

RT-DETR-SEA: Small object enhanced architecture for marine debris detection

Guangyu Chen

Basis International School Park Lane Harbor

Guangyu.Chen40730-biph@basischina.com

Abstract

Object detection poses significant challenges for small marine debris due to complex beach environments. The main difficulties are homogeneous and low-contrast backgrounds, color similarities between targets and surroundings, and small objects which are often occluded. To address these challenges, this paper proposes a Small Object Enhanced Architecture (SEA) to the RT-DETR hybrid encoder to improve small object detection performance in beach environments. The enhanced encoder integrates an SPDConv layer into the P2 feature layer to capture spatial information and introduces a CSPOmniKernel module after feature concatenation to build up multi-scale feature representation. Through ablation study and comparison with different models on three distinct datasets focused on marine debris, our RT-DETR-SEA showed a 13.51% increase in mAP@50 compared to the baseline RT-DETR-r18 model and a 9.34% increase in mAP@50 against directly adding a P2 layer. Furthermore, RT-DETR-SEA achieved a similar frames per second (FPS) rate with the baseline model. The source codes are available on [GitHub](#).

Keywords: RT-DETR-SEA, CSPOmniKernel, small object detection, multi-scale feature representation.

1. Introduction

Object detection is an important task in computer vision which carries out practical applications, such as environmental monitoring, autonomous systems, and resource management [1, 2]. One specific branch is the detection of small marine debris which can address environmental challenges [3, 4], as marine debris causes severe threats to marine life, including involuntary ingestion and entanglement [5]. For example, effective detection can be used to analyze pollution trends and help different cleanup activities [6, 7]. However, current object detection algorithms lack accuracy while working in beach settings, which is mainly due to the complex settings of locating small objects against both low-contrast and homogeneous backgrounds [8] and polluted therefore crowded ones.

Although models are improved in certain aspects due to

the development of algorithms like YOLO (You Only Look Once) [9] and DETR (Detection Transformer) [10], detection performance has remained challenging for special features related to beach environments [11, 12]. Beach areas tend to have complex backgrounds, such as large swaths of sand which lack high texture/contrast, or the sunlight on sand and water which may cause some of the image to be overexposed, taking away details that should remain visible. Also, the colors in which shells or wreckage are painted will effectively diminish color cues for the algorithm when they match those of the sandy background. It's worth adding that in real-time detection, there will also be dynamic factors such as waving motions which introduce more noisiness. Further, occlusions happen often on beaches with high density. Thus, this study attempts to solve these challenges for enhancing the performance of object detection models on marine debris, particularly in tasks where high precise detection of small objects is required.

Our main contributions are as follows:

- We integrate an SPDConv layer into the P2 feature layer of the RT-DETR model, which allows for down-sampling without loss of spatial information, and preserves fine-grained details that are crucial for detecting small objects against low-contrast backgrounds.
- We introduce the CSPOmniKernel module after the feature concatenation step, which processes the input through a convolution layer, extracts a portion of the channel data to pass through the OmniKernel module, and then concatenates it with the skipped portion before applying a final convolution.
- We validate our proposed enhancements through ablation and comparison studies on three marine debris datasets.

Received: 1 November 2024 **Accepted:** 15 December 2024

Published: 25 January 2025

Citation: Chen, G. (2025). RT-DETR-SEA: Small object enhanced architecture for marine debris detection. *The Young Scientist*, 4(1), 44-55.

<https://doi.org/10.5281/zenodo.14739335>

The results demonstrate that our model outperforms baseline RT-DETR models in real-time detections.

2. Related Work

2.1. Traditional Object Detection Algorithms

Traditional object detection algorithms have formed the grounds for computer vision, which used hand-engineered features together with comprehensive search methods. Classic early systems, such as those by Viola-Jones [13], used Haar-like features together with cascaded classifiers to realize real-time face detection. However, they are often limited in generalization and often fail to handle scale variations and complex backgrounds [14]. Other notable techniques are Histogram of Oriented Gradients (HOG) combined with Support Vector Machines (SVMs) for object detection. In particular, Dalal and Triggs [15] presented the HOG descriptors for pedestrian detection, which demonstrated robustness under changes in illumination and pose. They captured not only the edge structure but also the gradient structure of local shapes effectively, though computationally expensive due to dense calculations. Additionally, small-size objects were hard to detect because of limited feature representation. In general, most traditional algorithms are computationally intensive and require meticulous tuning of parameters, which limits their real-time application and performance in cluttered or low-contrast environments.

2.2. Deep Learning Based Methods

Object detection performance improved after the introduction of deep learning. Region-based Convolutional Neural Networks (R-CNN) [16] introduced region proposals via CNNs to accomplish object classification tasks. Fast R-CNN [1] enhanced this by unifying region proposal processing and classification in a single network, while Faster R-CNN [2] went one step ahead with the introduction of Region Proposal Networks (RPNs), leading to an entirely end-to-end trainable architecture with faster inference. Single-shot detectors like Yolo [9, 17] and SSD [18] were proposed with a focus on real-time detection times by eliminating the region proposal step and directly predicting object classes and bounding boxes from feature maps. Feature Pyramid Networks (FPN) [19] were proposed to use multiscale feature representations for enhancing detection ability by fusing low-level and high-level feature maps benefiting the detection of small objects. Additionally, RetinaNet [11] addressed the issue of class imbalance in single-stage detectors by introducing the focal loss, which down-weights easy examples and focuses training on hard negatives. However, current models often struggle with detecting small objects in cluttered environments [3, 4].

