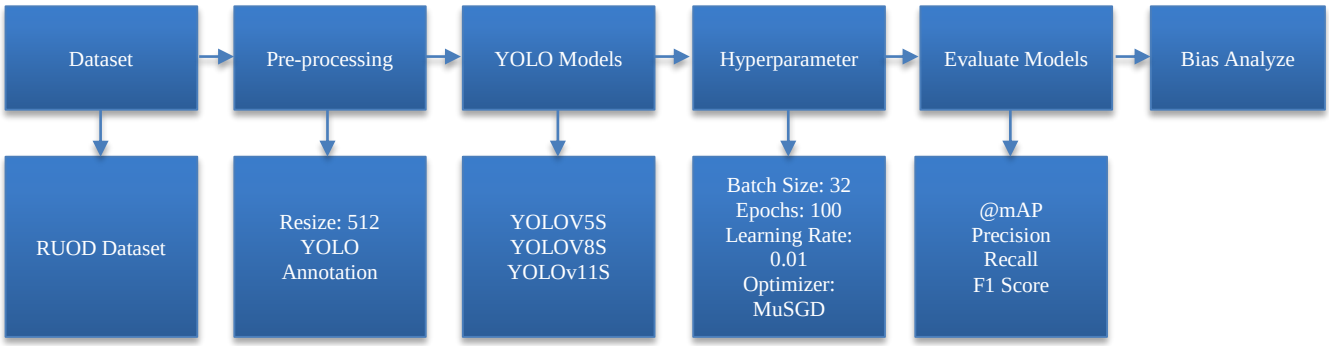


Fig. 7 Methodology of YOLO based object detection pipeline with bias analysis

Algorithm

Input: RUOD dataset D
Output: Trained models M and evaluation results R
1: Load dataset D
2: Pre-process dataset:
3: for each image I in D do
4: Resize I to 512 × 512
5: Convert annotations to YOLO format
6: end for
7: Define model set:
8: Models ← {YOLOv5s, YOLOv8s, YOLOv11s}
9: Initialize hyperparameters:
10: BatchSize ← 32
11: Epochs ← 100
12: LearningRate ← 0.01
13: Optimizer ← MuSGD
14: for each model m in Models do
15: Initialize model m with defined hyperparameters
16: Train model m on pre-processed dataset D
17: Evaluate model m using:
18: Precision
19: Recall

20: F1-score
21: mAP
22: Store evaluation results in R
23: end for
24: Perform bias analysis on results R
25: Return trained models M and evaluation results R

5.1. Dataset Description

Therefore, the RUOD dataset has been used for the tests. This dataset consists of 14,000 underwater images, annotated with information about 10 different object classes. These are Fish, Echinus, Corals, Starfish, Holothurian, Scallop, Diver, Cuttlefish, Turtle, and Jellyfish. These images capture diverse underwater environments, such as low visibility, color distortion, and cluttered backgrounds. The provided split was used with these images: 9,800 were used for training, and 4,200 for testing. The annotations were translated to the YOLO format for training purposes, and the first presentation was in the form of bounding boxes [55].

5.2. Data Pre-Processing

To get a consistent representation across the training process, all images were resized to a fixed size of 512 × 512. Data augmentation was utilized to improve robustness to underwater appearance and object configuration. The augmentation techniques were mosaic augmentation, horizontal flipping, scaling, translation, HSV-based colour transformations, and random erasing. The details of the augmentation parameters are given in Section 5.4.

5.3. Model Implementation

This paper aims to compare and contrast the three light versions of YOLO, namely YOLOv5s, YOLOv8s, and YOLOv11s, using the same experimental setting to perform underwater object detection on the Real-world Underwater Object Detection (RUOD). The three models were loaded with pre-trained weights and trained using the Ultralytics framework. All three architectures have been trained with the same data partition, data resolution, batch size, training time, optimization configuration, data augmentation process, and evaluation process, so that they can be compared objectively.

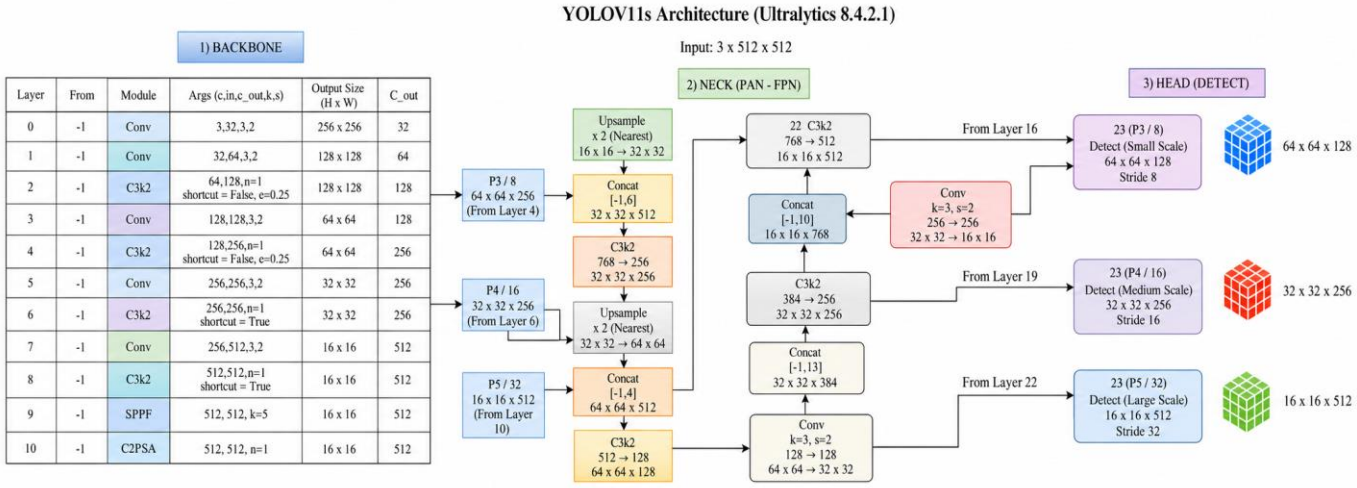
YOLOv5s is a tested and proven lightweight YOLO architecture that has been proven to deliver a balance of accuracy and inference speed. YOLOv8s introduces architectural improvements to increase detection accuracy, while YOLOv11s is a newer, lightweight model which has also been optimized in terms of architecture to enhance feature extraction and detection. Three architectures were then chosen to test whether the progression of architecture leads to an improvement in underwater object detection that can be measured within the computational efficiency.

The number of parameters and inference time of trained models are summarized in Table 1. YOLOv8s has 11.14 million parameters and 28.67 GFLOPs compared to YOLOv5s having 9.13 million parameters and 24.06 GFLOPs. The total number of parameters and GFLOPs of YOLOv11s is 9.43M and 21.57 GFLOPs, respectively. Therefore, YOLOv11s has the lowest number of operations out of the tested models, but the number of parameters is still comparable to YOLOv5s.

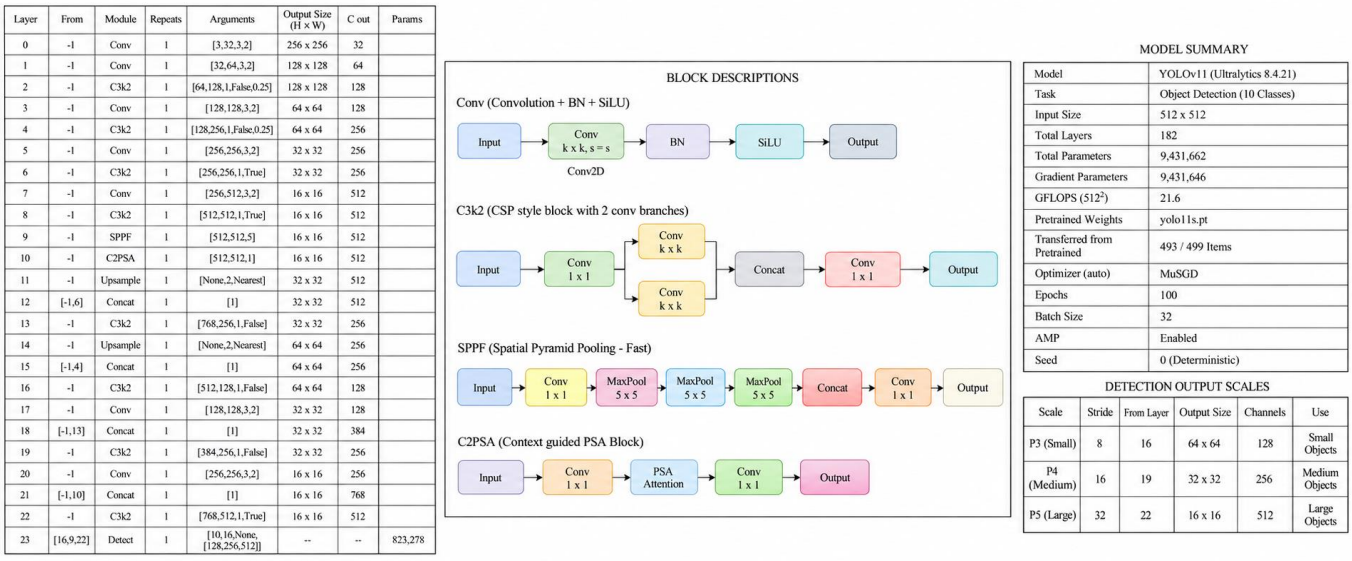
Table 1. Computational characteristics of the evaluated YOLO models

Model	Layers	Parameters	GFLOPs
YOLOv5s	154	9.13 M	24.06
YOLOv8s	130	11.14 M	28.67
YOLOv11s	182	9.43 M	21.57

5.3.1. Detailed YOLOv11s Architecture and Ablation Study



(a)



Layer	From	Module	Repeats	Arguments	Output Size (H x W)	C out	Params
0	-1	Conv	1	[3,32,3,2]	256 x 256	32	
1	-1	Conv	1	[32,64,3,2]	128 x 128	64	
2	-1	C3k2	1	[64,128,1,False,0.25]	128 x 128	128	
3	-1	Conv	1	[128,128,3,2]	64 x 64	128	
4	-1	C3k2	1	[128,256,1,False,0.25]	64 x 64	256	
5	-1	Conv	1	[256,256,3,2]	32 x 32	256	
6	-1	C3k2	1	[256,256,1,True]	32 x 32	256	
7	-1	Conv	1	[256,512,3,2]	16 x 16	512	
8	-1	C3k2	1	[512,512,1,True]	16 x 16	512	
9	-1	SPPF	1	[512,512,5]	16 x 16	512	
10	-1	C2PSA	1	[512,512,1]	16 x 16	512	
11	-1	Upsample	1	[None,2,Nearest]	32 x 32	512	
12	[-1,6]	Concat	1	[1]	32 x 32	512	
13	-1	C3k2	1	[768,256,1,False]	32 x 32	256	
14	-1	Upsample	1	[None,2,Nearest]	64 x 64	256	
15	[-1,4]	Concat	1	[1]	64 x 64	256	
16	-1	C3k2	1	[512,128,1,False]	64 x 64	128	
17	-1	Conv	1	[128,128,3,2]	32 x 32	128	
18	[-1,13]	Concat	1	[1]	32 x 32	384	
19	-1	C3k2	1	[384,256,1,False]	32 x 32	256	
20	-1	Conv	1	[256,256,3,2]	16 x 16	256	
21	[-1,10]	Concat	1	[1]	16 x 16	768	
22	-1	C3k2	1	[768,512,1,True]	16 x 16	512	
23	[16,9,22]	Detect	1	[10,16,None, [128,256,512]]	--	--	823,278

