electronics

MDPI

Article

FishDet-YOLO: Enhanced Underwater Fish Detection with Richer Gradient Flow and Long-Range Dependency Capture through Mamba-C2f

Chen Yang, Jian Xiang *, Xiaoyong Li † and Yunjie Xie †

School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Liuxia, Hangzhou 310023, China; 222208855003@zust.edu.cn (C.Y.)
* Correspondence: xiangjian@zust.edu.cn
† These authors contributed equally to this work.

Abstract: The fish detection task is an essential component of marine exploration, which helps scientists monitor fish population numbers and diversity and understand changes in fish behavior and habitat. It also plays a significant role in assessing the health of marine ecosystems, formulating conservation measures, and maintaining biodiversity. However, there are two main issues with current fish detection algorithms. First, the lighting conditions underwater are significantly different from those on land. In addition, light scattering and absorption in water trigger uneven illumination, color distortion, and reduced contrast in images. The accuracy of detection algorithms can be affected by these lighting variations. Second, the wide variation of fish species in shape, color, and size brings about some challenges. As some fish have complex textures or camouflage features, it is difficult to differentiate them using current detection algorithms. To address these issues, we propose a fish detection algorithm—FishDet-YOLO—through improvement in the YOLOv8 algorithm. To tackle the complexities of underwater environments, we design an Underwater Enhancement Module network (UEM) that can be jointly trained with YOLO. The UEM enhances the details of underwater images via end-to-end training with YOLO. To address the diversity of fish species, we leverage the Mamba model's capability for long-distance dependencies without increasing computational complexity and integrate it with the C2f from YOLOv8 to create the Mamba-C2f. Through this design, the adaptability in handling complex fish detection tasks is improved. In addition, the RUOD and DUO public datasets are used to train and evaluate FishDet-YOLO. FishDet-YOLO achieves mAP scores of 89.5% and 88.8% on the test sets of RUOD and DUO, respectively, marking an improvement of 8% and 8.2% over YOLOv8. It also surpasses recent state-of-the-art general object detection and underwater fish detection algorithms.

Keywords: underwater fish detection; underwater image enhancement; YOLO; Mamba; object detection

check for updates

Citation: Yang, C.; Xiang, J.; Li, X.; Xie, Y. FishDet-YOLO: Enhanced Underwater Fish Detection with Richer Gradient Flow and Long-Range Dependency Capture through Mamba-C2f. *Electronics* **2024**, *13*, 3780. https://doi.org/10.3390/electronics13183780

Academic Editor: Silvia Liberata Ullo

Received: 17 August 2024
Revised: 14 September 2024
Accepted: 18 September 2024
Published: 23 September 2024

1. Introduction

Underwater fish detection is indispensable in marine ecological conservation, fisheries management, and scientific research. Fish play a critical role in marine ecosystems and make a significant contribution to ecological balance [1–3]. Effective fish detection can enable researchers to better understand fish population dynamics, migration patterns, and habitat distribution and to prevent population decline or extinction through the formulation of effective conservation strategies [1]. Through underwater fish detection, scientists can gather extensive data on fish behavior, species diversity, and ecosystem health [2]. These data are vital for advancing marine biology and ecology, understanding the impacts of climate change on marine life, and providing marine conservation policies with a scientific foundation [3]. Therefore, the underwater fish detection task is of great significance. However, current underwater fish detection tasks are difficult to apply in real-world scenarios due to two main reasons. First, underwater images often suffer from issues such as blurriness, noise, and low quality, which result in limited critical information and interfere with fish detection. Second, the significant differences in morphology, color, and other characteristics among varIoUs fish species pose a challenge to the performance of detection systems. Addressing these two issues is crucial for practical applications.

With the widespread adoption of modern information technology and advancements in computational power, artificial intelligence has rapidly progressed. Computer vision technology, as a crucial branch of artificial intelligence, has proven highly effective for fish detection tasks. Several researchers have applied basic computer vision algorithms to underwater fish detection tasks [4–6]. For example, Minsung Sung et al. [4] applied YOLO [5] to fish object detection and achieved real-time fish detection. Furthermore, T Hong Khai et al. [6] categorized image data into three classes based on fish density: low, medium, and high. Through a parameter calibration strategy, they identified suitable parameters and employed a Mask R-CNN model with Mask priors [7] for fish detection and counting. Minsung Sung et al. used an earlier version of the YOLO algorithm with a lower accuracy. By contrast, T Hong Khai et al.'s Mask-RCNN required Mask annotations, which are scarce in fish detection tasks. Thus, earlier visual algorithms could not meet the standards for fish detection tasks.

Over nearly a decade, deep learning-based object detection technology has evolved, which is broadly categorized into two-stage algorithms [8–10] and single-stage algorithms [11–16]. The two-stage algorithms provide higher accuracy but poorer real-time performance. For instance, R Girshick et al. [8] proposed the R-CNN algorithm, which generated candidate regions through selective search, extracted features from these regions using convolutional neural networks, and employed classifiers for object classification and regressors for precise boundary localization. Later, Fast R-CNN [9] and Faster R-CNN [10] were introduced. Moreover, Faster R-CNN was incorporated with Region Proposal Networks (RPNs) to generate candidate regions on shared feature maps. While enhancing the efficiency of object detection, this achieves faster and more accurate results. Single-stage algorithms offer better real-time performance. The SSD algorithm proposed by W Liu et al. [11] generates predefined anchor boxes on multi-scale feature maps and performs object classification and location regression in a single forward pass. YOLOv1, introduced by Redmon et al. [5], reformulates object detection as a single neural network inference and predicts object classes and locations directly from images with significantly improved detection speed. Subsequent versions of YOLO, including YOLOv3 [12], further improve network structure with the deeper Darknet-53 and three detection heads for better performance in complex scenes and small objects. Later updates to YOLO include advancements in training strategies, such as mosaic data augmentation [13], self-adversarial training, and automated searches for efficient network modules using techniques such as NASNet [14]. The latest versions, YOLOv9 [15] and YOLOv10 [16], achieve state-of-the-art performance in both accuracy and real-time capability. Despite these advancements, YOLO and other convolutional neural network-based methods still have difficulty handling long-distance dependencies. This limitation affects their ability to differentiate complex fish shapes and textures and to address challenges such as murky water, low resolution, and shape deformations. Furthermore, it is still challenging for convolutional neural networks to distinguish subtle differences between some fish species. Hence, the current classical object detection algorithms have yet to provide an outstanding solution specifically suitable for fish detection.

In recent years, self-attention [17]-based network architectures have addressed the issue of long-distance dependencies that convolutional neural networks cannot handle. With the introduction of algorithms such as Vision Transformer [18] and Swin Transformer [19], Yi Xiao et al. [20] introduced the Top-k Token Selective Transformer (TTST), which optimizes self-attention by removing redundant tokens and using a multi-scale feed-forward layer. The TTST dynamically selects key tokens with a Residual Token Selective Group (RTSG) and enhances accuracy with Global Context Attention (GCA). In the object detection field, Transformers have been applied to detection algorithms to capture long-range dependencies. For example, N Carion et al. [21] proposed DETR, a Transformer-based object detection algorithm modeling set predictions directly, enabling end-to-end object detection without the need for anchor boxes and non-maximum suppression. Furthermore, X Zhu et al. [22] improved DETR with Deformable DETR, which focuses only on a few key sampling points around the target boxes and triggers better performance. However, Transformers have a computational complexity of $O\left(N^2\right)$, slower convergence, and higher computational resource requirements compared with the linear computational complexity of convolutional neural networks. Over the past few years, several researchers have applied the latest object detection algorithms and advanced Transformer technologies to fish detection in underwater environments [23–27]. For instance, Abdullah Al Muksit et al. [23] proposed YOLO-Fish, enhancing the model's ability to detect fish in dynamic environments through the incorporation of Spatial Pyramid Pooling into the network architecture. In order to improve fish species recognition, Chiranjibi Shah et al. [24] modified the depth scales of different layers in the backbone network of YOLOv5. In addition, they introduced Transformer Blocks in the backbone network to enhance long-range dependency perception and incorporated a class-balanced loss function to address class imbalance issues, so as to improve performance. DP-FishNet, a dual-path Pyramid Vision Transformer [27] (PVT) feature extraction network, was proposed by Yang Liu et al. [25]. The backbone network of DP-FishNet, DP-PVT, consists of PVT networks with two feature extraction paths, in which one Vision Transformer path is to extract global features to enhance the distinction between foreground and background in underwater images and the other convolutional neural network path is to extract local features to improve small object detection accuracy, so as to strengthen the ability to extract both global and local features from underwater images. Z Wang et al. [26] introduced DyFish-DET, an underwater object detection method. In DyFish-DET, a new backbone network, DyFishNet, was designed to better extract fish texture features. In addition, a streamlined hybrid encoder was developed to integrate fish body feature information. However, these works are confronted with two main issues. Firstly, the computational complexity of Transformers is still unresolved. Secondly, these studies primarily enhance accuracy by adding improved modules to the network structure, but they fail to address underwater image enhancement for complex underwater environments. Simple modification of the network structure is not enough to tackle the challenges posed by complex marine underwater environments and achieve ideal accuracy. In the design of YOLOv9, the gradient flow information in images is a crucial component of object detection. Thus, the latest version of YOLO incorporates new modules into the network architecture, in order to pass rich gradient flow information to the detection heads and allow them to determine the position and class of objects based on this information. However, when the gradient flow information in the original input images is sparse (e.g., in complex marine underwater images), this information is severely deficient. Thus, mere modification of the YOLO internal network structure does not generate richer gradient flow information and essentially fails to address the complexities of input marine environments.

Images consist of high-frequency and low-frequency components. Furthermore, high-frequency information includes details and edges, whereas low-frequency information represents smooth areas and coarse structures. Together, they determine the clarity and overall contour of the image, with gradient flow information primarily concentrated in the high-frequency part [28–30]. In order to enhance gradient flow information in underwater images, it is essential to improve the high-frequency components of the image. C. Huo et al. [28] proposed HHDNet, which uses dual-branch network architecture to handle high-frequency and low-frequency information. For the high-frequency component, HHDNet designs specialized attention modules for enhancement, while the basic convolutional modules are used for the low-frequency component, which successfully improves the quality of underwater images. Xin Liu et al. [29] introduced the Frequency Domain Enhancement Attention Network (FEAN), which decomposes features into high-frequency and low-frequency components through Laplacian decomposition and processes these features via Frequency Domain Enhancement Attention Modules (FEAMs). In the high-frequency path, the FEAN incorporates Multi-scale Attention Enhancement Blocks (MAEBs) to extract rich image textures and gradient flow information, while the low-frequency path employs simple convolution operations to adjust image brightness and contrast. K Jiang et al. [30] introduced the Mutual Retinex framework, which combines self-attention (SA) and convolutional neural networks (CNNs) to enhance low-light and underwater images. It uses a dual-branch structure to separate and optimize reflectance and illumination and a mutual learning mechanism with a Mutual Representation Module (MRM) to improve color consistency and naturalness. Experiments show that Mutual Retinex significantly outperforms existing methods in both low-light and underwater image enhancement. Although these methods effectively enrich gradient flow information for underwater image enhancement, they are separately trained underwater image network structures and require annotated information for supervised training, making them incompatible with joint training for fish detection.

Due to their strong capability in modeling long-range dependencies as well as their superior performance with linear time complexity, the state space model (SSM) methods like Mamba [31] can provide a new approach to address the computational and time complexity issues found in Transformer models. Researchers have successfully applied the Mamba architecture to the computer vision field and achieved promising results in image classification tasks [32,33]. Y Xiao et al. [34] applied the Vision State Space Model (Mamba) to super-resolution for remote sensing images, using linear complexity to manage large-scale RSI. They introduced an enhanced frequency-assisted Mamba framework (FMSR) that integrates spatial frequency fusion with a multi-level architecture, significantly improving super-resolution performance. Zhaohu Xing et al. [35] proposed SegMamba, a novel 3D medical image segmentation model based on SSM. SegMamba effectively captures long-range dependencies in volumetric features and outperforms Transformer methods in processing speed, maintaining high efficiency even at high resolutions. However, to date, there has still been no research applying the Mamba model to underwater fish detection tasks. Based on this, the following question is proposed: Can we introduce SSM structures into the object detection domain and leverage their advantages to enhance the performance of fish detection tasks?

Given the complexity of underwater environments and the importance of gradient flow information in object detection algorithms, this study aims to adopt design principles from underwater image enhancement algorithms to construct a new image enhancement subnetwork. This subnetwork is intended to enhance the high-frequency components of the image, enrich gradient flow information, and thus enable end-to-end joint training with object detection algorithms without the need for additional supervisory information. Due to the complexity of different fish species, this study seeks to incorporate the SSM model, such as Mamba, into fish detection tasks to improve detection accuracy without significantly increasing computational complexity. Thus, in combination with the advanced YOLOv8 algorithm, the FishDet-YOLO algorithm has been proposed. The main contributions are as follows:

(1) To address the complexities of underwater environments and the diversity of fish species, and to improve detection performance, we propose the FishDet-YOLO algorithm.

