remote sensing

MDPI

Article

RepLite-YOLO: A Parameter-Efficient Residual-Enhanced Detector for Ship Recognition in Remote Sensing Imagery

Ruijia Fu, Zuomin Wang *, Zijun Lin, Ying Li and Bingxin Liu

Navigation College, Dalian Maritime University, Dalian 116026, China; furuijia@dlmu.edu.cn (R.F.); lindaolin2lan@dlmu.edu.cn (Z.L.); yldmu@dlmu.edu.cn (Y.L.); gisbingxin@dlmu.edu.cn (B.L.)
* Correspondence: wangzuomin@dlmu.edu.cn

Highlights

What are the main findings?

- We propose RepLite-YOLO, a lightweight ship detection model based on YOLOv11n that integrates the existing ADown module, C3k2_OREPA_RS backbone, ELANFusion_Block, and Detect_DWLite head to effectively balance detection accuracy and edge computational limits.
- RepLite-YOLO reduces the parameter count by 57.4% and GFLOPs by 49.2% compared with YOLOv11n, while maintaining competitive detection accuracy and achieving slight mAP@50 improvements of 1.2 and 1.3 percentage points on the Vessel and Ship Detection datasets, respectively.

What are the implications of the main findings?

- The spatially complementary feature extraction and zero additional re-parameterization-induced inference overhead help alleviate feature degradation in complex, noisy sea conditions.
- RepLite-YOLO's favorable trade-off between computational cost and detection accuracy indicates its potential as an edge-oriented lightweight candidate for maritime monitoring on resource-constrained platforms.

Abstract

Remote sensing ship detection plays a pivotal role in maritime surveillance, safety assurance, and traffic management. However, current detection methods often face significant challenges due to complex sea-surface background noise, large target-scale variations, and edge-hardware limitations. In this paper, we propose RepLite-YOLO, a lightweight detection framework based on YOLOv11n. Specifically, to alleviate irreversible spatial information loss during downsampling, we adopt the ADown module, originally introduced in YOLOv9, to generate spatially complementary features through its two-branch downsampling mechanism. This design helps preserve salient hull-edge responses while suppressing part of the random sea-surface interference, thereby improving feature robustness for small ship targets. To achieve substantial structural streamlining while maintaining competitive representational capacity under strict hardware constraints, we design the C3k2_OREPA_RS module, utilizing online re-parameterization (OREPA) to efficiently reconstruct deep layers without additional re-parameterization-induced inference operations. Furthermore, we construct the ELANFusion_Block by integrating Depthwise Separable Convolutions (DSC) into the ELAN paradigm to alleviate the multi-scale aggregation bottleneck, and tailor the Detect_DWLite head for highly compressed decoupled prediction. Experimental results show that RepLite-YOLO achieves a favorable balance between detection accuracy and computational efficiency. Compared with YOLOv11n, it reduces the

Check for updates

Received: 17 May 2026
Revised: 25 June 2026
Accepted: 3 July 2026
Published: 6 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

number of parameters by 57.4% and GFLOPs by 49.2%, while maintaining competitive detection accuracy with slight mAP@50 improvements of 1.2 and 1.3 percentage points on the Vessel dataset and Ship Detection dataset, respectively.

Keywords: ship detection; YOLOv11n; lightweight networks; depthwise separable convolution

1. Introduction

Ship management and maritime surveillance are important components of national defense, fishery regulation, and traffic control. They are important for keeping the economy stable and protecting maritime rights. As remote sensing technology evolves, high-resolution optical satellites and Unmanned Aerial Vehicles (UAVs) have become essential tools for extensive and detailed ocean observation, providing an abundance of high-quality optical monitoring data [1]. At present, smart maritime supervision requires that these pictures be used to automatically find ships.

Deep learning has substantially transformed object detection. It has surpassed the limitations of traditional Constant False Alarm Rate (CFAR) methods and substantially improved performance in challenging scenarios. The YOLO family of single-stage detectors is the most popular choice for online applications because they are the best at balancing model efficiency and accuracy [2]. For instance, Redmon et al. added multi-scale feature fusion and Feature Pyramid Networks (FPNs) to YOLOv3 [3], which made it much better at finding small things while still being able to do so in real time. Bochkovskiy et al. also added the Bag of Freebies (BoF) and Bag of Specials (BoS) methods to YOLOv4 [4]. They accomplished this by employing training techniques and post-processing modules that incur no additional operational costs. Li et al. added re-parameterization structures to YOLOv6, which made re-parameterized Path Aggregation Network (Rep-PAN) necks and re-parameterized backbones that worked very well [5]. Hu et al. [6] subsequently advanced this concept by employing online convolutional re-parameterization (OREPA) at CVPR 2022.

In the specialized domain of remote sensing ship detection, a multitude of adaptive enhancements have emerged: Zeng et al. [7] proposed YOLO-Ssboat utilizing deformable convolutions for ultra-small vessels; Cao and Sui [8] introduced DRGD-YOLO for dynamic multi-scale fusion; He et al. [9] developed the lightweight SSGY; Yuan et al. [10] proposed YOLO-SBA to handle complex backgrounds; and various researchers have customized YOLO variants for Synthetic Aperture Radar (SAR) imagery [11–14] or specific challenges such as rotation invariance [15], or high-resolution optical imagery like GF-2 [16]. Even though these structural changes and attention mechanisms have significantly improved detection accuracy, deploying these advanced deep learning models on resource-constrained edge platforms, such as low-power UAVs and miniaturized satellites, introduces severe hardware limitations compared to ground-based server environments. In dynamic maritime monitoring tasks, detection models must operate continuously under strict power consumption and storage constraints. Traditional detection models, characterized by massive parameter counts and dense floating-point operations, require excessive memory access and compute power. This not only burdens the limited storage capacity of edge devices but also causes high dynamic power consumption, which severely shortens the mission endurance of battery-powered platforms. The primary objective of remote sensing ship detection has evolved from solely achieving the highest "inference speed" to identifying the optimal equilibrium among model compactness (for storage and transmission), theoretical computational efficiency, and detection accuracy. To resolve these contradictions

on resource-constrained edge devices and achieve high-performance maritime monitoring, current mainstream detection architectures still face four key technical bottlenecks:

1. Loss of small-target features: In remote sensing images, ships usually occupy only a few pixels. When standard strided convolution or pooling-based downsampling is used, the receptive field becomes larger, but fine-grained spatial information may be permanently lost. This makes the edges of small or thin ships look blurry or disappear in deep networks [17].
2. Semantic Interference in Complex Backgrounds: Nearshore ports and complex sea conditions, such as waves and wakes, contain many ship-like structures, including docks and breakwaters. If models do not have strong deep semantic discrimination, they are likely to have a lot of false alarms [18].
3. Inefficiency of Multi-Scale Feature Fusion: Traditional Feature Pyramid (neck) structures often need a lot of computing power to combine high-resolution spatial details with low-resolution semantic features. This makes it hard for information to flow quickly when there are lightweight constraints.
4. Computational Bottlenecks in Detection Head Design: Current decoupled detection heads enhance task specificity but also add a lot of parameters and Giga floating point operations (GFLOPs). Detecting small objects requires processing high-resolution inputs. This creates a lot of unnecessary computation, which makes it hard to make updates quickly on edge platforms with limited bandwidth.

While generic lightweight models (e.g., standard YOLO variants) excel in natural scenarios, directly applying them to remote sensing maritime detection often yields suboptimal results. Generic compression strategies inevitably exacerbate the feature degradation of tiny vessels, and simple channel reduction diminishes the network's semantic capability to suppress sea clutter. Therefore, a domain-specific structural redesign is required to optimize the trade-off between edge efficiency and maritime detection robustness.

To address these problems systematically, we propose RepLite-YOLO, a lightweight, high-efficiency ship detection network. Overall, the main contributions of this paper are as follows:

1. Against Feature Annihilation (ADown-Based Downsampling Adaptation): We adopt the existing ADown module to replace standard strided convolutions. Our contribution lies in adapting and evaluating this module for lightweight remote sensing ship detection. By employing a two-stage mechanism, it tends to decouple deterministic ship contours from random sea clutter. As visually supported by our feature heatmaps, this module generates complementary spatial representations to mitigate irreversible information loss.
2. Parameter-Efficient Feature Extraction (Zero-Latency Structural Streamlining): To resolve the heavy parameter burden of deep feature extraction stages, we design the C3k2_OREPA_RS module. By leveraging OREPA isolated via our controlled ablation studies, we construct a rich parameter space during training. These multi-branch structures mathematically collapse during inference, achieving highly competitive semantic capacity alongside massive parameter reduction with zero additional re-parameterization-induced inference latency.
3. Against Fusion Inefficiency (Synergistic Neck Reconstruction): We construct the ELANFusion_Block by integrating DSC into the Efficient Layer Aggregation Network (ELAN) paradigm. This tailored reconstruction alleviates the computational bottleneck in multi-scale aggregation, efficiently processing enriched multi-scale features with reduced GFLOPs.
4. Against Head Redundancy (Lightweight Decoupled Prediction): We tailor the Detect_DWLite head, explicitly decoupling localization and classification tasks to elimi-

nate computational redundancy while maintaining precision via Distribution Focal Loss (DFL).

2. Related Work

2.1. YOLOv11 Detection Algorithm

The YOLO series is a representative example of the single-stage object detection paradigm. It is very popular in both industry and academia because it strikes a great balance between model efficiency and detection accuracy. YOLOv11 [19] is based on the lightweight design ideas of earlier versions, such as YOLOv8. It completely redesigns the architecture and optimizes it in a modular way, which raises the standard for real-time detection. Figure 1 shows how the whole network is put together.

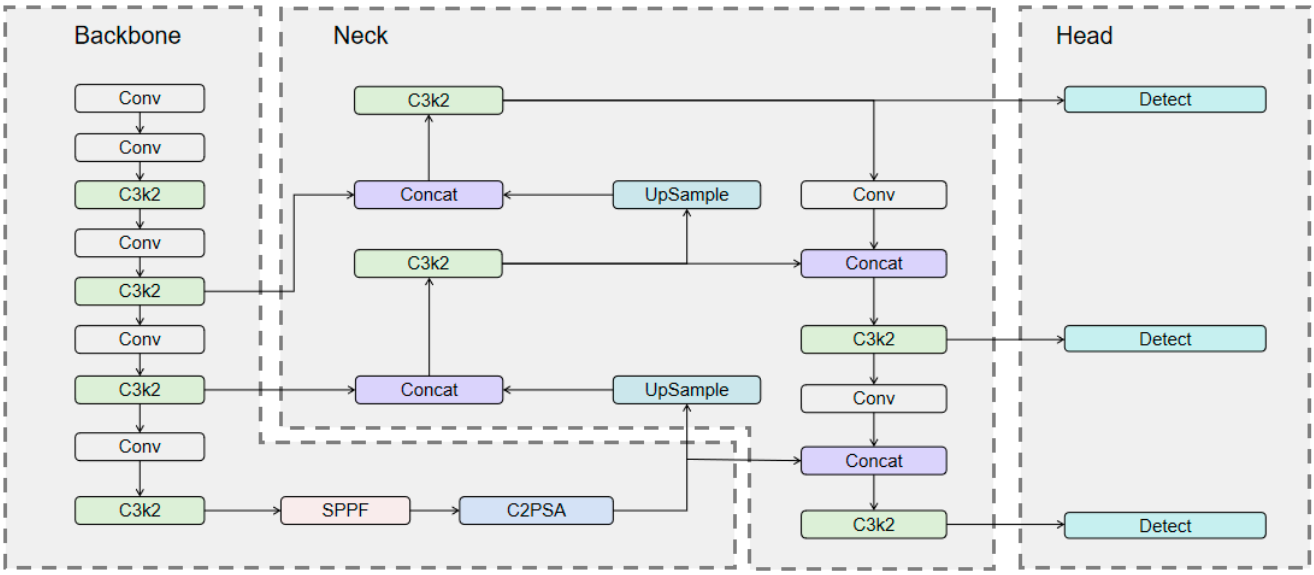


Figure 1. YOLOv11 network structure.

The architecture keeps the backbone design of Cross Stage Partial Network (CSPNet), which is good for extracting features quickly, but it replaces the old C2f module with a better C3k2 unit. The C3k2 module lets you change the receptive field in a flexible way by letting you use cross-stage partial structures with kernel sizes that you can change. This configuration facilitates feature extraction from ship targets with large scale variations in remote sensing images while requiring relatively low computational cost.

The Cross Stage Partial with Spatial Attention (C2PSA) module makes the most important distinguishing features in the feature fusion neck stand out more. This module adds spatial attention mechanisms directly to the CSP structure, which lets it focus on areas with high response while blocking out background noise. When there is a lot of noise in the environment, like sea waves and clouds, this makes it much easier to tell the difference between features. However, the dense attention mechanisms within C2PSA incur substantial computational overhead and Memory Access Costs (MACs). This heavy resource dependency makes it highly impractical for continuous deployment on power-constrained edge platforms, creating a stark dilemma between feature enhancement and hardware efficiency.

The parameter distribution and the detection head are also part of the optimization process. The model has a better Mean Average Precision (mAP) than previous versions, even though it has fewer parameters and floating point operations. These architectural features, which strike a balance between lightweight constraints and high precision, give

us a solid foundation for creating high-performance ship detection models that work well on edge devices with limited resources.

2.2. Downsampling Strategies in Modern Detectors

Downsampling is a basic operation in convolutional neural networks that makes receptive fields bigger while also reducing computational complexity. However, standard Strided Convolution often requires an irreversible loss of fine-grained spatial information. For an input feature map X, the output Y from a convolution with stride s can be written mathematically as:

$$Y_{ij} = \sum_{u,v} K_{u,v} \cdot X_{i \cdot s+u, j \cdot s+v} \quad (1)$$

where X and Y denote the input and output feature maps, respectively, and $K_{u,v}$ represents the convolution kernel weight at the spatial offset (u, v). The sampling interval s here makes it necessary to skip over neighboring pixels, which means that high-frequency information like edges and textures, which are important for finding small objects, is lost. To address this bottleneck, recent research has developed several mitigation mechanisms.

Jocher et al. [20] added the Focus module to YOLOv5 so that it could reorganize features without losing any information. It did this by using space-to-depth operations to split input tensors and combine them along the channel dimension, which kept the original data but lowered the spatial resolution. Sunkara et al. [21] improved on this by coming up with SPD-Conv, which lets space-to-depth work with any ratio and takes the stride out of later convolutions to keep features from being lost when the resolution is low and the objects are small.

Stergiou et al. [22] proposed the SoftPool module, which utilizes an exponentially weighted sum of local activations to preserve fine-grained structural information during downsampling. Li et al. [23], conversely, employed Haar Wavelet Downsampling in Convolutional Neural Networks (CNNs) to decompose features into low- and high-frequency components for downsampling that preserves structural integrity.

To address the small size and complex backgrounds of remote sensing ship targets, we used the ADown module proposed by Wang et al. [24]. This module effectively captures texture details while preserving background context by combining average and max pooling to improve the parallel aggregation strategy. This strikes a great balance between keeping features and making the computation work faster.

2.3. Reparameterized Residual Blocks

Structural re-parameterization separates the complicated multi-branch topology needed for strong training from the simple single-path structure needed for quick inference. This is accomplished through the linear superposition property of convolution operations. Separating these stages allows the model to enhance feature representation without introducing additional computational burden during deployment. If a training block has N parallel convolution branches with compatible geometric dimensions and weights $W_1, W_2, \ldots, W_N$, these can be mathematically combined into one equivalent inference kernel W_{eq} through linear transformation:

$$W_{eq}X = \sum_{i=1}^{N} (W_i X) \quad (2)$$

Ding et al. [25] were the first to use this method in RepVGG. They built multi-branch blocks with Identity, 1×1, and 3×3 convolutions to help the gradient flow, which are then combined into a single 3×3 convolution for inference. Expanding on this, DBB [26] incorpo-

rated heterogeneous transformations, such as average pooling and multi-scale convolutions, into the re-parameterizable branches, further enriching feature extraction complexity.

However, the separate Batch Normalization (BN) layers that are added to each branch in traditional re-parameterization methods use too much GPU memory and slow down training. To address this bottleneck, Ding et al. [27] proposed RepOptimizers to incorporate priors into optimizers rather than adding training-time structures. Concurrently, Hu et al. [6] came up with OREPA. This method uses a two-stage pipeline to linearly reorganize branch components during training, which directly creates a unified target kernel. This means that intermediate feature maps do not need to be generated and stored separately. Vasu et al. [28] confirmed the effectiveness of this strategy for compact mobile models in MobileOne.

We use these changes to make the C3k2_OREPA_RS module, which adds OREPA's high-efficiency training features to YOLOv11's C3k2 backbone architecture, along with a Residual Scaling mechanism. This design aims to make deep networks better at representing semantics by changing the parameters and using residual scaling factors to make gradient propagation smoother. This leads to a dual optimization of the stability of training and the speed of inference in complicated maritime situations.

2.4. Lightweight Feature Fusion in Necks

The neck serves as the main link between the backbone and the detection heads. Its role is to combine feature maps from different hierarchies to bridge the gap between the high spatial resolution of shallow layers and the strong semantic abstractions of deep layers. Formally, the feature fusion process at scale i can be conceptualized as:

$$P_i^{out} = F\left(P_i^{in}, R\left(P_{i+1}^{in}\right)\right) \quad (3)$$

In this case, R is the set of operations that change the resolution, such as upsampling or downsampling. F is the set of operations that combine data, such as concatenation or element-wise addition. Using the Path Aggregation Network (PANet), Liu et al. [29] made the bidirectional fusion paradigm, which made standard FPN better by adding bottom-up enhancement paths. But it uses too many standard convolutions, which makes it very expensive to calculate. Ghiasi et al. [30] developed NAS-FPN to discover optimal feature fusion topologies via neural architecture search; nevertheless, its highly irregular cross-layer connectivity results in significant Memory Access Latency on edge hardware.

