Received: 2 January 2025 | Accepted: 23 April 2025

DOI: 10.1111/mice.13501

RESEARCH ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

An effective ship detection approach combining lightweight networks with supervised simulation-to-reality domain adaptation

Ruixuan Liao[1] | **Yiming Zhang**[1] | **Hao Wang**[1] | **Linjun Lu**[2] | **Zhengyi Chen**[3] | **Xiaoyou Wang**[4] | **Wenqiang Zuo**[5]

[1] Key Laboratory of Concrete & Prestressed Concrete Structures of Ministry of Education, Southeast University, Nanjing, China

[2] Department of Engineering, University of Cambridge, Cambridge, UK

[3] Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

[4] Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong, China

[5] Jiangsu Key Laboratory of Construction Materials, School of Materials Science and Engineering, Southeast University, Nanjing, China

Correspondence

Yiming Zhang and Hao Wang, Key Laboratory of Concrete & Prestressed Concrete Structures of Ministry of Education, Southeast University, Nanjing, China.
Email: yiming.zhang@seu.edu.cn; wanghao1980@seu.edu.cn

Xiaoyou Wang, Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong, China.
Email: cexiaoyou.wang@polyu.edu.hk

Funding information

National Natural Science Foundation of China, Grant/Award Number: 52338011; The Start-up Research Fund of Southeast University, Grant/Award Number: RF1028624058; The Southeast University Interdisciplinary Research Program for Young Scholars

Abstract

Computer vision-based ship detection using extensively labeled images is crucial for visual maritime surveillance. However, such data collection is labor-intensive and time-demanding, which hinders the practical application of newly built ship inspection systems. Additionally, well-trained detectors are usually deployed on resource-constrained edge devices, highlighting the lowered complexity of deep neural networks. This study proposes a simulation-to-reality (Sim2Real) domain adaptation framework that alleviates the annotation burden and improves ship detection efficiency by a lightweight adaptive detector. Specifically, a proxy virtual environment is established to generate synthetic images. An automated annotation method is introduced for data labeling, creating a large-scale synthetic ship detection dataset termed SSDShips. The dataset comprises 4800 images, 23,317 annotated instances, six ship categories, and various scenarios. A novel multi-level fusion lightweight (MFL) network is developed based on the you only look once version 8 (YOLOv8) framework, referred to as MFL-YOLOv8. MFL-YOLOv8 is pre-trained on the SSDShips and fine-tuned using both realistic and pseudo-realistic data through a hybrid transfer learning strategy to minimize cross-domain discrepancies. The results show that MFL-YOLOv8 reduces model parameters by 20.5% and giga floating-point operations per second by 66.0%, while improving detection performance, compared to the vanilla YOLOv8. Sim2Real adaptation boosts the model generalization in practical situations, reaching mean average precision mAP@0.5 and mAP@0.5:0.95 scores of

98.8% and 81.8%, respectively. It also shrinks the size of real-world labeling by 66.4%, achieving superior detection effectiveness and efficiency, compared to existing ship detection methods within the specific domain. Deployed on the NVIDIA Jetson Orin Nano, the proposed method demonstrates reliable performance in edge-oriented ship detection. The SSDShips dataset is available at https://github.com/congliaoxueCV/SSDShips.

1 INTRODUCTION

Ships play an irreplaceable role in global trade, leading to the continual increase in both the diversity and number of ships in waterways (Pezeshki, Adeli, et al., 2023; M. Zhang et al., 2024). Ship-related incidents that threaten the security of maritime traffic have become increasingly common, including unauthorized ship intrusions, ship-to-ship collisions, and collisions between ships and waterborne structures (Javadinasab Hormozabad et al., 2021; T. Liu et al., 2024). These accidents often result in extensive structural damage, environmental pollution, injuries, fatalities, and serious economic losses, highlighting the critical need for effective monitoring of ship locations and navigation statuses (R. W. Liu et al., 2021; Pezeshki, Pavlou, et al., 2023). Modern ship perception systems utilize a range of sensors to gather ship-related data, incorporating technologies such as automatic identification systems (AIS), synthetic aperture radar, and surveillance camera systems, among others (Gundogdu et al., 2017; M. Zhang et al., 2024). Vision-based approaches have gained considerable attention, largely owing to their affordability and ability to deliver comprehensive information (Pan et al., 2023).

Traditional ship inspection in visible videos largely depends on manual operation by maritime staff, requiring continuous screen observation, which is subjective and inefficient, potentially leading to false detections and alarms (Gundogdu et al., 2017). Recently, object detection techniques based on deep learning have been extensively adopted for water traffic monitoring, owing to their strong ability to extract features autonomously through learning (Adeli & Hung, 1994; S. Wang et al., 2024). Once a ship is detected, its bounding box coordinates can be extracted and mapped to real-world spatial information using camera calibration and georeferencing techniques (B. Zhang et al., 2019). By further incorporating multi-frame tracking methods, such as ByteTrack (Pan et al., 2023), the ship's position, speed, and movement trajectory can be inferred, seamlessly integrating with maritime monitoring systems (R. W. Liu et al., 2021). Numerous efforts have been made to improve target detection performance by advancing deep learning algorithms through techniques such as data augmentation, multi-scale learning, contextual feature learning, and generative adversarial learning (Adeli, 2001; Rafiei & Adeli, 2017). However, the following two key obstacles should be resolved to achieve effective ship detection: (1) Most methods based on deep learning typically depend on extensive annotated datasets to facilitate supervised learning. Whereas, collecting such datasets is both labor-intensive and time-consuming. This limitation results in a cold-start problem for newly developed detection systems, where insufficient labeled real-world data can significantly hinder their initial performance (Adeli & Karim, 2000; Rafiei et al., 2024); (2) Inevitable background interference around ships, including variations in lighting, weather conditions, and occlusions, can adversely impact ship detection accuracy. Although deeper and wider network architectures have enhanced detection accuracy, they notably increase computational requirements and memory usage. As a result, deploying complex models with numerous parameters on resource-constrained edge devices, such as surveillance cameras, poses significant challenges for real-time observation (Pan et al., 2023; H. Wang et al., 2024).

This study proposes a simulation-to-reality (Sim2Real) domain adaptation framework to overcome the abovementioned challenges and improve ship detection reliability by a lightweight adaptive detector. A three-dimensional (3D) proxy virtual environment is constructed to generate a large-scale synthetic image dataset. The developed multi-level fusion lightweight (MFL) model is then utilized for cross-domain ship detection, successfully connecting the virtual and real worlds. The following is a summary of the contributions and innovation:

1. A large-scale synthetic ship detection dataset (termed SSDShips) is created to address the cold-start problem in the early stages of developing ship inspection systems. For alleviating the labeling cost, an automatic virtual image annotation technique is introduced. The dataset consists of 4800 images, with 23,317 annotated instances across six ship categories, and encompasses a range of challenging scenarios, including small targets, dense targets, low-light conditions, foggy weather, and their various combinations.

2. A novel MFL network is developed based on the you only look once version 8 (YOLOv8) framework, referred to as MFL-YOLOv8. It is specifically designed to balance parameter efficiency with detection accuracy. MFL-YOLOv8 integrates the optimized EfficientNetV2, lightweight bi-directional feature pyramid networks (BiFPN) embedded with group-shuffle convolutions (GSConv), large selective kernel (LSK) attention mechanisms, and a dehazing module.
3. A Sim2Real domain adaptation framework is proposed to boost the transferability of synthetic images to real-world applications by leveraging a hybrid transfer strategy that transfers both the model parameters and its gradient history. The cycle-consistent generative adversarial network (Cycle-GAN) operates for translating the synthesized images to pseudo-realistic ones, which are integrated into the hybrid transfer process to mitigate cross-domain disparities. The adapted model is deployed on the NVIDIA Jetson Orin Nano for edge inference.

This paper is structured in the following manner: Section 2 provides an overview of previous research on synthetic ship datasets, Sim2Real adaptation, and object detection algorithms. Section 3 details the proposed methodology for domain adaptive ship detection. Section 4 presents the validation design and results, followed by conclusions and future work in Section 5.

2 | RELATED WORKS

This paper primarily focuses on synthetic ship datasets, Sim2Real domain adaptation, and deep neural networks for ship detection; therefore, the related work in these three areas is summarized.

2.1 | Synthetic ship datasets

Deep learning-based approaches have recently been gained widespread adoption in ship perception systems for maritime surveillance, typically relying on well-annotated datasets with sufficient samples (Kaur et al., 2022; Zheng & Zhang, 2020). In specific maritime waterways, the tasks of ship classification, image acquisition, and annotation are long-lasting and labor-intensive (M. Zhang et al., 2024). Consequently, these systems frequently face a scarcity of labeled ship images during initial deployment, referred to as the cold-start problem, which impacts the effectiveness of ship detection (Huang et al., 2024).

The generation of proxy virtual datasets through computer graphic engines is a safe and cost-efficient alternative for supporting object detection (Gaidon et al., 2016; Huang et al., 2024). These synthetic scenes offer complete control over the data generation process, resulting in reduced costs, increased flexibility, and an infinite range and volume of data (Raza et al., 2022). Several synthetic ship image datasets have been developed and listed in Table 1. These datasets primarily consist of aerial or onboard ship images captured from a single perspective, lacking the diversity of environmental factors, such as illumination, occlusions, and fog. These limitations could undermine their effectiveness in practical applications (Bavelos et al., 2025). To mitigate such impact, this paper introduces SSDShips, a new large-scale synthetic dataset designed for ship detection with a diverse range of background variations around ships. It includes 23,317 meticulously annotated ships across 4800 imageries with a 1920 × 1080 resolution. SSDShips encompasses six ship types: passenger ships (PS), fishing boats (FB), general cargo ships (GCS), bulk cargo carriers (BCC), ore carriers (OC), and container ships (CS). The dataset covers a broad spectrum of challenging scenarios, including small targets, densely packed targets, low illumination conditions, foggy weather, and combinations of these factors.

2.2 | Sim2Real domain adaptation

Virtual worlds and synthetic datasets could be generated using 3D simulators, providing an unlimited supply of annotated images for training detectors (Gaidon et al., 2016). Therefore, it is beneficial to utilize the existing annotated data from the virtual domain to tackle similar tasks in the real-world domain. The transfer learning process is called Sim2Real domain adaptation (Kim et al., 2024), which addresses the challenge of domain shift caused by dataset bias in the virtual domain. Domain randomization (Tobin et al., 2017) is a cost-effective method for reducing the reality gap in virtual datasets by randomizing simulator parameters such as lighting, pose, and object textures. While this approach is suitable for simpler tasks, it struggles to completely eliminate the domain gap between simulated and real datasets (Oza et al., 2023). To enhance the generalization of detectors trained on synthetic datasets, Vazquez et al. (2013) presented the first work demonstrating Sim2Real domain adaptation for developing an object detector, with a focus on cross-domain pedestrian detection. Their work utilized a reasonable number of annotations from virtual worlds and some ones from the real world too. Following this, researchers began devoting efforts to Sim2Real domain adaptation tasks, such as vehicle detection and road or railway defect detection (Gaidon et al., 2016; Huang et al., 2024).

Domain adaptation methods can be broadly categorized into unsupervised, semi-supervised, and supervised approaches based on the amount of labeled data available

TABLE 1 A comparison with other synthetic ship datasets.

Name	Simulator	Image source	Images	Size	Instances	Categories	Years
UnityShip (He et al., 2021)	Unity3D	Aerial	105,086	1600 × 2000	194,054	10	2021
SimuShips (Raza et al., 2022)	AILiveSim	On-board	9471	1920 × 1080	–	8	2022
Unreal-ship (He et al., 2022)	UE4	Aerial	2000	768 × 768	5518	–	2022
Synthetic ship detection dataset (SSDShips; proposed)	Unity3D	Multi-camera views	4800	1920 × 1080	23,317	6	2024

in the target domain (Rafiei et al., 2024). Among these, unsupervised domain adaptation can most effectively lessen the need for labeled data in the target domain and has thus been extensively investigated. These unsupervised paradigms primarily include discrepancy-based methods, adversarial-based methods, and reconstruction-based methods (Rafiei et al., 2022; Alam et al., 2020). Even though unsupervised and semi-supervised methods have made enormous strides, they tend to work unfavorably in situations where there is a major divide between the domains, like the distinction between virtual and real maritime contexts (Pereira et al., 2020). Labeling a small collection of target domain data can be a workable strategy in certain circumstances (Goodman et al., 2023). Consequently, the supervised domain adaptation setting is the main focus of this study.

Within the scope of domain-adaptive ship detection, existing studies primarily focus on cross-domain detection in remote sensing images to minimize the expensive and time-consuming steps of data annotation (Shi et al., 2022; Tang et al., 2014). However, unlike visible light images, remote sensing images are often compromised by weather conditions such as waves and clouds, and they lack detailed ship features, making real-time waterway monitoring over extended periods challenging (K. C. Wang et al., 2010; M. Zhang et al., 2024). Research on domain adaptation for ship detection using visible images remains an open question. Therefore, this study proposes a supervised domain-adaptive detection framework aimed at leveraging readily accessible synthetic datasets to enhance real-world ship detection while reducing the dependency on labeled real-world images.

2.3 | Deep learning-based ship detection

In general, there are two categories of state-of-the-art deep learning-based object detection techniques (Hu et al., 2018; Redmon et al., 2016). The first is the two-stage approach, which involves detection frames followed by classifiers, exemplified by techniques such as faster region-based convolutional neural network (R-CNN; Girshick, 2015) and Cascade R-CNN (S. Wang et al., 2024). While these approaches provide high precision, they demand significant computational power, which limits their applicability for real-time monitoring (Jiang et al., 2024). The second approach involves one-stage object detection networks, with the YOLO series (Redmon et al., 2016) and single-shot multi-box detector (W. Liu et al., 2016) being classical examples. Such algorithms predict object category probabilities and location coordinates directly, attracting significant attention for their straightforward architecture and computational efficiency (Huang et al., 2024). Among them, YOLO-based models are commonly utilized in ship detection, owing to their rapid detection speed (R. W. Liu et al., 2021).

The diversity of ship types, coupled with the intricate and ever-changing characteristics of the waterway environment, frequently leads to unstable recognition features and subsequent confusion within the network (S. Wang et al., 2024). To suppress non-informative background interference, various strategies have been widely adopted and proven effective for ship detection, including optimization of feature pyramid networks (FPNs; Y. Guo et al., 2023), integration of attention mechanisms (H. Wang et al., 2024), and implementation of optimized convolution operations (Song et al., 2023). Although detection accuracy has been upgraded by these methods, the improvements come at the expense of increased calculation requirements and major memory consumption (Kaur et al., 2022). Cloud systems, recognized for their scalability, have become a widely adopted technological solution, granting researchers and practitioners access to virtually boundless resources (Pan et al., 2023). However, as these resources are typically accessed via remote network communication, issues such as delayed response times, reduced availability, and potential security risks arise (S. Liu et al., 2021).