(2) To address issues of uneven lighting, color distortion, and reduced contrast in underwater environments and to improve gradient flow information in images, we design the Underwater Enhancement Module (UEM). The UEM uses Laplacian pyramids for image decomposition and applies multi-scale fusion methods. Additionally, we design the Underwater Detail-Aware Block (UDAB). The UDAB processes the high-frequency and low-frequency components of the image at each scale separately. During training, high-frequency information is reinforced at each fusion stage. The UEM significantly enhances the details and gradient flow information in underwater images. Compared to other algorithms that use a two-stage approach, the UEM supports end-to-end joint training with object detection algorithms, improving application efficiency.

(3) Given the complexity of different fish species, we leverage the Mamba model's capability to enhance long-distance dependencies without increasing computational complexity. By integrating the C2f from YOLO with the Mamba, we design the Mamba-C2f, which replaces all C2f in YOLOv8. Compared to existing methods, our approach not only enhances long-distance dependency capabilities but also maintains the inductive bias of convolution, thereby improving adaptability and detection accuracy for complex fish species.

The rest of this paper can be organized as follows. Section 2 comprehensively describes FishDet-YOLO and its improvements, including the UEM network structure, the UDAB module within the UEM, and the newly improved Mamba-C2f feature extraction module. Furthermore, Section 3 presents experimental results and analysis, including a systematic analysis of the dataset, data preprocessing steps, evaluation metrics used, as well as the hardware and software configurations of the experiments. It also comprehensively evaluates FishDet-YOLO, covering comparative experiments, ablation studies, and visualization results. Section 4 discusses the research findings, limitations, and future work. Finally, Section 5 is the research summary.

2. Methods

Figure 1 shows the overall architecture of the FishDet-YOLO, which consists of two main components. The first component is the Underwater Enhancement Module (UEM), which takes the raw image as input and enhances the underwater image to improve the gradient flow information. The UEM is built on a multi-scale fusion approach, where the input image is decomposed into four components using the Laplacian pyramid [36], and each component is processed by the Underwater Detail-Aware Block (UDAB). The UDAB consists of two branches: the high-frequency branch and the low-frequency branch, which handle the high-frequency and low-frequency information of the components, respectively. In the high-frequency branch, the Sobel operator is used to enhance edge information and enrich the gradient flow. In the low-frequency branch, the designed module mainly preserves low-frequency characteristics.

The second component of FishDet-YOLO is the object detection network. It is built upon the state-of-the-art YOLO algorithm and consists of a backbone network, a PAFPN, and a three-branch decoder head. The backbone network extracts and fuses multi-scale features, while the PAFPN [37] further enhances feature transmission and fusion from higher to lower layers. The design of the three-branch decoder head helps maintain accuracy when detecting objects of different sizes. For the underwater fish detection task, the Mamba-C2f, which combines Mamba with C2f, is introduced to replace the original feature transformation module in YOLO, enhancing the long-range dependency awareness of the network. During training, the UEM network, which does not require additional image enhancement labels, is directly added before the object detection network and is jointly trained end-to-end with the object detection network.

2.1. Underwater Enhancement Module

The UEM is designed to address issues such as uneven lighting, color distortion, and reduced contrast in underwater environments and to enhance the gradient flow information in images. The UEM decomposes the input image into multiple scales using a Laplacian pyramid and processes them with a multi-scale fusion approach. At each scale, the UDAB is applied, which separately processes the high-frequency and low-frequency components of the image. During training, high-frequency information is reinforced at each fusion stage. This module significantly enhances the details and gradient flow information in underwater images and can be jointly trained end-to-end with the YOLO algorithm without the need for additional supervision, so that the perceptual capability of the detection network is improved. As shown in Figure 1, the UEM network uses the Laplacian pyramid to decompose the image into four different scales. The Laplacian pyramid focuses more on global information from the bottom layer to the top layer. Conversely, it also pays more attention to local details. This is the gradient flow information lost during image downsampling, which is the target of enhancement by our UEM.

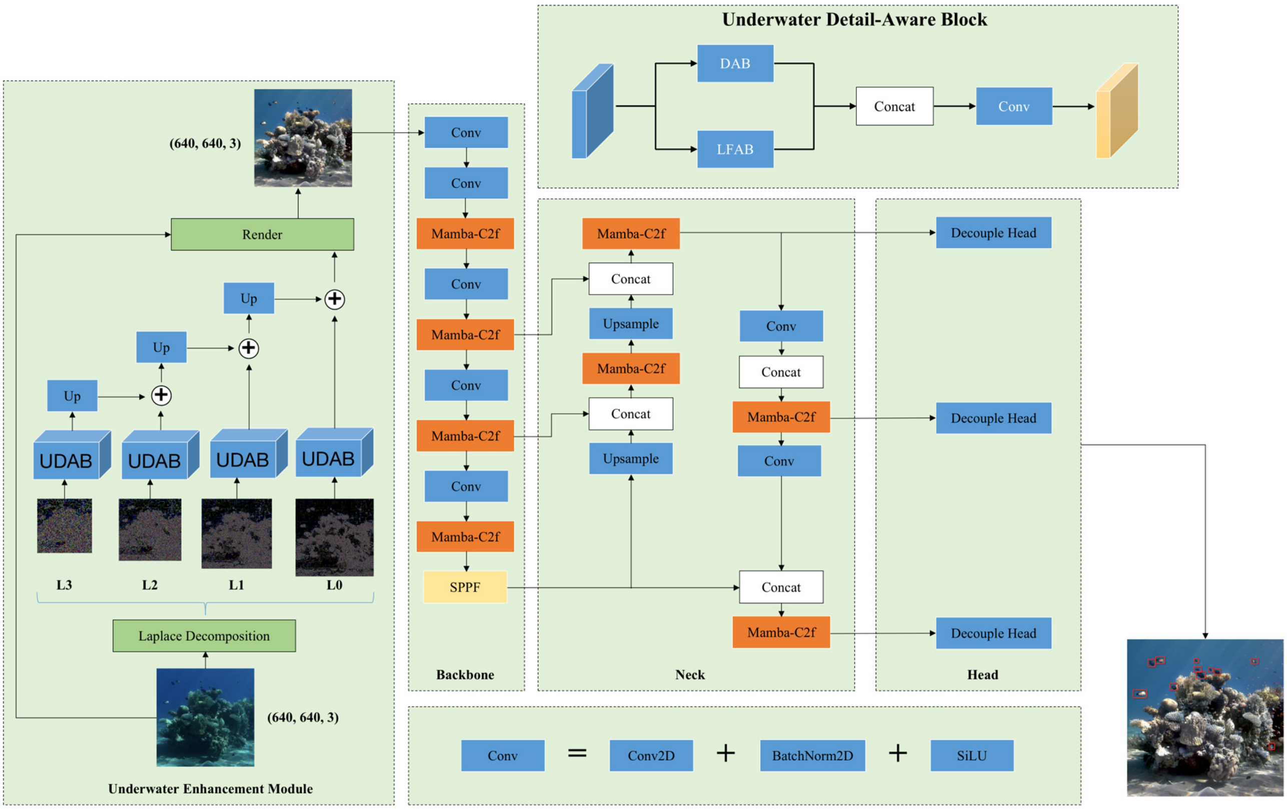


Figure 1. FishDet-YOLO network architecture. The red square represents each individual component.

Assuming the input image is $x \in \mathbb{R}^{HxWx3}$, the Laplacian pyramid is obtained using the following equation:

$$\mathrm{G(x) = Down(Gaussan(x))} \tag{1}$$

Down refers to downsampling, and Gaussian indicates the application of a Gaussian filter, with a kernel size of 5 × 5. Each time an image undergoes processing using the Gaussian pyramid, both its width and height are reduced by half, leading to a resolution that is only one-fourth of the original. Notably, the downsampling process in the Gaussian pyramid is irreversible. To retrieve the original high-resolution image after upsampling, it is necessary to compensate for the lost information. This missing information constitutes the elements of the Laplacian pyramid, which can be expressed as follows:

$$\mathrm{L_i = G_i - Up(G_{i+1})} \tag{2}$$

$\mathrm{L_i}$ represents the i-th layer of the Laplacian pyramid, $\mathrm{G_i}$ denotes the i-th Gaussian blur, and Up represents the bilinear upsampling operation. During the image reconstruction process, the original high-resolution image can be restored by performing the inverse transform corresponding to Equation (2).

After acquisition of the components of the Laplacian pyramid, each component undergoes feature enhancement through the UDAB. The top right corner of Figure 1 shows the network architecture of the UDAB. The UDAB includes two branches: the Detail-Aware Branch (DAB) and the Low-Frequency-Aware Branch (LFAB), which operate in parallel. In the wake of feature extraction, the outputs of the two branches are concatenated, and feature fusion is performed using a 1 × 1 convolution, generating the enhanced features.

In one of the UDAB branches, a DAB is proposed to enhance each component. This module is divided into the Context Block and the Edge Enhance Block.

In Figure 2, the Context Block is designed to capture contextual information by leveraging long-range dependencies to globally improve the components. The Edge Enhance Block employs Sobel operators in two orientations to compute gradients, which is helpful in edge detection and texture enhancement. The context branch incorporates residual blocks that process features before and after the long-range dependency capture, utilizing residual learning to transfer and construct detailed low-frequency information through skip connections. Specifically, the first residual block expands the channel dimension from 3 to 32, while the second block reduces it back to 3. The edge branch uses Sobel operators [38], which integrate Gaussian filters and differential derivatives to estimate gradients for edge detection. Furthermore, this study applies Sobel operators in both horizontal and vertical directions, re-extracts edge information through convolutional filters, and improves the information flow with residuals. This method can be summarized as follows:

$$\mathrm{EEB}(x) = F(\mathrm{Sobel}_h(x) + \mathrm{Sobel}_w(x)) + x \tag{3}$$

Sobel_h and Sobel_w represent sobel operations along the height h and width w directions, respectively. F represents the fusion of the Sobel_h and Sobel_w branches using convolution operations.

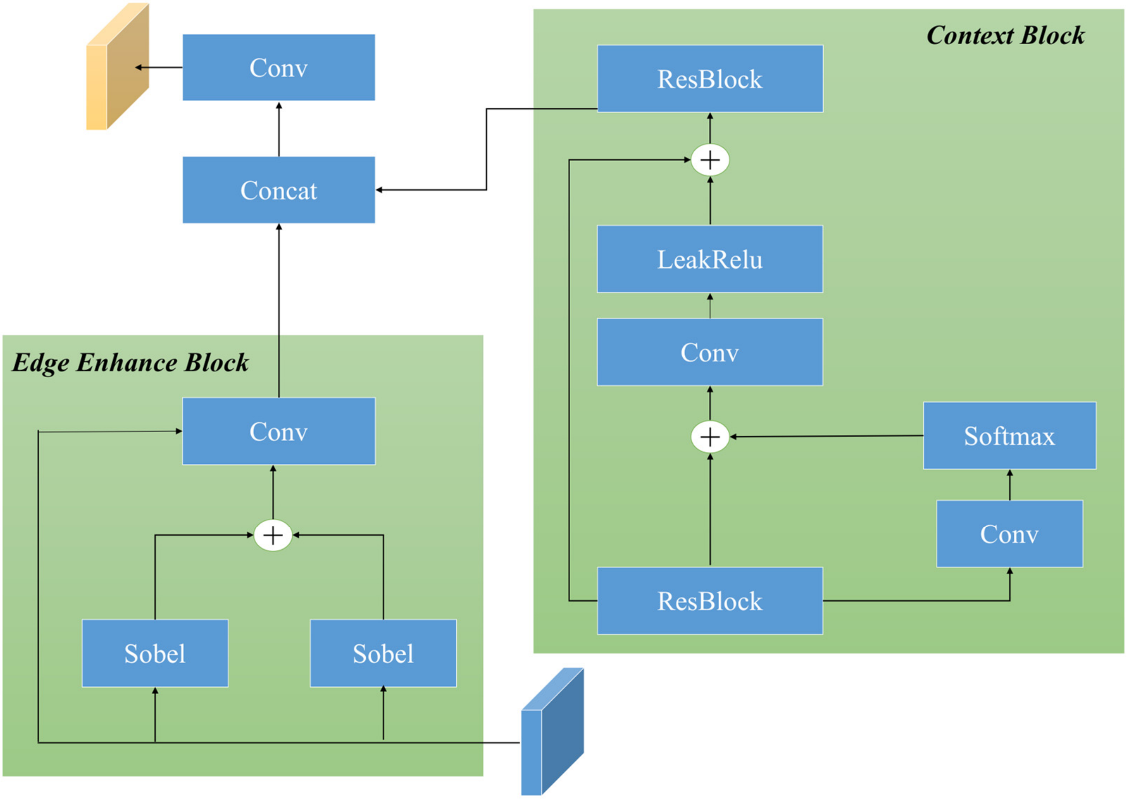


Figure 2. Structure of Detail-Aware Branch.

