EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems   Research Article   EAI.EU

Transformer Based Ship Detector: An Improvement on Feature Map and Tiny Training Set

Duc-Dat Ngo[1], Van-Linh Vo[1], My-Ha Le[1], Hoc-Phan [1], Manh-Hung Nguyen[1,*]

[1]HCMC University of Technology and Education- Faculty of Electrical and Electronics Engineering- Ho Chi Minh City (7000)-VietNam.

Abstract

The exponential increment of commodity exchange has raised the need for maritime border security in recent years. One of the most critical tasks for naval border security is ship detection inside and outside the territorial sea. Conventionally, the task requires a substantial human workload. Fortunately, with the rapid growth of the digital camera and deep-learning technique, computer programs can handle object detection tasks well enough to replace human labor. Therefore, this paper studies how to apply recent state-of-the-art deep-learning networks to the ship detection task. We found that with a suitable number of object queries, the Deformable-DETR method will improve the performance compared to the state-of-the-art ship detector. Moreover, comprehensive experiments on different scale datasets prove that the technique can significantly improve the results when the training sample is limited. Last but not least, feature maps given by the method will focus well on key objects in the image.

Received on 29 07 2024; accepted on 18 10 2024; published on 06 11 2024

Keywords: Maritime border security; Deformable DETR; Object detection; Hyper-parameter

doi:10.4108/eetinis.v12i1.6794

1. Introduction

The global commodity exchange has surged in recent years, leading to an increase in waterway transport services. While this growth benefits the economy, it also presents significant challenges in protecting national water borders. As a result, ship detection has become a crucial aspect of national security. Accurately detecting ships approaching the shore and thoroughly verifying their legality are essential tasks. Typically, data from coastal surveillance cameras is used to monitor passing ships, and human inspectors review the images. However, this manual process requires considerable human effort and is prone to errors due to various distractions. Therefore, developing an automated ship detection system can reduce costs and improve monitoring efficiency, particularly for developing countries. Nevertheless, there are still practical challenges that the ship detection model must overcome to achieve optimal performance, such as data imbalance, lighting conditions, occlusion, missing object parts, and size diversity, as shown in Figure 1.

Ship detection and classification from images are well-known applications in computer vision, traditionally addressed by object detection techniques. While earlier deep learning models like Region-based Convolutional Neural Network (R-CNN) [1] and FastRCNN [2] surpassed hand-crafted methods, their sluggish performance hindered real-world applications. One-stage detectors such as Single Shot Setector (SSD) [3] and You Only Look Once [4], though faster, rely on cumbersome anchor boxes and post-processing steps. Recent advancements like feature pyramid networks (FPN) [5] and decoupled heads [6] have partially mitigated these issues, but not all. To improve the performance of ship detectors, many researchers try to customize these detectors, such as network architecture [7–11], feature fusion [12, 13], or feature selection loss [14].

While many promising results have been reported, training a good detector is still an open question. Several hyper-parameters, such as anchor-box generation or non-maximum suppression procedure, should be

*Corresponding author. Email: hungnm@hcmute.edu.vn

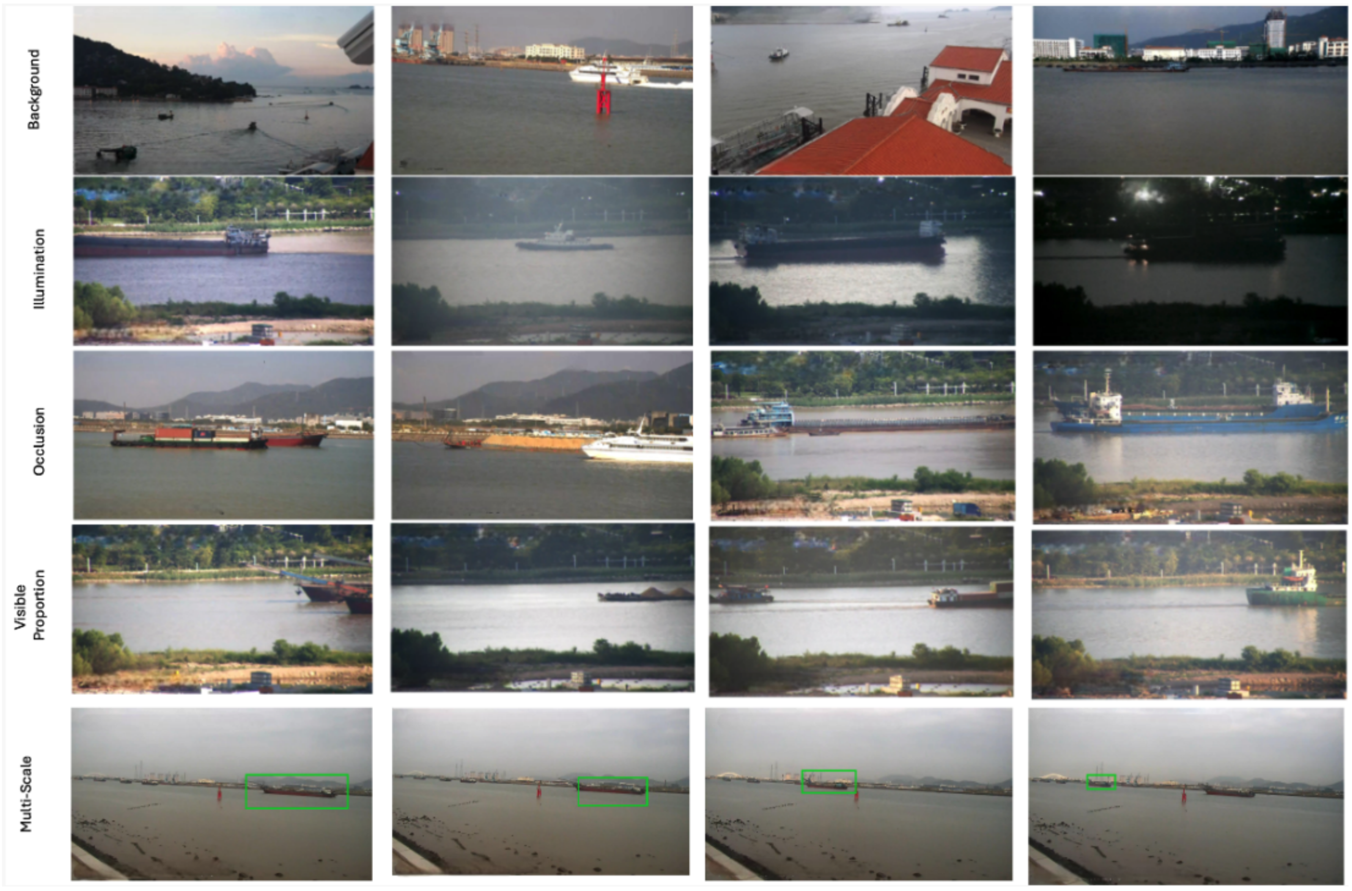

Figure 1. Challenges of ship detection.

carefully designed to ensure the success of training processes. Motivated by the observation, DETR [15, 16] provides end-to-end object detection with transformers. This method forces unique predictions via bipartite matching; hence a fixed small set of learned object queries may directly output the final set of predictions in parallel. Yani_2022 [17] had try to applied DETR to ship detection via distillation learning. Here, a teacher provides a prediction to train a student model. However, the performance is not comparable to CNN-based methods [12, 13] in the student model.