2.3. End-to-end Object Detectors

Attention mechanisms [20, 21] and transformer architectures [22] have also been integrated in recent advances of detection frameworks. Detection Transformer (DETR) [10] rethinks object detection as a direct set of prediction problems, which taking advantage of the global self-attention mechanism [23] in transformers modeled the interaction between individual objects with their surroundings better [22]. This also removes the requirement of anchor boxes, non-maximum suppression, etc. To overcome the convergence speed of DETR, as well as its difficulty in detecting small objects, Deformable DETR [24] devised a multi-scale deformable attention module dedicated to capturing essential spatial locations which led to out-performance in small objects and faster training. Nevertheless, despite those tuning path, DETR and its extensions, like RT-DETR [25], still have difficulties in detecting small objects that might be embedded in complex environments like beaches. The global attention mechanisms struggle to capture the fine-grained details for small objects as beach scenes often have homogeneous and low-contrast compositions and high glare, in addition, dynamic elements that distort the background further complicate global attention effects. DETR often confused small objects with the background and is less accurate than Faster R-CNN on detection. To overcome these limitations, multi-scale context to the feature representation [19], more fine-grained positional encoding [26] and specialized error functions like focal loss [11] have been employed to enhance model performance in adverse settings. As a result, methods tailored exclusively to the challenges of small object detection at beaches are vital; these steps would enhance the practicality of high-performance models like RT-DETR in environmental monitoring and conservation projects.

3. Methodology

Current small object detection algorithms lack accuracy in detecting marine debris on beaches, particularly because of the complex environment different beaches can have. For example, some beaches mostly consist of sand and low contrast backgrounds, while more polluted ones are more complicated and involve occlusions. Therefore, we aim to solve this issue and our improvements focus on modifying the hybrid encoder architecture by adding a P2 layer, introducing SPDConv [27], and embedding the CSPOmniKernel module to process feature extraction on a variety of scales.

3.1. P2 Layer Integration

The original RT-DETR architecture uses a three-layer hybrid encoder, consisting of P3, P4, and P5 tracks, which are obtained from the backbone network. This begins with processing these layers following a CSP (Cross Stage Partial) [28] architecture which concatenates the layers in an

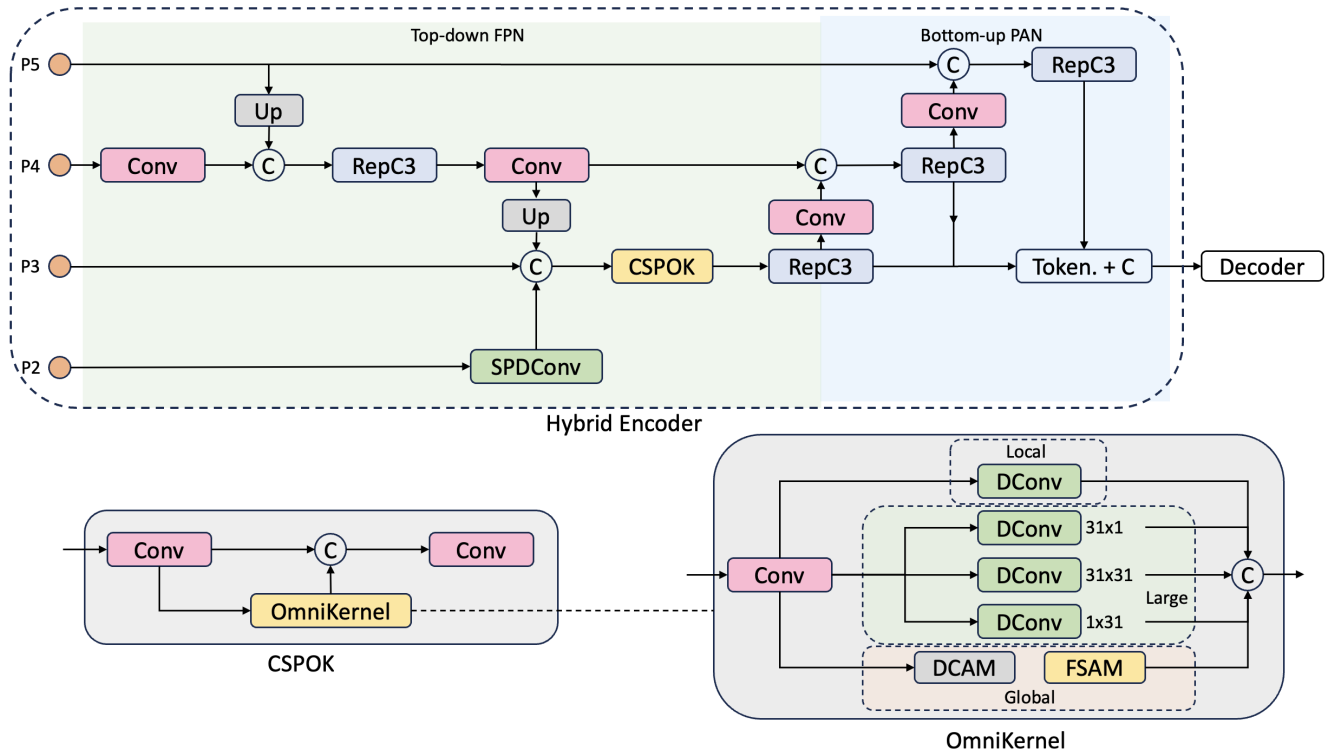


Figure 1. **Overview of the RT-DETR-SEA Hybrid Encoder Architecture.** The architecture is divided into two main sections: Top-down Feature Pyramid Network (FPN) and Bottom-up Path Aggregation Network (PAN). Layers P3, P4, and P5 are processed through convolutional operations, RepC3 modules, and concatenation operations to ensure multi-scale feature integration. The newly introduced P2 layer is routed through the SPDConv module, enhancing small object detection by retaining fine-grained spatial details. The CSPOmniKernel module processes features from P3, applying a combination of local and global attention mechanisms, including depth-wise convolutions (DConv), a dual-domain channel attention module (DCAM), and a frequency-based spatial attention module (FSAM).

upsampling and downsampling fashion. The first half is a top-down Feature Pyramid Network (FPN), where we do convolution on P4 and take a concatenation between the upsampled P5 and P4. If they are not skipped, P4 is subsequently processed similarly via an RepC3 module and a convolution, and upsampled to concatenate with the P3 features. The second half is a bottom-up Path Aggregation Network (PAN) [29], complementary to the FPN, where the features gradually go from the P3 track to the P5 track in a similar process. The idea behind this mechanism is intuitive because object detection requires integration of multi-scale features.