MODEL SUMMARY

Model	YOLOv11 (Ultralytics 8.4.21)
Task	Object Detection (10 Classes)
Input Size	512 x 512
Total Layers	182
Total Parameters	9,431,662
Gradient Parameters	9,431,646
GFLOPS (512^2)	21.6
Pretrained Weights	yolo11s.pt
Transferred from Pretrained	493 / 499 Items
Optimizer (auto)	MuSGD
Epochs	100
Batch Size	32
AMP	Enabled
Seed	0 (Deterministic)

DETECTION OUTPUT SCALES

Scale	Stride	From Layer	Output Size	Channels	Use
P3 (Small)	8	16	64 x 64	128	Small Objects
P4 (Medium)	16	19	32 x 32	256	Medium Objects
P5 (Large)	32	22	16 x 16	512	Large Objects

(b)

Fig. 8 Detailed architecture of the standard YOLOv11s

The experiments employed the standard YOLOv11s (YOLOv11s) object-detection architecture without any architectural modification. The model was initialized with pre-trained YOLOv11s.pt weights and fine-tuned for the desired object-detection problem, which consisted of 10 object classes. This implementation was done using Ultralytics version 8.4.21. The instantiated model was composed of 182 layers, 9,431,662 parameters, and 21.6 GFLOPs for an input resolution of 512 × 512 pixels. The overall layer architecture and number of parameters are shown in Figure 8 and Table 2. It is composed of a convolutional/C3k2-based backbone network, a deep-feature extraction layer based on SPPF and C2PSA, a feature-fusion neck like PAN-FPN, and a three-scale detection head. In the backbone, the spatial resolution decreases with stride-2 convolutions, and the number of channels grows from 32 to 64, 128, 256, and 512. The deepest representation of features is sequentially processed by SPPF and C2PSA modules, each of which has 512 channels.

The neck realizes multiscale feature fusion in both directions. In particular, the high-level feature representation is upsampled by nearest-neighbor interpolation and appended to the intermediate backbone features. The fused representations are then passed to the C3k2 blocks for processing. Then, a bottom-up path performs further stride-2 convolution and concatenation of features to reconstruct increasingly deeper representations.

The detection head gets three feature maps from layers 16, 19, and 22, the outputs from small-, medium-, and large-scale detection layers, respectively. For the 512 × 512 input resolution, these outputs have spatial resolutions of 64 × 64, 32 × 32, and 16 × 16, respectively, with 128, 256, and 512 feature channels. The detection head is set up to recognize 10 classes of objects.

The entire specification (source layer, module type, module arguments, output dimensions, and number of parameters) is given in Table 3. This detailed specification is provided to ensure reproducibility and also to differentiate the standard YOLOv11s architecture adopted in this study from any modified YOLO-based architecture.

Table 2. Layer-by-layer specification of the YOLOv11s architecture used in this study

Layer	From	Module	Repeats	Arguments	Parameters
0	-1	Conv	1	[3, 32, 3, 2]	928
1	-1	Conv	1	[32, 64, 3, 2]	18,560
2	-1	C3k2	1	[64, 128, 1, False, 0.25]	26,080
3	-1	Conv	1	[128, 128, 3, 2]	147,712
4	-1	C3k2	1	[128, 256, 1, False, 0.25]	103,360
5	-1	Conv	1	[256, 256, 3, 2]	590,336
6	-1	C3k2	1	[256, 256, 1, True]	346,112
7	-1	Conv	1	[256, 512, 3, 2]	1,180,672
8	-1	C3k2	1	[512, 512, 1, True]	1,380,352

9	-1	SPPF	1	[512, 512, 5]	656,896
10	-1	C2PSA	1	[512, 512, 1]	990,976
11	-1	Upsample	1	[None, 2, nearest]	0
12	[-1, 6]	Concat	1	[1]	0
13	-1	C3k2	1	[768, 256, 1, False]	443,776
14	-1	Upsample	1	[None, 2, nearest]	0
15	[-1, 4]	Concat	1	[1]	0
16	-1	C3k2	1	[512, 128, 1, False]	127,680
17	-1	Conv	1	[128, 128, 3, 2]	147,712
18	[-1, 13]	Concat	1	[1]	0
19	-1	C3k2	1	[384, 256, 1, False]	345,472
20	-1	Conv	1	[256, 256, 3, 2]	590,336
21	[-1, 10]	Concat	1	[1]	0
22	-1	C3k2	1	[768, 512, 1, True]	1,511,424
23	[16, 19, 22]	Detect	1	[10, 16, None, [128, 256, 512]]	823,278
Total	-	-	-	-	9,431,662

The architecture was instantiated at 512 × 512 input resolution and contained 182 computational layers, 9,431,662 total parameters, and 21.6 GFLOPs.

Table 3. Computational complexity of the YOLOv11s baseline

Property	Value
Model	YOLOv11s
Implementation	Ultralytics 8.4.21
Input resolution	512 × 512
Number of classes	10
Total layers	182
Total parameters	9,431,662
Gradient-bearing parameters during training	9,431,646
Computational complexity	21.6 GFLOPs
Pre-trained initialization	YOLOv11s.pt

Baseline Configuration Analysis

No further components were introduced in the network in this work, for which a standard component-wise ablation could be carried out due to the standard, unmodified YOLOv11s architecture. Thus, the analysis emphasizes the ability to establish the baseline configuration and to make it reproducible, rather than assigning changes in performance to each architectural change.

The evaluated YOLOv11s model has 21.6 GFLOPs when processing images of 512 × 512 pixels and has 9,431,662 parameters. The full layer setup, detection scales, and training parameters are described in the previous sections. This provides a clear baseline reference configuration for future architectural or training changes so that they can be quantitatively analyzed.

Baseline Detection Performance

The YOLOv11s model had a precision of 87.1%, a recall of 80.5%, an mAP@0.5 of 87.0%, and an mAP@0.5:0.95 of 64.8% on the validation set of 4,200 images and 22,969 annotated instances. The inference time per image for this model was 0.4ms on the NVIDIA A100-SXM4 40GB GPU, not including any pre-processing or post-processing time reported by the Ultralytics model validation procedure.

5.4. Training Configuration

The three models were trained with the same experimental setup in the Google Colab Pro environment on an Nvidia A100 GPU using Ultralytics. The models were trained for 100 epochs, with a batch size of 32 and an image size of 512 × 512 pixels. The models were used in the optimization with Ultralytics automatic optimizer selection (optimizer=auto). The learning rate was initially set to 0.01, the learning-rate factor was 0.01, the momentum was 0.937, and the weight decay was 0.0005. The warm-up momentum was set to 0.8, and the warm-up bias learning rate was set to 0.1. A 3-epoch warm-up procedure was used. The configuration of cos_lr=True learning rate was implemented. Each of the training images was augmented with mosaic augmentation, and they were stripped after the last 10 epochs of training. The following augmentations are included: horizontal flip (0.5), translation (0.1), scaling (0.5), HSV hue (0.015), HSV saturation (0.7), HSV value (0.4), and random

erasing (0.4). The augmentation strategy is set to be automatic and named "RandAugment".

The following perspective transformations have been turned off: vertical flipping, rotation, and shearing. In addition, the augmentation methods of MixUp, CutMix, and Copy-paste were also disabled. These settings were maintained throughout the various architectures to reduce implementation differences between the architectures.

5.5. Evaluation Metrics

Common object detection measures were used to measure and compare the performance of the models, including recall, precision, Intersection over Union (IoU), and mean Average Precision (mAP). Recall measures the ability of the detections to be completed, and precision measures how accurate the detections were, as shown in equations 1 and 2:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

True Positive (TP), False Positive (FP), and False Negative (FN) are the terms used, on the other hand.

Equation 3 shows that the overlap between the predicted and actual bounding boxes was measured using Intersection over Union (IoU), which was used to evaluate the localization accuracy:

$$\text{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (3)$$

The overall detection performance was analyzed using mean Average Precision (mAP), which is computed as the mean of Average Precision (AP) across all classes, as shown in equation 4:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (4)$$

Where N is the number of object classes. The Average Precision for each class is calculated as shown in equation 5, which is the area under the precision–recall curve.

$$AP = \int_0^1 p\,(r)\,dr \quad (5)$$

In this study, both mAP@0.5 and mAP@0.5:0.95 were considered, with the latter serving as the primary evaluation metric due to its stricter assessment across multiple IoU thresholds.

5.6. Bias Measurement and Validation

5.6.1. Dataset Class Representation Bias

In addition to conventional detection-performance metrics, the class representation of the RUOD dataset was quantified to characterize potential dataset-level imbalance. For each object class, a normalized class representation bias ratio (BR_C) was calculated as shown in Equation 6

$$BR_C = \frac{N_C}{N_{max}}, \quad (6)$$

Where N_c represents the number of annotated instances belonging to class c, and N_{max} represents the number of instances in the most represented class. The resulting value satisfies $0<BR_c \leq 1$. A value of $BR_c = 1$ indicates the highest-class representation, whereas smaller values indicate increasing underrepresentation relative to the most represented class.

Therefore, BR_c is used in this study as a class representation imbalance measure, rather than as a direct measure of detector accuracy or model fairness.

To provide an established reference for the observed class imbalance, the Imbalance Ratio (IR), Gini coefficient (G), and Coefficient of Variation (CV) were additionally calculated. The conventional Imbalance Ratio (IR) was also considered, as shown in Equation 7:

$$IR = \frac{N_{max}}{N_{Min}}, \quad (7)$$

Where N_{min} denotes the number of instances in the least represented class. In addition, the Gini coefficient (G) was used to quantify inequality across the complete class-frequency distribution as shown in equation 8:

$$G = \frac{\sum_{i=1}^{C}\sum_{j=1}^{C}|N_i - N_j|}{2C\sum_{1=1}^{C} N_{i\cdot}}, \quad (8)$$

Where C is the total number of object classes, and N_i and N_j denote the frequencies of classes i and j, respectively. The Coefficient of Variation (CV) was additionally calculated to quantify the relative dispersion of class frequencies as shown in equation 9:

$$CV = \frac{\sigma_N}{\bar{N}}, \quad (9)$$

Where σN and $\bar{N}$ represent the standard deviation and mean of the class frequencies, respectively. These measures provide complementary descriptions of class imbalance: BR_c describes representation at the individual-class level, whereas IR, G, and CV provide global descriptions of the distribution.

5.6.2. Detection Performance Bias Metric

To quantify class-wise disparity in detector performance, a Detection Performance Bias Index (DPBI) was defined using the class-wise AP@0.5 values obtained from each trained detector. The metrics used to quantify detection performance

and class representation bias are summarized in Table 4. The DPBI is defined as shown in Equation 10:

$$DPBI = \frac{1-AP_{min}}{AP_{max}} \quad (10)$$

Where AP_{min} and AP_{max} denote the minimum and maximum class-wise AP@0.5 values, respectively. The index ranges from 0 to 1, where a value approaching zero indicates more uniform detection performance across object classes, while a larger value indicates greater disparity between the best- and worst-performing classes. The metric, therefore, characterizes class-wise detection-performance disparity rather than dataset representation imbalance.