In our work, we revisit how to use DETR to train a ship-detector. As shown in Figure 2, each learnable query will provide one object detection result on the input image. Hence, too many queries will provide false detection, especially when the number of objects is sparse, like ships. Hence, a suitable number of object queries may directly affect the detection result.

We prove this viewpoint by comprehensive experiments on a well-known large-scale ship dataset [18]. First, we follow guidance from [12–14, 18] to prepare training and testing datasets and train our model. The experiment aims to compare our method with state-of-the-art methods. Second, we re-used the testing set but reduced the training set as in [14, 17, 19]. This experiment helps to evaluate the performance of our method when the training samples are limited. In addition, an ablation study is applied to evaluate the effect of the number of object queries. Last but not least, feature visualization explains why our method can work more robustly than CNN-based methods. In summary, our contributions are as bellowed:

(i) Our detector demonstrates performance that is on par with state-of-the-art (SoTA) methods on well-known benchmarks. It has shown comparable accuracy, precision, and robustness through rigorous testing and evaluation, effectively matching the results of leading models in the field.

(ii) When the number of training samples is limited, our method demonstrates a significant performance improvement. By leveraging advanced techniques and efficient learning algorithms, our approach maximizes the utility of available data, ensuring robust and accurate results even with constrained training sets. This capability highlights the strength and adaptability of our method, making it particularly valuable in scenarios where data acquisition is challenging or costly. Consequently, our method is reliable for achieving high performance in data-scarce environments.

(iii) Feature analysis provides insights into why our method operates robustly compared to traditional CNN-based approaches. By examining the features extracted by our model, we can see how it effectively captures and utilizes relevant patterns in the data, leading to improved performance.

2. Releated Work

2.1. Object Detection

Recently, deep learning has been considered an advantageous solution for computer vision tasks. Deep learning-based object detection can be categorized into two branches: region proposal-based methods and regression/classification-based methods. Region proposal-based methods usually have two steps. The first step is finding areas where an object is likely to exist. The second step is to perform the classification for that object. Therefore, this method is called the two-stage object detection method. Some well-known methods included R-CNN [1], Fast R-CNN [20], Faster R-CNN [21], FPN [5], etc.

On the other hand, regression/classification-based object detection will predict an object's location through regression and the object's label through classification. Instead of splitting the object detection into two steps, this technique uses a convolutional neural network to predict location information (regression results) and label information (classification results). Since only one CNN network is used, these are considered single-stage methods. Well-known methods of this approach are YOLO [4], SSD [3], etc. The single-stage methods have a much faster processing speed than the two-stage methods. For example, Fast R-CNN [20] can handle 0.5 frames per second, whereas the first version of YOLO can handle 45 frames per second, or SSD can handle 58 frames per second. Therefore, single-stage methods are often used in practice.

There is a trade-off between accuracy and inference time in object detection. While Fast R-CNN [20] achieved an accuracy of 0.7 mAP, it can process very slowly. The first version of YOLO called YOLOv1 [4], only achieved an accuracy of 0.63 mAP, but it is much faster than FastRCNN. For this reason, many researchers are trying to increase the accuracy of the YOLO structure. Recently, later versions of the YOLO have been discussed to overcome the disadvantages of the original YOLO model. For example, in YOLOv1, each cell can only predict one object at most. For cases where many objects are in the same cell, YOLOv1 may not have a good result. Moreover, YOLOv1 predicts the position of objects as a bounding box directly, and the objective functions of YOLOv1 [4] do not have a separate evaluation between the error caused by bounding box widths and heights.

Instead of predicting the absolute bounding box, YOLOv2 [22] considers anchor boxes of the main component to predict each relative bounding box. In detail, instead of indicating the position of a box on an image, the CNN network predicts the offset between the bounding box and predefined anchor boxes. Predicting the offset is much easier than predicting the box coordinates. If more or more anchor contours surround the object, it is possible to define the anchor contours that overlap with the labeled object contour.

One of the key challenges in object detection is handling the scale issue. Objects appear smaller when they are far from the camera and larger when they are close. The Feature Pyramid Network (FPN) [5] addresses this scale challenge by introducing pyramid features, where the model extracts features at different scales, similar to the layers of a pyramid. This approach allows the detection of objects at varying distances. However, FPN's drawbacks include high memory consumption, reduced detection speed, and increased model complexity. To extract multi-scale features without these burdens, the YOLOF model [23] was developed. Instead of employing the multi-input, multi-output architecture of FPN, YOLOF utilizes a single-input, single-output encoding architecture based on Dilated CNN [24]. This design enables YOLOF to consume less memory while maintaining accuracy comparable to FPN.

One of the challenges in object detection is determining the number and size of anchor boxes, especially due to varying distances between the object and the camera. Anchor boxes are usually determined using the k-means clustering algorithm, but this process still requires human intervention and is not fully automated for end-to-end training. YOLOX, a notable model, has made modifications to eliminate the anchor box selection process. It does this by replacing the coupled head with a decoupled head and predicting only one feature box for each location on the feature map. This effectively removes the need for anchors. Additionally, by separating the regression and classification tasks into two distinct branches, the model can converge more effectively.

Inspired by the success of transformers in natural language processing (NLP), researchers have begun applying transformer concepts to object detection. Transformers represent a significant departure from convolutional neural networks (CNNs). Although their use in vision tasks is still in its early stages, transformers have shown promising potential to replace convolutions. State-of-the-art transformer-based detectors have achieved impressive results on various object detection datasets, though they typically require more parameters than convolutional models. Recently, several transformer-based methods have been developed for object detection, including Vision-Transformer-Detection (ViTDET) [25], DETR [15, 16, 26], and Shift-Window-Transformer (Swin) [27]. Among these, DETR [15] stands out as one of the early transformer-based methods to provide competitive results compared to CNN-based detectors. Additionally, DETR operates more closely as an end-to-end training method than other CNN-based approaches.

2.2. Ship Detection

In the field of ship detection, various customizations have been made to popular detection frameworks like SSD (Single Shot Multibox Detector) [3] and YOLO (You Only Look Once) [4] to improve their performance. For example, Liu et al. [7] enhanced the SSD framework by adding a VGG backbone, which improved the detection of small objects. They also introduced a local attention network and a merge module to integrate features from different scales, leading to a significant improvement in accuracy. On the other hand, the "Cross-level Attention and Ratio Consistency Network" (CARC) [19] uses YOLO with a ResNet-34 backbone and incorporates cross-level attention modules to extract multi-scale features for enhanced detection capabilities. However, traditional frameworks may struggle with effectively handling scale variations, which can impact the accuracy across different object sizes.

