drones

MDPI

Article

A Parameter-Reconfigurable FPGA Acceleration Architecture with Unified Multi-Directional Convolution Mapping for UAV Ship Detection

Yao Zhang [1,2,3], Guanyu Li [1], Chao Dang [1], Chong Li [1,2], Qihua Zhu [1,2], Jun Zhu [1] and Pei Dang [1,*]

[1] Faculty of Geosciences and Engineering, Southwest Jiaotong University, Chengdu 611756, China; zhangyao2022@my.swjtu.edu.cn (Y.Z.); lgy@my.swjtu.edu.cn (G.L.); lichong@my.swjtu.edu.cn (C.L.); zqhgis@my.swjtu.edu.cn (Q.Z.)
[2] Technology Innovation Center for Emergency Surveying and Mapping, Ministry of Natural Resources, Chengdu 611756, China
[3] Sichuan Geomatics Center, Ministry of Natural Resources, Chengdu 610041, China
* Correspondence: dangpei@my.swjtu.edu.cn

Highlights

What are the main findings?

- MPR-ShipNet combines multi-receptive-field directional feature enhancement with progressive multiscale context aggregation, strengthening small-vessel representations while suppressing complex maritime background interference.
- An instruction-driven, parameter-reconfigurable FPGA architecture unifies horizontal, vertical, and standard convolutions on a shared PE array, enabling efficient hardware reuse across different detection layers.

What are the implications of the main findings?

- The shared PE array reduces redundant hardware units, off-chip memory access, and data movement overhead, improving FPGA resource utilization and configuration flexibility.
- The INT8 FPGA implementation achieves 91.5% mAP@0.5, 74 ms inference latency, and 2.59 W power consumption, with a 3.0-percentage-point accuracy reduction from the FP32 model.

Abstract

Motivated by the need for low-latency, energy-efficient, and reconfigurable computation in UAV-based maritime monitoring, this paper proposes a parameter-reconfigurable FPGA acceleration architecture that provides a unified mapping for multi-directional convolutions, using the lightweight MPR-ShipNet as the validation network. Through a direction-selection and zero-masking mechanism, the architecture maps horizontal 1×3, vertical 3×1, and conventional 3×3 convolutions onto a shared 3×3 processing-element (PE) array. Software instructions dynamically configure convolution, pooling, activation, upsampling, and data-transfer operations, thereby eliminating the need to construct dedicated hardware units for different convolutional operators. The design further incorporates an eight-channel INT8 quantization module, a multichannel pipeline, and on-chip data-reuse mechanisms, which collectively reduce intermediate-feature storage, off-chip memory accesses, and data-transfer overhead. Experimental results show that MPR-ShipNet achieves a precision of 93.7%, a recall of 95.6%, an mAP@0.5 of 94.5%, and an F1-score of 94.6% on the SeaShips dataset. It also maintains robust detection performance under disturbances arising from noise, illumination variation, blur, and occlusion. When implemented on an

Check for updates

Academic Editor: Yahui Zhang

Received: 30 August 2026
Revised: 15 October 2026
Accepted: 16 October 2026
Published: 22 October 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.

XC7Z100 FPGA operating at 200 MHz, the proposed accelerator achieves a throughput of 166.35 GOPS, a per-frame inference latency of 74 ms, and a power consumption of 2.59 W, corresponding to an energy efficiency of 64.23 GOPS/W. The INT8-deployed model achieves an mAP@0.5 of 91.5%, representing an absolute decrease of 3.0 percentage points, or a relative decrease of approximately 3.17%, compared with the 94.5% FP32 baseline. These results demonstrate a practical trade-off between detection accuracy and deployment efficiency for resource-constrained edge platforms.

Keywords: UAV maritime monitoring; ship detection; parameter-reconfigurable FPGA accelerator; unified multi-directional convolution mapping; hardware-software co-design; INT8 quantization; edge computing; low-power inference

1. Introduction

With the rapid advancement of optical imaging technologies, the temporal and spatial resolution of unmanned aerial vehicle (UAV) imagery has increased substantially [1], causing image data volumes to grow exponentially. However, bandwidth constraints [2] inevitably introduce considerable transmission delays when such large volumes of data are communicated. Moreover, a substantial share of these data may be uninformative, generating considerable redundancy [3]. Continuous and accurate monitoring of small vessels using optical imagery has been applied in fisheries management and military security [4]. Nevertheless, limitations in image resolution make the accurate and rapid detection of small fishing vessels and cargo ships in optical imagery a particularly challenging task [5].

Conventional vessel-detection methods typically rely on handcrafted feature extractors, such as edge detection [6], shape analysis [7], and texture analysis [8]. These approaches require manual specification of thresholds, filtering scales, and candidate-region selection rules for each application setting. Their performance depends heavily on prior knowledge and is generally insufficiently robust to complex backgrounds and noise [9]. In maritime environments, illumination variation [10], sea-surface reflections, wave textures, haze, cloud occlusion, and shoreline and port infrastructure can generate nuisance patterns resembling vessel contours or textures. Such interference reduces target-background contrast and increases both missed detections and false positives. These challenges are particularly acute in UAV-based maritime inspection. Flight altitude, viewing angle, and platform attitude vary continuously, while UAV vibration, rapid motion, and image-transmission compression can introduce blur and loss of detail. Consequently, the same vessel may exhibit substantial variation in scale, orientation, and image quality across observations. Small vessels at long range often occupy only a few pixels, and their weak edge and texture cues can be readily obscured by waves and specular reflections. Moreover, conventional methods typically require multiple processing stages for high-resolution imagery, including enhancement, denoising, tiling, candidate-region generation, and post-processing [11].

In recent years, deep convolutional neural networks have made substantial advances in object detection for natural-scene imagery [12]. Relative to conventional approaches that rely on handcrafted features, deep-learning methods can learn hierarchical feature representations end to end, ranging from low-level texture and edge cues to high-level semantic information. They therefore exhibit stronger adaptability in settings characterized by complex backgrounds, scale variation, and object occlusion. Contemporary object-detection algorithms are generally classified as either two-stage or single-stage detectors [13]. Two-stage detectors are exemplified by the region-based convolutional neural network (R-CNN) family [14]. These models typically first generate candidate regions that may contain

objects and then use detection heads to perform classification, bounding-box regression, and refinement of predicted boxes. By exploiting candidate-region features more thoroughly, two-stage methods often achieve high detection accuracy in dense-target settings and complex backgrounds. However, proposal generation and region-wise processing introduce substantial computational costs, memory-access overhead, and inference latency [15], making such methods difficult to reconcile with the real-time requirements of UAV-based vessel surveillance. Single-stage detectors, such as YOLO and SSD [16], integrate object localization and category prediction within a single network and directly produce dense predictions over multiscale feature maps. By avoiding proposal generation and redundant feature computation, they offer simpler architectures, faster inference, and greater suitability for edge deployment [17]. Although early single-stage detectors generally exhibited lower accuracy than two-stage approaches for small objects and complex backgrounds, advances in multiscale feature fusion, attention mechanisms, and lightweight network design have substantially narrowed this gap. These methods are therefore well suited as base models for real-time vessel detection in UAV imagery and FPGA-accelerated deployment.

Although CNN-based vessel-detection methods have achieved strong performance on public remote-sensing datasets [18], their deployment in UAV maritime monitoring remains challenging. Distant vessels occupy few pixels and are easily affected by waves, solar glare, shoreline infrastructure, occlusion, and background clutter; small targets are particularly susceptible to feature loss and background interference [19]. Meanwhile, CNN inference requires extensive convolution, feature-map buffering, and memory access [20], which are difficult to accommodate under the payload, power, thermal-dissipation, and flight-endurance constraints of UAV platforms [21].

Conventional UAV maritime-surveillance systems commonly transmit raw images or video to a ground station for processing [22]. However, limited communication bandwidth and link stability can increase transmission overhead and response latency [23]. Deploying detection networks at the edge can reduce unnecessary data transmission, although multiscale feature fusion and complex detection heads continue to impose substantial computational and memory demands [24], particularly on UAV platforms with restricted payload, power, and thermal-dissipation capacity [25].

General-purpose CPUs and GPUs support flexible CNN deployment [26], but CPUs may not satisfy real-time inference requirements, whereas GPUs generally require higher power consumption and more demanding thermal management. Specialized accelerators mainly include ASICs and FPGAs [27]. Although ASICs can provide high computational efficiency, their long development cycles, high design costs, and limited post-fabrication flexibility restrict their adaptability to evolving network architectures [28]. FPGAs provide configurable logic, parallel computation, and on-chip memory, making them more suitable for resource-constrained edge deployment. However, existing FPGA implementations are often designed for fixed network structures or limited operator types, and accelerator IP-based solutions still require improvements in processing-element utilization, data reuse, and adaptability to multiscale detection networks [29]. Moreover, previous studies generally optimize the detection algorithm and hardware accelerator separately. To address this limitation, the present study jointly designs MPR-ShipNet and a parameter-reconfigurable FPGA accelerator to balance detection accuracy, model complexity, inference latency, power consumption, and hardware resource utilization.

To address small vessel targets, complex maritime backgrounds, and the resource constraints associated with UAV-oriented edge deployment, this paper develops a hardware-software co-designed vessel-detection system that integrates the lightweight MPR-ShipNet with a parameter-reconfigurable FPGA accelerator. The main contributions are summarized as follows:

1. We develop MPR-ShipNet, a lightweight single-stage vessel-detection network incorporating a multi-receptive-field fine-grained enhancement module (MRFEM) and a progressive multiscale context aggregation module (PMCAM). MRFEM employs parallel multiscale and direction-aware branches to strengthen the edge, texture, and elongated structural features of small vessels, while PMCAM integrates shallow localization cues with deep semantic information to suppress complex maritime background interference.

2. We design an instruction-driven, parameter-reconfigurable FPGA dataflow architecture that abstracts convolution, pooling, activation, zero padding, and data transfer as configurable operators. These operators can be dynamically reconfigured according to layer-level requirements, improving hardware adaptability to different network operations and multiscale detection tasks.

3. We introduce a unified mapping scheme for multi-directional convolutions and a low-bit pipelining mechanism. Horizontal 1×3, vertical 3×1, and standard 3×3 convolutions are mapped onto a shared 3×3 processing-element array through direction selection and zero masking. Combined with eight-channel INT8 quantization, on-chip data reuse, and multichannel pipelining, the proposed accelerator achieves a throughput of 127.45 GOPS, an inference latency of 74 ms per frame, and a power consumption of 2.59 W on the XC7Z100 FPGA.

2. Materials and Methods

2.1. Hardware Circuit System Architecture

The defining characteristic of the UAV-based vessel-image detection accelerator developed in this study is its reconfigurability. Specifically, its computational functions can be selected through software parameters, including convolution, pooling, and activation operations. Its data-movement patterns are likewise software-controlled, allowing the configuration of input zero padding, data-transfer addresses, and transfer volumes. Figure 1 illustrates the overall dataflow architecture of the accelerator.