To validate and contextualize the DPBI, two established measures of distributional dispersion, namely the Gini coefficient and the coefficient of variation, were additionally calculated over the ten class-wise AP@0.5 values for each detector. These measures provide independent descriptions of the spread of class-wise detection performance and allow the proposed DPBI to be interpreted alongside established statistical measures.

Table 4. Class-wise detection-performance disparity measured using AP@0.5

Model	Mean AP@0.5	Minimum AP@0.5	Maximum AP@0.5	DPBI	Gini	CV
YOLOv5s	0.8654	0.745	0.974	0.2351	0.0529	0.0933
YOLOv8s	0.8681	0.749	0.977	0.2334	0.0521	0.0922
YOLOv11s	0.8701	0.749	0.977	0.2334	0.0518	0.0917

The three detectors exhibit relatively similar levels of class-wise detection disparity. YOLOv5s obtained a DPBI of 0.2351, while YOLOv8s and YOLOv11s obtained values of 0.2334. The corresponding Gini coefficients were 0.0529, 0.0521, and 0.0518, and the coefficients of variation were 0.0933, 0.0922, and 0.0917, respectively. The consistent ordering across these measures indicates that the proposed DPBI provides a stable summary of the disparity between the highest- and lowest-performing classes, while the Gini coefficient and coefficient of variation characterize the dispersion across all classes.

5.7. Model Selection Strategy

The training epoch was not equal to the last epoch, but instead a new checkpoint was chosen for each architecture based on the maximum validation mAP@0.5:0.95. The motivation for the strategy is that it reduces the risk of taking the model that was trained last by choosing the checkpoint that best performed on the validation set over all the datasets.

5.8. Experimental Reproducibility

The same general experimental setup was applied to YOLOv5s, YOLOv8s, and YOLOv11s in terms of dataset split, input resolution, batch size, training time, optimization configuration, data augmentation, random seed, and validation process for reproducibility reasons. Execution was set to deterministic, and the seed value was set to be zero. All experiments were conducted using an NVIDIA A100 GPU, Python 3.12.12, PyTorch 2.10.0, and CUDA 12.8. The environment(s) recorded were Ultralytics version 8.4.22 of YOLOv5s and 8.4.21 of YOLOv8s and YOLOv11s. These minor version differences in the framework are explicitly communicated for ease in reproducing the environment of the experiment. The architectures have been trained just once using the following architecture configuration. Consequently, the performance measures given are for one training run of the model and are not an average of independent runs. As such, standard deviations, confidence intervals, and formal statistical significance testing were not computed. Next, the comparison is interpreted descriptively based on the differences found in the quantitative and convergence results of the evaluation, as well as the differences in the quantitative and qualitative results of the class-wise results of the evaluation.

6. Experiments & Results

6.1. Quantitative Performance Comparison

The results for the evaluated models are presented in Table, which lists object detection metrics such as mean Average Precision (mAP), precision, and recall.

Table 5 demonstrates that all three architectures achieve good detection performance with relatively small performance variations across metrics evaluated. The overall detection and localization performance is the best for YOLOv11s, with mAP@0.5 score of 0.8721 and mAP@0.5:0.95 score of 0.6483, respectively. It is especially worth noting that the improvement in mAP@0.5:0.95 is especially relevant, since this metric evaluates localization at multiple IoU thresholds, making it a more stringent method to analyze the quality of the bounding box.

Table 5. Quantitative comparison of YOLOv5s, YOLOv8s, and YOLOv11s

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall
YOLOv5s	0.8663	0.63599	0.86873	0.80192
YOLOv8s	0.8663	0.6431	0.86883	0.80485
YOLOv11s	0.8721	0.6483	0.86604	0.80604

The mAP@0.5 result of YOLOv5s is 0.8663, while that of YOLOv8s is 0.8663. However, YOLOv8 outperforms YOLOv5 in terms of mAP@0.5:0.95, with respective scores of 0.6431 and 0.63599, which means that it has better localization performance when the IoU requirement is stricter.

However, YOLOv8s also has the highest precision (0.86883), while YOLOv11s has the highest recall (0.80604), meaning that YOLOv11s detects a slightly larger proportion of ground-truth objects.

The results obtained were then compared with those previously reported in the RUOD benchmark (Table 6) to further establish the contextualization. The proposed YOLOv11s configuration achieves an mAP@0.5 of 0.8721 and an mAP@0.5:0.95 of 0.6483, which is competitive with the recent underwater object-detection approaches. Specifically, the mAP@0.5:0.95 achieved is higher than YOLOv8s (0.639), YOLOv11s (0.638), YOLO11n (0.615), and AGS-YOLO (0.635), while still slightly lower than the YOLO-GE (0.655) under its experimental configuration. This comparison shows that the performance gain found in the current study is not just for internal comparison of YOLO architectures, but is also competitive with the recently reported RUOD-based approaches. The improvement can be linked to the YOLOv11 architecture used in this work, and the consistent training set, input resolution, augmentation strategy, and model selection procedure. The additional advantage of YOLOv11s over the other architectures evaluated is the increased mAP@0.5:0.95, which points to better localization performance at lower IoU levels. These results are based on differences in the training configurations and implementation details from published studies, and are not considered to be a "controlled" head-to-head comparison of the two models.

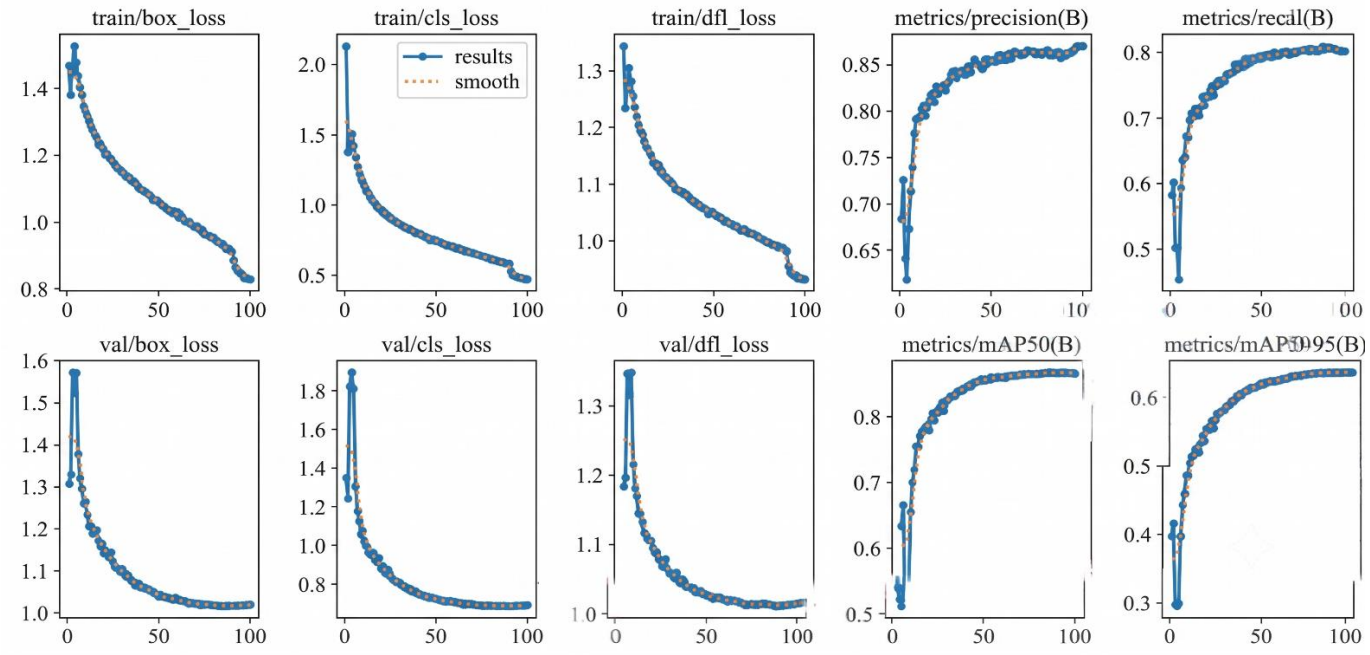
Overall, the quantitative results indicate that YOLOv11s provides the strongest aggregate performance, while YOLOv8s achieves the highest precision. Because each architecture was evaluated using one selected training run, these differences are interpreted as observed performance differences rather than statistically significant improvements. The performance differences among the three evaluated models are not drastic; YOLOv11s consistently demonstrates incremental improvements across multiple evaluation aspects.

Table 6. Comparison with previously reported approaches on the RUOD dataset

Method	mAP@0.5	mAP@0.5:0.95
YOLOv8s [58]	0.85	0.661
YOLOv11s [31]	0.61	0.85
YOLO11n [31]	0.56	0.81
YOLOv5s (this study)	0.8663	0.6360
YOLOv8s (this study)	0.8663	0.6431
YOLOv11s (this study)	0.8721	0.6483

6.2. Training Convergence Analysis

To evaluate the training convergence behavior of the evaluated models, the evolution of loss components and the performance metrics were investigated for the training epochs. In particular, the bounding-box loss, classification loss, Distribution Focal Loss (DFL), precision, recall, and mean Average Precision (mAP) were tracked to monitor the optimization stability and learning dynamics.



(a)