The standard approach, on the other hand, improves performance on small objects by adding a P2 layer in the hybrid encoder and adopting similar processing as P3, P4 and P5. In specific, this adds an entire new track for final concatenation to the decoder, as well as two connections with the P3 track. However, this can dramatically increase the computational overhead. Instead of introducing a new track for P2 as presented in Figure 1, we follow an alternative and more compact way of information flow, where P2 track is going through SPDConv module, then concatenated with the P3 layer. This way we can keep the fine-grained informa-

tion from P2 without dramatically enlarging the model.

3.2. SPDConv Module

The SPDConv module aims to overcome some of the shortcomings in traditional strided convolutions by disposing of non-adjacent tokens which provide finer patterns. To maintain the spatial information of feature maps, SPDConv adopts a space-to-depth (SPD) layer, further with a non-strided convolution. It preserves a great deal of salient information that is lost when sliding window-based operations are strided, making it easier to gain accurate detections on important details.

In our model, the P2 layer is convolved by SPDConv before concatenating with P3. The space-to-depth reshapes the input feature map to have smaller spatial dimensions, while simultaneously increasing the channel depth so that it maintains all of the original information. In this formula, for an input tensor $X' \in \mathbb{R}^{C \times H \times W}$, with C channels over spatial dimensions H and W , the transformation rearranges the spatial features as follows:

$$X' = \text{SPD}(X), \quad X' \in \mathbb{R}^{(C \times r^2) \times \frac{H}{r} \times \frac{W}{r}} \quad (1)$$

Here, r is the downscaling factor, and X' is the trans-

formed feature map with spatial dimensions reduced by r and channels increased by r^2 . This transformation ensures that no spatial information is lost during downsampling. Following the SPD transformation, a non-strided convolution is applied to process the high-dimensional feature map while preserving its resolution:

$$Y = W_{Conv} * X' \quad (2)$$

where W_{Conv} are the learned weights of the convolutional layer, and $*$ denotes the convolution operation. The convolution reduces the expanded channel dimension $C \times r^2$ back to a manageable size while allowing the network to learn discriminative features without spatial loss.

3.3. CSPOmniKernel Module

In addition, to enrich the ability of the model to obtain information with different scales and contexts, we propose a module CSPOmniKernel in the P3 trajectory. This module draws inspiration from the Cross Stage Partial (CSP) networks and the OmniKernel mechanism [30] to aggregate representations from global to local scales.

After the concatenation, we pass the combined feature map through a convolutional layer in this track. Then, we separately extract channels from the feature maps, then feed these through the OmniKernel layer which is intended to learn diverse receptive fields with few parameters.

3.3.1 Cross Stage Partial Integration

CSPOmniKernel initially uses the CSP strategy to split the input feature map into two parts. The first part goes through the layer of OmniKernel to get multi-scale features. The other part bypasses the OmniKernel layer and behaves as a shortcut (residual connection) to keep the original feature representations. This process greatly lightens the computational load with faster inference and better detection performance. This process can be described through a mathematical expression as follows:

$$F_a = F_{in}[:, 0 : X \cdot C, :, :] \quad (3)$$

$$F_{a'} = \text{OmniKernel}(F_a) \quad (4)$$

$$F_{final} = \text{Conv}(F_{a'} \oplus F_{in} \setminus F_a) \quad (5)$$

where \oplus denotes channel-wise concatenation and X is the extraction ratio of channels to be passed through the OmniKernel layer.

4. OmniKernel Layer

Given an input feature map $P_3 \in \mathbb{R}^{C \times H \times W} X$, where C , H , and W represent the number of channels, height, and width respectively, we first apply a 1×1 convolution

kernel to reduce the dimensionality and isolate important features. Then, we distribute the input into three separate branches: local, large, and global. The local branch captures fine-grained spatial details essential for small object detection by applying a x depth-wise convolution (DConv) kernel with a 1×1 kernel size:

$$Y_{Local} = W_{DConv}^{local} * X \quad (6)$$

where W_{DConv}^{local} denotes the weights of the depth-wise convolution responsible for processing local features. This branch ensures the capture of high-resolution details critical for detecting small objects that would otherwise be lost in larger-scale operations.

Adding on to the local branch, to better capture elongated or non-uniform object features that may be present in marine debris, we use the large branch which has depth-wise convolutions with both square and strip-shaped kernels:

$$Y_{Large} = W_{DConv}^{large} * X + W_{DConv}^{strip1} * X + W_{DConv}^{strip2} * X \quad (7)$$

Here, W_{DConv}^{large} corresponds to a $K \times K$ convolution, while W_{DConv}^{strip1} and W_{DConv}^{strip2} represent $1 \times K$ and $K \times 1$ convolutions, respectively. In our approach, we set $K = 31$, as the kernel sizes are adjusted to provide a balance between computational overhead and capturing diverse feature scales.

Furthermore, by employing dual-domain channel attention and frequency-based spatial attention blocks, the global branch also models large-scale contextual dependencies. Similarly, the attention mechanisms learn to amplify global structure and avoid simplifying some less important regions in the latter situation which highlights more relevant structures for larger debris like plastic bags and containers. We apply the frequency channel attention (FCA) to the input X_{Global} , represented as X_G , for descriptor.

$$X_{FCA} = \mathcal{IF}(\mathcal{F}(X_G) \odot W_{1 \times 1}^{FCA}(\text{GAP}(X_G))) \quad (8)$$

Here, \mathcal{F} and \mathcal{IF} denote the Fourier transform and its inverse, and $W_{1 \times 1}^{FCA}$ represents the weights of the 1×1 convolution used for global average pooling (GAP), and \odot is the element-wise multiplication. Next, we apply the spatial channel attention and get X_{DCAM} or X_D :

$$X_D = X_{FCA} \odot W_{1 \times 1}^{SCA}(\text{GAP}(X_{FCA})) \quad (9)$$

where $W_{1 \times 1}^{SCA}$ modulates the channel-wise features. Finally, frequency-based spatial attention (FSAM) is applied, and the final output Y_{Global} , or Y_G is:

$$Y_G = \mathcal{IF}(\mathcal{F}(W_{1 \times 1}^1(X_D)) \odot W_{1 \times 1}^2(X_D)) \quad (10)$$

where W^1 and W^2 are two separate weights. The outputs from the three branches are then concatenated:

$$Y_{CSPOK} = \text{Concat}(Y_{Local}, Y_{Large}, Y_{Global}) \quad (11)$$

This combined feature map is then passed through a final convolution kernel, which can refine the representation before continuing through the P3 layer.