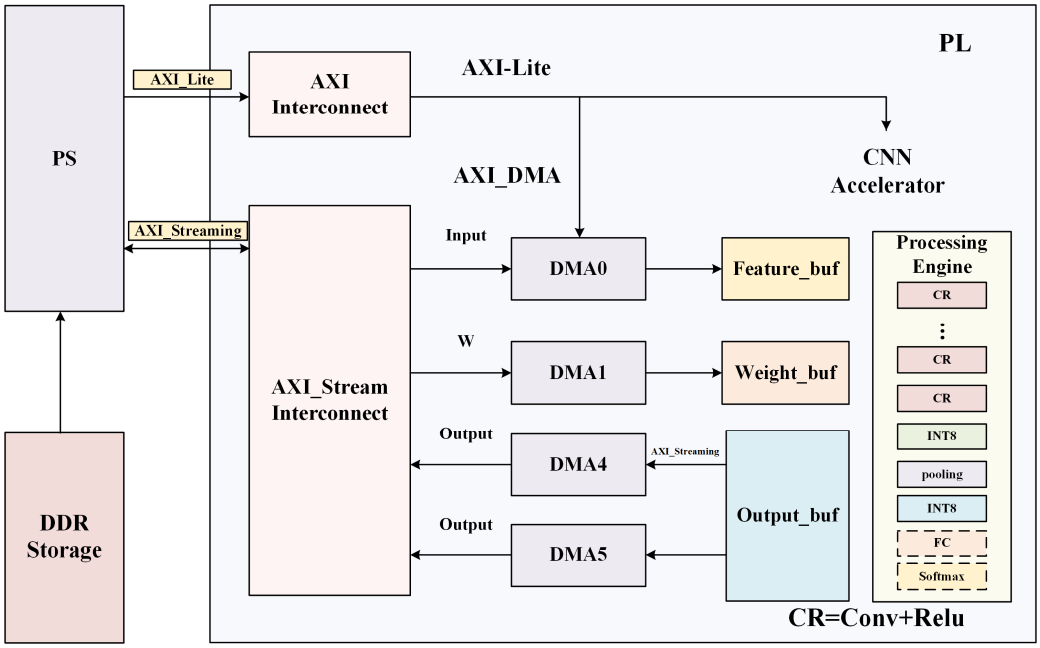


Figure 1. Overall architecture of the proposed accelerator.

Unlike conventional accelerator designs that implement dedicated hardware for all required network operations on chip, the proposed reconfigurable approach avoids duplicating functional modules and reduces redundant on-chip resource consumption. The

vessel-detection edge device is implemented on an XC7Z100-2FFG900 FPGA (Xilinx, Inc., San Jose, CA, USA) development board, which comprises a processing system (PS) and programmable logic (PL). The PL implements configurable modules for convolution, activation, pooling, feature enhancement, and quantization. The PS contains two 1.06 GHz ARM Cortex-A9 processors and provides an instruction set for invoking and coordinating the PL modules, while controlling the number of data-reuse cycles required by the overall algorithm. Communication between the PS and PL is supported by two buses: AXI4-Lite for register-command transmission and AXI4-Stream for data transfer. Direct memory access (DMA) manages data exchange between on-chip and off-chip memory. To improve the extensibility of the accelerator, the processing engine reserves optional FC and Softmax modules for potential support of future classification networks or other CNN architectures. These modules are not required by MPR-ShipNet and are disabled by the instruction controller during the current vessel-detection inference process. The output feature maps generated by the convolutional detection head bypass the FC and Softmax modules and are transferred directly to the output buffer for subsequent decoding and non-maximum suppression. Therefore, the reserved FC and Softmax modules do not participate in MRFEM computation or affect the detection results reported in this study.

2.1.1. Design of a Configurable Eight-Channel INT8 Quantization Module

To reduce feature-map storage requirements, on-chip and off-chip data-transfer bandwidth, and the computational overhead of subsequent operators when deploying the UAV vessel-detection network on an FPGA, this paper introduces a configurable INT8 quantization module after convolution and feature-fusion operations. The module takes the 32-bit fixed-point accumulation results produced by the convolutional PE array as input and rescales these high-precision intermediate values to signed 8-bit integers through integer multiplication, shift-based rounding, zero-point correction, and saturation clipping. The quantization process employs integer-approximate scaling and can be expressed as follows (1):

$$q = Sat8\left(Round\left(\frac{x \times M0}{2^n}\right) + Zero\right) \quad (1)$$

where x denotes the 32-bit accumulated result, M0 is the integer scaling multiplier, n is the arithmetic right-shift amount, Zer is the quantization zero point, and Sat8 denotes saturation clipping to the representable INT8 range.

As illustrated in Figure 2, the single-channel quantization submodule, quant_int8, comprises three cascaded computational stages. The first stage employs an extended-precision multiplier to multiply the accumulated input value by the scaling multiplier, while retaining guard bits to reduce intermediate truncation error. The second stage performs a configurable arithmetic right shift according to the shift parameter and uses rounding-control bits to compensate for precision loss caused by truncating low-order bits. In the third stage, the shifted result is added to the zero-point offset and then passed through saturation logic to produce 8-bit quantized data. Pipeline registers isolate all three stages, forming a stable positive-edge-triggered pipeline that sustains continuous data-stream processing while ensuring timing closure.

At the system level, the quant_int8_8ch module instantiates eight quant_int8 submodules in parallel, each corresponding to one output channel of the convolutional PE array. Each channel maintains an independent scaling multiplier, shift parameter, and zero-point offset to accommodate the numerical ranges of different network layers and feature channels. Given a continuous stream of valid inputs, all eight channels enter the pipeline synchronously and produce quantized results in parallel. During layer transitions, the processing system (PS) writes configuration parameters to programmable-logic (PL) reg-

isters through the AXI4-Lite interface, while quantized data are transferred through the AXI4-Stream data path. This parameterized design enables a single quantization array to support feature processing in different layers, including MRFEM, PMCAM, and the detection head, without requiring separate dedicated low-bit conversion modules.

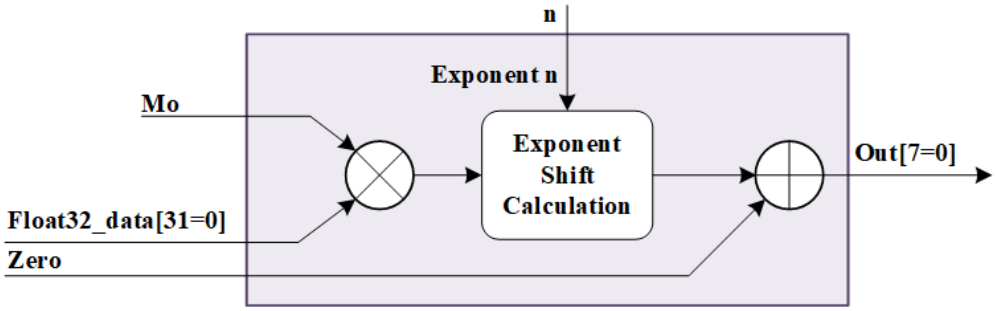


Figure 2. Single-Channel INT8 Quantization Datapath.

Accordingly, the proposed quantization module not only converts high-precision accumulated results into INT8 features, but also serves as a unified numerical interface between the algorithm and the hardware. It maps the layer-specific quantization requirements of different operators into programmable parameters. Combined with multichannel parallelism, pipelined registers, and interlayer parameter reconfiguration, the module controls quantization error while reducing FPGA memory-access pressure and implementation cost.

2.1.2. Hierarchical Data Reuse for Multichannel Convolution

Convolution is the central computational operation in convolutional neural networks. Its hardware implementation typically comprises an input-feature buffer, a weight-parameter buffer, a data-reformatting unit, and a processing-element (PE) array, as illustrated in Figure 3. Following image preprocessing, input feature maps are stored in off-chip memory in row-major order. At the start of convolution, the feature-data tile and weight parameters required for the current computation are loaded into the input-feature buffer and weight buffer, respectively. For convolutional layers that preserve the spatial dimensions of the output feature map, the padding module inserts zeros at feature-map boundaries according to configuration parameters, using state control and row-column counters. The data-reformatting unit subsequently employs shift registers and a three-row line-buffer structure to reorganize the sequential input data into sliding windows matched to the convolution-kernel size. For a 3 × 3 kernel, each window contains nine feature values, which are supplied to the PE array together with the corresponding nine weight values. The 3 × 3 line buffer and PE array represent the basic data-processing tile rather than the maximum supported kernel size. The 5 × 5 convolution follows the same multiplyaccumulate principle as the 3 × 3 convolution and is implemented by adjusting the input-feature dataflow. Specifically, the controller changes the feature-reading addresses and sliding-window offsets so that different portions of the 5 × 5 receptive field are sequentially mapped onto the shared 3 × 3 PE array. The three-row line buffer is reused in multiple passes, and the partial sums generated by the PE array are accumulated before the final output is written back. Therefore, the 3 × 3 and 5 × 5 convolutions share the same line buffer, PE array, and accumulation datapath; only the dataflow schedule, number of processing passes, and accumulation control differ.

For a convolutional layer with Cinput channels and a K × K convolution kernel, the output of the o th channel at spatial location (h, w) can be expressed as (2):

$$Y_{o,h,w} = b_o + \sum_{c=0}^{C_{in}-1} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \widetilde{X}_{c,hS+i,wS+j} \cdot W_{o,c,i,j} \quad (2)$$

where $\widetilde{X}$ denotes the padded input feature map, W represents the convolution-kernel weights, b is the bias term, and S denotes the convolution stride.

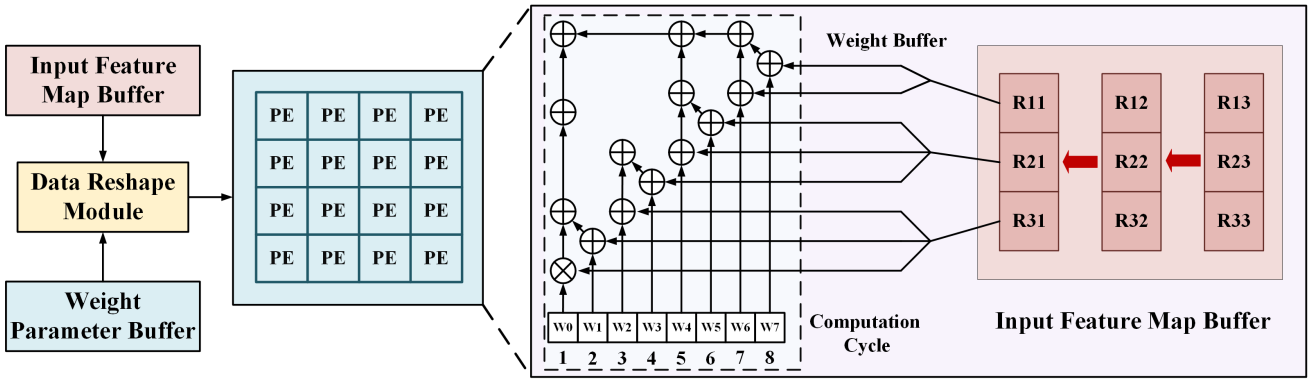


Figure 3. Convolution calculation module.

The processing element (PE) is the core computational unit for convolution and primarily performs local multiplyaccumulate operations within an individual input channel. As illustrated in Figure 4, each PE comprises a parallel multiplier array, a multistage adder tree, a bias-addition unit, pipeline registers, and control logic. The input feature vector and corresponding convolution-kernel weights first enter the multiplier array, where element-wise multiplications are performed in parallel. The resulting partial products are then accumulated through the multistage adder tree to obtain the partial sum for a single convolution window. When the bias-enable signal is asserted, the bias term is added to the accumulated result to produce the final convolution output.

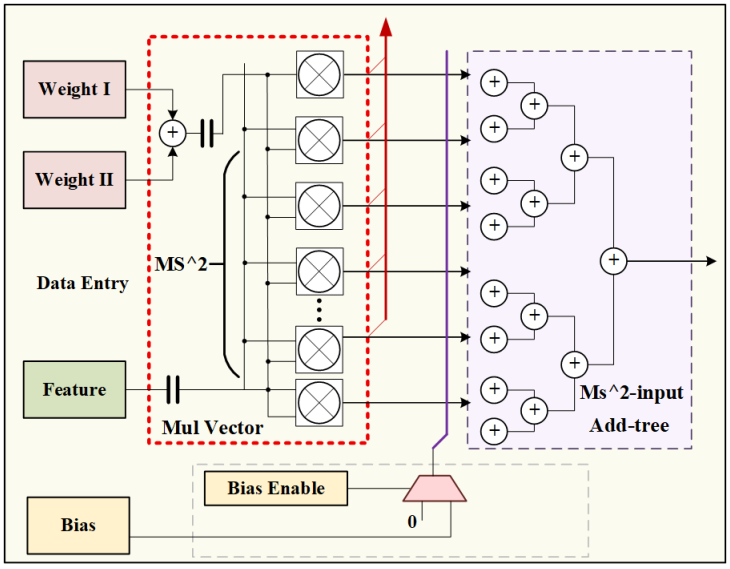


Figure 4. Internal Architecture of the Processing Element.