In the two components of the UADB, the low-frequency part of the component contains most of the semantic information of the image, which is crucial for the predictions of the detector. To enrich the semantic information of the reconstructed image, we propose the LFAB in another branch of the UADB module to capture the low-frequency information of the component (see Figure 3):

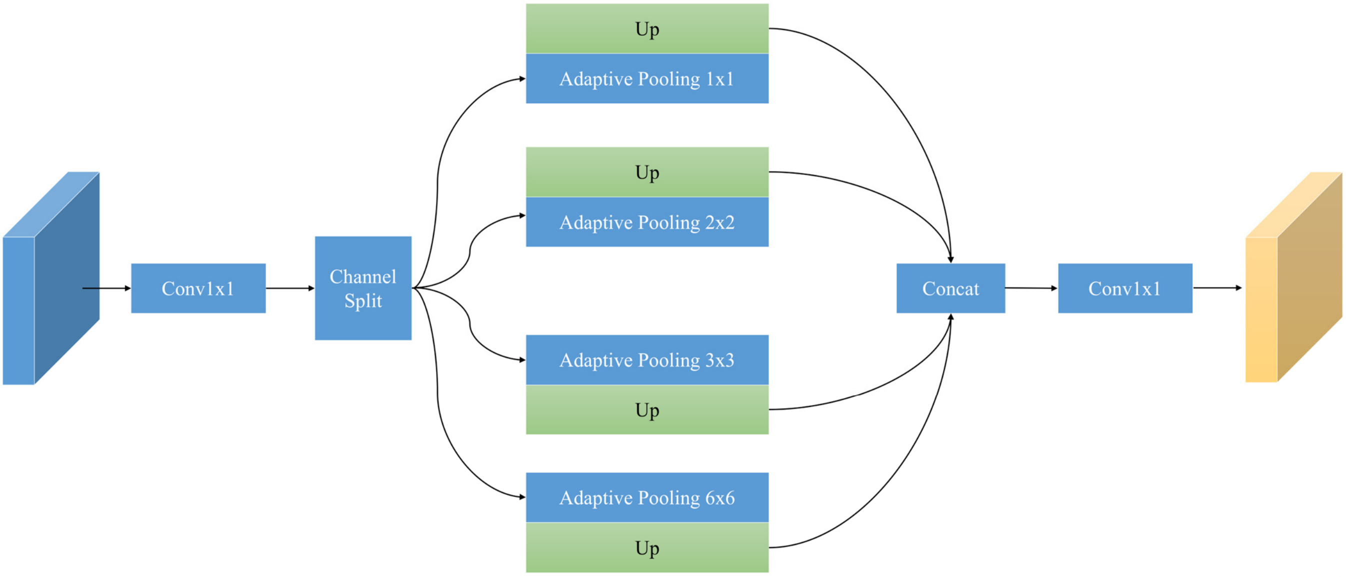


Figure 3. Structure of Low-Frequency-Aware Branch.

Assuming that the component is $f \in \mathbb{R}^{H \times W \times 3}$, this study first converts it to $f \in \mathbb{R}^{H \times W \times 32}$ through a convolutional layer and then applies a dynamic low-pass filter to capture low-frequency information and adopts average pooling for feature filtering, which allows only information below the cutoff frequency to pass through. Different semantic information has different low-frequency thresholds. Inspired by the multi-scale structure in the Inception network, this study uses adaptive average pooling with sizes 1×1, 2×2, 3×3, and 6×6, and it applies upsampling at the end of each scale to restore the original size of the features. The low-pass filter is formed through average pooling with different kernel sizes. Beyond that, f is divided into four parts $\{f_1, f_2, f_3, f_4\}$ through channel separation, with each part processed using pooling of different sizes, as described below:

$$\text{Filter}(f_i) = \text{Up}(P_s, f_i) \tag{4}$$

f_i represents the different parts obtained from channel splitting, where Up denotes bilinear interpolation upsampling and P_s represents adaptive average pooling with different sizes. After passing through the LFAB structure, the final output is $f \in \mathbb{R}^{H \times W \times 3}$.

After the components are enhanced by the DAB module, this study performs upsampling on the low-resolution components and integrates them with the high-resolution components. Let the original input image be $x \in \mathbb{R}^{H \times W \times 3}$ and the enhanced component after multi-scale fusion be $I_e \in \mathbb{R}^{H \times W \times 3}$; then:

$$I_u = \text{Render}(x, I_e) \tag{5}$$

$I_u \in \mathbb{R}^{H \times W \times 3}$ represents the image enhanced by the UEM network, which is the input to the subsequent detection network. It denotes the fusion method of x and I_e. In different low-level tasks, the rendering method varies. In this paper, an additive fusion method is adopted.

2.2. Mamba-C2f

This study leverages the strengths of the Mamba model in strengthening long-distance dependencies to enhance the ability of the detection algorithm to recognize varIoUs fish species. Mamba significantly improves the model's ability to perceive remote information without substantially increasing computational burden. Unlike Transformers, Mamba achieves long-distance dependency capture with computational complexity linear to that of convolutional operations. This makes the Mamba model particularly advantageous for capturing fine details and complex shapes of fish. Furthermore, this study combines the Mamba model with the C2f from YOLO to design the Mamba-C2f. The Mamba-C2f replaces all C2fs in YOLO and optimizes feature extraction and information fusion processes, thereby significantly improving adaptability and detection accuracy for complex fish species. The Mamba-C2f maintains computational efficiency, enhances the capability to process and recognize complex textures and shapes of fish, and improves detail capture. The module is shown in Figure 4.

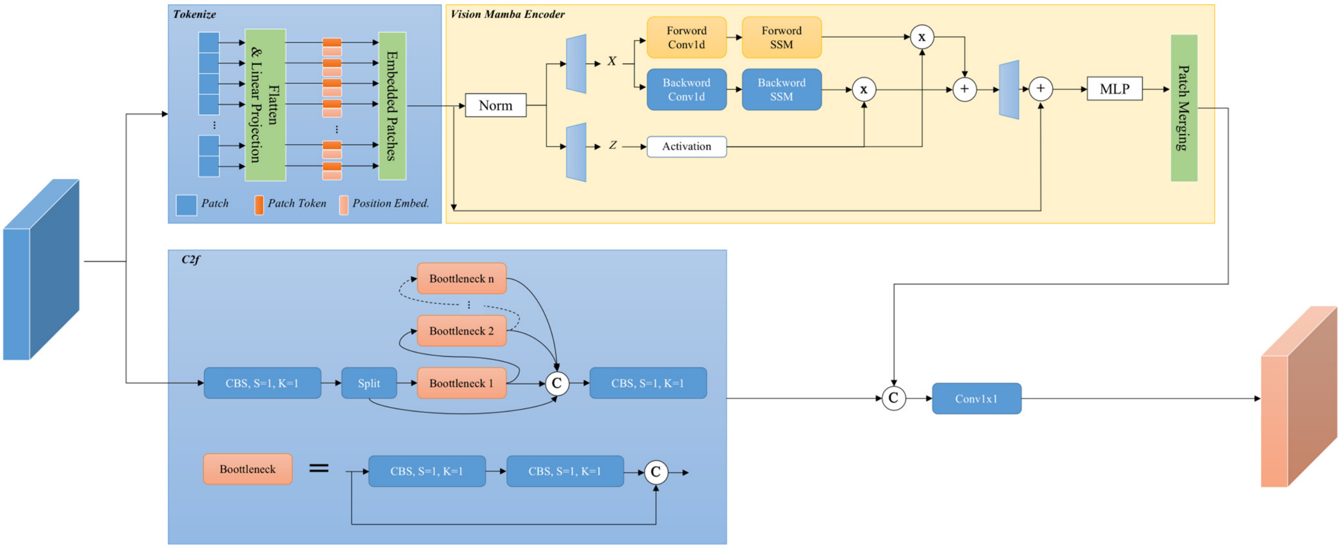


Figure 4. Network structure diagram of Mamba-C2f.

Mamba-C2f consists of the Vision Mamba Encoder and the C2f, as shown in Figure 4. Mamba-C2f includes two parallel branches: the Vision Mamba Encoder and the C2f. After the two branches output their respective features, the features are concatenated and then processed through a 1 × 1 convolution to adjust the number of channels and produce the final output.

The Vision Mamba Encoders are similar to the Transformer Encoder, while the standard Mamba is designed for one-dimensional sequences, which needs modifications for visual tasks. First, the two-dimensional image $\mathrm{t} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{C}}$ is converted into flattened two-dimensional image patches $\mathrm{x_p} \in \mathbb{R}^{\mathrm{J}\times(\mathrm{p}^2\mathrm{C})}$, where C represents the number of channels and p represents the patch size. Next, $\mathrm{x_p}$ is linearly projected into a vector of size D and positional embedding $\mathrm{E_{pos}} \in \mathbb{R}^{(\mathrm{J}+1)\times\mathrm{D}}$ is added as follows:

$$\mathrm{T}_0 = \left[\mathrm{t_{cls}}; \mathrm{t_p^1 W}; \mathrm{t_p^2 W}; \ldots; \mathrm{t_p^J W};\right] + \mathrm{E_{pos}} \tag{6}$$

$\mathrm{t_p^J}$ represents the j-th of the image t, and $\mathrm{W} \in \mathbb{R}^{(\mathrm{p}^2\mathrm{C})\times\mathrm{D}}$ represents the projection matrix. The sequence $\mathrm{T_{l-1}}$ is then sent to the l-th layer of the Vision Mamba Encoder to obtain the output $\mathrm{T_l}$. Finally, the output class tokens $\mathrm{t_0^L}$ are normalized and fed into the multi-layer perceptron (MLP) head to produce the final prediction p, as follows:

$$\mathrm{T_l} = \mathrm{Vim}(\mathrm{T_{l-1}}) + \mathrm{T_{l-1}} \tag{7}$$

$$\mathrm{f} = \mathrm{Norm}\left(\mathrm{t_0^L}\right) \tag{8}$$

$$\mathrm{p} = \mathrm{MLP}(\mathrm{f}) \tag{9}$$

VIM represents the Vision Mamba Encoder, Norm represents the normalization layer, and MLP represents the multi-layer perceptron. The original Mamba module, which is designed for one-dimensional sequences, is not suitable for vision tasks requiring spatial awareness. Therefore, the Vision Mamba Encoder incorporates bidirectional sequence modeling tailored for visual tasks (Algorithm 1).

Algorithm 1 Vision Mamba Encoder process.

```
Require: token sequence $\mathbf{T}_{l-1} : (B, M, D)$
Ensure: token sequence $\mathbf{T}_{l} : (B, M, D)$
/* normalize the input sequence $T_{l-1}$ */
$\mathbf{T}'_{l-1} : (B, M, D) \leftarrow \mathbf{Norm}(\mathbf{T}_{l-1})$
$\mathbf{x} : (B, M, E) \leftarrow \mathbf{Linear}^{\mathbf{x}}\left(\mathbf{T}'_{l-1}\right)$
$\mathbf{z} : (B, M, E) \leftarrow \mathbf{Linear}^{\mathbf{z}}\left(\mathbf{T}'_{l-1}\right)$
/* process with different direction */
for o in {forward, backward} do
    $\mathbf{x}'_o : (B, M, E) \leftarrow \mathbf{SiLU}(\mathbf{Conv1d}_0(\mathbf{x}))$
    $\mathbf{B}_o : (B, M, N) \leftarrow \mathbf{Linear}^{\mathbf{B}}_o(\mathbf{x}'_o)$
    $\mathbf{C}_o : (B, M, N) \leftarrow \mathbf{Linear}^{\mathbf{C}}_o(\mathbf{x}'_o)$
    /* softplus ensures positive $\Delta_o$*/
    $\Delta_o : (B, M, E) \leftarrow \log(1 + \exp(\mathbf{Linear}^{\Delta}_o(\mathbf{x}'_o) + \mathbf{Parameter}^{\Delta}_o))$
    /* shape of $\mathrm{Parameter}^{\mathbf{A}}_o$ is $(\mathbf{E}, \mathbf{N})$*/
    $\overline{\mathbf{A}}_o : (B, M, E, N) \leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{Parameter}^{\mathbf{A}}_o$
    $\overline{\mathbf{B}}_o : (B, M, E, N) \leftarrow \mathbf{\Delta}_o \bigotimes \mathbf{B}_0$
    $\mathbf{y}_o : (B, M, E) \leftarrow \mathbf{SSM}(\overline{\mathbf{A}}_o, \overline{\mathbf{B}}_o, \mathbf{C}_o)(\mathbf{x}'_o)$
end for
/* get gated $y_o$ */
$\mathbf{y}'_{\mathrm{forward}} : (B, M, E) \leftarrow \mathbf{y}_{\mathrm{forward}} \bigodot \mathbf{SiLU}(\mathbf{z})$
$\mathbf{y}'_{\mathrm{backward}} : (B, M, E) \leftarrow \mathbf{y}_{\mathrm{backward}} \bigodot \mathbf{SiLU}(\mathbf{z})$
/* residual connection */
$\mathbf{T}_l : (B, M, D) \leftarrow \mathbf{Linear}^{\mathrm{T}}(\mathbf{y}'_{\mathbf{forward}} + \mathbf{y}'_{\mathbf{backward}}) + \mathbf{T}_{l-1}$
Return $\mathbf{T}_l$
```

Vision Mamba Encoder processes each element in the sequence in both forward and backward directions simultaneously, with each direction potentially having different state space parameters. Through this approach, the model can simultaneously consider information from both the beginning and end of the sequence and capture and utilize spatial and contextual information in the image more comprehensively. Before the sequence processing, a Layer Normalization (LN) layer is used to standardize the input. This normalization is conductive to stabilizing the training process and improving model performance. Then, the normalized sequence is linearly projected into two different spaces for subsequent bidirectional processing and gating mechanisms. This step is achieved through two different linear layers. After the bidirectional processing, the sequence is handled in forward and backward directions. For each direction, a 1D convolution is applied to capture local dependencies. The resulting output is x'. Furthermore, this output is transformed through three linear layers to obtain the three key parameters B, C, and Δ. The parameter Δ is processed through a softplus operation to ensure its positivity, as it will be used for calculating time-scale transformations. The transformed Δ adjusts the evolution matrix A and the input matrix B. In practice, Δ serves as a scaling factor to adjust these matrices. In the wake of this transformation, the state space model computes the final output. The forward and backward outputs are combined through a gating mechanism and multiplied spatially. Subsequently, the results from both directions are summed to produce the final sequence output. This step, which involves the linear layer LinearT and residual connections, completes the final output sequence. It is then reconstructed into $t \in \mathbb{R}^{H \times W \times C}$, by the Patch Merging module.

