Efficient Underwater Garbage Detection Using GSConv Enhanced YOLOv8 with GD Mechanism

Xiaolong Fu [1], Xiaolong Gao [1], Zufeng Fu [1,2,*] , Zhuang Yang [1]
[1] School of Artificial Intelligence and Software Engineering, Nanyang Normal University, Nanyang 473061, China
[1] Collaborative Innovation Center of Intelligent Explosion-proof EquipmentHenan Province, Nanyang 473061, China
E-mail: xlfu123@163.com, xl_gao2025@163.com, fuzufeng@outlook.com, yz2242450789@163.com
* Corresponding Author

Keywords: Underwater garbage, Detection, YOLO-GGS, Gold-YOLO, GSConv

Received: July 20, 2025

Due to low visibility, scale variation, and complex backgrounds, detecting marine debris in underwater environments remains highly challenging. To address these issues, we propose YOLO-GGS, a lightweight yet high-performance object detector built upon YOLOv8. The framework incorporates three key innovations. First, the Gather-and-Distribute (GD) mechanism from Gold-YOLO is introduced into the neck, which unifies multi-scale feature aggregation while selectively injecting global context, thereby enhancing object perception across different scales. Second, GSConv-based hybrid convolutions are deployed in both the backbone and the injection module, effectively balancing rich channel interactions with reduced computational complexity. Third, a Slim-Neck design simplifies the feature fusion path by eliminating redundant operations, thus improving inference speed. Comprehensive experiments on the J-EDI and Brackish underwater datasets demonstrate the superior performance of YOLO-GGS, achieving mAP@0.5:0.95 values of 88.5% and 84.7%, which represent improvements of 4% and 2.5% over the baseline model, respectively. Moreover, real-time evaluation shows that YOLO-GGS reaches an inference speed of 108.4 FPS. These results highlight YOLO-GGS as an efficient and accurate solution for underwater debris detection, offering substantial potential for deployment on Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs).

Povzetek: Predstavljamo YOLO-GGS, lahek in visokozmogljiv model za zaznavanje podvodnih morskih odpadkov, ki s pomočjo mehanizma zbiranja in porazdelitve, hibridnih konvolucij GSConv in arhitekture Slim-Neck zagotavlja odlično večstopenjsko natančnost in zmogljivost v realnem času z bistveno manj parametri.

1 Introduction

Underwater garbage detection is crucial for marine environmental protection. It also plays an important role in maintaining ecological balance. Due to the increasing impact of human activities on the marine environment, marine litter has become a global concern [1]. Significant quantities of waste are introduced into the ocean from coastal zones, surface waters, the seabed, and various maritime sources [2]. Each year, approximately 19–23 million metric tons of plastic debris are discharged into aquatic environments. Without effective management, the volume of waste is expected to increase significantly by 2025, according to current forecasts [3]. Underwater garbage not only mars the marine landscape but also threatens marine life, disrupts ecosystems, and harms both human health and economic development. By monitoring and managing the marine environment through the detection of underwater litter, we can improve the sustainable use of marine resources[4]. According to [5, 6], underwater detection technology plays a vital role in identifying and managing marine debris. Regular detection allows for timely identification and removal of marine debris, minimizing ecological damage to marine life while safeguarding essential habitats such as coral reef systems and seagrass habitats. Additionally, underwater litter detection holds significant value in the military field. With technological advancements, monitoring systems applied to underwater garbage detection can now achieve fast and accurate results. This study presents YOLO-GSS, an enhanced YOLOv8[7]-based model designed for precise and efficient underwater trash detection and recognition. The proposed model reduces hardware costs and facilitates broader adoption. It can detect and promptly remove debris that may pose a threat to military equipment and operations. This helps ensure the smooth progress of

military activities at sea. This model also considers applications in ecological aquaculture, improving the accuracy of detecting underwater organisms. The proposed model reduces the model training cost, is more lightweight, and is conducive to wider adoption. The contributions of this research can be categorized as follows. First, the inclusion of GoldYOLO[8] in the head of YOLOv8 achieves significant performance improvements in target detection by introducing an innovative GD (Gather-and-Distribute) mechanism. Second, by incorporating the slim-neck3 structure and using GSConv[9], the number of parameters is reduced, leading to lower model complexity without compromising detection accuracy.

2 Related Work

The growing accumulation of marine debris poses severe threats to biodiversity, ecosystem stability, and human health [10]. Detecting underwater garbage is particularly challenging due to poor visibility, low contrast, and severe color distortion caused by light scattering and absorption, especially in deep-sea environments (Figure 8) [11]. High intra-class variability (e.g., plastic bags, fishing nets, cans), frequent occlusions, cluttered backgrounds, and visual similarity to marine organisms further increase false positives. Hazardous underwater conditions—high pressure, low temperature, and complex terrain—limit human exploration, leading to the deployment of Remotely Operated Vehicles (ROVs) [12] and Autonomous Underwater Vehicles (AUVs) [13], which enhance safety and data continuity but impose strict constraints on onboard computing resources. Consequently, robust yet lightweight target detection models capable of real-time inference without sacrificing accuracy are urgently needed [14]. Over time, target detection techniques have evolved from traditional methods (e.g., Viola-Jones, HOG, DPM) to deep learning-based approaches, including single-stage detectors (e.g., YOLO, SSD) and two-stage detectors (e.g., the R-CNN series), as summarized in Figure 1. These advances provide new opportunities to address the above challenges in practical underwater sensing systems for ecological protection and operational tasks.

Recent research has explored various strategies to overcome these challenges. Zhou et al.[11] introduced YOLOTrashCan, a lightweight detection network tailored for underwater debris. The model incorporates an ECA_DO-Conv_CSPDarknet53 backbone, which combines Efficient Channel Attention with Depthwise Over-parameterized Convolution to enhance semantic representation. In addition, it uses a DPMs_PixelShuffle_PANet for multi-scale feature fusion. Despite its compact size (214 MB), it achieves competitive accuracy on the TrashCan 1.0 dataset, demonstrating an effective balance of efficiency and performance under challenging conditions.

For underwater plastic waste detection, Teng et al.[4] proposed an enhanced YOLOv5-based framework optimized for AUV deployment. The improvements include anchor box refinement via a modified KMeans++ clustering algorithm. The bounding box loss was also replaced with Complete Intersection over Union (CIoU) loss, yielding more precise spatial overlap estimation and improved detection of diverse plastic debris.

To further advance detection performance, Ma et al.[3] developed MLDet, a specialized framework that outperforms conventional detectors such as Faster R-CNN and RetinaNet by 11.9–13.9 percentage points in mAP on the TrashCan-Material and TrashCan-Instance datasets. Key innovations include deformable convolutional networks for irregular debris modeling, adaptive training sample selection to improve robustness under occlusion, and a multi-task loss combining Quality Focal Loss and GIoU loss. MLDet achieves notable improvements, particularly for challenging debris categories such as plastic and rubber.

Zhu et al.[15] proposed YOLOv8-C2f-Faster-EMA, which improves the backbone, neck, and C2f modules. It also incorporates an attention mechanism to enhance small-scale underwater debris detection. Although it improves accuracy–efficiency trade-offs, it still struggles under complex backgrounds and low-visibility conditions. Similarly, Sarkar et al.[16] proposed U-YOLOv3, which integrates MIRNet for image enhancement and refines YOLOv3[17]. The refinements include optimizing anchor box sizes via K-means++, incorporating a Spatial Pyramid Pooling layer for multi-scale feature aggregation, and adding down- and upsampling pathways to enhance detection of both very small and large targets. Evaluated on Brackish and Trash-ICRA19[18] datasets, it improves mAP by 10% and 9% over YOLOv3. However, despite its enhanced scale adaptability, the computational complexity inherited from YOLOv3 limits its suitability for real-time AUV deployment.

