Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.028738

ARTICLE

EfficientShip: A Hybrid Deep Learning Framework for Ship Detection in the River

Huafeng Chen[1], Junxing Xue[2], Hanyun Wen[2], Yurong Hu[1] and Yudong Zhang[3,*]

[1]School of Computer Engineering, Jingchu University of Technology, Jingmen, 448000, China

[2]School of Computer Science, Yangtze University, Jingzhou, 434023, China

[3]School of Computing and Mathematic Sciences, University of Leicester, Leicester, LE1 7RH, UK

*Corresponding Author: Yudong Zhang. Email: yudongzhang@ieee.org

Received: 05 January 2023 Accepted: 05 May 2023 Published: 22 September 2023

ABSTRACT

Optical image-based ship detection can ensure the safety of ships and promote the orderly management of ships in offshore waters. Current deep learning researches on optical image-based ship detection mainly focus on improving one-stage detectors for real-time ship detection but sacrifices the accuracy of detection. To solve this problem, we present a hybrid ship detection framework which is named EfficientShip in this paper. The core parts of the EfficientShip are DLA-backboned object location (DBOL) and CascadeRCNN-guided object classification (CROC). The DBOL is responsible for finding potential ship objects, and the CROC is used to categorize the potential ship objects. We also design a pixel-spatial-level data augmentation (PSDA) to reduce the risk of detection model overfitting. We compare the proposed EfficientShip with state-of-the-art (SOTA) literature on a ship detection dataset called Seaships. Experiments show our ship detection framework achieves a result of 99.63% (mAP) at 45 fps, which is much better than 8 SOTA approaches on detection accuracy and can also meet the requirements of real-time application scenarios.

KEYWORDS

Ship detection; deep learning; data augmentation; object location; object classification

1 Introduction

With the continuous advancement of technology and the rapid development of industrial production, international trade is gradually increasing. The market of the shipping industry is also flourishing. In order to ensure the safety of ships and promote the orderly management of ships, satellites (generate SAR images) are used to monitor ships at sea [1] and surveillance cameras (generate optical images) are adopted for tracking ships in offshore waters [2,3]. At the technical level, with the maturity of artificial intelligence technology [4], computer-aided methods of ship classification, ship instance segmentation and ship detection from images are studied to reduce the burden on human monitors [5]. We focus on ship detection based on optical images generated by surveillance cameras in this paper.

In recent years, deep learning-based ship detection has become a hot research area [6–8]. Sea ship detection is one of the general object detections [9]. Researches on deep learning based object detection can be roughly split into two classifications: One-stage detectors and two-stage detectors [10]. One-stage detectors combine object location and classification in one deep learning framework, while two-stage detectors find object location in the first place and classify the potential objects secondly. Representative one-stage detection algorithms are RetinaNet [11], FCOS [12], CenterNet [13], ATSS [14], PAA [15], BorderDet [16], and YOLO series [17–21]. Mainstream two-stage object detection approaches are R-CNN [22], SPPNet [23], Fast RCNN [24], Faster RCNN [25], FPN [26], Cascade RCNN [27], Grid RCNN [28], and CenterNet2 [29].

Generally, the one-stage detector is considered to have a faster detection speed, while the two-stage detection algorithm has higher detection accuracy. While recent methods of ship detection [3,30–37] focus on improving one-stage detectors for real-time ship detection, they sacrifice the accuracy of detection. In this paper, we present a real-time two-stage ship detection algorithm, which improves detection accuracy while ensuring real-time performance. The algorithm includes two parts: the DLA-backboned object location (DBOL) and the CascadeRCNN-guided object classification (CROC). To further improve the accuracy of ship detection, we design a novel pixel-spatial-level data augmentation (PSDA) for increasing the number of samples at a high multiple and effectively. The PSDA, DBOL and CROC make up the proposed hybrid deep learning framework of EfficientShip.

The contributions of this study can be summarized as follows:

(1) The DBOL is presented for finding potential ship objects in real time. We integrate DLA [38], ResNet-50 [39] and CenterNet [13] into DBOL for evaluating object likelihoods quickly and accurately.

(2) The CROC is put forward to real-time categorize the potential ship objects. We calculate the category scores of suspected objects based on conditional probability and extrapolate the final detection.

(3) The PSDA is proposed to reduce the risk of the model overfitting. We amplify the original data by 960 times based on pixel and spatial image augmentation.

(4) Our EfficientShip (includes PSDA, DBOL and CROC) gets the best performance compared with 8 existing SOTA methods: 99.63% (accuracy) with 45 fps (speed).

2 Related Work

2.1 Ship Detection

Ship detection can be divided into SAR image-based [5,40] and optical image-based ship detection [2,3]. Here we focus on reviewing optical image-based ship detection. Traditional optical image-based ship detection use hand-crafted features which sliding window to obtain the candidate area of the ship target based on the saliency map algorithm or the visual attention mechanism. The features of the candidate target are extracted for training to obtain the detection model [41,42].

Recently, deep learning-based ship detection has attracted researchers' attention. Shao et al. [3] introduced a CNN framework on the basis of saliency-aware for ship detection. Based on YOLOv2, the ship's location and classification under a complex environment were inferred by CNN firstly and were refined through saliency detection. Sun et al. [32] presented an algorithm named NSD-SSD for real-time ship detection. They combined dilated convolution and multiscale feature to promote knockdown performance in detecting a small object of a ship. For getting the inferring score of every class and the variation of every prior bounding box, they also designed a batch of convolution filters at every trenchant feature layer. They finally reconstructed prior boxes with K-means clustering to advance visual accuracy and the ship-detecting efficiency.

Liu et al. [31] have designed an advanced CNN-enabled learning method for promoting ship detection under different weather conditions. On the basis of YOLOv3, they devised new scale of anchor boxes, localization probability of bounding boxes, soft non-maximum suppression, and medley loss function for advancing the CNN capacities of learning and expression. On the other hand, they introduced an agile DA tactics through produce synthetically-degraded pictures to enlarge the capacity and diversity of rudimentary ship detection dataset. Considering the influence of meteorological factors on ship detection accuracy, Nie et al. [30] synthesized foggy images and low visibility pictures via exploiting physical models separately. They trained YOLOv3 on the expanded dataset, including both composite and original ship pictures and illustrated that the trained model achieved excellent ship detection accuracy within a variety of weather conditions. For real-time ship detection, Li et al. [33] concentrated the network of YOLOv3 by training predetermined anchors based on the annotations of Seaship, instead max-pooling layer with convolution layer, expanding channels of prediction network to promote the detection ability of tiny object, and embedding CBAM attention module into the backbone network to facilitate the model focusing on the object. Liu et al. [43] proposed two new anchor-setting methods, the average method and the select-all method, for detecting ship targets on the basis of YOLOv3. Additionally, they adopt the feature fusion structure of cross PANet for combining the different anchor-setting methods. Chen et al. [35] introduced the AE-YOLOv3 for real-time end-to-end ship identification. AE-YOLOv3 was merged in the feature attention module, embedded with the feature extraction network, and fused through multiscale feature enhancement model.

Liu et al. [34] presented a method of RDSC on the basis of YOLOv4 by reducing more than 40% weights compared to the original one. The improved lightweight algorithm achieved a tinier network volume and preferable real-time performance on ship detection. Zhang et al. [36] presented a lightweight CNN named Light-SDNet for detecting ships under various weather conditions. Based on YOLOv5, they modificated CA-Ghost, C3Ghost, and DWConv to decrease the model parameters size. They designed a hybrid training tactic by deriving jointly-degraded pictures to expand the number of the primitive dataset. Zhou et al. [37] improved YOLOv5 for ship target detection, and named it as YOLO-Ship, which adopted MixConv to update classical convolution operation and concordant attention framework. At decision stage, they employed Focal Loss and CIoU Loss for optimizing raw cost functions.