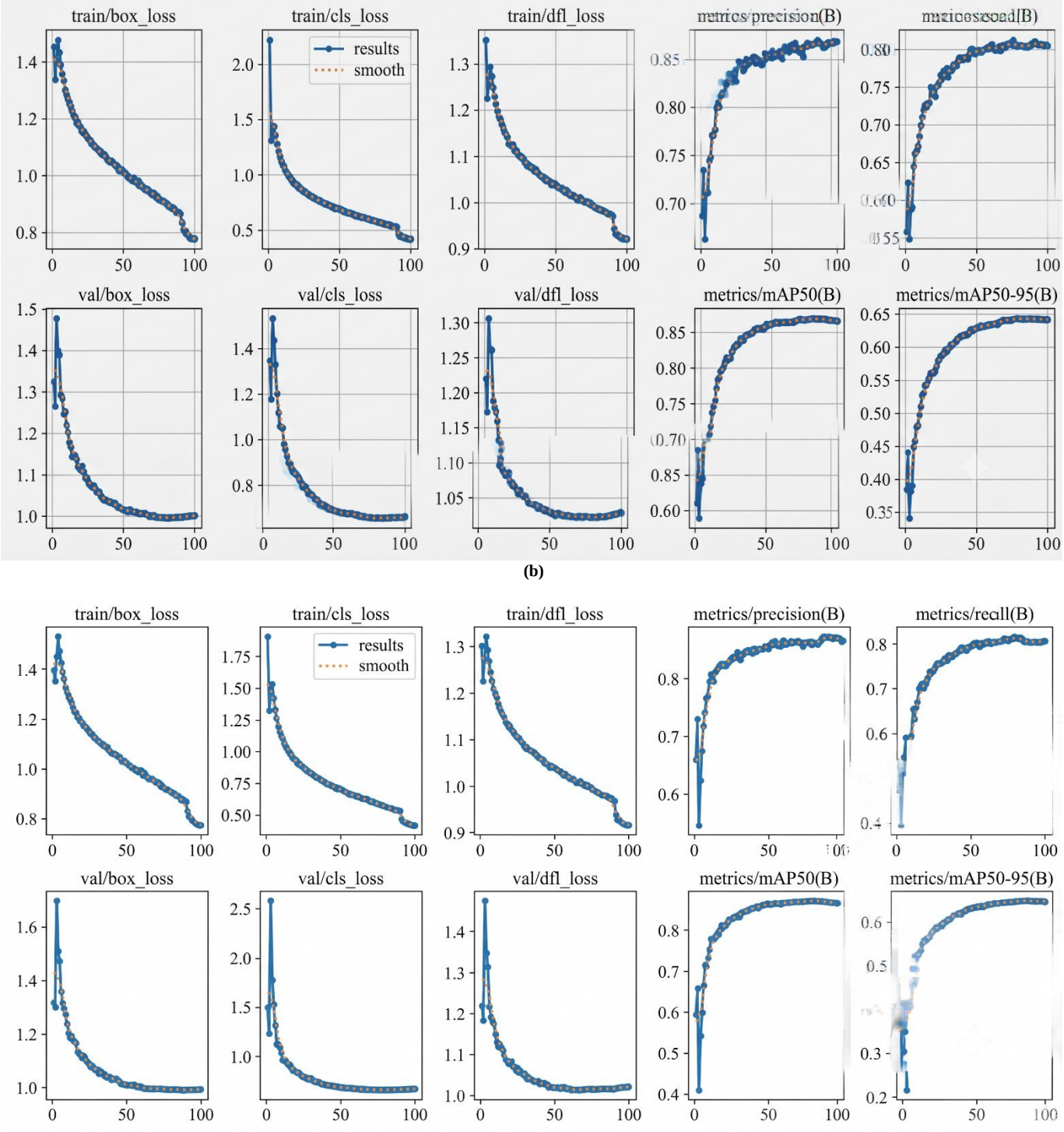


Fig. 9 Training and validation performance curves of (a) YOLOv5s, (b) YOLOv8s, and (c) YOLOv11s, illustrating loss components, precision, recall, and mAP trends across epochs.

Figures 9(a), 9(b), and 9(c) show the overall reduction in the loss values for all three models during the training process, which represents progressive optimization. YOLOv5s gradually reduces the losses and stabilizes towards the end of the training. YOLOv8s has a relatively steady optimization curve, and YOLOv11s achieves a relatively stable curve in terms of loss during the training process.

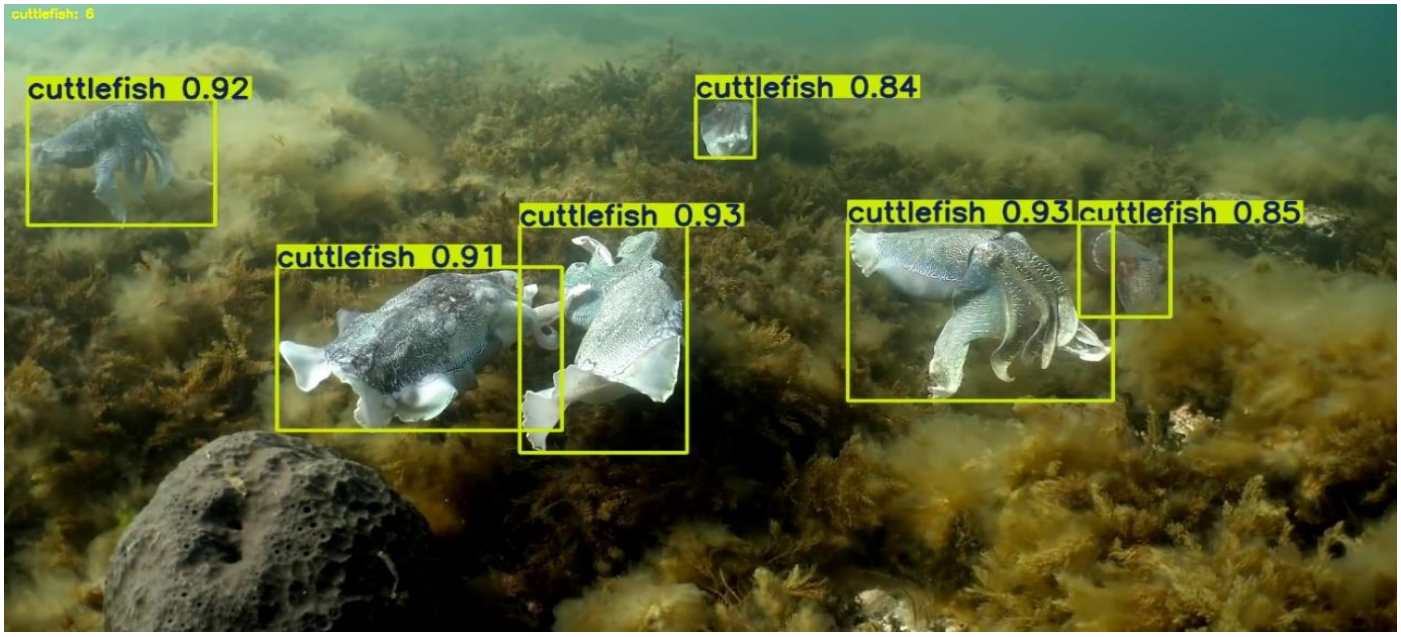
The precision and recall curves also show that the validation performance is improving over the training process. The differences between the models in their behavior of converging towards a solution are very minor, and the trends observed show that all three architectures were optimized properly with the same training set and configuration, as described in Section 5.4.

The convergence analysis thus serves as an additional source of information for the final quantitative comparison, and the validation metrics are still the main source of information for analyzing model performance.

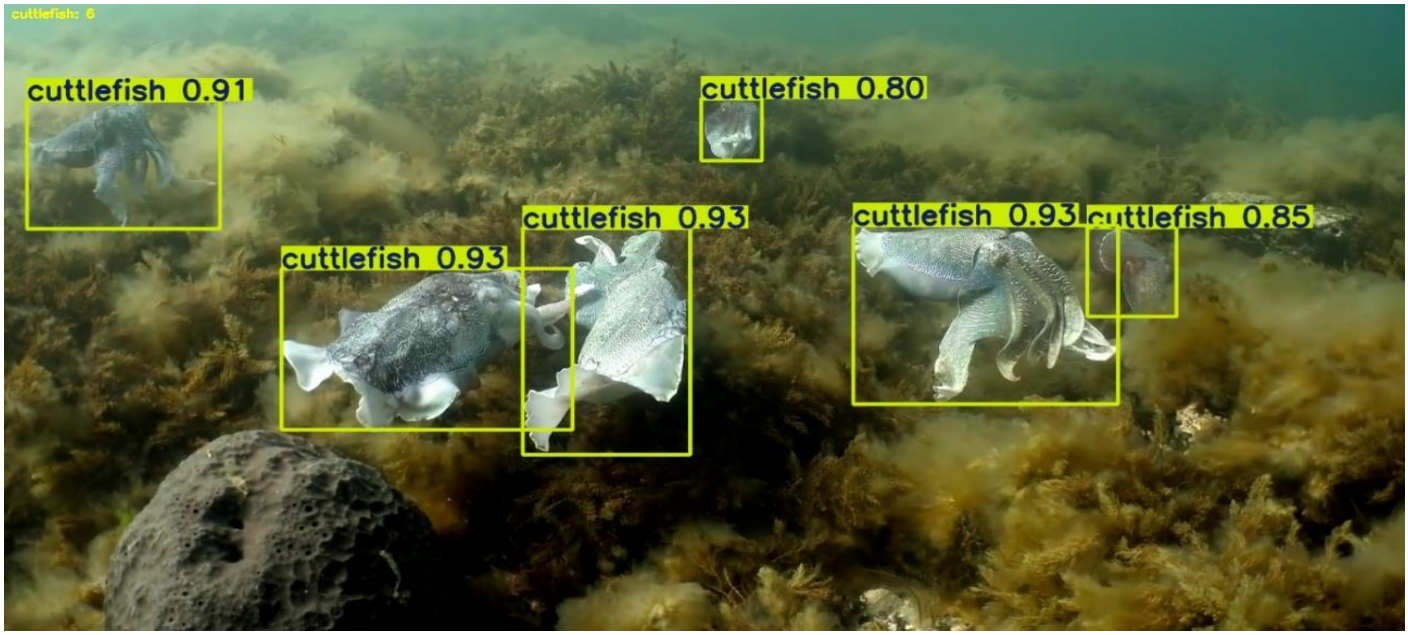
6.3. Detection Performance Analysis

The trained models were qualitatively tested by presenting them with the validation predictions and visualizing the Bounding Boxes and predicted class labels on the previously unseen validation images.

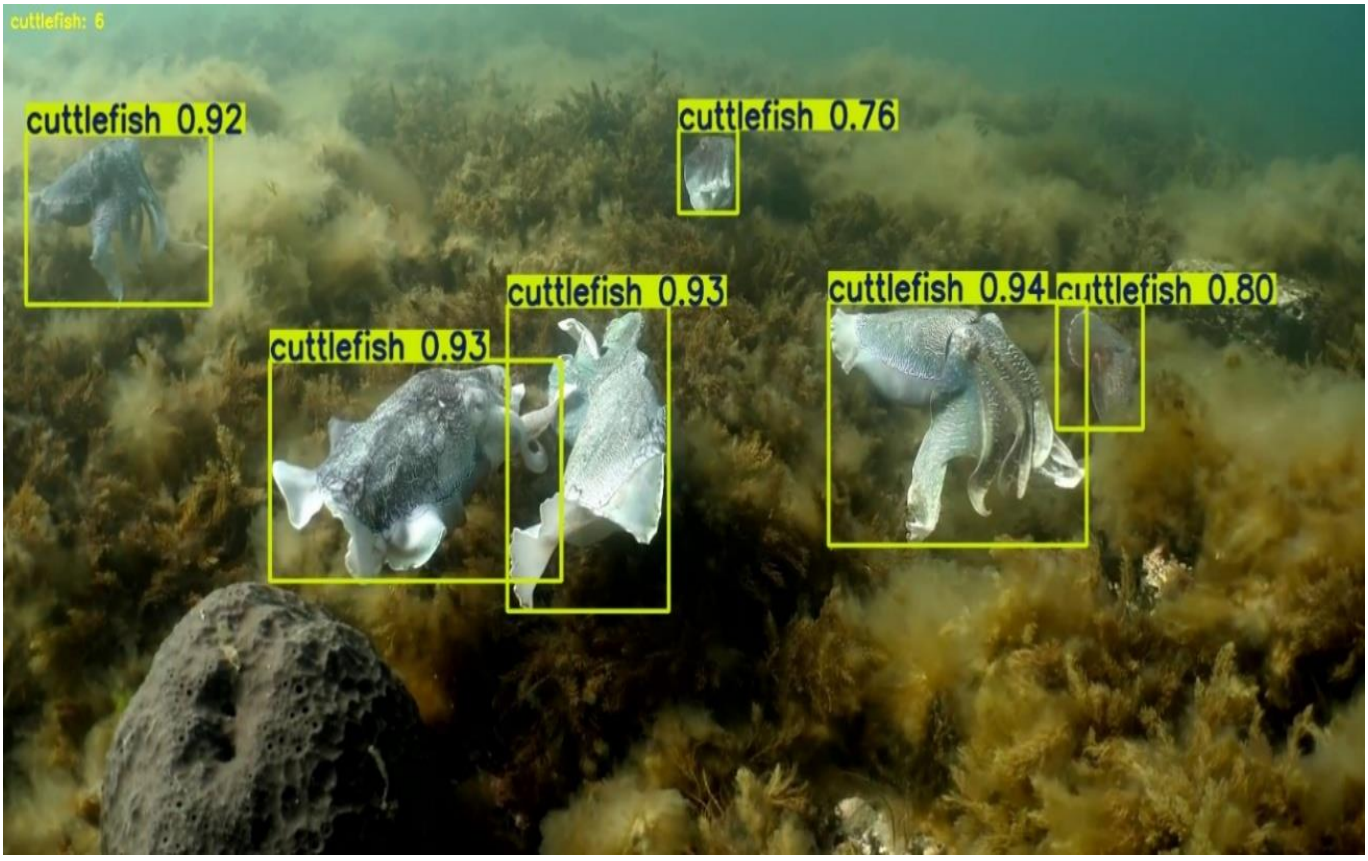
This analysis complements the other evidence presented about object localization and recognition in challenging underwater conditions such as low visibility, occlusion, cluttered backgrounds, and object scale variation.