Li et al. [31] proposed the GSConv and Slim-Neck architectures to support efficient inference, using Channel Shuffle to reduce complexity. Xu et al. [32] used re-parameterization in RepGFPN to make interactions as efficient as possible. We made the Deep ELAN Fusion Block (ELANFusion_Block) using the ELAN's gradient path optimization idea [33] and the parametric efficiency of DSCs [34]. This module cuts down on unnecessary calculations by using a lightweight "reduce-split-aggregate" method instead of dense convolutional stacking. It also improves the gradient flow across multi-level features, which helps the model find and combine ship targets of different sizes in remote sensing images.

2.5. Lightweight Detection Heads

The detection head is the most important part of detection architectures because it turns feature maps into probabilities for semantic classes and coordinates for geometric bounding boxes. Ge et al. [35] elucidated in YOLOX that a fundamental feature conflict exists between the translation invariance necessary for classification and the translation covariance essential for regression. This is because nascent architectures mostly used parameter-shared Coupled Heads. As a result, the Decoupled Head has become the most popular design style. Li et al. [36] developed Generalized Focal Loss and DFL to en-

hance localization accuracy, successfully quantifying boundary uncertainty by representing bounding box regression as a continuous probability distribution.

Decoupling makes detection much better, but adding extra convolutional branches makes the model's parameters and computational burden higher. To solve this problem, RangiLyu made NanoDet [37] less redundant in terms of parameters and less expensive to run by sharing lightweight convolution stacks, lowering stack depth, and compressing channel dimensions. Dai et al. [38] also came up with the Dynamic Head, which uses attention mechanisms to combine scale and task awareness.

With these efficiency-focused models, DSCs are now used instead of standard convolutions as a key way to make lightweight heads. In theory, a convolutional layer with input and output channels C and kernel size $k \times k$ costs C^2k^2 to compute (not counting spatial dimensions). DSC divides this up into pointwise convolution (C^2) and depthwise convolution (Ck^2). The ratio of their computational complexity is about:

$$\frac{Ck^2 + C^2}{C^2k^2} = \frac{1}{C} + \frac{1}{k^2} \approx \frac{1}{k^2} \tag{4}$$

This means that using DSC can cut down on floating point operations by a factor of approximately k^2, which may reduce the theoretical computational burden on edge devices. This design philosophy has been widely adopted by edge-oriented detection models like PFW-YOLO [39] and LightYOLO [40]. It is the theoretical basis for how we built the high-efficiency Detect_DWLite module.

2.6. SAR-Based Maritime Surveillance as Complementary Related Work

Although the present study focuses on optical remote sensing imagery, SAR-based ship monitoring is another important branch of maritime surveillance and is reviewed here as complementary related work. Compared with optical remote sensing imagery, synthetic aperture radar (SAR) imagery has been widely investigated for maritime surveillance because of its all-weather and day-night imaging capability. To provide a broader and more balanced overview, representative SAR ship datasets and survey studies from different research groups are also considered. For example, OpenSARShip provides Sentinel-1 SAR ship samples associated with AIS information [41], while HRSID offers high-resolution SAR images for ship detection and instance segmentation [42]. In addition, recent review work has summarized the latest advances in deep-learning-based SAR image ship interpretation, including ship detection, identification, and related emerging tasks [43]. Independent lightweight SAR ship detection studies have also been reported; for instance, FASC-Net was developed for fast and lightweight multi-scale SAR ship detection under complex backgrounds [44]. Recent SAR-based studies have further explored ship monitoring from different perspectives. For example, ref. [45] introduces density knowledge mining to address quantity-aware marine vessel surveillance under uneven ship distributions, while ref. [46] formulates a panorama-population-individual progressive screening paradigm for SAR ship capture. In addition, ref. [47] investigates domain-adaptive SAR ship detection by combining Mamba-based feature modeling with pseudo-label optimization. More recently, ref. [48] proposed a triple-level speckle awareness framework for speckle-robust marine ship surveillance using satellite SAR data, further emphasizing the importance of explicitly modeling speckle effects in SAR-based maritime monitoring.

Beyond ship detection, SAR and PolSAR target recognition studies also provide useful insights into target representation under microwave imaging mechanisms. For example, scattering-mechanism-driven attention has been explored for PolSAR vehicle recognition, showing the importance of physically meaningful scattering information in target interpretation [49]. These representative studies indicate that SAR target understanding often relies

on modality-specific characteristics such as scattering mechanisms, polarization attributes, speckle effects, and structural sparsity, which are fundamentally different from optical remote sensing imagery.

Although the above SAR-oriented datasets, surveys, and methods are highly relevant to maritime surveillance, they are not directly comparable with the present work because of the substantial modality gap between SAR and optical remote sensing imagery. SAR ship detection usually needs to handle speckle noise, scattering sparsity, polarization characteristics, and domain shifts among SAR sensors, whereas optical ship detection mainly faces challenges such as complex sea-surface backgrounds, scale variation, weak small-target features, and edge-deployment constraints. Therefore, the present work focuses on lightweight optical ship detection and aims to improve the accuracy-efficiency trade-off of YOLO-based detectors. The reviewed SAR studies are complementary to our work and broaden the discussion of maritime remote sensing, but cross-modal SAR-optical ship detection and deployment evaluation are left for future research.

3. Proposed Model

3.1. Overview of the Proposed Model

We propose RepLite-YOLO, a lightweight and highly efficient detection framework, to address the inherent challenges of remote sensing ship detection, particularly the small size of targets, significant aspect ratio variations, and the chaotic nature of sea surface backgrounds. This model makes a strategic change to the standard YOLOv11 architecture, as shown in Figure 2. It optimizes the backbone, feature fusion neck, and detection head to find the best balance between detection accuracy and the limits of edge computing.

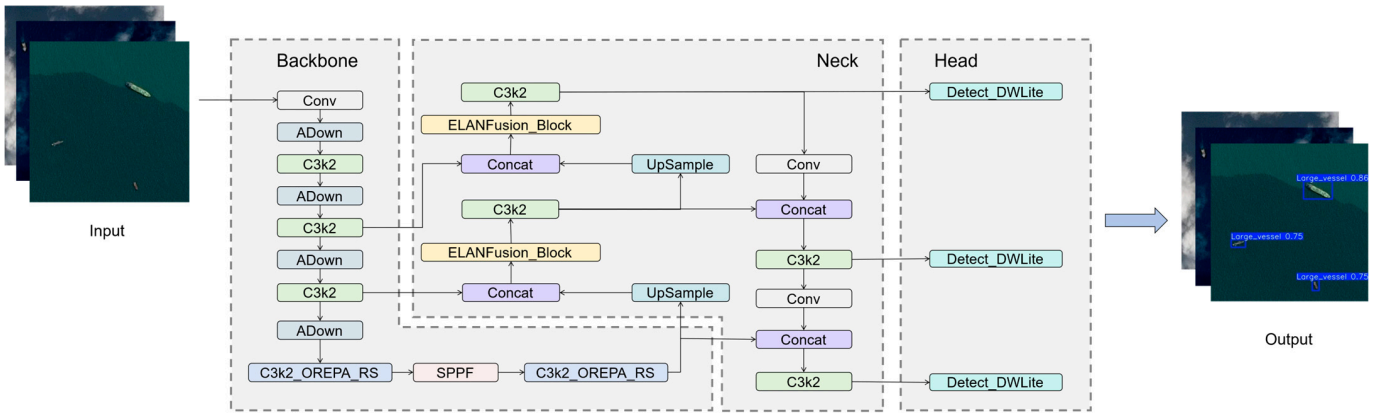


Figure 2. The overall structure of the RepLite-YOLO.

The backbone adopts the existing ADown module instead of traditional strided convolutional downsampling to stop the permanent loss of fine-grained spatial information during feature extraction. This mechanism helps preserve the edge and contour information of small ships by utilizing a shared-prefix dual-branch pathway. At the same time, the C3k2_OREPA_RS module strengthens the deep feature extraction stages. This unit improves the model's ability to tell the difference between different meanings in complex backgrounds without making the parameter envelope bigger.

The ELANFusion_Block connects the backbone and head, and it is designed to make the FPN-PAN structure stronger. This design greatly reduces computational redundancy during multi-scale feature fusion by using the efficient gradient aggregation principles of ELAN and reconstructing interaction pathways through DSCs. As a result, the network's ability to capture high-resolution, fine-grained features at the P3 and P4 levels is greatly improved.

3.2. ADown-Based Multi-Branch Downsampling

Remote sensing maritime imagery usually shows ship targets with very small scales, very high aspect ratios, and a lot of textural detail. Traditional downsampling methods, such as single-path strided convolution or aggressive max pooling, may cause irreversible information loss in the shallow layers of the backbone. This can make the hull contours and edge features that are important for accurate localization worse. RepLite-YOLO uses the ADown module [24] instead of standard downsampling operations at important backbone stages to fix this problem with the structure.

As shown in Figure 3, unlike traditional downsampling that relies on a single operator, the ADown architecture establishes a spatially-complementary feature preservation mechanism through a "Smooth-Split-Extract" paradigm. From a signal processing perspective, the module begin with a 2×2 Average Pooling operation on the input tensor $X \in \mathbb{R}^{B \times C \times H \times W}$. This acts as a spatial context smoother, generating a spatially smoothed representation X_{smooth} that dampens random local noise responses (e.g., sea waves) across the entire feature map. Subsequently, X_{smooth} is split along the channel dimension into two distinct groups, X_1 and X_2, to facilitate parallel dual-path downsampling. This process is formally described as:

$$X_{smooth} = \text{AvgPool}_{2\times2,s=1}(X) \tag{5}$$

$$X_1, X_2 = \text{Split}(X_{smooth}) \tag{6}$$

$$Y_1 = \text{Conv}_{3\times3,s=2}(X_1) \tag{7}$$

$$Y_2 = \text{Conv}_{1\times1}(\text{MaxPool}_{3\times3,s=2}(X_2)) \tag{8}$$

$$Y = \text{Concat}(Y_1, Y_2) \tag{9}$$

Within this residual extraction phase, the two parallel pathways fulfill heterogeneous roles to reconstruct features. The first branch (Y_1) utilizes a strided 3×3 convolution to capture local spatial contexts. Concurrently, the second branch (Y_2) employs a 3×3 Max Pooling operation to actively extract the most salient, robust structural edge activations (e.g., rigid ship edges) that survive the initial smoothing, followed by a 1×1 pointwise convolution for channel alignment. By concatenating these complementary representations (Y_1 and Y_2), the ADown module equips the network with both noise-free background context and sharp structural integrity without irreversible information loss.

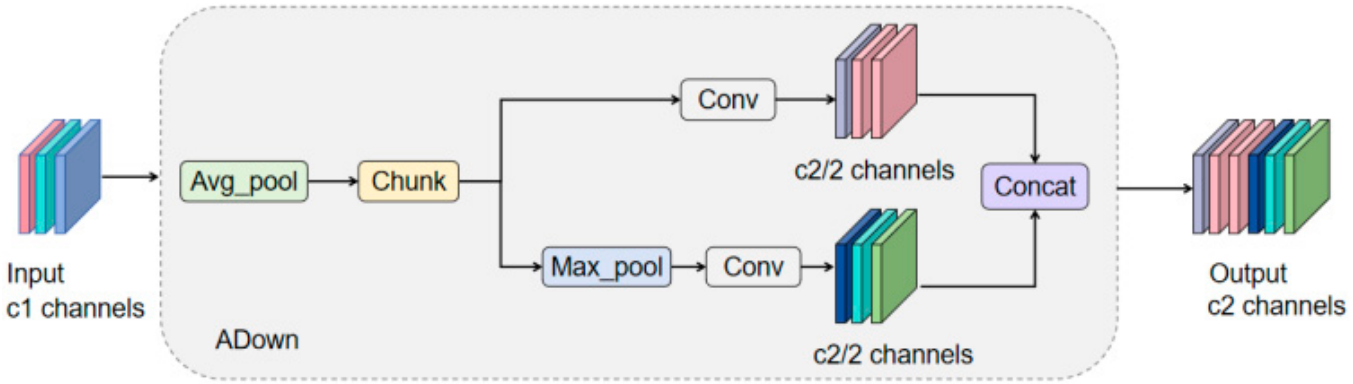


Figure 3. ADown structure.

To empirically validate this complementary mechanism, we visualize the intermediate feature maps of the two parallel branches in Figure 4. The distinct visual contrast between the branches confirms their heterogeneous roles. As shown in Figure 4c, the max-pooling branch acts as a non-linear strong-response feature extractor; while it retains

some local wave peaks, it captures the highly salient, crisp geometric boundaries of the ship hulls that would otherwise be smoothed out. Conversely, Figure 4b demonstrates that the convolutional branch functions as a context aggregator; it suppresses residual background sea clutter to almost zero, ensuring clean and robust spatial localization of the targets. By concatenating these two spatially-complementary representations, the ADown module effectively equips the network with both noise-robust spatial context and salient structural edge information, resolving the feature degradation dilemma for small-scale remote sensing targets.

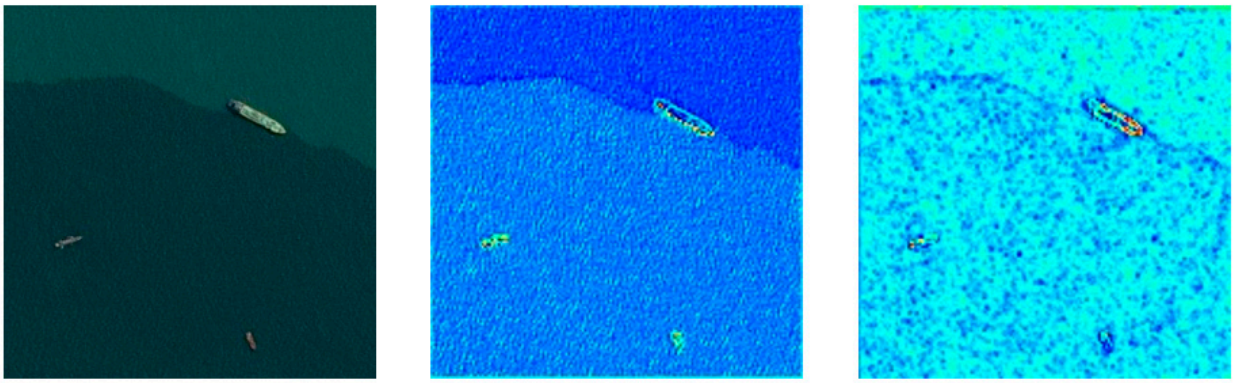


Figure 4. Feature visualization of the parallel branches within the ADown module. (**a**) The original optical input image. (**b**) The feature map of the convolutional branch. (**c**) The feature map of the max-pooling branch.

3.3. Hybrid Backbone with Re-Parameterized Structural Streamlining

Distinguishing vessels from complex coastal interference (e.g., docks, containers, and building shadows) strictly requires robust deep semantic features. However, directly appending heavy attention modules or dense convolutional layers fundamentally conflicts with the strict power and memory constraints of edge deployment.

To resolve this contradiction, we strategically restructure the deep semantic extraction stages of the YOLOv11 backbone to optimize the trade-off between semantic depth and parameter efficiency. To meet the harsh limits of resource-constrained environments, we first execute an aggressive compression in the P5 stage: the channel dimension is halved from 1024 to 512, and the resource-intensive C2PSA module from the baseline is completely removed. Empirically, as demonstrated in our controlled ablation (Section 4.6.2), this naive compressed configuration (C3k2) inevitably leads to semantic degradation, resulting in a slight decline in detection accuracy (mAP). To recover this loss in representational capacity without re-introducing inference overhead, we design the C3k2_OREPA_RS module as a customized replacement for the deep layers. By integrating OREPA, it constructs a complex, multi-branch topology during the training phase. This over-parameterization enforces rich gradient flows that help the network absorb the compressed parameter budget, learning discriminative coastal boundaries and abstract semantics from the maritime datasets. For inference, these branches mathematically collapse into a single-path structure, eliminating all training-phase overhead. As will be quantitatively detailed in Section 4.6.2, this design functions as a high-fidelity structural compressor: it pushes parameter efficiency significantly beyond what simple channel reduction achieves, while suffering no statistically meaningful accuracy penalty typical of aggressive lightweight strategies.

Traditional structural re-parameterization (e.g., RepVGG [25]) improves inference performance, but keeping multi-branch intermediate feature maps during training adds

computational and memory costs that grow linearly with the number of branches. To make things easier, we use the OREPA mechanism to improve the efficiency of end-to-end training.

OREPA moves the process of simplifying structures up to the training optimization phase, as shown in Figure 5 and explained in Figure 6. OREPA uses linear transformations to separate BN parameters and remove nonlinear barriers. This makes it possible to directly map and combine parallel convolution kernels of different sizes (e.g., 3×3, 1×1) into a single parameter space.