In order to reach the goal of real-time application while obtaining detection accuracy, most of the above algorithms choose a one-stage detection algorithm as the basis for improvement. Different from these methods, we present a real-time approach of two-stage detection as the main ship detection framework and verify its accuracy and real-time performance through experiments.

2.2 Data Augmentation (DA)

Image data collection and labeling are very labor-intensive. Due to funding constraints, ship detection datasets usually have only thousands of annotated images [2]. But the deep learning model has many parameters and requires tens of thousands of data for training. While a deep convolutional neural network (CNN) learns a function that has a very high correlation with the small training data, it is poorly generalizable to testing set (overfitting). Data augmentation technology can simulate training image data through lighting variations, occlusion, scale and orientation variations, background clutter, object deformatio, etc., so that the deep learning model is robust to these disturbances and reducing overfitting on testing data [44,45].

Image DA algorithms can be split into basic image manipulations and deep learning approaches [44]. Basic image manipulations change original image pixels while the image label is conserved. Basic image manipulations include geometric transformations, color space transformations, kernel filters and random erasing. Image geometric transformations shift the geometry of image without altering its actual pixel values. Simple geometric transformations cover flipping, cropping, rotation and translation. Color space transformations will shift pixel values through an invariable number, separate RGB color channel or limit pixel values into a range. The methods of kernel filter sharpen or blur original images via sliding of filter matrix across training image. Inspired by CNN dropout regularization, random erasing does the operation of masking training image patch with the values 0, 255, or random number. Taylor et al. proved the effectiveness of geometric and color space transformations [46], while Zhong et al. verified the performance of random erasing through experiments [47]. Xu et al. presented a novel shadow enhancement named SBN-3D-SD for higher detection-tracking accuracy [48].

Deep learning-based augmentation adopts learning methods to produce synthetic examples for training data. It can be divided into adversarial training based DA, GAN-based DA, neural transfer based DA, and meta-learning-based DA [44]. Adversarial training based DA generates adversarial samples and inserts them into the training set so that the inferential model can learn from the adversarial samples during training [49]. Method of GAN is an unsupervised generative model that can generate synthetic data given a random noise vector. Adding the data generated by GAN-based DA into the training set can optimize deep learning model parameters [50]. The idea of neural style transfer is to manipulate sequential features across a CNN so that the image pattern can be shifted into other styles while retaining its primitive substance. Meta-learning-based DA uses a pre-prepared neural network to learn DA parameters from medley images, Neural Style Transfer, and geometric transfigurations. The image generated by deep learning-based augmentation is abstract and cannot pinpoint target bounding boxes. So it is not suitable for ship detection.

3 Methodology

In this section, we describe the method of EfficientShip for ship detection. It includes proposed PSDA, DBOL and CROC (as shown in Fig. 1).

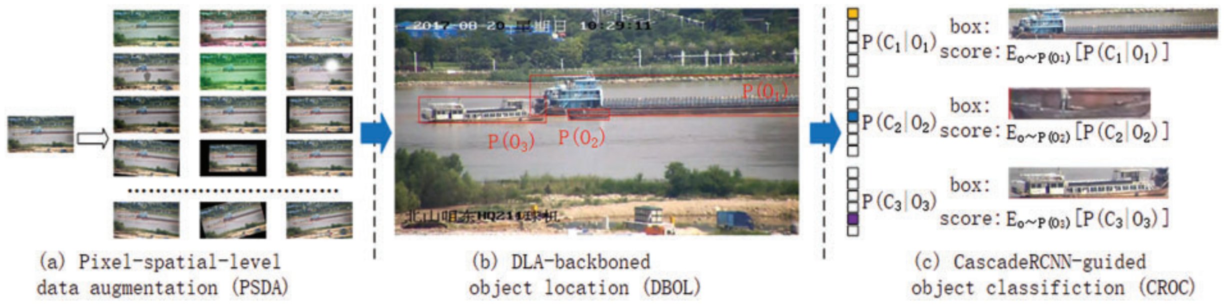
P(O1)
P(O3)
P(O2)
P(C1|O1)
box:
score: E_{o~P(O1)}[P(C1|O1)]
P(C2|O2)
box:
score: E_{o~P(O2)}[P(C2|O2)]
P(C3|O3)
box:
score: E_{o~P(O3)}[P(C3|O3)]
(a) Pixel-spatial-level data augmentation (PSDA)
(b) DLA-backboned object location (DBOL)
(c) CascadeRCNN-guided object classifiction (CROC)

Figure 1: The architecture of the proposed EfficientShip. PSDA is used for expanding the amount of image sample; DBOL is responsible for detecting potential objects; CROC tries to identify the potential objects

3.1 Proposed PSDA

The ship detection dataset is small for the current study. Therefore, we present a method named PSDA to counteract the overfitting of the ship detection model. PSDA includes pixel level DA (PDA), spatial level DA (SDA), and their combination. PDA will change the content of the input image at the pixel level, and SDA is to perform geometric transformations on it.

Suppose the number of DA methods we used is m_{da}, and a train image $x^{tr}(i) \in X^{tr}$, where X^{tr} indicates the train set. Each DA method will generate n_{da} (as shown in Fig. 2), for every image will produce $m_{da} \times n_{da}$ new images. At the pixel level, we will perform five subsequent DA methods for the training image set X^{tr}.

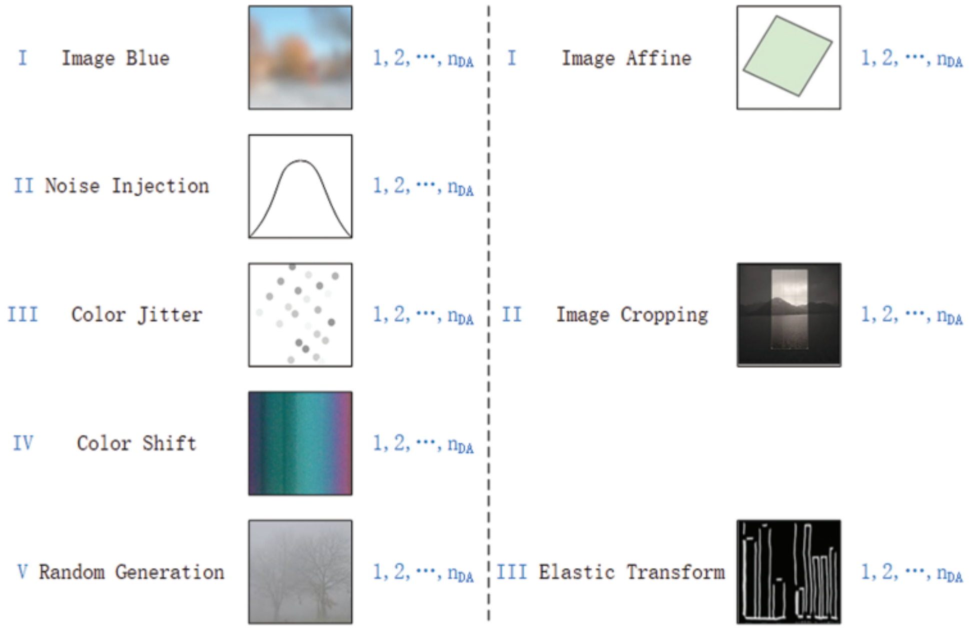

Figure 2: Schematic of proposed PSDA. (a) PDA is used for expanding the amount of image sample at pixel level; (b) SDA is used for expanding the amount of image sample at spatial level

(I) Image Blur

Applying an image blur algorithm to a raw image can generate n_{da} images.

$$\begin{aligned} x^{tr\_p1}\overrightarrow{(i)} &= F_{IB}[x^{tr}(i)] \\ &= [x_1^{tr\_p1}(i), \cdots, x_{n_{DA}}^{tr\_p1}(i)] \end{aligned} \tag{1}$$

where F_{IB} means a certain image blur function [51]. The functions include Gaussian blur, glass blur, median blur, motion blur, zoom blur, etc.