5. Experiments

5.1. Dataset

We used three separate datasets which all focused on debris on the beach, and they focused on different types of scenarios, including low contrast and frequent occlusion backgrounds as well as varying sizes of debris.

5.1.1 Global Solution Dataset

This publicly available dataset [31] includes 577 images with a total of 2,200 annotations across six different marine debris classes. Image annotations and augmentations were performed using Roboflow [32]. All images were resized to 640×640 dimensions and further augmented through horizontal flip, rotation, and brightness adjustments, generating 1,501 total images. Most marine debris were located on sandy beaches, which lack major textures and contrasts.

Figure 2 shows the dataset distribution and bounding box characteristics. The bar chart indicates an imbalance in the dataset, with the bottle class being the most frequent and others, such as fishing nets and aluminum cans, having significantly fewer instances. The bounding box visualization highlights the spatial spread of objects in the dataset. The scatter plots reveal that most debris is relatively small, which fits the criteria of our evaluation.

5.1.2 Marine Pollution Dataset

The Marine Pollution Dataset [33] includes 519 images and 831 annotations. The annotations were made using bounding boxes on Roboflow, and the dataset counts with 5 classes, with 'plastic bottle' class appearing most frequently and 'fishing waste' the least. The dataset includes images with different beach backgrounds and different lighting conditions, which can be a bit challenging for object detection.

For training, validation and test sets are used as 87% (1089 images), 8% (104 images) and 4% of total dataset i.e. (52 images). All the images had a preprocessing of resizing to 640×640 . During training, the augmentation techniques applied were horizontal flips, random crop zoom (0% min / 20% max range) and rotation in $[-15^\circ, +15^\circ]$, with three augmented outputs per training example.

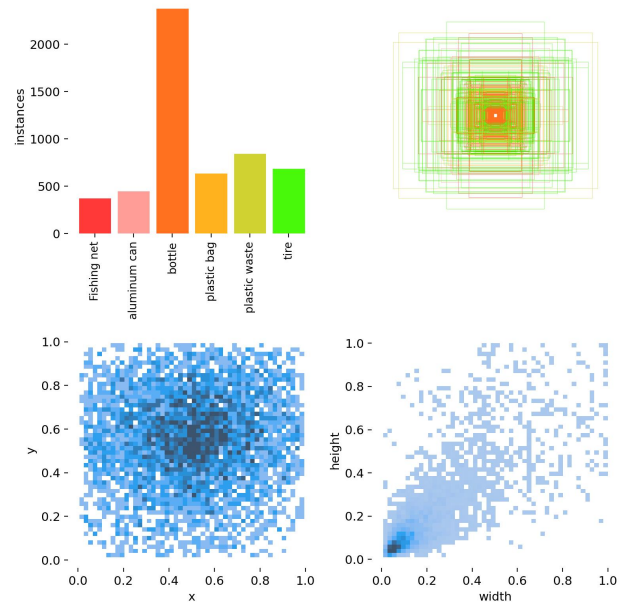


Figure 2. **Data distribution and characteristics of the Global Solution dataset.** Top left: Bar chart showing the class distribution of debris instances with the bottle class as the most frequent. Top right: Visualization of bounding boxes for each class across the entire dataset. Bottom left: Scatter plot of normalized x and y coordinates of bounding box centers. Bottom right: Scatter plot of bounding box width and height distribution.

5.1.3 Dapeng Bay Dataset

The third dataset includes a total of 514 1000×600 images covering 7 distinct debris classes, annotated and augmented using Roboflow. The data had been acquired in Dapeng Bay, Shenzhen, China using UAVs (drones), and was an area of tourism with substantial marine debris concentration. The new footage was filmed by two DJI drones, which were equipped with 12MP sensors and had the capacity to record in 4K at 30 fps as well. These drones were controlled manually at no more than 10 meters in height over the beach floor, while a coastline was included within 50 meters of the path. Videos were recorded across two different beaches on two separate days. The first has more of the existing low-contrast sand as a background, and the second includes far more occlusions of objects due to there being many additional items involved. Individual frames containing marine debris were extracted from these videos, giving approximately 10 square meters of coverage per image.

5.2. Experimental Setup

The environment used for experiments is shown in Table 1. The system ran on Ubuntu 20.04 LTS. The CPU was an Intel Xeon Gold 5418Y (96 cores, 3.8 GHz max), and the GPU was an NVIDIA GeForce RTX 4090 (24GB VRAM). CUDA version 12.4 was utilized for GPU accel-

eration, with Python 3.12.3 and PyTorch 2.3.1+cu118 (optimized for CUDA 11.8). The system also had 1.0 TiB of memory, enabling efficient handling of large datasets.

Table 1. Experimental Environment Configuration

Component	Version
Operational System	Ubuntu 20.04 LTS
CPU	Intel Xeon Gold 5418Y
GPU	NVIDIA GeForce RTX 4090
CUDA version	12.4
Python version	3.12.3
PyTorch version	2.3.1+cu118
Memory	1.0 TiB

The training setting is shown in Table 2. We used a batch size of 16 and the AdamW optimizer [34] with a learning rate of 0.0001 and weight decay of 0.0001. A pretrained model, rtdetr-r18, was utilized to initialize the training. The total number of epochs was set to 300 with 2000 iterations dedicated to warmup. The first dataset was split into training, validation, and testing sets with a ratio of 70%, 15%, and 15%, respectively. The second dataset was split as 70%, 20%, and 10%. For the third dataset, the split was 93% for training and 7% for validation.