To overcome such drawbacks, the on-device paradigm has surfaced as a feasible substitute (Kaur et al., 2022). This approach focuses on developing object detection algorithms optimized for low power consumption and minimal computational demands, allowing models to operate directly on resource-constrained client devices (S. Liu et al., 2021). Such capabilities are particularly suited to routine maritime surveillance applications. Surveillance camera systems generate a continuous stream of high-quality videos, and real-time processing of these streams still requires substantial computational resources (Pezeshki, Pavlou, et al., 2023). Hence, questions have

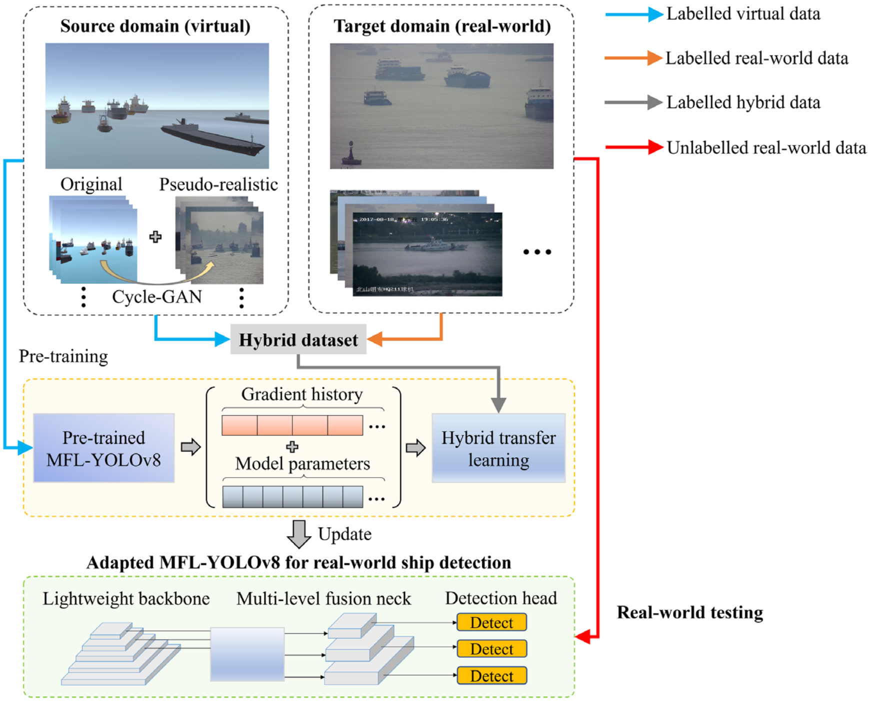


FIGURE 1 Architecture of the simulation-to-reality framework for domain adaptive ship detection. Cycle-GAN, cycle-consistent generative adversarial network; MFL-YOLOv8, multi-level fusion lightweight–you only look once version 8.

arisen regarding how to alleviate the run-time complexity of ship detection models. Lightweight object detection algorithms, such as EfficientNet (Tan et al., 2020) and GhostNet (Han et al., 2020), have received considerable interest. While these lightweight algorithms effectively lower model complexity, they may exhibit limited multi-level feature extraction capabilities in complex maritime environments, potentially affecting ship detection accuracy (D. Li et al., 2024). The limitation highlights the necessity of a ship detection network that strikes a balance between accuracy and parameter efficiency (T. Liu et al., 2024). Considering these factors, the YOLOv8s (J. Zhu et al., 2024) model serves as the baseline in this work to develop a novel MFL object detection network, termed MFL-YOLOv8, specifically tailored for ship detection in edge computing applications.

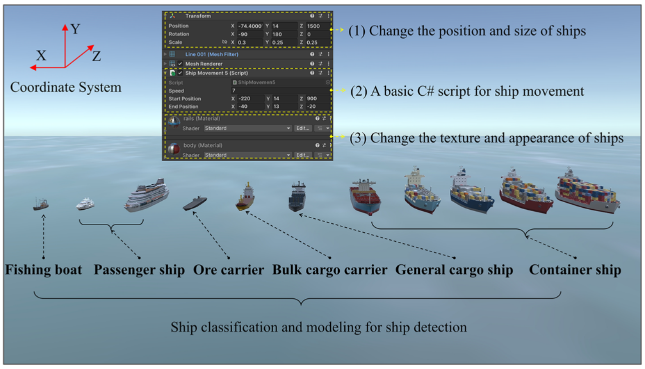
3 | METHODOLOGY

3.1 | Framework overview

The goal of this study is to improve the effectiveness of a lightweight ship detector pre-trained by a synthetic dataset while minimizing its reliance on extensive labeled real-world data through supervised domain adaptation. The Sim2Real framework architecture is shown in Figure 1. A refined 3D proxy virtual environment representing maritime navigation scenarios is established to generate a large-scale set of labeled synthetic images, creating a virtual domain (source domain). In contrast, real-world images, which are captured directly, sourced from open datasets, or obtained through web scraping, constitute the target domain. To facilitate the deployment of ship detectors on resource-constrained edge devices, MFL-YOLOv8 is proposed that integrates EfficientNetV2, lightweight BiFPN, LSK attention mechanisms, and a dehazing module. This architecture is designed to reduce the network's memory footprint while preserving robust detection performance. The MFL-YOLOv8 model is initially pre-trained using the generated dataset. The original synthesized images are translated into pseudo-realistic ones by the Cycle-GAN model to lower domain disparities. Subsequently, a hybrid dataset is constructed by combining real-world and reconstructed synthesized imageries to fine-tune the model weights using a hybrid transfer learning strategy. The transfer process involves both the model parameters and the gradient history, thereby enhancing generalization performance during model fine-tuning. Eventually, the adapted MFL-YOLOv8 is adopted to detect ships in realistic settings. The proposed approach is explored in detail in the following sections.

3.2 | Synthetic image generation

The important variations in shape and appearance across different ship types present a challenge for accurately detecting ships by category (Zheng & Zhang, 2020). Therefore, it is necessary to assign fine-grained labels to the ships. The method proposed in this paper primarily centers on civilian ship detection. It draws on the widely used benchmark dataset SeaShips (Shao et al., 2018) and incorporates common maritime waterway conditions to classify



Transform
Position X -74.4000 Y 14 Z 1500
Rotation X -90 Y 180 Z 0
Scale X 0.3 Y 0.25 Z 0.25
Line 001 (Mesh Filter)
Mesh Renderer
Ship Movement 5 (Script)
Script ShipMovemen5
Speed 7
Start Position X -220 Y 14 Z 900
End Position X -40 Y 13 Z -20
rails (Material)
Shader Standard Edit...
body (Material)
Shader Standard Edit...

Y
X
Z
Coordinate System
(1) Change the position and size of ships
(2) A basic C# script for ship movement
(3) Change the texture and appearance of ships
Fishing boat
Passenger ship
Ore carrier
Bulk cargo carrier
General cargo ship
Container ship
Ship classification and modeling for ship detection

FIGURE 2 Ship classification and modeling.

ships. Ships are grouped into six common types: PS, FB, GCS, BCC, OC, and CS. The basic virtual environment reflecting real-world maritime conditions is developed in-house using the fully integrated professional game engine Unity3D (Bavelos et al., 2025). It serves as the proxy world for data augmentation and virtual pre-training. An overview of the ship classification and modeling is provided in Figure 2.

In Unity3D, C# scripts are fundamental components used to control the behaviour and interactions of GameObjects (Gaidon et al., 2016). In this study, each ship model is assigned a dedicated C# script to control its movement and resemble practical navigation patterns. This is achieved by leveraging Unity's scripting application programming interface, which provides methods to adjust properties such as position, rotation, scale, and velocity. Ships are allowed to move vertically along the Y-axis within the range of [0 m, 1 m] to replicate the vertical motion brought on by waves. Each ship's orientation angle is changed between [−15°, 15°] to simulate more realistic trajectories (Pezeshki, Adeli, et al., 2023). These strategies make it possible to provide rich videos with varying ship behaviors. The pseudo-code for ship motion is listed in Algorithm 1. Additionally, to expand data diversity, the appearance of ships can be modified by changes in the materials and textures of their components.

There remains a significant disparity between synthesized and authentic images, which could impact the transferability of the generated dataset to real-life applications (Oza et al., 2023). Domain randomization is a promising and cost-effective approach to bridging this reality gap for object detectors trained in simulation (Tobin et al., 2017). To address the variability in real-world data, this technique introduces randomizations to simulator parameters such as lighting, pose, and haze. For instance, in Unity3D, the lighting environment can be adjusted by modifying directional lights, point lights, environment reflections, and skyboxes (Raza et al., 2022). Fog conditions can be simulated by setting fog particle effects and adjusting fog density (Vazquez et al., 2013). These randomizations help the network to concentrate on the key elements of the images (Gaidon et al., 2016).

ALGORITHM 1 Ship navigation in Unity3D

```
Input: Start position ($S$), end position ($E$), waypoints ($w_1, w_2, \ldots, w_n$), velocity ($v$), rotation angle ($\theta$), and vertical fluctuation ($h$).
Output: Real-time ship position and orientation.
Initialize $S$, first waypoint $w_1$, and velocity $v$ randomly in [$v_{\min}$, $v_{\max}$];
Compute movement direction toward the current target;
Adjust $v$ dynamically within [$v_{\min}$, $v_{\max}$];
While the ship has not reached the end position $E$:
    Move the ship toward the target at speed $v$;
    Apply vertical fluctuation $h$ within [0 m, 1 m];
    Smoothly rotate the ship, constraining $\theta$ within [−15°, 15°];
    Record the ship's position and orientation at the current time step;
    If the ship reaches the current target:
        Snap position to the target;
        Update to the next waypoint if available; otherwise, set $E$ as the target;
    Repeat until reaching $E$.
```

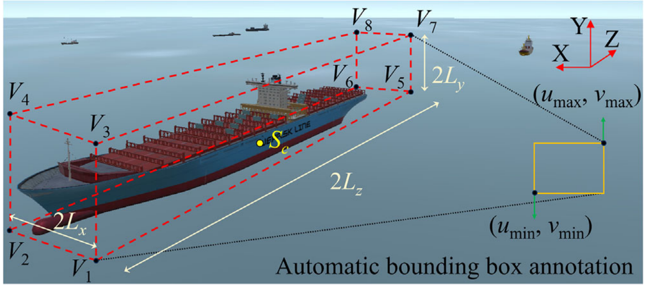


FIGURE 3 Automatic bounding box annotation procedure.

An automated annotation technique for synthetic ship images is proposed to lessen the labeling burden. In the Unity scene, the position coordinates and size parameters of the ship models are recorded. Based on these parameters, a 3D bounding box can be constructed around the ship model, centered at the model's center point. In the Unity coordinate system, let the center point of the 3D bounding box be $S_c(x_c, y_c, z_c)$ and the half-lengths of the box along each axis be L_x, L_y, and L_z as shown in Figure 3.

The eight vertices of the 3D bounding box have the following coordinates: $V_1(x_c\text{-}L_x, y_c\text{-}L_y, z_c\text{-}L_z)$,$V_1(x_c+L_x, y_c\text{-}L_y, z_c\text{-}L_z)$,$V_1(x_c\text{-}L_x, y_c+L_y, z_c\text{-}L_z)$,$V_1(x_c+L_x, y_c+L_y, z_c\text{-}L_z)$,$V_1(x_c\text{-}L_x, y_c\text{-}L_y, z_c+L_z)$,$V_1(x_c+L_x, y_c\text{-}L_y, z_c+L_z)$,$V_1(x_c\text{-}L_x, y_c+L_y, z_c+L_z)$,$V_1(x_c+L_x, y_c+L_y, z_c+L_z)$. These vertices are then transformed into 2D pixel coordinates based on the intrinsic and extrinsic matrices of the virtual camera (Pan et al., 2023). Let (X, Y, Z) represent the coordinates of each vertex of the 3D bounding box, and (u, v) denote the corresponding pixel coordinates after transformation. The conversion formula between the two is defined as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where R is the 3×3 rotation matrix, t is the 1×3 translation vector, f_x and f_y are the focal lengths of the camera in the horizontal and vertical directions, respectively, and (c_x, c_y) is the principal point, which is the point where the optical axis intersects the image plane. For each virtual camera, these parameters can be directly obtained from Unity.

Let the pixel coordinates of vertices of the 3D bounding box after transformation be denoted as (u_k, v_k), where $k = 1, 2, \ldots, 8$. The minimum and maximum coordinates in the 2D pixel coordinate system can be calculated by:

$$\begin{cases} u_{\min} = \min\left(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\right) \\ u_{\max} = \max\left(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\right) \\ v_{\min} = \min\left(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\right) \\ v_{\max} = \max\left(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\right) \end{cases} \tag{2}$$

The coordinates $(u_{\min}, v_{\min})$ and $(u_{\max}, v_{\max})$ are taken as the bottom-left and top-right corners of the annotation box, respectively. The information can be saved as a .txt file to support manual verification in annotation software, such as LabelImg (T. Liu et al., 2024). An example of the .txt file content is $[Calss, (u_{\max} + u_{\min})/2w, (v_{\max} + v_{\min})/2h, (u_{\max} - u_{\min})/w, (v_{\max} - v_{\min})/h]$, where w denotes the image width, h denotes the image height, and *class* represents the six ship types, with values ranging from 0 to 5 (0: OC, 1: BCC, 2: GCS, 3: CS, 4: FB, and 5: PS).

3.3 | Proposed lightweight network for ship detection

Edge computing is attracting growing attention in maritime surveillance systems (S. Liu et al., 2021). These systems produce a continuous stream of high-quality videos. Real-time processing of massive images on resource-restricted edge devices requires detectors to offer sufficient detection performance with low run-time complexity. However, this paradigm faces obstacles when dealing with highly complex models. To address this, this section focuses on developing lightweight backbone networks, optimizing multi-level feature fusion, and integrating dynamic attention mechanisms to achieve a balance between computational efficiency and detection performance.

3.3.1 | Overall architecture of the MFL-YOLOv8

Released in 2023, YOLOv8 was designed to integrate the strengths of previous YOLO models while incorporating new enhancements. Its key feature, scalability, offers significant advantages for further improvements in YOLO projects (J. Zhu et al., 2024). To balance processing efficiency and detection performance, YOLOv8s is utilized as the baseline (Zhou & Peng, 2023). In this work, EfficientNetV2 (Tan & Le, 2021), all-in-one dehazing networks (AOD-Net; B. Li et al., 2017), BiFPN (H. Wang et al., 2024), GSConv (J. Zhu et al., 2024), and LSK attention mechanisms (Y. Li et al., 2023) are integrated into YOLOv8s, developing an innovative MFL architecture named MFL-YOLOv8. Figure 4 depicts the proposed network's structure. The key modules are illustrated in Figure 4a–g.