To optimize detection precision in visually degraded environments, Zhao et al. [19] integrated super-resolution reconstruction (SRR) with a customized SFD-YOLO detector. Among seven SRR models, the RDN-based approach achieved the best results, reaching a mAP of 91.2% at a scale factor of 4. This demonstrates SRR's potential in mitigating optical distortions. Nonetheless, the added computational overhead of SRR poses challenges for real-time applications.

In summary, balancing detection accuracy and computational efficiency remains a central challenge in underwater debris detection. While recent approaches have improved accuracy, they often increase model complexity, limiting deployment on resource-constrained AUV platforms. Future work should focus on three key aspects: reducing computational over-

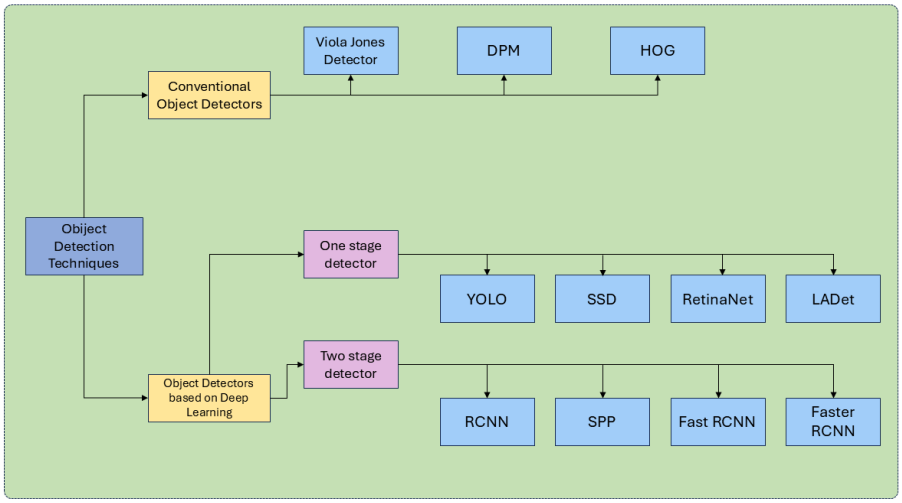


Figure 1: Traditional target detection algorithms and deep learning based target detection algorithms. Traditional target detection algorithms are Viola–Jones detector, DMP, HOG. Deep learning-based algorithms are categorized into single-stage and two-stage detectors. The representative single-stage ones are YOLO, SSD, and the representative two-stage ones are R-CNN, SPP, Fsat R-CNN, Faster R-CNN.

head while preserving precision, enhancing generalization across diverse underwater environments, and enabling real-time inference to meet practical operational demands. Main contribution of the article:

(1) This work proposes a detection algorithm specifically optimized for underwater trash detection, moving beyond conventional YOLO-based modifications. Built upon YOLOv8, the model is designed to handle low illumination, frequent occlusion, and diverse object scales, thereby achieving superior adaptability and robustness in complex underwater environments.

(2) To overcome scale variation and boundary ambiguity, we incorporate a Gather-and-Distribute (GD) mechanism into the neck. This design unifies multi-scale feature aggregation and selectively injects global context, enabling the model to capture both fine-grained and semantic features more effectively. Enhanced multi-scale fusion and global perception lead to improved recognition of objects with diverse sizes.

(3) We design an efficient architecture that integrates GSConv-based hybrid convolutions in both the backbone and the injection module, together with a Slim-Neck structure for streamlined feature fusion. This combination enriches channel interactions, reduces computational complexity, and removes redundant operations, thereby improving inference speed while maintaining high detection accuracy. The lightweight design makes the model particularly suitable for deployment on resource-constrained underwater or edge devices.

(4) We conduct extensive evaluations on both the J-EDI and Brackish Underwater datasets. YOLO-GGS achieves an mAP@0.5:0.95 of 88.5% on J-EDI, surpassing YOLOv8n by 4%. The cross-domain generalization experiment on Brackish Underwater dataset further demonstrates strong robustness to unseen environments. In addition, real-time evaluation confirms that the proposed model delivers the fastest inference (9.2 ms per frame, 108.4 FPS) with the lowest memory footprint (414 MB), highlighting its balanced trade-off among accuracy, efficiency, and practicality.

3 Improve strategy

In object detection, we reviewed several relevant studies, including the YOLO enhancements by Zhao et al.[20] and Cheng et al.[21]. Our improvement strategy leverages the GSConv module and Slim-neck structure from Gold-YOLO. Specifically, the YOLOv8 Head adopts Gold-YOLO's architectural design. The Backbone replaces standard convolutions with lightweight GSConv, enhancing feature extraction and improving model accuracy. Gold-YOLO also introduces a Gather-and-Distribute (GD) mechanism, which addresses the limitations of traditional Feature Pyramid Networks (FPN) [22] in information flow, improving detection across object scales. The GD mechanism comprises two branches: Low-Gather-and-Distribute (Low-GD) for small targets and High-Gather-and-Distribute (High-GD) for larger targets. This design effectively balances detection accuracy and speed, which is critical for real-time underwater debris detection.

3.1 YOLOv8

The YOLOv8 architecture (Figure 2) consists of four modules: Input, Backbone, Neck, and Head. The Input module uses Mosaic augmentation, while larger

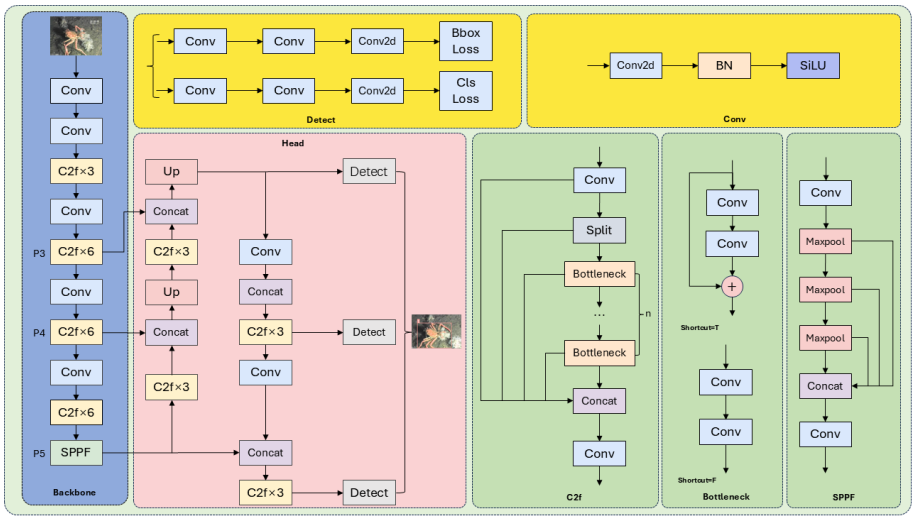


Figure 2: Improvements based on the legacy YOLOV8 network. The traditional yolov8 network structure contains backbone and C2f, Neck, head, Conv, SPPF, and Bottleneck.

models also apply MixUp and CopyPaste for better training diversity. The Backbone extracts features via a cross-stage local network, with an SPPF (Fast Spatial Pyramid Pooling) module for multi-scale context.

The Neck efficiently fuses multi-resolution features. Unlike YOLOv5's PAN-FPN, YOLOv8 performs sequential down- and up-sampling without extra convolutions. The C3 modules are replaced by lightweight C2f blocks (Figure 3), enhancing adaptability to targets of varying sizes and shapes.

The decoupled Head has parallel branches for regression and classification, reducing parameters and improving robustness. YOLOv8 adopts an anchor-free design to predict target centers and aspect ratios, enhancing speed and accuracy.

The C2f block, a two-layer CSPBottleneck, boosts feature extraction efficiency. YOLOv8 also integrates the E-ELAN architecture from YOLOv7 [14] with cross-layer branching for better gradient propagation. Shortcut connections remain in the Backbone but are disabled in the Neck. The Neck forwards fused features to the decoupled Head, yielding improved overall performance.

