electronics

MDPI

Article

Improving the Detection and Positioning of Camouflaged Objects in YOLOv8

Tong Han [1], Tieyong Cao [1,*], Yunfei Zheng [2], Lei Chen [1], Yang Wang [1] and Bingyang Fu [3]

[1] Institute of Command and Control Engineering, Peoples Liberation Army Engineering University, Nanjing 210007, China; dolphin_han@aeu.edu.cn (T.H.); b101180011@smail.nju.edu.cn (L.C.)
[2] The Army Artillery and Defense Academy of PLA, Nanjing 210007, China
[3] The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China
* Correspondence: cty_ice@aeu.edu.cn

Abstract: Camouflaged objects can be perfectly hidden in the surrounding environment by designing their texture and color. Existing object detection models have high false-negative rates and inaccurate localization for camouflaged objects. To resolve this, we improved the YOLOv8 algorithm based on feature enhancement. In the feature extraction stage, an edge enhancement module was built to enhance the edge feature. In the feature fusion stage, multiple asymmetric convolution branches were introduced to obtain larger receptive fields and achieve multi-scale feature fusion. In the post-processing stage, the existing non-maximum suppression algorithm was improved to address the issue of missed detection caused by overlapping boxes. Additionally, a shape-enhanced data augmentation method was designed to enhance the model's shape perception of camouflaged objects. Experimental evaluations were carried out on camouflaged object datasets, including COD and CAMO, which are publicly accessible. The improved method exhibits enhancements in detection performance by 8.3% and 9.1%, respectively, compared to the YOLOv8 model.

Keywords: camouflaged object; object detection; feature enhancement; data augmentation

Citation: Han, T.; Cao, T.; Zheng, Y.; Chen, L.; Wang, Y.; Fu, B. Improving the Detection and Positioning of Camouflaged Objects in YOLOv8. *Electronics* **2023**, *12*, 4213. https://doi.org/10.3390/electronics12204213

Academic Editor: Praveen Kumar Donta

Received: 12 September 2023
Revised: 9 October 2023
Accepted: 9 October 2023
Published: 11 October 2023

1. Introduction

Camouflage is a survival strategy employed by organisms to hide and avoid predators. It allows them to seamlessly blend into their surroundings, posing a challenge for computer vision in detecting them. Figure 1 illustrates an example of camouflaged objects.

Camouflage detection has significant research value in areas such as species conservation, wilderness rescue, and healthcare. Recently, remarkable advancements have been achieved in using deep learning to detect camouflaged objects [1–6]. Existing methods mainly rely on image segmentation techniques to predict the contour of camouflaged objects. However, in practical applications, it is often only necessary to quickly detect and locate camouflaged objects without the need for pixel-level segmentation.

Object detection refers to the task of recognizing specific objects in an image and marking their positions with rectangular boxes. YOLOv8 is currently a relatively advanced object detection model. However, its accuracy for camouflaged objects in the COD10K [1] dataset only achieves 62.2%. Due to the difficulty of distinguishing camouflaged objects from the background, the model needs to catch more details while obtaining global perception capabilities. The YOLOv8 model weakens the edge features and detailed information of targets through deep networks. Additionally, its simple context aggregation strategy falls short of meeting the specific requirements of camouflaged object detection [7]. Its non-maximum suppression algorithm cannot effectively filter detection boxes.

Moreover, the color and texture similarities between camouflaged objects and their surrounding environment weaken the structural characteristics of the targets. These factors result in high missed detection rates and inaccurate localization of camouflaged objects.