For the C th input channel, the local convolution result computed by the PE can be expressed as:

$$P_{o,c,h,w} = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \widetilde{X}_{c,hS+i,wS+j} \cdot W_{o,c,i,j} \quad (3)$$

The local results from different input channels are subsequently accumulated and combined with the bias term to obtain the output defined in Equation (2). Pipeline registers placed at the multiplication, accumulation, and output stages allow computations for

different convolution windows to overlap across multiple clock cycles, thereby increasing PE data throughput and convolution efficiency.

2.1.3. Unified Mapping Mechanism Based on Direction Selection and Zero Masking

To support a unified hardware implementation of the horizontal 1×3, vertical 3×1, and standard 3×3 convolutions used in MPR-ShipNet, this paper introduces a direction-selection and zero-masking mechanism into the multichannel convolution data path. The mechanism is positioned between sliding-window generation and the PE array. For each input channel, the programmable logic (PL) first generates a fixed-size 3×3 feature window. A mode-configuration register then selects the horizontal mode H, vertical mode V, or standard mode S according to the requirements of the current network layer.

As shown in Figure 5, mode H retains only the three valid feature values in the center row of the 3×3 window and sets the remaining six positions to zero. Mode V retains only the three valid values in the center column, whereas mode S preserves the complete 3×3 window. The weight buffer follows the same mapping rule: the weights of a horizontal 1×3 kernel are loaded into the center row of a 3×3 weight matrix, while those of a vertical 3×1 kernel are loaded into the center column; all remaining weight positions are set to zero. After mapping, all three convolution types are supplied through a fixed nine-input interface to the same 3×3 PE array. Because products involving inactive positions are zero, the number of multipliers, the adder-tree architecture, and the cross-channel accumulation procedure remain unchanged.

$$Y_{o,h,w}{}^{(m)} = b_o + \sum_{c=0}^{C_{in}-1} \sum_{i=0}^{2} \sum_{j=0}^{2} M_{i,j}{}^{(m)} X_{c,hS+i,wS+j} W_{o,c,i,j}{}^{(m)} \quad (4)$$

where m denotes the convolution mode, taking the value H, V, or S, and M denotes the binary directional mask. Under mode H, only the mask values in the center row equal 1; under mode V, only those in the center column equal 1; and under mode S, all mask values equal 1. Mathematically, the mask specifies the valid multiplyaccumulate positions; in hardware, it is implemented through input selection and zero-value injection. The PS writes the mode parameter to the configuration register during network-layer transitions, while feature windows and weight data continue to stream into the PE array. Consequently, the design does not increase the size of the convolutional compute array and preserves the fixed data interface and initiation interval of the multichannel pipeline.

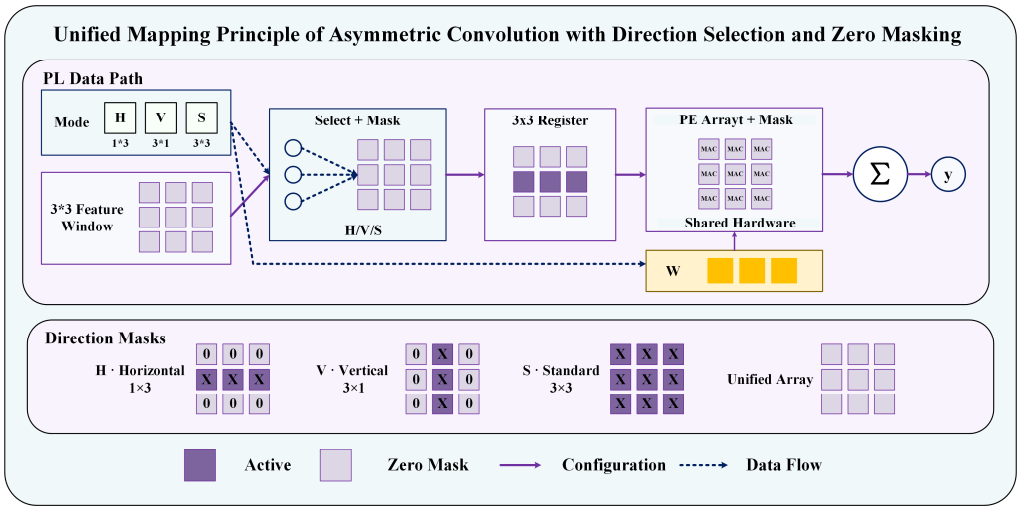


Figure 5. Direction Selection and Zero-Masking Unified Mapping Mechanism for Multichannel Convolution.

After directional mapping, the local multiplyaccumulate results from each input channel are accumulated along the channel dimension as defined in Equation (4). A multistage adder tree and bias-processing unit then generate the corresponding output features. In this manner, the direction-aware capability of asymmetric convolutions is recast as a parameterized data-selection problem on a regular array, reconciling algorithmic expressiveness with FPGA resource reuse.

2.1.4. Comparator-Tree Pooling and Downsampling Unit for Continuous Feature Streams

To implement max pooling, the first value in each pooling window is stored in a register as the initial candidate, as illustrated in Figure 6. Each subsequent value is compared with the value held in the register; the larger value is retained, and the register is updated accordingly. This process continues until all values within the local window have been examined, such that only the maximum value is preserved. Once the comparisons for a local matrix block are complete, the maximum value stored in the register is emitted as the pooled output for that region. The same procedure is applied to each pooling window until the entire feature map has been processed. The pooled results are then transferred to on-chip buffers. By halving the spatial dimensions of the feature map, the pooling operation reduces the volume of subsequent feature processing and storage while preserving salient information. This downsampling process improves computational efficiency and supports effective model execution.

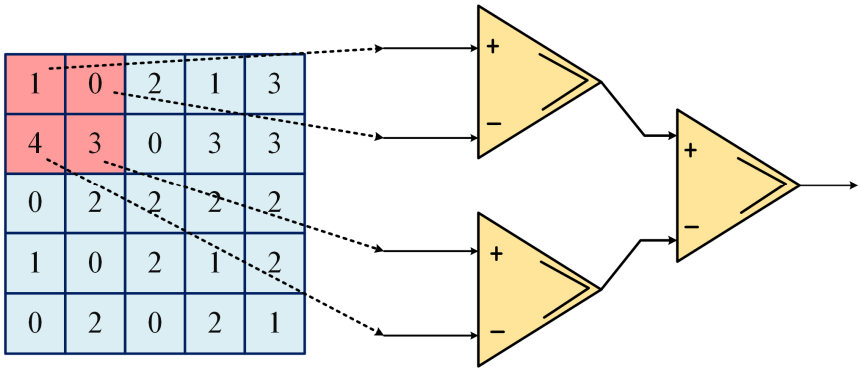


Figure 6. The implementation of the maximum pooling module.

For the c th input channel, the pooling result at location (i,j) in the output feature map can be expressed as (5):

$$Y_{c,i,j} = max\left(X_{c,2i,2j}, X_{c,2i,2j+1}, X_{c,2i+1,2j}, X_{c,2i+1,2j+1}\right) \quad (5)$$

where X and Y denote the feature maps before and after pooling, respectively; c is the channel index; and (i,j) denotes the coordinate in the output feature map. The pooling results are subsequently written to on-chip buffers for use by later network layers. Because both the pooling-window size and stride equal 2, the width and height of the output feature map are approximately halved. This reduces the computational cost of subsequent convolutional layers, on-chip buffering requirements, and off-chip data-transfer overhead, while preserving the most salient response within each local region.

Table 1 presents the register-transfer-level (RTL) implementation of Equation (5). For the current 2 × 2 pooling window, input ports win0, win1, win2, and win3 correspond to the four input feature values in Equation (5). The design first uses row1_max and row2_max to perform first-stage comparisons within the two rows. The two row maxima are then compared on the rising clock edge, and the final result is written to the max_val register. Thus, max_val corresponds to the pooled output Y. The design uses two first-stage

comparators and one second-stage comparator to form a two-level comparison tree, reducing combinational-logic depth while preserving the correctness of maximum-value computation.

Table 1. The Pseudocode for Max Pooling.

Maximum Pooling Pseudocode

module max_pool_2x2 (
input clk,
input [7:0] win0, win1,
input [7:0] win2, win3,
output reg [7:0] max_val
);
 	
wire [7:0] row1_max = (win0 > win1) ? win0: win1;
wire [7:0] row2_max = (win2 > win3) ? win2: win3;
 	
always @(posedge clk) begin
 	
max_val <= (row1_max > row2_max) ? row1_max: row2_max;
end
endmodule

2.1.5. Configurable Upsampling Unit

As shown in Figure 7. From a memory-access perspective, upsampling is a form of data replication. Accordingly, the upsampling module requires only a one-to-four parallel data-replication unit. The ARM processor first provides the data-transfer module with the number of input-feature channels and the external-memory address pointers for the input and output feature maps. Using control signals supplied by this module, the control unit schedules the data-loading module to read the input data sequentially. After one-to-four replication expands the spatial dimensions, the data-output module streams the upsampled results to external memory in real time. The control unit traverses the feature map along the channel dimension and implements upsampling through a dataflow architecture. By avoiding intermediate data buffering during input and output, this design substantially reduces demand for on-chip storage resources.

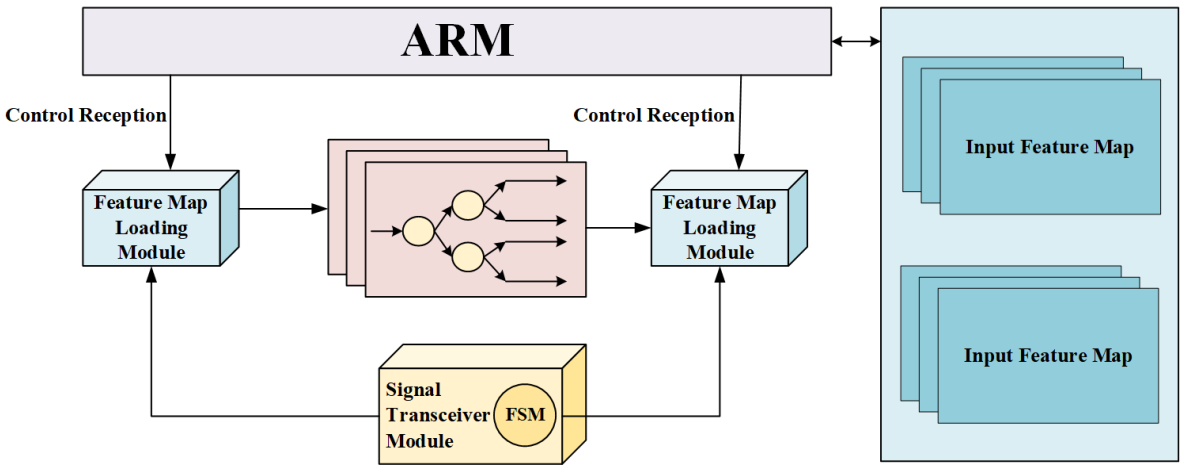


Figure 7. The upsampling module implementation.

2.2. Ship Inspection Framework

Vessel detection in UAV-based maritime monitoring is challenged by weak small-object features, severe background interference, and substantial variation in target orientation. Distant vessels occupy only a small number of pixels in the full image and consequently exhibit missing texture detail, blurred boundaries, and limited semantic representation. At the same time, nonvessel objects and patterns, including rocks, floating debris, sea-surface reflections, and wave crests, may share local contours or intensity characteristics with vessels, increasing the risk of false detections. To meet real-time requirements, lightweight networks commonly reduce backbone depth and channel width, but this compression can weaken feature-extraction capacity. After repeated downsampling, the original details and positional cues of small targets can be overwhelmed by background information, resulting in insufficient semantic information in high-level feature maps and a mismatch between the receptive field and target scale. Moreover, vessels in UAV imagery often appear as elongated rectangles with large aspect ratios or as irregular block-like structures. Depending on vessel heading and viewing angle, they may be oriented horizontally, vertically, or obliquely. Conventional square convolution kernels have symmetric receptive fields in the horizontal and vertical directions. Although they can extract general local features, they do not adequately emphasize a vessel's longitudinal edges, directional textures, and elongated geometry. It is therefore necessary to design feature-extraction modules that jointly capture multiscale information and directional cues, for example, by combining horizontal, vertical, and standard square-convolution branches, to strengthen representations of small, elongated, and multi-oriented vessels in complex backgrounds.