A better backbone can significantly enhance accuracy. Consequently, researchers such as Cui [9], Liu [12], and Li [8] have based their ship detection models on YOLOv3. Their modifications focus on model customization, incorporating attention modules to detect targets at different scales. Despite these advancements, challenges persist in accurately localizing and classifying small or occluded objects.

Recent advancements continue to refine ship detection capabilities, incorporating models such as YOLOv4 (Zhang_2021 [10], Han_2021 [28]) and YOLOv5 (SDNet_2022 [11]) with simplified networks and attention mechanisms. Subsequently, Zhang_2022 [13] and VIB_2023 [14] have refined YOLOX [29], a lightweight method for feature fusion to address inconsistencies in feature map scales. While methods like those proposed by Zhang_2021 [10], Han_2021 [28], SDNet_2022 [11], and Zhang_2022 [13] focusing on feature fusion modules to enhance feature learning, VIB_2023 [14] introduces a feature selection loss to enforce the model to learn sparse and discriminative features, helping the feature map focus more on the detected objects.

Not only CNN-based frameworks but also DETR can be used to detect ships. Yani et al. [17] utilize DETR [15] with distillation learning for ship detection. First, a teacher model is trained using the DETR method. Then, distillation loss and Hungarian loss are applied to train a lightweight DETR student model. Unlike Han_2021 [28] or SDNet_2022 [11], which use a default setting on a large-scale dataset [30] to train a detector, Yani [17] defined a new setting where 50% of the data is used for testing. In this scenario, DETR outperforms CNN-based detectors. However, the distilled model does not maintain the same high accuracy as the original DETR model.

3. Methodology

3.1. DETR Architecture

Model Description. The DETR model comprises a feature extractor, an encoder, and a decoder, as depicted in Figure 2. The feature extractor is a CNN backbone that extracts high-level information from an image; a 2D sinusoidal positional encoding also helps encode position information for each pixel. Image features and positional features are concatenated and fed into the transformer-based encoder. The encoder consists of multiple stacked multi-head self-attention layers. Features from the encoder are then passed to the transformer-based decoder. The decoder also takes several learnable object queries as input. These queries serve as a latent of suitable positions on the image. Given one query, the decoder will use a feature from the encoder to predict if there is any object at the positional query.

In the feature extractor, given an input image with a $H_0 \times W_0 \times 3$ tensor, the CNN backbone generates a feature map of size $H \times W \times C$. According to [15], $C = 2048$, $H = H_0/32$, and $W = W_0/32$. A 1x1 convolution layer reduces the feature channels from C to d ($d < C$), creating a new feature map of size $H \times W \times d$. Because The transformer-based encoder requires a 2D input, the feature map is resized to a $d \times HW$ matrix. Rows of the matrix are named tokens, each token being a d-dimensional vector.

The encoder includes many stacked multi-head self-attention layers. Each multi-head self-attention block consists of several self-attention modules. In a multi-head self-attention module, features extracted from self-attention modules are concatenated and projected to the output via a linear projection. Three individual linear projections extract a tuple ($key, value, query$) in a self-attention module. The similarities between key and $query$ serve as attention factors among $value$, fusing $value$ into self-attention features.

The decoder takes new features from the encoder and learnable object queries. Features from the encoder represent image information at every position, while queries serve as learnable positional encodings, questioning whether there is an object at a specific location. The decoder uses the image information

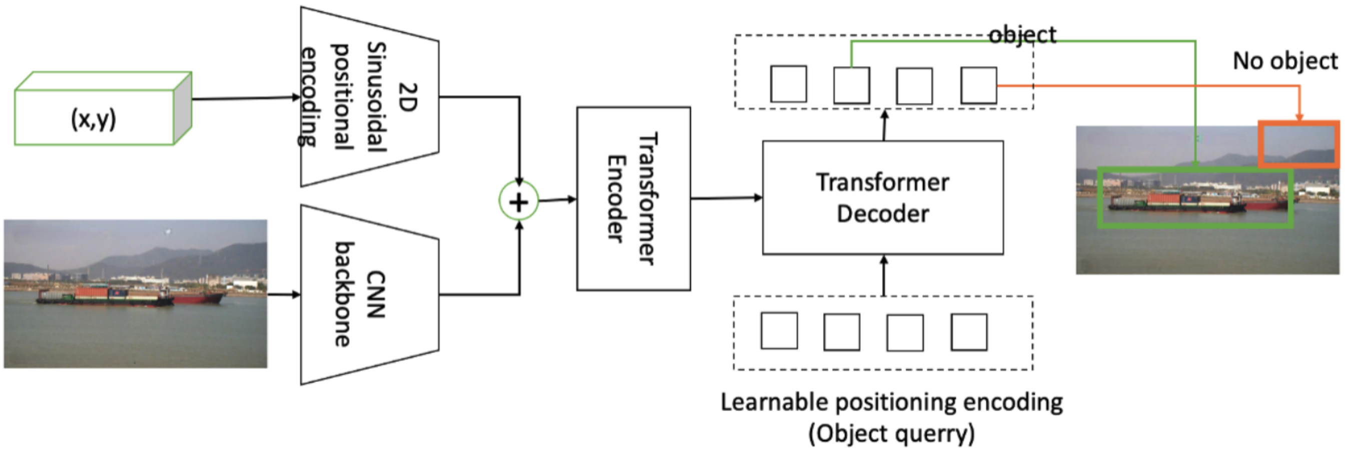
(x,y)
2D Sinusoidal positional encoding
CNN backbone
+
Transformer Encoder
Transformer Decoder
object
No object
Learnable positioning encoding (Object querry)

Figure 2. DETR Architecture

to determine if the encoded location contains any class. These queries are learnable from the dataset. The architecture of the encoder and decoder modules is shown in Figure 2. The outputs of the decoder module are fed into a feedforward neural network to predict the object's position and category. These outputs correspond to each object query learned in the decoder block. If an object query encodes no object at a position, the classification head result is "No Object".