(a)



(b)



(c)

Fig. 10 Detection outputs comparing, (a) YOLOv5s, (b) YOLOv8s, and (c) YOLOv11s on the same underwater scene, illustrating differences in detection accuracy and localization performance.

Figure 10 shows some detection examples of YOLOv5s, YOLOv8s, and YOLOv11s on similar underwater images. The three models manage to detect multiple object categories, and in the majority of cases, they obtain well-localized bounding boxes. The results show that the architectures evaluated can deal with complex underwater scenes with multiple objects and different visual conditions. YOLOv11s exhibits relatively uniform prediction in dense scenes, while YOLOv8s exhibits more balanced prediction in the cases that are analyzed. YOLOv5s also yields good detections, but with some variations in localization and misses in visually complex areas.

Qualitative observations are provided to supplement the quantitative results presented in Table 1 and are not meant to replace quantitative evaluation.

6.4. Confusion Matrix Analysis

Normalized confusion matrixes were analyzed to gain deeper insight into the detection behavior of the three investigated architectures on a class-wise basis.

The matrixes offer insight into the patterns of prediction in each class and the classes that are misclassified.

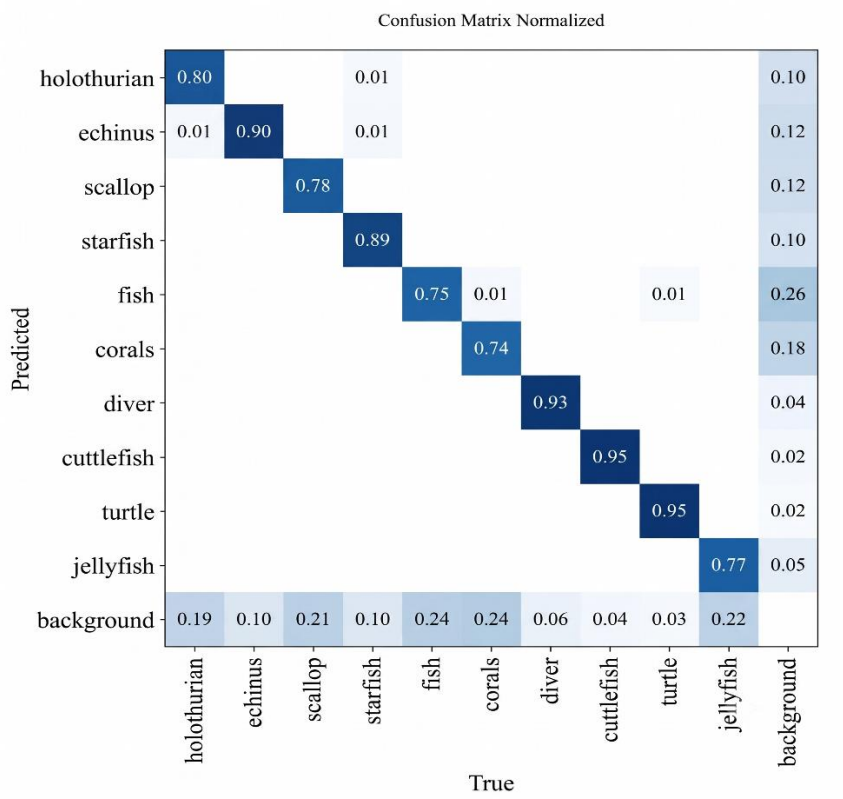
As shown in Figure 11, the three models generally show strong responses along the diagonal, suggesting successful class discrimination.

However, some categories of underwater objects that are similar in appearance are still mixed up. Confusion may be caused by similarities in shape, texture, appearance, and the effects caused by underwater illumination and turbidity.

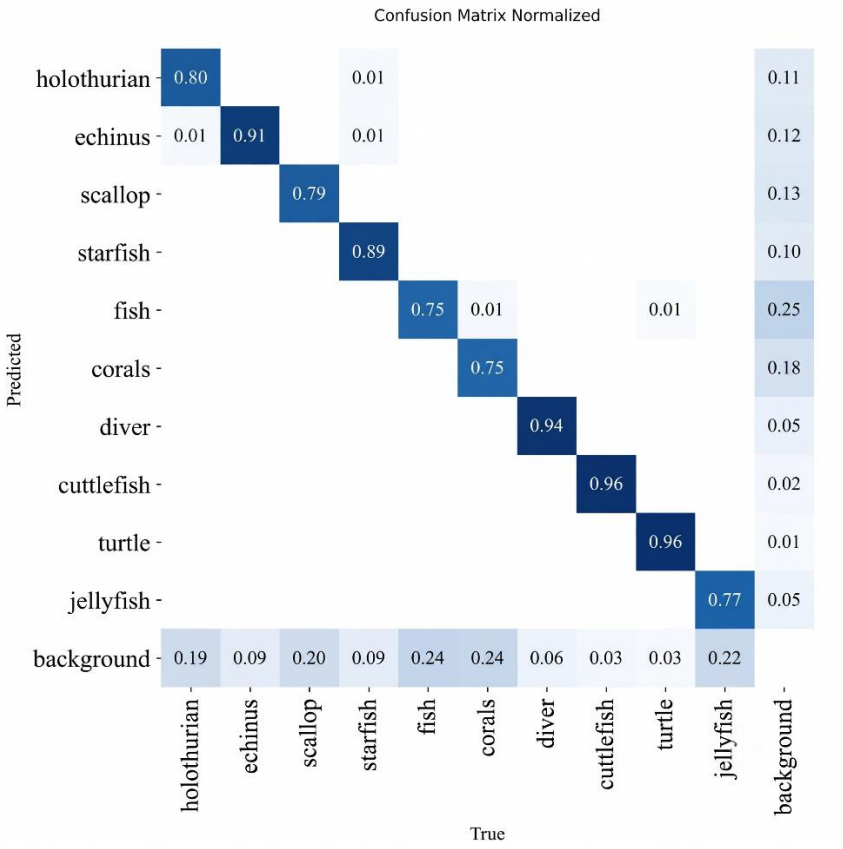
The class-wise predictions for YOLOv11s are comparatively good for the various classes evaluated, and similarly for YOLOv8s and YOLOv5s. The confusion matrices are thus another source of class-level evidence to complement the overall metrics given in Table 1.

Based on the analysis of the confusion matrix, it is generally concluded that all models are reliable for classifying underwater objects, and there are a few challenges that still have to be overcome in the presence of visually overlapping classes.

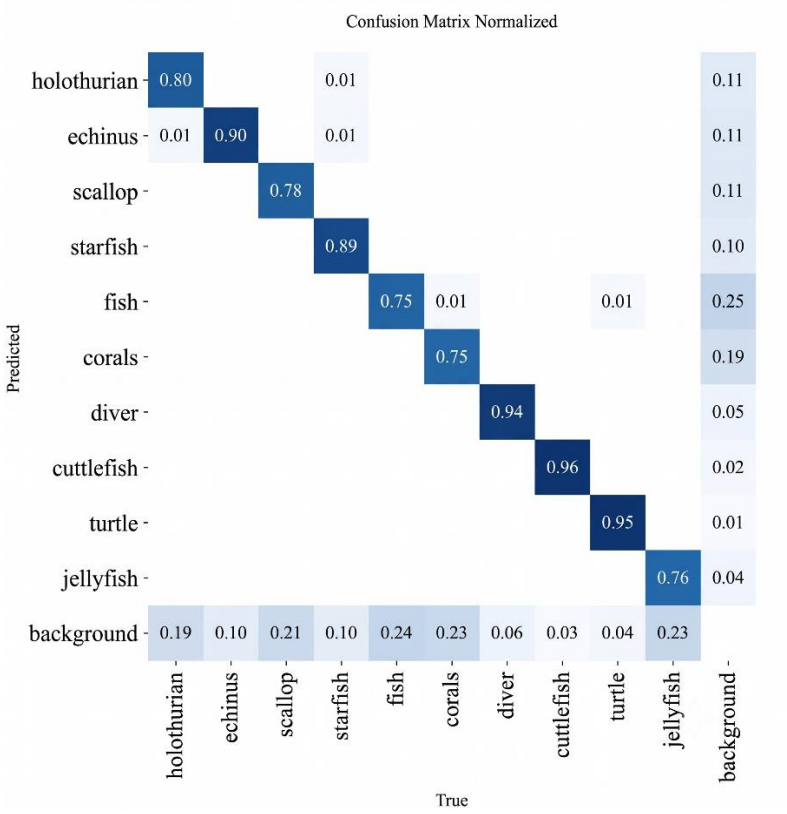
The quantitative measurements and qualitative detection results shown in the preceding sections were consistent with these results.


Confusion Matrix Normalized
holothurian 0.80 0.01 0.10
echinus 0.01 0.90 0.01 0.12
scallop 0.78 0.12
starfish 0.89 0.10
fish 0.75 0.01 0.01 0.26
corals 0.74 0.18
diver 0.93 0.04
cuttlefish 0.95 0.02
turtle 0.95 0.02
jellyfish 0.77 0.05
background 0.19 0.10 0.21 0.10 0.24 0.24 0.06 0.04 0.03 0.22
Predicted
holothurian
echinus
scallop
starfish
fish
corals
diver
cuttlefish
turtle
jellyfish
background
True

(a)


Confusion Matrix Normalized
holothurian 0.80 0.01 0.11
echinus 0.01 0.91 0.01 0.12
scallop 0.79 0.13
starfish 0.89 0.10
fish 0.75 0.01 0.01 0.25
corals 0.75 0.18
diver 0.94 0.05
cuttlefish 0.96 0.02
turtle 0.96 0.01
jellyfish 0.77 0.05
background 0.19 0.09 0.20 0.09 0.24 0.24 0.06 0.03 0.03 0.22
Predicted
holothurian
echinus
scallop
starfish
fish
corals
diver
cuttlefish
turtle
jellyfish
background
True

(b)



(c)

Fig. 11 Confusion matrix illustrating class-wise prediction performance for (a) YOLOv5s, (b) YOLOv8s, and (c) YOLOv11s.