MPR-ShipNet employs an anchor-based detection head that produces predictions at three feature-map scales for vessels of different sizes. During training, each ground-truth box is matched to the predefined anchor with the highest IoU according to its width and height, and the corresponding grid location is assigned as a positive sample. The training loss consists of bounding-box regression loss, objectness loss, and classification loss. CIoU loss is used for bounding-box regression, while binary cross-entropy loss is used for objectness prediction and vessel-category classification. During inference, the objectness score and class probability are combined to obtain the final confidence score. Predictions with confidence scores below 0.25 are removed, after which class-wise non-maximum suppression with an IoU threshold of 0.45 is applied to eliminate duplicate detections.

Building on this analysis, this paper proposes MPR-ShipNet, a lightweight vessel-detection network for UAV-based maritime monitoring, through the co-design of the network architecture and FPGA data path. MPR-ShipNet denotes the Multi-Receptive-Field and Progressive-Context Reparameterized Ship Network. As shown in Figure 8, the encoder is built around a four-stage ResBlock backbone and incorporates a multi-receptive-field fine-grained enhancement module (MRFEM). Through multiscale and direction-aware convolutions, MRFEM enhances the edges, local textures, and elongated contours of small vessels. At the skip-connection stages, a progressive multiscale context aggregation module (PMCAM) integrates shallow localization cues with deep semantic features and suppresses interference from reflections, wave crests, and floating objects. To facilitate FPGA deployment, the network relies on operations that are amenable to pipelined implementation, including standard convolution, concatenation, upsampling, and activation. Horizontal 1×3, vertical 3×1, and standard 3×3 convolutions are uniformly mapped onto the FPGA's 3×3 PE array. Input selection and zero masking enable the reuse of different directional receptive fields without requiring dedicated asymmetric-convolution units. The decoder employs RepVGG modules to progressively restore feature-map resolution; their multibranch training structure can be fused into an equivalent single-branch convolution during inference, reducing on-chip resource consumption and control complexity. Network

channel widths and feature-map transfer schemes are further designed according to PE parallelism and on-chip memory constraints, while INT8 quantization reduces memory-access and data-transfer overhead. Finally, multiple output branches provide auxiliary supervision only during training and are removed during deployment, thereby jointly optimizing detection accuracy, hardware-resource utilization, and real-time performance.

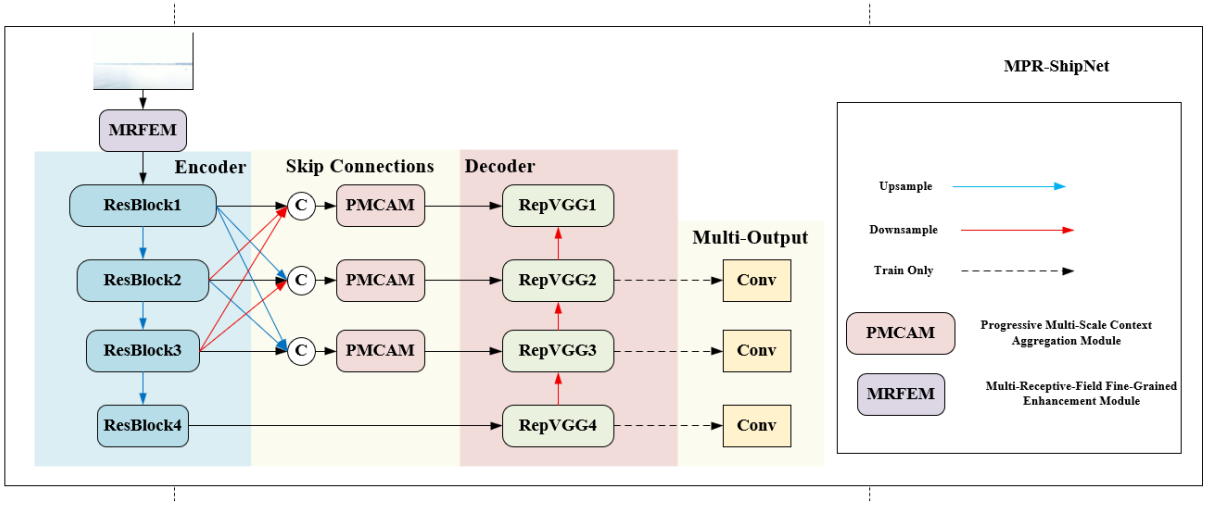


Figure 8. MPR-ShipNet primarily consists of five key components: input feature enhancement, encoder feature extraction, cross-layer skip connections, decoder feature restoration, and multi-output su-pervision.

To strengthen the fine-grained representation of small vessels in UAV-based maritime monitoring and alleviate semantic ambiguity induced by complex backgrounds, MPR-ShipNet incorporates a multi-receptive-field fine-grained enhancement module and a progressive multiscale context aggregation module. The former emphasizes the extraction of local details over different spatial extents at the early encoding stage, whereas the latter progressively integrates cross-layer contextual information through skip connections. Together, these modules improve small-target localization and background suppression.

As shown in Figure 9., the multi-receptive-field fine-grained enhancement module (MRFEM) is positioned between the network input and the encoder backbone. The module first performs channel projection through a 1×1 convolution and then distributes the resulting features across multiple parallel branches. These branches employ 3×3 and 5×5 convolutions to extract spatial features with different receptive fields. Smaller kernels help preserve vessel boundaries and local texture details, while larger receptive fields incorporate sea-surface context surrounding the vessel. Standard convolutions further enhance local pattern modeling while controlling the number of parameters and computational cost.

Let the projected input features enter K receptive-field branches after the 1×1 convolution. The output of the i branch can be expressed as (6):

$$F_i = B_i(Conv_{1\times1}(F_{in})), \quad i = 1, 2, \cdots, K \tag{6}$$

where B denotes the i-th parallel receptive-field branch. To avoid redundant feature accumulation arising from direct concatenation, MRFEM applies learnable weighted fusion to the branch outputs. Its output is defined as (7):

$$F_{MRFEM} = \sum_{i=1}^{K} \alpha_i F_i \tag{7}$$

where α denotes the learnable fusion weight associated with the i-th branch. This design enables the network to adaptively adjust the contribution of features at different scales during training, thereby improving the discriminability of small vessels with weak textures against complex maritime backgrounds. From an FPGA implementation perspective, the standard 3×3 and 5×5 convolutions in MRFEM are executed using the same shared 3×3 PE array. The 3×3 convolution is completed in a single window-processing pass, whereas the 5×5 convolution is completed through multiple passes by adjusting the feature-reading addresses and window offsets and accumulating the resulting partial sums. Depthwise convolution adopts the same spatial dataflow but processes each input channel independently without cross-channel accumulation. The subsequent 1×1 pointwise convolution performs channel fusion by enabling the central PE position and accumulating the results across input channels. The different MRFEM branches are scheduled sequentially and reuse the same feature buffer, weight buffer, line buffer, and PE array, thereby avoiding dedicated hardware for the 5×5 and depthwise separable convolutions., avoiding the need to construct separate compute arrays for each receptive-field branch.

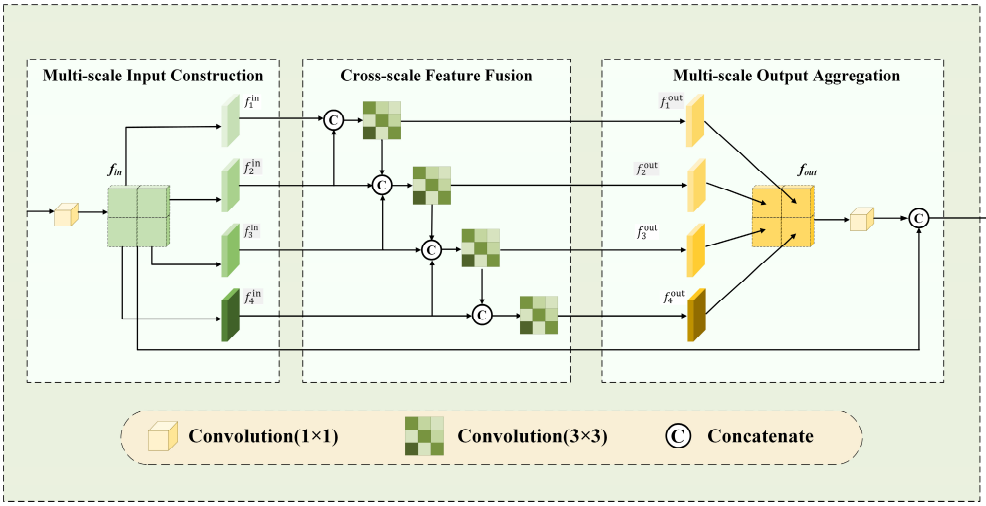


Figure 9. Multi-Receptive-Field Fine-Grained Enhancement Module.

As shown in Figure 10, the progressive multiscale context aggregation module (PM-CAM) is deployed within the skip connections between the encoder and decoder. The module first uses a 1×1 convolution to adjust the channel dimension of the input features and partitions them into multiple subfeature groups. These groups are then processed sequentially from shallow to deep. The first group passes through a 3×3 convolution to produce an initial contextual representation. Each subsequent group is concatenated with the output of the preceding stage along the channel dimension and then processed by a 3×3 convolution for feature updating. This procedure progressively propagates local details into higher-level contextual representations.

Let F denote the J-th input feature group and P denote the progressive aggregation output at stage J. The first feature group and the subsequent feature groups are computed as follows (8) and (9):

$$P_1 = Conv_{3\times3}(F_1) \tag{8}$$

$$P_j = Conv_{3\times3}\big(Concat\big(F_j,\ P_{j-1}\big)\big), \quad j = 2,\ \cdots,\ M \tag{9}$$

After all stage outputs have been generated, they are concatenated and fused through a 1×1 convolution, with a skip connection to the original input. The module output is given by:

$$F_{PMCAM} = Concat(F_{in},\ Conv_{1\times1}(Concat(P_1,\ \cdots,\ P_M))) \tag{10}$$

This cascaded aggregation scheme preserves shallow spatial-localization information while progressively expanding the effective receptive field and supplementing high-level semantic information. It thereby improves the distinction between vessels and background interference, including wave crests, specular reflections, and floating objects. PMCAM uses only 1×1 convolutions, 3×3 convolutions, and channel-wise concatenation. These operations map directly onto the convolution, data-movement, and feature-buffering functions of an FPGA accelerator. By parameterizing concatenation addresses and data-transfer volumes, the same hardware data path can process skip features at different scales.

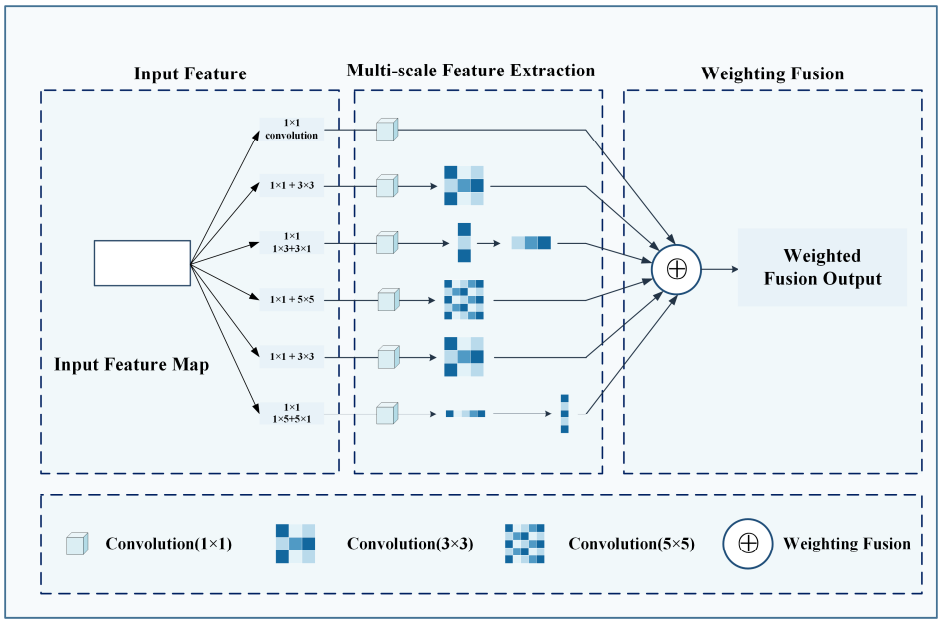


Figure 10. Progressive Multi-Scale Context Aggregation Module.