For the C2f, the input data first pass through the initial convolutional layer, and the output is split into two parts. One part is directly fed to the final output, while the other part undergoes processing through several Bottleneck modules. Then, these two results are concatenated along the channel dimension and processed via a second convolutional layer to produce the final output. This branch design enhances the network's non-linearity and representational capacity while improving its ability to model complex data. The C2f enriches feature representation by combining features from different branches along the channel dimension, ensuring that the final concatenated features incorporate information from multiple pathways.

Finally, if the output of the Vision Mamba Encoder is $f_v \in \mathbb{R}^{H\times W\times C}$ and the output of the C2f is $f_c \in \mathbb{R}^{H\times W\times C}$, then the output M of the Mamba-C2f is:

$$M = F(\mathrm{Concat}(f_v, f_c)) \tag{10}$$

F represents a 1 × 1 convolution that maps the concatenated features from 2c to c.

2.3. Loss Function

We use three loss functions from YOLOv8: Varifocal Loss [39] (VFL Loss), Distribute Focal Loss [40] (DFL Loss), and Complete IoU Loss [41] (CIoU Loss).

The VFL Loss function is used to optimize model performance in object detection tasks. It combines the Intersection over Union (IoU) between the predicted bounding box (bbox) and the ground truth (gt) with the score for calculation, which can be expressed as follows:

$$\mathrm{VFL}(p,q)\begin{cases} -q(q\log(p) + (1-q)\log(q-p)) & q > 0 \\ -\alpha p^{\gamma}\log(1-p) & q = 0 \end{cases} \tag{11}$$

q represents the IoU between the bbox and the gt, and p represents the score. In the VFL Loss function, IoU is used as a soft label q. Unlike the basic formula, which directly divides by the sample size, the VFL Loss function improves upon this by multiplying by q. When q is greater than 0, it indicates that there is an intersection between the predicted box and the ground truth box, which can be considered a positive sample. When q is equal to 0, it means there is no intersection between the two boxes, which can be regarded as a negative sample.

The DFL Loss is used to help the network quickly focus on values near the label, maximizing the probability density at the label. The idea is to use a cross-entropy function to optimize the probabilities of the two positions adjacent to the label y, making the network's distribution concentrate around the label value.

$$\mathrm{DFL}(S_i, S_{i+1}) = -(y_{i+1} - y)\log S_i + (y - y_i)\log S_{i-1} \tag{12}$$

y_i represents the predicted value, y_{i+1} represents the adjacent predicted value, and S represents the output distribution.

CIoU Loss is designed to be more comprehensive and refined, in which multiple factors such as the shape, position, and orientation of the gt are considered. The comprehensiveness of CIoU Loss is reflected in its consideration of multiple factors, including position, shape, and orientation. This allows the model to learn the features of the target box more comprehensively, thereby improving the model's performance in complex scenarios. Whether dealing with minor or significant changes in the target box, CIoU Loss effectively guides the model in learning and optimization.

$$\mathrm{CIOU\ Loss} = \mathrm{IOU} - \left(\frac{\rho^2(b, b^{gt})}{c^2} + \sigma v\right) \tag{13}$$

$$v = \frac{4}{\pi^2}\mathrm{IOU}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \tag{14}$$

$$\sigma = \frac{v}{(1-\mathrm{IOU}) + v} \tag{15}$$

w and h refer to the width and height of the predicted box, while w^{gt} and h^{gt} refer to the width and height of the ground truth box. ρ^2 represents the squared distance between the two center points, b denotes the center coordinates of the predicted box, b^{gt} denotes the center coordinates of the ground truth box, and c represents the diagonal length of the two enclosing rectangles. v is used to measure the consistency of the relative proportions between two bounding boxes. σ represents weights of v. Finally, the overall loss is calculated as follows:

$$\text{Total Loss}(x) = \alpha \text{Loss}_{\text{VFL}}(x) + \beta \text{Loss}_{\text{DFL}}(x) + \gamma \text{Loss}_{\text{CIOU}}(x) \tag{16}$$

α, β, and γ are the loss weights. In the YOLO framework, these weights α, β, and γ are dynamically adjusted during training to ensure that each component of the loss reaches a balanced state.

3. Experiments and Analysis

3.1. Datasets

Underwater fish object detection, as a specialized type of object detection, is significantly more complex than general object detection. It faces two main challenges. First, the complexity of the targets themselves presents a challenge: targets in the ocean, such as fish and corals, have small scales, varIoUs postures, and camouflage, exhibiting significant intra-class variation and similar inter-class morphology, which greatly increases the difficulty of detection. Second, the complexity of the environment also poses a challenge. Issues such as fog effects, color distortion, and light interference are common in underwater environments, further complicating detection. Therefore, the chosen dataset must encompass these characteristics. Consequently, the RUOD dataset [42] and the DUO dataset [43] are selected for training and testing.

The RUOD dataset offers a wide range of general underwater scenes and addresses varIoUs underwater detection challenges. It includes ten target categories such as fish, divers, starfish, corals, sea turtles, sea urchins, sea cucumbers, scallops, squid, and jellyfish. The dataset features images captured in complex underwater environments with effects like fog, color distortion, and light interference, enabling a thorough evaluation of detector performance. It contains 7466 images for training and 1867 images for testing and evaluation. The distribution of samples across training and test sets, along with the number of samples per category, is shown in Figure 5.

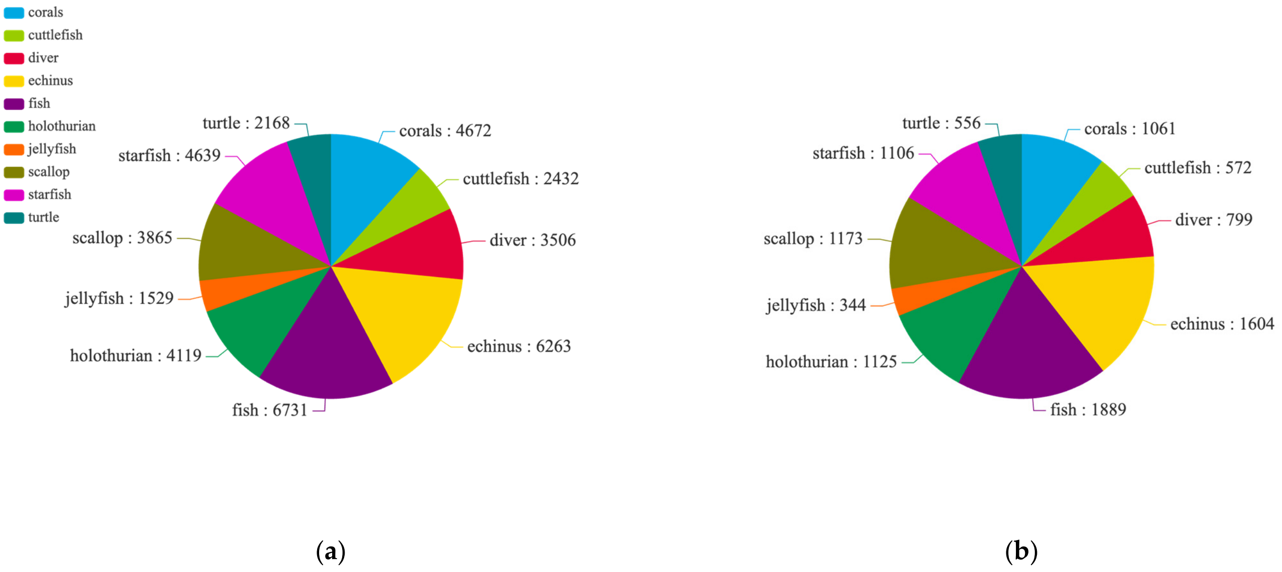


Figure 5. RUOD dataset distribution: (**a**) Number of samples per category in training set. (**b**) Number of samples per category in test set.

The DUO dataset contains 7782 accurately labeled images, with 6671 used for training and 1111 used for testing. The dataset consists of four typical aquaculture species: sea urchins, sea cucumbers, scallops, and starfish. The images in the DUO dataset exhibit typical underwater characteristics such as color distortion, low contrast, uneven lighting, blurriness, and high noise, presenting challenges for accurately detecting different aquaculture species. Meanwhile, it reflects the real-world issues faced in marine environment target detection. The distribution of the data and the number of samples per category in the training and test sets are showcased in Figure 6.

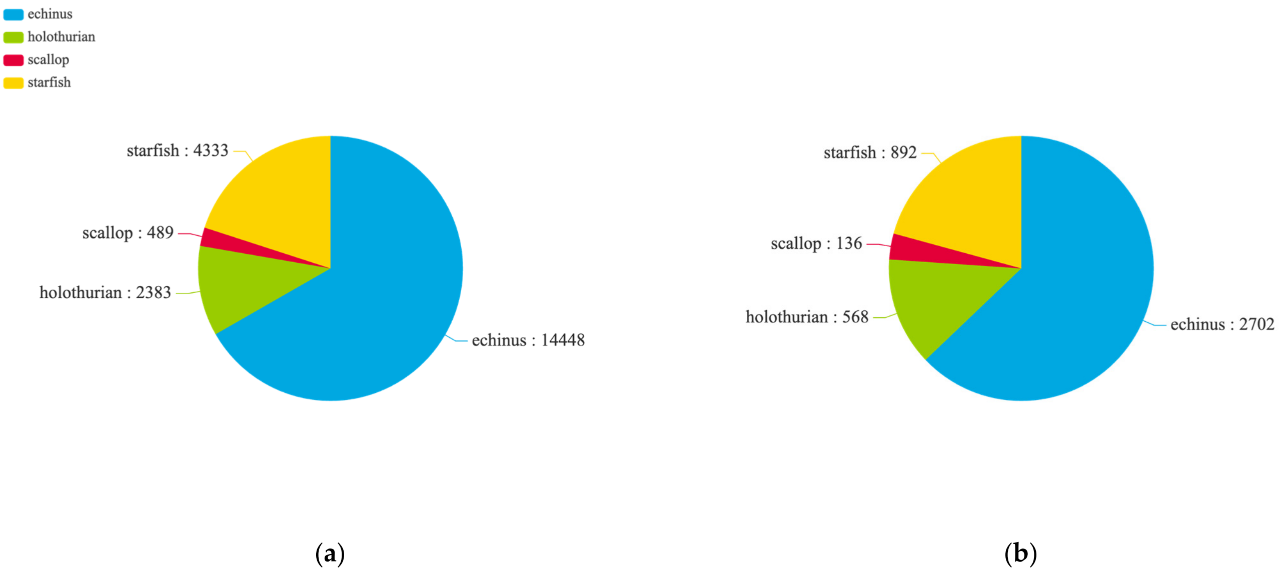


Figure 6. DUO dataset distribution: (**a**) Number of samples per category in training set. (**b**) Number of samples per category in test set.

3.2. Experimental Setup

In terms of training settings, the model undergoes 100 epochs, with each batch processing 16 images of size 640 × 640 pixels. For preventing overfitting, an early stopping mechanism is configured. Training is halted if there is no significant improvement within 50 epochs. Additionally, the initial learning rate is set to 0.01, and optimization is conducted using a momentum optimizer with weight decay to ensure training stability and effectiveness. For further enhancing training efficiency, automatic mixed Precision training is employed, significantly reducing computational resource usage without sacrificing accuracy.

For data augmentation, varIoUs image processing techniques are applied to enhance the robustness of model in different environments. In data augmentation, the probability of horizontal flipping is set to 50%, which can help the model maintain high recognition rates for objects facing different directions. The hue, saturation, and brightness enhancement ratios are set to 0.015, 0.7, and 0.4, respectively, which helps the model adapt to different lighting conditions and color variations. The configuration also allows for turning off mosaic augmentation during the final 10 epochs of training to avoid negative impacts on model performance from excessive image distortion.

During the inference phase, multiple optimization options are available, including a cosine learning rate scheduler to fine-tune model parameter adjustments. The training is conducted using four RTX 4090 GPUs, each with 24 GB of memory, and the training is carried out concurrently through the Distributed Data Parallel (DDP) mode.