Four main parts make up the construction of MFL-YOLOv8: the input, backbone, neck, and detecting head. Prior to being fed into the backbone network, image data are pre-processed and normalized at the input stage. The lightweight backbone, integrating EfficientNetV2 and AOD-Net, is specifically designed to reduce the effects of image blurriness while effectively extracting ship-related features of the input imageries. The neck employs a hybrid architecture combining cross stage partial bottleneck with two convolutions (C2f) modules and GSConv modules, enhanced by BiFPN (Xie et al., 2024). The structure improves the network's feature fusion performance while effectively reducing network parameters and improving detection speed. To improve detection performance for ships of varying scales, LSK attention mechanisms are incorporated before the detection head. The receptive field's scale is dynamically altered across the feature maps by this adaptive mechanism. The detection head is tasked with object detection and localization within the image, and it operates across three distinct detection scales (20 × 20, 40 × 40, 80 × 80). Hierarchical feature extraction ensures that smaller ships are detected at higher-resolution layers (80 × 80), while larger ships are handled by lower-resolution scales (20 × 20 and 40 × 40). Although some studies have proposed adding large-scale detecting layers (e.g., 160 × 160) to strengthen the detection performance of very small targets, such procedures notably expand the computational burden, which is detrimental to model deployment on edge devices (H. Wang et al., 2024). The selected detection scales offer a suitable balance between accuracy and processing efficiency as shown by earlier studies and experimental findings on maritime datasets (e.g., SeaShips, FAIR1M) (R. W. Liu et al., 2021; Shao et al., 2018). Ultimately, non-maximum suppression is employed to eliminate unnecessary bounding boxes, ensuring the

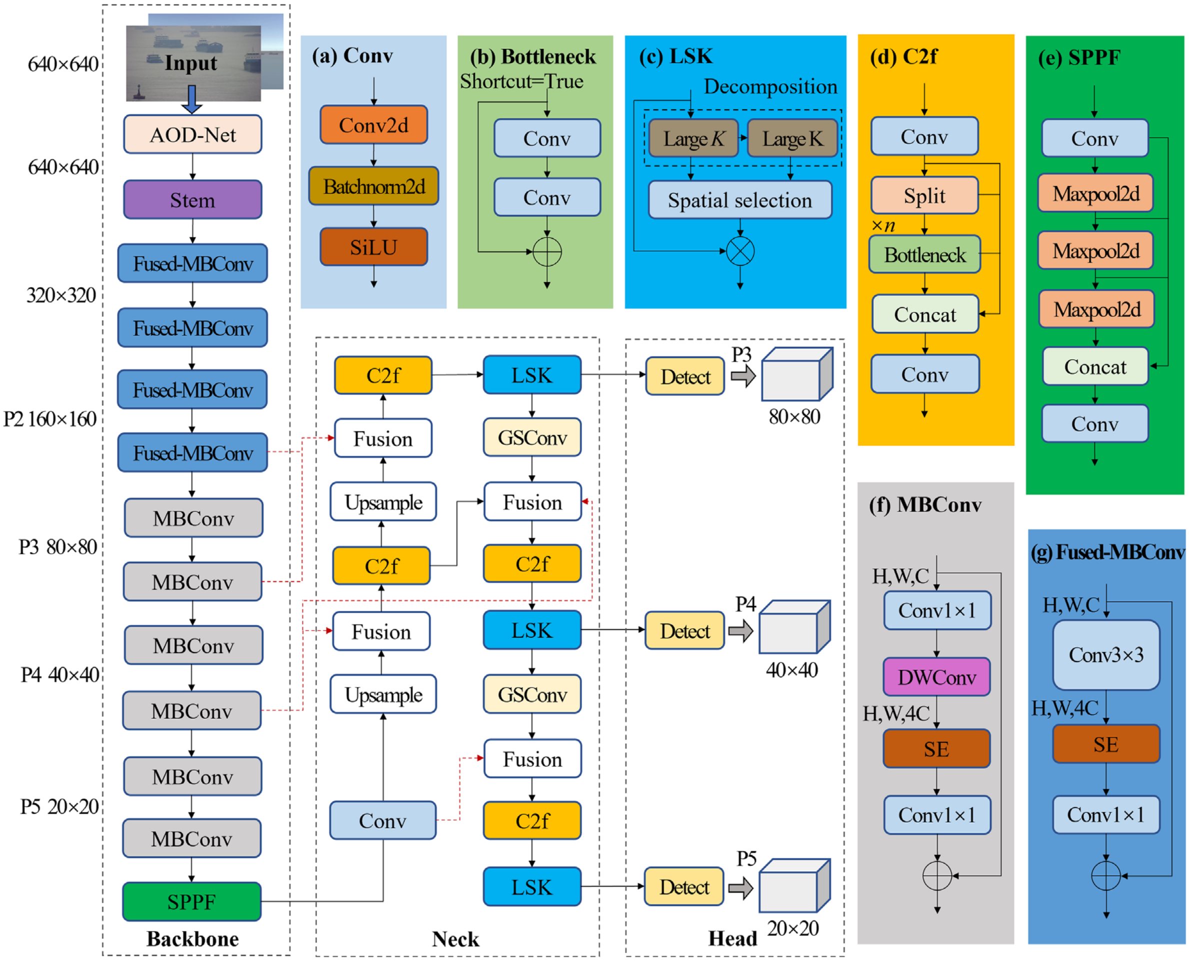


FIGURE 4 Detailed architecture of the proposed multi-level fusion lightweight–you only look once version 8 (MFL-YOLOv8). AOD-Net, all-in-one dehazing networks; C2f, cross stage partial bottleneck with two convolutions; LSK, large selective kernel, MBConv, mobile inverted bottleneck convolution; SPPF, spatial pyramid pooling - fast.

selection of the most precise prediction frames (Pan et al., 2023).

3.3.2 | Lightweight and dehazing backbone based on EfficientNetV2

EfficientNet is an efficient, lightweight feature extraction network. As an extension of EfficientNet, EfficientNetV2 offers faster training speeds and improved parameter efficiency, compared to earlier models (Tan & Le, 2021). Leveraging this network, a lightweight backbone is designed and integrated with AOD-Net to improve ship-detecting performance in intricate situations.

The EfficientNetV2 architecture achieves a balance between processing efficiency and accuracy by using the complementary strengths of the mobile inverted bottleneck convolution (MBConv) and Fused-MBConv modules (Jiang et al., 2024). The MBConv module features linear bottleneck layers with inverted residuals and depth-wise convolutions (DWConv; Tan et al., 2020). It enhances feature extraction efficiency by concentrating computational resources on DWConv, which processes each channel independently, requiring fewer operations than standard convolutions (J. Zhu et al., 2024). Fused-MBConv, a key innovation in EfficientNetV2, improves computational efficiency and accelerates training in early network layers (Jiang et al., 2024). It replaces the two-step process of 1 × 1 channel expansion and DWConv in MBConv with a single 3 × 3 convolution, eliminating redundant computations and reducing memory access overhead. This streamlined design is particularly effective in early blocks, where large input resolutions amplify computational demands (Tan et al., 2020). However, it should be noted that applying Fused-MBConv could result in a reduction in precision (Jiang et al., 2024). To mitigate such impact, Fused-MBConv modules are positioned early in the backbone, where the input images retain higher spatial resolution and are less affected by minor variations in feature extraction (J. Zhu et al., 2024). To improve the model's

TABLE 2 Parameter setting of the designed backbone.

Stage	Module	Kernel size	Stride	Channels	Layers
0	All-in-one dehazing networks (AOD-Net)	3 × 3	1	3	1
1	Stem	3 × 3	2	24	1
2	Fused-MBConv1	3 × 3	1	24	2
3	Fused-MBConv4	3 × 3	2	48	1
4	Fused-MBConv4	3 × 3	1	48	3
5	Fused-MBConv4	3 × 3	1	64	3
6	MBConv4, SE0.25	3 × 3	2	128	1
7	MBConv4, SE0.25	3 × 3	1	128	5
8	MBConv6, SE0.25	3 × 3	1	160	1
9	MBConv6, SE0.25	3 × 3	1	160	8
10	MBConv4, SE0.25	3 × 3	2	272	1
11	MBConv4, SE0.25	3 × 3	1	272	14
12	SPPF	5, 9, 13	–	1024	1

Note: The number following Fused-MBConv or MBConv indicates the expansion ratio of the first convolutional layer in the main branch of the module, and SE0.25 signifies the nodes in the initial fully connected layer of the squeeze-and-excitation (SE) blocks constitute a quarter of the channels in the input MBConv feature matrix.

Abbreviation: MBConv, mobile inverted bottleneck convolution; SPPF, spatial pyramid pooling - fast.

capacity to concentrate on important aspects, squeeze-and-excitation (SE) blocks (Tan & Le, 2021) are incorporated into both MBConv and Fused-MBConv. SE blocks recalibrate channel-wise feature responses, prioritizing important features while suppressing less relevant ones (Hu et al., 2018). As a lightweight attention mechanism, SE blocks offer an efficient balance between effectiveness and computational expense, making them well-suited for resource-limited applications. The neural architecture search approach (Tan & Le, 2021) is employed to identify the optimal combination of MBConv and Fused-MBConv modules to form an efficient backbone. The search space includes the number of layers, kernel sizes {3 × 3, 5 × 5}, and expansion ratios {1, 4, 6} (Jiang et al., 2024). Table 2 presents the specific parameters of the backbone modules. Stage 0 is set to AOD-Net, Stage 1 comprises a regular convolutional layer, Stages 2–11 are set to MBConv and Fused-MBConv operations, and the final stage is set to the Spatial Pyramid Pooling - Fast (SPPF) block.

AOD-Net is developed using a redefined atmospheric scattering model (Satyanath et al., 2023), producing dehazed images immediately through a lightweight CNN. To enhance model performance under hazy conditions, this network is seamlessly embedded at the beginning of the backbone (T. Liu et al., 2024). It integrates all the components of the atmospheric scattering equation by restructuring it as (B. Li et al., 2017)

$$J(x) = K(x)I(x) - K(x) + b \tag{3}$$

with

$$K(x) = \frac{\frac{1}{t(x)}(I(x) - A) + (A - b)}{I(x) - 1} \tag{4}$$

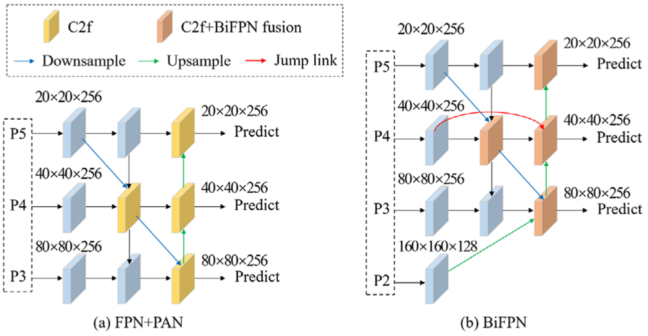
$$t(x) = e^{-\beta d(x)} \tag{5}$$

where $J(x)$ is the dehazed image, $I(x)$ is the input hazy picture, $t(x)$ is a transmission matrix, A is the global atmospheric light, β is the atmospheric scattering coefficient, $K(x)$ is an intermediate variable used to describe the transformation process of I(x), and b is constant bias, typically set to a value of 1. A CNN structure known as the K-estimation module (Satyanath et al., 2023) is used to learn the parameter K(x) from $I(x)$.

3.3.3 | Enhanced BiFPN for multi-level feature fusion

In the context of YOLOv8, the integration of FPN and path aggregation networks (PANet) facilitates multi-scale feature fusion (S. Wang et al., 2024). However, both FPN and PANet encounter several challenges, including: (1) disparities in feature data at different sizes and (2) high computational expenses or ineffective feature fusion (Y. Guo et al., 2023). To address these issues, improved BiFPN structure and GSConv are integrated into the neck part, termed BiFPN-GSConv.

As illustrated in Figure 5, the BiFPN network enhances feature fusion by treating each bidirectional path (top-down and bottom-up) as a distinct network layer. It achieves this by differentially fusing various features through weighted feature fusion, which learns the relative importance of the different input features (Xie et al., 2024). In contrast to the feature fusion strategy of PAN paired with FPN used in the original YOLOv8, BiFPN's bidirectional cross-scale links minimize single-scale feature network nodes that contribute less to feature fusion and concentrate on multi-scale feature fusion nodes. BiFPN employs residual-like skip connections between the inputs



(a) FPN+PAN

(b) BiFPN

FIGURE 5 Different approaches of feature fusion. BiFPN, bi-directional feature pyramid networks; C2f, cross stage partial bottleneck with two convolutions.

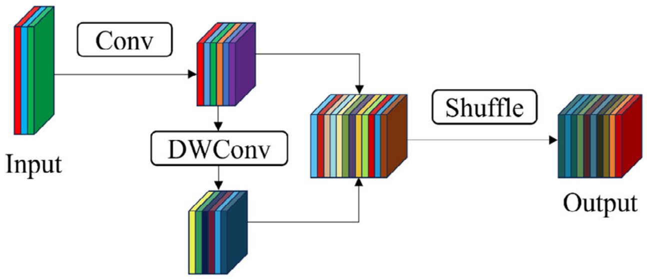


FIGURE 6 The structure of group-shuffle convolutions. DWConv, depth-wise convolutions.

and outputs of the same layer to facilitate multi-scale feature fusion (A. Guo et al., 2024). Unlike the standard BiFPN, the feature extraction of the P2 stage is incorporated in this study to diversify the extraction scale. That is, the developed BiFPN performs multi-scale feature fusion on the corresponding output feature maps of the P2–P5 layers. As a result, more features can be fused without significantly raising the cost, and each bidirectional path can be reused to accomplish higher level feature fusion by treating it as a feature network layer. The fast normalized fusion approach is employed to assign additional weights to each input (Xie et al., 2024). For instance, considering layer i, the formulas for the fused features in BiFPN are given by:

$$O = \sum_i \frac{w_i}{\sum_j w_j + \varepsilon} \cdot I_i \tag{6}$$

$$P_i^{td} = Conv\left(\frac{w_1 \cdot P_i^{\text{in}} + w_2 \cdot P_{i+1}^{\text{in}}}{w_1 + w_2 + \varepsilon}\right) \tag{7}$$

$$P_i^{\text{out}} = Conv\left(\frac{w_1 \cdot P_i^{\text{in}} + w_2 \cdot P_i^{td} + w_3 \cdot P_{i-1}^{\text{out}}}{w_1 + w_2 + w_3 + \varepsilon}\right) \tag{8}$$

where O is the feature map after fusion, I_i is the input feature map being fused, w_i denotes the weight assigned to the ith input I_i, ε represents a small constant to ensure numerical stability and prevent division by zero, P_i^{td} and P_i^{out} represent the intermediate transition feature of the ith layer on the top-down pathway and the final output feature of the ith layer on the bottom-up pathway, P_i^{in} is the input feature at the current layer i, P_{i+1}^{in} denotes the input feature at the next layer $i + 1$ (higher-resolution), and $P_{i\text{-}1}^{\text{out}}$ represents the output feature from the previous layer $i - 1$ (bottom-up pathway). In Equation (7), w_1 and w_2 are the weighting parameters for the inputs of the current layer and the next layer, which are necessary to compute the intermediate transition features. In Equation (8), w_1, w_2, and w_3 represent the weights of the inputs of the current layer, the outputs of the transition units within the current layer, and the outputs of the previous layer, respectively.

The weights w_i are initialized with a small random strategy and then passed through a Rectified Linear Unit (ReLU) activation function to ensure $w_i \geq 0$. This maintains numerical stability and prevents negative weights that could destabilize training. A feature fusion approach based on BiFPN structure with channel tuning is employed. This approach not only ensures the effective fusion of multi-scale features but also minimizes the feature network nodes of a single scale, which contribute less to feature fusion. The 80 × 80, 40 × 40, and 20 × 20 feature maps produced by channel tuning BiFPN correlate to P3, P4, and P5 values. The deeper layer's feature map has a higher chance of representing ship features on a larger scale. The shallower layer's feature map is more likely to depict ship features at a smaller scale (A. Guo et al., 2024). A certain level of model light-weighting is achieved by adjusting the number of feature fusion channels to 256, which reduces redundant channels while maintaining useful feature information in comparison to the original BiFPN.

To further reduce the calculating resources required, standard convolution operations are replaced by GSConv modules. As shown in Figure 6, GSConv employs uniform mixing operations that allow the seamless integration of information from standard convolutions with that from shuffle operations (J. Zhu et al., 2024). This design lessens computational overhead and time loss. However, it may also limit the exchange of information between feature maps, potentially impacting model performance (Han et al., 2020). To balance these trade-offs, GSConv is applied only in the neck of the network, replacing ordinary convolutions before the fast normalized fusion operations, forming the BiFPN-GSConv architecture. This placement minimizes redundant information and eliminates the need for further compression.