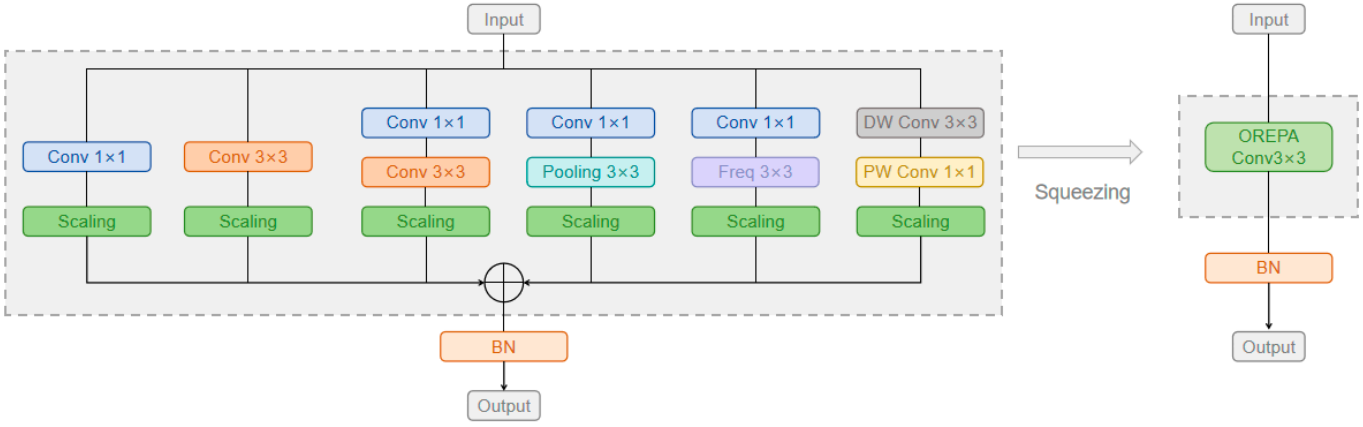


Figure 5. Structural diagram of the OREPA module design.

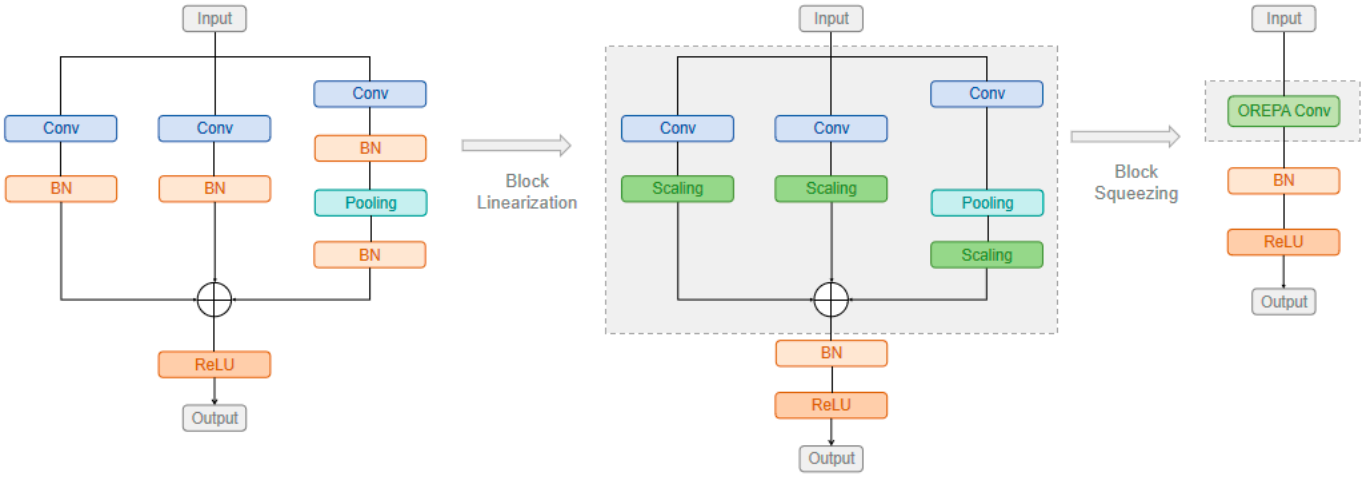


Figure 6. OREPA basic flow.

The diagram shows the basic CBS unit, the Bottleneck_OREPA_RS unit, which combines OREPA with residual scaling, and the C3k_OREPA_RS module, which uses the Cross Stage Partial (CSP) architecture. The right panel shows the different ways that data can move through the C3k2 module depending on how it is set up (c3k = False vs. c3k = True).

Based on this mechanism, we rebuild the Bottleneck unit of YOLOv11 to suggest the C3k2_OREPA_RS module, which is shown in Figure 7.

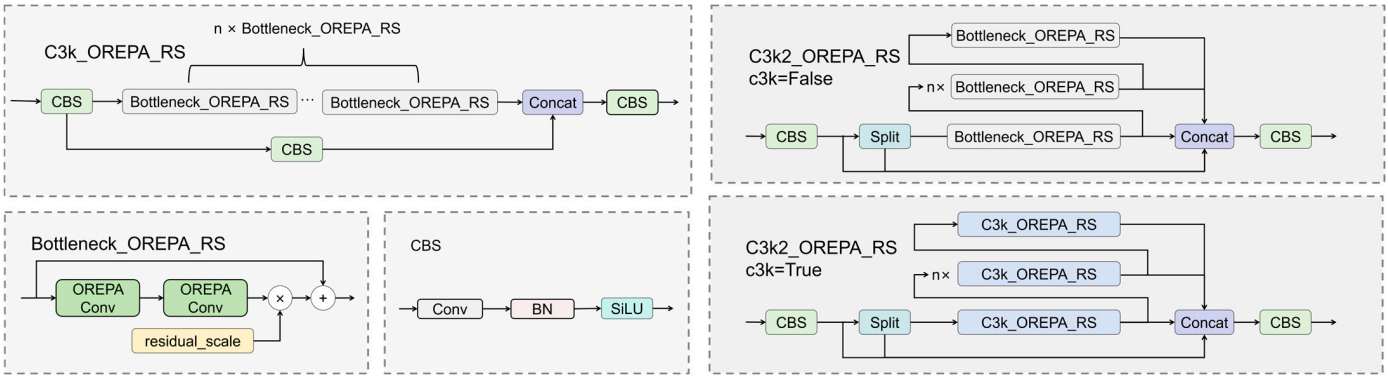


Figure 7. Structural diagram of the improved C3k2 module and its parts.

During training, the OREPA block expands the optimization space through multiple linear convolutional components. Since these components are aggregated before nonlinear activation, they can be algebraically transformed into an equivalent single-path convolution for inference. To clarify this conversion process, we explicitly formulate the equivalent kernel construction used before deployment.

For the (i)-th convolutional component, let (W_i) and (b_i) denote the convolution kernel and bias, respectively. If this component is followed by a batch normalization layer with parameters (γ_i), (β_i), running mean (μ_i), running variance (σ_i^2), and numerical constant (ϵ), the convolution and BN can be fused as:

$$\hat{W}_i = \frac{\gamma_i}{\sqrt{\sigma_i^2 + \epsilon}} W_i \tag{10}$$

$$\hat{b}_i = \beta_i + \frac{\gamma_i}{\sqrt{\sigma_i^2 + \epsilon}} (b_i - \mu_i) \tag{11}$$

For branches with different kernel forms, a transformation operator $(T_i(\cdot))$ is used to align them into the target 3×3 kernel space. Specifically, a 1×1 kernel is zero-padded to the center of a 3×3 kernel, an identity branch is represented as a Dirac delta kernel, and pooling-based or sequential linear branches can be converted into equivalent convolutional kernels. Therefore, the final equivalent kernel and bias are obtained as:

$$W_{eq} = \sum_{i=1}^{N} \alpha_i T_i(\hat{W}_i) \tag{12}$$

$$b_{eq} = \sum_{i=1}^{N} \alpha_i \hat{b}_i \tag{13}$$

where (N) denotes the number of linear branches, and (α_i) is the learnable scaling coefficient of the (i)-th branch. After this conversion, the computation of the OREPA block can be formulated as a standard 3×3 convolution function:

$$\widetilde{X}_k = \sigma(\mathrm{BN}(\mathrm{DWConv}_{3\times3}(X_{k-1}))) \tag{14}$$

Thus, the multi-branch topology is used to improve representational capacity and gradient flow during training, while the deployed inference graph contains only a plain single-path 3×3 convolution. This is why the proposed C3k2_OREPA_RS introduces no additional re-parameterization-induced latency during inference.

The module also uses a Residual Scaling (RS) strategy to help with problems with gradient instability or vanishing during deep semantic propagation. We model the final output Y as:

$$Y^{(l)} = X^{(l)} + \beta^{(l)} \cdot F(l)(X(l); W_{eq}^{(l)}) \tag{15}$$

In this equation, $X^{(l)}$ denotes the input feature of the l-th Bottleneck_OREPA_RS module, and $F^{(l)}(\cdot)$ represents the transformation function of its re-parameterized branch. $\beta^{(l)}$ is not a fixed manually tuned hyperparameter, but a learnable residual scaling scalar specific to the l-th bottleneck. In our implementation, each $\beta^{(l)}$ is independently initialized to 1.0 and is updated through back-propagation together with the convolutional weights and other learnable parameters. The residual scaling parameters are not shared across different Bottleneck_OREPA_RS modules. This design is intended to help stabilize residual learning by allowing each bottleneck to adaptively regulate the amplitude of its re-parameterized transformation branch.

3.4. ELANFusion_Block-Enhanced Neck for Efficient Multi-Scale Feature Fusion

In remote sensing maritime surveillance, targets are distinguished by extreme scale variations. High-resolution shallow features are essential to capture morphological details like hull edges, while deep semantic features are strictly required to suppress background sea clutter. In our proposed architecture, the synergistic integration of the frequency-preserving ADown module (shallow layers) and the semantic-compensated C3k2_OREPA_RS backbone (deep layers) creates a highly enriched, domain-specific feature flow. However, fusing these enriched multi-scale features using traditional FPN with standard dense convolutions would inevitably trigger an explosion in computational cost (GFLOPs), directly violating the strict constraints of edge hardware.

To resolve this systematic bottleneck, we construct the ELANFusion_Block by integrating DSC into the ELAN paradigm. Rather than a naive operator substitution, this reconstruction is tailored to balance feature retention and efficiency: as shown in Figure 8, it explicitly replaces dense convolutional layers with DSCs to ensure parametric frugality, while strictly preserving ELAN's gradient path optimization to avoid feature degradation during multi-scale interaction. Consequently, this synergistic neck design acts as an efficient feature digestor. It successfully bridges the high-frequency structural details from ADown and the deep semantics from OREPA, minimizing computational redundancy without sacrificing the multi-scale representations required for varying ship sizes.

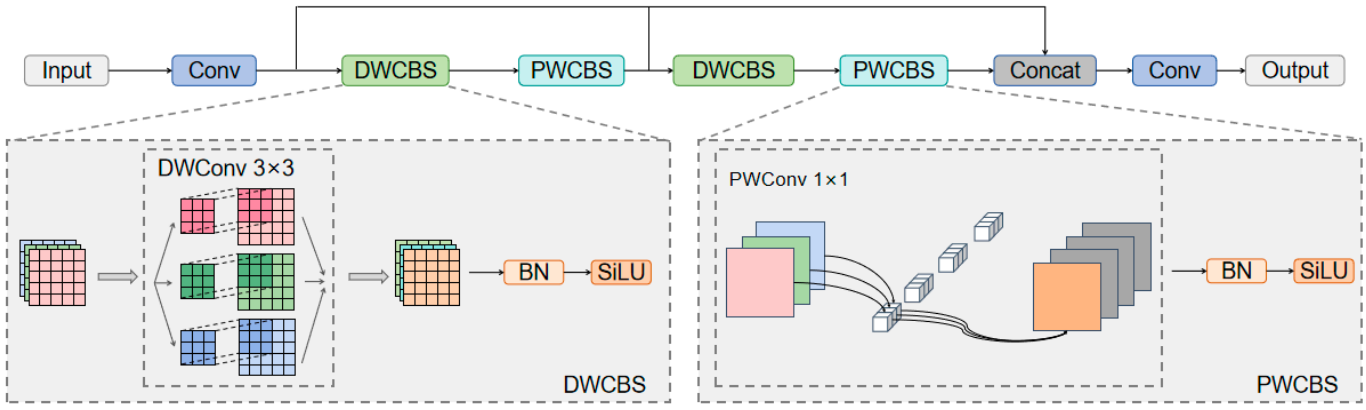


Figure 8. A schematic diagram of the ELANFusion_Block.

In formal terms, the ELANFusion_Block takes an input feature map $X \in R^{B\times c_1\times H\times W}$ and a target output channel count c_2. It then uses a 1×1 convolution to remove redundant information and map the input to an intermediate channel space c_*:

$$c_* = \lfloor e \cdot c_2 \rfloor \tag{16}$$

$$X_0 = \mathrm{Conv}_{1\times1}(X) \in R^{B\times c_*\times H\times W} \tag{17}$$

In this case, $e \in (0, 1]$ is the channel scaling factor, which is set to 0.5 in this work. Then, the module uses a cascade of n lightweight sub-blocks to gradually improve the features. Each sub-block uses a DSC structure to separate spatial feature extraction from channel-wise information interaction. The function of the k-th sub-block is defined as:

$$\widetilde{X}_k = \sigma(\mathrm{BN}(\mathrm{DWConv}_{3\times3}(X_{k-1}))) \tag{18}$$

$$X_k = \sigma\big(\mathrm{BN}\big(\mathrm{PWConv}_{1\times1}\big(\widetilde{X}_k\big)\big)\big) \tag{19}$$

In this formulation, feature extraction sequentially combines a 3×3 depthwise convolution ($DWConv_{3\times3}$) to capture spatial context and a 1×1 pointwise convolution ($PWConv_{1\times1}$) to merge features across channels. σ and BN denote the SiLU activation function and batch normalization, respectively. This decomposition strategy greatly reduces floating point operations while keeping the features expressive.

To improve the flow of gradients and keep multi-level features, the module uses an ELAN-style cascaded aggregation strategy. All outputs from the sub-blocks are aggregated and concatenated along the channel dimension. Then, a projection layer restores them to the target dimension:

$$X_{\mathrm{Concat}} = \mathrm{Concat}([X_0, X_1, \ldots, X_n]) \in \mathbb{R}^{B\times(n+1)c_*\times H\times W} \tag{20}$$

$$Y = \mathrm{Conv}_{1\times1}(X_{\mathrm{Concat}}) \in \mathbb{R}^{B\times c_2\times H\times W} \tag{21}$$

The cooperation between DSC and ELAN is not a simple replacement of standard convolutions. Instead, it follows a divide-and-aggregate principle. In each lightweight sub-block, the depthwise convolution first performs channel-wise spatial filtering, which allows each semantic channel to independently capture local ship structures, such as hull edges, elongated contours, and small-scale texture responses. Then, the pointwise convolution performs cross-channel interaction to recombine the spatially filtered responses from different semantic channels. In this way, DSC "digests" the enriched multi-scale features from the ADown-enhanced shallow layers and the OREPA-enhanced deep layers with substantially reduced computational cost.

Meanwhile, the ELAN-style aggregation prevents feature degradation caused by excessive lightweight decomposition. Instead of only using the output of the last DSC block, ELAN concatenates $X_0, X_1, \ldots, X_n$, which correspond to features with different transformation depths. Therefore, shallow details, intermediate spatial responses, and deeper semantic abstractions are jointly preserved before the final projection layer. From the perspective of gradient propagation, the concatenation operation provides direct gradient paths from the output projection to each intermediate feature state, allowing both short-path and long-path information flows. This design maintains multi-scale feature diversity while avoiding the bottleneck that would occur if all information were forced to pass through a single lightweight convolutional chain.

Compared with dense convolutional fusion, the proposed DSC-ELAN design significantly reduces the dense computational burden associated with simultaneous spatial-channel mixing while retaining the multi-depth feature aggregation ability of ELAN. Consequently, the ELANFusion_Block balances parameter efficiency and feature preservation, making it suitable for remote sensing ship detection where small targets require both fine spatial details and robust semantic context.

At the concatenation stage, where high-level semantics from the backbone merge with mid-to-shallow details (e.g., P5 $\rightarrow$ P4), the "Top-down + ELANFusion_Block + C3k2" setup makes sure that high-resolution feature maps (P3, P4) are filled with both strong semantic context and fine-grained structural details at the same time. This architecture is different

from designs that use complicated attention mechanisms or Transformer modules. Instead, it only uses universal convolution and concatenation operators. This effectively minimizes computational redundancy, exhibiting strong theoretical feasibility for deployment on resource-constrained edge devices.

3.5. Lightweight Decoupled Detect_DWLite Head with Depthwise Separable Convolutions

In object detection architectures, the prediction head bridges feature maps to final categorical probabilities and geometric coordinates, typically dominating the model's parameter budget. For remote sensing ship detection, accurately localizing small vessels necessitates processing high-resolution feature maps. However, feeding these dense features into standard coupled or heavy decoupled heads incurs a prohibitive computational overhead, directly violating the latency and power constraints of edge platforms.

To resolve this specific bottleneck, we tailor the Detect_DWLite head. Rather than adopting generic dense components, this customized head employs a "Compression-Depthwise-Projection" strategy to explicitly decouple the translation-invariant classification task (discriminating ships from sea clutter) from the translation-covariant regression task (precisely localizing rigid hull boundaries). By strategically utilizing DSCs across its parallel branches, the tailored Detect_DWLite heavily prunes redundant parameters and GFLOPs. This streamlined design ensures that the high-precision localization benefits of DFL are fully maintained for tiny ships, but at a fraction of the computational cost of the baseline YOLOv11 head.