3.3. Evaluation Metrics

For the evaluation of the algorithm, four metrics, mAP@50, mAP@50:95, Precision–Recall, and P-R Curve, are selected to comprehensively assess the accuracy of the improved model.

(1) Precision: Precision represents the proportion of actual positive samples among those predicted as positive. The formula can be expressed as:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{17}$$

where TP (true positive) denotes the number of true positives and FP (false positive) denotes the number of false positives.

(2) Recall: Recall indicates the proportion of actual positive samples that are correctly predicted as positive. The formula can be expressed as:

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{18}$$

where FN (false negative) represents the number of false negatives. Generally, Precision and Recall are somewhat in conflict. Increasing Precision tends to decrease Recall, and vice versa.

(3) mAP (mean Average Precision) is a commonly used computer vision evaluation metric that measures the performance of object detection algorithms. After the introduction of Precision and Recall, AP (Average Precision) needs to be understood. AP represents the Average Precision at a confidence threshold, which is the area under the Precision and Recall curve. To calculate AP, Precision and Recall are plotted as curves, and the area under the curve is computed. In practice, AP is often calculated via a discretization method, where Recall values are fixed at 0, 0.1, 0.2, . . ., 1. For each Recall value, the maximum Precision value is determined. Then, each Precision value is multiplied by the difference in Recall values between points, and all these products are summed to obtain AP.

mAP, as the average AP across multiple confidence thresholds, is about calculating the average of several AP values. In practice, confidence thresholds tend to start at 0.5 and increase in steps of 0.05 or 0.1. AP is calculated for each confidence threshold. Furthermore, the average of all AP values is taken as mAP. mAP is an important metric for evaluating the performance of object detection algorithms, as it can comprehensively consider the Precision and Recall performance of the algorithm across different confidence thresholds and provide a complete reflection of the algorithm's performance.

3.4. Ablation Experiments

In FishDet-YOLO, the UEM is introduced to enhance the gradient flow information in images. The UEM incorporates the UDAB module, which integrates the DAB module to extract high-frequency information and the LFAB module to obtain low-frequency information. In the detection network architecture, the Mamba-C2f is designed to enhance the algorithm's ability to capture long-range dependencies. In addition, ablation experiments are conducted to validate the effectiveness of each innovative module. Table 1 presents the ablation experiment results of FishDet-YOLO on RUOD. According to the results, enabling the DAB module or LFAB module individually improves the model's performance, with mAP50 reaching 83.8% and 83.1%, respectively, and corresponding improvements in Precision and Recall. When both the DAB and LFAB modules are enabled simultaneously, the model's mAP50 further increases to 86.3%. After the Mamba-C2f is incorporated, all performance metrics of the model improve significantly, with the final mAP50 reaching 89.5%, mAP50–95 reaching 65.9%, and Precision and Recall at 87.2% and 82.3%, respectively.

Table 1. RUOD ablation experiments; in the experimental data, bold numbers represent the highest accuracy, √ indicates the use of the module, and x indicates the non-use of the module.

UEM		Mamba-C2f	mAP50 (%)	mAP50–95 (%)	Precision	Recall
DAB	LFAB					
x	x	x	81.5	55.2	82.6	74.4
x	x	√	84.2	58.0	84.9	77.2
x	√	x	83.1	56.8	83.7	75.9
√	x	x	83.8	57.5	84.4	76.5
√	√	x	86.3	61.0	86.1	79.1
√	√	√	**89.5**	**65.9**	**87.2**	**82.3**

Table 2 shows the ablation experiment results on the DUO dataset. The baseline model has an mAP50 of 80.6%, an mAP50–95 of 60.2%, a Precision of 82.0%, and a Recall of 73.0%. After the introduction of the Mamba-C2f, mAP50 and mAP50–95 increase to 82.7% and 63.3%, respectively, with Precision and Recall reaching 83.6% and 75.2%. The addition of the LFAB module further increases mAP50 to 84.4% and mAP50–95 to 66.4%, with extra improvements in Precision and Recall. The model with the DAB module shows mAP50 at 83.9% and mAP50–95 at 64.9%. Combining the DAB and LFAB modules, mAP50 rises to 86.1% and mAP50–95 reaches 68.2%, with Precision and Recall at 85.8% and 79.1%, respectively. Finally, the model incorporating all features achieves an mAP50 of 88.8%, an mAP50–95 of 69.4%, a Precision of 86.4%, and a Recall of 80.8%.

Table 2. DUD ablation experiments; in the experimental data, bold numbers represent the highest accuracy, √ indicates the use of the module, and x indicates the non-use of the module.

UEM		Mamba-C2f	mAP50 (%)	mAP50–95 (%)	Precision	Recall
DAB	LFAB					
x	x	x	80.6	60.2	82.0	73.0
x	x	√	82.7	63.3	83.6	75.2
x	√	x	84.4	66.4	85.1	77.6
√	x	x	83.9	64.9	84.2	76.7
√	√	x	86.1	68.2	85.8	79.1
√	√	√	**88.8**	**69.4**	**86.4**	**80.8**

As revealed by the results in Tables 1 and 2, the DAB and LFAB modules enrich the gradient flow information in images. The Mamba-C2f plays a crucial role in enhancing the model's ability to capture long-distance dependency features in complex scenes, significantly improving the performance of FishDet-YOLO in underwater fish detection tasks.

As shown in Tables 3 and 4, to examine the effect of Laplacian decomposition component count in the UEM on final accuracy, we conducted an ablation study analyzing the impact of decomposing the image into 2 to 5 components. The results indicate that the highest accuracy was achieved when the image was decomposed into four components.

Table 3. Ablation study of different Laplacian component counts on RUOD dataset. Bold numbers represent the highest accuracy.

Laplacian Component Count	mAP50 (%)	mAP50–95 (%)	Precision (%)	Recall (%)
2	88.8	65.3	86.4	82.0
3	89.1	65.5	86.9	82.1
4	**89.5**	**65.9**	**87.2**	**82.3**
5	89.3	65.8	87.0	82.2

Table 4. Ablation study of different Laplacian component counts on DUD dataset. Bold numbers represent the highest accuracy.

Laplacian Component Count	mAP50 (%)	mAP50–95 (%)	Precision (%)	Recall (%)
2	87.9	68.8	86.0	80.3
3	88.5	69.2	86.1	80.6
4	**88.8**	**69.4**	**86.4**	**80.8**
5	88.6	69.1	86.3	80.5

As shown in Tables 5 and 6, we studied the impact of different loss functions on model performance. The experimental results show that using VFL Loss, DFL Loss, and CIoU Loss together achieves the highest detection accuracy.

Table 5. An ablation study of different loss functions on the RUOD dataset. Cls Loss represents classification loss, Reg Loss represents regression loss, CE Loss represents the cross-entropy loss, and L1 Loss represents the mean absolute error Loss. Bold numbers represent the highest accuracy.

Cls Loss	Reg Loss	mAP50 (%)	mAP50–95 (%)	Precision (%)	Recall (%)
CE Loss	L1 Loss	84.4	61.6	83.5	77.9
VFL Loss	L1 Loss	85.7	62.4	84.2	78.6
DFL Loss	L1 Loss	86.3	63.1	85.7	79.2
CE Loss	CIoU Loss	87.8	64.8	86.7	80.4
VFL Loss + DFL Loss	L1 Loss	87.2	64.1	86.0	80.1
VFL Loss + DFL Loss	CIoU Loss	**89.5**	**65.9**	**87.2**	**82.3**

Table 6. An ablation study of different loss functions on the DUD dataset. Cls Loss represents classification loss, Reg Loss represents regression loss, CE Loss represents the cross-entropy loss, and L1 Loss represents the mean absolute error Loss. Bold numbers represent the highest accuracy.

Cls Loss	Reg Loss	mAP50 (%)	mAP50–95 (%)	Precision (%)	Recall (%)
CE Loss	L1 Loss	84.4	61.6	83.5	77.9
VFL Loss	L1 Loss	85.2	63.1	84.6	78.5
DFL Loss	L1 Loss	86.0	62.8	85.3	79.0
CE Loss	CIoU Loss	87.1	65.0	85.2	80.2
VFL Loss + DFL Loss	L1 Loss	86.9	66.2	85.8	80.4
VFL Loss + DFL Loss	CIoU Loss	**88.8**	**69.4**	**86.4**	**80.8**

3.5. Comparative Experiments

This paper introduces FishDet-YOLO and compares it with 10 target detection algorithms, including the classic SSD [11] and Faster R-CNN [10], as well as recent versions of the YOLO series models, YOLOv8, YOLOv9 [15], and YOLOv10 [16], for evaluating its advantages. Additionally, this paper includes Transformer-based target detection algorithms such as DETR [21] and Deformable-DETR [22], two state-of-the-art (SOTA) algorithms from the past two years, YOLO-Fish [23] and UODN [44], as well as the latest fish detection algorithm of 2024, DP-FishNet [24].

Table 7 shows the comparison results on the RUOD dataset. It can be observed from the experimental results that FishDet-YOLO performs excellently across all metrics, especially in the key indicators of mAP50 and mAP50–95, achieving 89.5% and 65.9%, respectively, significantly higher than other models. Compared with the classic SSD and Faster R-CNN, FishDet-YOLO improves mAP50 by 17.2% and 14.3%, respectively, indicating superior overall performance in accuracy and Recall. Compared with the latest YOLO series versions, FishDet-YOLO also demonstrates clear advantages. Compared with the latest YOLOv10, FishDet-YOLO improves mAP50 and mAP50–95 by 6.7% and 6.4%, respectively, further validating the effectiveness of the UEM and Mamba-C2fs introduced in our network architecture. Moreover, FishDet-YOLO excels in both Precision and Recall, achieving 87.2%

and 82.3%, surpassing the most advanced YOLO models and Transformer-based DETR series. In the context of SOTA algorithms for underwater fish detection tasks, especially in comparison with UODN, YOLO-Fish, and DP-FishNet, FishDet-YOLO has maintained significant performance improvements, surpassing these three SOTA algorithms for fish detection and demonstrating its exceptional performance in complex fish detection tasks.

Table 7. Comparative experiments on the RUOD dataset, with bold text in the experimental data representing the highest Precision. Bold numbers represent the highest accuracy.

Method	mAP50 (%)	mAP50–95 (%)	Precision (%)	Recall (%)
SSD [11]	72.3	45.1	78.2	68.5
Faster R-CNN [10]	75.2	47.3	80.7	71.8
YOLOv8	81.5	55.2	82.6	74.4
YOLOv9 [15]	82.3	58.7	83.9	75.1
YOLOv10 [16]	82.8	59.5	84.8	75.2
DETR [21]	77.6	50.4	79.3	70.9
Deformable-DETR [22]	82.1	56.3	83.0	75.1
YOLO-Fish [23]	84.9	58.9	84.0	77.3
DP-FishNet [24]	86.2	61.5	85.0	78.3
UODN [44]	86.7	62.1	85.5	79.4
FishDet-YOLO	**89.5**	**65.9**	**87.2**	**82.3**

In Table 8, SSD and Faster R-CNN show relatively low performance on the DUO dataset, with mAP50 scores of 68.9% and 73.5%, respectively. The YOLO series models perform exceptionally well, with YOLOv8 achieving an mAP50 of 80.6%, while YOLOv9 and YOLOv10 further improve to 82.2% and 84.1%, respectively. DETR and its variants have certain advantages in complex scenarios but still lag behind the YOLO series. YOLO-Fish has an mAP50 of 85.7%, UODN scores 87.1%, and FishDet-YOLO further increases to 88.8%. As indicated by these results, FishDet-YOLO significantly improves Precision and Recall in fish detection tasks, demonstrating its substantial advantage in complex underwater scenes.

Table 8. Comparative experiments on the DUO dataset, with bold text in the experimental data representing the highest Precision. Bold numbers represent the highest accuracy.

Method	mAP50 (%)	mAP50–95 (%)	Precision	Recall
SSD [11]	68.9	42.7	75.4	64.2
Faster R-CNN [10]	73.5	48.1	78.9	69.3
YOLOv8	80.6	60.2	82.0	73.0
YOLOv9 [15]	82.2	62.8	83.2	74.4
YOLOv10 [16]	84.1	65.7	84.5	76.2
DETR [21]	76.4	53.4	78.7	70.1
Deformable-DETR [22]	80.3	58.6	81.4	72.8
YOLO-Fish [23]	85.7	67.5	84.8	78.0
DP-FishNet [24]	86.3	68.2	84.9	78.8
UODN [44]	86.1	68.3	85.4	79.1
FishDet-YOLO	**88.8**	**69.4**	**86.4**	**80.8**

Table 9 shows the performance comparison with the comparison algorithm, including the number of parameters, FLop. According to the results, the algorithm in this paper surpasses the detection algorithm designed by the Transformer in terms of performance, and it also surpasses DP-FishNet and UODN. In terms of comprehensive accuracy and speed, Fish Det-YOLO achieves superior results. However, it is slightly lower than the YOLOv8 algorithm, which is also the direction for future optimization.

Table 9. Performance metrics comparison. The bold text represents the algorithm proposed in this paper.