Loss Function. The network returns a set of N detection results corresponding to N object queries. Each result is a tuple that includes $(class, box)$ and represents one and only one object without duplication. Therefore, a matching process is needed to map one prediction to one ground truth. Because the number of ground truth objects in an image is smaller than the number of object query N, a set $\varnothing$ of patches cropped randomly from the background is used as extra ground truth. Therefore, "background" objects with an arbitrary position and the label "No Object" help to balance the numbers of actual labeled locations and predictions. Denote σ is a matching solution; the optimal matching $\hat{\sigma}$ is the solution of an optimization process as in Equation 1. Where y_i is a ground truth of a box that includes class label and bounding box label (c_i, b_i); and $\hat{y}_i$ is a prediction result. This optimal assignment is solved by the Hungarian algorithm [31]. Note that this cost is not calculated on each object but as a collective combination of N objects generated for each image.

$$\hat{\sigma} = \underset{\sigma \in \Xi_N}{\operatorname{argmin}} \sum_{i}^{N} L_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right) \tag{1}$$

A good model can accurately predict objects and bounding boxes with higher overlapping contours. Therefore, the objective function $L_m\left(y_i, \hat{y}_{\sigma(i)}\right)$ should include these criterions as Equation 2.

$$L_m\left(y_i, \hat{y}_{\sigma(i)}\right) = -1_{\{c_i \neq \varnothing\}} \log \hat{p}_{\sigma(i)}\left(c_i\right) + 1_{\{c_i \neq \varnothing\}} L_{box}\left(b_i, \hat{b}_{\sigma(i)}\right) \tag{2}$$

The first term of Equation 2 guides the model to predict the object's category accurately. The second term helps to predict better bounding boxes. Here $c_i \neq \varnothing$ refers to detecting objects that are not dummy objects. Because each ground truth object is detected once, the equation 2 is applied to all ground truth objects in the image.

The optimal matching $\hat{\sigma}$ is estimated on a matching that includes ground truth objects and dumpy objects (defined by $\varnothing$). Because the boxes of dumpy objects are meaningless, the box loss is only valid if the box is not in the $\varnothing$ set. In contrast, the dumpy objects should be predicted as "No Object" correctly. Therefore, the classification loss should include the $\varnothing$ set. Motivated by the observation, the Hungary loss (L_H) [31] in Equation 3 is used to train the model.

$$L_H(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\sigma(i)}\left(c_i\right) + 1_{\{c_i \neq \varnothing\}} L_{\text{box}}\left(b_i, \hat{b}_{\sigma(i)}\right)\right]. \tag{3}$$

According to the literature reviews, influence scaling is critical in bounding box estimation. For example, a bounding box of a large object would have a width of 0.2, while a bounding box of a small object would have a width of 0.02. If the conventional Euler distance is used to measure the bounding box loss, the model may be too biased towards large objects and ignore small objects. Therefore, the Generalized Intersection over Union (GIOU) [32] loss function is introduced to calculate the bounding box loss together with the L1-norm [33] loss function. Denote $\lambda_{iou}, \lambda_{L1} \in R$ are hyper-parameters that control a learning process for GIOU loss and L1-norm loss; the formula of the boundary loss is in Equation 4.

$$L_{\text{box}}\left(b_i, \hat{b}_{\sigma(i)}\right) = \lambda_{\text{iou}} L_{\text{iou}}\left(b_i, \hat{b}_{\sigma(i)}\right) + \lambda_{L1}\left\|b_i - \hat{b}_{\sigma(i)}\right\|_1 \tag{4}$$

3.2. Deformable DETR

Deformable Attention block. In the DETR model, the attention block transmits almost uniform attention weights to all pixels in the feature map. Thus, training requires a longer time. Using a deformable mechanism, the custom attention block only samples a small set of crucial points around a reference point, regardless of the feature map size in the spatial domain. This design allows for a reduction in training time. Besides, the computational complexity and needed memory are reduced. Figure 3 illustrates how deformable convolution is integrated into a multi-head self-attention layer, and the equation 5 is the model representation of the layer.

$$\text{DeformAttn}\left(z_q, p_q, x\right) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} . W'_m x\left(p_q + \Delta p_{mqk}\right)\right] \tag{5}$$

where,

- $x \in R^{C\times H\times W}$ is an input feature map.
- M is the sum of interactions, where $m \in [1, M]$.
- K is the total number of standard keys sampled (K < HW), index $k \in [1, K]$.
- $W'_m x$ is a linear projection of the input feature map. W'_m extract *value* of a self-attention layer.
- Δp_{mqk} represents the sampling deviation of the k^{th} sampling point in the m^{th} attention head. Given a query z_q, a linear projection extracts a $2 \times (M * K)$ to represent K offset vectors for M attention head.
- A_{mqk} is the attention map of the k^{th} sampling point in the m^{th} attention output. Given a query z_q, a linear projection extracts a $M \times K$ tensor as an attention map A_{mqk}. The softmax function is applied on every head to ensure $\sum_{k=1}^{K} A_{mqk} = 1$.
- The term $\left[\sum_{k=1}^{K} A_{mqk} . W'_m x\left(p_q + \Delta p_{mqk}\right)\right]$ represents the interaction among sampling values to encode a new feature. The new feature is the output of a self-attention layer. Concatenating several self-attention features and then projecting the feature to output, we have a multi-head self-attention.

Multi-scale Deformable Attention Module. The multi-scale technique is commonly used to detect objects of different sizes. Feature maps at different scale levels detect objects from different distances. Multi-scaling techniques are integrated into a Deformable Attention block as Equation 6.

$$\text{MSDeformAttn}\left(z_q, \hat{p}_q, \left\{x^l\right\}_{l=1}^{L}\right) = \sum_{m=1}^{M} W_m \left[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlqk} \cdot W'_m x^l\left(\varnothing_l\left(\hat{p}_q\right) + \Delta p_{mlqk}\right)\right] \tag{6}$$

where,

- L is the input feature level, index $l \in [1, L]$
- $\{x^l\}_{l=1}^{L}$ is the feature map at different scales; with $x^l \in R^{(C\times H_l \times W_l)}$
- Δp_{mlqk} represents the sampling deviation of the k^{th} sampling point at the l^{th} feature level and the m^{th} attention head.
- $\hat{p}_q \in [0; 1]^2$ is the normalized coordinates. The upper left point is (0;0); and the lower right point is (1;1).
- The function $\varnothing_l(\hat{p}_q)$ will re-scale the normalized coordinates $\hat{p}_q$ to the input feature map of level l.

4. Experiment

4.1. Dataset Description

In order to demonstrate the effectiveness of the Deformable DETR framework for ship detection, we utilize a widely recognized large-scale dataset referenced in the work of [18]. This dataset originates from a comprehensive video monitoring system situated around Hengqin Island in Zhuhai City, China. It comprises a diverse range of ship images captured under different sea conditions, collected from 6:00 am to 8:00 pm daily.

For a fair comparison, we follow the setup outlined in [18] for preparing the training and testing sets. This approach is widely used in various research works such as Liu (2020) [12], Liu (2022) [7], Han (2021) [28], SDNet (2022) [11], and VIB (2023) [14], and has become a well-known benchmark. Under this setup, 80% of the dataset is used for training, and the remaining 20% is used for testing. The training and testing datasets are denoted as D_1^{Train} and D_1^{Test} respectively.