6.5. Performance Analysis of YOLOv5s, YOLOv8s and YOLOv11s

Comparative results indicate that the three lightweight architectures presented in YOLO deliver competitive performance in underwater object detection, and show that each model presents different levels of accuracy, localization, computational complexity, and convergence.

YOLOv11s outperforms additional models in terms of mAP@0.5 and mAP@0.5:0.95, demonstrating the model's highest overall object detection and localization accuracy. YOLOv8s has the best precision, while YOLOv11s has the best recall. The performance variations are not significant, meaning that all three architectures can be used to achieve the goal of the underwater detection task.

Clarifying differences between the models is the computational analysis. YOLOv11s needs only 21.57 GFLOPs, whereas YOLOv5s and YOLOv8s require 24.06 GFLOPs and 28.67 GFLOPs, respectively. Therefore, YOLOv11s not only has the best overall detection accuracy among the three models but also has the least computational cost. This combination is useful for applications where detection accuracy and computational efficiency have to be taken into account together.

The convergence analysis also shows that the models can be trained well with the same training configuration, but have different learning paths. The qualitative detection results and the confusion matrices support each other in verifying their capability of underwater object detection and discrimination in visually challenging situations.

In summary, the experimental results show that YOLOv11s has the best overall performance in the current experimental environment in terms of detection accuracy and computational complexity. The modifications noted, however, must be viewed as "real," or empirical, performance differences, since each model was trained only once.

Table 7. Model complexity & bias analysis

Model	Complexity	Parameters	No of Layer	Training Time	Inference Time
YOLOv5s	24.06	9.13	154	2.015 Hrs	80.5 MS
YOLOv8s	28.67	11.14	130	1.887 Hrs	7.9 MS
YOLOv11s	21.57	9.43	182	2.229 Hrs	107 MS

Table 7 compares YOLOv5s, YOLOv8s, and YOLOv11s in terms of complexity, size, and computational efficiency. YOLOv8s is the most complicated and contains the greatest number of parameters and the shortest training time, as well as significantly lower inference time, which implies greater optimization and applicability in real-time. YOLOv5s is fairly complex and demonstrates stable performance on all measures. Conversely, YOLOv11s is more efficient and has reduced training and inference time even though it has more layers, indicating that more architectural depth may not always be associated with better computational efficiency. YOLOv8s offers the most effective trade-off between complexity and speed overall.

6.6. Dataset Bias Analysis

The class representation of the RUOD dataset was also analysed, and the dependency between the detection performance and the classes was investigated to further examine the robustness and generalization ability of the trained models. The relative representation of each class with respect to the category with the highest number of instances was measured using the normalized Class Representation Bias Ratio (BRC) as in Section 5. This analysis separates the class representation imbalance in the datasets from the detection performance of YOLOv5s, YOLOv8s, and YOLOv11s.

6.6.1. Class Representation Bias

As shown in Figure 12, there is a significant modification in class distribution within the RUOD training dataset. Fish is the most common, with 9,090 occurrences and a normalized bias ratio value of 1.00. There are 7,867 instances of Echinus, compared to 6,132 instances of corals (BR=0.67). By contrast, turtle and jellyfish cases are significantly underrepresented, with 2,824 (BR=0.31) and 1,881 (BR=0.21) cases, respectively. The intermediate class has a ratio of representation between 0.41 and 0.63 in the remaining classes. The results show that the RUOD dataset is not uniformly distributed in terms of the number of instances used for training each individually class, with several classes having significantly fewer instances than the majority.

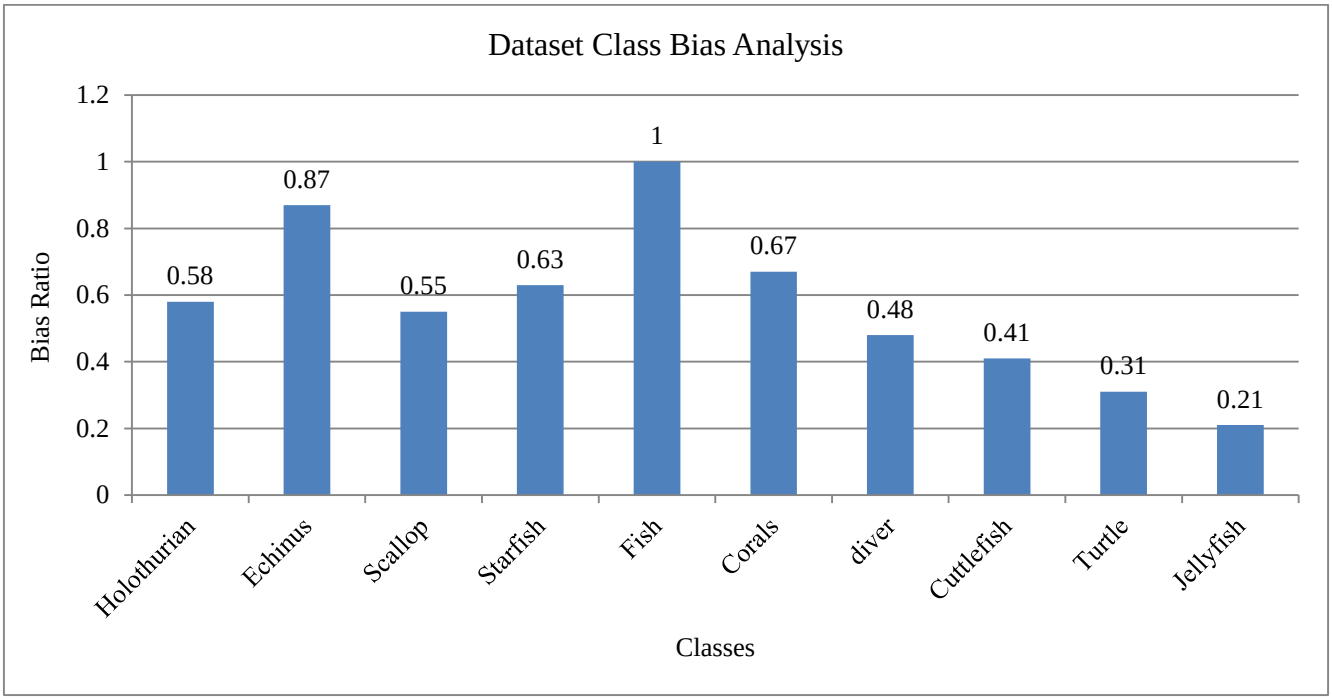


Fig. 12 Class-wise bias ratio distribution of YOLOv5s, YOLOv8s and YOLOv11s

The dataset is highly imbalanced, with the number of instances of fish being approximately 17.50%, echinus approximately 15.15%, jellyfish approximately 3.62%, turtle approximately 5.44%, and cuttlefish approximately 7.16% of all the instances annotated.

This imbalance is reflected in the normalized values in the range 0.21–1.00, with jellyfish at the lowest (0.21) and fish at the highest (1.00) values. More importantly, these ratios are indicative of the distribution of training instances, and not a direct measure of detection bias.

6.6.2. Validation against Established Class-Imbalance Measures

To validate and contextualize the class representation bias identified using BR_C, the class distribution was additionally evaluated using established measures of class imbalance. The imbalance ratio was calculated using the most and least represented classes, as shown in Equation 11.

The RUOD training set contains 9,090 instances for the most represented class (fish) and 1,881 instances for the least represented class (jellyfish), resulting in

$$IR = \frac{9090}{1881} = 4.83 \quad (11)$$

This indicates that the most represented class contains approximately 4.83 times as many instances as the least represented class.

Because the imbalance ratio considers only the two extreme classes, the Gini coefficient was additionally calculated using all ten class frequencies. The resulting Gini coefficient was 0.2244. The coefficient of variation was also calculated and yielded a value of 0.3991. These values independently confirm that the class frequencies are not uniformly distributed across the dataset.

The established measures are consistent with the class-wise normalized bias ratio. The BR_C identifies fish as the most represented class (BR=1.00) and jellyfish as the least represented class (BR=0.21), while the imbalance ratio, Gini coefficient, and coefficient of variation provide complementary global evidence of class-frequency inequality. Therefore, the normalized bias ratio provides an interpretable class-level description of representation imbalance, whereas the established measures quantify the overall degree of inequality in the dataset. The resulting class-representation statistics and the corresponding imbalance measures are summarized in Table 8.

Table 8. Class representation and imbalance measures for the RUOD training dataset

Measure	Value	Interpretation
Total annotated instances	51,931	All ten classes
Maximum class frequency	9,090	Fish
Minimum class frequency	1,881	Jellyfish
Maximum (BR_c)	1.00	Fish
Minimum (BR_c)	0.21	Jellyfish
Imbalance Ratio ((IR))	4.83	Maximum/minimum frequency
Gini coefficient ((G))	0.2244	Overall class-frequency inequality
Coefficient of variation ((CV))	0.3991	Relative frequency dispersion

As shown in Table 8, the RUOD training dataset contains 51,931 annotated instances distributed across ten object classes. The maximum class frequency is 9,090 instances for fish, whereas jellyfish has the minimum frequency of 1,881 instances. Accordingly, the maximum and minimum values are 1.00 and 0.21, respectively. The imbalance ratio of 4.83 indicates that the most represented class contains approximately 4.83 times as many instances as the least represented class. The Gini coefficient of 0.2244 and the coefficient of variation of 0.3991 further indicate measurable inequality and dispersion in the class-frequency distribution. These results are consistent with the class-wise pattern observed in Figure 11 and provide established distribution-level measures for validating the proposed representation measure.

6.6.3. Relationship between Class Representation and Detection Performance

The class-wise average precision at AP@0.5 of the YOLOv5s, YOLOv8s, and YOLOv11s models were compared with the normalized class representation ratio to see if there is a correlation between class representation and detection performance. In contrast to the dataset-level analysis performed in Figure 11, this comparison will determine if the class-frequency imbalance observed in the dataset is also reflected in the detection performance of the three trained detectors, as shown in Figure 13.