(II) Noise Injection

New n_{da} images were generated by noise injection.

$$\begin{aligned} x^{tr\_p2}\overrightarrow{(i)} &= F_{NI}[x^{tr}(i)] \\ &= [x_1^{tr\_p2}(i), \cdots, x_{n_{DA}}^{tr\_p2}(i)] \end{aligned} \tag{2}$$

where F_{NI} means a noise injection function [51]. Noise injection algorithms include Gaussian noise, ISO noise, multiplicative noise, etc.

(III) Color Jitter

Color jitter generates a minor variations of color values in the training image.

$$\begin{aligned} x^{tr\_p3}\overrightarrow{(i)} &= F_{CJ}[x^{tr}(i)] \\ &= [x_1^{tr\_p3}(i, h_b^f), x_1^{tr\_p3}(i, h_c^f), x_1^{tr\_p3}(i, h_s^f), \cdots, x_{nDA}^{tr\_p3}(i, h_s^f)] \end{aligned} \tag{3}$$

where F_{CJ} denotes color jitter [51]. Color jitter can be operated from three aspects: h_b^f-brightness, h_c^f-contrast and h_s^f-saturation.

(IV) Color Shift

Color shift is color variation caused by different fade rates of dyes or imbalance of dyes within a picture patch.

$$\begin{aligned} x^{tr\_p4}\overrightarrow{(i)} &= F_{CS}[x^{tr}(i)] \\ &= [x_1^{tr\_p4}(i, t_r^f), x_1^{tr\_p4}(i, t_b^f), x_1^{tr\_p4}(i, t_g^f), \cdots, x_{nDA}^{tr\_p4}(i, t_g^f)] \end{aligned} \tag{4}$$

where F_{CS} means color shift [51]. Color shift can be operated from three channels: t_r^f-red, t_b^f-blue and t_g^f-green.

(V) Random Generation

Random generation method can generate new images by performing multiple operations on original image pixels, such as brightness, contrast, gamma correction, curve, fog, rain, shadow, snow, sun flare, etc. Each training image in X^{tr} is operated n_{da} times through random generation g_{op}. The variation range of g_{op} is $[-az, +az]$ and complies with the distribution V.

$$g_{op}^i \sim V[-M_{SR}, +M_{SR}] \tag{5}$$

where M_{SR} is the maximum operation range [52]. Hence, we have

$$\begin{aligned} x^{tr\_p5}(i) &= F_{RG}[x^{tr}(i)] \\ &= [x_1^{tr\_p5}(i, g_{op}^1), x_1^{tr\_p5}(i, g_{op}^2), \cdots, x_{nDA}^{tr\_p5}(i, g_{op}^{nDA})] \end{aligned} \tag{6}$$

where F_{RG} means random generation [45].

At the spatial level, the image transformation will not change the original image content, but the object bounding box will be transformed along with the transformation. The main transformations are:

(I) Image Affine

Image affine is a common geometric transformation that preserves the collinearity between pixels. It includes translation, rotation, scaling, shear and their combination.

$$\begin{aligned} x^{tr\_s1}(i) &= F_{IR}[x^{tr}(i)] \\ &= [x_1^{tr\_s1}(i, h_a), x_2^{tr\_s1}(i, h_a), \cdots, x_{nDA}^{tr\_s1}(i, h_a)] \end{aligned} \tag{7}$$

where F_{IR} means the image affine function, h_a represents an operation of translation, rotation, scaling, or shear [45].

(II) Image Cropping

Image cropping can freely crop the input image to any size.

$$\begin{aligned} x^{tr\_s2}(i) &= F_{IC}[x^{tr}(i)] \\ &= [x_1^{tr\_s2}(i), x_2^{tr\_s2}(i), \cdots, x_{n_{DA}}^{tr\_s2}(i)] \end{aligned} \tag{8}$$

where F_{IC} means the image cropping function [52].

(III) Elastic Transform

Elastic transformation alters the silhouette of the input picture upon the application of a force within its elastic limit. It is controlled by the parameters of the Gaussian filter and affine.

$$\begin{aligned} x^{tr\_s3}(i) &= F_{ET}[x^{tr}(i)] \\ &= [x_1^{tr\_s3}(i), x_2^{tr\_s3}(i), \cdots, x_{n_{DA}}^{tr\_s3}(i)] \end{aligned} \tag{9}$$

where F_{ET} means the elastic transform function [45].

Algorithm 1 shows the pseudocode of PSDA on one training image $x^{tr}(i)$.

Algorithm 1: Pseudocode of PSDA on a training image

```
Input: A raw training image $x^{tr}(i)$
Output: A new dataset $\Phi$ emanated with $|\Phi| = m_{da} \times n_{da}$
1:  #PDA:
2:  Apply image blur for generating $x^{tr\_p1}$
3:  Apply noise injection for generating $x^{tr\_p2}$
4:  Apply color jitter for generating $x^{tr\_p3}$
5:  Apply color shift for generating $x^{tr\_p4}$
6:  Apply random generation for generating $x^{tr\_p5}$
7:  #SDA:
8:  Apply image affine for generating $x^{tr\_s1}$
9:  Apply image cropping for generating $x^{tr\_s2}$
10: Apply elastic transform for generating $x^{tr\_s3}$
11: Combine above outputs, $|\Phi| = x^{tr\_p1} \cup x^{tr\_p2} \cup x^{tr\_p3} \cup x^{tr\_p4} \cup x^{tr\_p5} \cup x^{tr\_s1} \cup x^{tr\_s2} \cup x^{tr\_s3}$
```

3.2 Proposed DLA-Backboned Object Location (DBOL)

The main task in the first step of two-stage object detection is to produce a number of patch bounding boxes with different proportions and sizes according to the characteristic features such as texture, color and other details of the image. Some of the patches represented by bounding boxes contain target, while others only involve background.

As Fig. 1 illustrated, the first step of two-stage ship detection is to generate a set of K ship detections as bounding boxes $b_1, \cdots, b_K$. We use $P(O_k)$ to indicates the likelihood of the object O_k with an unknown category. We can get

$$P(O_k) = \begin{cases} 0, & \text{background} \\ 1, & \text{target waiting to be classified} \end{cases} \tag{10}$$

where $P(O_k) = 0$ shows the object O_k is the background while $P(O_k) = 1$ implies the things O_k in bounding box is a target waiting to be classified [29].

The network architecture of the proposed DBOL is shown n Fig. 3. We select compact DLA [38] as CNN backbone for inferring $P(O_k)$ in the first stage of real-time object detection. The compact DLA runs on the basis of ResNet-50 [39]. The method of CenterNet [13] is used for finding objects as keypoints and regressing to bounding box parameters. The DLA-based feature pyramid generates feature maps from stride 8 to 128. A 4-level regression branch and classification branch are used for all feature pyramids to generate a detection heatmap and bounding box map. During the phase of training, annotations of the actual center are allocated to given feature pyramid levels based on the object scale. Locations are added into the 3×3 neighbor of the center, which will yield superior bounding box as positives. The distance between boundaries is used as the representation of the bounding box, and the gIoU cost is adopted for bounding box regression.