Table 2. Training Settings

Setting	Value
Batch Size	16
Optimizer	AdamW
Learning Rate	0.0001
Weight Decay	0.0001
Epochs	300

5.3. Evaluation Metrics

We employed standard object detection metrics: precision, recall, mean Average Precision at an Intersection over Union (IoU) threshold of 0.5 (mAP@50), and mean Average Precision averaged over IoU thresholds from 0.5 to 0.95 in increments of 0.05 (mAP@[.5:.95]).

Precision (P) and recall are defined as:

$$P = \frac{TP}{TP + FP} \quad (12)$$

$$R = \frac{TP}{TP + FN} \quad (13)$$

where TP , or true positives, is the number of correctly detected objects, FP , or false positives, is the number of incorrect detections, and FN , or false negatives, is the number of actual objects that the model failed to detect.

The Average Precision (AP) metric calculating the area under the precision-recall curve, and provides a single value that balances both precision and recall across different thresholds. It is computed as:

$$AP = \int_0^1 P(R) dR \quad (14)$$

In practice, we approximated this using a finite sum over recall levels where precision values change. The mean Average Precision at IoU 0.5 (mAP@50) is obtained by averaging the AP values for all object classes at a fixed IoU threshold of 0.5:

$$\text{mAP@50} = \frac{1}{N} \sum_{i=1}^N AP_i^{\text{IoU}=0.5} \quad (15)$$

where N is the total number of object classes, and $AP_i^{\text{IoU}=0.5}$ is the Average Precision for class i at an IoU threshold of 0.5. We also calculate the mean Average Precision across multiple IoU thresholds from 0.5 to 0.95 in increments of 0.05, denoted as mAP@[.5:.95]. This is computed as:

$$\text{mAP@[.5:.95]} = \frac{1}{T} \sum_{t=1}^T \left(\frac{1}{N} \sum_{i=1}^N AP_i^{\text{IoU}=\tau_t} \right) \quad (16)$$

where $T = 10$ is the number of IoU thresholds considered ($\tau_t \in \{0.5, 0.55, \dots, 0.95\}$), and $AP_i^{\text{IoU}=\tau_t}$ is the Average Precision for class i at IoU threshold τ_t . This metric evaluates the model's performance across a spectrum of localization strictness levels, making it particularly relevant for small object detection where precise localization is challenging yet important.

The Intersection over Union (IoU) metric itself measures the overlap between the predicted bounding box B_p and the ground truth bounding box B_{gt} , as:

$$\text{IoU} = \frac{\text{Area}(B_p \cap B_{gt})}{\text{Area}(B_p \cup B_{gt})} \quad (17)$$

6. Results

6.1. Ablation Study

To evaluate the effectiveness of the proposed enhancements to the RT-DETR architecture, we conducted an ablation study focusing on four variations of the model: (1) incorporating only the SPDConv layer without the CSPOmniKernel module, and (2) integrating the CSPOmniKernel module with different extraction percentages—25% (SEA-25), 50% (SEA-50), and 75% (SEA-75) of the channel data processed through the OmniKernel. The varying extraction rates aim to balance detection accuracy with real-time performance, as extracting less data enhances computational speed.

Table 3. Performance Comparison of RT-DETR-SEA Variants

Class	SPDConv Only		SEA-25		SEA-50		SEA-75	
	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95
Dataset 1								
All	55.7	37.3	59.4	38.6	59.7	39.6	58.8	38.7
Fishing net	73.6	47.0	77.6	43.0	81.7	45.7	79.1	43.3
Aluminum can	70.6	38.3	69.6	42.2	64.1	40.5	69.3	43.8
Bottle	62.2	34.6	76.9	45.4	69.9	40.6	67.6	36.2
Plastic bag	31.9	26.9	26.9	22.7	38.8	31.4	36.0	31.8
Plastic waste	16.0	11.8	27.8	19.2	23.7	16.6	24.2	15.5
Tire	80.1	65.4	77.9	59.0	79.7	62.6	76.5	61.9
Dataset 2								
All	71.0	51.8	79.5	57.9	74.1	55.5	77.2	56.0
Fishing net	93.3	65.1	90.7	58.2	92.6	70.0	92.9	64.7
Glass Bottle	69.7	53.2	94.9	76.9	67.0	51.0	64.5	51.8
Aluminum Can	55.6	49.3	81.5	66.5	77.5	62.7	99.5	71.1
Plastic Bottle	69.6	49.7	63.9	44.7	67.3	52.1	62.8	50.5
Fishing wastes	66.5	42.2	66.5	43.3	66.5	41.6	66.5	41.6
Dataset 3								
All	68.8	35.7	70.9	36.7	69.5	37.4	68.6	34.9
Can	99.5	69.7	99.5	69.7	99.5	69.7	99.5	59.7
Carton	64.7	29.2	70.0	31.3	56.9	26.2	60.1	24.8
Plastic bag	50.8	24.7	63.9	30.5	53.7	25.6	56.4	28.4
Plastic bottle	77.7	32.1	83.7	37.4	81.7	33.8	77.5	35.9
Plastic container	64.4	26.0	63.0	23.9	63.1	32.2	66.0	27.3
Styrofoam	78.7	42.1	74.5	42.9	83.2	43.8	81.5	42.9
Tire	45.7	25.9	41.4	20.9	48.7	30.2	39.5	25.3

Across all three datasets, as shown in Table 3, the SEA-25 variant consistently demonstrated superior general performance compared to the other models. In Dataset 1, the SEA-25 model achieved an mAP@50 of 59.4% and an mAP@50-95 of 38.6% for all classes, surpassing the SPDConv-only model, which recorded an mAP@50 of 55.7% and an mAP@50-95 of 37.3%. Notably, the SEA-25 model showed significant improvements in detecting smaller objects. For the 'plastic waste' class, characterized by low contrast and small object size, the mAP@50 increased from 16.0% in the SPDConv-only model to 27.8% with SEA-25. Similarly, for 'plastic bag' the mAP@50 remained comparable, but the SEA-25 variant maintained consistent performance across different IoU thresholds, as indicated by the mAP@50-95 scores.