3.2 GSConv

To enhance the model's representational capacity while reducing computational cost and parameters for edge deployment, this study introduces GSConv, an efficient hybrid convolution module. GSConv achieves a balance between feature richness and efficiency through a dual-branch design (Figures 4).

Standard convolution produces dense feature maps but incurs high computational overhead:

$$F_{std} = K \times K \times C_{in} \times C_{out} \times H \times W \qquad (1)$$

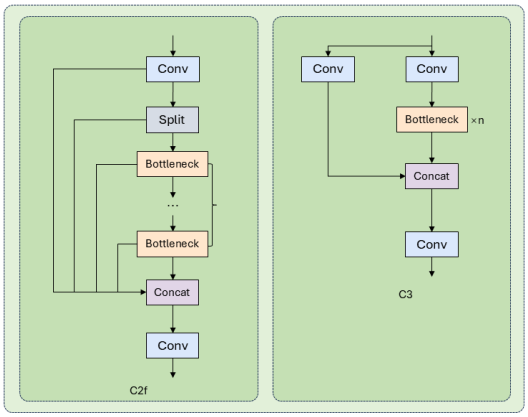


Figure 3: In YOLOv8's Backbone, the original C3 block based on CSPNet is replaced by the lighter C2f module, which improves efficiency while preserving feature extraction ability.

where K is the kernel size, C_{in} and C_{out} are input and output channels, and H, W are spatial dimensions.

Depthwise separable convolution (DSC) reduces computation by splitting a standard convolution into two steps: depthwise convolution (DWConv), which performs spatial convolution for each channel, and pointwise convolution (PWConv), which fuses the channels.

$$F_{dsc} = K \times K \times C_{in} \times H \times W + C_{in} \times C_{out} \times H \times W \qquad (2)$$

DSC lowers computation, but limited cross-channel interaction in DWConv reduces feature richness, especially in compact networks.GSConv overcomes this by combining standard convolution's expressiveness with DSC efficiency and enabling global channel interaction via channel shuffling. As shown in Fig.4, its forward pass consists of three steps:

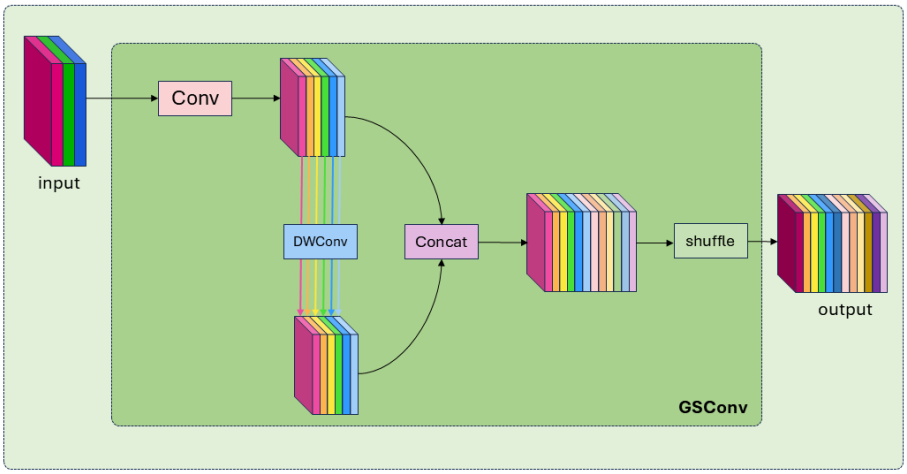


Figure 4: In the GSConv structure, a standard convolution is first applied for downsampling, followed by a depthwise convolution (DWConv). The outputs of both convolutions are then concatenated and passed through a final shuffle operation.

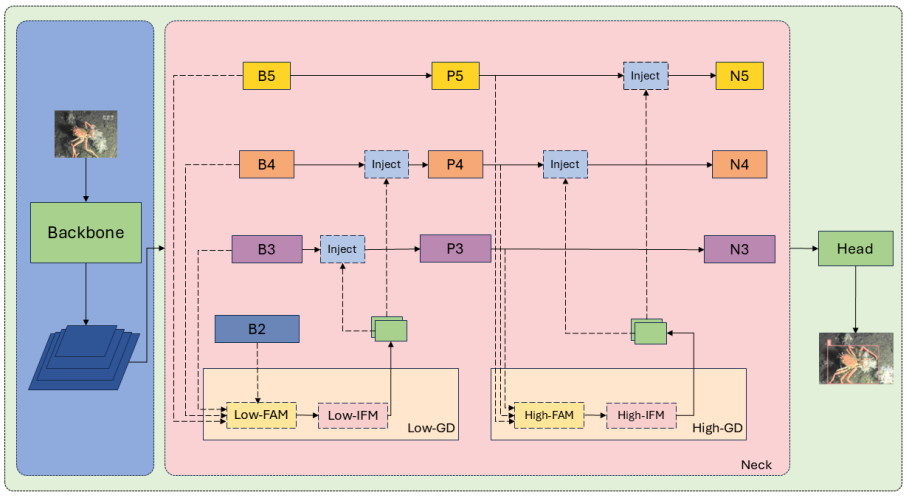


Figure 5: The GD module in Gold-YOLO introduces a Gather-and-Distribute mechanism. Information is collected and fused across layers, then redistributed to each layer. This process involves three components: the Feature Alignment Module (FAM), the Information Fusion Module (IFM), and the Information Injection Module (IIM).

1. Feature Extraction with Channel Reduction: Standard convolution reduces the input tensor $X \in \mathbb{R}^{C_{in} \times H \times W}$ to $C_{out}/2$ channels:

$$F_{std} = \text{StdConv}Cout/2(X) \tag{3}$$

2. Efficient Feature Processing: In parallel, DSC produces $C_{out}/2$ complementary channels:

$$F_{dsc} = \text{DSC}Cout/2(F_{std}) \tag{4}$$

3. Feature Fusion with Channel Integration: The two outputs are concatenated and shuffled for full channel interaction:

$$F_{out} = \text{ChannelShuffle}([F_{std}, F_{dsc}]) \tag{5}$$

Algorithm 1 GSConv Forward Propagation Algorithm

Input feature map $X \in R^{C_{in} \times H \times W}$, output channels C_{out} Output feature map $F_{out} \in R^{C_{out} \times H \times W}$
Standard convolution branch
$F_{std} \leftarrow \text{StdConv}\, Cout/2(X)$;
Depthwise separable convolution branch
$F_{dw} \leftarrow \text{DWConv}\, Cout/2(F_{std})$;
$F_{pw} \leftarrow \text{PWConv}\, Cout/2(F_{dw})$;
Concatenate and channel shuffle
$F_{cat} \leftarrow \text{Concat}(F_{std}, F_{pw})$;
$F_{out} \leftarrow \text{ChannelShuffle}(F_{cat})$;
return F_{out}

This dual-branch design preserves dense features in

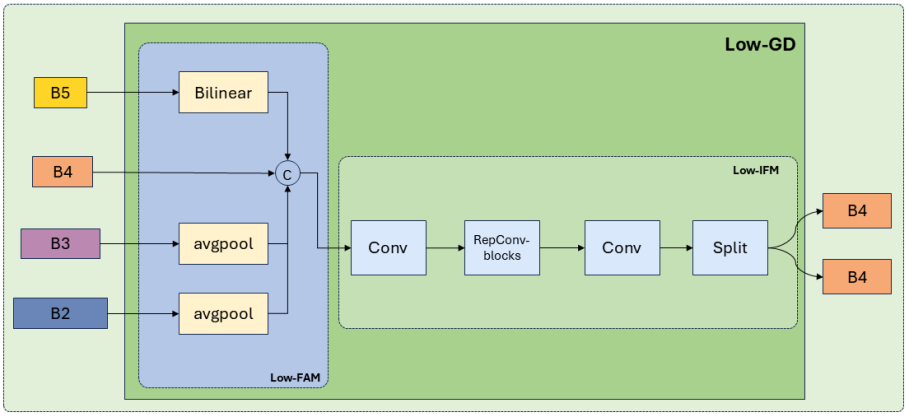


Figure 6: Low-GD fuses the shallow feature information of the model. It includes Low-FAM and Low-IFM. Low-FAM uses average pooling to obtain uniformly sized features, while Low-IFM consists of a multi-layer reparameterized convolutional block followed by a split operation.