3.3.4 | LSK-based attentional feature fusion

Ship detection has traditionally focused on enhancing the representation of oriented bounding boxes, often

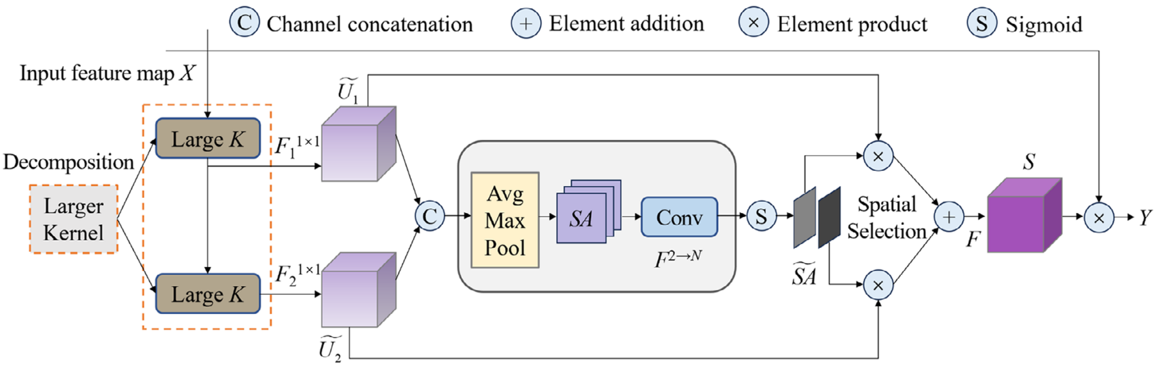


FIGURE 7 The flow chart of the large selective kernel module.

overlooking the unique prior knowledge related to ship characteristics in images (T. Liu et al., 2024). This problem is especially noticeable when detecting small-sized ships at long camera-to-object distances (Song et al., 2023). Because long-distance targets take up fewer pixels in photos, texture details are less noticeable, and models have a harder time extracting enough characteristics for precise classification. Besides, small targets are vulnerable to interference from background noise due to the low contrast between them and complex backdrops, which can result in erroneous detections (R. W. Liu et al., 2021). Further contributing to detection errors is the substantial scale fluctuations of long-distance targets in photos, which might make model flexibility difficult. In YOLOv8, the feature fusion process may exacerbate these issues by introducing redundant information and failing to capture the specific contextual requirements of different target types (S. Wang et al., 2024).

To address such issues, the LSK attention mechanism is introduced during the final stage of the feature integration. This mechanism adaptively modifies the receptive field by selecting kernel sizes that are suited to the spatial needs of each target (Y. Li et al., 2023). Through adaptive weighting and spatial fusion of features processed by kernels of DWConv, the LSK mechanism allows the model to emphasize the most pertinent background information. This significantly improves detection accuracy, particularly for small ships, which would otherwise lack sufficient context for correct identification (S. Wang et al., 2024). Figure 7 shows how the LSK module is structured.

The input feature map X is processed through 1 × 1 deep convolution, where features from different sensory fields are initially extracted. The processed features are subsequently concatenated and subjected to max- and average-pooling. From the combined features, a following convolutional layer creates N spatial attention maps, and each spatial attention map is subjected to a sigmoid activation function. The results are element-wise multiplied with X to produce the output feature Y. The procedure can be expressed as

$$Y = X \cdot F$$
$$\times \left(\sum_{i=1}^{N} \left(S \left(F^{2-N} \left(\left[P_{\mathrm{avg}} \left(U_i \right) ; P_{\max} \left(U_i \right) \right] \right) \right) \cdot F_i^{1\times1} \left(U_i \right) \right) \tag{9}$$

where $F(\cdot)$ is the convolution operation, N is the quantity of decomposition kernels, U_i is the feature derived from the ith layer of X, S is the sigmoid activation function, P_{avg} is the average-pooling, and $P_{\max}$ denotes the max-pooling.

3.4 | Supervised virtual and real-world adaptation

Most current research on domain adaptation emphasizes the unsupervised setting, where no labels are available from the target domain. For cases with important domain differences, such as virtual and real maritime contexts, these unsupervised settings may lead to suboptimal performance (Goodman et al., 2023) Therefore, the focus is on the supervised domain adaptation paradigm. In the realm of cross-domain ship detection, a completely labeled source domain (virtual domain) $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N}$ is available, comprising N images, where x_i^s and y_i^s denote the synthetic images and the corresponding ship annotations, respectively. Similarly, a target domain (real-world domain) $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{M}$ is given, consisting of M images, where x_i^t and y_i^t represent the real-world images and the corresponding ship annotations (Vazquez et al., 2013). The goal of the proposed approach is to reduce the size of real-world data labeling by transferring knowledge from a fully labeled virtual domain to a partially labeled real-world domain.

Conventional supervised adaptation methods include freezing or regularizing certain network layers, shifting model weights, and then fine-tuning using only labeled target domain data (Rafiei et al., 2022). However, such approaches could fail to leverage additional information from the source domain, such as gradient history, which in turn increases the risk of catastrophic forgetting of previously learned representations from the source domain (Goodman et al., 2023; Rafiei & Adeli, 2018; Rafiei et al., 2024). Transfer learning incorporating partial gradient history has been shown to improve fine-tuning effectiveness (Finn et al., 2017). Unlike traditional supervised adaptation methods that directly fine-tune model parameters using only target data, this study develops a hybrid transfer learning approach. This approach transfers both model parameters and gradient history to enhance smoother adaptation using hybrid datasets that include both source and target data.

Let $\theta \in \Omega$ represent the collection of network parameters trained on the source domain $\mathcal{D}_s$. The network's state within the parameter space is described by the parameter set θ and its gradient $L^{'}(\theta)$. The gradient that is utilized to optimize the loss function is written as

$$L'_{source}(\theta) = \frac{1}{n}\sum_{i=1}^{n_s}\frac{d}{d\theta}\left[\lambda_{bbox}l_{bbox}\left(x_i^s, y_i^s, \theta\right) + \lambda_{cls}l_{cls}\left(x_i^s, y_i^s, \theta\right)\right] \tag{10}$$

$$L'_{target}(\theta) = \frac{1}{n}\sum_{i=n_s+1}^{n}\frac{d}{d\theta}\left[\lambda_{bbox}l_{bbox}\left(x_i^t, y_i^t, \theta\right) + \lambda_{cls}l_{cls}\left(x_i^t, y_i^t, \theta\right)\right] \tag{11}$$

$$L'(\theta) = \frac{n_s}{n}L'_{source}(\theta) + \left(1 - \frac{n_s}{n}\right)L'_{target}(\theta) \tag{12}$$

where l_{bbox} represents the bounding box regression loss, l_{cls} denotes the classification loss, λ_{bbox} and λ_{cls} are hyperparameters that regulate the contribution of each loss term, n is the batch's entire amount of data points, and n_s represents the amount of data points from the source domain in the batch.

The target domain's instances are progressively added to the optimization procedure, while gradients continue to be computed on source samples. This method ensures a smoother transition than standard fine-tuning, which typically involves an abrupt shift from source to target domain. As a result, additional information is effectively conveyed from the pre-training step to the fine-tuning phase. The pseudo-code for the hybrid transfer learning process is summarized in Algorithm 2. The gradient calculated in batches is an approximation, and using this approximated gradient to update weights can be a noisy procedure because it may point in a different direction than the actual gradient. To mitigate this effect, an exponential moving average (EMA) of gradients from previous batches is continuously updated (Rafiei et al., 2024), defined as

$$\overline{L'}(\theta)^{(i)} = \alpha\overline{L'}(\theta)^{(i-1)} + (1 - \alpha)L'(\theta) \tag{13}$$

where $\overline{L'}(\theta)^{(i)}$ is the smoothed gradient at iteration i, $\overline{L'}(\theta)^{(i-1)}$ is the previous smoothed gradient, and α is the exponential decay factor, empirically set to 0.9 to balance stability and responsiveness (Rafiei et al., 2022).

ALGORITHM 2 Hybrid transfer learning process

```
Input: source domain $\mathcal{D}_s$, target domain $\mathcal{D}_t$, num_of_steps.
Output: adapted model parameters $\theta^{(i)}$.
Initialize: $\theta^{(0)} \leftarrow \theta_{pre-trained}$, $\bar{g}^{(0)}_{source} \leftarrow 0$, $\bar{g}^{(0)}_{target} \leftarrow 0$, decay factor $\alpha$
for $i$ in 1 to num_of_steps do
    Compute number of target and source samples:
    $n\_target \leftarrow int(\frac{current\_epoch}{num\_epochs} \times batch\_size)$
    $n\_source \leftarrow batch\_size—n\_target$;
    Sample batches:
    $source\_batch \leftarrow sample\,(source, n\_source)$
    $target\ \_batch \leftarrow sample\,(target, n\_\ target)$;
    Compute instantaneous gradients:
    $g^{(i)}_{source} \leftarrow \sum_{x \in source\_batch} \nabla L'(\theta^{(i-1)}, \mathcal{D}^{(i)}_s)$
    $g^{(i)}_{target} \leftarrow \sum_{x \in target\_batch} \nabla L'(\theta^{(i-1)}, \mathcal{D}^{(i)}_t)$
    Update EMA of gradients:
    $\bar{g}^{(i)}_{source} \leftarrow \alpha\bar{g}^{(i-1)}_{source} + (1 - \alpha)g^{(i)}_{source}$
    $\bar{g}^{(i)}_{target} \leftarrow \alpha\bar{g}^{(i-1)}_{target} + (1 - \alpha)g^{(i)}_{target}$
    $\bar{g}^{(i)}_{combined} \leftarrow \frac{n_s}{n}\bar{g}^{(i)}_{source} + (1 - \frac{n_s}{n})\bar{g}^{(i)}_{target}$
    Update model parameters:
    $\theta^{(i)} \leftarrow$ optimiser.step($\theta^{(i-1)}, \bar{g}_{combined}$);
End for
```

This exponentially weighted average assigns greater importance to recent gradients, while older gradients decay at a rate determined by the factor α^k, where k denotes the number of steps prior to the current iteration (Finn et al., 2017). This approach facilitates stable and adaptive optimization over time.

While the aforementioned technique can optimize the fine-tuning process, the domain discrepancy remains and may impact the transfer performance. To minimize the image-level discrepancy before the hybrid transfer, Cycle-GAN (J. Y. Zhu et al., 2017), a widely used style transfer technique in reconstruction-based domain adaptation, is employed to reconstruct annotated pseudo-realistic data. Cycle-GAN takes two separate discriminators and two coupled generators to create a ring network. The objective function is given by:

$$L_{\text{Cycle-GAN}} = L_{\text{GAN}}\left(G^{\mathcal{D}_t}_{\mathcal{D}_s}, C_t, \mathcal{D}_s, \mathcal{D}_t\right) + L_{\text{GAN}}$$

$$\times\left(G_{\mathcal{D}_t}^{\mathcal{D}_s}, C_s, \mathcal{D}_s, \mathcal{D}_t\right) + \lambda L_{cyc}\left(G_{\mathcal{D}_s}^{\mathcal{D}_t}, G_{\mathcal{D}_t}^{\mathcal{D}_s}\right) \tag{14}$$

where $G_{\mathcal{D}_s}^{\mathcal{D}_t}$ and $G_{\mathcal{D}_t}^{\mathcal{D}_s}$ define mapping functions that transform $\mathcal{D}_s$ to $\mathcal{D}_t$ and $\mathcal{D}_t$ to $\mathcal{D}_s$, respectively, C_s and C_t denote discriminators of the two distinct domains, respectively, L_{GAN} and L_{cyc} denote adversarial loss and cycle consistency loss, respectively, and λ is a cycle-consistency weight for L_{GAN} and L_{cyc}.

Only the trained translator that converts source images to target images is used to reconstruct images. The source and pseudo-realistic imageries have the same annotations since the translator converts between domains without changing the content information (J. Y. Zhu et al., 2017). These reconstructed images are used to supplement hybrid datasets. Experiments will compare two approaches for constructing hybrid datasets: one using original synthesized images combined with real-world images and the other utilizing Cycle-GAN-generated pseudo-realistic images with realistic images. This comparison will validate the efficacy of Cycle-GAN in domain adaptation.

4 | CASE STUDY

4.1 | Synthetic dataset

As described in Section 3.2, domain randomization techniques are adopted to broaden the variability of the source domain and help bridge the reality gap. The ship-related parameters, including type, quantity, size, and speed, as well as environmental factors such as lighting and atmospheric conditions, are all randomized within specified ranges, as detailed in Table 3. These variations remain straightforward to implement.

A synthesized image dataset called SSDShips is produced in the Unity3D environment by adjusting the variables listed in Table 3 and employing the introduced automatic annotation approach. All labels have been manually verified in LabelImg software to ensure annotation accuracy (Guan et al., 2023). This dataset consists of 4800 images and 23,317 targets representing six ship types. The dataset distribution is shown in Figure 8. Figure 9 presents some example images illustrating these variations. It covers a variety of complex scenarios, such as small targets, multiple targets, low-light conditions, foggy weather, and combinations of these factors.

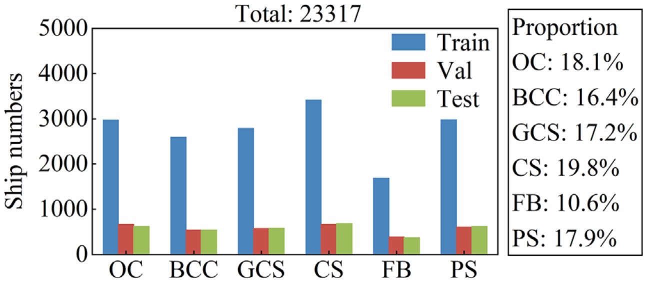


FIGURE 8 The dataset distribution. BCC, bulk cargo carriers; CS, container ships; FB, fishing boats; GCS, general cargo ships; OC, ore carriers; PS, passenger ships.

The largest percentage of ships are CS, which are prime targets for maritime traffic surveillance due to their huge sizes and capacities to transport a wide variety of vital goods (M. Zhang et al., 2024). On the other hand, the proportion of FB is lower because they are much smaller and have less of an impact on maritime safety. The remaining ship types are distributed fairly evenly, with quantities falling between CS and FB. Of the total dataset, 3360 are used for training, 720 for validation, and the remaining 720 for testing.

4.2 | Virtual pre-training for ship detection

4.2.1 | Implementation details and evaluation metrics

The image's height and width are set to 640 × 640 pixels during preprocessing. The weights learnt from the Microsoft Common Objects in COntext (COCO) dataset are used to initialize MFL-YOLOv8 (Xie et al., 2024). All models are trained for 300 epochs, with a batch size of 16. Mosaic data augmentation is executed throughout the training stage, yet it is turned off in the final 10 epochs to expedite model convergence (Y. Guo et al., 2023). With a weight decay of 0.0005, a momentum of 0.937, and an initial learning rate of 0.001, the adaptive moment estimation (Adam) optimizer is employed in this study. The system processor is the central processing unit (CPU) 12th Gen Intel(R) Core(TM) i7-12700F, with both training and testing carried out on an NVIDIA GeForce RTX 3060 graphics processing unit (GPU).

This study uses a number of metrics, such as average precision (AP), mean AP (mAP), giga floating-point operations per second (GFLOPs), parameters (Params), and Frames Per Second (FPS), to assess the object identification model (Girshick, 2015). The detection accuracy improves with increasing AP and mAP values. In particular, mAP@0.5 is the mAP computed at an intersection over union threshold of 0.5, and mAP@0.5:0.95 is the mAP computed from 0.5 to 0.95, with a 0.05 step increment (T. Liu et al., 2024). Params is used to quantify the model's parameters (Tan & Le, 2021), GFLOPs quantify the model's complexity (A. Guo et al., 2024), and FPS is used to evalu-

TABLE 3 Factors in the generation of synthesized ship images.