Some works (Biaohua_2022 [19] and Yani_2022 [17]) prepare a challenging scenario where 50% dataset is used for training and 50% dataset is used for testing. In this scenario, the training and testing datasets are denoted as D_2^{Train} and D_2^{Test} in our paper. Furthermore,

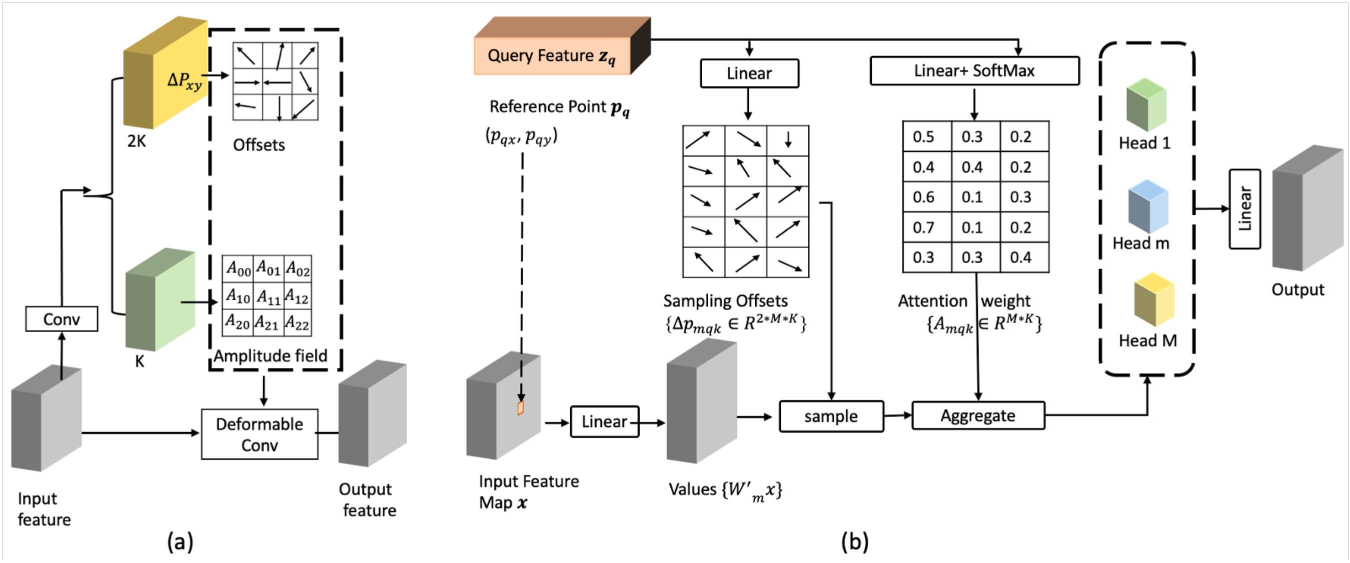

Figure 3. Illustration of the Deformable Attention module [16]. (a) The concept of deformable convolution. (b) Deformable convolution in multi-head attention.

Table 1. Performance comparisons of Deformable DETR with the number of queries. The best results are marked in **bold**.

Deformable DETR												
Class	300 query			200 query			100 query			50 query		
	dets	recall	AP	dets	recall	AP	dets	recall	AP	dets	recall	AP
fishing boat	204449	0,913	0,798	208817	0,985	0,970	119774	**0,986**	**0,969**	96577	0,972	0,949
container ship	6300	0,989	0,894	11460	0,995	0,995	31083	0,995	0,989	18949	**0,998**	**0,997**
ore carrier	53520	0,987	0,923	49362	**0,997**	**0,992**	62122	0,996	0,989	18949	0,993	0,987
bulk cargo carrier	45541	0,985	0,912	47754	0,996	**0,992**	46490	**0,997**	0,986	76230	0,996	**0,989**
passenger ship	22498	0,968	0,610	16199	0,970	**0,957**	14866	0,968	0,936	20043	**0,972**	0,926
general cargo ship	17692	0,987	0,930	16408	0,995	**0,992**	75665	0,995	0,991	98537	**0,996**	**0,990**
mAP			0.844			**0.982**			0.977			0.973

to evaluate the performance on limited datasets, VIB_2023 [14] introduced S_1, S_2, and S_3 subsets which are randomly selected from D_2^{Train}. These datasets contain 30%, 70%, and 100% of D_2^{Train} , respectively. These subsets enable thorough investigation into the scalability and adaptability of proposed methods across varying dataset sizes and training conditions.

Our experiment uses AdamW optimizer, learning rate = 0.0001, weight decay = 0.0001, n_epoch=200, batch_size=8, $\lambda_{iou} = 2.0$, and $\lambda_{L1} = 5.0$. We use the reduce-mean operator on batch data and the reduce-sum operator on prediction output. The mAP is used to select the best model. These experiments are implemented based on the mmdetection library [34].

4.2. Hyper-parameter selection

As discussed in Section 3, the number of queries can affect the number of outputs of a DETR-based detector. Therefore, this section tries to select the most suitable parameter to control the model. We compare the performance when $n_{queries}$ receives a vault in the $[300, 200, 100, 50]$ list. The D_2^{Train} and D_2^{Test} are selected as the training and testing dataset to ensure a challenging setting. It means 50% of the dataset is used for testing.

The findings are presented in Table 1. In mmdetection, the default value for the number of queries ($n_{queries}$) is 300. It is observed that using the default setting leads to an increased number of detections (dets). For example, the number of detections for fishing boats is 204449. This tendency results in a decrease in average precision (AP) to 0.798, while the recall increases to 0.913. Additionally, the higher number of detections for fishing boats can be attributed to the higher frequency of the corresponding label in the dataset. By reducing $n_{queries}$, the bias in detections is mitigated. Specifically, when $n_{queries}$ are set to 200, 100, and 50, the detections for fishing boats decrease to 208817, 119774, and 96577 respectively. Furthermore, this reduction in queries minimizes the bias in detections across different ship categories. With $n_{queries} = 300$, the lowest number

Table 2. Performance comparisons of Deformable DETR given by various learning rates. The best results are marked in **bold**.

	Lr=10^{-3}			Lr=10^{-4}			Lr=10^{-5}		
	Dets	recall	AP	Dets	recall	AP	Dets	recall	AP
fishing boat	75600	0.122	0.001	24874	**0.971**	**0.912**	15869	0.986	0.877
container ship	12600	0.223	0.007	9193	**0.993**	**0.986**	17028	0.988	0.968
ore carrier	12600	0.254	0.005	33544	**0.994**	0.965	24892	0.988	0.919
bulk cargo carrier	12600	0.401	0.016	35317	**0.995**	**0.949**	21334	0.997	0.906
passenger ship	14000	0.211	0.001	14543	**0.952**	0.861	42789	0.982	0.760
general cargo ship	12600	0.361	0.009	22529	**0.960**	**0.969**	18088	0.994	0.916
mAP			0.006			**0.941**			0.891

of detections is for container ships (6300). However, when $n_{queries} = 50$, the lowest number of detections is around 18949, and the detections for container ships, ore carriers, and passenger ships are quite similar.