Figure 9 shows how Detect_DWLite rebuilds the regression and classification branches while keeping the DFL system. The module has a "Compression-Depthwise Extraction-Linear Projection" bottleneck architecture that was made to reduce the number of redundant parameters. We first use a channel scaling strategy to map an input feature map $X_i \in R^{B\times C_i\times H_i\times W_i}$ at scale i to an intermediate dimension C_{mid}. To do this, set $C_{mid} = \max(16, \lfloor \gamma \cdot C_i \rfloor)$, where $\gamma = 0.25$ is the compression ratio. Then, the feature stream divides into the Regression (Reg) and Classification (Cls) branches, which are handled like this:

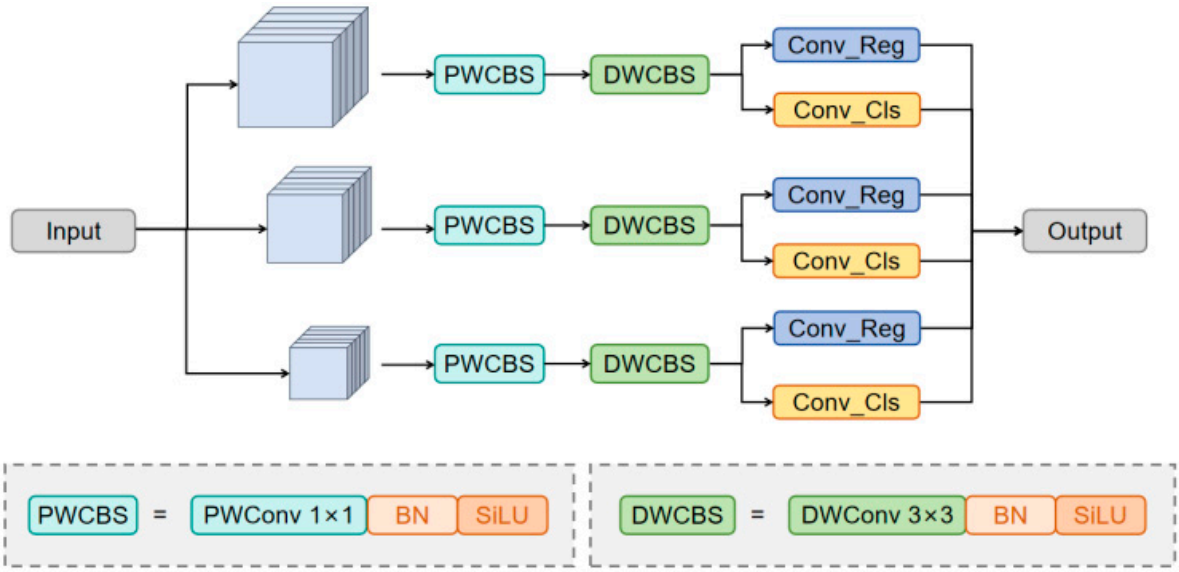


Figure 9. The overall structure of the proposed lightweight detection head, Detect_DWLite.

Regression Branch:

$$M_i^{reg} = \sigma(BN(Conv_{1\times1}(X_i))) \quad (22)$$

$$Z_i^{reg} = \sigma\left(BN\left(DWConv_{3\times3}\left(M_i^{reg}\right)\right)\right) \quad (23)$$

$$R_i = f_{out}^{reg}\left(Z_i^{reg}\right) \in \mathbb{R}^{B\times4M\times H_i\times W_i} \quad (24)$$

Classification Branch:

$$M_i^{cls} = \sigma(BN(Conv_{1\times1}(X_i))) \tag{25}$$

$$Z_i^{cls} = \sigma\left(BN\left(DWConv_{3\times3}\left(M_i^{cls}\right)\right)\right) \tag{26}$$

$$P_i = f_{out}^{cls}\left(Z_i^{cls}\right) \in \mathbb{R}^{B\times N_c\times H_i\times W_i} \tag{27}$$

Here, we map the input feature map X_i to a reduced intermediate dimension using a 1×1 convolution ($Conv_{1\times1}$). Subsequently, a 3×3 depthwise convolution ($DWConv_{3\times3}$) extracts spatial context, followed by Batch Normalization (BN) and SiLU activation (σ) to minimize redundancy. Finally, f_{out}^{reg} denotes the final 1×1 pointwise convolution layer projecting the features to the target regression dimensions ($4 \times M$). The classification branch generates class probabilities P_i via an analogous transformation to output N_c classes.

The DFL integration mechanism decodes the regression output into final coordinates to reduce boundary ambiguity and facilitate small-ship localization. For a specific spatial location (pixel) p on the feature map, the predicted distance $\hat{d}_{p,j}$ for each bounding box edge $j \in \{l, t, r, b\}$ is determined as the expectation of a Softmax distribution:

$$\hat{d}_{p,j} = \sum_{k=0}^{M-1} S(R_{p,j})_k \cdot k \tag{28}$$

where $S(\cdot)$ is the Softmax function and M is the number of intervals for discretization. During training, the prediction head is optimized using a composite detection loss that jointly supervises bounding-box localization, category prediction, and distribution-based boundary regression. The overall detection loss is formulated as:

$$\mathcal{L}_{det} = \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{dfl}\mathcal{L}_{dfl} \tag{29}$$

where $\mathcal{L}_{box}$ denotes the IoU-based bounding-box regression loss, $\mathcal{L}_{cls}$ denotes the classification loss, and $\mathcal{L}_{dfl}$ denotes the Distribution Focal Loss used to supervise the discrete bounding-box distance distribution. In our experiments, the corresponding weights were set to $\lambda_{box} = 7.5$, $\lambda_{cls} = 0.5$, and $\lambda_{dfl} = 1.5$, respectively. This same loss configuration was used for YOLOv11n and all modified models to ensure a fair comparison.

To accelerate training convergence, we also use a logarithmic bias initialization strategy at the end of the classification branch to address the severe foreground–background imbalance in remote sensing images.

Detect_DWLite reduces the computational cost by approximately 25.4% compared to the YOLOv11n baseline without compromising precision, by using DSCs and separating the prediction tasks. This design works together with the previous ADown backbone and ELANFusion neck to make RepLite-YOLO better aligns with the low-power and low-storage requirements of edge computing devices while still providing high-precision localization.

4. Experiment

4.1. Experiment Dataset

The Vessel Dataset [50], released on the Roboflow platform (Roboflow, Inc., Des Moines, IA, USA) on 6 September 2021, is designed to detect small vessels from low-altitude drone or satellite views. The dataset contains 3188 remote sensing images and follows the official Roboflow split, including 2232 training images, 478 validation images, and 478 testing images. During training, the training subset was used for parameter optimization, and the validation subset was used for early stopping and model selection.

The testing subset was kept independent and was used only for final evaluation. The most important thing about it is that it has a strong Small-object Dominance distribution. Figure 10 shows that the images show nearshore waters and port berths, which makes the background look complicated. Figure 11 shows the results of a statistical analysis that shows that small vessels (pixel dimensions < 32 × 32) make up an amazing 82% of the cases, while medium and large vessels only make up 17.5% and 0.5%, respectively. This dataset is a great way to test a model's perceptual acuity when it comes to small objects because the scale is so uneven.



Figure 10. Representative samples from the Vessel Dataset.

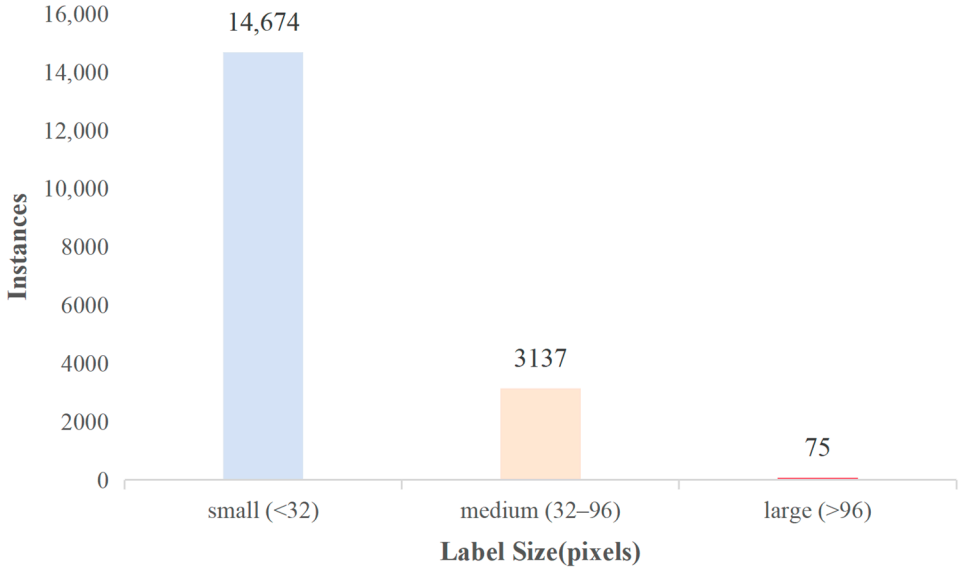


Figure 11. Quantity of targets of different sizes within the Vessel Dataset.

The images illustrate the diversity of maritime scenarios, including (from left to right) complex coastal backgrounds, cloud occlusion, open sea with wake trails, and low-contrast conditions caused by haze.

To conduct a more thorough assessment of the model's overall efficacy across various scenarios, we employed the colab_version2 iteration of the Ship Detection Dataset [51]. This repository has 12,849 labeled examples in 4126 remote sensing images, and it is split into three sets: a training set with 2882 images, a validation set with 826 images, and a testing set with 418 images. The samples in Figure 12 show that this variety of environments provides a useful test of the model's feature extraction and generalization ability under complex interference. Figure 13 shows that small vessels still make up the majority (54.34%), but the percentages of medium (25.93%) and large (19.73%) ships are much higher than in the Vessel Dataset. This dataset also includes a wider range of environmental factors, including sea states from calm waters to micro-waves, as well as weather conditions like clear skies, overcast weather, and thin fog.

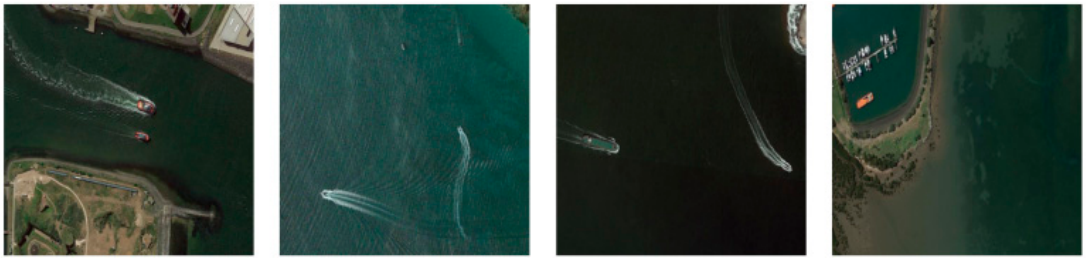


Figure 12. Representative samples of ship detection images.

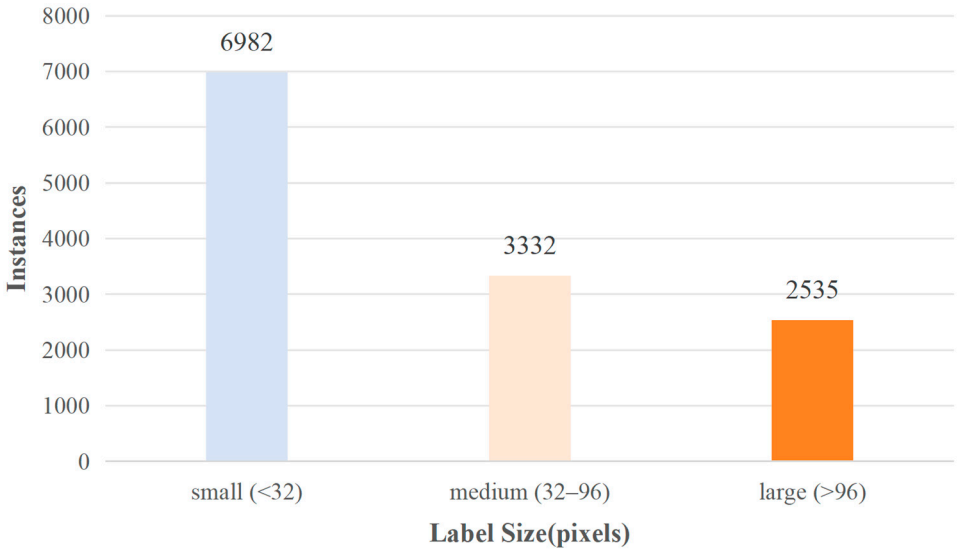


Figure 13. Quantity of targets of different sizes within the Ship Detection Dataset.

The figure demonstrates the dataset's diversity, covering various scenarios such as narrow inland waterways, open seas with distinct wake patterns, and complex harbor environments with docked vessels.

4.2. Experiment Setup

All empirical evaluations were conducted on the AutoDL cloud computing platform under a Linux Ubuntu environment. The GPU resource available in the experiment environment was an NVIDIA vGPU with 32 GB memory. Since the platform provides a virtualized GPU profile and does not expose the underlying physical GPU model to users, the exact physical GPU type could not be retrieved. Table 1 lists the hardware and software configuration used in our experiments.

Table 1. Experimental Software and Hardware Configuration.

Computer Configurations	Version
GPU	NVIDIA vGPU-32GB
CPU	Intel(R) Xeon(R) Gold 6430
Cuda	13.0
Python	3.10.14
Pytorch	2.2.2
Torchvision	0.17.2

Although the underlying physical GPU model was unavailable, all baseline and proposed models were trained and evaluated under the same virtualized hardware environment. Therefore, the fairness of the model-to-model comparison is maintained. In

addition, the primary efficiency metrics reported in this work, namely parameter count and GFLOPs, are hardware-independent model complexity indicators.

The Stochastic Gradient Descent (SGD) optimizer controlled the parameter updates during training. The batch size was set to 32 to find the best balance between GPU memory usage and gradient stability. Data loading was performed in parallel using 12 worker threads to accelerate the training pipeline. The training took 300 epochs. The initial learning rate was set to 0.01 and was updated using a cosine learning-rate schedule, with the final learning-rate factor set to 0.01. The weight decay coefficient was set at 0.0005 to reduce overfitting. An Early Stopping protocol was put in place to save computer power. If validation performance did not improve over a continuous window of 60 epochs, this protocol would automatically stop the training process. The overall detection loss consisted of bounding-box regression loss, classification loss, and Distribution Focal Loss. The corresponding loss weights were set to 7.5, 0.5, and 1.5, respectively. Specifically, the official validation subset of each dataset was used for early stopping and model selection, while the testing subset was kept independent and used only for final evaluation. Table 2 lists the complete hyperparameter settings used in the experiments.

Table 2. Experimental Parameter Settings.

Parameter	Setting
Batch size	32
Epochs	300
The initial learning rate	0.01
workers	12
Optimizer	SGD
Weight decay	0.0005
Early stop rounds	60

For data augmentation, Mosaic augmentation was enabled during training with mosaic = 1.0. To improve training stability in the final stage, Mosaic augmentation was disabled in the last 10 epochs by setting close_mosaic = 10. In addition, standard color and geometric augmentations were applied, including HSV augmentation (hsv_h = 0.015, hsv_s = 0.7, hsv_v = 0.4), random translation (translate = 0.1), random scaling (scale = 0.5), and random horizontal flipping (fliplr = 0.5). Random rotation, shear transformation, perspective transformation, vertical flipping, MixUp, and copy-paste augmentation were not used, with degrees = 0.0, shear = 0.0, perspective = 0.0, flipud = 0.0, mixup = 0.0, and copy_paste = 0.0. No independent random cropping operation was applied. The same augmentation configuration was used for all compared models to ensure that the reported performance differences were caused by architectural changes rather than inconsistent training protocols.

It should be emphasized that the YOLOv11n baseline results reported in this paper were not taken from the original YOLOv11 paper or any official benchmark. Instead, YOLOv11n was retrained by us under the same experimental protocol as the proposed RepLite-YOLO. Specifically, all compared models were trained and evaluated using the same dataset splits, input image size, optimizer, learning-rate schedule, early-stopping strategy, loss weights, and data augmentation settings. Therefore, the reported comparison reflects a controlled evaluation under identical training conditions, rather than a comparison between results collected from different sources.

To address the possible influence of run-to-run variance, we further conducted five independent training runs for YOLOv11n and RepLite-YOLO using different random seeds. All runs followed the same dataset splits, input image size, optimizer, learning-rate schedule, early-stopping strategy, loss weights, and data augmentation settings. For mAP@50,

we report the mean value, sample standard deviation, and 95% confidence interval. In addition, paired *t*-tests were performed between YOLOv11n and RepLite-YOLO under matched random seeds to determine whether the observed mAP@50 differences were statistically significant. All values in the statistical table are rounded to three decimal places.

4.3. Experiment Evaluation Metrics

To fully evaluate the operational effectiveness of RepLite-YOLO in the area of remote sensing ship detection, we use a two-part evaluation framework that looks at both detection accuracy and computational efficiency.

Precision (P), Recall (R), and mAP are the most important measures of detection accuracy. In particular, P measures how well the model can reduce false alarms in the background, and R measures how well it can find all of the ship targets. Because maritime surveillance has a zero-tolerance policy for missed detections, it is very important to keep a high Recall rate to ensure safe navigation. We define the calculations for these metrics as follows:

$$P = \frac{TP}{TP + FP} \tag{30}$$

$$R = \frac{TP}{TP + FN} \tag{31}$$

where TP, FP, and FN stand for the number of true positives, false positives, and false negatives, respectively. To get mAP, you add up the areas under the Precision–Recall (P-R) curve and average them across all categories. The mathematical expressions are:

$$AP = \int_0^1 P(R)\, dR \tag{32}$$

$$mAP = \frac{1}{N_c} \sum_{i=1}^{N_c} AP_i \tag{33}$$

where N_c denotes the total number of target categories, and AP_i represents the Average Precision for the i-th category.