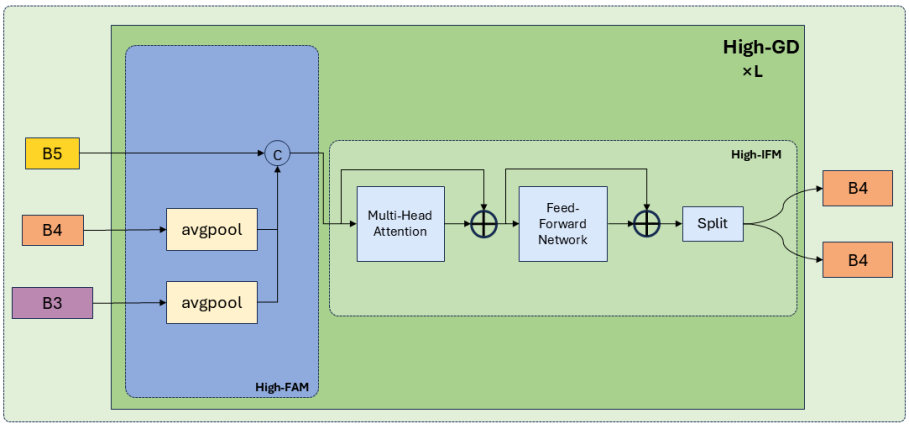


Figure 7: High-GD is similar in structure to Low-GD. It fuses features obtained from Low-GD (P3, P4, P5) and also contains High-FAM and High-IFM. High-FAM aligns feature sizes using global average pooling, while High-IFM includes multiple transformer blocks followed by a split operation.

half of the channels while maintaining DSC-level efficiency in the other half. Channel shuffling ensures effective integration, achieving representational capacity close to standard convolution with computation similar to DSC. Integrating GSConv into key network components enables high performance under constrained computational resources.

3.3 Gather-and-Distribute Mechanism

To bridge the semantic gap between shallow and deep features while enhancing multi-scale representation, we adopt the Gather-and-Distribute (GD) mechanism originally proposed in GOLD-YOLO(Figure 5). The GD mechanism is designed with three key components:

1)Feature Alignment Module (FAM): aligns feature maps from different scales into a unified resolution.

2)Information Fusion Module (IFM): integrates the aligned features to obtain global context.

3)Information Injection Module (IIM): redistributes the fused global information back to the backbone or neck layers.

The overall GD mechanism operates in two stages: a *Low-stage* and a *High-stage*, each tailored for different levels of semantic abstraction.

3.3.1 Low-stage GD

The low-stage GD focuses on enhancing fine-grained details from shallow features(Figure 6). Given backbone features $\{B2, B3, B4, B5\}$, the Low-FAM first aligns them to the spatial size of $B4$ using bilinear interpolation and average pooling:

$$F^{l}_{\text{align}} = \{A(B2), A(B3), A(B4), A(B5)\} \quad (6)$$

where $A(\cdot)$ denotes the alignment function combining bilinear resizing and average pooling.

Next, the Low-IFM aggregates the aligned features using 1×1 convolution and RepConv blocks:

$$F^{l}_{\text{global}} = \text{RepConv}\big(\text{Conv}_{1\times1}(\text{Concat}(F^{l}_{\text{align}}))\big) \quad (7)$$

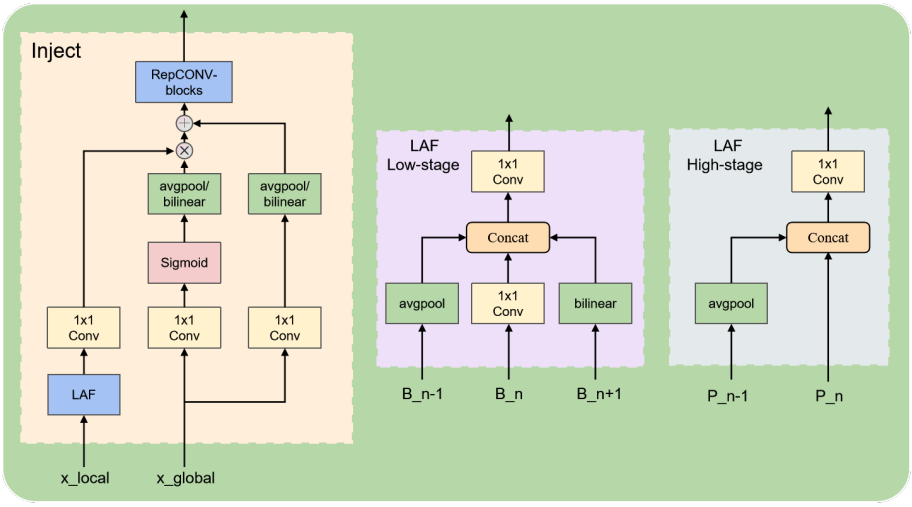


Figure 8: The structure of the information injection module.

In the Low-IIM module (Figure 8), the original feature map is first aligned in scale through average pooling and bilinear interpolation within the LAF module. It is subsequently multiplied element-wise with the global feature refined by convolution and sigmoid activation. Finally, a residual connection adds the 1×1 convolved global feature, thereby generating the fused feature map $P_{3,4}$ that effectively incorporates global contextual information.

3.3.2 High-stage GD

The high-stage GD focuses on semantic-rich high-level featuress (Figure 7). Given $\{P3, P4, P5\}$, the High-FAM aligns them to the spatial size of $P5$ using average pooling:

$$F^h_{\text{align}} = \{P(P3), P(P4), P(P5)\} \tag{8}$$

where $P(\cdot)$ denotes pooling-based alignment.

The High-IFM employs a multi-head self-attention (MHSA) module followed by a feed-forward network (FFN) to capture long-range dependencies and fuse features:

$$F^h_{\text{global}} = \text{FFN}(\text{MHSA}(F^h_{\text{align}})) \tag{9}$$

In the High-IIM module (Figure 8), the overall procedure is identical to that of the Low-stage IIM. The only difference lies in the LAF module, which employs average pooling alone to align the multi-level original features to a unified scale, without performing bilinear interpolation. This simplification preserves the capability of global context fusion while reducing computational overhead, and the resulting fused features effectively integrate high-level semantic information.

3.3.3 Pseudocode

The overall procedure of the GD mechanism is summarized in Algorithm 2.

Algorithm 2 Gather-and-Distribute (GD) Mechanism

Backbone features $\{B2, B3, B4, B5\}$
Neck features $\{P3, P4, P5\}$
Enhanced features $\{B3', B4', P4', P5'\}$
Low-stage GD:
Align $\{B2, B3, B4, B5\}$ to $B4$ size $\rightarrow F^l_{\text{align}}$
Fuse by Conv + RepConv $\rightarrow F^l_{\text{global}}$
Inject into $B3, B4 \rightarrow B3', B4'$
High-stage GD:
Align $\{P3, P4, P5\}$ to $P5$ size $\rightarrow F^h_{\text{align}}$
Fuse by MHSA + FFN $\rightarrow F^h_{\text{global}}$
Inject into $P4, P5 \rightarrow P4', P5'$
return $\{B3', B4', P4', P5'\}$

In summary, the GD mechanism enables effective cross-scale information interaction. The low-stage branch enhances shallow features with fine-grained local cues, while the high-stage branch provides semantic-rich global context. Together, they significantly improve the representational capacity of the detector across varying object scales.

4 Experiments

4.1 Experimental Configuration and Reproducibility

All experiments were conducted under consistent hardware and software settings to ensure reproducibility. The environment consisted of an NVIDIA GeForce RTX 4070 Ti GPU (12 GB VRAM), a 12-core Intel Xeon Platinum 8255C CPU, and 32 GB RAM, running Ubuntu 22.04 (64-bit) with CUDA 12.2, Python 3.10.12, and PyTorch 2.2.2 (Table 1).