Besides the $n_{queries}$, the learning rate (Lr) is another important factor that affects the system's performance. Default, Lr is set as 10e-4. In the revised version, we have modified the Lr to 10e-5 and 10e-3. The small set (S_1 dataset) is used for training, and the D_2^{Test} dataset is used for testing.

The results presented in Table 2 provide a quantitative evaluation of different learning rates. When a high learning rate (Lr=10e-3) is used, the model exhibits poor generalization, reflected in low recall and AP values. This is likely due to excessively large parameter updates during training, leading to instability or failure to properly converge. On the other hand, with a low learning rate (Lr=10e-5), while the model maintains high recall, the slight decrease in AP indicates that the learning rate may be too low, resulting in slower convergence and insufficient fine-tuning of parameters. The optimal learning rate is found to be (Lr=10e-4), where the model achieves the best balance between parameter updates and stability. At this rate, the model attains high precision and recall across all ship classes, making it the most effective choice in this scenario.

4.3. Compare with SoTA

This section compares our proposed method with SoTA on the mAP metric. For each method, we use a corresponding training and testing dataset. For instance, we use D_1^{Train} and D_1^{Test} to train our model and compare with Zhang_2022 [13], and Zhang_2021 [10], Liu_2020 [12], Liu_2022 [7], Han_2021 [28], and SDNet_2022 [11], and VIB_2023 [14]. Also, we use D_2^{Train} and D_2^{Test} to train another model and compare to Biaohua_2022 [19], Yani_2022 [17], and VIB_2023 [14].

Because the mAP is maximum when $n_{queries} = 200$ for our method, as shown in Table 1, we select the setting in this experiment. The comparison among SoTA is reported in Table 3, and some conclusions can be drawn as below:

- Baseline framework is the key factor to have a better result. Cui_2019 [7] and Liu_2020 [12] base on YoloV3. Therefore, its performance is not as good as Liu_2022 [7], which is based on the SSD framework. Han_2021 [28] is based on YoloV4, and its performance does not show an improvement compared to Liu_2020 [12] base on YoloV3. Using the advantage YoloV5, SDNet_2022 [11] significantly improved compared with Liu_2020. The YoloV5 framework helps mAP increases up to 8% compared to the YoloV3 framework. VIB_2023 [14] is based on YoLoX, and our method is based on the DETR backbone. These frameworks have recently been advantageous methods for object detection. Hence, the results are better than others. It is worth noting that ship detection research typically leverages an object detection framework as its foundation, often with some custom modifications. Therefore, inheriting the capabilities of such a novel and powerful framework naturally leads to improved results.

- When the number of samples in the training set is reduced, DETR-based methods tend to work better than CNN-based methods. In the table, Biaohua_2022 [19] is a CNN-based detector, and Yani_2022 [17] is a DETR-based detector. The mAP given by Yani_2022 and Biaohua_2022 are 0.965 and 0.9963, respectively. However, the performance could be improved if we select a suitable hyperparameter. In our work, by setting $n_{queries} = 200$ the mAP could improve to 0.981.

- Our method is comparable to the best CNN-based method for ship detection. If the training dataset has more samples than the testing dataset, our method is comparable to the Yolo-based method like VIB_2023 [14]. In detail, when D_1^{Train} and D_1^{Test} are used for training and testing, both mAPs for our method and VIB_2023 [14] are similar. However, when the number of samples in the training set is equal to that in the testing set, our method is slightly better than the VIB_2023

Table 3. Performance comparisons of various methods. The best results are marked in **bold**.

Method	Train+Val / Test (in %)	fishing boat	container ship	ore carrier	bulk cargo carrier	passenger ship	general cargo ship	mAP
Zhang_2022 [13]	90/10	0.824	0.940	0.859	0.915	0.787	0.914	0.873
Zhang_2021 [10]	90/10	-	-	-	-	-	-	0.946
Cui_2019 [9]	80/20	0.900	0.940	0.90	0.910	0.910	0.900	0.910
Liu_2020 [12]	80/20	-	-	-	-	-	-	0.908
Han_2021[28]	80/20	-	-	-	-	-	-	0.906
Liu_2022 [7]	80/20	-	-	-	-	-	-	0.964
SDNet_2022 [11]	80/20	**0.986**	0.995	**0.989**	0.990	0.982	0.989	0.988
VIB_2023 [14]	80/20	0.979	**1**	0.987	**0.994**	0.994	**0.993**	**0.991**
Ours (n_{query} = 200)	80/20	0.982	**1**	**0.989**	0.991	**0.995**	0.990	**0.991**
Yani_2022 (ESDT) [17]	50/50	-	-	-	-	-	-	0.593
Yani_2022 (DETR) [17]	50/50	-	-	-	-	-	-	0.965
Biaohua_2022 [19]	50/50	0.940	0.987	0.966	0.978	0.937	0.972	0.963
VIB_2023 [14]	50/50	0.970	0.986	0.984	**0.991**	**0.964**	0.989	0.98
Ours (n_{query} = 200)	50/50	0.970	**0.995**	**0.992**	0.992	0.957	**0.992**	**0.982**

[14]. In detail, when D_2^{Train} and D_2^{Test} are used for training and testing, our method is slightly better than the SoTA in VIB_2023 [14].

In Table 3, the results indicate that the DETR method generally outperforms the YoLoX-based method [14] when the number of training samples is reduced. Therefore, we designed an experiment using D_2^{Test} as the testing set and a subset of D_2^{Train} as the training set. These subsets are categorized into $Small$, $Medium$, and $Large$ levels, corresponding to S_1, S_2, and S_3 subsets. The results in Figure 4 compare our method with the state-of-the-art (SOTA) method proposed in VIB_2023 [14].

If the training dataset is S_1, the mAP given by our method improves 17.5% compared to VIB_2023 [14]. When the number of training samples is increased, the improvement is reduced. The enhancement on mAP is 3.3% if S_2 subset is used for training and if the training dataset is S_3, the mAPs from both settings are pretty equivalent. The observation clearly points out that the DETR-based method could help if the number of training samples is limited. Table 4 reports the detailed detection result for each ship category.