In this work, we adhere to the standard object detection evaluation protocol without modifying the official Ultralytics YOLOv11 evaluation logic. To rigorously determine True Positives (TP) and False Positives (FP), all predicted bounding boxes are first sorted in descending order based on their confidence scores. For a given Intersection over Union (IoU) threshold, a prediction is counted as a TP only if it has the correct class label and its IoU with an unmatched ground-truth box is greater than or equal to that threshold. Each ground-truth box can be matched to at most one prediction. Duplicate predictions assigned to an already matched ground-truth box, or predictions failing to meet the IoU threshold, are rigidly counted as FPs.

Consequently, the metric mAP@50 denotes the mean AP computed at a single IoU threshold of 0.50. The metric mAP@50-95 represents the average of AP values computed over ten distinct IoU thresholds ranging from 0.50 to 0.95 with a step size of 0.05.

We use Parameter count (Params) and GFLOPs to see how well a model works by looking at how complicated it is. The former reflects storage requirements, whereas the latter reflects the theoretical computational complexity required for inference. These numbers are very important for making sure that a model works with the strict resource limits that come with resource-constrained environments.

4.4. Ablation Experiment

We conducted a systematic ablation study using the Vessel and Ship Detection Datasets to quantitatively separate the individual contributions and synergistic interactions of the

core components in RepLite-YOLO. The architecture was gradually improved by adding the ADown downsampler, C3k2_OREPA_RS feature extractor, ELANFusion_Block neck, and Detect_DWLite head. It all started with YOLOv11n. Table 3 shows the detailed metrics that were gathered.

Table 3. Ablation Experiment (a: ADown, b: C3k2_OREPA_RS, c: ELANFusion_Block, d: Detect_DWLite). Check marks (√) indicate the use of the corresponding modules, while blank cells indicate that the modules are not used. The row without check marks under a–d represents the original YOLOv11n baseline without any of the proposed modules.

Dataset	a	b	c	d	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
Vessel Dataset					0.752	0.718	0.749	0.394	2.58	6.3
	√				0.763	0.705	0.749	0.398	2.10	5.1
		√			0.746	0.723	0.753	0.393	2.06	5.9
			√		0.763	0.712	0.755	0.398	2.27	6.2
				√	0.751	0.713	0.752	0.390	2.21	4.7
	√		√		0.764	0.700	0.753	0.389	1.92	5.2
	√	√			0.757	0.717	0.754	0.397	1.57	4.7
		√		√	0.748	0.715	0.754	0.393	1.68	4.3
	√	√		√	0.755	0.724	0.758	0.391	1.20	3.1
	√		√	√	0.780	0.708	0.754	0.394	1.60	3.6
	√	√	√	√	0.757	0.729	0.761	0.402	1.10	3.2
Ship Detection Dataset					0.873	0.764	0.853	0.458	2.58	6.3
	√				0.861	0.785	0.860	0.469	2.10	5.1
		√			0.874	0.784	0.855	0.453	2.06	5.9
			√		0.853	0.799	0.860	0.463	2.27	6.2
				√	0.887	0.767	0.850	0.450	2.21	4.7
	√		√		0.863	0.781	0.858	0.460	1.92	5.2
	√	√			0.889	0.776	0.855	0.464	1.57	4.7
		√		√	0.877	0.789	0.857	0.452	1.68	4.3
	√	√		√	0.883	0.779	0.864	0.463	1.20	3.1
	√		√	√	0.905	0.764	0.865	0.465	1.60	3.6
	√	√	√	√	0.900	0.783	0.866	0.459	1.10	3.2

Empirical evidence underscores the critical role of each module in advancing the efficiency-accuracy frontier. The main way to cut down on redundant computations was to add the Detect_DWLite head. This change alone cut the Vessel Dataset's computational cost from 6.3 to 4.7 GFLOPs without lowering detection accuracy. When it came to feature enhancement, the ELANFusion_Block was great at combining features from different scales. This was especially clear in the Ship Detection Dataset, where the target scales change a lot. The module made a big difference in single-component performance, raising mAP_{50} to 0.860 and proving that it can keep important semantic context in complicated situations.

When these modules were integrated simultaneously, the parameter count decreased and detection performance improved. The fully assembled RepLite-YOLO architecture did better than the YOLOv11n baseline by 1.2% and 1.3% on the Vessel and Ship Detection Datasets, with the latter getting a precision of 0.900. The most important thing about these results is how well they work: the total number of parameters went down by 57.4% (from 2.58 M to 1.10 M), and the number of floating-point operations went down by 49.2% (from 6.3 GFLOPs to 3.2 GFLOPs). These numbers show that the suggested lightweight architecture got rid of extra calculations and made it easier to find things in images taken from a distance, which are often very complicated.

To provide a qualitative interpretation of the feature responses, we employed Grad-CAM to generate attention heatmaps of deep features and compared them with those of the YOLOv11n baseline (Figure 14). It should be noted that Grad-CAM provides qualitative visual evidence and cannot independently prove a physical decoupling or

frequency-domain mechanism. In scenarios dominated by strong wave textures (Row 1), the baseline YOLOv11n tends to produce more dispersed activation responses over sea clutter regions. In contrast, RepLite-YOLO shows more compact activation regions around ship hulls. This observation visually supports that the ADown-based downsampling strategy may help reduce background interference and preserve ship-related structural responses. Furthermore, in scenes with dense small targets (Row 3), the baseline exhibits relatively blurred activation boundaries, while RepLite-YOLO produces more concentrated heatmaps around small-vessel regions. This observation is consistent with the quantitative results and suggests that the C3k2_OREPA_RS backbone and ELANFusion_Block may help maintain useful target-related responses under lightweight constraints. Overall, the Grad-CAM results are used as qualitative supporting evidence for the observed target-concentrated responses, rather than as direct proof of the internal physical mechanism.

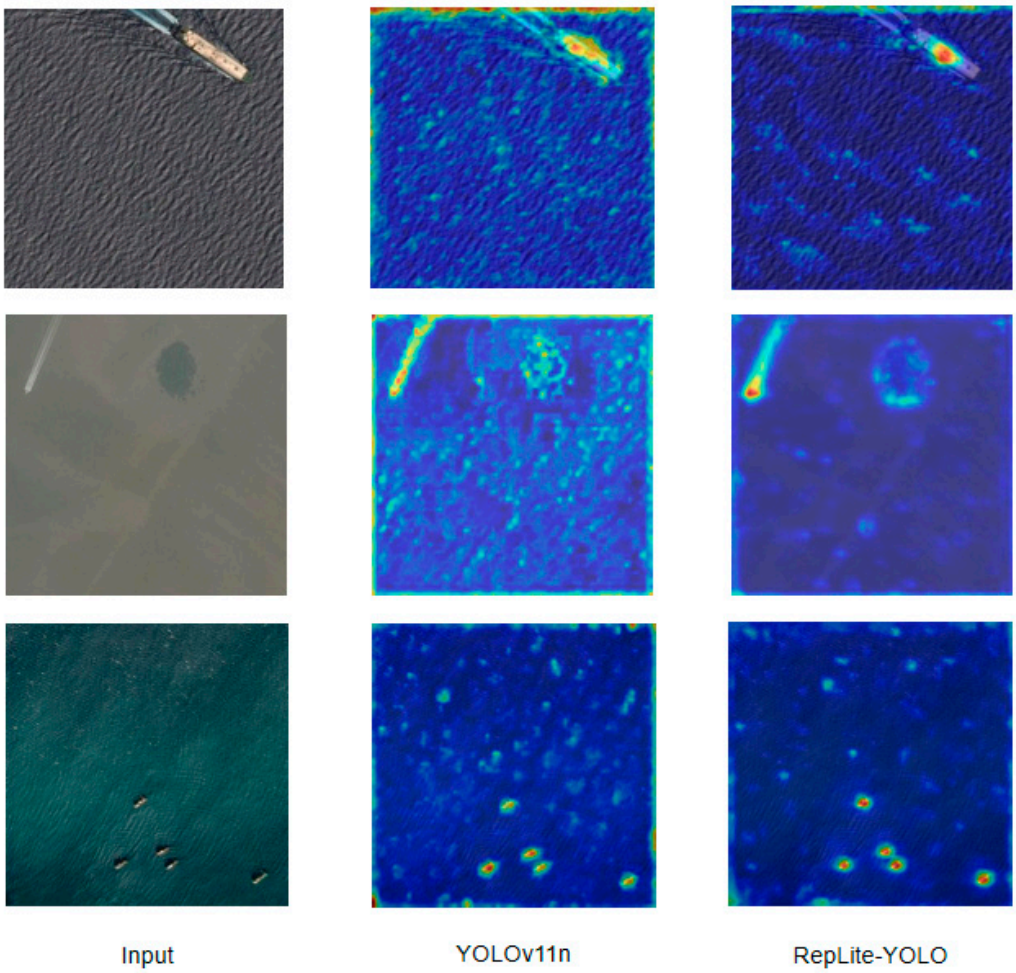


Figure 14. Visual comparison of Grad-CAM feature activation maps between the baseline YOLOv11n and RepLite-YOLO. The first column displays the original input images, while the second and third columns depict the attention heatmaps generated by YOLOv11n and RepLite-YOLO, respectively. For a fair comparison, the Grad-CAM target layers were consistently set to the multi-scale detection-head layers corresponding to the P3, P4, and P5 levels. Specifically, the layer indices are [16,19,22] for YOLOv11n and [18,21,24] for RepLite-YOLO, ensuring exact semantic alignment despite architectural modifications.

4.5. Comparison with Other Models

We used the Vessel and Ship Detection Datasets to compare our framework to other common detection architectures in order to fully test RepLite-YOLO's ability to generalize and work well in complex maritime environments. The assessment includes

anchor-free detectors (YOLOv6, YOLOv10), iterative YOLO variants (YOLOv11, v12 [52], v13 [53]), lightweight baselines (YOLOv3-tiny), and the Transformer-based RT-DETR [54] series, and recent specialized lightweight models for remote sensing (LI-YOLOv8 [55] and SD-YOLO [56]) with quantitative results presented in Table 4.

Table 4. Comparison with Other Object Detection Algorithms.

Dataset	Model	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
Vessel Dataset	RT-DETR-L	0.648	0.677	0.671	0.343	31.99	103.4
	RT-DETR-Resnet50	0.662	0.671	0.681	0.349	41.94	125.6
	YOLOv3-tiny	0.713	0.646	0.672	0.347	9.52	14.3
	YOLOv6n	0.750	0.708	0.746	0.394	4.23	11.7
	YOLOv10n	0.723	0.698	0.728	0.388	2.27	6.5
	YOLOv11n	0.752	0.718	0.749	0.394	2.58	6.3
	YOLOv12n	0.749	0.688	0.738	0.386	2.51	5.8
	YOLOv13n	0.733	0.685	0.730	0.375	2.45	6.1
	LI-YOLOv8	0.765	0.721	0.759	0.403	2.71	7.0
	SD-YOLO	0.737	0.738	0.763	0.406	1.06	10.4
	ours	0.757	0.729	0.761	0.402	1.10	3.2
Ship Detection Dataset	RT-DETR-L	0.804	0.778	0.834	0.420	31.99	103.4
	RT-DETR-Resnet50	0.836	0.801	0.850	0.436	41.94	125.6
	YOLOv3-tiny	0.861	0.725	0.806	0.424	9.52	14.3
	YOLOv6n	0.847	0.788	0.842	0.448	4.23	11.7
	YOLOv10n	0.847	0.772	0.857	0.461	2.27	6.5
	YOLOv11n	0.873	0.764	0.853	0.458	2.58	6.3
	YOLOv12n	0.844	0.786	0.846	0.460	2.51	5.8
	YOLOv13n	0.885	0.762	0.855	0.461	2.45	6.1
	LI-YOLOv8	0.871	0.776	0.851	0.458	2.72	7.0
	SD-YOLO	0.870	0.792	0.872	0.463	1.06	10.4
	ours	0.900	0.783	0.866	0.459	1.10	3.2

Although Table 4 shows that RepLite-YOLO achieves higher mAP@50 than YOLOv11n on both datasets, the absolute improvements are modest. Therefore, to examine whether these improvements are caused by random training fluctuations, we further performed five independent runs for YOLOv11n and RepLite-YOLO using different random seeds. The statistical results are reported in Table 5.

Table 5. Multi-run statistical comparison between YOLOv11n and RepLite-YOLO over five independent random seeds.

Dataset	Model	mAP@50 Mean ± Std	95% CI of mAP@50	Parameters (M)	GFLOPs
Vessel Dataset	YOLOv11n	0.749 ± 0.001	[0.748, 0.751]	2.58	6.3
Vessel Dataset	RepLite-YOLO	0.761 ± 0.001	[0.759, 0.762]	1.1	3.2
Ship Detection Dataset	YOLOv11n	0.853 ± 0.002	[0.850, 0.856]	2.58	6.3
Ship Detection Dataset	RepLite-YOLO	0.866 ± 0.002	[0.864, 0.868]	1.1	3.2

Based on Table 5, RepLite-YOLO achieves an average mAP@50 of 0.761 ± 0.001 on the Vessel Dataset, with a 95% confidence interval of [0.759, 0.762], while YOLOv11n obtains 0.749 ± 0.001, with a 95% confidence interval of [0.748, 0.751]. The paired t-test produced $p < 0.05$, indicating that the mAP@50 difference is statistically significant. On the Ship Detection Dataset, RepLite-YOLO achieves an average mAP@50 of 0.866 ± 0.002,

with a 95% confidence interval of [0.864, 0.868], while YOLOv11n obtains 0.853 ± 0.002, with a 95% confidence interval of [0.850, 0.856]. The paired *t*-test also produced $p < 0.05$, indicating statistical significance. In addition, Table 5 shows that RepLite-YOLO reduces the parameter count from 2.58 M to 1.10 M and GFLOPs from 6.3 to 3.2, demonstrating a favorable accuracy-efficiency trade-off.

In addition to the final performance metrics, we also analyzed the training curves of Precision, Recall, mAP@50, and mAP@50:95 on both datasets, as shown in Figures 15 and 16. RepLite-YOLO shows a stable convergence trend during training and reaches competitive final performance under the same training protocol. Although specialized remote sensing models such as SD-YOLO and LI-YOLOv8 also achieve strong detection accuracy, RepLite-YOLO maintains comparable performance with a substantially smaller computational budget. Therefore, the convergence results further support the efficiency-oriented design goal of RepLite-YOLO, namely preserving competitive detection accuracy while reducing model parameters and GFLOPs.

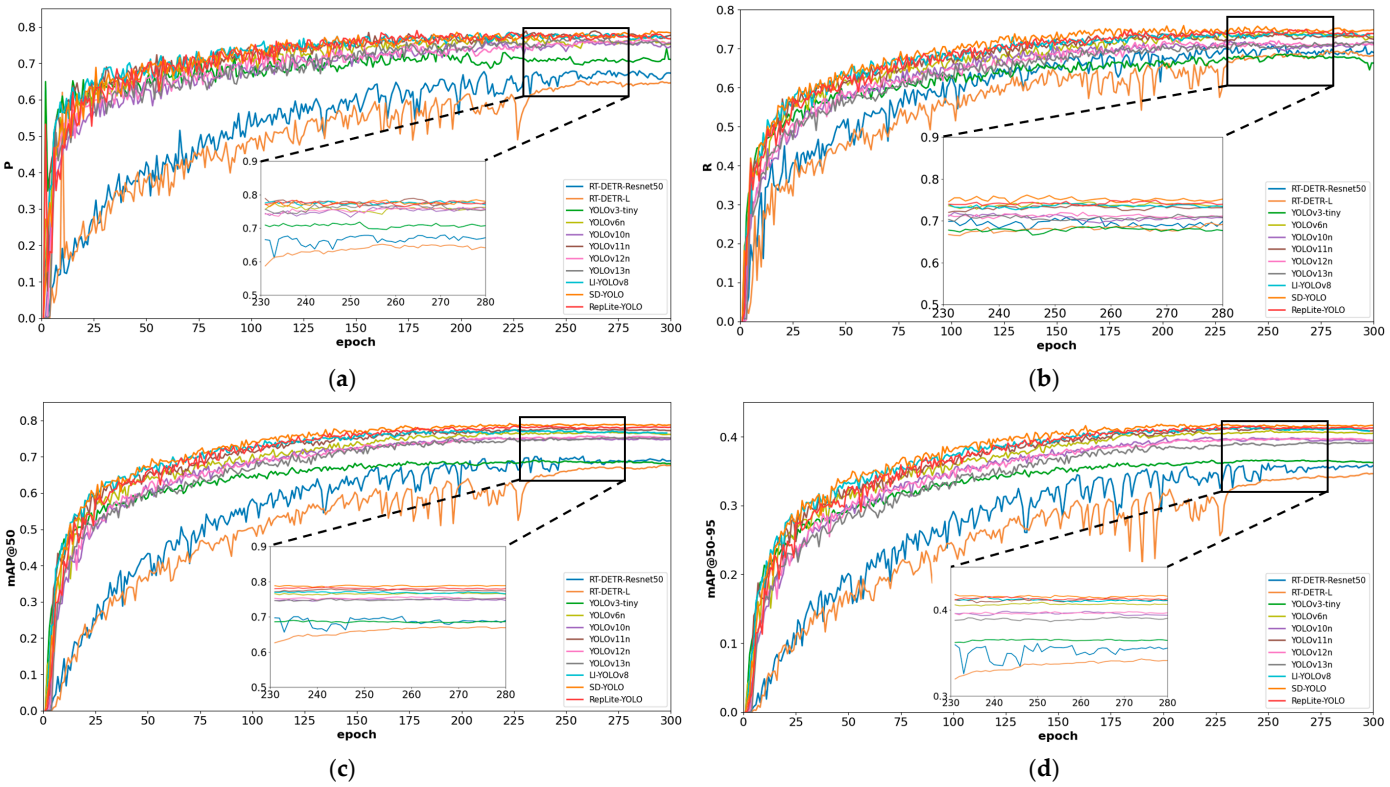


Figure 15. Comparative data plots based on the Vessel Dataset: (**a**) precision, (**b**) recall, (**c**) mAP@50, (**d**) mAP@50-95.