Factor name	Factor description	References
Ship types	The dataset consists of six types of ships: ore carriers (OC), bulk cargo carriers (BCC), general cargo ships (GCS), container ships (CS), fishing boats (FB), and passenger ship (PS). Each image could contain between 0 and 6 ship categories	Shao et al. (2018)
Ship numbers	The number of ships in the scene ranges from 0 to 20, encompassing a variety of scenarios from no objects to sparse and dense distributions	Gundogdu et al. (2017)
Ship sizes	FB: *L* 8–20 m, *W* 2–4 m, *H* 2–6 m; PS: *L* 20–200 m, *W* 10–30 m, *H* 5–40 m; OC: *L* 50–120 m, *W* 15–25 m, *H* 3–10 m; GCS: *L* 80–140 m, *W* 15–30 m, *H* 10–20 m; BCC: *L* 100–160 m, *W* 15–35 m, *H* 10–25 m; CS: *L* 150–250 m, *W* 25–50 m, *H* 20–45 m. Here, *L*, *W*, and *H* denote the length, width, and height of the ships, respectively	Gundogdu et al. (2017), Pezeshki, Adeli, et al. (2023)
Ship positions	In the Unity scene, the movement range of ships along both the X- and Z-axes is set to [0 m, 1000 m]	Gaidon et al. (2016)
Route length	The route length of each ship is within the range of [100 m, 1800 m]	Shao et al. (2018)
Ship speed	FB: 2–7 m/s; PS: 5–15 m/s; OC: 4–10 m/s; GCS: 5–12 m/s; BCC: 4–10 m/s; CS: 6–14 m/s	Chen et al. (2021)
Ship textures	Alterations to the ship textures, such as hull color, wood texture, metal finish, roughness map, and specular map, are applied	Huang et al. (2024), Raza et al. (2022)
Background	The water color, flow velocity, and skyboxes are systematically modified to enhance the diversity of the scenes	M. Zhang et al. (2024)
Atmospheric conditions	Different levels of fog are simulated in the Unity scenes using the exponential mode, with the fog density interval set to [0, 0.0018]. This range covers conditions from light mist to dense fog	Gundogdu et al. (2017), T. Liu et al. (2024)
Illumination	Directional lights with intensity interval [0, 1], point lights with intensity interval [0, 10], and skyboxes with light intensity interval [0, 1] in Unity3D are combined to modify scene illumination, covering a wide range of possible lighting variations. All lighting modes are set to the Shadowmask mode	He et al. (2021, 2022)
Scale changes	Ship detection tasks typically require the identification of ship targets at multiple scales. Consequently, virtual images must include both large targets (more than 700 × 700 pixels) and small targets (less than 50 × 50 pixels)	Guan et al. (2023; Shao et al. (2018)
Viewpoints	Each category of ship is captured from multiple perspectives to obtain different angles of the target, including the frontal view, side view, and intermediate angles between the frontal and side views	Zheng and Zhang (2020)
Visible proportions	The dataset includes images with different visible proportions of ships. A label is assigned even when the ship is only partially visible in the image, allowing for the training of a more robust model	M. Zhang et al. (2024)
Occlusion	The dataset is constructed to realistically represent physical contexts, encompassing scenarios where ships may occlude one another in the background.	Zheng and Zhang (2020)

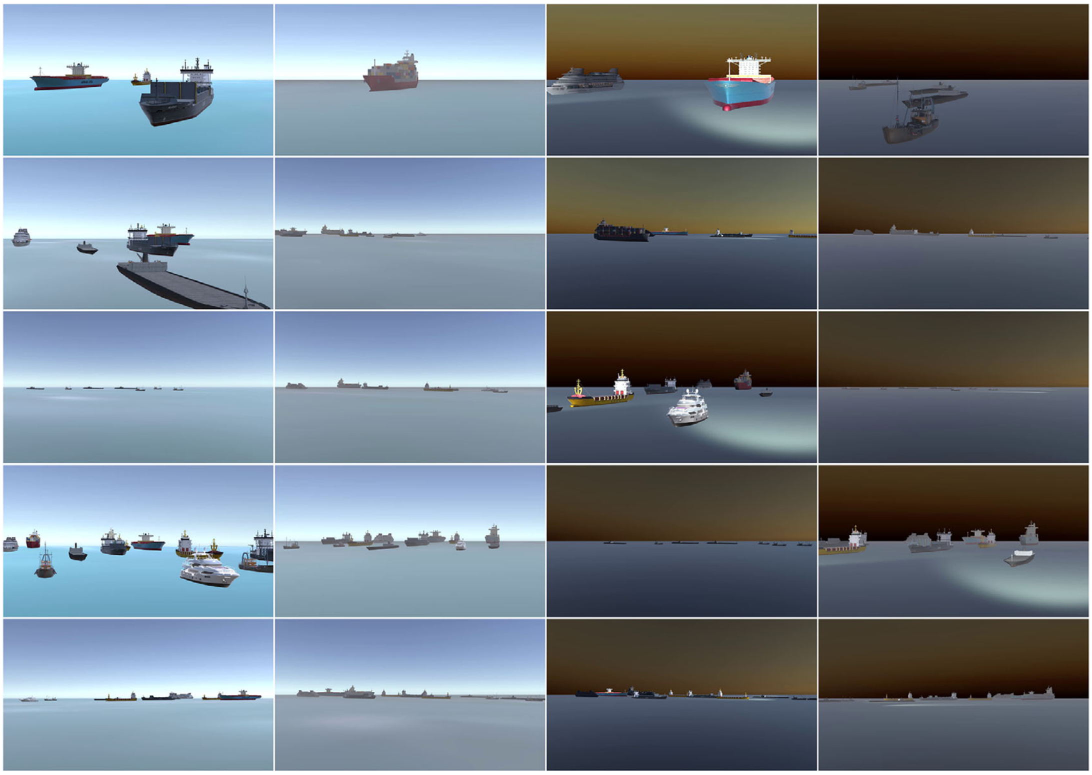

(a) Normal illumination (b) Foggy images under normal illumination (c) Low illumination (d) Foggy images under low illumination

FIGURE 9 Examples of six different ship types in a variety of situations.

ate the detecting speed of the model (Shao et al., 2018). The computation methods for these metrics are given as

$$\mathrm{AP} = \int_0^1 P(R)dR \tag{15}$$

$$\mathrm{mAP} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{AP}_k \tag{16}$$

$$\mathrm{Params} = O\left(\sum_{i=1}^{n} M_i^2 K_i^2 C_{i-1} C_i\right) \tag{17}$$

$$\mathrm{GFLOPs} = O\left(\sum_{i=1}^{n} K_i^2 C_{i-1}^2 C_i + \sum_{i=1}^{n} m^2 C_i\right) \tag{18}$$

$$FPS = \frac{1}{t_{img}} \tag{19}$$

where P is the percentage of accurately detected objects compared to all objects detected, R is the proportion of identified objects to those really present, AP_k is the AP value for category k, N is the number of ship categories, O is the constant order, n is the total number of network layers, K_i is the number of convolutional kernels in the ith layer, $C_{i\text{-}1}$ is the number of input channels for the ith layer, C_i is the number of output channels for the ith layer, M_i represents the coefficient of the weight matrix for the ith layer, m is the coefficient for fully connected layers or other operations, and t_{img} is the time required for processing an image.

4.2.2 | Ablation experiment on overall architecture

This study, which measures the disparity between the real and synthetic worlds in the context of ship detection, complements the commonly adopted method of applying synthetic data to train models for real-world applications. Therefore, experiments are conducted to assess the efficacy of SSDShips in training the MFL-YOLOv8 model. A number of ablation studies are conducted to determine the efforts of different parts in the suggested MFL-YOLOv8 model. Table 4 displays the outcomes of these tests on the SSDShips dataset, where ✓ and × denote the inclusion and exclusion of specific modules in the methods under comparison.

As observed from Table 4, using EfficientNetV2 as the backbone network results in a 25.0% reduction in parameters and a 71.5% decrease in GFLOPs for the YOLOv8s model. However, this is coincided with a drop of around 21.5% in FPS as well as a 1.4% and 2.3% decline in mAP@0.5 and mAP@0.5:0.95, respectively. Without replacing the YOLOv8 backbone with EfficientNetV2, incorporating

TABLE 4 Findings from ablation tests.

Baseline	EfficientNetV2	AOD-Net	BiFPN-GSConv	LSK	Mean average precision (mAP)@0.5 (%)	mAP@0.5: 0.95 (%)	Params (M)	Giga floating-point operations per second (GFLOPs)	Frames Per Second (FPS)
✓	×	×	×	×	99.0	82.6	11.2	28.8	158
✓	✓	×	×	×	97.7	80.4	8.4	8.2	124
✓	×	✓	×	×	99.1	82.5	11.6	30.2	142
✓	×	×	✓	×	99.0	82.7	11.5	29.7	148
✓	×	×	×	✓	99.1	82.8	12.0	30.8	150
✓	✓	✓	×	×	98.3	81.2	8.4	9.6	120
✓	✓	×	✓	×	98.2	81.5	8.4	8.2	146
✓	✓	×	×	✓	98.4	81.3	8.6	8.5	127
✓	×	✓	✓	×	99.1	82.6	11.9	31.1	141
✓	×	✓	×	✓	98.9	82.4	12.2	32.3	139
✓	×	×	✓	✓	99.2	82.7	12.7	30.9	147
✓	✓	✓	✓	×	98.6	81.9	8.4	9.6	133
✓	✓	×	✓	✓	98.8	82.3	8.6	8.5	144
✓	✓	✓	×	✓	98.7	81.6	8.7	9.9	123
✓	×	✓	✓	✓	99.2	82.8	13.3	32.4	145
✓	✓	✓	✓	✓	99.2	82.9	8.9	9.8	142

Abbreviations: BiFPN, bi-directional feature pyramid networks; GSConv, group-shuffle convolutions; LSK, large selective kernel.

LSK, AOD-Net, BiFPN-GSConv, and their combinations can improve detection accuracy to a certain extent. Nevertheless, these improvements come at the cost of increased computational complexity and a higher number of parameters. When EfficientNetV2 is integrated with these modules, the experimental results show varying degrees of efficacy enhancement. The findings indicate that the proposed MFL-YOLOv8 model achieves superior detection performance, with a slight improvement in precision, compared to the vanilla YOLOv8s, while cutting the number of GFLOPs and parameters by 66.0% and 20.5%, respectively. This substantially lowers the model's computational complexity. Additionally, the detection speed attained by the model is 142 FPS, thus far exceeding the requirements for real-time detection. It has been noted that integrating AOD-Net improves model accuracy while maintaining a low parameter count. This displays the effectiveness of placing AOD-Net at the backbone's head. Overall, MFL-YOLOv8 effectively balances detection accuracy, speed, and parameter efficiency.

Figure 10 illustrates the detection outcomes of both the proposed MFL-YOLOv8 and the vanilla YOLOv8s model for six different types of multi-scale ships, evaluated across a range of scenarios. To more effectively assess the detection performance on ships of various scales, three categories are defined: large-scale (area ≥ 0.075), small-scale (area < 0.025), and mid-scale (area ≥ 0.025 and < 0.075; Y. Guo et al., 2023). The area represents the proportion of the bounding box dimensions relative to the overall image size. For large-scale images, whether under normal or low illumination or in foggy conditions, ships can be accurately detected by both YOLOv8s and MFL-YOLOv8, with no significant detection difference, as seen in scenes (b), (g), and (q). For mid-scale images, both YOLOv8s and MFL-YOLOv8 can detect ships, with MFL-YOLOv8 showing slightly higher accuracy, as illustrated in scenes (f) and (l). In scenes (a), (c), (d), (e), (h)–(j), (k), (m)–(o), (p), (r), and (t), for small-scale images, YOLOv8s exhibit false detections or missed detections. In contrast, regardless of whether the targets are sparse or numerous, and under varying levels of illumination and fog, MFL-YOLOv8 not only detects all targets in the image but also provides more accurate bounding boxes than YOLOv8s. It can be concluded that MFL-YOLOv8 maintains its lightweight design while still demonstrating high robustness in ship detection under complex scenarios.

4.2.3 | Effect of attention mechanisms

This section investigates the impact of different attention mechanisms on the performance of MFL-YOLOv8 model. Several experiments are conducted by replacing the LSK module in the MFL-YOLOv8 model with other widely used attention modules, including SE, convolutional block attention module (CBAM; A. Guo et al., 2024), and Vision Transformer with bi-level routing (BiFormer; L. Zhu et al., 2023) attention mechanisms. Table 5 presents

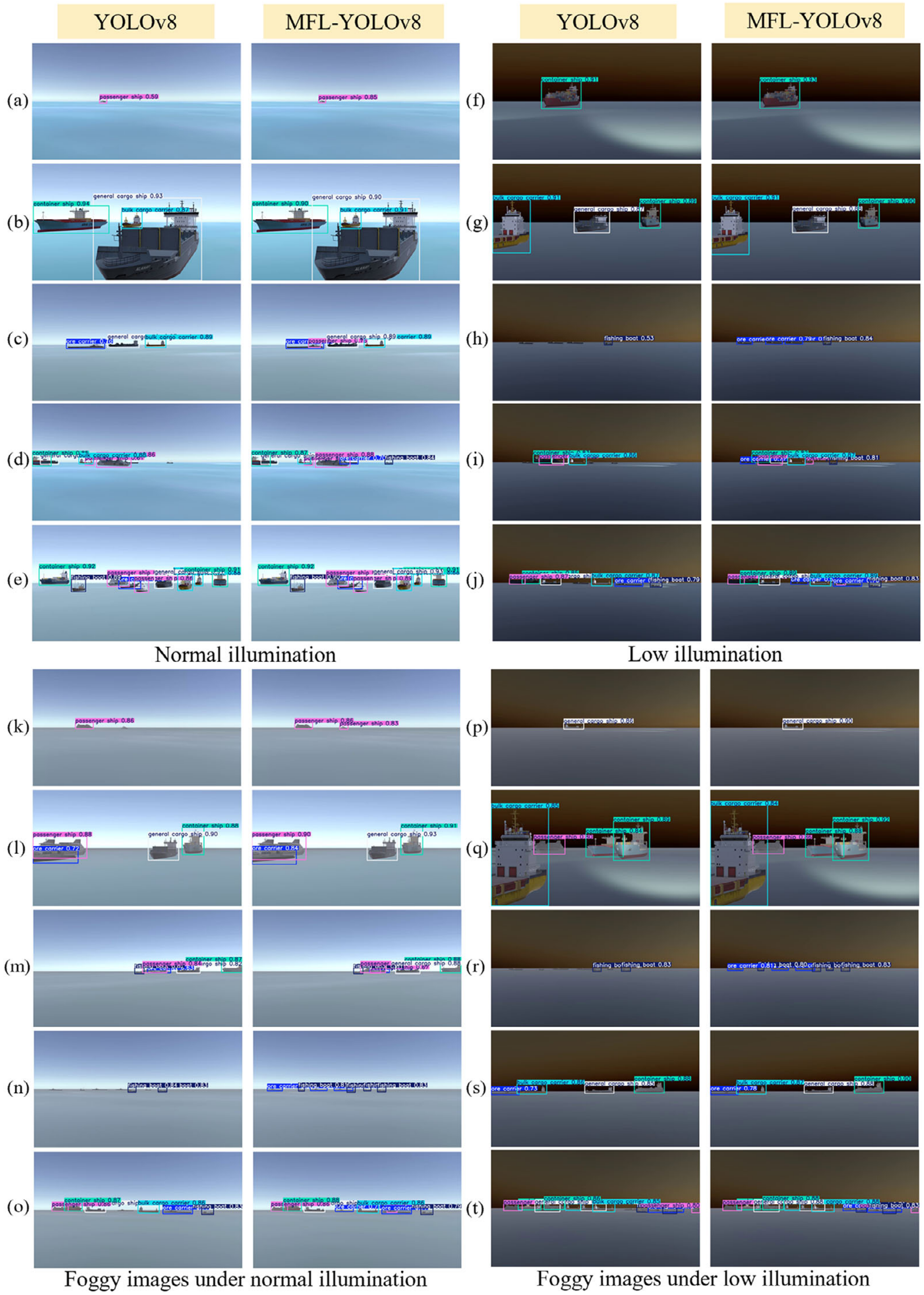


FIGURE 10 Detection results of multi-scale ships across various scenarios.

the experimental results using different modules. The network's ability to recognize small targets is improved by the LSK block, which dynamically selects convolutional layers with varied receptive field sizes for various objects. This mechanism offers the highest accuracy and detection speed among other methods. Furthermore, the addition of the LSK mechanism does not notably increase the run-time complexity of the model, being comparable to adding SE or CBAM. When compared to embedding the BiFormer, the number of parameters and GFLOPs is decreased by 13.6% and 57.3%, respectively.