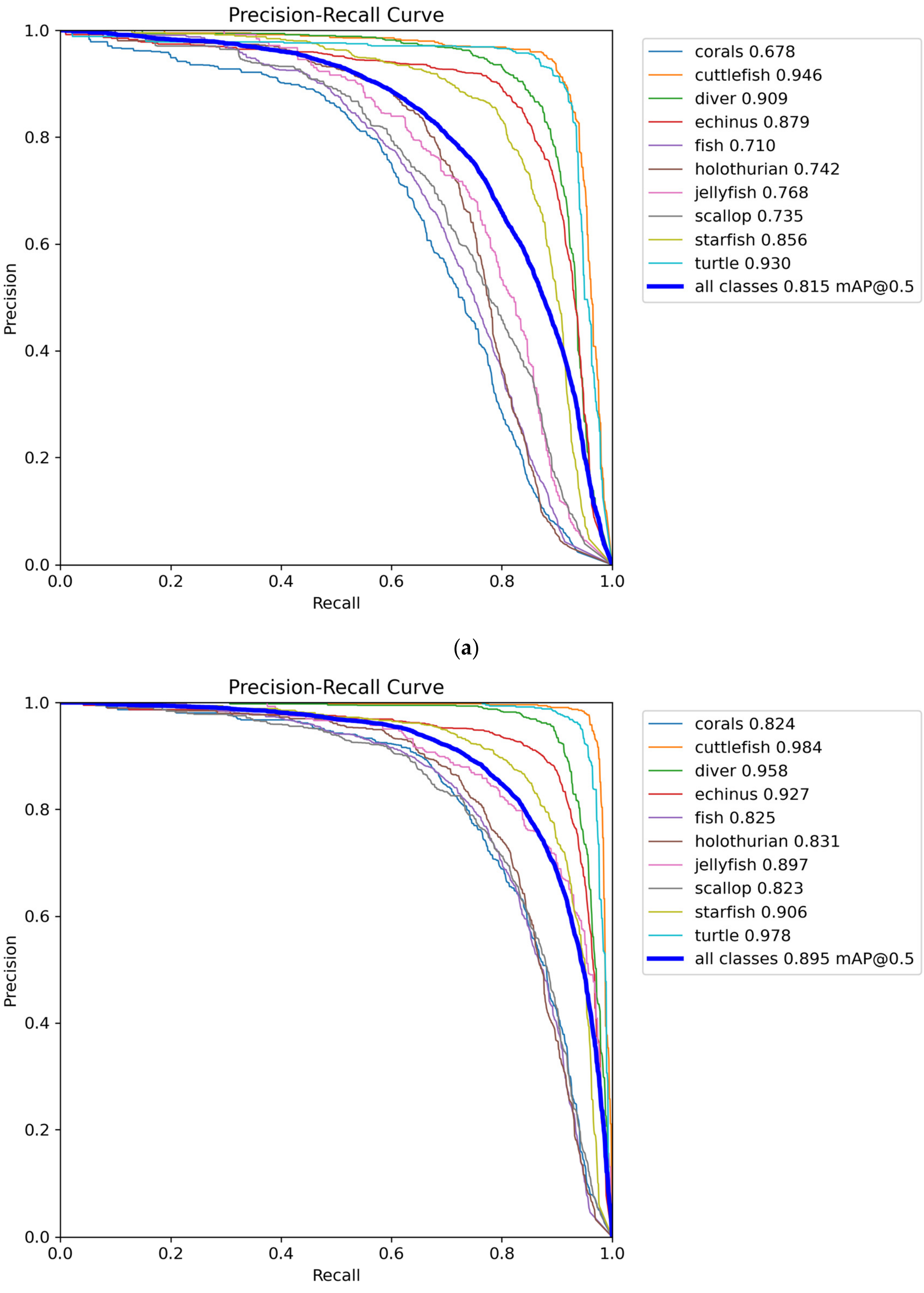
Method	Params	Flops	FPS
SSD [11]	71.6 M	87.5 G	22
Faster R-CNN [10]	123.6 M	213.3 G	7
YOLOv8	5.2 M	8.7 G	169
YOLOv9 [15]	7.1 M	26.4 G	114
YOLOv10 [16]	7.2 M	21.6 G	125
DETR [21]	41.3 M	86.4 G	28
Deformable-DETR [22]	40.3 M	173.6 G	19
YOLO-Fish [23]	86.0 M	201.2 G	8
DP-FishNet [24]	27.5 M	102.5 G	31
UODN [44]	19 M	43.9 G	123
FishDet-YOLO	6.8 M	19.5 G	143

3.6. Visual Qualitative Analysis Experiments

In Figure 7, the P-R Curve shows the experimental results of the baseline model YOLOv8 and the proposed improved algorithm FishDet-YOLO on the RUOD test set. As shown by the figure, the mAP for each fish species clearly presents the optimization trend. The improved algorithm increases the mAP from 81.5% to 89.5%, a significant increase of 8%. The AP for the corals category improves from 0.678 to 0.824, a 14.6% increase, demonstrating notable improvement in coral detection. The AP for cuttlefish increases from 0.946 to 0.984. In spite of the smaller increase (3.8%), it still indicates enhanced accuracy in cuttlefish recognition. The AP for the diver category rises from 0.909 to 0.958, an increase of 4.9%, showing more precise performance in fish detection. The AP for echinus improved from 0.879 to 0.927, a 4.8% increase, indicating significant improvement in sea urchin detection. Other categories, such as fish, holothurian, jellyfish, scallop, starfish, and turtle, also see increases in AP of 11.5, 8.9, 12.9, 8.8, 5.0, and 4.8%, respectively, further proving the effectiveness of the improved algorithm across varIoUs object detection tasks.

Figure 8 showcases the results of FishDet-YOLO on the DUO test set. Significant improvements in AP (Average Precision) are achieved across all categories, suggesting a comprehensive enhancement in detection performance. The AP for holothurian increases from 0.779 to 0.887, a rise of 11.4 percentage points, revealing a significant improvement in accuracy for sea cucumber detection. The AP for echinus rises from 0.915 to 0.935, an increase of 2.0 percentage points, reflecting stable performance improvements in sea urchin detection. The AP for scallop surges from 0.617 to 0.786, a 16.9 percentage point increase, highlighting a notable improvement in scallop detection performance. The AP for starfish improves from 0.912 to 0.943, a gain of 3.1 percentage points, further verifying that the algorithm's accuracy in starfish detection is enhanced. Overall, these results reflect positive progress in the detection performance of varIoUs target categories with the improved algorithm.

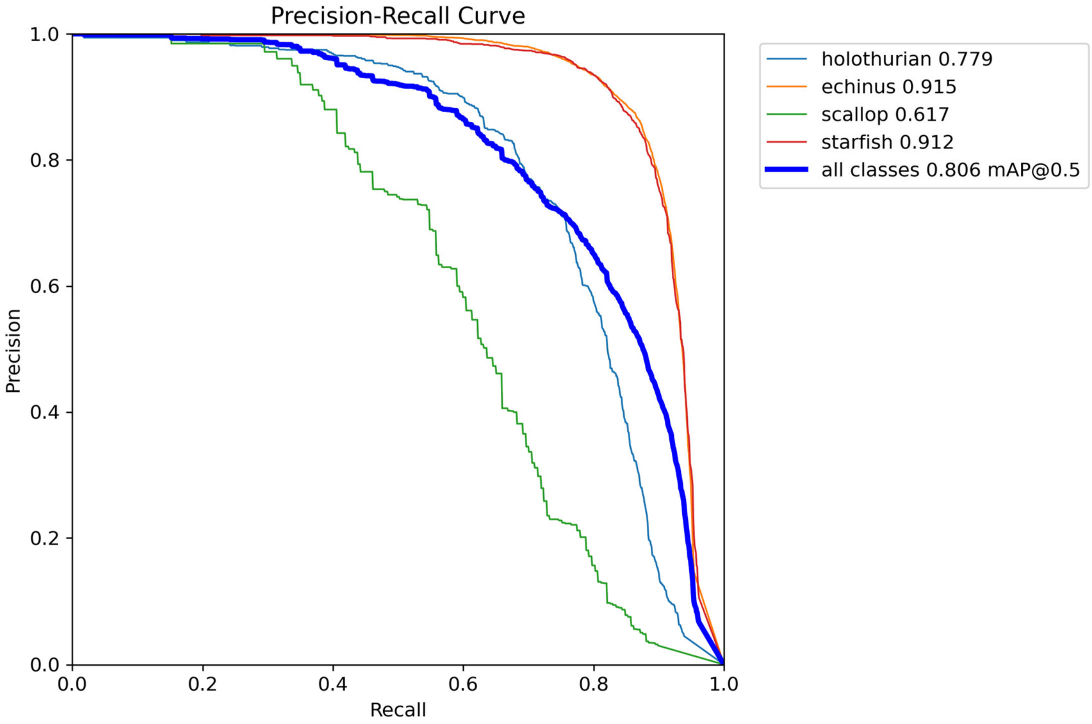
In Figures 9 and 10, a visual comparison of the detection results is given. Evidently, the model before improvement had certain issues with false positives and missed detection, particularly in cases where marine organisms were overlapped, triggering lower accuracy. In contrast, the FishDet-YOLO model proposed in this paper shows detection results much closer to the ground truth, demonstrating higher accuracy.



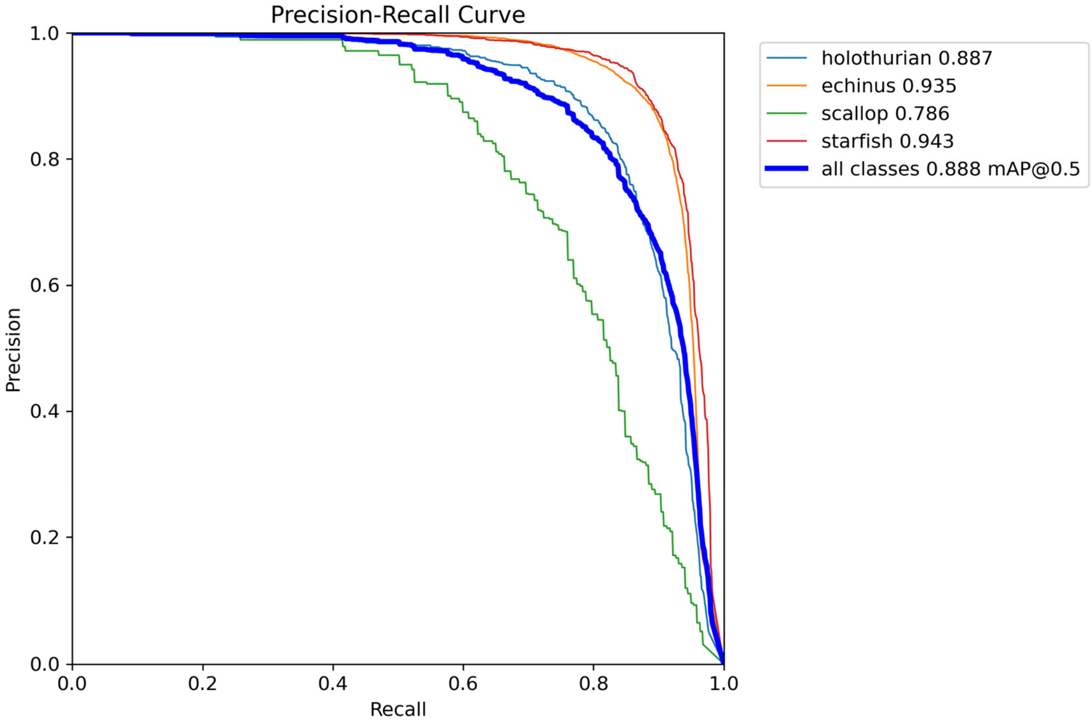
(**a**)

(**b**)

Figure 7. P-R Curves on the RUOD dataset for YOLOv8 and FishDet-YOLO. (**a**) P-R Curve for YOLOv8; (**b**) P-R Curve for FishDet-YOLO.



(**a**)



(**b**)

Figure 8. P-R Curves on the DUO dataset for YOLOv8 and FishDet-YOLO. (**a**) P-R Curve for YOLOv8; (**b**) P-R Curve for FishDet-YOLO.