We made a "Computational Cost—Detection Accuracy" trade-off analysis chart (Figure 17) to help us intuitively compare the overall competitiveness of different models in resource-constrained edge scenarios. The horizontal axis of this graph shows floating-point operations (GFLOPs), which shows how much work the computer has to do. The vertical axis shows detection accuracy (mAP@50), and the bubble area shows the number of parameters in the model, which shows how much storage space is needed.

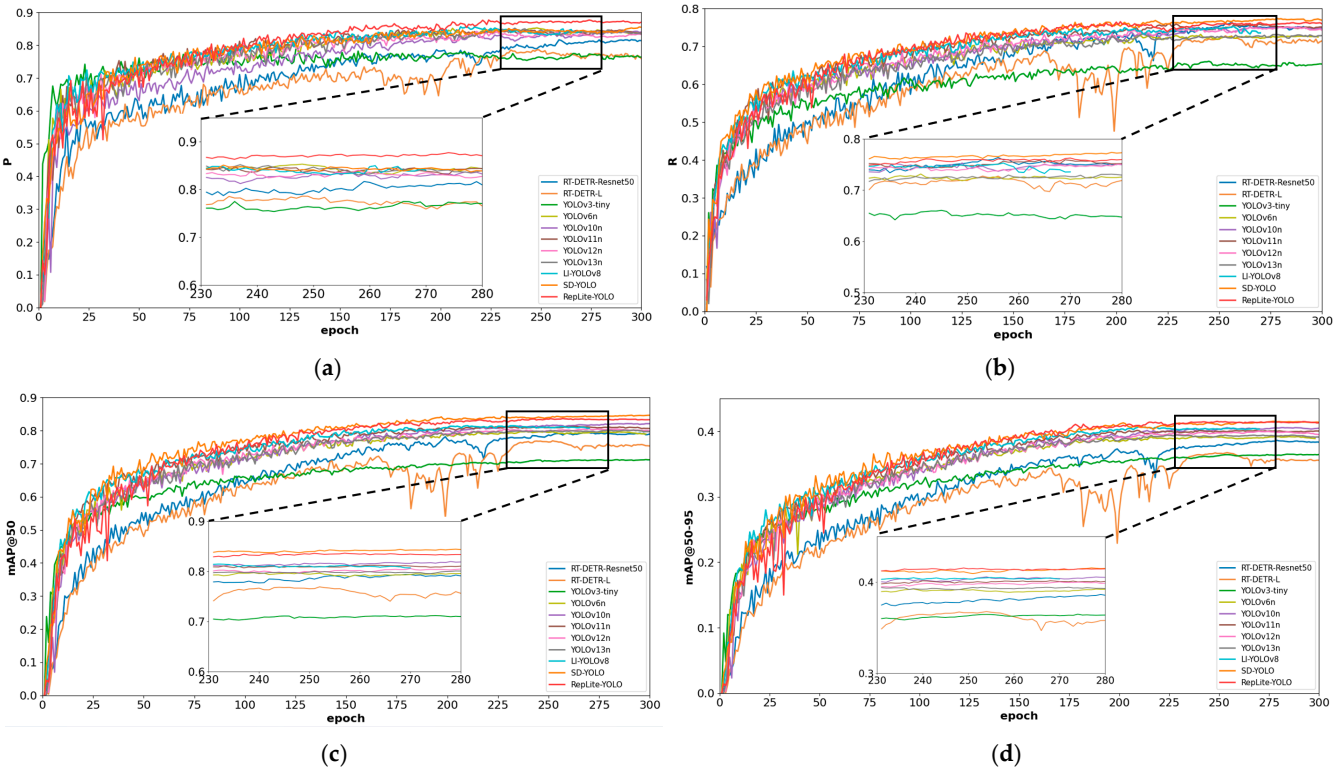


Figure 16. Comparative data plots based on Ship Detection Dataset: (**a**) precision, (**b**) recall, (**c**) mAP@50, (**d**) mAP@50-95.

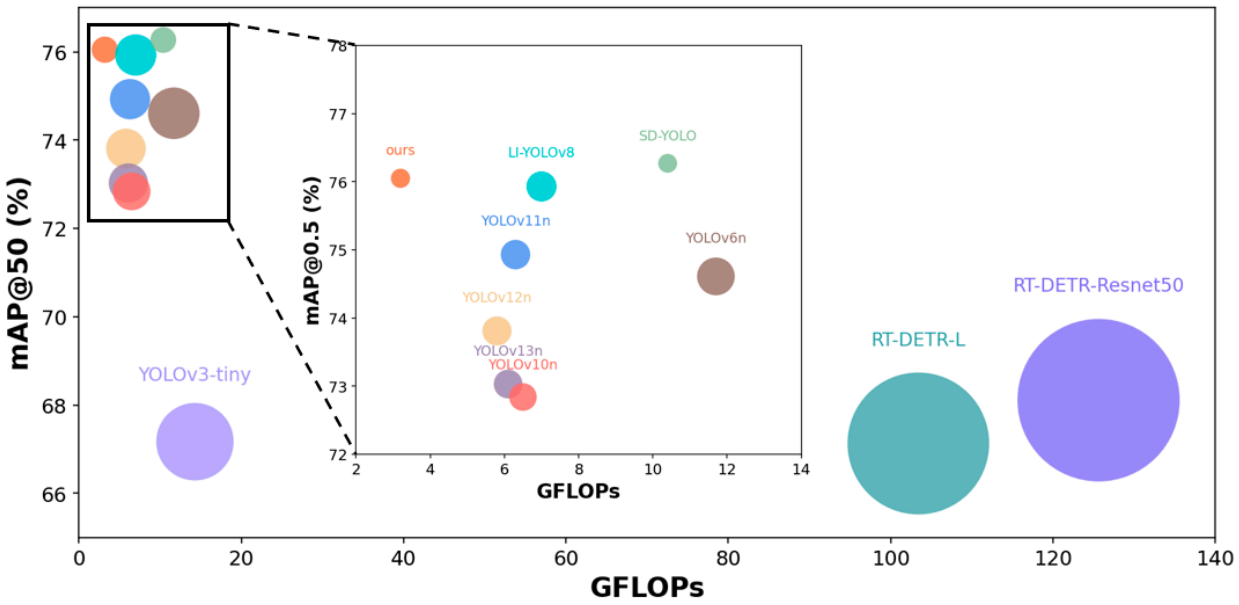


Figure 17. Trade-off analysis comparison of computational cost (GFLOPs) vs. detection accuracy (mAP@50) for different models on the Vessel Dataset.

The results for the Vessel Dataset in Figure 17 show clear performance stratification. Heavyweight models like RT-DETR are on the right side of the chart. They are not very accurate, and their computational costs are over 100 GFLOPs. The orange bubble for RepLite-YOLO is located near the upper-left region of the chart, indicating a favorable balance between detection accuracy and computational cost. In the enlarged view of the GFLOPs < 10 area, our method lowers the computational complexity by almost 50% (from 6.3 to 3.2 GFLOPs) compared with the baseline YOLOv11n, while maintain-

ing competitive mAP@50 with a slight improvement. When compared to the added remote sensing specialized models, although SD-YOLO achieves a marginally higher mAP (0.763 vs. 0.761), it comes at the cost of more than triple the computational overhead (10.4 GFLOPs vs. 3.2 GFLOPs). Similarly, LI-YOLOv8 requires 7.0 GFLOPs but achieves a lower mAP (0.759). The Ship Detection Dataset (Figure 18) shows a similar accuracy-efficiency trend. RepLite-YOLO achieves competitive detection performance with a small model size and low computational cost. When compared to the specialized remote sensing model SD-YOLO on this dataset, although RepLite-YOLO has a marginally higher parameter count (1.10 M vs. 1.06 M) and slightly lower mAP@50 (0.866 vs. 0.872), it achieves this highly competitive accuracy with less than one-third of the computational overhead (3.2 GFLOPs vs. 10.4 GFLOPs). Similarly, compared with LI-YOLOv8, our model achieves comparable or better detection accuracy with lower GFLOPs. These visual results indicate that RepLite-YOLO achieves a favorable accuracy-efficiency trade-off among the compared models. Rather than aiming for absolute accuracy dominance, the proposed model focuses on maintaining competitive detection accuracy while substantially reducing parameters and GFLOPs. This makes it a lightweight candidate for maritime monitoring on resource-constrained edge platforms.

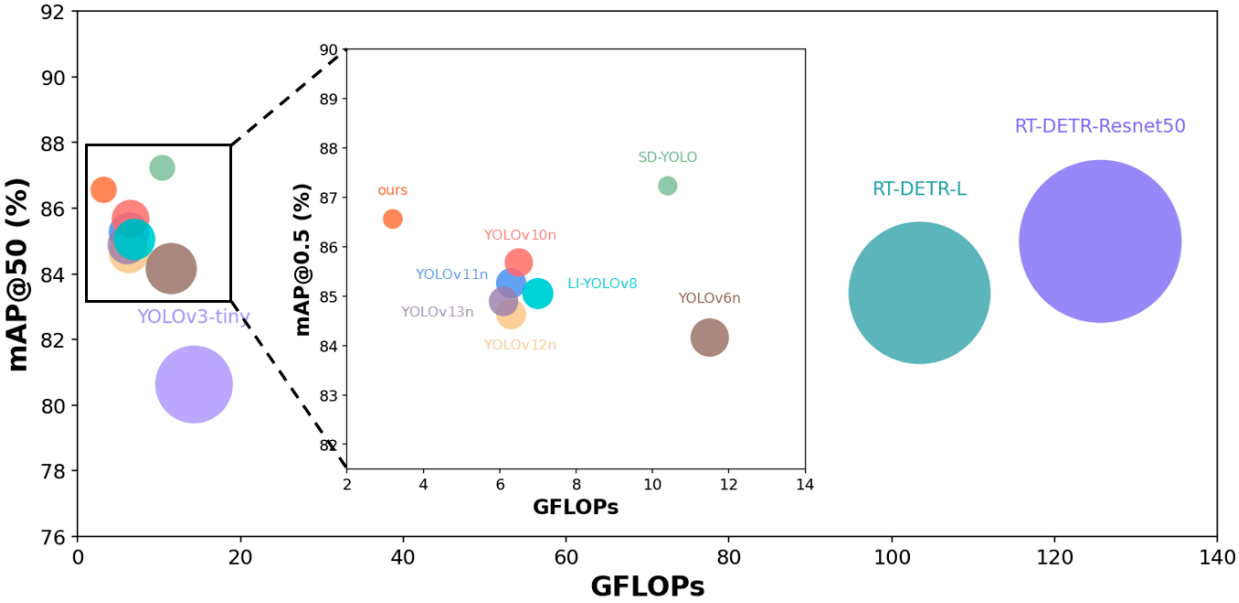


Figure 18. Trade-off analysis comparison of computational cost (GFLOPs) vs. detection accuracy (mAP@50) for different models on the Ship Detection Dataset.

Figure 19 shows a comparison of detection results for different models in three common challenge scenarios: dense small targets in the open sea (left column), complex harbor backgrounds (middle column), and strong wake interference (right column). This is to visually check how well the model works in different sea conditions. The visualization results show that baseline models have different levels of problems: In open sea situations, YOLOv13n and RT-DETR have a lot of missed detections (shown by red frames) because they have trouble telling tiny ships apart from the waves. In nearshore harbor situations, on the other hand, YOLOv11n and YOLOv12n have high false detection rates (shown by yellow frames) because they are affected by the texture of wharf facilities and land structures. In contrast, RepLite-YOLO can accurately lock onto targets in all three of the scenarios above. Missed detections or false alarms happen very rarely, which shows how well its lightweight architecture works when there is a lot of background noise and small target features get worse.