4.2.4 | Impact of the defogging module

A detailed comparative analysis of detection performance under varying fog conditions before and after defogging is conducted to demonstrate the effectiveness of the

TABLE 5 Effect of different attention mechanisms.

Attention	mAP @0.5 (%)	mAP@0.5: 0.95 (%)	Params (M)	GFLOPs	FPS
SE	98.6	81.8	8.6	9.6	136
CBAM	98.2	81.9	8.8	9.8	128
BiFormer	98.9	82.7	10.3	23.2	82
Ours	99.2	82.9	8.9	9.8	142

Abbreviations: BiFormer, Vision Transformer with bi-level routing; CBAM, convolutional block attention module.

TABLE 6 Performance before and after defogging.

	Performance under different fog density interval				
Model	Metrics	0	[0, 0.0005]	[0.0005, 0.0012]	[0.0012, 0.0018]
You only look once version 8 (YOLOv8)	mAP@0.5 (%)	99.2	99.0	98.5	97.4
	Average FPS	158			
YOLOv8+ AOD-Net	mAP@0.5 (%)	99.0	99.2	99.2	98.7
	Average FPS	150			
Multi-level fusion lightweight (MFL)-YOLOv8-non AOD-Net	mAP@0.5 (%)	99.3	99.2	98.9	98.3
	Average FPS	145			
MFL-YOLOv8	mAP@0.5 (%)	99.2	99.4	99.3	99.0
	Average FPS	142			

TABLE 7 The quantity of pictures for each ship type.

Ship type	Instances	Proportion (%)
OC	2426	19.2
BCC	1997	15.8
GCS	2275	18.0
CS	2553	20.2
FB	1428	11.3
PS	1958	15.5
Total	12,637	100

defogging module. Based on the fog density intervals defined in Unity (as summarized in Table 3), the fog conditions are classified into four levels: clear (no fog), light fog [0,0.0005], moderate fog (0.0005,0.0012], and heavy fog (0.0012,0.0018]. For each fog level, 150 images are randomly selected from the test set of the SSDShips to construct four evaluation subsets. Four models are trained on the training set of the SSDShips: (1) vanilla YOLOv8, (2) YOLOv8+AOD-Net (with the AOD-Net module integrated into the backbone of YOLOv8s), (3) MFL-YOLOv8-non AOD-Net (i.e., a variant of the MFL-YOLOv8 architecture with only the AOD-Net module removed), and (4) the proposed MFL-YOLOv8. All models are tested on each fog-specific subset to compare their detection performance across different fog levels. The comparison results are listed in Table 6.

The detection results demonstrate that integrating AOD-Net consistently improves detection accuracy under foggy conditions, particularly in moderate and heavy fog scenarios. The proposed MFL-YOLOv8 outperforms all other configurations across varying fog levels, achieving up to a 1.6% increase in mAP@0.5, compared to the vanilla YOLOv8. On clear images, the inclusion of AOD-Net led to a marginal degradation in accuracy, with mAP@0.5 reduced by no more than 0.2%, which is considered an acceptable level of overprocessing (T. Liu et al., 2024). Although the integration of AOD-Net results in a slight reduction in inference speed, the FPS remains competitive, indicating a favorable trade-off between computational overhead and performance gains.

4.3 | Adaptation of the virtual ship detector to realistic domains

4.3.1 | Real-world data description

To facilitate the generalization of the MFL-YOLOv8 model pre-trained by synthesized data across diverse practical scenarios, a real-world dataset comprising 9332 images and 12,637 instances is constructed by collecting images from four distinct sources. The number of instances for every ship category in the real-world dataset is shown in Table 7.

1. Images sourced from the SeaShips dataset (7000 images; Shao et al., 2018): The SeaShips dataset is a popular benchmark for ship detection. These images cover a broad range of real-world maritime scenes.
2. Images sourced from additional open-source datasets (1160 images; Gundogdu et al., 2017; Zheng & Zhang, 2020): The SeaShips dataset lacks sufficient representation of complex scenarios, including dense targets, small-sized targets, low visibility, foggy environments, and combinations of these scenes. Images from other open-source datasets are incorporated to expand the coverage of such challenging situations.
3. Web-scraped images (772 images; M. Zhang et al., 2024): These images often exhibit higher contrast and more diverse background colors than typical images, enriching the diversity of real-world data.

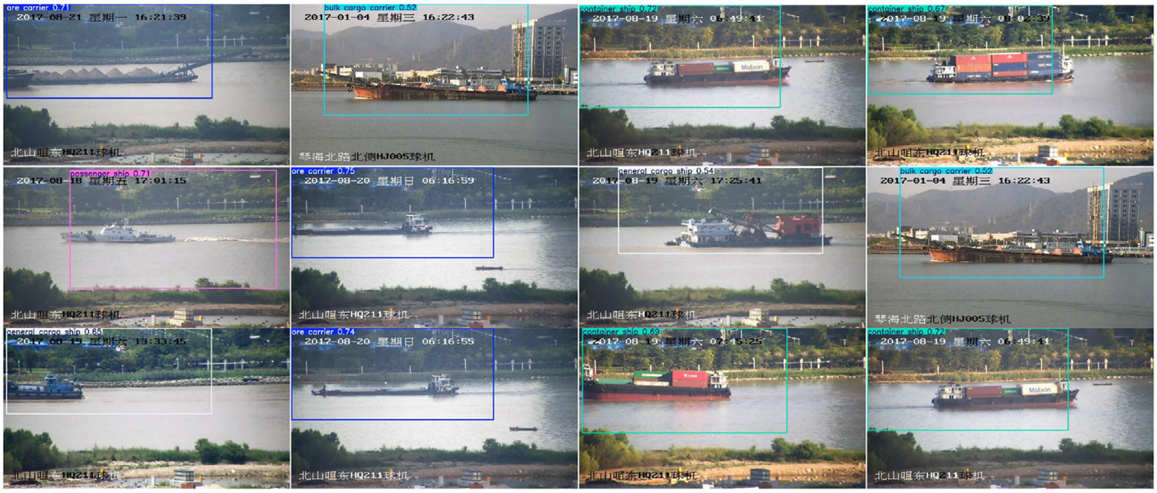


FIGURE 11 Real-world detection results from the virtual pre-trained model.

4. Self-captured images (400 images): Waterway ship images captured using cameras and mobile phones. These images are exclusively used for testing to better demonstrate the practical effectiveness of the proposed cross-domain detection approach.

The MFL-YOLOv8 model pre-trained using the SSD-Ships dataset is adopted to detect real-world ships, with several detection examples presented in Figure 11. The pre-trained model demonstrates the capability to detect ships, achieving detection accuracy of up to 75% for certain ship types, thus validating the effectiveness of the SSDShips dataset. Nevertheless, discrepancies between the predicted bounding boxes and the actual ships are found in the images remain. These errors could be addressed in the following section using a domain adaptation approach.

4.3.2 | Hybrid transfer learning

To further reduce cross-domain disparities, image translation based on Cycle-GAN is implemented. Since low-resolution images are the focus of the classic Cycle-GAN model, source images are cropped to 256 × 256 pixels during the training phase (Kim et al., 2024). The output images retain the same resolution as the input. Training is conducted for 200 epochs with a batch size of 1, using the Adam optimizer and a learning rate of 0.0002. A cycle-consistency weight of $\lambda = 10$ is applied. The overall training procedure follows the strategies reported by J. Y. Zhu et al. (2017). The image transition results can be found in Figure 12. The first column displays real-world images, the second column features original virtual images, and the third column represents the virtual images reconstructed by the Cycle-GAN model.

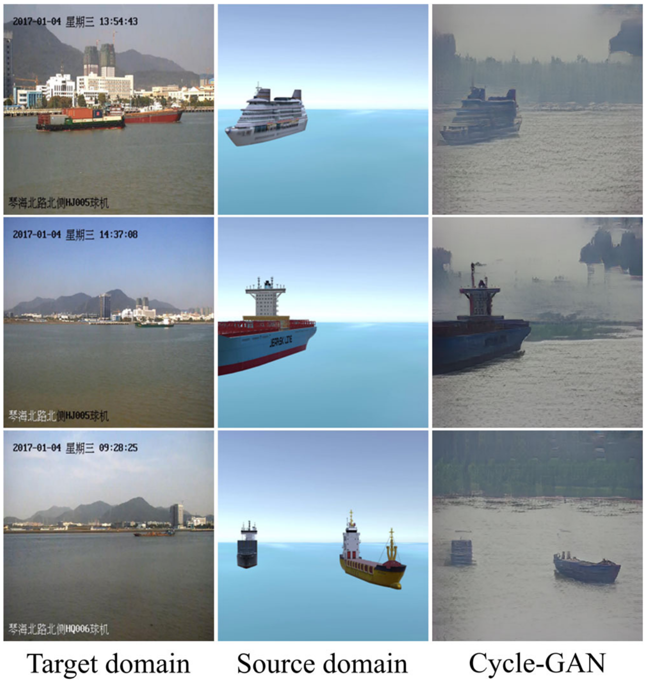


FIGURE 12 Cycle-consistent generative adversarial network (Cycle-GAN) model-based image transition.

Two commonly used metrics, Fréchet inception distance (FID) and structural similarity index (SSIM; Heusel et al., 2017), are adopted to evaluate the performance of image-to-image translation. A lower FID means the translated images are more similar to those in the target domain, while a higher SSIM (ranging from 0 to 1) reflects stronger structural consistency and better image fidelity. In this case, the FID score is 28.6, and the SSIM is 0.78 ± 0.05, with the latter calculated as the mean and standard deviation over 2240 virtual-to-real image pairs. These results suggest that the quality of the generated pseudo-realistic images is acceptable for Sim2Real transfer (Kim et al., 2024; J. Y. Zhu et al., 2017).

Hybrid datasets for supervised Sim2Real transfer learning are created by mixing original synthesized images with real pictures and pseudo-realistic images with true images, respectively, as explained in Section 3.4. To mitigate the risk of overfitting during fine-tuning, for the combination of original virtual images with true images, only the 720 designated test images from the SSDShips dataset are included in the hybrid transfer learning stage. Many comparative experiments are carried out to assess the suggested cross-domain detection method's effectiveness in strengthening real-world ship detection efficiency and reducing the size of real-world data labeling. The study evaluates multiple training strategies, including: (1) training solely on real-world data without any domain adaptation, (2) standard supervised adaptation using real data, (3) unsupervised adaptation using either original synthetic images or Cycle-GAN-generated pseudo-realistic images, and (4) the proposed hybrid transfer strategy, which combines real data separately with either original synthetic images or pseudo-realistic images. The implementation details are consistent with those outlined in Section 4.3.1. The configurations of hybrid datasets and performance outcomes are provided in Table 8.

Regardless of whether it is MFL-YOLOv8 or YOLOv8, pre-training on synthesized data followed by standard supervised adaptation with real-world data consistently outperforms training exclusively on real-world data (non-domain adaptation) in terms of detection accuracy. This suggests that the synthesized dataset advances the performance of real-world ship detection. The main reason for the upgrade is the simulated images produced by refined modeling successfully increase the dataset for sharing some characteristics with real-world data. Additionally, simulated image variations (e.g., illumination and fog) may help the network learn appropriate features. It is worth noting that, in both non-domain adaptation and supervised adaptation scenarios, MFL-YOLOv8 consistently exhibits better detection capability than YOLOv8, further confirming the benefit of the network improvements in this study. Therefore, subsequent comparative analyses will focus exclusively on MFL-YOLOv8.

When fine-tuning the model using only original synthetic images, the model exhibits poor generalization performance. Replacing these images with pseudo-realistic images generated by Cycle-GAN leads to a noticeable improvement. Nonetheless, the overall accuracy remains inadequate. This suggests that the standalone Cycle-GAN cannot fully bridge the domain gap between virtual and real maritime scenes, highlighting the limitations of relying solely on general unsupervised adaptation in this context. The proposed hybrid transfer learning strategy could effectively alleviate this restriction.

When hybrid transfer learning is applied with 720 initial synthetic images and 5500 real-world images, the mAP@0.5 and mAP@0.5:0.95 of the fine-tuned model perform better than those of the model fine-tuned with 6532 real-world images alone. This result demonstrates that the hybrid transfer learning strategy effectively reduces the dependence on labeled real-world data for supervised domain adaptation and elevates model performance. Fine-tuning the model with 720 reconstructed images and 4000 real images yielded a higher level of accuracy. This demonstrates that Cycle-GAN adequately boosts the transfer from virtual to real domains and further reduces the requirement for real-world labeling. Increasing the number of reconstructed and true images to 1652 and 6532, respectively, substantially promotes the model's detection performance, with mAP@0.5:0.95 of 81.8% and mAP@0.5 of 98.8%. Nevertheless, model performance is not further improved by increasing the number of reconstructed images. This could be explained by a bottleneck in Cycle-GAN's capacity to completely reduce cross-domain discrepancies.

Figure 13 compares visual detection results on real-world data with and without the use of supervised domain adaptation techniques to clarify the benefits of the suggested approach. Across the four types of real-world data, particularly in complex scenarios involving small targets, low visibility, and dense fog, the non-adaptive model exhibits false detections, omissions, and duplicate detections, as observed in scenes (c), (i), (j), (l), (n), (o), and (t). Even for larger ships, detection accuracy remains suboptimal due to wave interference as shown in scenes (d) and (h). In contrast, the proposed method mitigates these issues to a certain extent, consistently delivering more complete and accurate bounding boxes in both general and challenging conditions. It also exhibits better detection performance for relatively low-frequency ship types, that is, FB, even in the presence of occlusion or low contrast, as illustrated in scenes (b) and (c). This indicates that the supervised Sim2Real domain adaptation effectively boosts the model generalization. Still, in overly complex circumstances, such as extreme environmental variations and ultra-long-range detection of small ships, Sim2Real transfer may exhibit misdetections (see scenes (i) and (t)). Despite this, the proposed method consistently demonstrates greater robustness than non-adaptation methods. These errors are considered acceptable within the scope of the study, as its primary focus is maritime surveillance on computationally resource-limited platforms (M. Zhang et al., 2024).

TABLE 8 Performance results for Sim2Real adaptation training.