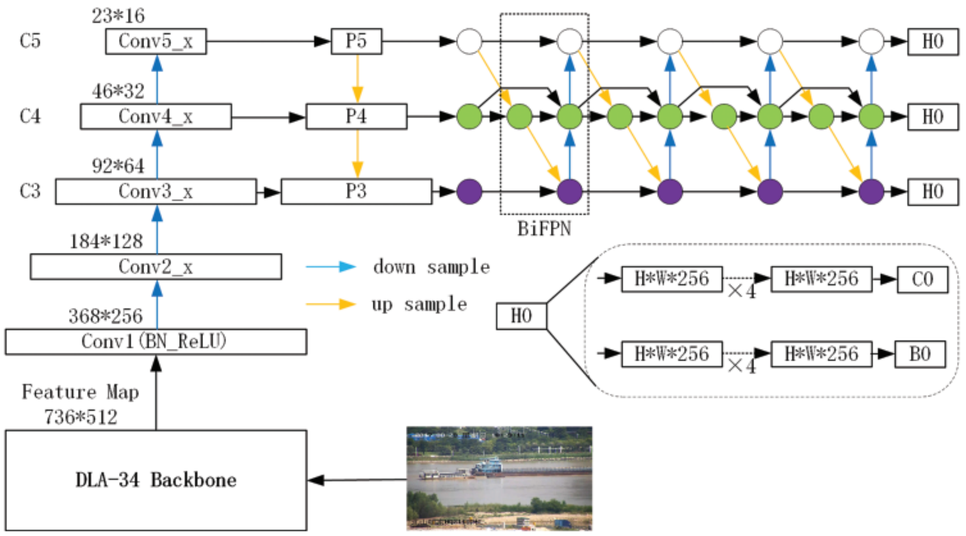

Figure 3: The architecture of the proposed DBOL. "Conv∗" is convolution operation, "C3, C4, C5" denote the feature maps of the backbone network, "P3, P4, P5" are the feature levels used for the final prediction, "H∗" is network head, "B∗" is bouding box of proposals, "C0" is object classification

3.3 Proposed CascadeRCNN-Guided Object Classification (CROC)

For every ship target k, the class distribution is $d_k(c) = P(C_k = c)$ to class $c \in C \cup \{background\}$, where C is a collection of all ship classes. And $P(C_k|O_k)$ designates the conditional categorical classification at the second detection stage. If the equation $P(O_k) = 0$ holds, then $C_k = background$, which means $P(C_k = background|O_k = 0) = 1$.

The conjoint category distribution of the ship detection is

$$P(C_k) = \sum_o P(C_k|O_k = o)P(O_k = o) \tag{11}$$

where o indicates an arbitrary object in the image [29]. Maximum likelihood estimation is employed for training the detectors. For every labeled object, we maximize

$$\log P(C_k) = \log P(C_k|O_k = 1) + \log P(O_k = 1) \tag{12}$$

to decrease to conjoint maximum likelihood objects of the two stages, respectively [29]. The maximum-likelihood objective of the background class is

$$\log P(background) = \log(P(background|O_k = 1)P(O_k = 1) + P(O_k = 0)) \tag{13}$$

The architecture of the proposed CROC is shown in Fig. 4. In this stage of detection, we select CascadeRCNN [27] for inferring $P(C_k|O_k)$ on the basis of $P(O_k)$, which is deduced from the first stage. At each cascade stage t, CascadeRCNN has a classifier h_t optimal for IoU threshold value u^t ($u^t > u^{t-1}$). This is learned through reducing the cost

$$L(x^t, g) = L_{cls}(h_t(x^t), y^t) + \lambda[y^t \geq 1]L_{loc}(f_t(x^t, b^t), g) \tag{14}$$

where $b^t = f_{t-1}(x^{t-1}, b^{t-1})$, g is the ground truth object classification for x^t, $\lambda = 1$ is the trade-off coefficient, $[\cdot]$ is the indicator function, y^t is the label of x^t under given u^t [27].

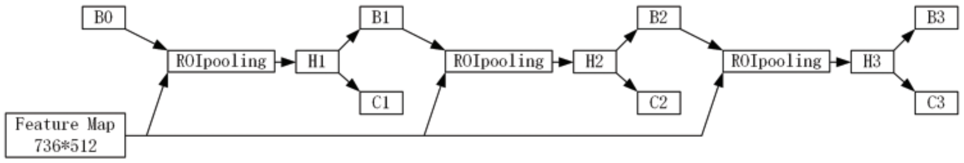

Figure 4: The architecture of the proposed CROC. The Feature Map is generated from DLA-34 backbone network, "H∗" is the network head, "B∗" is the bouding box of proposals, "B0" is the bounding box of proposals produced in Fig. 3

Algorithm 2 shows the pseudocode of the CROC training process.

```
Algorithm 2: Pseudocode of CROC training process
Input: Training images
Output: Trained CNN model
1:  Maximize $\log P(C_k)$ (See Eq. (12))
2:  Factorize $\log P(background)$ (See Eq. (13))
3:  Reduce the cost $L(x^t, g)$ (See Eq. (14))
```

4 Experimental Result and Analysis

In this section, we evaluate the proposed EfficientShip on Seaships [2] dataset. The experiments use Pytorch (1.11.0) library which is installed in Ubuntu 20.04. The model parameters are trained on an NVIDIA GeForce RTX 3090 GPU with 24 GB RAM. And the CPU is Intel(R) Xeon(R) Platinum 8255C with 45 GB RAM.

4.1 Dataset and Evaluation Metrics

The dataset we selected in this paper is SeaShips [2]. The dataset has 7000 images and includes six categories: bulk cargo carrier, container ship, fishing boat, general cargo ship, ore carrier, and passenger ship. Fig. 5 shows the appearance of different ships in SeaShips. The resolution of images is 1920 × 1080. All pictures in the dataset are selected from 5400 real-world video segments generated by 156 monitoring cameras in the coastline surveillance system. It covers targets of different backgrounds, scales, hull parts, illumination, occlusions and viewpoints. We randomly divide the dataset into a training set and a test set with proportion of 9:1 for the experiments followed by [35].