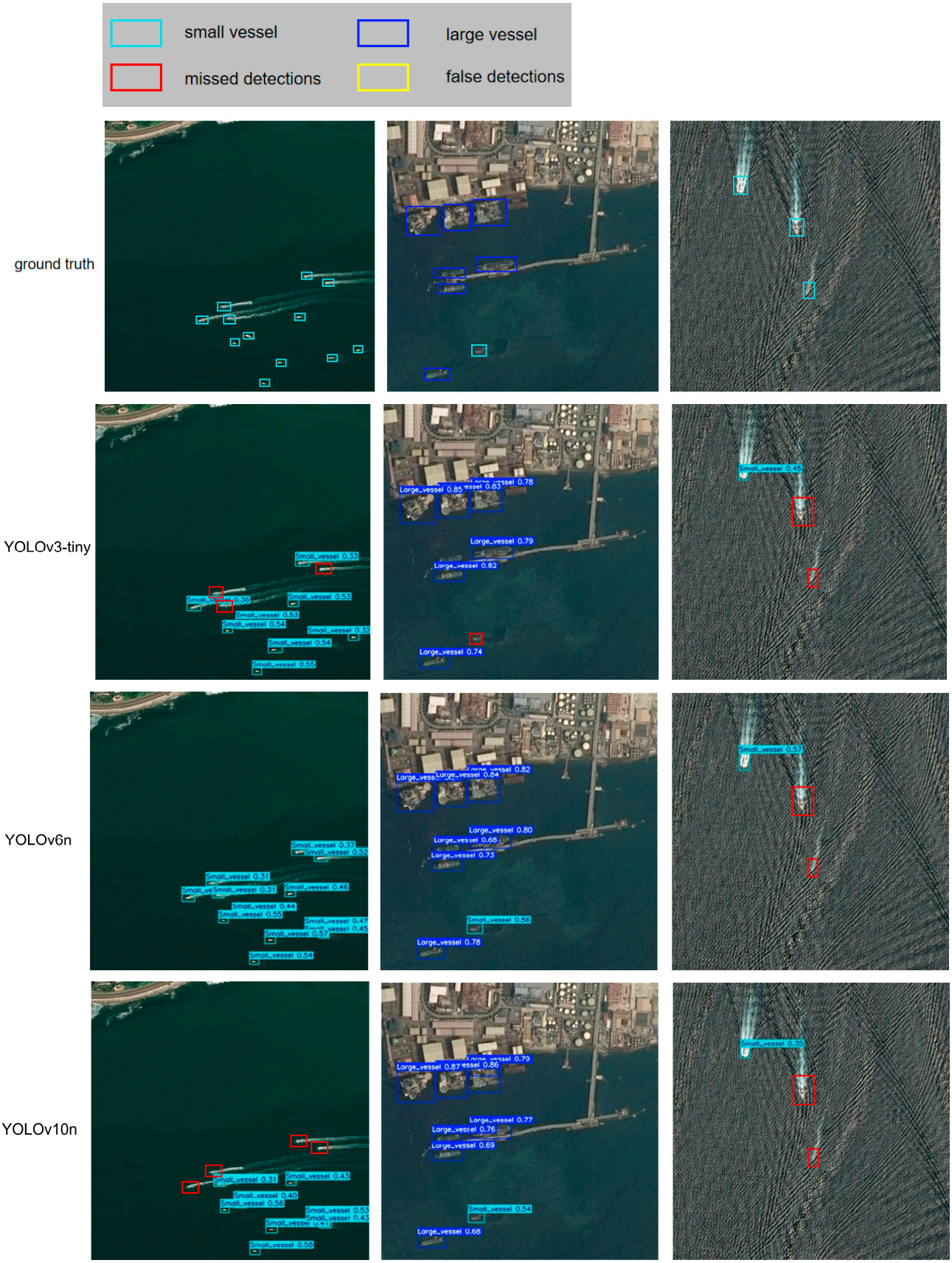


Figure 19. *Cont.*

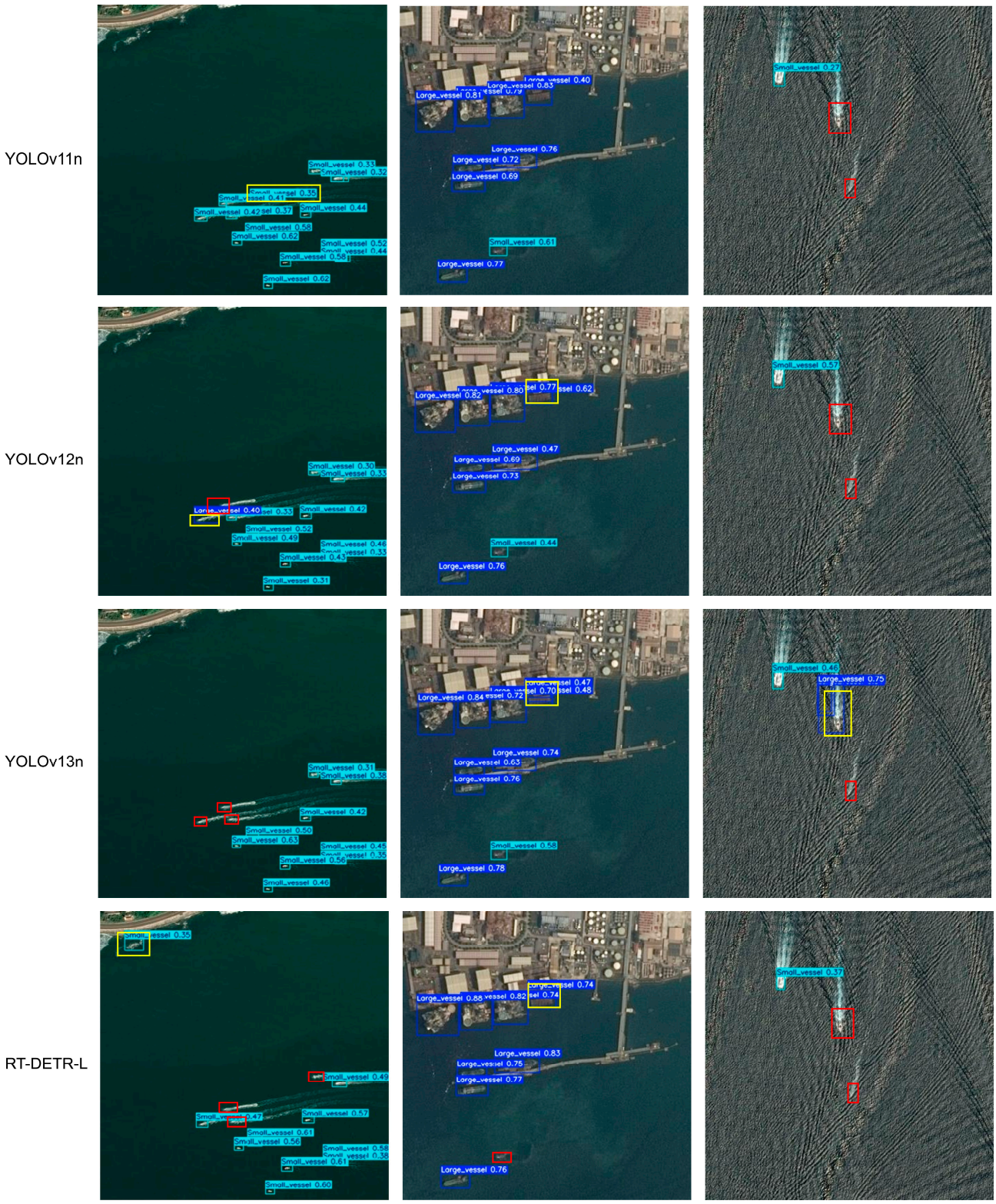


Figure 19. *Cont.*

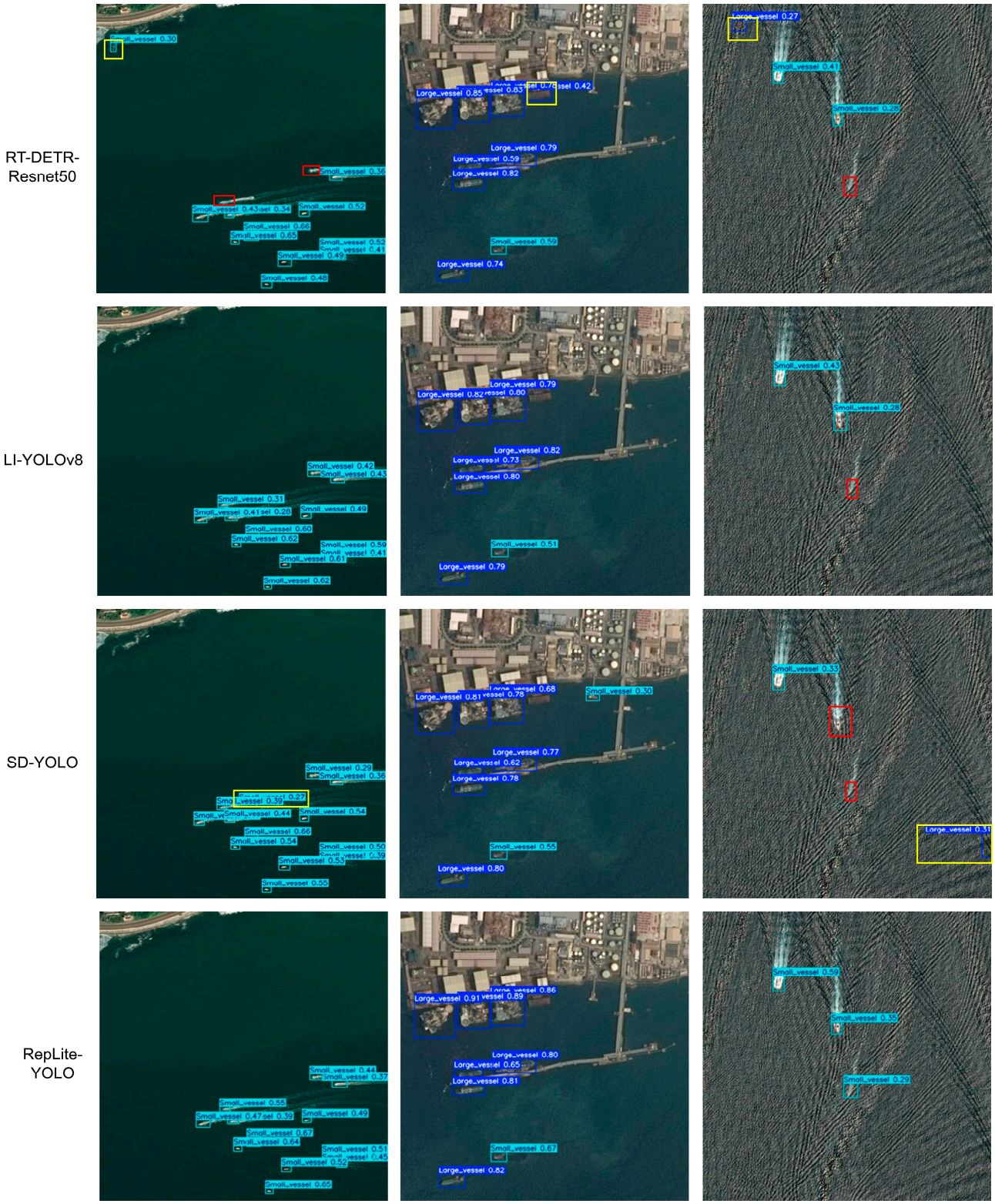


Figure 19. Qualitative comparison of detection results on the test set across diverse maritime conditions. The columns correspond to three distinct scenarios: open water with dense small ships (**Left**), complex harbor backgrounds with mixed-scale vessels (**Middle**), and small ships subject to wake interference (**Right**). Row 1 displays the ground truth, while rows 2–12 show the results of YOLOv3-tiny, YOLOv6n, YOLOv10n, YOLOv11n, YOLOv12n, YOLOv13n, RT-DETR-L, RT-DETR-Resnet50, LI-YOLOv8, SD-YOLO and our proposed RepLite-YOLO, respectively. Red and yellow bounding boxes indicate false negatives (missed detections) and false positives, while blue and cyan labels denote the confidence scores for small and large vessels.

4.6. Comparative Experiment on Improving Modules

4.6.1. Comparison of Different Downsampling Strategies

To find the suitable downsampling architecture for remote sensing platforms with limited resources, we compared the standard baseline to our chosen ADown module and two common lightweight mechanisms: SCDown [57] and V7DownSampling [58]. SCDown uses a spatial and channel decoupling strategy to get rid of unnecessary calculations during feature extraction. V7DownSampling, on the other hand, uses parallel max-pooling and strided convolution branches to protect against the information entropy that comes with downsampling.

Table 6 shows the quantitative results. V7DownSampling has the highest Precision (76.6%), but its Recall (69.5%) is much lower than the baseline, and it requires substantial computation (5.8 GFLOPs). This kind of shortfall makes it more likely that important targets will be missed during maritime surveillance operations. Even though SCDown does well on the mAP@50-95 metric (39.8%), it does not do as well as ADown when it comes to compressing parameters.

Table 6. A comparison of Different Downsampling Strategies.

Model	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
YOLOv11n	0.752	0.718	0.749	0.394	2.58	6.3
ADown	0.763	0.705	0.749	0.398	2.10	5.1
SCDown	0.748	0.712	0.746	0.398	2.16	5.6
V7DownSampling	0.766	0.695	0.746	0.393	2.35	5.8

On the other hand, the architecture that works with ADown finds the favorable balance between efficiency and detection accuracy. This module compresses the model parameter envelope to 2.10 M, which is about 18.7% less than the baseline. At the same time, it reduce the computational cost (GFLOPs) to a field-best 5.1. The results indicate that the dual-branch mechanism of ADown can effectively extract key structural features of ship targets while reducing hardware requirements. This is shown by its high Precision of 76.3% and mAP@50-95 of 39.8%, which are better than the YOLOv11n baseline. This makes it the suitable choice for building lightweight detection networks.

4.6.2. Comparison of Different Feature Extraction Modules

To rigorously isolate the true contribution of the OREPA mechanism from generic channel compression, and to ensure a strictly fair horizontal comparison with other lightweight mechanisms, we designed a highly controlled evaluation matrix. All feature extraction variants (C3k2, GhostConv [59], ScConv [60], and our OREPA_RS) were evaluated under the exact same macroscopic architecture, including the aggressive halving of the P5 channel dimension (1024 $\rightarrow$ 512).

As presented in Table 7, compared to the baseline YOLOv11n, the naive compressed C3k2 variant (with halved channels and no C2PSA) significantly reduces the parameter count to 2.06 M and GFLOPs to 5.9. However, this aggressive compression causes an inevitable semantic loss, dropping the mAP@50 to 0.747 and Recall to 0.712. Remarkably, when the OREPA mechanism is integrated (C3k2_OREPA_RS) under the exact same macroscopic architecture and parameter budget (2.06 M), the network effectively rebounds. The mAP@50 improves to 0.753, and the Recall significantly rises to 0.723, surpassing even the heavier baseline. This strict control experiment rigorously validates that OREPA_RS is not a parameter-reduction tool, but a powerful zero-latency semantic compensator. It successfully recovers the fragile target features lost to channel compression by utilizing

multi-branch over-parameterization during training, delivering pure performance gains during inference without incurring any additional computational cost.

Table 7. A comparison of different feature extraction modules.

Model	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
YOLOv11n	0.752	0.718	0.749	0.394	2.58	6.3
C3k2_OREPA_RS	0.746	0.723	0.753	0.393	2.06	5.9
C3k2	0.752	0.712	0.747	0.392	2.06	5.9
C3k2_Ghost	0.746	0.710	0.748	0.392	2.02	6.0
C3k2_ScConv	0.747	0.720	0.751	0.395	2.06	6.0

Furthermore, under the same compressed conditions, C3k2_OREPA_RS decisively outperforms alternative lightweight modules. It achieves higher accuracy than both GhostConv (0.748) and ScConv (0.751), while maintaining the lowest computational cost (5.9 GFLOPs). Beyond theoretical metrics, alternative mechanisms pose latent bottlenecks for edge deployment. GhostConv relies on generating features through cheap operations followed by concatenations, while ScConv mandates complex spatial and channel splitting-and-reconstruction mechanisms. These operation-intensive and fragmented designs exacerbate MAC and memory fragmentation on edge chips lacking optimized schedulers. In stark contrast, C3k2_OREPA_RS leverages structural re-parameterization to mathematically collapse into a single, plain 3×3 convolution during inference. This guarantees zero additional memory fragmentation, emerging as an optimal solution that robustly aligns with the hardware execution constraints of remote sensing edge deployment.

It should be noted that while DSC-based modules (including those in our ELANFusion_Block and Detect_DWLite) inherently exhibit lower arithmetic intensity than dense convolutions, a characteristic we critically analyze in Section 5, their memory access patterns remain structurally regular and predictable. In contrast, the concatenation-dependent feature generation of GhostConv and the spatial-channel splitting-reconstruction pipeline of ScConv introduce irregular memory access granularity, which amplifies scheduling overhead on lightweight edge processors lacking advanced instruction-level parallelism. This distinction further solidifies C3k2_OREPA_RS, which collapses into a single regular convolution, as the structurally optimal choice under the holistic constraints of remote sensing edge deployment.

4.6.3. Comparison of Different Feature Fusion Modules

For remote sensing image interpretation, it is important to connect the high-resolution spatial details of shallow layers with the abstract representations of deep layers. We redesigned the neck architecture using the ELANFusion_Block to increase multi-scale interaction capabilities while keeping strict lightweight limits. We tested this mechanism very thoroughly against two typical fusion paradigms: SBA [61], which uses attention mechanisms to improve boundaries, and BiFusion [62], which uses complex bidirectional pathways to combine features.

The quantitative comparison results are shown in Table 8. BiFusion had the highest mAP_{50}, but this small improvement came at a huge cost to computing: GFLOPs rose to 13.7 (about 2.2 times the baseline) and parameters rose to 3.62 M. Because of all this extra work, it becomes prohibitive for resource-constrained edge hardware. The SBA module, on the other hand, did not work well. It became less accurate at finding things (the mAP dropped to 72.9%) and took longer to process (it rose to 8.1 GFLOPs).

Table 8. A comparison of Different Feature Fusion Modules.

Model	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
YOLOv11n	0.752	0.718	0.749	0.394	2.58	6.3
ELANFusion_Block	0.763	0.712	0.755	0.398	2.27	6.2
SBA	0.712	0.700	0.729	0.382	2.14	8.1
BiFusion	0.761	0.730	0.758	0.399	3.62	13.7

The ELANFusion_Block, on the other hand, was able to find a good balance between performance and efficiency. This module raised mAP_{50} to 75.5% and Precision to 76.3%. At the same time, it shrank the model parameter envelope to 2.27 M, which is 12.0% less than the baseline, and kept the lowest computational complexity at 6.2 GFLOPs. These metrics confirm that the ELANFusion_Block effectively consolidates multi-scale semantic information through optimized gradient pathways, avoiding unnecessary computations and confirming it as the best architectural choice for lightweight ship detection.

4.6.4. Comparison of Different Detection Heads

The detection head often accounts for a large part of a model's computational budget because it converts feature maps into semantic probabilities and geometric bounding box coordinates. To better fit with the strict deployment requirements of edge hardware, we built the Detect_DWLite module, which is based on DSCs. To empirically validate the superiority of this design, we benchmarked it against two advanced lightweight head architectures: Detect_Efficient, inspired by the compound scaling philosophy of EfficientDet [63], and Detect_LADH [64], which employs an asymmetric decoupled structure.

Table 9 shows that the baseline YOLOv11n model requires 6.3 GFLOPs and contains 2.58 M parameters. Detect_DWLite had the biggest effect on computational cost reduction compared to this standard. It cut the number of parameters to 2.21 M and the cost of computation to 4.7 GFLOPs (a 25.4% drop). Remarkably, this substantial reduction in complexity did not compromise the module's detection ability. In fact, it got the highest mAP@50 (75.2%), which was better than both the baseline and the heavier Detect_Efficient. Detect_LADH had the highest Precision (77.0%), but its Recall dropped significantly (to the lowest at 70.3%), and it had more computational overhead (5.2 GFLOPs) than our proposed method. Detect_DWLite thus finds the favorable balance between speed and accuracy, proving that restructuring decoupled heads with DSCs is a powerful way to improve ship detection efficiency.

Table 9. A comparison of Different Detection Heads.

Model	P	R	mAP@50	mAP@50-95	Parameters (M)	GFLOPs
YOLOv11n	0.752	0.718	0.749	0.394	2.58	6.3
Detect_DWLite	0.751	0.713	0.752	0.390	2.21	4.7
Detect_Efficient	0.759	0.713	0.750	0.395	2.31	5.1
Detect_LADH	0.770	0.703	0.750	0.392	2.28	5.2

5. Discussion

RepLite-YOLO's architectural philosophy deals with the main problem in remote sensing ship detection: how to keep fragile feature details safe while working with limited edge computational resources. The experimental results indicate that, despite the model relinquishing 57.4% of its parameter weight and 49.2% of its computational burden, it maintains competitive detection accuracy with modest improvements. This improvement in performance is not by chance; it is the result of a carefully balanced equilibrium between

keeping fine-grained spatial details in the spatial domain and distilling strong semantic abstractions in the feature domain.

The ADown module uses a parallel processing strategy that works as a Spatially Complementary Feature Preservation Mechanism. To provide a qualitative interpretation of the feature responses, we revisit the Grad-CAM activation maps in Figure 14. The Grad-CAM visualizations suggest that standard generic convolutions in the baseline YOLOv11n tend to show more dispersed activation responses—their attention is blindly scattered across random local noise responses from sea clutter (Figure 14, Row 1). In optical remote sensing imagery, dynamic sea surface clutter contains complex wave-texture patterns. The ADown architecture leverages an initial average-pooling operation acting as a spatial smoothing operation that may reduce these local high-frequency radiation variations. Subsequently, within its parallel design, the max-pooling branch preserves the salient hull-edge cues of tiny vessels, while the convolutional branch aggregates clean spatial context. This complementary design may help reduce the interference between target-related responses and complex ocean backgrounds at shallow layers, which is consistent with the improved recall observed in the experiments.

Building upon this shallow-layer noise suppression, the C3k2_OREPA_RS module may alleviate interference from structured linear patterns in deep layers, such as ship wakes (Figure 14, Row 2). Standard single-path networks easily misclassify the high reflectance of linear sea wakes as the ship hull. Rather than merely 'improving gradient flow', the multi-branch topology of OREPA constructs a multi-scale receptive field feature space during training. This may help distinguish elongated wake-like responses from more compact ship-body responses. Consequently, RepLite-YOLO shows relatively more compact activation responses around the target, reducing attention to linear wake interference. When the system is deployed, these branches collapse into a single-path structure for efficient semantic extraction.