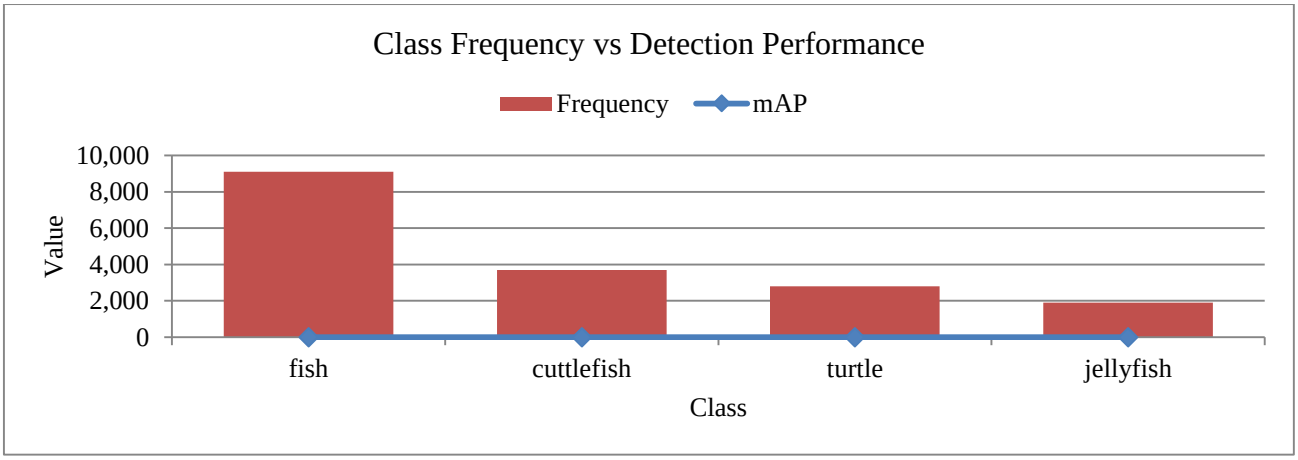


Fig. 13 Relationship between class frequency and detection performance (mAP@0.5) for YOLOv5s, YOLOv8s and YOLOv11s

The comparison shows that there is no direct relationship between class frequency and detection performance. The most represented class, fish (BR=1.00), obtained AP@0.5 values of 0.770, 0.774, and 0.776 for YOLOv5s, YOLOv8s, and YOLOv11s, respectively. In contrast, turtle (BR=0.31) achieved AP@0.5 values of 0.974, 0.977, and 0.977, while cuttlefish (BR=0.41) achieved 0.973, 0.975, and 0.977. Lower class representation, however, did not necessarily correlate with lower detection performance; several underrepresented classes had higher AP@0.5 than the predominant class.

Overall, though, the performance of jellyfish, the least represented detector, was comparatively poor (BR=0.21) in all three detectors. There could be an underrepresentation effect on the detection performance of some classes but not of all classes. Detection performance can also be influenced by a number of class-specific characteristics, such as the visual properties of the objects, their appearance, and the separability between the classes.

A similar class-frequency distribution was found in the validation set, suggesting that the train-validation split does not substantially change the class representation. Therefore, the distribution of the evaluation set is more similar to that of the training set. That is, this consistency is crucial in the interpretation of the detection results since the difference in performance between the classes cannot be blamed on the fact that the distribution of classes across the two sets (training, validation) is not significantly different.

In general, the detection performance increases with the number of classes when they are represented as observed in Figure 12, but this is not a linear relationship. Several underrepresented classes, especially cuttlefish and turtle in YOLOv8s and YOLOv11s, still got a high detection performance, demonstrating that the model can learn discriminative feature representations from a relatively small number of training samples. Jellyfish was the least represented class; however, across the three detectors, there was relatively lower performance in this class, indicating that extreme underrepresentation may have a greater impact on detection performance.

6.6.4. Detection Performance Bias across Classes

The class-wise detection results were also analyzed to see whether the imbalance in class representation reflects the imbalance in detection performance across the three YOLO detectors. The Detection Performance Bias Index (DPBI) is an index that measures the differences in the performance of the detector in each class with respect to class-wise AP@0.5, unlike the BRC index, which characterizes the distribution of training instances.

YOLOv5s achieved a DPBI value of 0.2351, and YOLOv8s and YOLOv11s achieved 0.2334. The difference between the three is rather small, suggesting that all three detectors have similar class-wise performance differences. The Gini coefficient for the class-wise AP@0.5 values for YOLOv5s, YOLOv8s, and YOLOv11s was 0.0529, 0.0521, and 0.0518, respectively, and the corresponding coefficients of variation were 0.0933, 0.0922, and 0.0917, respectively. These results help interpret YOLOv11s as the best model in terms of class-wise detection performance, with a marginal difference compared to the other two models.

Notably, the dataset-level imbalance measures were significantly larger than the detection-performance dispersion measures. The training dataset had an imbalance ratio of 4.83, a Gini coefficient of 0.2244, and a coefficient of variation of 0.3991, while the Gini coefficients and coefficient of variations for the three detectors were computed for each class, with AP@0.5 values near 0.052 and below 0.10, respectively. This means that the high class-frequency imbalance of the RUOD training data was not directly translated to a similar detection-performance imbalance.

The class-wise results also reveal that the number of training instances is not the only factor affecting the detection performance. For instance, fish have significantly more training instances compared to turtles and cuttlefish, and the AP@0.5 values are above 0.97 for these three detectors, respectively. On the other hand, fish is the most represented class with AP@0.5 of 0.770, 0.774, and 0.776 for YOLOv5s, YOLOv8s, and YOLOv11s, respectively. The class with the lowest representation is jellyfish, which also has comparatively low performance. These observations indicate that class representation may be one factor in the occurrence of detection disparity; however, class-specific visual characteristics and detection difficulty are also important.

6.6.5. Discussion

According to the overall analysis, the RUOD dataset is highly imbalanced with respect to the different classes, supported by other imbalance measures and confirmed by the normalized class representation bias ratio. Moreover, when detection performance is compared, the frequency of a class does not consistently influence the performance of the detection of any particular category of objects. The moderately underrepresented classes had high detection performance, while the severely underrepresented jellyfish class had relatively lower detection performance in all three models. These results indicate that class imbalance at the dataset level may be a potential source of detection bias, but this may vary depending on the nature of the individual classes.

In general, it can be concluded that the RUOD dataset does not have a uniform class representation, which is confirmed by the normalized class representation bias ratio and is supported by the established class imbalance ratio, Gini coefficient, and coefficient of variation. The comparison of the detection performance of the different classes further reveals

that the class representation is not proportional to the detection performance. While several underrepresented classes have high AP@0.5, the most represented class does not have the best detection performance. The least represented category of jellyfish shows relatively poor performance on the three detectors. The results suggest that class imbalance in a dataset may be a significant potential source of detection bias, though this bias varies by class and cannot be determined from the number of classes.

7. Challenges and Future Directions

7.1. Challenges

There are certain challenges associated with the detection of underwater objects due to environmental biasing, due to light absorption, scattering, and distortion of color in the underwater situation. They can lead to significant deterioration of image quality and dataset inconsistency, causing poor model accuracy and generalization. Some of the challenges that need to be addressed further are discussed below, as represented in Figure 14.

Fig. 14 Key challenges in Underwater Object Detection Bias

7.1.1. Tiny Oriented Object Detection

To start with, oriented tiny object detection is a pervasive problem both across applications (e.g., autonomous driving, medical imaging, defect inspection) and modalities (e.g., SAR, thermal, X-ray). The dataset is derived from RGB aerial imagery, with an emphasis on representative environments in which oriented tiny objects are commonly encountered. By isolating this regime, a controlled evaluation environment is established for grounding methods and analyses while providing a foundation for extending the approach to broader scenarios and modalities. Future research could incorporate complementary cues from multiple modalities or leverage temporal information to further enhance oriented tiny-object detection across diverse scenarios [1].

7.1.2. Full Annotations from Training Set

Second, the proposed approach is developed within a closed-set environment based on complete annotations of the training dataset. The object annotations of oriented tiny objects are, however, unusual, and the way of acquiring them is also not easy, particularly towards the open world. Experimental results have shown that label-efficient methods show competitive performance compared to fully supervised methods. Thus, it is worth further exploring the simplification

of annotations and the enhancement of performance with limited annotations [1].

7.1.3. Foundational Models

Foundation models are emerging to facilitate various research directions, while this work did not analyze or improve relevant works. Performance of foundation models and their adaptation to oriented tiny objects are other questions that may be investigated in the future [1].

7.1.4. Class Imbalance Bias

In many underwater datasets, there are significantly more common category samples than rare ones, and thus the detectors normally pick frequent samples and tend to miss the minority samples like small fish, debris, or a rare marine species. This results in biased performance that appears to perform well on a benchmark measure and poorly in actual deployments [2, 26].

7.1.5. Small-Object Bias

Underwater targets tend to be small, remote, or partially hidden, and thus, the detectors are biased toward large and easy-to-detect objects. This is particularly an issue since small objects will tend to be of most significance in surveillance, ecology, and rescue applications [2, 31].

7.1.6. Image-Quality Bias

Turbidity of water, scattering and low light, blur, and color distortion decrease feature quality and move the detector to less turbid water or less challenging scenes. The models that are trained on cleaner subsets would perform poorly on degraded underwater images [26, 31].

7.1.7. Noisy-Label Bias

The task of annotating underwater images is challenging, and thus, there can be erroneous or inconsistent labels in the datasets, which can bias learning to incorrect object boundaries or categories. This usually occurs in cases where the images are blurred, crowded, and low contrast [26].

7.1.8. Dataset and Domain Bias

Most of the datasets are restricted in species, location, depths, or camera configurations, and thus models trained in one environment do not predict well in new conditions in the sea. This causes the accuracy that is reported to be excessively optimistic when the test data is excessively close to the training data [26].

7.2. Limitations & Further Scope

Although the evaluated YOLO architectures achieved good detection performance, some limitations should be noted. First, the result experiments were performed on the RUOD dataset, and thus the results obtained may not be fully representative of generalizing to underwater environments with significantly different imaging conditions, object distributions, and camera types. Second, there is some obvious class imbalance in the dataset, as the jellyfish, turtle, and cuttlefish are minor classes when compared with the dominant classes of fish and echinus. While the models were found to be very robust to this imbalance, the comparatively lower performance of the most underrepresented class shows that data scarcity is still a problem. Thirdly, each architecture was tested with a single training run, so the differences reported must be interpreted in terms of the observed empirical differences, and not as statistically proven improvements over repeated runs.

Future work will center on increasing the generalizability capabilities using larger and more varied underwater datasets, targeted augmentation, and data-balancing approaches for minority classes, and testing on other independent underwater datasets. Other experiments could also explore lightweight modification of architecture and optimization for deployment to reduce the computational demands and still preserve the accuracy of the detection. There could also be repeated training runs performed with controlled random seeds to gain a quantitative understanding of how the performance varies and to get a more statistically robust comparison between the different architectures.

7.3. Future Directions

- Bias-aware dataset design: Future research must create more balanced datasets in terms of categories, object sizes, habitats, and visibility levels to minimize class and domain imbalance. The detectors would be fairer in underwater conditions when more diverse data is used [26, 31].
- Small-object-focused architectures: Fusion of multi-scale features, attention modules, and frequency-aware designs are potential to minimize bias against small and partially hidden targets. They are more useful in maintaining weak object signals in complex scenes in an underwater environment [2, 31].
- Cross-domain and zero-shot generalization: Further studies are required regarding adapting models to various water types, depths, and sensors without complete retraining. This would reduce the dependence on one dataset and improve practical fairness across environments [26].

8. Conclusion

This study aims to comparatively evaluate object detection performance of YOLOv5s, YOLOv8s and YOLOv11s, and to gain insight into the effects of dataset bias on the detection performance of the three YOLO models on the RUOD dataset. The results show that YOLOv11s has the highest overall performance, gaining an advantage in all IoU thresholds, whereas YOLOv8s and YOLOv5s have a slightly lower performance but are close to YOLOv11s.