In Dataset 2, the SEA-25 model again outperformed the other variants. It achieved an all-class mAP@50 of 79.5% and an mAP@50-95 of 57.9%, compared to the SPDConv-only model's mAP@50 of 71.0% and mAP@50-95 of 51.8%. Particularly for glass bottles, which include smaller and more challenging objects with faint edges, the SEA-25 model significantly improved mAP@50 from 69.7% to

94.9% and mAP@50-95 from 53.2% to 76.9%.

Dataset 3, which consists mostly of small targets, further shows the improvements of the SEA-25 variant. The SEA-25 model achieved an all-class mAP@50 of 70.9% and mAP@50-95 of 36.7%, slightly surpassing the SPDConv-only model's mAP@50 of 68.8% and mAP@50-95 of 35.7%. For specific small object classes like 'carton' and 'plastic bag' the SEA-25 model showed notable improvements. The mAP@50 for "carton" increased from 64.7% to 70.0%, and for "plastic bag," it rose from 50.8% to 63.9%. These enhancements indicate that the SEA-25 model is more adept at detecting small objects amidst complex backgrounds.

Through comparing the SEA-25 model to the SEA-50 and SEA-75 variants, we found that extracting only 25% of the channel data is closer to the balance between detection performance and computational efficiency. While the SEA-50 and SEA-75 models occasionally achieve slightly higher mAP@50 scores in certain classes, they have more computational load, which is less desirable.

6.2. Comparison With Other Models

To further validate the effectiveness of the proposed model enhancements, we conducted a comparison between our improved models and two baseline models: (1) the original RT-DETR-r18 (denoted as r18), which serves as the pretrained model used in our experiments, and (2) a variant that adds a P2 layer directly to the hybrid encoder (denoted as P2), a common approach to enhance small object detection in RT-DETR architectures. The comparison focuses on two key aspects: detection performance metrics and real-time processing capabilities.

6.2.1 Comparison of Detection Performance

The detection performance of the models was evaluated using standard metrics: Precision (P), Recall (R), mAP@50, and mAP@50-95. Figure 3 shows the heatmap comparison between the SEA-25 model and the P2 baseline model (bottom images). The direct P2 layer integration resulted in wrong identification of small bottles in the upper part, while the SEA-25 model captures smaller features more precisely. Furthermore, the SEA-25 model showed a better ability to extract relevant features as shown in its more active heatmap. The metric curves in the figure 5 show that the SEA variants consistently outperform both the r18 and P2 baseline models in terms of Precision, Recall, mAP@0.5, and mAP@0.5:0.95 across 300 epochs in Dataset 1. Notably, SEA-25 and SEA-75 have the most stable and best performance on all metrics, while the baseline models fail to reach similar levels of accuracy and consistency. Table 4 summarizes the results across the three datasets, with Figure 4 visualizing the metrics over bar graphs.

In Dataset 1, which contains the largest number of images, the SEA-25 model outperformed both baseline models and the SPDConv-only variant. It achieved an mAP@50 of 59.4% and an mAP@50-95 of 38.6%, surpassing the r18 model's mAP@50 of 52.6% and mAP@50-95 of 35.0%. While the P2 model improved upon the r18 baseline with an mAP@50 of 55.2% and mAP@50-95 of 38.1%, it did not exceed the performance of SEA-25. Notably, the SEA-50 model achieved a slightly higher mAP@50 of 59.7% and mAP@50-95 of 39.6%, but this marginal gain comes with trade-offs in computational efficiency, as discussed later. In Dataset 2, which includes various types of fishing waste, the SEA-25 model again demonstrated superior performance over the baseline models. It achieved an mAP@50 of 79.5% and an mAP@50-95 of 57.9%, compared to the r18 model's mAP@50 of 66.2% and mAP@50-95 of 48.2%. The P2 model showed some improvement with an mAP@50 of 70.0% and mAP@50-95 of 52.5% but still fell short of the SEA-25 variant. For dataset 3, SEA-25 achieved an mAP@50 of 70.0% and mAP@50-95 of 36.7%, outperforming both baseline models. The r18 model recorded an mAP@50 of 0.663 and mAP@50-95 of 35.1%, while the

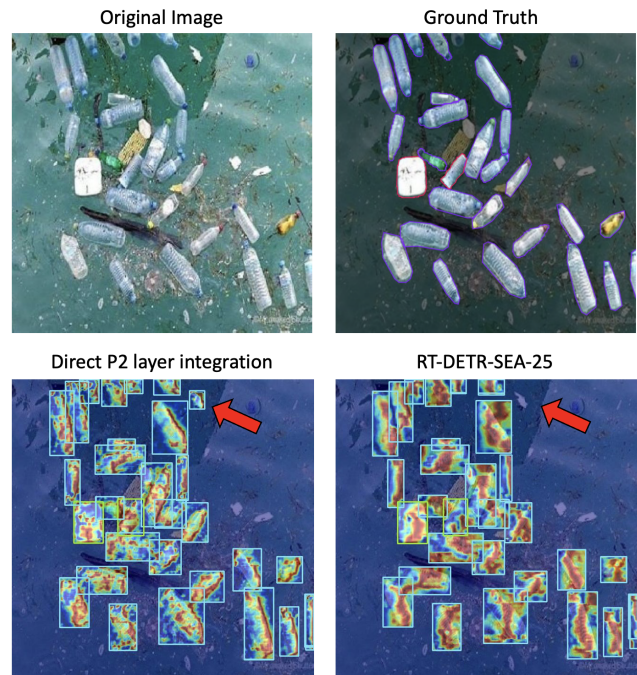


Figure 3. **Comparison of bottle detection performance.** The upper left is the original image; The upper right is the ground truth labeled image; The bottom left is the heatmap of bottle detection by the P2 model; The bottom right is the heatmap of bottle detection by the SEA-25 model. The SEA-25 model shows higher accuracy and more extracted features, while P2 shows more error predictions. The red arrow pointed out a prediction error of P2 model which SEA-25 mitigates.

P2 model showed slight improvement with an mAP@50 of 66.7% and mAP@50-95 of 36.9%.