In summary, MRFEM enhances local details across multiple receptive fields, whereas PMCAM aggregates cross-layer contextual semantics. Together, they improve the representation of small, weak-textured vessel targets in complex backgrounds through fine-grained feature extraction and multiscale semantic fusion. They also provide a network foundation based primarily on regular operators for subsequent low-bit FPGA inference and pipelined acceleration.

Note on AI-assisted language editing: During the preparation of this manuscript, an AI-powered language tool (OpenAI's ChatGPT-4o) was utilized solely to refine English grammar, syntax, and overall readability. The tool was not involved in study design, data collection, data analysis, interpretation of results, or the generation of any figures or tables. All scientific content remains the original work of the authors.

3. Experimental Configuration and Dataset Description

To evaluate the effectiveness of the proposed method, all experiments were conducted on the hardware and software platform summarized in Table 2. Model training and testing were performed on a computing node equipped with an Intel Xeon Silver 4210R CPU (Intel Corporation, Santa Clara, CA, USA) operating at 2.40 GHz, an NVIDIA GeForce RTX 3060 GPU (NVIDIA Corporation, Santa Clara, CA, USA), and 16 GB of system memory. Hardware deployment and validation were conducted on a Xilinx Zynq-7100 FPGA platform (Xilinx, Inc., San Jose, CA, USA). The software environment consisted of Windows 10, Python 3.8.18, and the CUDA 12.2 toolkit for accelerated computation. Before being fed into the network, all images were resized to a fixed input resolution of 416×416 pixels. The same input resolution was used during training, validation, testing, and FPGA deployment

to ensure consistency across the reported accuracy and hardware-performance results. During training, stochastic gradient descent (SGD) was used as the optimizer. The model was trained for 200 epochs with a batch size of 8. To promote stable convergence, the initial learning rate was set to 10^{-3} and progressively decayed to 10^{-5} over the course of training.

Table 2. Experimental environment settings.

Hardware environment	CPU	Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz
	GPU	NVIDIA GeForce RTX 3060
	RAM	16G
	FPGA	Xilinx 7100
Software Environment	OS	Windows 10
	CUDA Toolkit	12.2
	Python	3.8.18
Training information	Optimizer	SGD
	Epoch	200
	Batch size	8
	Learning range	1×10^{-3}–1×10^{-5}

This study uses the publicly available SeaShips dataset for model training, validation, and performance evaluation. SeaShips contains 31,455 visible-light images extracted from videos captured by fixed shore-based surveillance cameras and provides approximately 40,077 annotated vessel instances. The annotations include vessel categories and bounding boxes covering six categories: ore carriers, bulk carriers, general cargo vessels, container ships, fishing vessels, and passenger ships. Because SeaShips does not provide a universally adopted official training-validation-test partition, the dataset was divided in this study at the image level using an 8:1:1 ratio. The resulting training, validation, and test sets contain 25,164, 3145, and 3146 images, respectively, with no overlap among the subsets.

Cross-dataset generalization is evaluated using SeaDronesSee, a UAV-based maritime dataset containing 5630 annotated RGB images acquired under varying flight altitudes, viewing angles, illumination conditions, and sea states. For this study, annotations for swimmers, jet skis, life-saving appliances, and buoys were excluded, and the evaluation was restricted to boat targets. To reconcile the different label taxonomies of the two datasets, predictions from the six SeaShips vessel categories were collapsed into a unified vessel category. Cross-domain inference was then performed using the model weights learned exclusively from SeaShips, without target-domain retraining, fine-tuning, or parameter adaptation. This setting provides a direct assessment of the model's generalization from shore-based surveillance imagery to UAV-acquired maritime imagery.

4. Results

4.1. Detection Performance Evaluation of MPR-ShipNet

4.1.1. Ablation Experiment

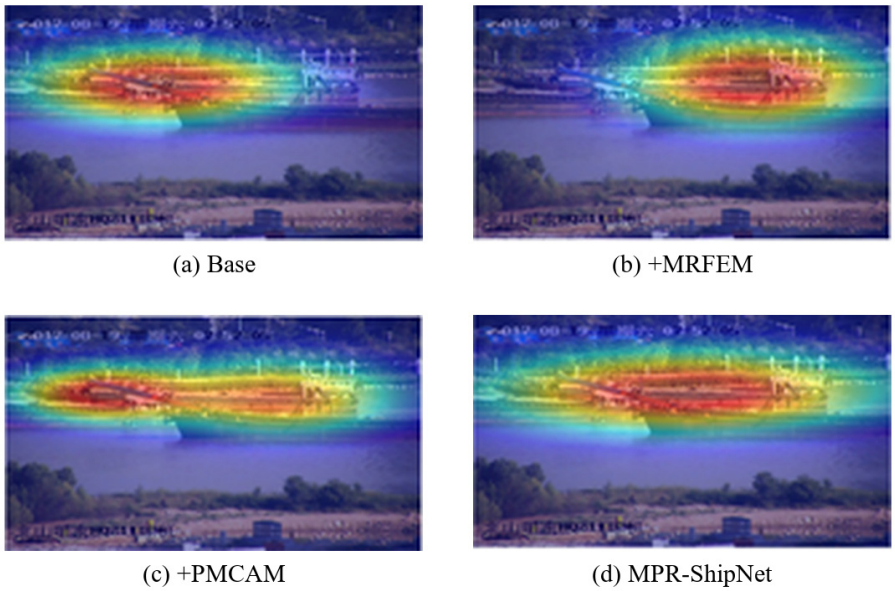
To evaluate the statistical reliability of the ablation results, each configuration was independently trained five times using different random seeds, while the dataset split, training parameters, and evaluation protocol were kept unchanged. Table 3 reports the mean ± standard deviation across the five trials. The baseline model achieves an mAP@0.5 of 90.2 ± 0.36%. Introducing MRFEM increases the mAP@0.5 to 91.8 ± 0.51%, whereas introducing PMCAM increases it to 92.4 ± 0.30%, corresponding to improvements of 1.6 and 2.2 percentage points, respectively. When both modules are incorporated, MPR-ShipNet achieves an mAP@0.5 of 94.5 ± 0.43%, a precision of 93.7 ± 0.32%, a recall of 95.6 ± 0.44%, and an F1-score of 94.6 ± 0.38%. Compared with the baseline, these results represent improvements of 4.3, 1.6, 4.1, and 3.2 percentage points, respectively. Two-sided Welch's t-tests show that the mAP@0.5 improvements obtained using MRFEM, PMCAM,

and both modules are statistically significant compared with the baseline ($p < 0.001$). The small standard deviations across all metrics indicate that the improvements are consistent across independent trials. Although the complete model increases the parameter count from 4.25 M to 5.04 M, it provides the largest and most stable performance improvement, confirming the complementary effects of MRFEM and PMCAM.

Table 3. Ablation experiment results of MRFEM and PMCAM on the SeaShips dataset.

MRFEM (①)	PMCAM (②)	mAP@0.5 (%)	Precision (%)	Recall (%)	F1-Score (%)	Params (M)
-	-	90.2 ± 0.36	92.1 ± 0.70	91.5 ± 0.34	91.8 ± 0.46	4.25
✓	-	91.8 ± 0.51	93.5 ± 0.34	92.8 ± 0.64	93.1 ± 0.22	4.45
-	✓	92.4 ± 0.30	93.4 ± 0.47	93.6 ± 0.44	93.5 ± 0.15	4.52
✓	✓	94.5 ± 0.43	93.7 ± 0.32	95.6 ± 0.44	94.6 ± 0.38	5.04

As shown in Figure 11. To qualitatively assess the effects of MRFEM and PMCAM on feature extraction, feature-response heatmaps were visualized for the different ablation models. As shown in the figure, the baseline model attends to the main vessel region, but its high-response areas are relatively dispersed and extend to the surrounding water surface and background. This pattern indicates limited capacity to represent vessel structure and suppress background interference. After MRFEM is introduced, the activation response more completely covers the vessel body, bow, and stern, suggesting that the module improves the extraction of information across different scales and structural patterns. With PMCAM, responses over vessel regions become more pronounced, whereas activations over the surrounding water background are reduced. This result indicates that PMCAM strengthens foreground-object features and suppresses background interference. When both modules are incorporated, MPR-ShipNet produces the most concentrated, continuous, and complete high-response distribution over vessel regions while maintaining low responses in nontarget areas. These findings demonstrate that MRFEM and PMCAM are complementary in multiscale feature enhancement and the joint refinement of foreground and background representations.



(a) Base (b) +MRFEM

(c) +PMCAM (d) MPR-ShipNet

Figure 11. Vessel feature-response heatmaps under different module configurations. *Base* denotes the baseline model without MRFEM or PMCAM; *+MRFEM* and *+PMCAM* denote the baseline model augmented with the corresponding module; and *MPR-ShipNet* denotes the model incorporating both MRFEM and PMCAM. Red and yellow regions indicate stronger feature responses, whereas blue regions indicate weaker responses.

4.1.2. Generalization Evaluation

As shown in Table 4 MPR-ShipNet was compared with FFCA-YOLO, YOLO-World, ORFENet, YOLOoW, RT-DETR, and DAB-DETR to evaluate its generalization performance on the SeaShips dataset. MPR-ShipNet achieves a precision of 93.7%, a recall of 95.6%, an mAP@0.5 of 94.5%, and an F1-score of 94.6%. Although some recent studies have reported higher values for individual detection metrics on the SeaShips dataset, the objective of this work is not to pursue the highest detection accuracy alone, but to achieve balanced performance across detection accuracy, model complexity, computational cost, inference latency, power consumption, and FPGA resource utilization. Compared with the second-best result for each metric among the evaluated methods, MPR-ShipNet improves precision, recall, mAP@0.5, and F1-score by 1.2, 3.5, 1.9, and 2.9 percentage points, respectively. These results indicate that the model effectively reduces both false positives and missed detections while providing more complete and stable detection results. Relative to Transformer-based detectors, MPR-ShipNet improves mAP@0.5 by 3.3 and 5.9 percentage points over RT-DETR and DAB-DETR, respectively, while improving F1-score by 4.0 and 6.8 percentage points. This finding suggests stronger feature-extraction and object-recognition capability under complex backgrounds and scale variation. In addition, MPR-ShipNet contains only 5.04 M parameters, 57.3% fewer than YOLOoW, the comparison method with the fewest parameters. Its computational cost is only 12.31 GFLOPs, 54.4% lower than that of YOLOoW and substantially below all competing methods. Thus, MPR-ShipNet provides competitive detection accuracy with relatively low model complexity and computational cost. More importantly, its network structure is jointly designed with the FPGA accelerator, enabling a balanced combination of detection performance, inference efficiency, power consumption, and hardware resource utilization for resource-constrained edge deployment.

Table 4. Performance evaluation of seven different methods.