Bias analysis of both training and validation sets confirms a significant imbalance in class distribution, with certain

classes, such as fish and echinus, being overrepresented, while others, including turtle, cuttlefish, and jellyfish, remain underrepresented. However, an important observation is that detection performance does not strictly follow class frequency. Several underrepresented classes achieved high precision and recall, whereas some dominant classes exhibited comparatively lower performance.

This suggests that dataset bias in underwater object detection is not solely dependent on class frequency, but is also influenced by factors such as object visibility, feature distinctiveness, and environmental conditions.

Additionally, differences in convergence behavior indicate that newer architectures, particularly YOLOv11s, learn more efficiently, achieving optimal performance at earlier training stages.

Overall, the findings emphasize that while architectural advancements improve detection performance, they do not fully eliminate dataset bias. Future work should focus on bias-aware dataset design, targeted augmentation for underrepresented classes, and adaptive training strategies to enhance robustness in real-world underwater environments.

Compliance with Ethical Standards

Funding
Not Applicable.

Conflict of Interest
The authors declare that they have no conflicts of interest.

Ethical Approval
We would like to highlight that this research did not involve the use of human or animal subjects.

Data Availability
The data that support the findings of this research are publicly available. The Real-world Underwater Object Detection (RUOD) dataset can be obtained from its official source. All preprocessing procedures, model configurations, and training settings are fully described in the methodology section to facilitate reproducibility of the results.

Clinical Trial Number
Not applicable.

Consent to Publish Declaration
Not applicable.

Competing Interests
The authors declare that they have no competing interests.

Consent to Participate
Not applicable.

References
[1] Nicolas Carion et al., "End-to-End Object Detection with Transformers," *Computer Vision – ECCV 2020, Lecture Notes in Computer Science*, vol. 12346, pp. 213-229, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Wei Liu et al., "SSD: Single Shot MultiBox Detector," *Computer Vision – ECCV 2016, Lecture Notes in Computer Science*, vol. 9905, pp. 21-37, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[3] Joseph Redmon, and Ali Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Shaoqing Ren et al., "Object Detection Networks on Convolutional Feature Maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1476-1481, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Zhaohui Zheng et al., "Zone Evaluation: Revealing Spatial Bias in Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 8636-8651, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Aharon Azulay, and Yair Weiss, "Why Do Deep Convolutional Networks Generalize So Poorly to Small Image Transformations?," *Journal of Machine Learning Research*, vol. 20, no. 184, pp. 1-25, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Anadi Chaman, and Ivan Dokmanić, "Truly Shift-Invariant Convolutional Neural Networks," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, pp. 3772-3782, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Osman Semih Kayhan, and Jan C. van Gemert, "On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 14262-14273, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Richard Zhang, "Making Convolutional Networks Shift-Invariant Again," *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 7324-7334, 2019. [Google Scholar] [Publisher Link]

[10] Sourajit Saha, and Tejas Gokhale, "Improving Shift Invariance in Convolutional Neural Networks with Translation Invariant Polyphase Sampling," *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Tucson, AZ, USA, pp. 620-629, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Renato M. Silva et al., "Vulnerable Road User Detection and Safety Enhancement: A Comprehensive Survey," *Expert Systems with Applications*, vol. 292, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[12] Joshua Siegel, and Georgios Pappas, "Morals, Ethics, and the Technology Capabilities and Limitations of Automated and Self-Driving Vehicles," *AI and Society*, vol. 38, no. 1, pp. 213-226, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Manisha Saini, and Seba Susan, "Tackling Class Imbalance in Computer Vision: A Contemporary Review," *Artificial Intelligence Review*, vol. 56, pp. 1279-1335, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Kemal Oksuz et al., "Imbalance Problems in Object Detection: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3388-3415, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Manisha Saini, and Seba Susan, "Bag-of-Visual-Words Codebook Generation Using Deep Features for Effective Classification of Imbalanced Multi-Class Image Datasets," *Multimedia Tools and Applications*, vol. 80, no. 14, pp. 20821-20847, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Joy Buolamwini, and Timnit Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, pp. 77-91, 2018. [Google Scholar] [Publisher Link]

[17] Daeun Lee, and Jinkyu Kim, "Resolving Class Imbalance for LiDAR-Based Object Detector by Dynamic Weight Average and Contextual Ground Truth Sampling," *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, pp. 682-691, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Mark Mazumder et al., "DataPerf: Benchmarks for Data-Centric AI Development," *Advances in Neural Information Processing Systems*, Red Hook, NY, USA, pp. 5320-5347, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Hussain Alibrahim, and Simone A. Ludwig, "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization," *2021 IEEE Congress on Evolutionary Computation (CEC)*, Kraków, Poland, pp. 1551-1559, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Angelina Wang et al., "Revise: A Tool for Measuring and Mitigating Bias in Visual Datasets," *International Journal of Computer Vision*, vol. 130, pp. 1790-1810, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Dewant Katare et al., "Analyzing and Mitigating Bias for Vulnerable Road Users by Addressing Class Imbalance in Datasets," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 6, pp. 590-604, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Syed Sahil Abbas Zaidi et al., "A Survey of Modern Deep Learning-Based Object Detection Models," *Digital Signal Processing*, vol. 126, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Zeran Wang et al., "The Evolution of Object Detection from CNNs to Transformers and Multi-Modal Fusion," *Scientific Reports*, vol. 16, pp. 1-20, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[24] Bimsara Pathiraja, Caleb Liu, and Ransalu Senanayake, "Fairness in Autonomous Driving: Towards Understanding Confounding Factors in Object Detection under Challenging Weather," *arXiv preprint*, pp. 1-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Haiping Ma et al., "Weighted Multi-Error Information Entropy Based You Only Look Once Network for Underwater Object Detection," *Engineering Applications of Artificial Intelligence*, vol. 130, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Long Chen et al., "Underwater Optical Object Detection in the Era of Artificial Intelligence: Current, Challenge, and Future," *ACM Computing Surveys*, vol. 58, no. 3, pp. 1-34, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[27] Joseph L. Walker et al., "Underwater Object Detection Under Domain Shift," *IEEE Journal of Oceanic Engineering*, vol. 49, no. 4, pp. 1209-1219, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Haojie Chen et al., "UAMFDet: Acoustic-Optical Fusion for Underwater Multi-Modal Object Detection," *Journal of Field Robotics*, vol. 42, no. 2, pp. 970-983, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[29] Shucheng Li, and Xing Peng, "DyAqua-YOLO: A High-Precision Real-Time Underwater Object Detection Model Based on Dynamic Adaptive Architecture," *Frontiers in Marine Science*, vol. 12, pp. 1-18, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[30] Jinghua Huang et al., "YOLOv8-UC: An Improved YOLOv8-Based Underwater Object Detection Algorithm," *IEEE Access*, vol. 12, pp. 172186-172195, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Wenling Wang, Zhibin Yu, and Mengxing Huang, "Refining Features for Underwater Object Detection at the Frequency Level," *Frontiers in Marine Science*, vol. 12, no. 11, pp. 1-11, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[32] Shifeng Zhang et al., "Bridging the Gap between Anchor-Based and Anchor-Free Detection via Adaptive Training Sample Selection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 9756-9765, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[33] Mingxing Tan, Ruoming Pang, and Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 10778-10787, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Tsung-Yi Lin et al., "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318-327, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] Joseph Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 779-788, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[36] Joseph Redmon, and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 6517-6525, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[37] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, pp. 7464-7475, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[38] Shaoqing Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[39] Zhaowei Cai, and Nuno Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 6154-6162, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[40] Chenping Fu et al., "Rethinking General Underwater Object Detection: Datasets, Challenges, and Solutions," *Neurocomputing*, vol. 517, pp. 243-256, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[41] Yan′e Duan et al., "Review on Visual Attributes Measurement Research of Aquatic Animals Based on Computer Vision," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 31, no. 15, pp. 1-11, 2015. [Google Scholar] [Publisher Link]

[42] Muwei Jian et al., "Underwater Object Detection and Datasets: A Survey," *Intelligent Marine Technology and Systems*, vol. 2, pp. 1-12, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[43] David B. Gillis, "An Underwater Target Detection Framework for Hyperspectral Imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 1798-1810, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[44] Muwei Jian et al., "Underwater Image Processing and Analysis: A Review," *Signal Processing: Image Communication*, vol. 91, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[45] Xiaoting Shi et al., "Underwater Cage Boundary Detection Based on GLCM Features by Using SVM Classifier," *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Hong Kong, China, pp. 1169-1174, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[46] Huilin Ge et al., "A Deep Learning Model Applied to Optical Image Target Detection and Recognition for the Identification of Underwater Biostructures," *Machines*, vol. 10, no. 9, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[47] Huilin Ge et al., "Single-Stage Underwater Target Detection Based on Feature Anchor Frame Double Optimization Network," *Sensors*, vol. 22, no. 20, pp. 1-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[48] Fei Lei, Feifei Tang, and Shuhan Li, "Underwater Target Detection Algorithm Based on Improved YOLOv5," *Journal of Marine Science and Engineering*, vol. 10, no. 3, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[49] Martin Zurowietz, and Tim W. Nattkemper, "Unsupervised Knowledge Transfer for Object Detection in Marine Environmental Monitoring and Exploration," *IEEE Access*, vol. 8, pp. 143558-143568, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[50] Dewant Katare et al., "Bias Detection and Generalization in AI Algorithms on Edge for Autonomous Driving," *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, Seattle, WA, USA, pp. 342-348, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[51] Chunhui Zhang, and Shenming Gu, "Fish Object Detection Based on Spatial Bias Pyramid Pooling: Improved BIAS-YOLO," *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, Xi'an, China, pp. 1976-1979, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[52] Aryan Tummala, Vyaas Baskar, and Sohail H. Zaidi, "Impact of Lighting-Based Biases on the Performance of YOLOv8 Object Detection Models," *2024 IEEE International Conference on Control & Automation, Electronics, Robotics, Internet of Things, and Artificial Intelligence (CERIA)*, Bandung, Indonesia, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[53] Ross Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 580-587, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[54] Sandeep Nandal, and Alka Chaudhary, "Objects Relativity Bias: For Detected Objects if They Are in the Range of Their Bigger Compositions," *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, pp. 1-4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[55] Singharat Rattanaphan, and Alexia Briassouli, "Evaluating Generalization, Bias, and Fairness in Deep Learning for Metal Surface Defect Detection: A Comparative Study," *Processes*, vol. 12, no. 3, pp. 1-32, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[56] Chang Xu et al., "Oriented Tiny Object Detection: A Dataset, Benchmark, and Dynamic Unbiased Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 48, no. 3, pp. 3167-3184, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[57] Edwine Nabahirwa et al., "A Structured Review of Underwater Object Detection Challenges and Solutions: From Traditional to Large Vision Language Models," *arXiv preprint*, pp. 1-72, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[58] Yifan Wei et al., "RHS-YOLOv8: A Lightweight Underwater Small Object Detection Algorithm Based on Improved YOLOv8," *Applied Sciences*, vol. 15, no. 7, pp. 1-21, 2025. [CrossRef] [Google Scholar] [Publisher Link]