6.2.2 Real-Time Performance Analysis

We measured the frames per second (FPS) for each model on all datasets. Table 5 presents the latency and FPS results.

The r18 model achieved the highest, 384 FPS (frames per second), in all datasets. The SEA-25 model performed similarly, with an average of 371 fps, demonstrating that we maintain real-time capabilities by only introducing the CSPOmniKernel module with a 25% extraction. The SEA-SPDonly model also yielded high FPS values, comparable to the combined SEA-25 variant. In contrast, the P2 model resulted in a large drop in FPS (and low accuracies), with an average of 254 FPS across the datasets. This decrease in real-time performance likely reflects the additional computational overhead of embedding a P2 layer directly into the hybrid encoder. While the P2 model did show some improvements in detection metrics over the r18 baseline, the trade-off in processing speed is considerable. The SEA-50 and SEA-75 models have slight decreases in FPS compared to the SEA-25 variant, averaging around 366 FPS and 370 FPS. The minor reductions in real-time performance for

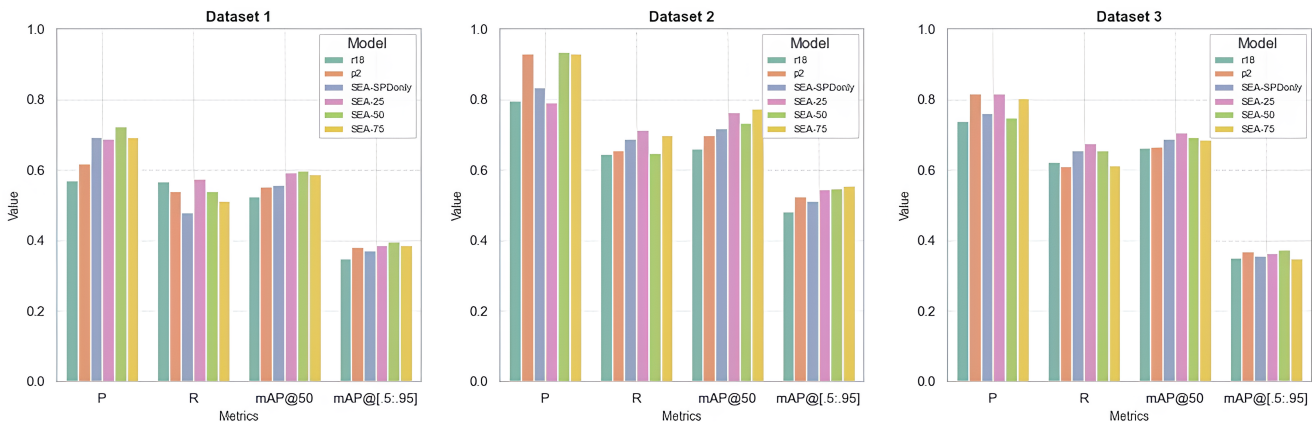


Figure 4. Performance comparison of different models across three datasets (Dataset 1, Dataset 2, and Dataset 3) based on four key metrics: Precision (P), Recall (R), mAP@50, and mAP@50:0.95. Visualization of Table 4

Table 4. Detection Performance Metrics across Different Models and Datasets								
Model	Dataset 1				Dataset 2			
	P	R	mAP50	mAP50-95	P	R	mAP50	mAP50-95
r18	57.1	56.8	52.6	35.0	79.8	64.7	66.2	48.2
P2	61.8	54.1	55.2	38.1	93.0	65.6	70.0	52.5
SEA-SPDonly	69.4	48.1	55.7	37.3	83.6	69.0	71.0	51.8
SEA-25	68.9	57.5	59.4	38.6	79.3	71.4	79.5	57.9
SEA-50	72.3	54.0	59.7	39.6	93.6	64.9	74.1	55.5
SEA-75	69.4	51.3	58.8	38.7	93.1	69.9	77.2	56.0

Dataset 3				
Model	P	R	mAP50	mAP50-95
r18	73.8	62.3	66.3	35.1
P2	81.0	61.1	66.7	36.9
SEA-SPDonly	76.1	65.7	68.8	35.7
SEA-25	81.7	66.2	70.9	36.7
SEA-50	75.0	65.7	69.5	37.4
SEA-75	80.5	61.3	68.6	34.9

these models, combined with only marginal gains in detection accuracy, suggest that extracting more than 25% of the channel data through the OmniKernel module may not be the most efficient approach for real-time applications.

Overall, the SEA-25 model achieves better detection metrics compared to simply adding a P2 layer and maintains high FPS rates comparable to the r18 model.

7. Discussion

This study focuses on how the performance of RT-DETR can be refined to better detect small objects under beach environment conditions (i.e., low-contrast and cluttered backgrounds). To keep fine-grained spatial information and improve multi-scale feature representation while avoiding huge additional computation, we concatenate the SPDConv layer in P2 level of features and introduce a CSPOmniKer-

nel module after concatenation of features. Ablation analysis in our study indicated that the SEA-25 variant, which extracts 25% of channel data for feeding into OmniKernel module showed an intuitive tradeoff between detection accuracy and real-time processing capabilities. The SPDConv enables downsample-invariant spatial processing, while the CSPOmniKernel facilitates multi-scale and contextual learning to improve feature representation in the model. In practice with our comparison studies, we demonstrated that SEA-25 was significantly outperforming both the original RT-DETR-r18 model and a standard improvement (having an additional P2 layer for all datasets). The SEA-25 model offered better detection metrics, especially in data containing a larger fraction of small objects, and came with higher frames per second (FPS) rates similar to the baseline models.