		Data type and amount									
Method	**Model**	**Train**	**Validate**	**Metrics**	**All**	**OC**	**BCC**	**GCS**	**CS**	**FB**	**PS**
Non-domain adaptation	YOLOv8-N	**R** 6532	**R** 1399	mAP@0.5 (%)	95.5	95.3	95.5	95.5	98.2	96.6	91.9
				mAP@0.5:0.95 (%)	76.7	73.3	77.8	78.2	83.2	75.4	72.1
	MFL-YOLOv8-N	**R** 6532	**R** 1399	mAP@0.5 (%)	95.9	96.0	95.9	95.7	98.5	97.1	92.1
				mAP@0.5:0.95 (%)	77.2	74.4	78.4	78.5	83.8	75.6	72.7
Standard supervised adaptation	YOLOv8-P	**R** 6532	**R** 1399	mAP@0.5 (%)	96.1	96.5	96.3	96.0	98.5	96.7	92.8
				mAP@0.5:0.95 (%)	77.9	76.7	77.5	79.8	84.2	75.6	73.9
	MFL-YOLOv8-P	**R** 6532	**R** 1399	mAP@0.5 (%)	96.7	97.2	97.1	96.3	98.6	96.9	94.1
				mAP@0.5:0.95 (%)	79.8	80.1	78.3	82.7	86.4	75.8	75.4
Unsupervised adaptation	MFL-YOLOv8-P	**S$_o$** 720	**R** 3266	mAP@0.5 (%)	3.5	0.8	3.9	4.2	7.2	1.9	2.9
	MFL-YOLOv8-P	**S$_f$** 720	**R** 3266	mAP@0.5 (%)	10.6	7.2	12.5	7.6	17.2	8.4	10.8
	MFL-YOLOv8-P	**S$_f$** 1652	**R** 3266	mAP@0.5 (%)	24.2	17.3	27.2	26.6	29.8	15.2	28.9
	MFL-YOLOv8-P	**S$_f$** 2240	**R** 3266	mAP@0.5 (%)	23.8	16.7	27.4	25.9	28.2	15.6	29.0
Hybrid transfer learning	MFL-YOLOv8-P	**S$_o$** 720 + R 5500	**R** 1916	mAP@0.5 (%)	97.3	97.7	97.3	97.3	99.2	97.7	94.7
				mAP@0.5:0.95 (%)	80.7	80.5	79.0	83.6	86.2	77.8	77.4
	MFL-YOLOv8-P	**S$_f$** 720 + R 4000	**R** 2666	mAP@0.5 (%)	98.1	98.3	97.6	98.5	98.9	97.3	98.2
				mAP@0.5:0.95 (%)	80.9	77.2	82.0	84.9	86.6	75.1	79.5
	MFL-YOLOv8-P	**S$_o$** 720 + R 6532	**R** 1399	mAP@0.5 (%)	98.3	98.1	97.8	98.9	99.2	97.7	98.3
				mAP@0.5:0.95 (%)	81.5	77.2	82.7	86.1	87.4	75.5	80.3
	MFL-YOLOv8-P	**S$_f$** 1652 + R 5500	**R** 1916	mAP@0.5 (%)	98.6	98.5	98.1	99.2	99.3	98.1	98.5
				mAP@0.5:0.95 (%)	81.7	77.6	82.8	86.0	87.7	75.7	80.3
	MFL-YOLOv8-P	**S$_f$** 1652 + R 6532	**R** 1916	mAP@0.5 (%)	98.8	98.7	98.5	99.3	99.3	98.3	98.7
				mAP@0.5:0.95 (%)	81.8	77.7	83.1	86.2	87.6	76.0	80.4
	MFL-YOLOv8-P	**S$_f$** 2240 + R 6532	**R** 1399	mAP@0.5 (%)	98.7	98.7	98.3	99.2	99.3	98.1	98.7
				mAP@0.5:0.95 (%)	81.6	77.8	82.7	86.2	87.2	75.6	80.2

Note: R denotes real-world images, S$_o$ represents original images from SSDShips, and S$_f$ is pseudo-realistic images transformed by Cycle-GAN. The numbers following R, S$_o$, and S$_f$ indicate the respective quantities of images. MFL-YOLOv8-P and YOLOv8-P refer to models pre-trained using SSDShips, whereas MFL-YOLOv8-N and YOLOv8-N denote models without pre-training, that is, the non-domain adapted models.
Abbreviations: Cycle-GAN, cycle-consistent generative adversarial network; Sim2Real, simulation-to-reality.

4.3.3 | Comparison of detection methods

The proposed domain-adaptive ship detection framework is benchmarked against several existing methods (Chen et al., 2021; R. W. Liu et al., 2021; Zhou & Peng, 2023; T. Liu et al., 2024) using the SeaShips dataset, to provide a more comprehensive evaluation of its efficacy. All comparative methods are evaluated under the same hardware conditions. In this study, images sourced from the SeaShips dataset (7000 images) are mixed with 1652 Cycle-GAN-generated pseudo-realistic images to create hybrid datasets. Based on the quantity of incorporated real-world data, three different data mixing strategies are established, with the training and validation set distributions as follows: (1) S$_f$ 1652 + R 2886 and R 2057; (2) S$_f$ 1652 + R 3577 and R 1712; and (3) S$_f$ 1652 + R 4232 and R 1384. Table 9 provides a comparison of the proposed Sim2Real approach with other ship-detecting techniques specific to the same domain.

It is evident that the Sim2Real adaptation approach accomplishes high performance with a notably decreased amount of labeled real-world data. In particular, the mAP@0.5 reached 97.7% when 2886 real-world images were used for hybrid transfer learning. This outperforms the result reported by Zhou and Peng (2023), whose model is trained with 6000 real-world images, as well as those reported by Liu et al. (2021) and T. Liu et al. (2024), both trained with over 6000 real-world images. Increasing the number of labeled real-world images for hybrid transfer learning to 4232 yields an mAP@0.5 of 98.7%, which matches the accuracy achieved by Chen et al. (2021) using 12,600 real-world images. This represents a 66.4% reduction in the size of real-world labeling. Additionally, the model parameters are lowered by 85.5%, while

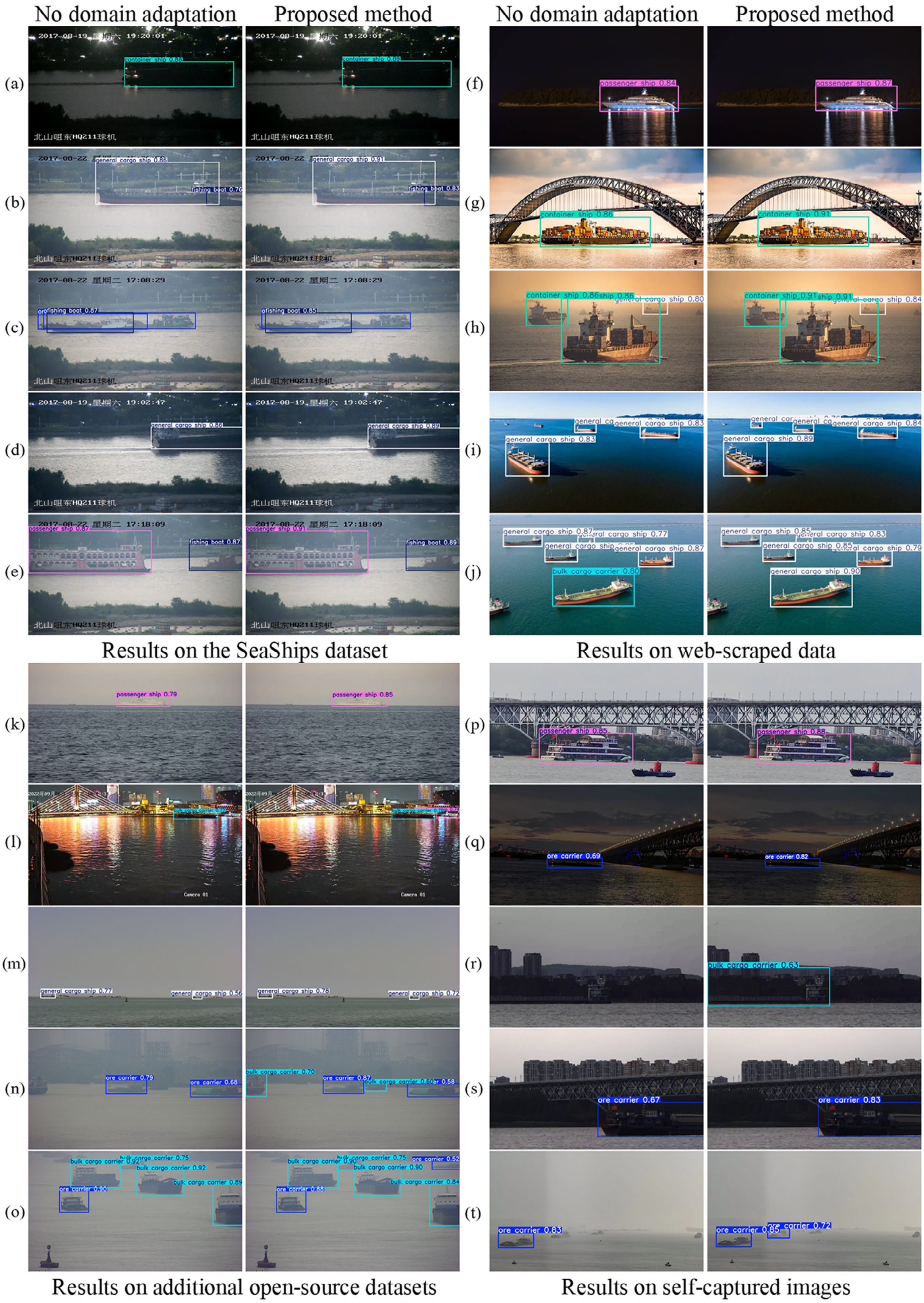

No domain adaptation
Proposed method
No domain adaptation
Proposed method
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
Results on the SeaShips dataset
Results on web-scraped data
(k)
(l)
(m)
(n)
(o)
(p)
(q)
(r)
(s)
(t)
Results on additional open-source datasets
Results on self-captured images

FIGURE 13 Comparison of the adaptive and non-adaptive models' detection results.

achieving a 376.7% improvement in FPS. In terms of inference latency, measured in milliseconds (ms), the proposed method is particularly more efficient than those developed by Liu et al. (2021), Chen et al. (2021), and Zhou and Peng (2023). It is only 3.5 ms slower than the method proposed by T. Liu et al. (2024), which sacrifices considerable detection accuracy for speed. In summary, the MFL-YOLOv8 model developed in this study capably balances lightweight design, detection accuracy, and speed, while substantially decreasing the demand

TABLE 9 Comparison of results on the SeaShips dataset.

Method	Real-world data	Metrics (%)	All	OC	BCC	GCS	CS	FB	PS	Params (M)	Inference time (ms)	FPS
Liu et al. (2021)	>6000	mAP@0.5	87.7	90.6	85.2	88.0	91.1	89.5	82.1	–	33.3	30
Chen et al. (2021)	12,600	mAP@0.5	98.7	99.0	99.0	99.0	100.0	98.0	97.0	61.5	31.3	32
Zhou and Peng. (2023)	6000	mAP@0.5	97.6	98.4	96.3	97.5	98.3	97.2	98.0	40.3	16.7	60
		mAP@0.5:0.95	71.0	64.4	67.8	74.1	79.4	65.6	74.4			
T. Liu et al. (2024)	>6000	mAP@0.5	97.1	94.8	98.7	97.6	96.2	97.7	97.3	4.9	3.5	290
Ours-1	2886	mAP@0.5	97.7	98.0	97.6	97.8	99.3	97.8	95.4	8.9	6.7	148
		mAP@0.5:0.95	80.8	80.7	79.3	83.5	86.2	77.7	77.6			
Ours-2	3577	mAP@0.5	98.3	98.5	98.1	97.9	99.4	98.2	97.5	8.9	6.8	146
		mAP@0.5:0.95	81.1	80.8	79.5	83.6	86.6	78.2	77.8			
Ours-3	4232	mAP@0.5	98.7	98.7	98.7	98.4	99.4	98.6	98.2	8.9	7.0	143
		mAP@0.5:0.95	81.6	81.4	80.3	83.9	86.9	78.8	78.5			

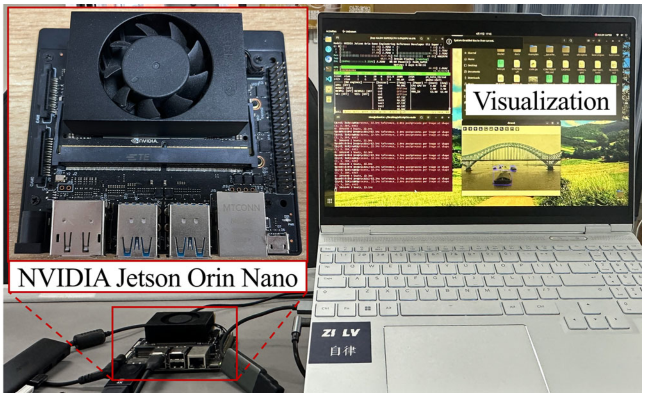


FIGURE 14 The actual deployment scenario.

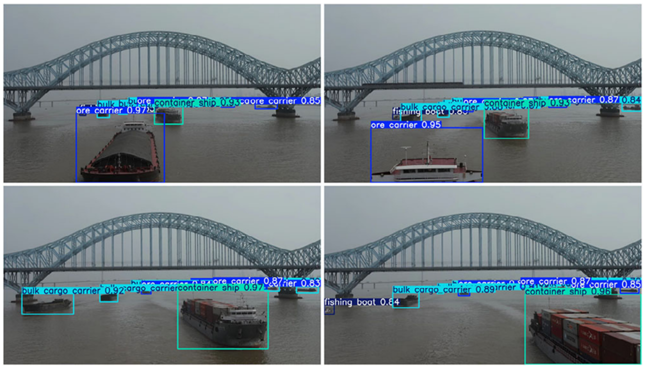


FIGURE 15 Video detection of the adapted MFL-YOLOv8.

for labeled real-world data through supervised Sim2Real adaptation.

4.3.4 | Edge inference applications

To evaluate the stability of the proposed method in continuous video processing for edge inference, the model is deployed on the NVIDIA Jetson Orin Nano Developer Kit. The embedded platform integrates a 1024-core Ampere GPU with 32 Tensor Cores, a 6-core CPU, and 8 GB of memory. A self-captured waterway ship video with a resolution of 1920 × 1080 pixels at 25 FPS (7625 frames in total) is used for testing. The actual implementation situation is provided in Figure 14.

During inference, all input frames are resized to a fixed resolution of 640 × 640 pixels. The deployment is carried out under JetPack 5.1, which provides the necessary drivers and libraries for GPU-accelerated inference on the Jetson Orin Nano (Nnadozie et al., 2024). All tests are conducted with the MAXN power configuration enabled to ensure stable and consistent performance throughout the experiment. Table 10 presents the statistics for edge inference applications.

The adapted MFL-YOLOv8 achieves the lowest misdetection and omission rates among all compared models. Besides, it outperforms YOLOv8-N in terms of model size and computational complexity. In addition to its architectural enhancements, the adapted model is able to maintain an inference speed of 35 FPS on the single-board computer platform, which is generally considered sufficient for practical ship detection in real-time video applications (Shao et al., 2018). Figure 15 displays the visual detection results of the adapted MFL-YOLOv8 implemented on the Jetson Orin Nano. Experimentally, since the model has already fully learned the distinctive features of various ships, the proposed method showcases robustness in real-time ship detection and reduces run-time complexity compared to the vanilla YOLOv8.