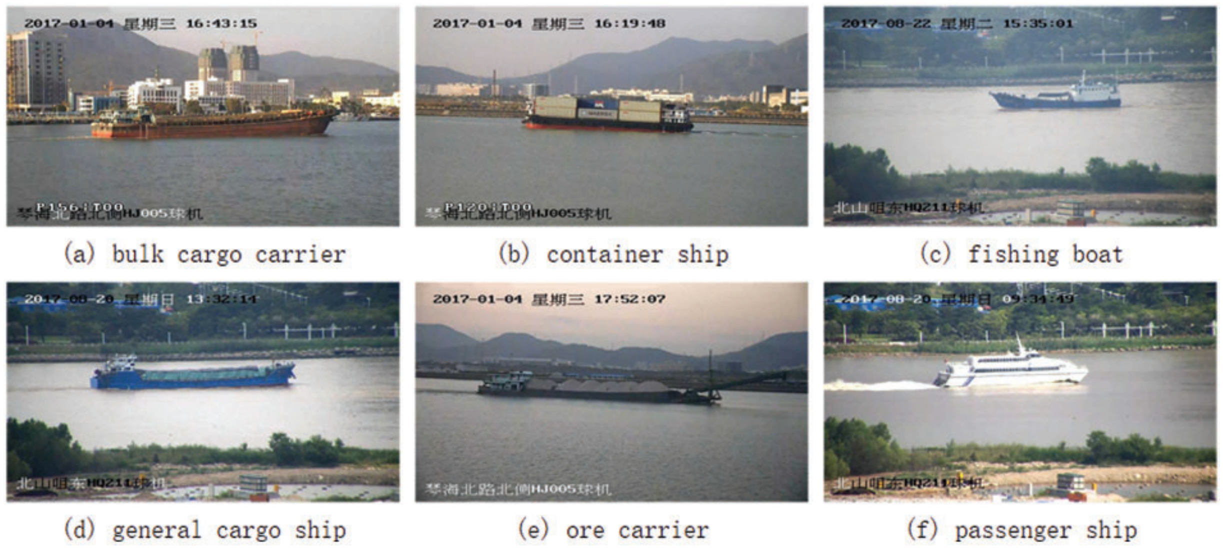

Figure 5: Illustration of different ship samples and their labels in the SeaShips dataset. (a) bulk cargo carrier; (b) container ship; (c) fishing boat; (d) general cargo ship; (e) ore carrier; (f) passenger ship

Experimental evaluation metrics include ship detection accuracy and runtime. The runtime is reported by fps, and the detection accuracy is evaluated by standard mAP which defined as

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{15}$$

where $K = 6$ for all ship categories in SeaShips.

4.2 Parameter Setting

PSDA. For PDA, we select 33 augmentation methods (with 40 adjustable parameters) for every original training image. There are 15 parameter variations for each adjustable parameter setting shown in Table 1. For one raw image, 600 new images can be augmented at this stage. Fig. 6 displays the augmentation results of methods RandomFog and ColorJitter(in brightness). We choose 24 augmentation algorithms at the stage of SDA which generates $24 * 15 = 360$ new images with spatial variation. The spatial parameter settings are listed in Table 2, and images generated by methods Affine(rotate) and Resize are illustrated in Fig. 7. We construct a total of 960 new images for each original training image in SeaShips [2] through PSDA.

DBOL & CROC. The method of DLA [38] is selected as the backbone of the first ship detection stage. We extend DLA through a 4-layer BiFPN [53] with 160 feature channels. We reduce the output FPN levels to 3 levels with strides 8-32. The model parameters in the first stage are trained with a long schedule that repetitively fine-tunes. The amount of object proposals is reduced to 128 in the target-detecting stage. For the second stage, the detection part of CascadeRcNN [27] is adopted for recognizing the proposals. We raise the positive IoU threshold value from 0.6 to 0.8 for the method of CascadeRcNN to reimburse the IoU distribution variation.

Table 1: Pixel level DA parameter settings

DA algorithm	Parmeter setting
AdvancedBlur	blur_limit=[1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57]
Blur	blur_limit=[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
CLAHE	clip_limit=[1, 1.4, 1.8, 2.2, 2.6, 3.0, 3.4, 3.8, 4.2, 4.6, 5, 5.4, 5.8, 6.2, 6.6]
ColorJitter(brightness)	brightness=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45]
ColorJitter(contrast)	contrast=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45]
ColorJitter(saturation)	saturation=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45]
Defocus	radius=[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
Downscale	scale_min=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45]
Emboss	alpha=(0.2, [0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65])
FancyPCA	alpha=[0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 1.0, 1.15, 1.3, 1.45, 1.6, 1.75, 1.9, 2.05, 2.3]
GaussNoise	var_limit=(10, [200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550])
GaussianBlur	blur_limit=(3, [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31])
GlassBlur	sigma=[0.3, 0.34, 0.38, 0.42, 0.46, 0.5, 0.54, 0.58, 0.62, 0.66, 0.7, 0.74, 0.78, 0.82, 0.86]
HueSaturationValue(hue_shift)	hue_shift_limit=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
HueSaturationValue(sat_shift)	sat_shift_limit=[-29, -25, -21, -17, -13, -9, -5, -1, 3, 7, 11, 15, 19, 23, 27]
HueSaturationValue(val_shift)	val_shift_limit=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
ISONoise	intensity=(0.1, [0.4, 0.44, 0.48, 0.52, 0.56, 0.6, 0.64, 0.68, 0.72, 0.76, 0.8, 0.84, 0.88, 0.92, 0.96])
ImageCompression	quality_lower=[1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43], quality_upper=[99, 96, 93, 90, 87, 84, 81, 78, 75, 72, 69, 66, 63, 60, 57]
MedianBlur	blur_limit=[3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]
MotionBlur	blur_limit=[3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]
MultiplicativeNoise	multiplier=[0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.2, 1.25, 1.3]
RGBShift(r_shift)	r_shift_limit=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
RGBShift(g_shift)	g_shift_limit=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
RGBShift(b_shift)	b_shift_limit=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
RandomBrightnessContrast (brightness)	brightness_limit=[-0.175, -0.15, -0.125, -0.1, -0.075, -0.05, -0.025, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2]
RandomBrightnessContrast (contrast)	contrast_limit=[-0.175, -0.15, -0.125, -0.1, -0.075, -0.05, -0.025, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2]

(Continued)

Table 1 (continued)

DA algorithm	Parmeter setting
RandomFog	fog_coef_lower=[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15], fog_coef_upper=[0.52, 0.54, 0.56, 0.58, 0.6, 0.62, 0.64, 0.66, 0.68, 0.7, 0.72, 0.74, 0.76, 0.78, 0.8]
RandomGamma	gamma_limit=([83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97], [103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117])
RandomRain	slant_lower=[-15, -13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13], slant_upper=[-8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
RandomShadow	num_shadows_lower=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], num_shadows_upper=[11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
RandomSnow	snow_point_lower=[0.05, 0.07, 0.09, 0.11, 013, 0.15, 0.17, 0.19, 0.21, 0.23, 0.25, 0.27, 0.29, 0.31, 0.33], snow_point_upper=[0.55, 0.57, 0.59, 0.61, 0.63, 0.65, 0.67, 0.69, 0.71, 0.73, 0.75, 0.77, 0.79, 0.81, 0.83]
RandomSunFlare	angle_lower=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45], angle_upper=[0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85]
RandomToneCurve	scale=[0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52]
RingingOvershoot	blur_limit=[5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61]
Sharpen	alpha=[0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45], [0.6, 0.625, 0.65, 0.675, 0.7, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95])
Solarize	threshold=[50, 55, 60, 65, 70, 75, 80, 85, 95, 100, 105, 110, 115, 120, 125]
Spatter	std=[0.1, 0.115, 0.13, 0.145, 0.16, 0.175, 0.19, 0.205, 0.22, 0.235, 0.25, 0.265, 0.28, 0.295, 0.305]
Superpixels	n_segments=[86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114]
UnsharpMask	blur_limit=[21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 137, 141, 151, 161]
ZoomBlur	max_factor=[1.01, 1.015, 1.02, 1.025, 1.03, 1.035, 1.04, 1.045, 1.05, 1.055, 1.06, 1.065, 1.07, 1.075, 1.08]