Furthermore, in low-resolution satellite/drone imagery, tiny vessels often occupy merely a few pixels, making their spatial information entropy extremely fragile (Figure 14, Row 3). Standard dense multi-scale feature fusions inevitably cause feature degradation, which may weaken target-related responses. The synergistic integration of the ELANFusion_Block and Detect_DWLite is not merely a generic parameter-reduction technique. Instead, by utilizing DSCs alongside parallel gradient aggregation, it acts as a feature-preservation mechanism. It restricts excessive spatial mapping depth during multi-scale interactions, helping mitigate feature degradation of the fragile hull contours extracted by the ADown module. This lets the limited computational budget focus only on the most important morphological integrity, contributing to a favorable lightweight accuracy-efficiency trade-off under strict edge constraints.

It is important to critically evaluate the practical deployment implications of RepLite-YOLO. While the model halves the theoretical computational burden (−49.2% GFLOPs), the decrease in inference latency on highly parallelized devices (e.g., generic GPUs) may not scale perfectly linearly. This non-linearity is a well-known architectural characteristic of DSC, which heavily populate our tailored ELANFusion_Block and Detect_DWLite head. Because DSCs possess a lower arithmetic intensity compared to dense convolutions, the MAC can become a limiting factor, potentially shifting the workload from a "compute-bound" to a "memory-bound" regime on standard GPUs. However, in the specific context of remote sensing edge deployments, this architectural trade-off is deliberate and theoretically advantageous. On such power-constrained platforms, dynamic power consumption is generally more sensitive to total floating-point operations than to MAC-induced latency fluctuations. Consequently, the 49.2% GFLOPs reduction and 57.4% parameter decrease remain well-aligned with the holistic efficiency requirements of these scenarios. Furthermore,

the extreme compression of the parameter envelope (from 2.58 M to 1.10 M) significantly alleviates local storage limitations on edge devices and has the potential to reduce the communication window required for Over-The-Air (OTA) model updates in bandwidth-constrained environments. Therefore, despite the inherent MAC characteristics of DSCs, our tailored design aligns closely with the holistic constraints of maritime monitoring, providing a viable theoretical foundation for future hardware-specific deployments.

6. Conclusions

We proposed RepLite-YOLO to address the practical trade-off between the extreme spatial degradation of small ships and the strict computational constraints of edge platforms. By restructuring the YOLOv11 framework, we incorporated the ADown module to help preserve ship-contour responses while reducing sea-clutter interference by generating spatially-complementary features. To counteract semantic degradation, we designed the C3k2_OREPA_RS backbone; it helps compensate for semantic degradation after channel compression, retaining deep representational capacity despite aggressive channel compression. Its contribution is evaluated through controlled ablation studies. Furthermore, we constructed the ELANFusion_Block by integrating DSC into the ELAN aggregation paradigm, successfully optimizing multi-scale fusion with minimal GFLOPs, and tailored the Detect_DWLite head for highly compressed task decoupling.

Experimental results on maritime datasets show that RepLite-YOLO reduces the parameter count by 57.4% and GFLOPs by 49.2% relative to the YOLOv11n baseline. Five-run statistical analysis further demonstrates that RepLite-YOLO achieves modest but statistically significant average mAP@50 improvements of 1.2 and 1.3 percentage points on the Vessel and Ship Detection datasets, respectively. These results indicate that RepLite-YOLO provides a favorable accuracy-efficiency trade-off by maintaining reliable detection performance while substantially reducing model complexity.

Building upon our theoretical evaluations of architectural efficiency, future work will expand across three dimensions. First, Scenario Adaptation: evolving from Horizontal Bounding Boxes (HBB) to Oriented Bounding Boxes (OBB) for high-density port berthing situations. Second, Multi-Modal Fusion: investigating feature-level fusion between optical and SAR data to enhance all-weather perceptual abilities. Third, Practical Edge Deployment and Hardware-Aware Optimization: Although RepLite-YOLO substantially reduces theoretical parameters and GFLOPs, its actual inference latency on physical edge devices (e.g., Jetson Nano or ARM CPUs) was not measured in this study due to current hardware unavailability. Therefore, our next critical step is to deploy RepLite-YOLO on representative edge platforms to evaluate practical metrics such as latency, FPS, and energy consumption under different inference backends (e.g., TensorRT, ONNX Runtime). Furthermore, we will leverage Hardware-Aware Neural Architecture Search (NAS) to specialize operators for the specific memory access patterns of target embedded chips, mitigating potential MAC bottlenecks and translating the theoretical floating-point benefits into real-world edge inference speedups. Beyond hardware-oriented deployment, future work may also explore cross-modal SAR–optical maritime surveillance, where sparsity-aware SAR modeling and multilevel SAR target representation may provide complementary cues for optical ship detection [65,66].

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

Funding: This research was funded by the National Natural Science Foundation of China, grant number U25B20229; the Natural Science Foundation of Liaoning Province, grant number 2025-MSLH-098; and the Fundamental Research Funds for the Central Universities, grant number 3132026142.

Data Availability Statement: Publicly available datasets were analyzed in this study. The Vessel Dataset can be found here: https://universe.roboflow.com/igorvirag-8tfp1/vessel_detection_416_416 (accessed on 6 September 2021). The Ship Detection Dataset can be found here: https://universe.roboflow.com/sharath-velishala-pgpib/ship_detection-uiank (accessed on 24 August 2023).

Acknowledgments: During the preparation of this manuscript, the author(s) used Gemini 3.1 pro preview for the purposes of English language translation, grammar checking, and text polishing to improve readability. The authors have thoroughly reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kanjir, U.; Greidanus, H.; Oštir, K. Vessel detection and classification from spaceborne optical images: A literature survey. *Remote Sens. Environ.* **2018**, *207*, 1–26. [CrossRef] [PubMed]
2. 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 (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
3. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
4. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
5. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976. [CrossRef]
6. Hu, M.; Feng, J.; Hua, J.; Lai, B.; Huang, J.; Gong, X.; Hua, X. Online Convolutional Re-Parameterization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 568–577. [CrossRef]
7. Zeng, Y.; Wang, X.; Zou, J.; Wu, H. YOLO-Ssboat: Super-Small Ship Detection Network for Large-Scale Aerial and Remote Sensing Scenes. *Remote Sens.* **2025**, *17*, 1948. [CrossRef]
8. Cao, R.; Sui, J. A Dynamic Multi-Scale Feature Fusion Network for Enhanced SAR Ship Detection. *Sensors* **2025**, *25*, 5194. [CrossRef] [PubMed]
9. He, F.; Wang, C.; Guo, B. SSGY: A Lightweight Neural Network Method for SAR Ship Detection. *Remote Sens.* **2025**, *17*, 2868. [CrossRef]
10. Yuan, Y.; Wei, Y.; Zhou, X.; Guo, Y.; Chen, J.; Jiang, T. YOLO-SBA: A Multi-Scale and Complex Background Aware Framework for Remote Sensing Target Detection. *Remote Sens.* **2025**, *17*, 1989. [CrossRef]
11. Guan, T.; Chang, S.; Wang, C.; Jia, X. SAR Small Ship Detection Based on Enhanced YOLO Network. *Remote Sens.* **2025**, *17*, 839. [CrossRef]
12. Yu, C.; Shin, Y. An efficient YOLO for ship detection in SAR images via channel shuffled reparameterized convolution blocks and dynamic head. *ICT Express* **2024**, *10*, 673–679. [CrossRef]
13. He, R.; Han, D.; Shen, X.; Han, B.; Wu, Z.; Huang, X. AC-YOLO: A lightweight ship detection model for SAR images based on YOLO11. *PLoS ONE* **2025**, *20*, e0327362. [CrossRef] [PubMed]
14. Cui, Z.; Xu, Y.; Jin, X.; Li, Y. An Improved Lightweight Model for Defect Detection on Paths in Images. *Appl. Sci.* **2025**, *15*, 7014. [CrossRef]
15. Li, Y.; Li, M.; Li, S.; Li, Y. Improved yolov5 for remote sensing rotating object detection. In Proceedings of the 2021 6th International Conference on Communication, Image and Signal Processing (CCISP), Chengdu, China, 19–21 November 2021; pp. 64–68. [CrossRef]
16. Zhang, L.; Zhao, S.; An, D.; Wang, P.; Guo, B.; Sun, H. Marine Ship Detection from GF-2 High-Resolution Remote Sensing Images with improved YOLOv13 model. *Front. Environ. Sci.* **2025**, *13*, 1707611. [CrossRef]
17. Li, K.; Wan, G.; Cheng, G.; Meng, L.; Han, J. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS J. Photogramm. Remote Sens.* **2020**, *159*, 296–307. [CrossRef]
18. Guo, Y.; Chen, S.; Zhan, R.; Wang, W.; Yang, J. LMSD-YOLO: A lightweight YOLO algorithm for multi-scale SAR ship detection. *Remote Sens.* **2022**, *14*, 4801. [CrossRef]
19. Khanam, R.; Hussain, M. YOLOv11: An overview of the key architectural enhancements. *arXiv* **2024**, arXiv:2410.17725. [CrossRef]

20. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; Kwon, Y.; Michael, K.; Fang, J.; Zeng, Y.; Wong, C.; Montes, D.; et al. Ultralytics/YOLOv5: V7.0—YOLOv5 SOTA Realtime Instance Segmentation. *Zenodo*. 2023. Available online: https://zenodo.org/records/7347926 (accessed on 10 May 2024).
21. Sunkara, R.; Luo, T. No more strided convolutions or pooling: A new CNN building block for low-resolution images and small objects. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Grenoble, France, 19–23 September 2022*; Springer: Cham, Switzerland, 2022; pp. 443–459. [CrossRef]
22. Stergiou, A.; Poppe, R.; Kalliatakis, G. Refining activation downsampling with SoftPool. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 10337–10346. [CrossRef]
23. Li, Q.; Shen, L.; Guo, S.; Lai, Z. Wavelet integrated CNNs for noise-robust image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 7245–7254. [CrossRef]
24. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. Yolov9: Learning what you want to learn using programmable gradient information. In *Proceedings of the European Conference on Computer Vision (ECCV), Milan, Italy, 29 September–4 October 2024*; Springer: Cham, Switzerland, 2024; pp. 1–21. [CrossRef]
25. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13733–13742. [CrossRef]
26. Ding, X.; Zhang, X.; Han, J.; Ding, G. Diverse branch block: Building a convolution as an inception-like unit. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10886–10895. [CrossRef]
27. Ding, X.; Chen, H.; Zhang, X.; Han, J.; Ding, G. Re-parameterizing your optimizers rather than architectures. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023. [CrossRef]
28. Vasu, P.K.A.; Gabriel, J.; Zhu, J.; Tuzel, O.; Ranjan, A. MobileOne: An Improved One millisecond Mobile Backbone. *arXiv* **2022**, arXiv:2206.04040. [CrossRef]
29. 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 (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768. [CrossRef]
30. Ghiasi, G.; Lin, T.Y.; Le, Q.V. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7036–7045. [CrossRef]
31. Li, H.; Li, J.; Wei, H.; Liu, Z.; Zhan, Z.; Ren, Q. Slim-neck by GSConv: A lightweight-design for real-time detector architectures. *J. Real-Time Image Process.* **2024**, *21*, 62. [CrossRef]
32. Xu, X.; Jiang, Y.; Chen, W.; Peng, Y.; Wang, Z.; Zhao, H. Damo-yolo: A report on real-time object detection design. *arXiv* **2022**, arXiv:2211.15444. [CrossRef]
33. Wang, C.Y.; Liao, H.Y.M.; Yeh, I.H. Designing network design strategies through gradient path analysis. *arXiv* **2022**, arXiv:2211.04800. [CrossRef]
34. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
35. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430. [CrossRef]
36. 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. [CrossRef]
37. Lyu, R. NanoDet: Super Fast and Lightweight Anchor-Free Object Detection Model. Real-Time on Mobile Devices. 2020. Available online: https://github.com/RangiLyu/nanodet (accessed on 8 January 2024).
38. Dai, X.; Chen, Y.; Xiao, B.; Chen, D.; Liu, M.; Yuan, L.; Zhang, L. Dynamic head: Unifying object detection heads with attentions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 7373–7382. [CrossRef]
39. Hong, Y.; Wang, H.; Guo, S. PFW-YOLO: Lightweight helmet detection algorithm. *IEEE Access* **2025**, *13*, 70752–70759. [CrossRef]
40. Al Ahmadi, S.; Mohammad, F.; Al Dawsari, H. Efficient YOLO-based deep learning model for arabic sign language recognition. *J. Disabil. Res.* **2024**, *3*, 20240051. [CrossRef]
41. Huang, L.; Liu, B.; Li, B.; Guo, W.; Yu, W.; Zhang, Z.; Yu, W. OpenSARShip: A Dataset Dedicated to Sentinel-1 Ship Interpretation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 195–208. [CrossRef]
42. Wei, S.; Zeng, X.; Qu, Q.; Wang, M.; Su, H.; Shi, J. HRSID: A High-Resolution SAR Images Dataset for Ship Detection and Instance Segmentation. *IEEE Access* **2020**, *8*, 120234–120254. [CrossRef]
43. Qiao, S.; Zhang, Q.; Wang, Z. A Review of Deep-Learning-Based SAR Image Ship Interpretation Technology: The Latest Advances. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2025**, *18*, 26152–26185. [CrossRef]
44. Yu, J.; Zhou, G.; Zhou, S.; Qin, M. A Fast and Lightweight Detection Network for Multi-Scale SAR Ship Detection under Complex Backgrounds. *Remote Sens.* **2022**, *14*, 31. [CrossRef]

45. Zhang, T.; Zhang, X.; Gao, G. Density knowledge mining for quantity-aware marine vessel surveillance using satellite SAR data. *IEEE Trans. Ind. Inform.* **2026**, *22*, 5173–5184. [CrossRef]
46. Zhang, T.; Gao, G.; Zhang, X. Glance-focus-gaze: A novel eagle-eye vision-inspired panorama-population-individual progressive screening paradigm to capture ships in SAR images. *ISPRS J. Photogramm. Remote Sens.* **2026**, *235*, 241–260. [CrossRef]
47. Tu, H.; Wang, W.; Guo, Y.; Chen, S. Mamba-UDA: Mamba unsupervised domain adaptation for SAR ship detection. *IEEE Geosci. Remote Sens. Lett.* **2025**, *22*, 3595843. [CrossRef]
48. Zhang, T.; Zhang, X.; Gao, G. TLSA: A triple-level speckle awareness framework for speckle-robust marine ship surveillance using satellite synthetic aperture radar. *IEEE Trans. Aerosp. Electron. Syst.* 2026; *early access*. [CrossRef]
49. Deng, J.; Wang, W.; Zhang, H.; Yang, J. PolSAR vehicle recognition via scattering mechanism-driven hybrid attention. *Pattern Recognit.* **2026**, *180*, 114061. [CrossRef]
50. Virag, I. Vessel_Detection_416_416. Computer Vision Project, Version 1. Available online: https://universe.roboflow.com/igorvirag-8tfp1/vessel_detection_416_416 (accessed on 6 September 2021).
51. Velishala, S. Ship_Detection. Computer Vision Project, Roboflow, Colab_Version2. Available online: https://universe.roboflow.com/sharath-velishala-pgpib/ship_detection-uiank (accessed on 24 August 2023).
52. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-centric real-time object detectors. *arXiv* **2025**, arXiv:2502.12524. [CrossRef]
53. Lei, M.; Li, S.; Wu, Y.; Hu, H.; Zhou, Y.; Zheng, X.; Ding, G.; Du, S.; Wu, Z.; Gao, Y. YOLOv13: Real-time object detection with hypergraph-enhanced adaptive visual perception. *arXiv* **2025**, arXiv:2506.17733. [CrossRef]
54. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs beat YOLOs on real-time object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 16965–16974. [CrossRef]
55. Yan, P.; Qi, X.; Jiang, L. LI-YOLOv8: Lightweight small target detection algorithm for remote sensing images that combines GSConv and PConv. *PLoS ONE* **2025**, *20*, e0321026. [CrossRef] [PubMed]
56. Huynh, P.T.; Le, M.T.; Tan, T.D.; Huynh-The, T. SD-YOLO: A lightweight and high-performance deep model for small and dense object detection. *Signal Image Video Process.* **2025**, *19*, 1346. [CrossRef]
57. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 107984–108011.
58. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. 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 (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475.
59. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589. [CrossRef]
60. Li, J.; Wen, Y.; He, L. Scconv: Spatial and channel reconstruction convolution for feature redundancy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 6153–6162. [CrossRef]
61. Tang, F.; Xu, Z.; Huang, Q.; Wang, J.; Hou, X.; Su, J.; Liu, J. DuAT: Dual-aggregation transformer network for medical image segmentation. In *Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Xiamen, China, 13–15 October 2023*; Springer: Singapore, 2023; pp. 343–356. [CrossRef]
62. Zhang, Y.; Liu, H.; Hu, Q. Transfuse: Fusing transformers and cnns for medical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), Strasbourg, France, 27 September–1 October 2021*; Springer: Cham, Switzerland, 2021; pp. 14–24. [CrossRef]
63. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790. [CrossRef]
64. Zhang, J.; Chen, Z.; Yan, G.; Wang, Y.; Hu, B. Faster and lightweight: An improved YOLOv5 object detector for remote sensing images. *Remote Sens.* **2023**, *15*, 4974. [CrossRef]
65. Zhang, T.; Gao, G.; Zhang, X. Triple-level sparsity awareness for marine ship surveillance using satellite synthetic aperture radar. *IEEE Trans. Autom. Sci. Eng.* **2026**, *23*, 5155–5166. [CrossRef]
66. Wang, W.; Zhang, H.; Cheng, J.; Xiang, D.; Tian, Z.; Liu, L. MLCF-Net: Multilevel capsule fusion network combining attribute features for SAR vehicle target recognition. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2026**, *19*, 3629541. [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.