Images were resized to 640 × 640 pixels with

padding to preserve aspect ratio and normalized to [0, 1]. Bounding boxes were converted to $(x_{\text{center}}, y_{\text{center}}, w, h)$ format, and labels were one-hot encoded. Data augmentation included Mosaic (0.5), MixUp ($\alpha = 0.2$), CopyPaste (for larger models), random horizontal flips (0.5), and color jittering (± 0.2 in brightness, contrast, saturation), implemented using Albumentations 1.3.0.

Training used SGD with an initial learning rate (lr0) of 0.01, decay (lrf) of 0.01 under a cosine annealing schedule, momentum 0.937, and weight decay 0.0005. A 3-epoch warm-up gradually increased the learning rate. The batch size was set to -1 to allow automatic adjustment based on GPU memory. Models were trained for 300 epochs with 640×640 inputs, balancing accuracy and efficiency. Losses included IoU-based box regression, objectness, and classification. Key hyperparameters and training configurations are summarized in Table 2.

Name	Configuration
CPU	12-core Intel(R) Xeon(R) Platinum 8255C
GPU	NVIDIA GeForce RTX 4070 Ti
Memory	12 GB
Environment	Ubuntu 22.04
CUDA	12.2
Python	3.10.12
PyTorch	2.2.2

Table 1: Configuration of experimental equipment and experimental environment requirements.

Hyperparameter	Value	Description
learning rate (lr0)	0.01	Controls the step size for weight updates during training
learning decay rate (lrf)	0.01	Regulates the decay of learning rate over epochs
momentum	0.937	Stabilizes training by smoothing gradient updates
weight_decay	0.0005	Prevents overfitting by penalizing large weights
warmup_epochs	3.0	Gradually increases initial learning rate to prevent instability
batch	-1	Auto-adjusts batch size based on available GPU memory
imgsz	640	Input image size in pixels for model training

Table 2: Important hyperparameters and configurations for YOLOv8 training.

4.2 Data set

This study utilizes two complementary underwater datasets. The first, from the J-EDI dataset [23], contains 5,720 images of diverse marine debris captured in authentic underwater environments, with bounding-box annotations for debris, biological entities, and remotely operated vehicles (ROVs). The second, the Brackish Underwater Dataset [24], comprises 14,674 real European underwater images annotated for fish, crabs, and small objects, covering a wide range of sizes and densities. Together, these datasets provide a comprehensive benchmark for underwater object detection under realistic conditions. Figures 9 and 10 show representative examples from the Brackish Underwater Dataset and the J-EDI dataset, respectively.

4.3 Model Evaluation Indicators

To evaluate model performance, we adopt standard object detection metrics, including Precision, Recall, Average Precision (AP), and mean Average Precision (mAP). Their mathematical definitions are given in Eqs. (10)–(12).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

$$AP = \sum_{n} (\text{Recall}_{n+1} - \text{Recall}_n) \cdot \text{Precision}_n \tag{12}$$

True Positive (TP) denotes correctly predicted positive samples that match the ground truth. False Negative (FN) occurs when positive samples are missed, and False Positive (FP) refers to incorrect positive predictions. Higher Precision indicates fewer false alarms, while higher Recall reflects fewer missed detections.

Detection performance is evaluated using mAP at different IoU thresholds: mAP@0.5 and mAP@0.5:0.95. The former measures average precision at a single threshold of 0.5, while the latter averages AP over ten thresholds from 0.5 to 0.95 in steps of 0.05, providing a comprehensive assessment of detection capability (Eq. 13).

$$mAP@0.5:0.95 = \frac{\sum i = 0.5^{0.95} AP_{\text{IoU}=i}}{n} \tag{13}$$

The accuracy of predicted bounding boxes is measured by Intersection over Union (IoU), a key metric in object detection. IoU quantifies the overlap between a predicted box and the ground truth, reflecting localization precision during training and evaluation (Figures 11). It is defined as:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{14}$$

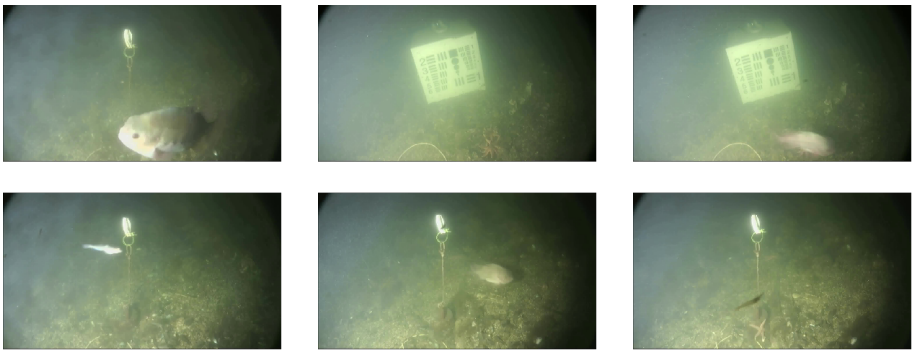


Figure 9: Representative samples from the Brackish Underwater Dataset

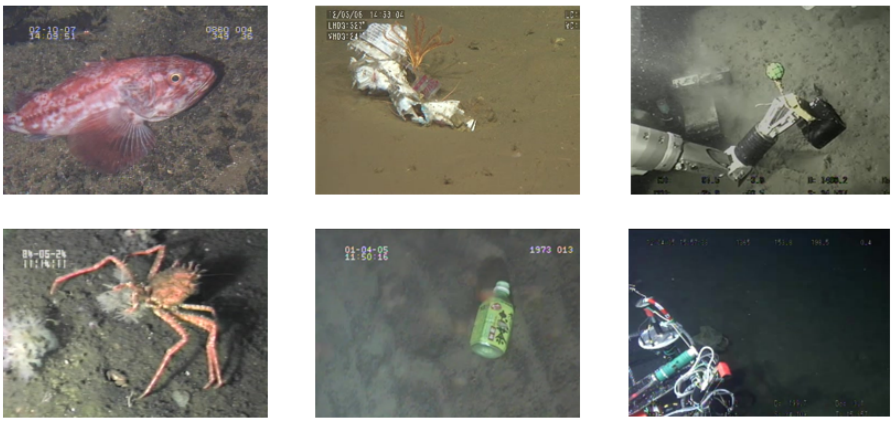


Figure 10: Examples of all categories in the dataset; (a) underwater organism; (b) underwater trash ; (c) rov.

As shown in Figure 10, IoU is the ratio of the overlapping area to the combined area of two boxes. Higher IoU indicates greater overlap, while values near 0 indicate minimal overlap. An IoU of 1 represents perfect alignment with the ground truth.

YOLOv8	GD	GSConv	Input	mAP@0.5:0.95
✓			640×640	84.5%
✓	✓		640×640	85.6%
✓		✓	640×640	84.4%
✓	✓	✓	640×640	88.5%

Table 3: The results of the ablation experiments. A '✓' indicates the use of the corresponding module.

4.4 Ablation experiment

To evaluate the effectiveness of the proposed improvements, ablation studies were conducted using the same training strategy and hyperparameters. As shown in Table 2, incorporating only the GSConv module into the backbone reduces FLOPs with a slight performance drop. Modifying only the neck to adopt Gold-YOLO's structure significantly improves detection accuracy, though it increases computation. Combining both Gold-YOLO's neck and GSConv modules achieves the highest mAP@0.5:0.95, demonstrating the effectiveness of the proposed approach.