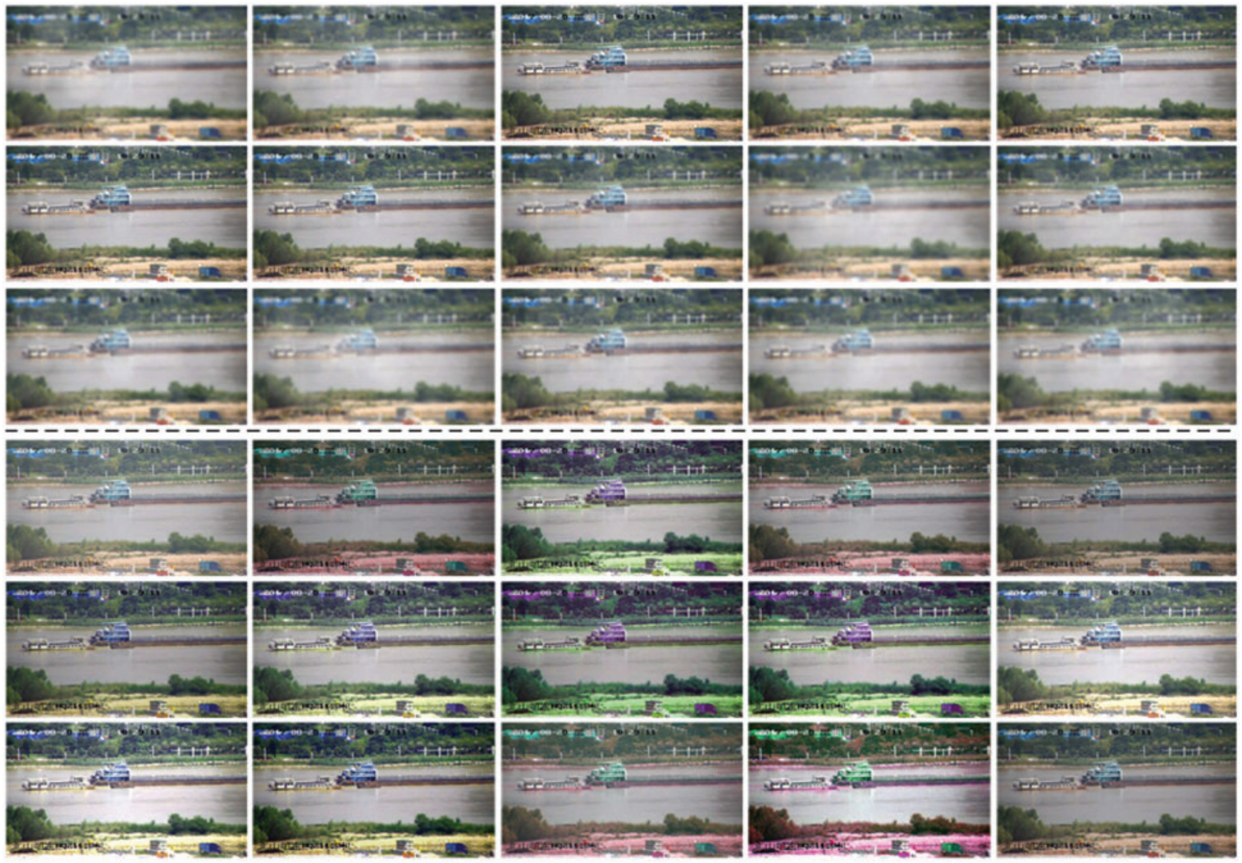
Figure 6: Illustration of pixel level DA. Upper: Augmentation with RandomFog; Under: Augmentation with ColorJitter(brightness)

Table 2: Space level DA parameter settings

DA algorithm	Parmeter setting
Affine(scale)	scale=[0.7, 0.72, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97]
Affine(translate)	translate_px=random.randint(0, 50)
Affine(rotate)	rotate=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
Affine(shear)	shear=[-13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15]
BBoxSafeRandomCrop	erosion_rate=[0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19, 0.21, 0.23, 0.25, 0.27, 0.29]
CenterCrop	height, width=margin-[5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47]
CropAndPad	px=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150]
ElasticTransform	sigma=[36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64]
GridDistortion	distort_limit=[-0.026, -0.022, -0.018, -0.014, -0.01, -0.006, -0.002, 0.002, 0.006, 0.01, 0.014, 0.018, 0.022, 0.026, 0.03]

(Continued)

Table 2 (continued)

DA algorithm	Parmeter setting
LongestMaxSize	max_size=[1640, 1660, 1680, 1700, 1720, 1740, 1760, 1780, 1800, 1820, 1840, 1860, 1880, 1880, 1900]
OpticalDistortion	distort_limit=[-0.0025, -0.002, -0.0015, -0.001, -0.0005, 0.0005, 0.001, 0.0015, 0.002, 0.0025, 0.003, 0.0035, 0.004, 0.0045, 0.005]
Perspective	scale=[0.054, 0.057, 0.06, 0.063, 0.066, 0.069, 0.072, 0.075, 0.078, 0.081, 0.084, 0.087, 0.09, 0.093, 0.096]
PiecewiseAffine	scale=[0.0125, 0.015, 0.0175, 0.02, 0.0225, 0.025, 0.0275, 0.03, 0.0325, 0.035, 0.0375, 0.04, 0.0425, 0.045, 0.0475]
PixelDropout	dropout_prob=[0.00125, 0.0025, 0.00375, 0.005, 0.00625, 0.0075, 0.00875, 0.01, 0.01125, 0.0125, 0.01375, 0.015, 0.01625, 0.0175, 0.01875]
RandomCrop	height=900, width=1600
RandomCropFromBorders	crop_left=0.1, crop_right=0.1, crop_top=0.1, crop_bottom=0.1
RandomResizedCrop	height=900, width=1600
RandomScale	scale_limit=0.1
RandomSizedBBoxSafeCrop	height=900, width=1600
RandomSizedCrop	min_max_height=[720, 1080], height=900, width=1600
Resize	height, width=margin-[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150]
Rotate	limit=10
SafeRotate	limit=10
ShiftScaleRotate	shift_limit=0.0625, scale_limit=0.1, rotate_limit=5

4.3 Results and Analysis

(I) Ablation Study

We design the different experiments on the modules of the proposed framework to find their effectiveness. We first select the EfficientShip with non-DA as a baseline. Then we add pixel-level and spatial-level DA separately on the basis of the ship detection. Finally, we test the whole hybrid ship detection framework which includes three complete steps. Details of the experimental results are presented in Table 3. We can observe that the basic EfficientShip with non-DA yields the lowest mAP value of 98.85%, and the baseline plus SDA can get a 0.43% boost. The baseline plus PDA yields a 0.62% improvement which shows PDA is much better than SDA. The whole proposed EfficientShip achieves a detection accuracy of 99.63%.

Fig. 8 shows the mAP comparison chart of different modules. It also indicates the changes in detection accuracy among various categories of the SeaShips dataset. Relatively, the bulk cargo carrier is the most recognizable object, while the passenger ship is the most difficult target to identify. After superimposing DA on the basis of two-stage detection, each category of detection accuracy is gradually approaching 100%.