While DETR-based detectors can achieve higher accuracy on smaller datasets, it is important to also assess their computational complexity compared to CNN-based detectors. Table 5 presents a comparison of the computational performance between two object detection models: VIB_2023 [14] and Deformable DETR. Three key metrics are compared: Frames Per Second (FPS), GFlops, and the number of

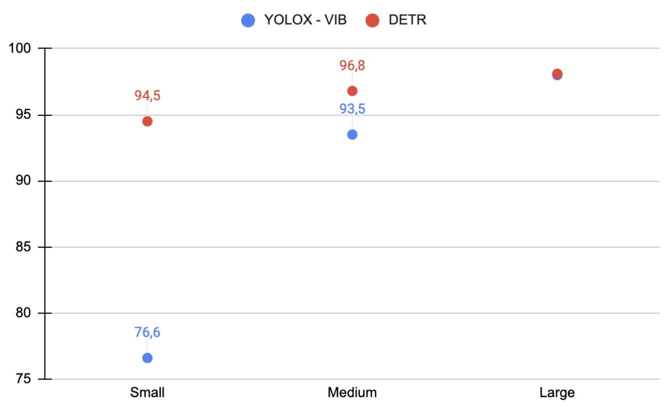

Figure 4. Performances when the number of training samples is limited. "Small" means 30% training samples, "Medium" means 70% training samples, and "Large" means 100% training samples from D_2^{Train}.

parameters (in millions). Both models show similar FPS, with DETR slightly outperforming VIB_2023 [14] at 12.6 FPS versus 12.39 FPS on a TiTanRTX GPU. However, DETR exhibits higher computational complexity, requiring 11.01 GFlops, compared to the more efficient 9.92 GFlops of VIB_2023 [14]. Despite its higher computational demands, DETR uses fewer parameters, with 39.82 million compared to VIB_2023's 55.33 million. This suggests that DETR maintains competitive speed with fewer parameters but at the cost of increased computational complexity.

Table 4. Performance on small datasets. S_1 means 30% training samples, S_2 means 70% training samples, S_3 means 100% training samples from D_2^{Train}.

Scenario	**Metrics**	**fishing boat**	**container ship**	**ore carrier**	**bulk cargo carrier**	**passenger ship**	**general cargo ship**	mAP
S_1 by VIB	recall	0.878	0.941	0.924	0.913	0.657	0.946	
	AP	0.796	0.884	0.831	0.765	0.524	0.794	0.766
S_1 by DETR	recall	0.971	0.993	0.994	0.995	0.952	0.96	
	AP	0.912	0.986	0.965	0.949	0.861	0.969	0.941
S_2 by VIB	recall	0.940	0.984	0.962	0.971	0.891	0.964	
	AP	0.922	0.980	0.935	0.953	0.873	0.946	0.935
S_2 by DETR	recall	0.977	1	0.992	0.995	0.968	0.997	
	AP	0.958	0.995	0.967	0.978	0.922	0.986	0.968
S_3 by VIB	recall	0.978	0.986	0.990	0.995	0.972	0.993	
	AP	0.970	0.986	0.984	0.991	0.964	0.989	0.98
S_3 by DETR	recall	0.985	0.995	0.999	0.998	0.980	0.997	
	AP	0.970	0.995	0.992	0.992	0.957	0.992	0.982

Table 5. Complexity comparison between VIB-detector and DETR-detector.

	VIB_2023 [14]	DETR
FPS	12.39	12.6
GFlops	9.92	11.01
#parameters (M)	55.33	39.82

4.4. Ablation study of loss functions.

This section discuss an ablation study on training losses. Deformable DETR employs multiple loss functions for training, including focal loss, GIoU loss, and L1 loss, each serving a distinct purpose: focal loss for classification, GIoU for bounding box regression, and L1 for object detection. The combination of these losses ensures the success of the training process. While all loss functions are crucial, their contributions can be adjusted. By default, the weights are set at 2.0 for focal loss, 2.0 for GIoU loss, and 5.0 for L1 loss. To evaluate the impact of each, we reduced the weight of these losses by a factor of ten, one at a time, and compared the results to the default setting. The results in Table 6 demonstrate that L_{cls} (classification loss) is the most significant; reducing its weight leads to a significant performance drop. Conversely, reducing the object loss has less impact, with performance remaining close to the original setting. Localization is slightly affected by GIoU loss, as mAP decreases to 0.931 compared to 0.941 in the original setting.

Table 6. mAP comparisons of Deformable DETR when reducing training losses.

	L_{GIoU}	L_1	L_{cls}
fishing boat	0.899	0.899	0.227
container ship	0.976	0.985	0.246
ore carrier	0.953	0.947	0.212
bulk cargo carrier	0.951	0.966	0.151
passenger ship	0.845	0.849	0.0173
general cargo ship	0.964	0.969	0.160
mAP	0.931	0.936	0.178

4.5. Feature analysis

Experimental results in Section 4.3 point out that the DETR method is better than the CNN methods if the number of samples in the training set is limited. However, it is worth explaining why the DETR method can have a better result in the special scenario. The major difference between the two methods is the attention module in the DETR encoder. This module allows non-local interaction to learn a better feature. Therefore, we visualize the features given by both methods after a model's backbone, neck, and head. Given one input image, feature maps are extracted after one module. The sum of one feature map represents whether this feature map is important or not. Therefore, we selected 20 important feature maps for each backbone, neck, and head to create a heat map. The map is averaged from all important feature maps and represents key points on an image.

Figure 5 illustrates examples of heat maps generated from an input image. The first row displays feature maps from DETR, the second row shows heat maps from the VIB_2023 [14] method, which employs a feature selection loss for learning features, and the third row presents heat maps from YOLOX [29], which relies purely on CNN networks. The results indicate that DETR, with its attention mechanism, can better