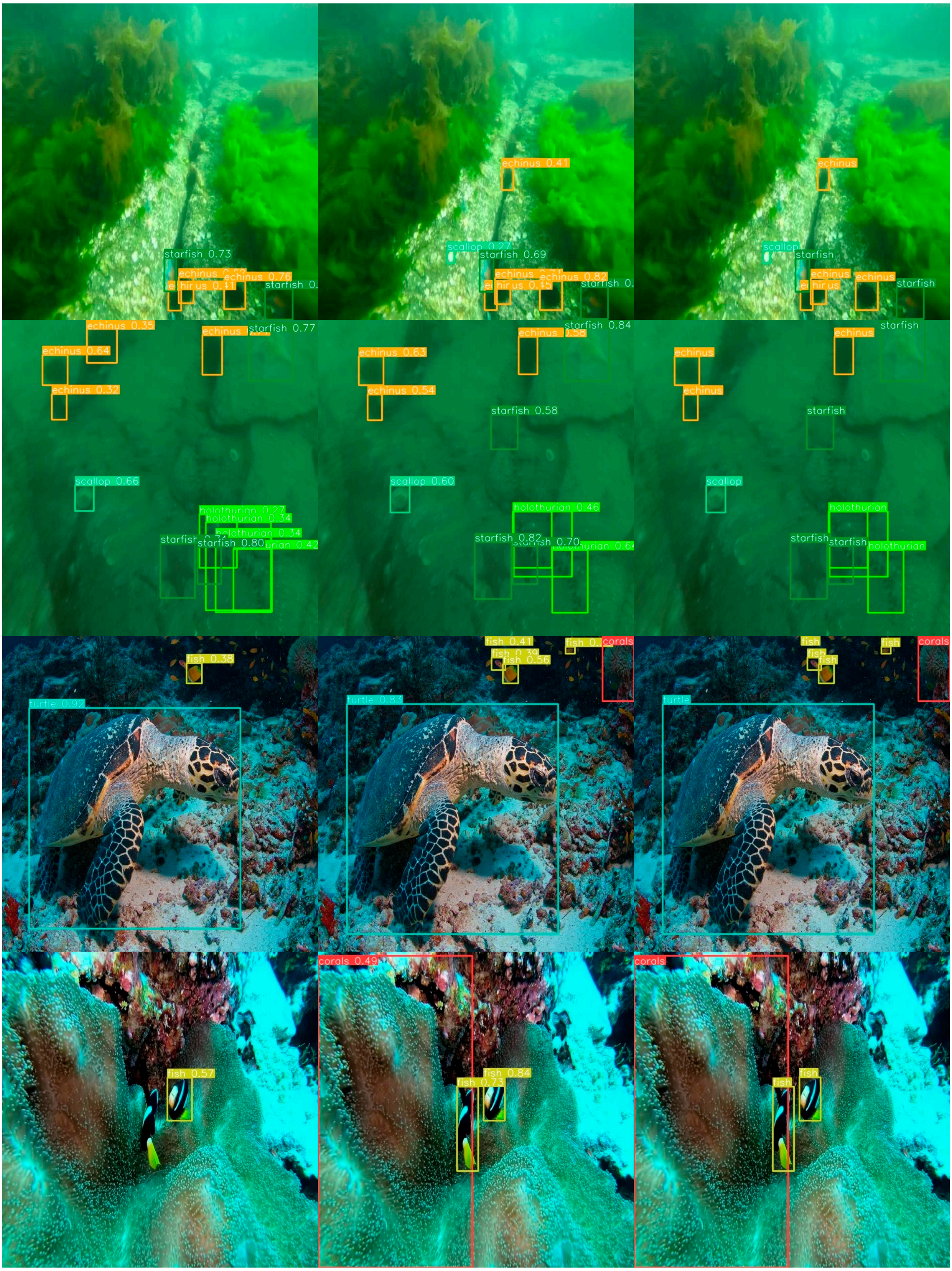
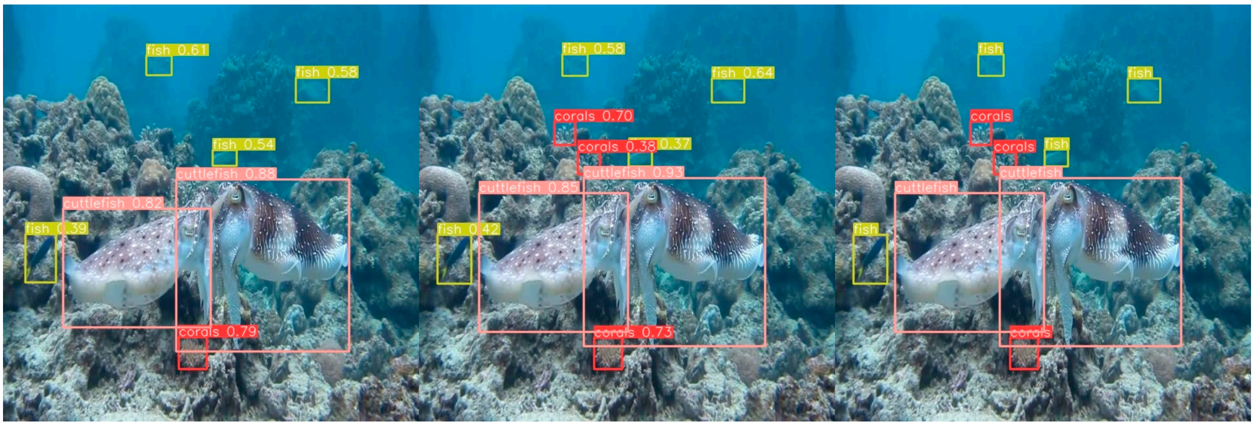
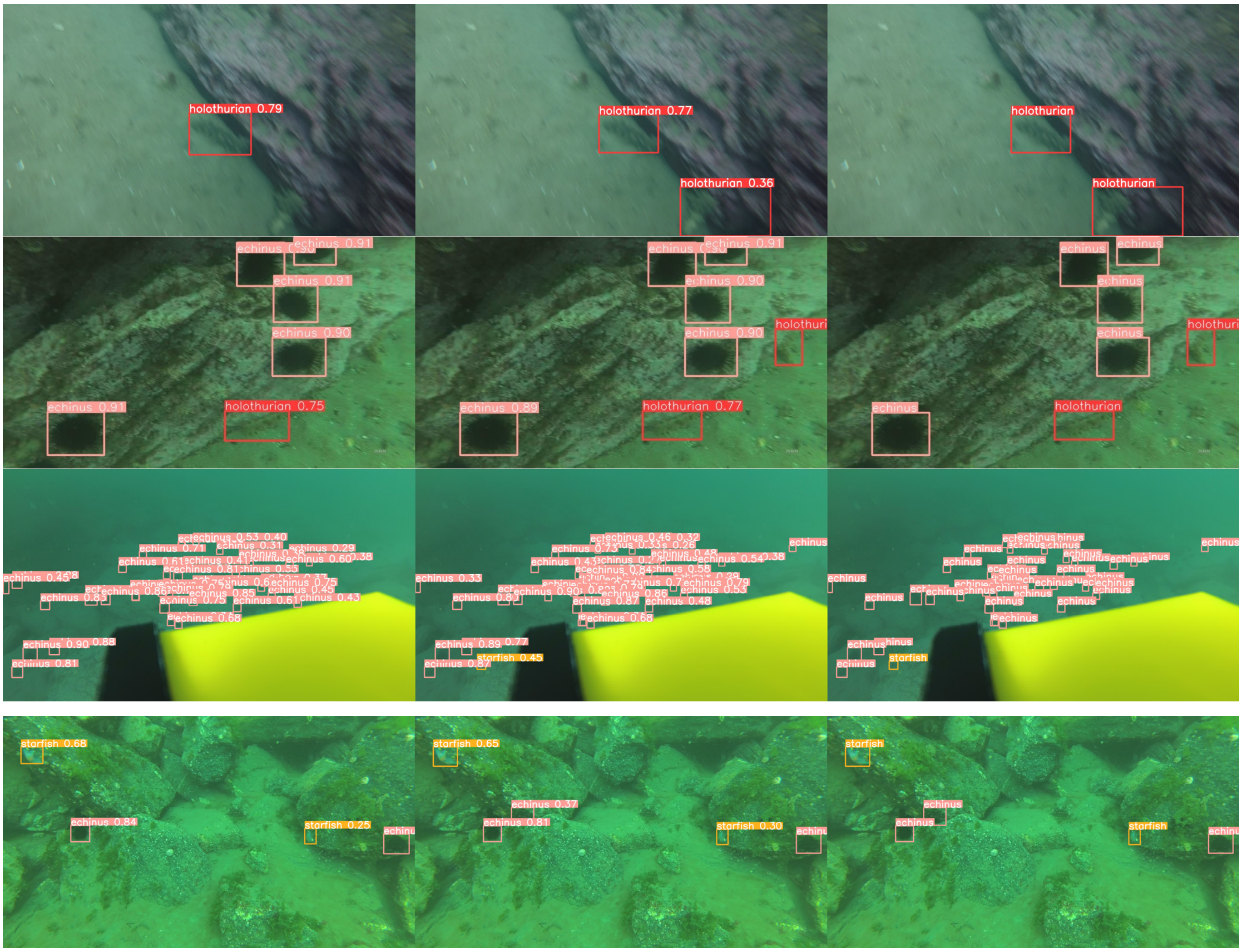


Figure 9. *Cont.*



(**a**) (**b**) (**c**)

Figure 9. Detection visualization results on the RUOD dataset. (**a**) YOLOv8 detection results; (**b**) FishDet-YOLO detection results; (**c**) ground truth.



(**a**) (**b**) (**c**)

Figure 10. Detection visualization results on the DUO dataset. (**a**) YOLOv8 detection results; (**b**) FishDet-YOLO detection results; (**c**) ground truth.

In addition to detection visualization, this study visualizes the output of the UEM to further investigate how the UEM enhances gradient flow information. Figure 11 shows the results of a comparison between the UEM network output and the original input image. From the perspective of image clarity, the UEM output exhibits higher image sharpness. With increased image clarity, the content of high-frequency information in the image is richer, and the gradient flow information is also more abundant. From the dimensions of color and saturation, the UEM output provides a better visual experience. This further explains the mechanism of the UEM.

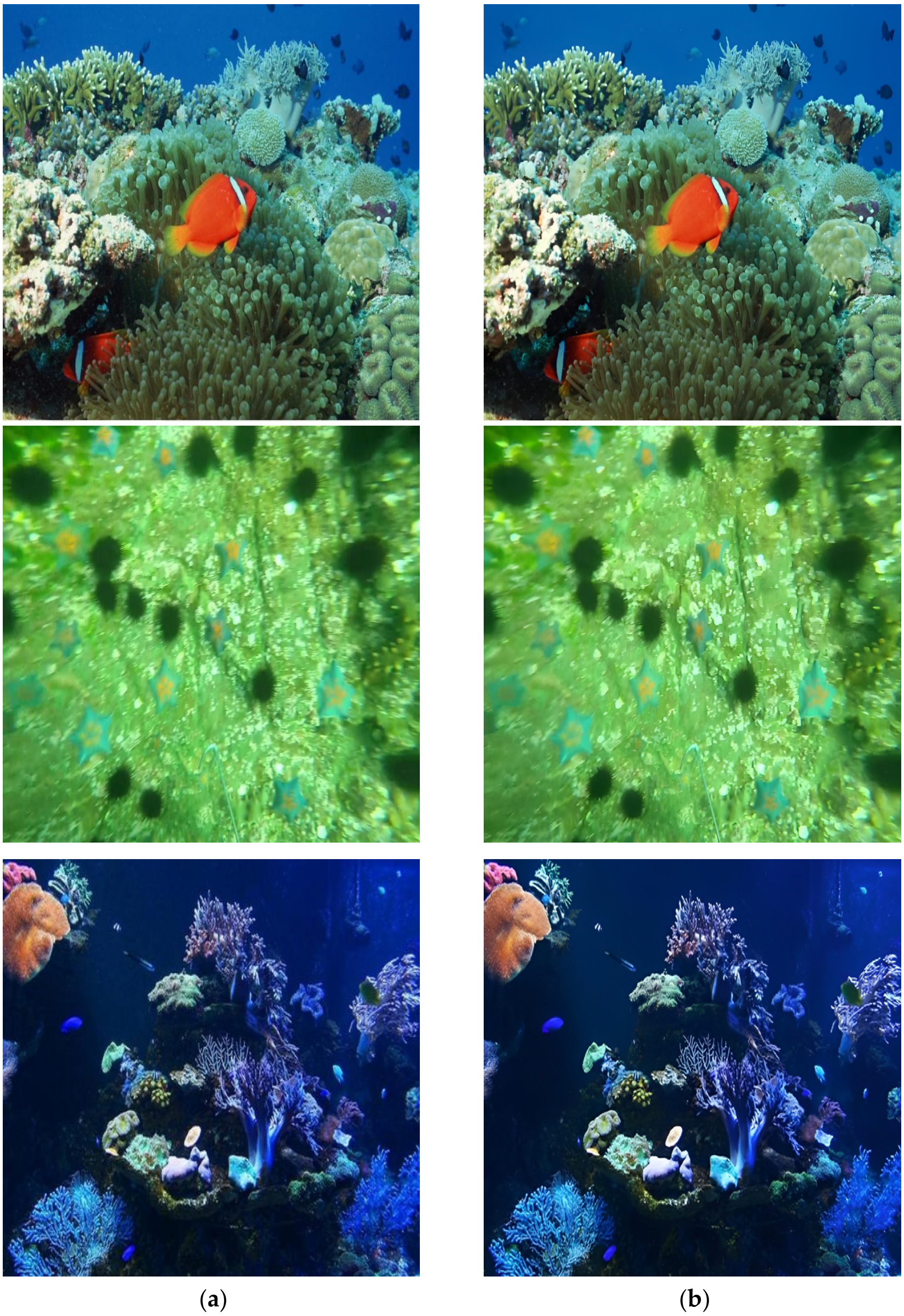


Figure 11. Visualization comparison of UEM output: (**a**) original image; (**b**) UEM output.