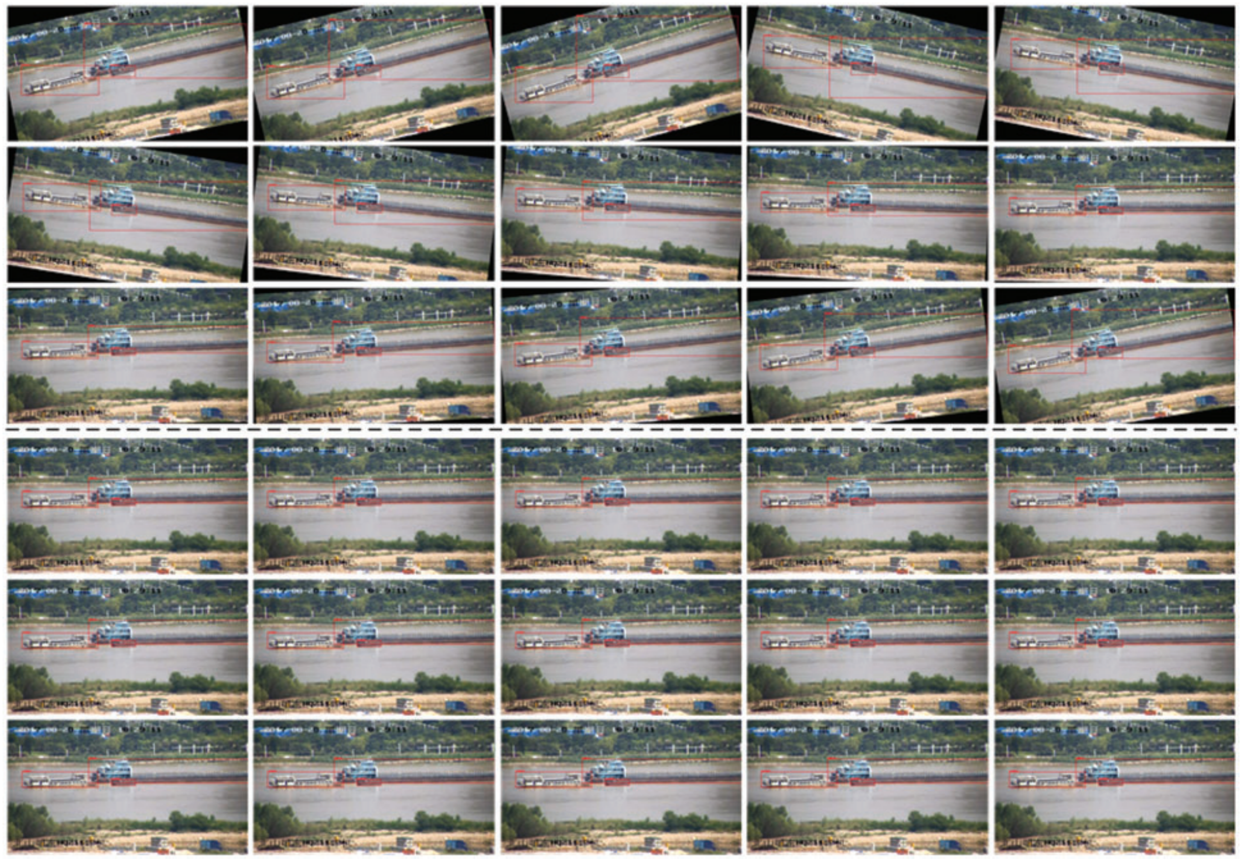

Figure 7: Illustration of space level DA. Upper: Augmentation with Affine(rotate); Under: Augmentation with PixelDropout

Table 3: Comparision of ship detection accuracy of different modules

Method	mAP (%)
EfficientShip (non-DA)	98.85
EfficientShip (PDA)	99.28
EfficientShip (SDA)	99.47
EfficiengShip (PSDA)	**99.63**

(II) Comparison to State-of-the-Art Approaches

We compare the proposed approach with 8 SOTA methods [2,3,31–35,43] from accuracy and efficiency of ship detection, as shown in Table 4. The data values of all SOTA algorithms are derived from their original papers. Although the algorithm speed is not comparable because of the difference in the platform on which the algorithm runs. However, it can be seen from Table 4 that the speeds of all methods meet the requirements of real-time application scenarios. Compared with the earliest sea ship detection algorithm [2], the accuracy of our method has improved detection accuracy by 16.63%. The accuracy of proposed algorithm is 99.63%, which has a 0.93% increase over the best SOTA-performing algorithm [35].

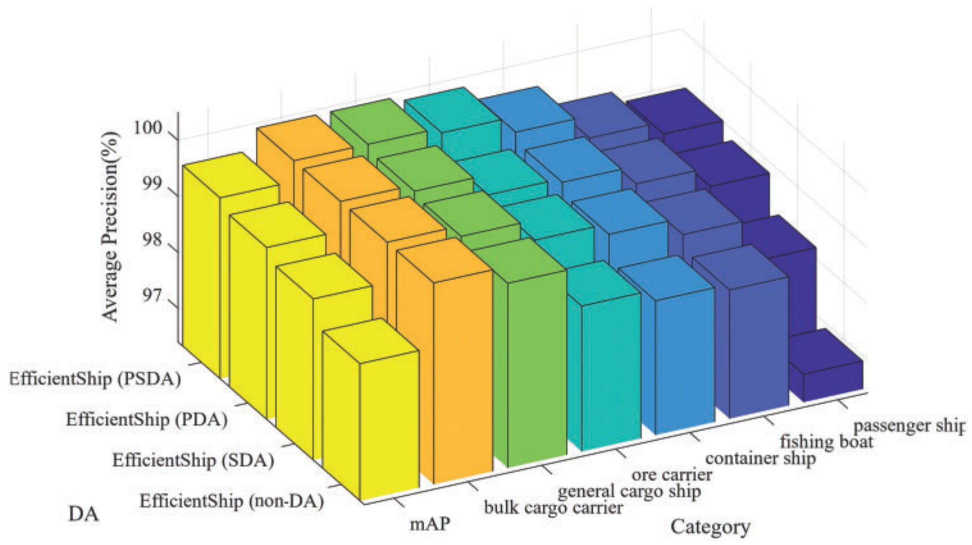

Figure 8: Comparison of AP curves of different modules: (a) EfficientShip (non-DA); (b) EfficientShip (PDA); (c) EfficientShip (SDA); (d) EfficeingShip (PSDA)

Table 4: Comparison of EfficientShip with SOTA

Method	mAP (%)	FPS
SeaShips [2]	83	83
SaliencyCNN [3]	87.4	49
eYOLO [31]	87.74	30
NSD-SSD [32]	89.3	45
ImprovedYOLOv3 [43]	90.58	37
RDSC [34]	94.6	68
EnhancedYOLO [33]	97	**135**
AE-YOLOv3 [35]	98.7	32
EfficientShip	**99.63**	45

5 Conclusions

Different from the traditional one-stage real-time ship detection methods, we fully utilized the latest real-time algorithms of object detection to construct a novel two-stage ship detection named EfficientShip. It includes DBOL, CROC, and PSDA. The DBOL is responsible for producing high-quality bounding boxes of the potential ship, and the CROC undertakes object recognition. We train the two stages jointly to boost the log-likelihood of actual objects. We also designed the PSDA to make further efforts of promoting the accuracy of target detection. Experiments on the dataset SeaShips show that the proposed EfficientShip has the highest ship detection accuracy among SOTA methods on the premise of achieving real-time performance. In the future, we will further verify the proposed algorithm on some new larger datasets, such as LS-SSDD-v1.0 and Official-SSDD [54].