An important finding from our ablation studies is that

Table 5. Real-Time Performance Metrics across Different Models and Datasets

Model	Dataset 1			Dataset 2			Dataset 3		
	Latency (s)	Std Dev (s)	FPS	Latency (s)	Std Dev (s)	FPS	Latency (s)	Std Dev (s)	FPS
r18	0.00273	0.00329	386.7	0.00257	0.00191	388.9	0.00266	0.00495	376.4
P2	0.00393	0.00051	254.2	0.00397	0.00113	252.0	0.00392	0.00027	255.2
SPDonly	0.00267	0.00324	375.0	0.00265	0.00400	377.6	0.00270	0.00392	369.9
SEA-25	0.00266	0.00373	375.6	0.00269	0.00603	372.3	0.00273	0.00721	365.8
SEA-50	0.00272	0.00423	367.7	0.00269	0.01245	372.0	0.00279	0.00630	358.9
SEA-75	0.00266	0.00331	376.3	0.00274	0.00779	365.6	0.00271	0.01115	369.6

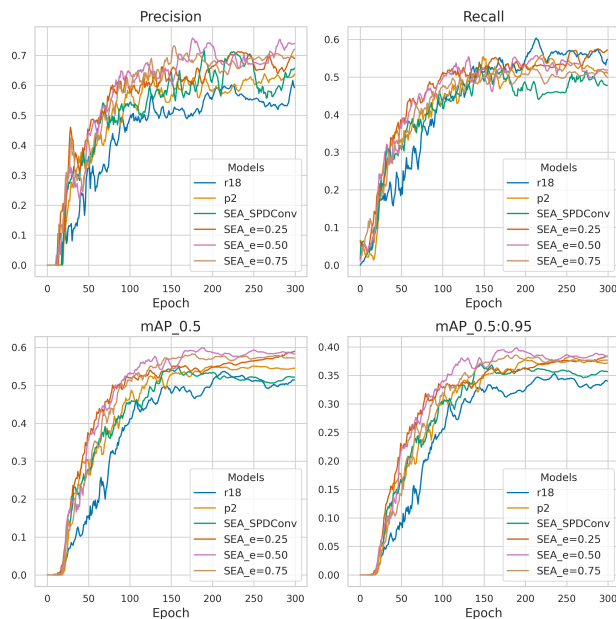


Figure 5. **Metric comparison of different models across 300 epochs for Dataset 1.** Precision (top left), Recall (top right), mAP@0.5 (bottom left), and mAP@0.5:0.95 (bottom right) show the SEA models outperform the r18 and P2 baselines.

extracting only 25% of the channel data to pass through the OmniKernel layer yields better performance than extracting larger portions like 50% or 75%. The OmniKernel layer is designed to capture multi-scale and contextual information by processing features through various convolutional operations that focus on different receptive fields. By limiting the extraction to 25%, the model can select the most important features, and this prevents the original fine-grained spatial information captured by the SPDConv layer from being turned to less representative ones.

However, the fixed extraction rate of 25% in the CSPOmniKernel module, while generally effective, may not be optimal for other scenarios. Future work could explore adaptive extraction rates which adjust dynamically based on the input data. Additionally, researching more advanced attention mechanisms or alternative architectures within the OmniKernel module might further enhance detection per-

formance.

In conclusion, the proposed improvements of RT-DETR enable reliable detection of small objects in sea regions while being able to run efficiently in real-time. The effectiveness and applicability of the model for practical deployment in marine debris detection can be strengthened by resolving these limitations, as well as exploring future directions proposed.

References

- [1] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.
- [2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.
- [3] Xi Liang, Jing Zhang, Li Zhuo, Yuzhao Li, and Qi Tian. Small object detection in unmanned aerial vehicle images using feature fusion and scaling-based single shot detector with spatial context analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1758–1770, 2019.
- [4] Yun Ren, Changren Zhu, and Shunping Xiao. Small object detection in optical remote sensing images via modified faster r-cnn. *Applied Sciences*, 8(5):813, 2018.
- [5] Sarah C Gall and Richard C Thompson. The impact of debris on marine life. *Marine pollution bulletin*, 92(1-2):170–179, 2015.
- [6] Stephen DA Smith and Robert J Edgar. Documenting the density of subtidal marine debris across multiple marine and coastal habitats. *PloS one*, 9(4):e94593, 2014.
- [7] BD Hardesty and C Wilcox. A risk framework for tackling marine debris. *Analytical Methods*, 9(9):1429–1436, 2017.

- [8] Xin Xu, Shiqin Wang, Zheng Wang, Xiaolong Zhang, and Ruimin Hu. Exploring image enhancement for salient object detection in low light images. *ACM transactions on multimedia computing, communications, and applications (TOMM)*, 17(1s):1–19, 2021.
- [9] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
- [11] T Lin. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.
- [12] Feng-Yang Hsieh, Chin-Chuan Han, Nai-Shen Wu, Thomas C Chuang, and Kuo-Chin Fan. A novel approach to the detection of small objects with low contrast. *Signal Processing*, 86(1):71–83, 2006.
- [13] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.
- [14] Bo Wu and Ramakant Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 90–97. IEEE, 2005.
- [15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.
- [16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [17] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia computer science*, 199:1066–1073, 2022.
- [18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.
- [19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [20] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [21] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4794–4803, 2022.
- [22] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [23] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.
- [24] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.
- [25] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Dets beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2024.
- [26] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13713–13722, 2021.
- [27] Raja Sunkara and Tie Luo. No more strided convolutions or pooling: A new cnn building block for low-resolution images and small objects. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 443–459. Springer, 2022.

- [28] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.
- [29] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Ji-aya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.
- [30] Yuning Cui, Wenqi Ren, and Alois Knoll. Omnikernel network for image restoration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1426–1434, 2024.
- [31] reconhecimentoimgs. global solution dataset. <https://universe.roboflow.com/reconhecimentoimgs/global-solution>, jun 2024. URL <https://universe.roboflow.com/reconhecimentoimgs/global-solution>. visited on 2024-09-21.
- [32] Qinjie Lin, Guo Ye, Jiayi Wang, and Han Liu. Roboflow: a data-centric workflow management system for developing ai-enhanced robots. In *Conference on Robot Learning*, pages 1789–1794. PMLR, 2022.
- [33] Ratheshan Sathiyamoorthy. Marine pollution detection dataset. <https://universe.roboflow.com/ratheshan-sathiyamoorthy/marine-pollution-detection>, mar 2022. URL <https://universe.roboflow.com/ratheshan-sathiyamoorthy/marine-pollution-detection>. visited on 2024-09-30.
- [34] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.