In comparison with satellite-based radio navigation systems such as AIS, which rely on intermittent signal transmissions and may encounter delays (with update intervals sometimes extending to several minutes), the proposed vision-based method offers a markedly faster response with millisecond-level latency (M. Zhang et al.,

TABLE 10 Experimental results for edge computing scenarios.

Model	Misdetection (%)	Omission (%)	Latency (ms)	FPS	Model size (MB)	Params (M)	GFLOPs
YOLOv8-N	4.2	16.1	23.8	42	21.4	11.2	28.8
MFL-YOLOv8-N	2.9	12.5	29.4	34	17.2	8.9	9.8
Adapted MFL-YOLOv8	1.0	6.3	28.6	35	17.2	8.9	9.8

Note: Misdetection is the percentage of incorrect identifications among all detected ships, omission is the percentage of missed targets among all targets, latency refers to the average execution time required to process a single frame of the test video during inference, FPS indicates the average inference speed, and model size is the storage space required for the model.

2024). In addition, the absence or deliberate deactivation of transponders on some ships can cause signal interruptions and data loss. Therefore, the proposed approach serves as a valuable complementary solution, particularly in maritime surveillance where rapid response and high perceptual sensitivity are required.

5 | CONCLUSION

This work proposes a Sim2Real domain adaptation framework that leverages synthetic data to pre-train a lightweight ship detector, enabling its effective application in real-world settings. Initially, a refined 3D virtual environment is built to generate a synthetic image dataset named SSDShips. Based on the YOLOv8 framework, a novel lightweight network named MFL-YOLOv8 is developed, integrating EfficientNetV2 with the optimized BiFPN, LSK attention mechanism, and AOD-Net. MFL-YOLOv8 is initially pre-trained with the SSDShips dataset. Cycle-GAN is used to deliver more realistic images for lessening domain gaps, and these pseudo-realistic images are combined with real data to construct hybrid datasets. Hybrid datasets are sent to the fine-tuning step that transfers both model parameters and the gradient history. Edge inference is conducted using the adapted model on the NVIDIA Jetson Orin Nano. The research leads to the following conclusions:

1. MFL-YOLOv8 pre-trained using the generated synthetic dataset demonstrates the capability to detect real-world ships, establishing it as a valuable benchmark for virtual pre-training in ship detection.
2. Due to its lightweight design and multi-level feature extraction, MFL-YOLOv8 reduces model parameters by 20.5% and GFLOPs by 66.0%, compared to its baseline, while attaining stronger detection accuracy.
3. The proposed Sim2Real adaptation framework adequately minimizes the need for labeled real-world data while maintaining high precision and efficiency in ship detection. Its execution on the NVIDIA Jetson Orin Nano displays stable performance in continuous video processing. Hence, the method provides a lightweight solution to address the cold-start problem in newly developed ship detection systems.

Despite advancements in model light-weighting and data efficiency, this study still has several limitations. Efficacy under extreme weather conditions and ultra-long-range detection of small ships remains suboptimal. Future work will focus on optimizing models for these difficult situations. More unseen ship categories, such as military ships, are expected to be introduced into the framework to elevate model generalization across a broader range of domains. Additionally, this work primarily focuses on supervised domain adaptation, which still requires some labeled data for fine-tuning. It is important to explore combining other sophisticated supervised machine learning/classification algorithms such as the neural dynamic classification algorithm (Rafiei & Adeli, 2017), finite element machine for fast learning (Pereira et al., 2020), and specially self-supervised learning (Rafiei et al., 2024), to further bridge the gap between virtual and real domains.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. 52338011), the Start-up Research Fund of Southeast University (No. RF1028624058), and the Southeast University Interdisciplinary Research Program for Young Scholars.

REFERENCES

Adeli, H. (2001). Neural networks in civil engineering: 1989–2000. *Computer-Aided Civil and Infrastructure Engineering*, *16*(2), 126–142.

Adeli, H., & Hung, S. L. (1994). *Machine learning: Neural networks, genetic algorithms, and fuzzy systems*. John Wiley & Sons, Inc.

Adeli, H., & Karim, A. (2000). Fuzzy-Wavelet RBFNN Model for Freeway Incident Detection. *Journal of Transportation Engineering*, *126*(6), 464–471.

Alam, K. M. R., Siddique, N., & Adeli, H. (2020). A dynamic ensemble learning algorithm for neural networks. *Neural Computing and Applications*, *32*(12), 8675–8690.

Bavelos, A. C., Anastasiou, E., Dimitropoulos, N., Michalos, G., & Makris, S. (2025). Virtual reality-based dynamic scene recreation and robot teleoperation for hazardous environments. *Computer-Aided Civil and Infrastructure Engineering*, *40*, 392–408.

Chen, D., Sun, S., Lei, Z., Shao, H., & Wang, Y. (2021). Ship target detection algorithm based on improved YOLOv3 for maritime image. *Journal of Advanced Transportation*, *2021*(1), 9440212.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning, PMLR*, Sydney, Australia (pp. 1126–1135).

Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV (pp. 4349–4349).

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile (pp. 1448–1448).

Goodman, S., Greenspan, H., & Goldberger, J. (2023). Supervised domain adaptation by transferring both the parameter set and its gradient. *Neurocomputing*, *560*, 126828.

Guan, W., Chen, Z., Wang, S., Wang, G., Guo, J., & Liu, Z. (2023). A deep learning approach for construction vehicles fill factor estimation and bucket detection in extreme environments. *Computer-Aided Civil and Infrastructure Engineering*, *38*(13), 1857–1878.

Gundogdu, E., Solmaz, B., Yücesoy, V., & Koc, A. (2017). MARVEL: A large-scale image dataset for maritime vessels. In S. H. Lai, V. Lepetit, K. Nishino, & Y. Sato (Eds.), Lecture notes in computer science: Vol. 10115. *Computer vision–ACCV 2016: 13th Asian conference on computer vision* (pp 165–180). Springer.

Guo, A., Jia, Z., Wang, J., Zhou, G., Ge, B., & Chen, W. (2024). A lightweight weed detection model with global contextual joint features. *Engineering Applications of Artificial Intelligence*, *136*, 108903.

Guo, Y., Xu, Y., Niu, J., & Li, S. (2023). Anchor-free arbitrary-oriented construction vehicle detection with orientation-aware Gaussian heatmap. *Computer-Aided Civil and Infrastructure Engineering*, *38*(7), 907–919.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020). Ghostnet: More features from cheap operations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA (pp. 1589–1589).

He, B., Huang, B., Shen, Y., & Wu, L. (2022). Delve into balanced and accurate approaches for ship detection in aerial images. *Neural Computing and Applications*, *34*, 15293–15312.

He, B., Li, X., Huang, B., Gu, E., Guo, W., & Wu, L. (2021). UnityShip: A large-scale synthetic dataset for ship recognition in aerial images. *Remote Sensing*, *13*(24), 4999.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANS trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30, Long Beach, CA.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT (pp. 7141–7141).

Huang, Q., Wang, J., Song, Y., Cui, W., Li, H., Wang, S., Dai, P., Zhao, X., & Li, Q. (2024). Synthetic-to-realistic domain adaptation for cold-start of rail inspection systems. *Computer-Aided Civil and Infrastructure Engineering*, *39*(3), 424–437.

Javadinasab Hormozabad, S., Gutierrez Soto, M., & Adeli, H. (2021). Integrating structural control, health monitoring, and energy harvesting for smart cities. *Expert Systems*, *38*(8), e12845.

Jiang, T., Li, L., Samali, B., Yu, Y., Huang, K., Yan, W., & Wang, L. (2024). Lightweight object detection network for multi-damage recognition of concrete bridges in complex environments. *Computer-Aided Civil and Infrastructure Engineering*, *39*, 3646–3665.

Kaur, P., Aziz, A., Jain, D., Patel, H., Hirokawa, J., Townsend, L., Reimers, C., & Hua, F. (2022). Sea situational awareness (seasaw) dataset. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA (pp. 2587–2587).

Kim, H.-S., Seong, J., & Jung, H.-J. (2024). Optimal domain adaptive object detection with self-training and adversarial-based approach for construction site monitoring. *Automation in Construction*, *158*, 105244.

Li, B., Peng, X., Wang, Z., Xu, J., & Feng, D. (2017). AOD-Net: All-in-one dehazing network. *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy (pp. 4778–4778).

Li, D., Liu, M., Yang, L., Wei, H., & Guo, J. (2024). A non-contact identification method of overweight vehicles based on computer vision and deep learning. *Computer-Aided Civil and Infrastructure Engineering*, *39*(22), 3452–3476.

Li, Y., Hou, Q., Zheng, Z., Cheng, M.-M., Yang, J., & Li, X. (2023). Large selective kernel network for remote sensing object detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Paris, France (pp. 16805–16805).

Liu, R. W., Yuan, W., Chen, X., & Lu, Y. (2021). An enhanced CNN-enabled learning method for promoting ship detection in maritime surveillance system. *Ocean Engineering*, *235*, 109435.

Liu, S., Ha, D. S., Shen, F., & Yi, Y. (2021). Efficient neural networks for edge devices. *Computers & Electrical Engineering*, *92*, 107121.

Liu, T., Zhang, Z., Lei, Z., Huo, Y., Wang, S., Zhao, J., Zhang, J., Jin, X., & Zhang, X. (2024). An approach to ship target detection based on combined optimization model of dehazing and detection. *Engineering Applications of Artificial Intelligence*, *127*, 107332.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture notes in computer science: Vol. 9905. Computer vision–ECCV 2016: 14th European conference* (pp. 21–37). Springer.

Nnadozie, E. C., Casaseca-de-la-Higuera, P., Iloanusi, O., Ani, O., & Alberola-López, C. (2024). Simplifying YOLOv5 for deployment in a real crop monitoring setting. *Multimedia Tools and Applications*, *83*(17), 50197–50223.

Oza, P., Sindagi, V. A., Sharmini, V. V., & Patel, V. M. (2023). Unsupervised domain adaptation of object detectors: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Pan, X., Xiao, Y., Yao, H., Yang, T. Y., & Adeli, H. (2023). Vision-based real-time structural vibration measurement through deep-learning-based detection and tracking methods. *Engineering Structures*, *281*, 115676.

Pereira, D. R., Piteri, M. A., Souza, A. N., Papa, J., & Adeli, H. (2020). FEMa: A finite element machine for fast learning. *Neural Computing and Applications*, *32*, 6393–6404.

Pezeshki, H., Adeli, H., Pavlou, D., & Siriwardane, S. C. (2023). State of the art in structural health monitoring of offshore and marine structures. *Proceedings of the Institution of Civil Engineers—Maritime Engineering*, *176*(2), 89–108.

Pezeshki, H., Pavlou, D., Adeli, H., & Siriwardane, S. C. (2023). Modal analysis of offshore wind turbine structures: An analytical solution. *ASME Journal of Offshore Mechanics and Arctic Engineering*, *145*(1), 010907.

Rafiei, M. H., & Adeli, H. (2017). A new neural dynamic classification algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(12), 3074–3083.

Rafiei, M. H., & Adeli, H. (2018). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, *156*, 598–607.

Rafiei, M. H., Gauthier, L., Adeli, H., & Takabi, D. (2022). Self-supervised learning for electroencephalography. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(2), 1457–1471.

Rafiei, M. H., Gauthier, L., Adeli, H., & Takabi, D. (2024). Self-supervised learning for near-wild cognitive workload estimation. *Journal of Medical Systems*, *48*(1), 107.

Raza, M., Prokopova, H., Huseynzade, S., Azimi, S., & Lafond, S. (2022). SimuShips–A high resolution simulation dataset for ship detection with precise annotations. *OCEANS 2022*, Chennai, India (pp. 1–5).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV (pp. 788–788).

Satyanath, G., Sahoo, J. K., & Roul, R. K. (2023). Smart parking space detection under hazy conditions using convolutional neural networks: A novel approach. *Multimedia Tools and Applications*, *82*(10), 15415–15438.

Shao, Z., Wu, W., Wang, Z., Du, W., & Li, C. (2018). SeaShips: A large-scale precisely annotated dataset for ship detection. *IEEE Transactions on Multimedia*, *20*(10), 2593–2604.

Shi, Y., Du, L., Guo, Y., & Du, Y. (2022). Unsupervised domain adaptation based on progressive transfer for ship detection: From optical to SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–17.

Song, R., Li, T., & Li, T. (2023). Ship detection in haze and low-light remote sensing images via colour balance and DCNN. *Applied Ocean Research*, *139*, 103702.

Tan, M., & Le, Q. (2021). EfficientNetV2: Smaller models and faster training. *International Conference on Machine Learning*, PMLR, Virtual (pp. 10106–10106).

Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA (pp. 10790–10790).

Tang, J., Deng, C., Huang, G.-B., & Zhao, B. (2014). Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(3), 1174–1185.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada (pp. 30–30).

Vazquez, D., Lopez, A. M., Marin, J., Ponsa, D., & Geronimo, D. (2013). Virtual and real world adaptation for pedestrian detection. *IEEE transactions on Pattern Analysis and Machine Intelligence*, *36*(4), 797–809.

Wang, H., Liu, C., Cai, Y., Chen, L., & Li, Y. (2024). YOLOv8-QSD: An improved small object detection algorithm for autonomous vehicles based on YOLOv8. *IEEE Transactions on Instrumentation and Measurement*, *73*, 1–16.

Wang, K. C., Hou, Z., & Gong, W. (2010). Automated road sign inventory system based on stereo vision and tracking. *Computer-Aided Civil and Infrastructure Engineering*, *25*(6), 468–477.

Wang, S., Li, Y., & Qiao, S. (2024). ALF-YOLO: Enhanced YOLOv8 based on multiscale attention feature fusion for ship detection. *Ocean Engineering*, *308*, 118233.

Xie, W., Sun, X., & Ma, W. (2024). A light weight multi-scale feature fusion steel surface defect detection model based on YOLOv8. *Measurement Science and Technology*, *35*(5), 055017.

Zhang, B., Zhou, L., & Zhang, J. (2019). A methodology for obtaining spatiotemporal information of the vehicles on bridges based on computer vision. *Computer-Aided Civil and Infrastructure Engineering*, *34*(6), 471–487.

Zhang, M., Zhang, Q., Song, R., Rosin, P. L., & Zhang, W. (2024). Ship landmark: An informative ship image annotation and its applications. *IEEE Transactions on Intelligent Transportation Systems*, *25*(11), 17778–17793.

Zheng, Y., & Zhang, S. (2020). Mcships: A Large-Scale Ship Dataset For Detection And Fine-Grained Categorization In The Wild. *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. https://doi.org/10.1109/icme46284.2020.9102907

Zhou, W., & Peng, Y. (2023). Ship detection based on multi-scale weighted fusion. *Displays*, *78*, 102448.

Zhu, J., Wu, Y., & Ma, T. (2024). Multi-object detection for daily road maintenance inspection with UAV based on improved YOLOv8. *IEEE Transactions on Intelligent Transportation Systems*, *25*(11), 16548–16560.

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy (pp. 2232–2232).

Zhu, L., Wang, X., Ke, Z., Zhang, W., & Lau, R. W. (2023). Biformer: Vision transformer with bi-level routing attention. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Vancouver, BC, Canada (pp. 10333–10333).

How to cite this article: Liao, R., Zhang, Y., Wang, H., Lu, L., Chen, Z., Wang, X., & Zuo, W. (2025). An effective ship detection approach combining lightweight networks with supervised simulation-to-reality domain adaptation. *Computer-Aided Civil and Infrastructure Engineering*, 1–26. https://doi.org/10.1111/mice.13501