4.5 Contrast experiment

The experimental performance of our method was evaluated and compared with several state-of-the-art lightweight target detection algorithms on both the J-EDI and Brackish Underwater datasets. As shown in Table 4 (J-EDI dataset), our method achieves a mAP@0.5:0.95 of 88.5%, outperforming all compared methods, including a 5.1% improvement over YOLOv5n (2021). Table 5 presents the results on the Brackish Underwater dataset, where our approach consistently achieves superior performance across all evaluated metrics. Integrating the GSConv module into YOLOv8n reduces model size but leads to a slight decrease in mAP@0.5:0.95, highlighting the trade-off between efficiency and detection accuracy. Overall,

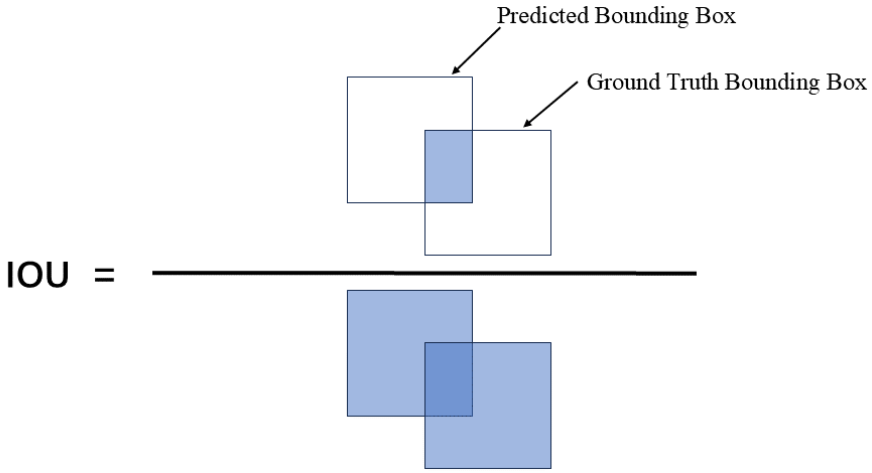


Figure 11: IOU is a measure of the effectiveness of the model by calculating the ratio of the area of intersection of the predicted and real frames to the area of concatenation, and a higher IOU means that the accuracy of the model's prediction is also higher.

these results indicate that our method maintains a favorable balance between accuracy and computational cost across different underwater scenarios.

Although YOLOv8n and Slim-Neck+v8 show advantages in computational efficiency and parameter reduction, their detection accuracies remain relatively low across both datasets. On the J-EDI dataset, they achieve 84.5% and 84.7%, respectively, while on the Brackish Underwater dataset, their accuracies are 82.2% and 82.3%. In contrast, our proposed method attains higher accuracies of 88.5% on J-EDI and 84.7% on Brackish Underwater, with only a slight increase in computational complexity. Compared to the original Slim-Neck method, our approach improves mAP@0.5:0.95 by 3.8% and 2.4%, respectively, while also reducing model size. Notably, our model is 38% smaller than YOLOv8s. Experiments incorporating the SEAttention mechanism or integrating DSConv [25] did not yield significant improvements, indicating that the performance gains mainly stem from our architectural modifications. Overall, comprehensive comparisons demonstrate that our method provides a more favorable trade-off between detection accuracy and computational efficiency, especially in underwater scenarios with complex visual characteristics.

To assess the robustness and generalization of the proposed YOLO-GGS model, we compared it with the YOLOv5 baseline, monitoring performance every 50 training epochs. As summarized in Table 6, YOLO-GGS demonstrates superior stability and consistently outperforms YOLOv5 throughout training.

The trained YOLO-GGS model weights were used to detect images from various underwater scenes. The results, shown in Figure 14, indicate enhanced robustness and higher confidence levels. The model maintains strong recognition ability even in complex or occluded environments, effectively meeting the demands of diverse underwater scenarios.

Figure 15 presents experimental results on the J-EDI dataset, comparing our model with mainstream frameworks, including YOLOv3-tiny, YOLOv5, and YOLOv8. Our model achieves substantially higher detection accuracy, with the highest mAP@0.5:0.95 among all compared methods. YOLOv3-tiny shows signs of overfitting around epoch 272, likely due to the mismatch between its model complexity and the dataset characteristics. Despite this, its overall training performance remains competitive. These results demonstrate that our approach improves detection accuracy while reducing model parameters, highlighting its effectiveness and efficiency.

Models	Input	Batch	Epoch	mAP@0.5:0.95
YOLOv3	640×640	-1	300	85.5%
YOLOv5n	640×640	-1	300	83.4%
YOLOv5s	640×640	-1	300	84.1%
YOLOv6n	640×640	-1	300	83.9%
DSConv+v5	640×640	-1	300	83.2%
YOLOv8n	640×640	-1	300	84.5%
Slim-Neck+v8	640×640	-1	300	84.7%
DSConv+v8	640×640	-1	300	84.4%
SEattention+v8	640×640	-1	300	83.9%
GD+v8	640×640	-1	300	85.6%
Mamba-YOLO	640×640	-1	300	85.2%
Ours	**640×640**	**-1**	**300**	**88.5%**

Table 4: Performance comparison of different object detection models on the J-EDI dataset.

Models	Input	Batch	Epoch	mAP@0.5:0.95
YOLOv3	640×640	-1	300	72.49%
YOLOv5	640×640	-1	300	81.2%
YOLOv5s	640×640	-1	300	81.7%
YOLOv6	640×640	-1	300	81.4%
YOLOv7	640×640	-1	300	69.5%
YOLOv8n	640×640	-1	300	82.2%
Slim-Neck+v8	640×640	-1	300	82.3%
DSConv+v8	640×640	-1	300	81.9%
SEattention+v8	640×640	-1	300	81.5%
GD+v8	640×640	-1	300	82.8%
Mamba-YOLO	640×640	-1	300	83.3%
Ours	**640×640**	**-1**	**300**	**84.7%**

Table 5: Evaluation results of various object detection models on the Brackish Underwater Dataset.

We also evaluated Mamba-YOLO on the underwater litter dataset. Despite its advanced architecture, multiple training runs revealed a tendency for overfitting. Specifically, validation losses significantly exceeded training losses after several epochs. As a result, its final detection accuracy was lower than that of our proposed method. These observations suggest that Mamba-YOLO may require additional task-specific adjustments or regularization strategies, which we plan to investigate in future work.

Furthermore, we conducted a comprehensive comparison of YOLO-GGS with representative YOLO architectures, including YOLOv3 variants (YOLOv3, YOLOv3-SPP, YOLOv3-Tiny), YOLOv5 variants (n, s, m, l, x), and YOLOv8 variants (n, s, m, l, x). Detailed results are presented in Table 7, highlighting the effectiveness of our proposed model across diverse architectures and scales.

Models	Input	Epoch	mAP@0.5:0.95	cls_loss
YOLOv5n	640×640	50	73.9%	0.547
	640×640	100	80.2%	0.417
	640×640	150	82.0%	0.374
	640×640	200	82.9%	0.366
	640×640	250	83.2%	0.359
	640×640	300	83.4%	0.356
Ours	640×640	50	80.1%	0.442
	640×640	100	84.7%	0.342
	640×640	150	86.5%	0.323
	640×640	200	87.8%	0.316
	640×640	250	88.3%	0.313
	640×640	300	**88.5%**	**0.309**

Table 6: Comparison of YOLOv5n and our model across different training epochs on the J-EDI dataset, showing that the proposed method consistently achieves higher stability and accuracy in terms of classification loss and overall performance.