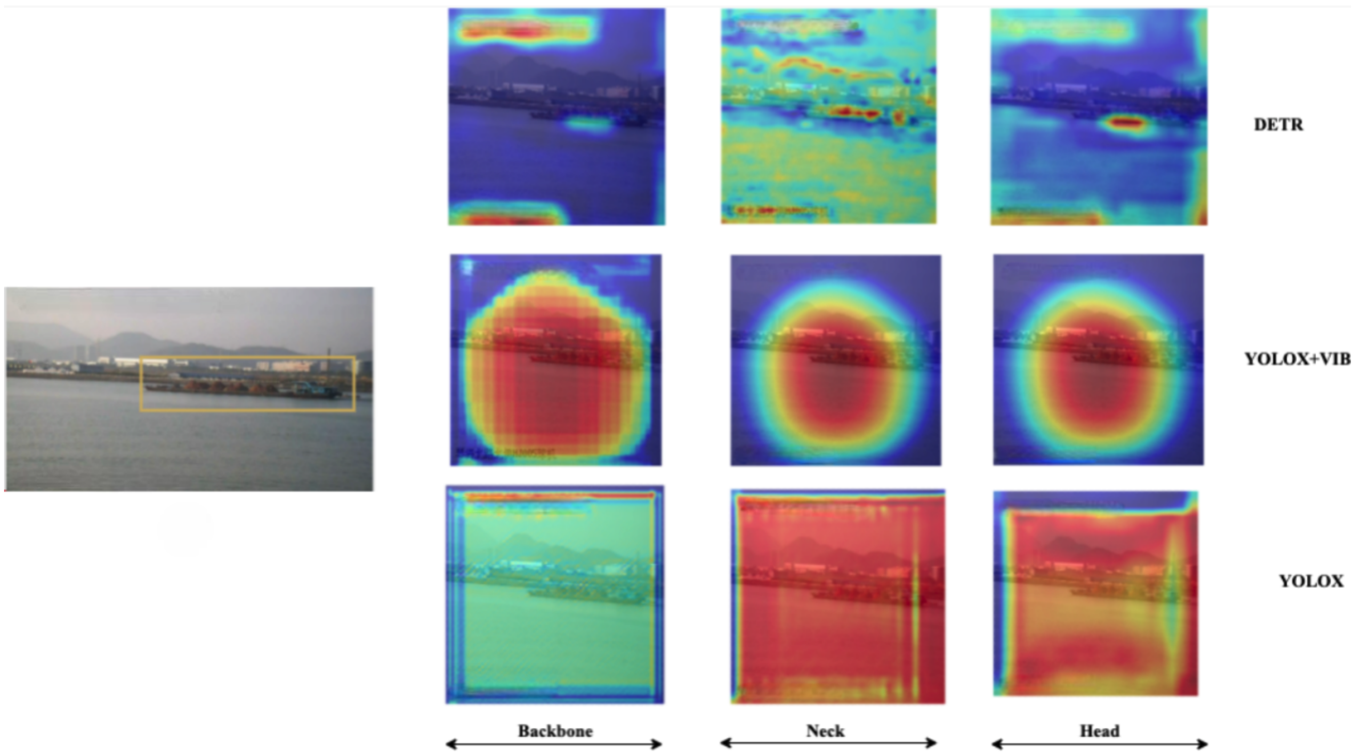

Figure 5. Feature maps on the classifier head, the neck and the backbone. (Text on original image had been removed.)

focus on non-background objects. For instance, after the head, the feature map highlights the text with a score on the survival system, and the ship is also highlighted, though not as prominently as the text. As the features are processed through the neck, higher semantic features are learned, causing the ship to become more prominent while the focus on the text diminishes. Key points are concentrated on the ship at the head, with reduced attention on the text.

In contrast, VIB_2023 [14] produces sparse heat maps where many pixels at the image's edge do not respond. However, these maps do not precisely focus on the object. This occurs because VIB_2023 [14] uses a feature selection loss to identify important features, leading to sparse and highly discriminated feature maps that do not exactly center on the object. The map distributions are quite similar overall, with slight improvements from the backbone to the head. Without the feature selection loss, the distribution of heat maps would likely be more uniform as shown in the third row.

5. Conclusion

This paper discusses a simple but efficient method for ship detection. By adjusting the number of object queries in the DETR model, we can have a comparable detector for ship detection on a large-scale dataset. In addition, the method shows a significant improvement if we do not have enough data in a training set. Heat map visualization explains why our method could be better than CNN-based methods. The features are learned to focus on non-background information, and the discriminated level of the feature map is better.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, pp. 21–37, Springer International Publishing, 2016.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas Blvd), pp. 779–788, 2016.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016.

[6] Q. Li, D. Xiao, and F. Shi, "A decoupled head and coordinate attention detection method for ship targets in sar images," *IEEE Access*, vol. 10, pp. 128562–128578, 2022.

[7] J. Zheng and Y. Liu, "A study on small-scale ship detection based on attention mechanism," *IEEE Access*, vol. 10, pp. 77940–77949, 2022.
[8] H. Li, L. Deng, C. Yang, J. Liu, and Z. Gu, "Enhanced yolo v3 tiny network for real-time ship detection from visual image," *IEEE Access*, vol. 9, pp. 16692–16706, 2021.
[9] H. Cui, Y. Yang, M. Liu, T. Shi, and Q. Qi, "Ship detection: An improved yolov3 method," in *OCEANS 2019 - Marseille*, pp. 1–4, 2019.
[10] T. Liu, B. Pang, L. Zhang, W. Yang, and X. Sun, "Sea surface object detection algorithm based on yolo v4 fused with reverse depthwise separable convolution (rdsc) for usv," *Journal of Marine Science and Engineering*, vol. 9, no. 7, 2021.
[11] M. Zhang, X. Rong, and X. Yu, "Light-sdnet: A lightweight cnn architecture for ship detection," *IEEE Access*, vol. 10, pp. 86647–86662, 2022.
[12] T. Liu, B. Pang, S. Ai, and X. Sun, "Study on visual detection algorithm of sea surface targets based on improved yolov3," *Sensors*, vol. 20, no. 24, 2020.
[13] Q. Zhang, Y. Huang, and R. Song, "A ship detection model based on yolox with lightweight adaptive channel feature fusion and sparse data augmentation," in *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–8, 2022.
[14] D.-D. Ngo, V.-L. Vo, T. Nguyen, M.-H. Nguyen, and M.-H. Le, "Image-based ship detection using deep variational information bottleneck," *Sensors*, vol. 23, no. 19, 2023.
[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-End Object Detection with Transformers*, pp. 213–229. 11 2020.
[16] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," arXiv, 2020.
[17] Y. Zhang, M. J. Er, W. Gao, and J. Wu, "High performance ship detection via transformer and feature distillation," in *2022 5th International Conference on Intelligent Autonomous Systems (ICoIAS)*, pp. 31–36, 2022.
[18] Z. Shao, W. Wu, Z. Wang, W. Du, and C. Li, "Seaships: A large-scale precisely annotated dataset for ship detection," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2593–2604, 2018.
[19] B. Ye, T. Qin, H. Zhou, J. Lai, and X. Xie, "Cross-level attention and ratio consistency network for ship detection," in *2022 26th International Conference on Pattern Recognition (ICPR)*, pp. 4644–4650, 2022.
[20] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, (Santiago, Chile), pp. 1440–1448, 2015.
[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.
[22] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.
[23] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun, "You only look one-level feature," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13034–13043, 2021.
[24] X. Lei, H. Pan, and X. Huang, "A dilated cnn model for image classification," *IEEE Access*, vol. 7, pp. 124087–124095, 2019.
[25] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," *arXiv preprint arXiv:2203.16527*, 2022.
[26] Z. Zong, G. Song, and Y. Liu, "Detrs with collaborative hybrid assignments training," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6725–6735, 2022.
[27] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.
[28] X. Han, L. Zhao, Y. Ning, and J. Hu, "Shipyolo: An enhanced model for ship detection," *Journal of Advanced Transportation*, vol. 2021, pp. 1–11, 06 2021.
[29] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021.
[30] Z. Zhang, L. Zhang, Y. Wang, P. Feng, and R. He, "Shiprsimagenet: A large-scale fine-grained dataset for ship detection in high-resolution optical remote sensing images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 8458–8472, 2021.
[31] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, March 1955.
[32] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Long Beach, CA, USA), pp. 658–666, 2019.
[33] P. Hinz, "The layer-wise l1 loss landscape of neural nets is more complex around local minima," 2021.
[34] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.