Methods	Model	Publication	Precision%	Recall%	mAP@0.5%	F1%	Param(M)	GFLOPs
N/A	MPR-ShipNet	Ours	93.7	95.6	94.5	94.6	5.04	12.31
single-stage detectors algorithm	FFCA-YOLO [30]	TGRS2024	91.2	90.5	91.8	90.8	12.40	28.5
	YOLO-world [31]	CVPR2024	88.5	89.2	89.5	88.8	14.20	32.4
	ORFENet [32]	TGRS2024	92.5	91.0	92.6	91.7	16.50	38.2
	YOLOoW [33]	TGRS2024	90.8	92.1	92.0	91.4	11.80	27.0
Transformer-based Global Object Detection Algorithm	RT-DETR [34]	CVPR2024	91.5	89.8	91.2	90.6	20.20	46.5
	DAB-DETR [35]	CVPR2024	87.2	88.5	88.6	87.8	24.50	56.0

Figure 12 presents feature-response heatmaps for different methods on the SeaShips dataset. Red and yellow regions indicate stronger feature responses, whereas blue regions indicate weaker responses. MPR-ShipNet consistently concentrates high responses over vessel bodies and boundaries, while maintaining relatively complete and continuous responses for distant small targets, multiple vessels, low-contrast targets, and complex sea-surface backgrounds. At the same time, most nontarget regions retain low responses, indicating that the model effectively suppresses interference from water surfaces, shorelines, and the sky. By contrast, FFCA-YOLO, YOLO-World, ORFENet, YOLOoW, RT-DETR, and DAB-DETR exhibit more dispersed responses in some scenes, including incomplete activations within targets, insufficient attention to small vessels, and elevated responses in background regions. These results indicate that MPR-ShipNet learns more target-relevant deep features, strengthens the representation of vessel structures and boundaries, and reduces the influence of complex backgrounds on detection performance.

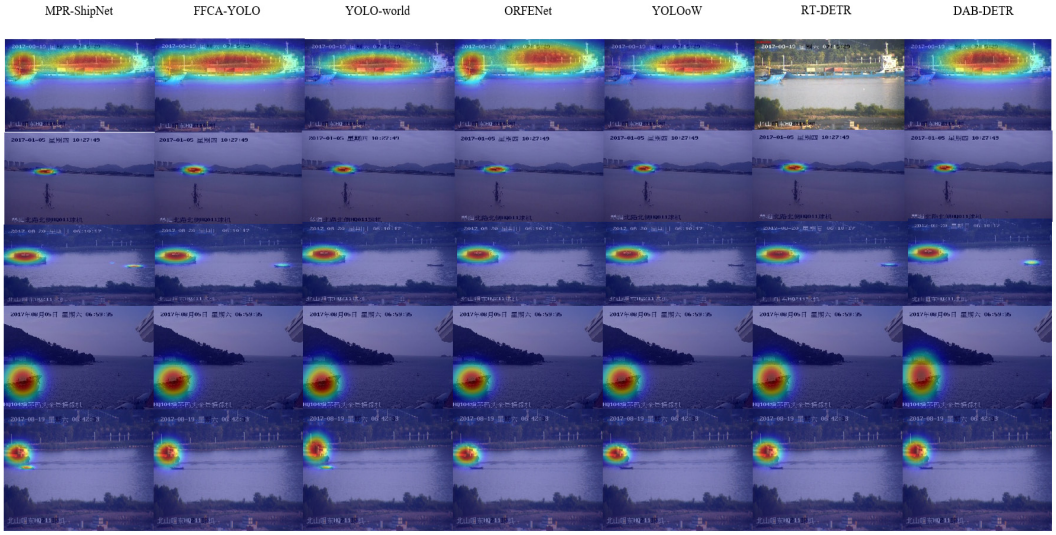


Figure 12. Visualization of Feature-Response Heatmaps for Different Methods on the SeaShips Dataset.

Figure 13 presents selected qualitative detection results for MPR-ShipNet, FFCA-YOLO, YOLO-World, ORFENet, YOLOoW, RT-DETR, and DAB-DETR on the SeaShips dataset. The selected scenes encompass variation in target scale, imaging distance, illumination, background conditions, and vessel density. MPR-ShipNet maintains stable detection performance for both large vessels and distant small targets. Its predicted bounding boxes more completely cover vessel hulls and provide more accurate localization. In challenging scenes involving multiple vessels, low contrast, small targets, and partial occlusion, MPR-ShipNet continues to identify multiple vessel targets effectively, whereas some comparison methods exhibit missed detections, low-confidence predictions, or offset bounding boxes. In particular, for targets with complex background textures or colors similar to those of the water surface, MPR-ShipNet demonstrates stronger target discrimination and more complete detections. These findings suggest that MRFEM contributes to multiscale feature extraction, while PMCAM helps suppress background interference and refine target features. Overall, the qualitative results show that MPR-ShipNet improves the accuracy and stability of vessel detection in complex maritime scenes.

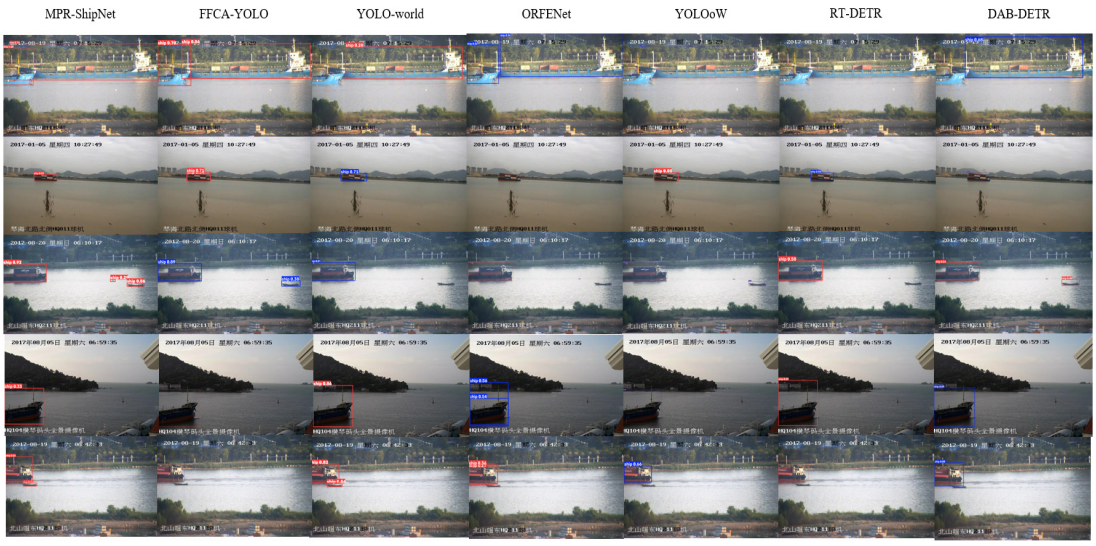


Figure 13. Partial detection results of different methods on the SeaShips dataset. From left to right: MPR-ShipNet, FFCA-YOLO, YOLO-World, ORFENet, YOLOoW, RT-DETR, and DAB-DETR.

4.1.3. Sensitivity Analysis Under Image Corruptions

This experiment was designed to examine the sensitivity and performance degradation boundaries of MPR-ShipNet under image corruptions relevant to practical deployment, rather than to rank the relative robustness of different detection models. Therefore, Gaussian noise, salt-and-pepper noise, gamma correction, brightness variation, Gaussian blur, and occlusion were applied only to the proposed model under identical test conditions. The results are reported in Table 5. Relative to the mAP@0.5 of 94.5% under undisturbed conditions, the model maintains high detection accuracy under mild perturbations, with mAP@0.5 remaining above 92.0% in all cases. In particular, it achieves an mAP@0.5 of 93.6% under gamma correction with a value of 0.8, indicating good adaptability to modest illumination variation and noise. As perturbation severity increases, model performance declines, but MPR-ShipNet remains reasonably robust. Under Gaussian noise with an intensity of 0.5, salt-and-pepper noise with a probability of 0.2, strong brightness enhancement, and gamma correction with a value of 1.5, mAP@0.5 remains at 85.5%, 87.0%, 88.0%, and 88.8%, respectively. These results indicate continued resilience to noise and illumination changes. By contrast, severe blur, substantial brightness reduction, and extensive occlusion have more pronounced effects. When the Gaussian-blur intensity is 7, the brightness-reduction coefficient is 0.4, and the occlusion ratio is 50%, mAP@0.5 declines to 84.2%, 85.2%, and 81.2%, respectively. Occlusion produces the largest performance loss, suggesting that the removal of critical structural and semantic information materially impairs detection. Overall, MPR-ShipNet maintains relatively stable performance under mild and moderate image corruptions, while severe blur, brightness reduction, and occlusion cause clear performance degradation. Because the comparison models were not evaluated under the same corruptions, these results should be interpreted as a within-model sensitivity analysis and do not establish superior robustness relative to other methods.

Table 5. Sensitivity of MPR-ShipNet to different image corruption conditions.

Corruption	Level	P%	R%	mAP@50%
Gaussian Noise	$\sigma = 0.1$	92.5	94.2	93.1
	$\sigma = 0.3$	89.8	91.5	90.2
	$\sigma = 0.5$	85.2	86.8	85.5
Salt & Pepper	$p = 0.05$	92.8	94.5	93.4
	$p = 0.1$	90.5	92.1	91.0
	$p = 0.2$	86.5	88.2	87.0
Gamma Correction	$\gamma = 0.8$	93.0	94.8	93.6
	$\gamma = 1.2$	92.2	93.5	92.5
	$\gamma = 1.5$	88.5	89.8	88.8
(Brightness Enhance)	factor = 1.2	92.8	94.5	93.3
	factor = 1.3	91.0	92.8	91.5
	factor = 1.5	87.5	89.0	88.0
Brightness Reduce	factor = 0.8	92.5	94.0	92.9
	factor = 0.6	89.5	91.2	90.0
	factor = 0.4	84.8	86.5	85.2
Gaussian Blur	$\sigma = 3$	91.5	93.2	92.0
	$\sigma = 5$	88.2	90.5	89.0
	$\sigma = 7$	83.5	85.8	84.2
Occlusion	10%	92.0	93.5	92.5
	30%	88.5	89.0	88.5
	50%	82.0	81.5	81.2

4.1.4. Cross-Dataset Generalization Evaluation on SeaDronesSee

Seven models—including MPR-ShipNet, RT-DETR, FFCA-YOLO, DAB-DETR, ORFENet, YOLOoW, and YOLO-World—were all trained on the same SeaShips training split and subsequently evaluated on the same SeaDronesSee V2 validation set, comprising 1547 images. The validation set, rather than the official test set, was used for this evaluation. Model

weights and inference settings were determined solely using source-domain training and validation data and remained fixed throughout the SeaDronesSee evaluation. Therefore, Table 6 compares the performance of these models under a unified source-only cross-dataset evaluation protocol, rather than comparing models trained exclusively on the source domain with models trained on the target dataset. Table 6 presents the evaluation results on the SeaDronesSee V2 validation set. MPR-ShipNet achieves a precision of 87.5%, a recall of 86.2%, an mAP@0.5 of 86.4%, an mAP@0.5:0.95 of 56.8%, and an F1-score of 86.8%. RT-DETR achieves the second-best result for each of these metrics. Compared with RT-DETR, MPR-ShipNet improves precision, recall, mAP@0.5, mAP@0.5:0.95, and F1-score by 3.4, 3.7, 4.1, 7.6, and 3.5 percentage points, respectively. These results indicate that MPR-ShipNet provides higher detection accuracy than the compared methods in the reported evaluation.

Table 6. Cross-dataset generalization performance on the SeaDronesSee dataset.

Model	Precision/%	Recall/%	mAP@0.5/%	mAP@0.5:0.95/%	F1/%
MPR-ShipNet (Ours)	87.5	86.2	86.4	56.8	86.8
FFCA-YOLO	84.1	82.5	82.3	49.2	83.3
YOLO-world	81.3	80.8	78.5	47.5	81.0
ORFENet	79.5	78.2	76.8	45.1	78.8
YOLOoW	78.2	76.5	75.4	44.3	77.3
RT-DETR	75.6	77.1	73.2	41.8	76.3
DAB-DETR	73.8	75.4	71.5	39.5	74.6

Figure 14 provides a qualitative comparison of the detection results produced by the seven models on representative SeaDronesSee scenes. MPR-ShipNet detects vessel targets more completely and consistently across sparse, dense, and small-target scenarios. In comparison, some competing methods exhibit missed detections, lower-confidence predictions, or incomplete responses to distant vessels, particularly in scenes containing multiple small targets and complex sea-surface backgrounds. The qualitative results are consistent with the quantitative improvements reported in Table 6 and further demonstrate the robustness of MPR-ShipNet under real UAV maritime imaging conditions.