4. Discussion

4.1. Findings

In this paper, a novel underwater fish detection algorithm, FishDet-YOLO, is proposed. It significantly enhances gradient flow information in images by introducing the UEM and optimizing high-frequency and low-frequency information extraction with the UDAB module. The Mamba-C2f is designed within the detection network to improve the model's ability to capture long-range dependencies in complex scenes.

Systematic ablation experiments validate the effectiveness of each innovative module. The results reveal that the DAB and LFAB submodules in the UEM module are crucial for improving FishDet-YOLO's performance. According to the ablation experiments on the RUOD dataset, activating either the DAB or LFAB module alone significantly improves the performance of the model, with mAP50 reaching 83.8% and 83.1%, respectively. When both the DAB and LFAB modules are activated, mAP50 further increases to 86.3%. After introducing the Mamba-C2f, the model's mAP50 reaches 89.5%, and mAP50–95 increases to 65.9%, indicating that the Mamba-C2f plays a key role in capturing long-range dependency features. In ablation experiments on the DUO dataset, FishDet-YOLO also demonstrates strong performance advantages. After adding the Mamba-C2f, the model's mAP50 increases to 82.7%. With the introduction of LFAB and DAB modules, mAP50 reaches 84.4% and 83.9%, respectively. When both DAB and LFAB modules work together, mAP50 further improves to 86.1%. Ultimately, when all modules are integrated, the model's mAP50 reaches 88.8% and mAP50–95 reaches 69.4%, fully validating the effectiveness of these modules in enhancing model performance.

In comparative experiments with other object detection algorithms, FishDet-YOLO presents superior performance. On the RUOD dataset, FishDet-YOLO achieves mAP50 and mAP50–95 of 89.5% and 65.9%, respectively, significantly surpassing varIoUs advanced models, including the latest YOLOv10 and Transformer-based detection algorithms. The results reveal that FishDet-YOLO not only has a significant advantage in detection accuracy but also excels in handling long-range dependency features in complex underwater environments. On the DUO dataset, FishDet-YOLO also shows strong competitiveness, with a final mAP50 of 88.8%, the best among all compared algorithms. This further validates the effectiveness of the proposed UEM and Mamba-C2fs, especially in handling complex underwater scenes, where these modules significantly enhance the detection capability and robustness of the algorithm.

In the visual qualitative analysis, the results presented through the P-R Curves further support these conclusions. FishDet-YOLO demonstrates remarkable AP improvements in varIoUs fish categories, such as corals, cuttlefish, and scallop. These improvements not only validate the algorithm's effectiveness in varIoUs object detection tasks but also indicate that FishDet-YOLO has excellent application potential in underwater fish detection. Finally, the effectiveness of detection before and after the improvements is demonstrated through visual comparisons. Additionally, the UEM's inputs and outputs are compared to explain the functioning of the UEM.

In terms of model efficiency, FishDet-YOLO demonstrates superior performance with 6.8 M parameters, 19.5 G FLOPs, and a processing speed of 143 FPS. Compared to Transformer-based methods, such as YOLO-Fish (86.0 M parameters, 201.2 G FLOPs, and 8 FPS) and Deformable-DETR (40.3 M parameters, 173.6 G FLOPs, and 19 FPS), FishDet-YOLO offers a more efficient use of computational resources and faster inference speed. This efficiency allows FishDet-YOLO to maintain high detection accuracy while optimizing computational performance, making it a robust solution for practical applications.

According to the experimental results, the design in FishDet-YOLO plays a significant role in addressing the challenges of complex scenes. We hope to provide new perspectives and references for underwater fish detection research.

4.2. Limitations and Future Works

Although FishDet-YOLO achieves significant performance improvements on the RUOD and DUO datasets, there are still some limitations. Firstly, the complexity of the model is high, triggering increased computational demands, which may require substantial computational resources in practical applications. Additionally, the generalization ability of the model still needs further validation when applied to other types of underwater images or fish detection in different environments. Moreover, underwater environments present varIoUs challenges, such as low-light conditions and turbidity, which require further investigation and validation. Lastly, since the model will be deployed on devices with limited computational power in practical applications, the complexity of the model under these constraints must be considered.

In future work, it will be essential to focus on the following areas: First, the model's computational efficiency should be further optimized to reduce redundant calculations, so as to enable its effective operation in resource-constrained environments. Secondly, more universally applicable enhancement methods can be explored to improve the model's generalization capabilities across varIoUs scenarios. Additionally, FishDet-YOLO can be applied to a broader range of underwater biological detection tasks to verify its effectiveness in diverse application contexts. At the same time, detection accuracy and robustness could be further enhanced by incorporating more multimodal data (e.g., sonar or laser scanning data) and integrating them with existing visual information robustness.

5. Conclusions

To summarize, we introduced the FishDet-YOLO algorithm to address the challenges posed by the complexity of underwater environments and the diversity of fish species, with the aim of enhancing both the accuracy and efficiency of underwater fish detection. First, to mitigate issues such as uneven lighting, color distortion, and reduced contrast in underwater environments, we developed the Underwater Enhancement Module (UEM). This module utilizes Laplacian pyramids and multi-scale fusion techniques to enhance high-frequency information in images, enrich gradient flow information, and enable end-to-end joint training with the YOLO algorithm, thereby significantly improving detection performance. Second, to address the complexity of varIoUs fish species, we leveraged the long-range dependency capabilities of the Mamba model and integrated them with YOLO's C2f to create the Mamba-C2f. This approach effectively improves the accuracy and adaptability of fish detection.

Despite its strong performance on the RUOD and DUO datasets, the FishDet-YOLO algorithm has several limitations. Firstly, the model's high complexity results in substantial computational resource requirements, which may limit its applicability in real-time scenarios. Secondly, although the model demonstrates good performance on specific datasets, its generalization ability across diverse underwater environments and fish species requires further validation. Additionally, the model may lack robustness in extreme environmental conditions or with images of poor quality.

Future research should focus on the following directions: (1) optimizing the model's computational efficiency to reduce resource demands and enhance real-time applicability; (2) improving the model's generalization capability by expanding the training dataset and adopting more universal feature extraction methods to enhance performance across varied underwater environments and fish species; (3) exploring the model's potential applications in other underwater biological detection tasks, such as monitoring different marine organisms; and (4) increasing the model's robustness to variations in image quality under challenging environmental conditions to ensure consistent performance. Through these efforts, we anticipate that FishDet-YOLO will offer more effective solutions and advance the field of underwater target detection.

Author Contributions: Conceptualization, C.Y. and J.X.; methodology, C.Y.; software, C.Y.; validation, C.Y., J.X. and X.L.; formal analysis, C.Y.; investigation, C.Y.; resources, C.Y.; data curation, C.Y.; writing—original draft preparation, C.Y.; writing—review and editing, C.Y. and J.X.; visualization, C.Y.; supervision, J.X.; project administration, J.X.; funding acquisition, Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The third-party data used in this paper include the RUOD dataset at https://github.com/dlut-dimt/RUOD (accessed on 15 July 2024) and the DUO dataset at https://github.com/chongweiliu/DUO (accessed on 15 July 2024), with the download links provided in this paper.

Conflicts of Interest: The authors declare no conflicts of interest regarding the publication of this paper. All data, analyses, and findings have been conducted and presented without any external influence or bias. Any affiliations or financial involvements that might be perceived as influencing the interpretation of data are transparently disclosed in the appropriate sections of the manuscript.

References

1. Alsmadi, M.K.; Almarashdeh, I. A survey on fish classification techniques. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 1625–1638. [CrossRef]
2. Cui, S.; Zhou, Y.; Wang, Y.; Zhai, L. Fish detection using deep learning. *Appl. Comput. Intell. Soft Comput.* **2020**, *2020*, 3738108. [CrossRef]
3. Jalal, A.; Salman, A.; Mian, A.; Shortis, M.; Shafait, F. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecol. Inform.* **2020**, *57*, 101088. [CrossRef]
4. Sung, M.; Yu, S.-C.; Girdhar, Y. Vision based real-time fish detection using convolutional neural network. In Proceedings of the OCEANS 2017-Aberdeen, Aberdeen, UK, 19–22 June 2017; pp. 1–6.
5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
6. Hong Khai, T.; Abdullah, S.N.H.S.; Hasan, M.K.; Tarmizi, A. Underwater fish detection and counting using mask regional convolutional neural network. *Water* **2022**, *14*, 222. [CrossRef]
7. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
8. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
9. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
10. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef]
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14. pp. 21–37.
12. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
13. Kaur, P.; Khehra, B.S.; Mavi, E.B.S. Data augmentation for object detection: A review. In Proceedings of the 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 9–11 August 2021; pp. 537–543.
14. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710.
15. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv* **2024**, arXiv:2402.13616.
16. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *arXiv* **2024**, arXiv:2405.14458.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
18. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
19. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10012–10022.

20. Xiao, Y.; Yuan, Q.; Jiang, K.; He, J.; Lin, C.-W.; Zhang, L. TTST: A top-k token selective transformer for remote sensing image super-resolution. *IEEE Trans. Image Process.* **2024**, *33*, 738–752. [CrossRef] [PubMed]
21. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
22. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
23. Al Muksit, A.; Hasan, F.; Emon, M.F.H.B.; Haque, M.R.; Anwary, A.R.; Shatabda, S. YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment. *Ecol. Inform.* **2022**, *72*, 101847. [CrossRef]
24. Shah, C.; Alaba, S.Y.; Nabi, M.; Prior, J.; Campbell, M.; Wallace, F.; Ball, J.E.; Moorhead, R. An enhanced YOLOv5 model for fish species recognition from underwater environments. In Proceedings of the Ocean Sensing and Monitoring XV, Orlando, FL, USA, 3–4 May 2023; pp. 238–246.
25. Liu, Y.; An, D.; Ren, Y.; Zhao, J.; Zhang, C.; Cheng, J.; Liu, J.; Wei, Y. DP-FishNet: Dual-path Pyramid Vision Transformer-based underwater fish detection network. *Expert Syst. Appl.* **2024**, *238*, 122018. [CrossRef]
26. Wang, Z.; Ruan, Z.; Chen, C. DyFish-DETR: Underwater Fish Image Recognition Based on Detection Transformer. *J. Mar. Sci. Eng.* **2024**, *12*, 864. [CrossRef]
27. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 568–578.
28. Huo, C.; Zhang, D.; Yang, H. An Underwater Image Denoising Method Based on High-Frequency Abrupt Signal Separation and Hybrid Attention Mechanism. *Sensors* **2024**, *24*, 4578. [CrossRef]
29. Liu, X.; Gu, Z.; Ding, H.; Zhang, M.; Wang, L. Underwater Image Super-Resolution Using Frequency-Domain Enhanced Attention Network. *IEEE Access* **2024**, *12*, 6136–6147. [CrossRef]
30. Jiang, K.; Wang, Q.; An, Z.; Wang, Z.; Zhang, C.; Lin, C.-W. Mutual retinex: Combining transformer and cnn for image enhancement. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 2240–2252. [CrossRef]
31. Gu, A.; Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv* **2023**, arXiv:2312.00752.
32. Zhu, L.; Liao, B.; Zhang, Q.; Wang, X.; Liu, W.; Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv* **2024**, arXiv:2401.09417.
33. Huang, T.; Pei, X.; You, S.; Wang, F.; Qian, C.; Xu, C. Localmamba: Visual state space model with windowed selective scan. *arXiv* **2024**, arXiv:2403.09338.
34. Xiao, Y.; Yuan, Q.; Jiang, K.; Chen, Y.; Zhang, Q.; Lin, C.-W. Frequency-Assisted Mamba for Remote Sensing Image Super-Resolution. *arXiv* **2024**, arXiv:2405.04964.
35. Xing, Z.; Ye, T.; Yang, Y.; Liu, G.; Zhu, L. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *arXiv* **2024**, arXiv:2401.13560.
36. Burt, P.J.; Adelson, E.H. The Laplacian pyramid as a compact image code. In *Readings in Computer Vision*; Elsevier: Amsterdam, The Netherlands, 1987; pp. 671–679.
37. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768.
38. Kanopoulos, N.; Vasanthavada, N.; Baker, R.L. Design of an image edge detection filter using the Sobel operator. *IEEE J. Solid-State Circuits* **1988**, *23*, 358–367. [CrossRef]
39. Zhang, H.; Wang, Y.; Dayoub, F.; Sunderhauf, N. Varifocalnet: An IoU-aware dense object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8514–8523.
40. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21002–21012.
41. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 12993–13000.
42. Fu, C.; Liu, R.; Fan, X.; Chen, P.; Fu, H.; Yuan, W.; Zhu, M.; Luo, Z. Rethinking general underwater object detection: Datasets, challenges, and solutions. *Neurocomputing* **2023**, *517*, 243–256. [CrossRef]
43. Liu, C.; Li, H.; Wang, S.; Zhu, M.; Wang, D.; Fan, X.; Wang, Z. A dataset and benchmark of underwater object detection for robot picking. In Proceedings of the 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Shenzhen, China, 5–9 July 2021; pp. 1–6.
44. Zhou, H.; Kong, M.; Yuan, H.; Pan, Y.; Wang, X.; Chen, R.; Lu, W.; Wang, R.; Yang, Q. Real-time underwater object detection technology for complex underwater environments based on deep learning. *Ecol. Inform.* **2024**, *82*, 102680. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.