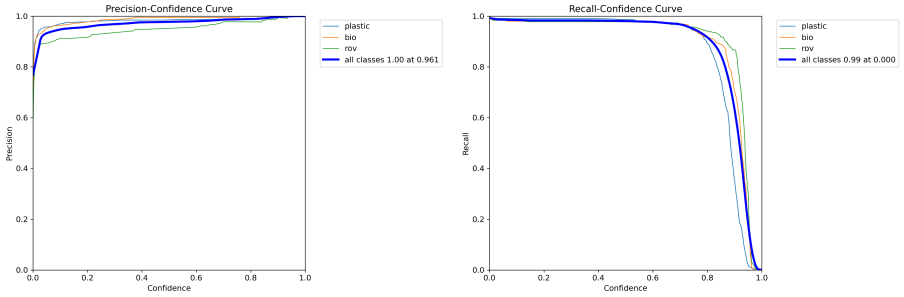
Models	Input	Batch	Epoch	mAP@0.5:0.95
YOLOv3	640×640	-1	300	85.5%
YOLOv3-spp	640×640	8	300	85.3%
YOLOv3-tiny	640×640	-1	300	82.1%
YOLOv5n	640×640	-1	300	83.4%
YOLOv5s	640×640	-1	300	85.3%
YOLOv5m	640×640	-1	300	82.1%
YOLOv5l	640×640	-1	300	85.5%
YOLOv5x	640×640	-1	300	85.3%
YOLOv8n	640×640	-1	300	84.5%
YOLOv8s	640×640	-1	300	84.9%
YOLOv8m	640×640	-1	300	85.4%
YOLOv8l	640×640	-1	300	85.5%
YOLOv8x	640×640	8	300	85.8%
Ours	**640×640**	**-1**	**300**	**88.5%**

Table 7: Comparison of detection accuracy on the J-EDI dataset among YOLOv3, YOLOv5, and YOLOv8.

4.6 Real-Time Performance Evaluation

To further analyze real-time performance, we compare not only the average inference time and memory footprint but also the trade-off between computational efficiency and deployment feasibility. As shown in Table 8, the proposed method achieves the lowest memory consumption (414 MB) and fastest inference (9.2 ms, 108.4 FPS), significantly exceeding the 30 FPS threshold required for real-time applications. Compared with YOLOv8n, it reduces memory usage by 13% and delivers 17% faster inference, demonstrating a superior efficiency-to-accuracy ratio. All experiments were conducted on a single NVIDIA RTX 4070Ti GPU with an input image size of 640.

The lightweight design of the model ensures stable inference latency and enables deployment on edge devices with limited GPU memory. It supports multi-camera parallel inference and real-time decision-making in robotics and underwater detection systems. Furthermore, its compact architecture provides flexibility for integration into mobile and embedded platforms, reducing deployment cost and enhancing applicability in resource-constrained scenarios. These advantages highlight the method's potential for broader real-world applications requiring both high speed and efficiency.



(a) (b)

Figure 12: (a) shows the relationship between Precision and Confidence. The horizontal axis represents Confidence, with higher values indicating greater certainty in the detection results. The vertical axis represents Precision, defined as the proportion of correctly detected spam among all items detected as garbage. (b) shows the relationship between Recall and Confidence. The horizontal axis again represents Confidence. The vertical axis represents Recall, defined as the proportion of detected spam among all actual garbage items. As confidence increases, precision generally improves, while recall may vary depending on the threshold, reflecting the model's detection effectiveness.

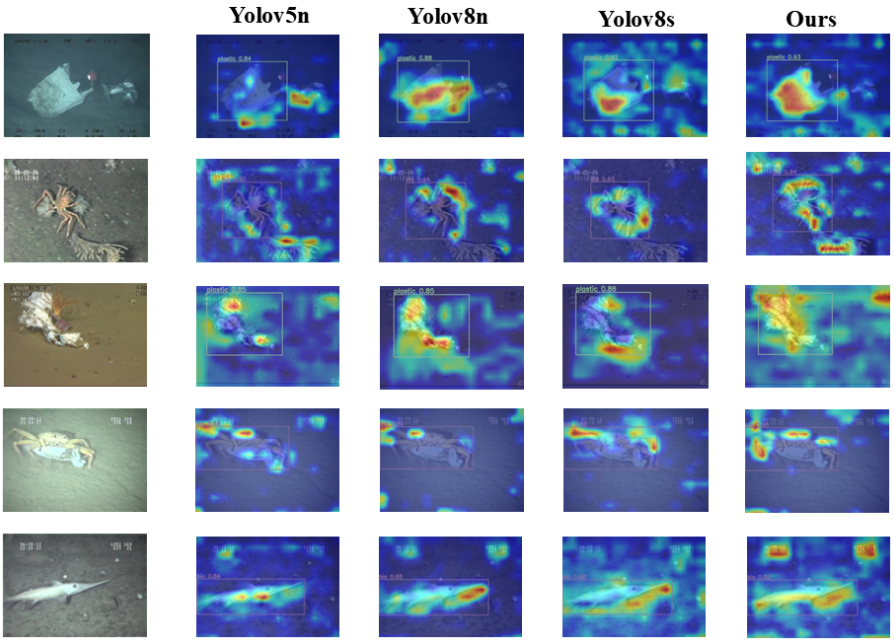


Figure 13: We used Grad-CAM (Gradient Weighted Class Activation Map) method to compare the improved model with the heat maps generated by the mainstream YOLOv5, YOLOv8n and YOLOv8s models. In the comparison map, it can be found that the attention of our improved model is more focused and more sensitive to the detection target.

Models	Memory usage (Mb)	Inference time (ms)	FPS (frames/s)
YOLOv3	1355	33.3	30.3
YOLOv5n	455	12.7	78.8
YOLOv6n	440	10.6	94.5
YOLOv7	462	11.35	88.1
YOLOv8n	477	10.8	92.6
Mamba-YOLO	521	12.3	81.3
Ours	**414**	**9.2**	**108.4**

Table 8: Memory and inference speed comparison of YOLO-based models on single-frame input.

4.7 Overall experimental analysis

To conduct a comprehensive evaluation, we compared our model with several mainstream detectors under identical configurations. Results show that our method achieves higher accuracy with a more compact size. Grad-CAM [26] visualizations in Figure 13 show clearer and more focused attention regions than YOLOv5s, YOLOv8n, and YOLOv8s. This indicates stronger feature localization for underwater litter detection.

Figure 14: Plot of results of detection in different underwater scenarios. The trained model is used in different detection scenarios and different detection objects for detection and the results are shown in Fig. The results show that our model has strong generalization ability in underwater detection sites.

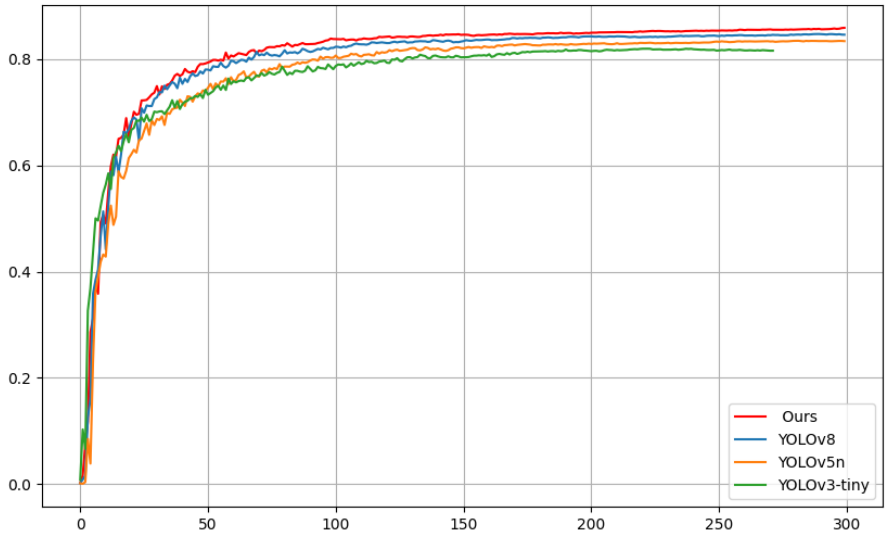


Figure 15: Our model is compared with yolov3-tiny, yolov5, and yolov8 for the experimental procedure mAP@0.5:0.95 values. The figure shows that our model always maintains a more stable effect during the overall process of training, with the highest training accuracy.

Figure 12 presents precision–confidence and recall–confidence curves. Higher precision at elevated confidence levels indicates reliable predictions, while recall declines as low-confidence true positives are excluded. Category-specific differences are observed, with the ROV class showing slightly lower precision due to more false positives. Overall, precision approaches 1.0 and recall nearly 0.99 at high thresholds.