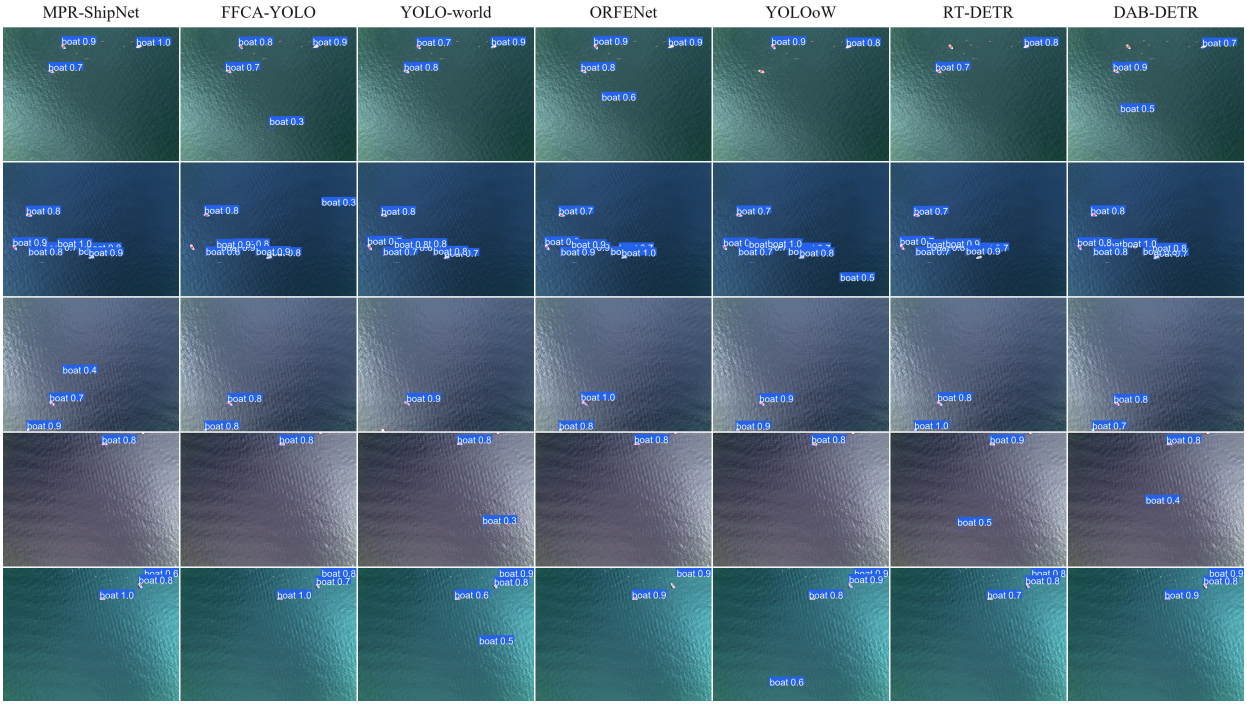


Figure 14. Qualitative comparison of vessel-detection results obtained by different models on the SeaDronesSee dataset.

4.2. FPGA Implementation and Acceleration Performance Evaluation

Hardware Performance Comparison of FPGA-Based Neural Network Accelerators

As shown in Table 7, the proposed INT8-quantized MPR-ShipNet accelerator is deployed on an XC7Z100 FPGA to address vessel-detection requirements in UAV-based maritime monitoring. Operating at 200 MHz, the design achieves a computational throughput of 127.45 GOPS, a per-frame inference latency of 74 ms, and a processing rate of approximately 13.5 frames per second. These characteristics can provide timely onboard or near-UAV inference for UAV cruising surveillance, sea-surface target screening, and abnormal-vessel warning. Table 7 compares the hardware-acceleration performance of representative FPGA-based CNN implementations. Although the cited studies employ different CNN architectures and address different application tasks, the comparison does not involve task-specific accuracy or application-level performance. Instead, it focuses on hardware-related indicators, including clock frequency, numerical precision, DSP and LUT utilization, computational throughput, power consumption, energy efficiency, and normalized resource efficiency. Because inference latency is also affected by network complexity and input configuration, it is reported as an implementation reference rather than as the sole basis for ranking the accelerators. For UAV platforms, payload capacity, power supply, thermal dissipation, and flight endurance are all tightly constrained, making low power consumption and low resource utilization particularly important. The proposed accelerator achieves 127.45 GOPS at 200 MHz with a power consumption of 2.59 W. It provides an energy efficiency of 49.2 GOPS/W, a DSP efficiency of 0.414 GOPS/DSP, and a LUT efficiency of 8.65×10^{-3} GOPS/LUT. These results indicate that the proposed hardware-software co-design achieves a favorable balance among computational throughput, energy consumption, and FPGA resource utilization.

Table 7. Performance Comparison with Related Hardware Acceleration Studies.

	Ref. [36]	**Ref. [37]**	**Ref. [38]**	**Ref. [39]**	**Ours**
FPGA Development Board	ZYNQ7020	ZC706	AXU15EG	AX7Z20	XC7Z100
Network Model	YOLOv4-Tiny	DCNN	2D–3D CNN	YOLOv2-Tiny	Ours
Clock Frequency (MHz)	180	200	200	100	200
Quantization	fixed16	INT8	INT8	fixed16	INT8
DSP Utilization	216	610	1592	224	308
LUTs Utilization	41,953	84,000	47,012	37,219	19,241
GOPS	9.24	464.5	150.0	23.00	166.35
Power/W	2.86	10.25	7.84	3.41	2.59
Inference Latency/ms	376	27.8	24.7	301.2	74
Energy Efficiency/(GOPS/W)	3.23	45.3	19.1	7.0	64.24
GOPS/DSP	0.043	0.761	0.094	0.103	0.540
GOPS/LUTs	2.20×10^{-4}	5.53×10^{-3}	3.19×10^{-3}	6.18×10^{-4}	8.65×10^{-3}

To evaluate the edge-deployment suitability of the proposed method, single-image inference time was measured on SeaShips and SeaDronesSee V2 using CPU, CPU + GPU, Raspberry Pi 4B+, and ZYNQ-7100 FPGA platforms, with the results summarized in Table 8. Detection accuracy on SeaDronesSee V2 was evaluated using the validation set comprising 1547 images. The CPU and CPU + GPU platforms executed the original FP32 model, whereas the Raspberry Pi 4B+ and ZYNQ-7100 platforms executed the same INT8-quantized model. On SeaShips, the FP32 and INT8 implementations achieved mAP@0.5 values of 94.5% and 91.5%, respectively, representing an absolute decrease of 3.0 percentage points. On SeaDronesSee V2, the corresponding values were 86.4% and 83.6%, representing an absolute decrease of 2.8 percentage points. The CPU + GPU platform achieved the shortest single-image inference times, at 34 ms on SeaShips and 92 ms on SeaDronesSee V2, but

its reported power consumption of 174 W may limit its suitability for power-constrained deployment. The Raspberry Pi 4B+ consumed 3 W but required 233 ms and 580 ms per image on the two datasets, respectively. In comparison, the ZYNQ-7100 FPGA required 74 ms per image on SeaShips and 185 ms per image on SeaDronesSee V2, corresponding to approximately 13.5 and 5.4 frames per second, respectively, with a reported power consumption of 2.59 W. The longer inference times observed on SeaDronesSee V2 may partly reflect its more complex scenes and target distributions, which can increase detection post-processing overhead; however, stage-level timing is needed to establish the contribution of this factor. Compared with the CPU and Raspberry Pi 4B+ platforms, the FPGA reduced single-image inference time by 22.1% and 68.2% on SeaShips and by 35.1% and 68.1% on SeaDronesSee V2, respectively. Although slower than the CPU + GPU platform, the FPGA offered a practical balance among detection accuracy, inference time, and reported power consumption under the tested conditions, supporting its potential for resource-constrained vessel-screening and alerting applications.

Table 8. Performance Comparison Across Hardware Platforms.

Metric	CPU	CPU + GPU	Raspberry Pi+	ZYNQ7100
Number of test images	200	200	200	200
Numerical precision	FP32	FP32	INT8	INT8
Inference latency (SeaShips)	95	34	233	74
Inference latency (SeaDronesSee V2)	285	92	580	185
SeaShips mAP@0.5 (%)	94.5	94.5	91.5	91.5
SeaDronesSee V2 mAP@0.5 (%)	86.4	86.4	83.6	83.6
Power consumption (W)	93	174	3	2.59

5. Discussion

The experimental results indicate that MRFEM and PMCAM improve ship detection through complementary mechanisms. MRFEM focuses on enhancing the boundaries, directional textures, and elongated structures of small ships, whereas PMCAM integrates shallow localization information with deep semantic features to suppress interference from sea-surface reflections and waves. Their combination produces a more pronounced improvement in recall, suggesting that the primary advantage of the model lies in reducing missed detections of small and low-contrast ships. At the hardware level, uniformly mapping 1×3, 3×1, and 3×3 convolutions onto a shared PE array reduces duplicated computational units and improves architectural reconfigurability. However, zero masking during asymmetric convolution leaves some multipliers temporarily inactive, while INT8 quantization introduces a certain degree of accuracy loss. The proposed design therefore represents a balanced trade-off among detection accuracy, resource utilization, inference latency, and power consumption. Han et al. [40] reported an mAP@0.5 of 99.1% on SeaShips using GEW-YOLO, compared with 94.5% for our FP32 model, although these literature-reported results do not constitute a comparison under a verified common experimental protocol. Gao et al. [41] reported an mAP@0.5 of 98.5% on SeaShips using DSSM–LightNet, whereas our work emphasizes FPGA hardware-software co-design rather than the highest detection accuracy alone.

Future research should focus on improving model stability under severe occlusion, strong blur, and low-light conditions. Potential directions include occlusion-aware data augmentation, temporal feature fusion across consecutive UAV video frames, and multimodal fusion of visible-light, infrared, or radar information. Because the current experiments are primarily based on the SeaShips dataset and a single FPGA platform, additional ship-detection datasets, cross-scene evaluations, and real-world UAV flight experiments are required to further validate the model's generalization capability. Accordingly, the present

study focuses on validating the proposed FPGA hardware-software co-design under laboratory conditions that emulate the computational, power, and real-time constraints of UAV edge deployment, while full system integration and validation on an actual UAV platform remain subjects of future work. Quantization-aware training, mixed-precision computation, and parallel scheduling of directional convolutions should also be investigated to reduce quantization-induced accuracy loss and further improve PE-array utilization.

6. Conclusions

This paper proposes MPR-ShipNet, a lightweight vessel-detection method based on network-hardware co-design, to address small target scales, pronounced orientation variation, complex maritime backgrounds, and the limited computational and energy resources of UAV platforms. At the network level, a multi-receptive-field fine-grained enhancement mechanism strengthens vessel edges, textures, and elongated structural features, while a progressive multiscale context aggregation mechanism integrates shallow localization cues with deep semantic information. These components improve the recognition of small, low-contrast, and background-cluttered targets. At the hardware level, a parameter-reconfigurable FPGA accelerator uniformly maps 1×3, 3×1, and 3×3 convolutions onto a shared 3×3 processing-element array. Combined with multichannel pipelining, data reuse, and INT8 quantization, this architecture reduces on-chip resource use, memory-access overhead, and inference power consumption. Ablation experiments show that, when both feature-enhancement modules are included, the model achieves an mAP@0.5 of 94.5%, a precision of 93.7%, a recall of 95.6%, and an F1-score of 94.6%. These values improve on the baseline by 4.3, 1.6, 4.1, and 3.2 percentage points, respectively. Feature-response heatmaps further show that the model focuses more completely and consistently on vessel bodies and boundaries while effectively suppressing sea-surface background interference. On the SeaShips dataset, MPR-ShipNet outperforms the compared detection methods under the same evaluation protocol. In the evaluation on SeaDronesSee, MPR-ShipNet achieves a precision of 87.5%, a recall of 86.2%, an mAP@0.5 of 86.4%, an mAP@0.5:0.95 of 56.8%, and an F1-score of 86.8%. showing that it retains favorable vessel-detection performance when transferred from shore-based surveillance imagery to UAV-acquired maritime imagery.