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported by the Outstanding Youth Science and Technology Innovation Team Project of Colleges and Universities in Hubei Province (Grant No. T201923), Key Science and Technology Project of Jingmen (Grant Nos. 2021ZDYF024, 2022ZDYF019), LIAS Pioneering Partnerships Award, UK (Grant No. P202ED10), Data Science Enhancement Fund, UK (Grant No. P202RE237), and Cultivation Project of Jingchu University of Technology (Grant No. PY201904).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Huafeng Chen; data collection: Junxing Xue; analysis and interpretation of results: Huafeng Chen, Junxing Xue, Yudong Zhang; draft manuscript preparation: Huafeng Chen, Hanyun Wen, Yurong Hu, Yudong Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data can be download from http://www.lmars.whu.edu.cn/prof_web/shaozhenfeng/datasets/SeaShips(7000).zip.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Zhang, T., Zhang, X. (2019). High-speed ship detection in SAR images based on a grid convolutional neural network. *Remote Sensing, 11(10),* 1206.
2. 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.
3. Shao, Z., Wang, L., Wang, Z., Du, W., Wu, W. (2019). Saliency-aware convolution neural network for ship detection in surveillance video. *IEEE Transactions on Circuits and Systems for Video Technology, 30(3),* 781–794.
4. Tutsoy, O. (2021). Pharmacological, non-pharmacological policies and mutation: An artificial intelligence based multi-dimensional policy making algorithm for controlling the casualties of the pandemic diseases. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(12),* 9477–9488.
5. Zhang, T., Zhang, X., Ke, X., Liu, C., Xu, X. et al. (2021). HOG-ShipCLSNet: A novel deep learning network with hog feature fusion for SAR ship classification. *IEEE Transactions on Geoscience and Remote Sensing, 60,* 1–22.
6. Dai, W., Mao, Y., Yuan, R., Liu, Y., Pu, X. et al. (2020). A novel detector based on convolution neural networks for multiscale SAR ship detection in complex background. *Sensors, 20(9),* 2547.
7. Cao, C., Wu, J., Zeng, X., Feng, Z., Wang, T. et al. (2020). Research on airplane and ship detection of aerial remote sensing images based on convolutional neural network. *Sensors, 20(17),* 4696.
8. Zou, Z., Shi, Z., Guo, Y., Ye, J. (2019). Object detection in 20 years: A survey. arXiv preprint arXiv:1905.05055.

9. Rao, Y., Mu, H., Yang, Z., Zheng, W., Wang, F. et al. (2022). B-PesNet: Smoothly propagating semantics for robust and reliable multi-scale object detection for secure systems. *Computer Modeling in Engineering & Sciences, 132(3),* 1039–1054. https://doi.org/10.32604/cmes.2022.020331
10. Soviany, P., Ionescu, R. T. (2018). Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Piscataway, IEEE.
11. Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy.
12. Tian, Z., Shen, C., Chen, H., He, T. (2019). FCOS: Fully convolutional one-stage object detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Korea.
13. Zhou, X., Wang, D., Krähenbühl, P. (2019). Objects as points. arXiv preprint arXiv:1904.07850.
14. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S. Z. (2020). Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA.
15. Kim, K., Lee, H. S. (2020). Probabilistic anchor assignment with IoU prediction for object detection. *European Conference on Computer Vision*, Glasgow, UK, Springer.
16. Qiu, H., Ma, Y., Li, Z., Liu, S., Sun, J. (2020). BorderDet: Border feature for dense object detection. *European Conference on Computer Vision*, Glasgow, UK, Springer.
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA.
18. Redmon, J., Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA.
19. Redmon, J., Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.
20. Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.
21. Glenn, J., Alex, S., Jirka, B., NanoCode012, Ayush, C. et al. (2021). ultralytics/yolov5: v5.0-yolov5-p6 1280 models. https://github.com/ultralytics/yolov5
22. Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA.
23. He, K., Zhang, X., Ren, S., Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9),* 1904–1916.
24. Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile.
25. Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems, 28,* 1–9.
26. Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B. et al. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA.
27. Cai, Z., Vasconcelos, N. (2018). Cascade R-CNN: Delving into high quality object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, USA.
28. Lu, X., Li, B., Yue, Y., Li, Q., Yan, J. (2019). Grid R-CNN. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, USA.
29. Zhou, X., Koltun, V., Krähenbühl, P. (2021). Probabilistic two-stage detection. arXiv preprint arXiv:2103.07461.

30. Nie, X., Yang, M., Liu, R. W. (2019). Deep neural network-based robust ship detection under different weather conditions. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, IEEE.
31. 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.
32. Sun, J., Xu, Z., Liang, S. (2021). NSD-SSD: A novel real-time ship detector based on convolutional neural network in surveillance video. *Computational Intelligence and Neuroscience, 2021,* 1–16.
33. Li, H., Deng, L., Yang, C., Liu, J., Gu, Z. (2021). Enhanced YOLO v3 tiny network for real-time ship detection from visual image. *IEEE Access, 9,* 16692–16706.
34. Liu, T., Pang, B., Zhang, L., Yang, W., Sun, X. (2021). Sea surface object detection algorithm based on YOLO v4 fused with reverse depthwise separable convolution (RDSC) for USV. *Journal of Marine Science and Engineering, 9(7),* 753.
35. 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,* 1–11.
36. Zhang, M., Rong, X., Yu, X. (2022). Light-SDNet: A lightweight CNN architecture for ship detection. *IEEE Access, 10,* 86647–86662.
37. Zhou, S., Yin, J. (2022). YOLO-ship: A visible light ship detection method. *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, Guangzhou, China, IEEE.
38. Yu, F., Wang, D., Shelhamer, E., Darrell, T. (2018). Deep layer aggregation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, USA.
39. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA.
40. Zhang, T., Zhang, X., Ke, X., Zhan, X., Shi, J. et al. (2020). LS-SSDD-v1.0: A deep learning dataset dedicated to small ship detection from large-scale sentinel-1 SAR images. *Remote Sensing, 12(18),* 2997.
41. Chen, Z., Li, B., Tian, L. F., Chao, D. (2017). Automatic detection and tracking of ship based on mean shift in corrected video sequences. *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, Chengdu, China, IEEE.
42. Zhang, Y., Li, Q. Z., Zang, F. N. (2017). Ship detection for visual maritime surveillance from non-stationary platforms. *Ocean Engineering, 141,* 53–63.
43. Liu, T., Pang, B., Ai, S., Sun, X. (2020). Study on visual detection algorithm of sea surface targets based on improved YOLOv3. *Sensors, 20(24),* 7263.
44. Shorten, C., Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6(1),* 60.
45. Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M. et al. (2020). Albumentations: Fast and flexible image augmentations. *Information, 11(2),* 125.
46. Taylor, L., Nitschke, G. (2018). Improving deep learning with generic data augmentation. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Bengaluru, India, IEEE.
47. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y. (2020). Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34. New York, USA.
48. Xu, X., Zhang, X., Zhang, T., Yang, Z., Shi, J. et al. (2022). Shadow-background-noise 3D spatial decomposition using sparse low-rank gaussian properties for video-SAR moving target shadow enhancement. *IEEE Geoscience and Remote Sensing Letters, 19,* 1–5.
49. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*. https://openreview.net/forum?id=rJzIBfZAb

50. Huang, S. W., Lin, C. T., Chen, S. P., Wu, Y. Y., Hsu, P. H. et al. (2018). AugGAN: Cross domain adaptation with GAN-based data augmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany.

51. Wang, S. H., Govindaraj, V. V., Górriz, J. M., Zhang, X., Zhang, Y. -D. (2021). COVID-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. *Information Fusion, 67,* 208–229.

52. Zhang, Y., Zhang, X., Zhu, W. (2021). ANC: Attention network for COVID-19 explainable diagnosis based on convolutional block attention module. *Computer Modeling in Engineering & Sciences, 127(3),* 1037–1058. https://doi.org/10.32604/cmes.2021.015807

53. 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, USA.

54. Zhang, T., Zhang, X., Li, J., Xu, X., Wang, B. et al. (2021). SAR ship detection dataset (SSDD): Official release and comprehensive data analysis. *Remote Sensing, 13(18),* 3690.