During training, most models used a batch size of 16, while YOLOv3-SPP and YOLOv8x used 8 due to GPU memory limits. Despite achieving similar mAP@0.5:0.95 values of 85.8%, these models contain 15× more parameters, resulting in slower inference and higher computational cost. In contrast, our model maintains high accuracy with fewer parameters, striking a better balance between precision and complexity.

For real-time evaluation, single-frame inference time and GPU memory usage were measured. Our method achieves the fastest inference (9.2 ms, 108.4 FPS) and lowest memory footprint (414 MB), surpassing YOLOv3, YOLOv5n, YOLOv6n, YOLOv7, YOLOv8n, and Mamba-YOLO, demonstrating its suitability for resource-constrained platforms.

Nonetheless, limitations remain. By comparing the model's performance improvements and detection performance on the two datasets, we found that the model's performance may degrade in complex underwater scenes with severe occlusion or poor lighting, while fine-grained categories such as small ROVs still show high false positives. Further optimization is

needed to ensure stable real-time operation on embedded devices.

5 Conclusion

In this study, we propose YOLO-GSS, an improved YOLOv8-based model integrating the Gather-and-Distribute (GD) mechanism, GSConv, and a Slim-Neck structure. The model achieves a balance between high detection accuracy and low computational cost for real-time underwater litter detection. Experimental results on the J-EDI and Brackish datasets demonstrate mAP@0.5:0.95 values of 88.5% and 84.7%, respectively, outperforming mainstream YOLO variants, while Grad-CAM visualizations confirm strong feature localization.

Single-frame inference analysis shows that YOLO-GSS processes each frame in 9.2 ms (108.4 FPS) using only 414 MB of memory, highlighting its suitability for deployment on resource-constrained AUV and ROV platforms. Despite these advantages, performance may degrade for small targets under extreme illumination, severe occlusion, or cluttered scenes, indicating that further optimization is needed for consistent real-time operation.

Future work will focus on enhancing multi-scale feature fusion for small and occluded objects, exploring efficient attention mechanisms, and developing dynamic inference strategies to improve generalization. Additionally, we plan to deploy the model in real underwater embedded systems to facilitate practical applications in marine environmental protection and intelligent monitoring.

References

[1] Dimitris V Politikos et al. "Automatic detection of seafloor marine litter using towed camera images and deep learning". In: *Marine Pollution Bulletin* 164 (2021), p. 111974. DOI: https://doi.org/10.1016/j.marpolbul.2021.111974.

[2] Vishal Verma et al. "A deep learning-based intelligent garbage detection system using an unmanned aerial vehicle". In: *Symmetry* 14.5 (2022), p. 960. DOI: https://doi.org/10.3390/sym14050960.

[3] Dongliang Ma et al. "MLDet: Towards efficient and accurate deep learning method for Marine Litter Detection". In: *Ocean & Coastal Management* 243 (2023), p. 106765. DOI: https://doi.org/10.1016/j.ocecoaman.2023.106765.

[4] Xiaowen Teng et al. "The Object Detection of Underwater Garbage with an Improved YOLOv5 Algorithm". In: *Proceedings of the 2022 International Conference on Pattern Recognition and Intelligent Systems*. 2022, pp. 55–60. DOI: https://doi.org/10.1145/3549179.3549189.

[5] Faiza Rehman et al. "Optimized YOLOV8: An efficient underwater litter detection using deep learning". In: *Ain Shams Engineering Journal* 16.1 (2025), p. 103227. DOI: https://doi.org/10.1016/j.asej.2024.103227.

[6] Lifu Wei et al. "Image semantic segmentation of underwater garbage with modified U-Net architecture model". In: *Sensors* 22.17 (2022), p. 6546. DOI: https://doi.org/10.3390/s22176546.

[7] Mupparaju Sohan et al. "A review on yolov8 and its advancements". In: *International Conference on Data Intelligence and Cognitive Informatics*. Springer. 2024, pp. 529–545. DOI: https://doi.org/10.1007/978-981-99-7962-2_39.

[8] Chengcheng Wang et al. "Gold-YOLO: Efficient object detector via gather-and-distribute mechanism". In: *Advances in Neural Information Processing Systems* 36 (2024). DOI: https://doi.org/10.48550/arXiv.2309.11331.

[9] Hulin Li et al. "Slim-neck by GSConv: a lightweight-design for real-time detector architectures". In: *Journal of Real-Time Image Processing* 21.3 (2024), p. 62. DOI: https://doi.org/10.1007/s11554-024-01436-6.

[10] Ping-I Lin et al. "Investigating sources of marine litter and developing coping strategies in scuba diving spots in Taiwan". In: *Sustainability* 14.9 (2022), p. 5726. DOI: https://doi.org/10.3390/su14095726.

[11] Wei Zhou et al. "YOLOTrashCan: a deep learning marine debris detection network". In: *IEEE Transactions on Instrumentation and Measurement* 72 (2022), pp. 1–12. DOI: https://doi.org/10.1109/TIM.2022.3225044.

[12] Yan Zhai. "River ship monitoring based on improved deep-sort algorithm". In: *Informatica* 48.9 (2024). DOI: https://doi.org/10.31449/inf.v48i9.5886.

[13] Xiaolong Qi. "Event-triggered predictive control algorithm for multi-auv formation modeling". In: *Informatica* 48.9 (2024). DOI: https://doi.org/10.31449/inf.v48i9.5890.

[14] 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". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 7464–7475. DOI: arXiv:2207.02696.

[15] Jin Zhu et al. "YOLOv8-C2f-Faster-EMA: an improved underwater trash detection model based on YOLOv8". In: *Sensors* 24.8 (2024), p. 2483. DOI: https://doi.org/10.3390/s24082483.

[16] Pratima Sarkar, Sourav De, and Sandeep Gurung. "U-YOLOv3: A Model Focused on Underwater Object Detection". In: *Informatica* 49.6 (2025). DOI: https://doi.org/10.31449/inf.v49i6.6642.

[17] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018). DOI: https://doi.org/10.48550/arXiv.1804.02767.

[18] Michael S Fulton, Jungseok Hong, and Junaed Sattar. "Trash-icra19: A bounding box labeled dataset of underwater trash". In: (2020). DOI: https://doi.org/10.13020/x0qn-y082.

[19] Fan Zhao et al. "Seafloor debris detection using underwater images and deep learning-driven image restoration: A case study from Koh Tao, Thailand". In: *Marine Pollution Bulletin* 214 (2025), p. 117710. DOI: https://doi.org/10.1016/j.marpolbul.2025.117710.

[20] Xuemeng Zhao and Yinglei Song. "Improved ship detection with YOLOv8 enhanced with MobileViT and GSConv". In: *Electronics* 12.22 (2023), p. 4666. DOI: https://doi.org/10.3390/electronics12224666.

[21] Tianheng Cheng et al. "Yolo-world: Real-time open-vocabulary object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 16901–16911. DOI: https://doi.org/10.1109/CVPR52733.2024.01599.

[22] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125. DOI: https://doi.org/10.1109/CVPR.2017.106.

[23] Michael Fulton et al. "Robotic detection of marine litter using deep visual detection models". In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 5752–5758. DOI: https://doi.org/10.1109/ICRA.2019.8793975.

[24] Malte Pedersen et al. "Detection of Marine Animals in a New Underwater Dataset with Varying Visibility". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[25] Marcelo Gennari do Nascimento, Roger Fawcett, and Victor Adrian Prisacariu. "Dsconv: Efficient convolution operator". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 5148–5157. DOI: https://doi.org/10.1109/ICCV.2019.00525.

[26] Ramprasaath R Selvaraju et al. "Grad-CAM: visual explanations from deep networks via gradient-based localization". In: *International journal of computer vision* 128 (2020), pp. 336–359. DOI: https://doi.org/10.1007/s11263-019-01228-7.