Robustness experiments indicate that the model maintains high detection accuracy under noise, illumination variation, and mild blur. However, performance declines substantially under severe occlusion, strong blur, and pronounced brightness reduction, suggesting that further work is needed to improve robustness to structural information loss and extreme imaging conditions. Hardware-deployment results show that the XC7Z100 FPGA accelerator delivers 127.45 GOPS throughput, 74 ms per-frame inference latency, and 2.59 W power consumption at 200 MHz, corresponding to an energy efficiency of 49.2 GOPS/W. Overall, MPR-ShipNet achieves a favorable balance among detection accuracy, robustness, computational complexity, and hardware-deployment efficiency, offering a practical solution for UAV-based maritime inspection, sea-surface target screening, and abnormal-vessel warning.

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

Funding: This research was funded by the Technology Innovation Center for Emergency Surveying and Mapping, MNR (No. YJCX-2026-ZD-01), and the National Natural Science Foundation of China (No. 42601750).

Data Availability Statement: The publicly available SeaShips dataset analyzed in this study can be accessed at https://github.com/jiaming-wang/SeaShips (accessed on 10 September 2026). The processed data, detailed experimental results, trained model parameters, and supporting materials are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the author(s) used OpenAI's ChatGPT-4o for the purposes of English language editing and grammar checking. The authors have 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. Xie, Y.; Feng, D.; Shen, X.; Liu, Y.; Zhu, J.; Hussain, T.; Baik, S.W. Clustering feature constraint multiscale attention network for shadow extraction from remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 4705414. [CrossRef]
2. Koubaa, A.; Ammar, A.; Abdelkader, M.; Alhabashi, Y.; Ghouti, L. AERO: AI-Enabled Remote Sensing Observation with Onboard Edge Computing in UAVs. *Remote Sens.* **2023**, *15*, 1873. [CrossRef]
3. McEnroe, P.; Wang, S.; Liyanage, M. A Survey on the Convergence of Edge Computing and AI for UAVs: Opportunities and Challenges. *IEEE Internet Things J.* **2022**, *9*, 15435–15459. [CrossRef]
4. Paolo, F.S.; Kroodsma, D.; Raynor, J.; Hochberg, T.; Davis, P.; Cleary, J.; Marsaglia, L.; Orofino, S.; Thomas, C.; Halpin, P. Satellite Mapping Reveals Extensive Industrial Activity at Sea. *Nature* **2024**, *625*, 85–91. [CrossRef] [PubMed]
5. McCarthy, M.J.; Hughes, D.; Patterson, J.; Channer, M.A.; McGee, M.; Meade, S.; Moss, K.; Trask, P. Automated Remote Sensing Tools to Counter Illicit Maritime Activity: Vessel Detection, Bathymetry and Topography from WorldView Imagery. *Int. J. Remote Sens.* **2023**, *44*, 6117–6130. [CrossRef]
6. Xie, X.; Li, L.; An, Z.; Lu, G.; Zhou, Z. Small Ship Detection Based on Hybrid Anchor Structure and Feature Super-Resolution. *Remote Sens.* **2022**, *14*, 3530. [CrossRef]
7. Zhao, T.; Wang, Y.; Li, Z.; Gao, Y.; Chen, C.; Feng, H.; Zhao, Z. Ship Detection with Deep Learning in Optical Remote-Sensing Images: A Survey of Challenges and Advances. *Remote Sens.* **2024**, *16*, 1145. [CrossRef]
8. Tian, Y.; Liu, J.; Zhu, S.; Xu, F.; Bai, G.; Liu, C. Ship Detection in Visible Remote Sensing Image Based on Saliency Extraction and Modified Channel Features. *Remote Sens.* **2022**, *14*, 3347. [CrossRef]
9. Wu, R.; Zhang, R.; Cai, J.; Cai, K.; Wu, T.; Lv, J.; Shi, Y.; Liu, G. Glacial lake extraction framework based on coupling of GEE and historical glacial lake position. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 2001005. [CrossRef]
10. Lv, J.; Zhang, R.; Yu, B.; Pang, J.; Liao, M.; Liu, G. A GPS-IR method for retrieving NDVI from integrated dual-frequency observations. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 8015005. [CrossRef]
11. Sui, B.; Cao, Y.; Bai, X.; Zhang, S.; Wu, R. BIBED-Seg: Block-in-block edge detection network for guiding semantic segmentation task of high-resolution remote sensing images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 1531–1549. [CrossRef]
12. Yang, L.; Zhong, J.; Zhang, Y.; Bai, S.; Li, G.; Yang, Y.; Zhang, J. An improving Faster-RCNN with multi-attention ResNet for small target detection in intelligent autonomous transport with 6G. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 7717–7725. [CrossRef]
13. Xu, W.; Feng, Z.; Wan, Q.; Xie, Y.; Feng, D.; Zhu, J.; Liu, Y. Building height extraction from high-resolution single-view remote sensing images using shadow and side information. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 6514–6528. [CrossRef]
14. Wang, Y.; Song, X.; Feng, L.; Zhai, Y.; Zhao, Z.; Zhang, S.; Wang, Q. MCI-GLA plug-in suitable for YOLO series models for transmission line insulator defect detection. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 9002912. [CrossRef]
15. Xie, Y.; Zhan, N.; Zhu, J.; Xu, B.; Chen, H.; Mao, W.; Luo, X.; Hu, Y. Landslide extraction from aerial imagery considering context association characteristics. *Int. J. Appl. Earth Obs. Geoinf.* **2024**, *131*, 103950. [CrossRef]
16. Lin, Y.; Li, J.; Shen, S.; Wang, H.; Zhou, H. GDRS-YOLO: More efficient multiscale features fusion object detector for remote sensing images. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 6008505. [CrossRef]
17. Farrukh, F.U.D.; Zhang, C.; Jiang, Y.; Zhang, Z.; Wang, Z.; Wang, Z.; Jiang, H. Power efficient tiny YOLO CNN using reduced hardware resources based on Booth multiplier and WALLACE tree adders. *IEEE Open J. Circuits Syst.* **2020**, *1*, 76–87. [CrossRef]
18. Cao, S.; Feng, D.; Liu, S.; Xu, W.; Chen, H.; Xie, Y.; Zhang, H.; Pirasteh, S.; Zhu, J. BEMRF-Net: Boundary enhancement and multiscale refinement fusion for building extraction from remote sensing imagery. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 16342–16358. [CrossRef]
19. Li, B.; Huang, S.; Zhong, G. LTEA-YOLO: An improved YOLOv5s model for small object detection. *IEEE Access* **2024**, *12*, 99768–99778. [CrossRef]
20. Gao, T.; Liu, Z.; Wu, G.; Li, Z.; Wen, Y.; Liu, L.; Chen, T.; Zhang, J. A self-supplementary and revised network for remote sensing object detection. *IEEE Geosci. Remote Sens. Lett.* **2024**, *21*, 6003105. [CrossRef]

21. Bhavana, N.; Kodabagi, M.M.; Kumar, B.M.; Ajay, P.; Muthukumaran, N.; Ahilan, A. POT-YOLO: Real-time road potholes detection using edge segmentation-based YOLO V8 network. *IEEE Sens. J.* **2024**, *24*, 24802–24809. [CrossRef]
22. Hua, W.; Zhang, C.; Xie, W.; Jin, X. Polarimetric SAR image classification based on ensemble dual-branch CNN and superpixel algorithm. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 2759–2772. [CrossRef]
23. Behera, T.K.; Bakshi, S.; Nappi, M.; Sa, P.K. Superpixel-based multiscale CNN approach toward multiclass object segmentation from UAV-captured aerial images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 1771–1784. [CrossRef]
24. Hedjam, R.; Abdesselam, A.; Melgani, F. Change detection in unlabeled optical remote sensing data using Siamese CNN. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 4178–4187. [CrossRef]
25. Chen, X.; Han, Z.; Li, Y.; Ma, M.; Mei, S.; Cheng, W. Attention-aware deep feature embedding for remote sensing image scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 1171–1184. [CrossRef]
26. Lu, W.; Lan, C.; Niu, C.; Liu, W.; Lyu, L.; Shi, Q.; Wang, S. A CNN-Transformer hybrid model based on CSWin Transformer for UAV image object detection. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 1211–1231. [CrossRef]
27. Sanjeet, S.; Sahoo, B.D.; Fujita, M. Energy-efficient FPGA implementation of power-of-2 weights-based convolutional neural networks with low bit-precision input images. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 741–745. [CrossRef]
28. Li, J.; Un, K.-F.; Yu, W.-H.; Mak, P.-I.; Martins, R.P. An FPGA-based energy-efficient reconfigurable convolutional neural network accelerator for object recognition applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 3143–3147. [CrossRef]
29. Fang, Z. A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 5514–5526. [CrossRef] [PubMed]
30. Zhang, Y.; Ye, M.; Zhu, G.; Liu, Y.; Guo, P.; Yan, J. FFCA-YOLO for small object detection in remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5611215. [CrossRef]
31. Cheng, T.; Song, L.; Ge, Y.; Liu, W.; Wang, X.; Shan, Y. YOLO-World: Real-time open-vocabulary object detection. In *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: New York, NY, USA, 2024.
32. Liu, D.; Zhang, J.; Qi, Y.; Wu, Y.; Zhang, Y. Tiny object detection in remote sensing images based on object reconstruction and multiple receptive field adaptive feature enhancement. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5616213. [CrossRef]
33. Xu, J.; Fan, X.; Jian, H.; Xu, C.; Bei, W.; Ge, Q.; Zhao, T. YoloOW: A spatial scale adaptive real-time object detection neural network for open water search and rescue from UAV aerial imagery. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5623115. [CrossRef]
34. 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 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: New York, NY, USA, 2024; pp. 16965–16974.
35. Liu, S.; Li, F.; Zhang, H.; Yang, X.; Qi, X.; Su, H.; Zhu, J.; Zhang, L. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022.
36. Xu, S.; Zhou, Y.; Huang, Y.; Han, T. YOLOv4-Tiny-based coal gangue image recognition and FPGA implementation. *Micromachines* **2022**, *13*, 1983. [CrossRef] [PubMed]
37. Zhao, R.; Niu, X.; Wu, Y.; Luk, W.; Liu, Q. Optimizing CNN-based object detection algorithms on embedded FPGA platforms. In *Proceedings of the International Symposium on Applied Reconfigurable Computing, Delft, The Netherlands, 3–7 April 2017*; Springer: Cham, Switzerland, 2017.
38. He, W.; Yang, Y.; Mei, S.; Hu, J.; Xu, W.; Hao, S. Configurable 2D–3D CNNs accelerator for FPGA-based hyperspectral imagery classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 9406–9421. [CrossRef]
39. Huang, H.; Liu, Z.; Chen, T.; Hu, X.; Zhang, Q.; Xiong, X. Design space exploration for YOLO neural network accelerator. *Electronics* **2020**, *9*, 1921. [CrossRef]
40. Han, Y.; Wang, H.; Renjin, N.; Song, J.; Cui, G.; Wang, Y.; Zhou, F. Multi-ship detection and classification with feature enhancement and lightweight fusion. *Sci. Rep.* **2025**, *15*, 38075. [CrossRef] [PubMed]
41. Gao, Z.; Yu, X.; Rong, X.; Wang, W. Improved YOLOv8n for Lightweight Ship Detection. *J. Mar. Sci. Eng.* **2024**, *12*, 1774. [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.