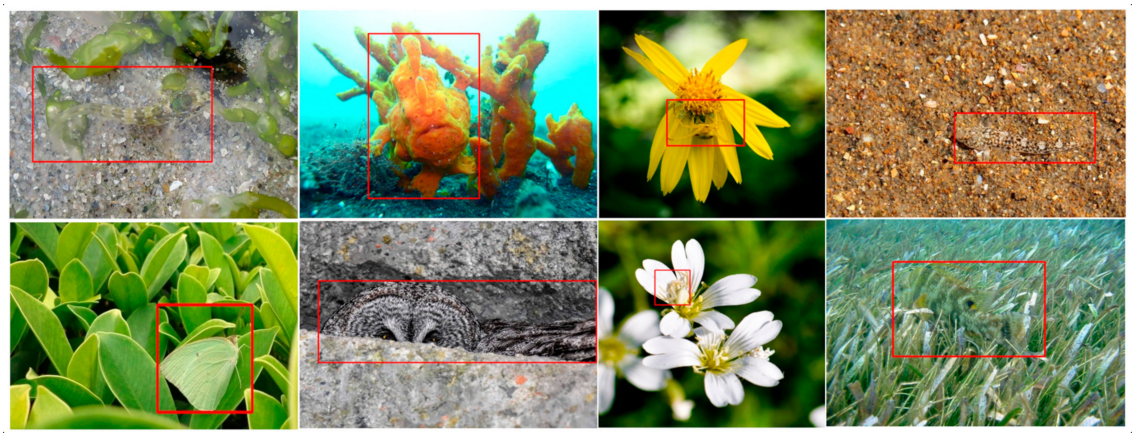


Figure 1. Camouflaged objects.

Edge information represents the intensity variations in different regions of an image, while shape information describes the global contour of an object. Both of these contain precise positional information. We believe enhancing edge and shape features can help a model accurately locate targets. Thus, we propose a feature-enhanced method building upon the YOLOv8 model, and the main improvements of our work can be outlined as follows:

(1) We propose a feature enhancement network. In the feature extraction network, an edge enhancement module was built to alleviate the issue of deep networks being insensitive to edge information. A multi-branch convolution module was introduced to help the model capture local details while also perceiving global contextual information of camouflaged objects.
(2) We introduce a novel non-maximum suppression (NMS) algorithm that combines intersection-over-union and an attenuation function to reduce the detection loss rate of the target.
(3) We present a shape-enhanced data augmentation method specifically designed for camouflaged objects, which effectively improved the model's perception of shape features in natural camouflaged object datasets.

The structure of this article is outlined below. Section 2 offers a concise summary of the research status on camouflaged object detection, object detection techniques, and data augmentation techniques. Section 3 briefly introduces the YOLOv8 object detection model and presents a detailed description of the improved method. Section 4 presents the results of the comparative experiments and ablation studies conducted on publicly available datasets, while also analyzing the limitations of the proposed method. Section 5 provides a concise overview of our research.

2. Related Work

2.1. Camouflaged Object Detection (COD)

Unlike saliency detection tasks, COD tries to detect concealed objects in complex environments. Valuable research has been conducted on COD. Existing studies may be generally divided into the following three types:

The first type focuses on designing advanced network modules or architectures to effectively explore distinctive features. SINet [1] searches for camouflaged objects in the image and then performs identity verification for all collected objects. Qin et al. [2] utilized edge cues to obtain fine-grained structures of camouflaged objects and proposed BASNet. BASNet compares the coarse predicted map with the ground truth labels and refines the less accurate predictions by learning the residuals between them.

The second type involves incorporating auxiliary tasks (such as edge extraction and salient object detection) into a multi-task learning framework. This approach allows for valuable additional clues to be discovered from shared features, thus significantly

enhancing the feature representation of camouflaged objects. Zhai [3] designed a Mutual Graph Learning (MGL) model that decomposes an image into two feature maps for specific tasks: one for coarse object localization and another for capturing its boundary details accurately. Li [4] considered a method that can favor both salient object detection and camouflaged object detection by utilizing contradictory clues. Simple samples from the camouflage dataset are treated as challenging examples for salient object detection. Under an adversarial learning framework, a joint network is used to explicitly model the prediction uncertainty by incorporating the detection of objects with or without saliency.

The third type adopts bioinspired methods by mimicking the behavioral habits of wild animals during predation in the natural world or the mechanism of the human visual system to design biomimetic networks. Yan et al. [5] proposed MirrorNet, a mirror-bioinspired adversarial network that utilizes biological visual characteristics to tackle the segmentation of camouflaged objects through instance segmentation and adversarial networks. Mei et al. [6] developed a bioinspired framework called PFNet, which simulates the predation process in nature.

The existing research is based on image segmentation techniques. However, for practical applications, this study considers the discovery of camouflaged objects as an object detection task. In such object detection tasks, the true labels of camouflaged objects are bounding boxes enclosing the objects, which also contain a large amount of background information. This makes camouflage detection more challenging.

2.2. Object Detection

Deep learning-based algorithms for object detection can be classified into three main categories: two-stage algorithms, one-stage algorithms, and transformer-based algorithms.

Two-stage algorithms segment the process detection into two distinct steps: candidate region extraction and classification. Girshick [8] proposed the first method to use CNN for feature extraction, namely R-CNN. It opened up a new chapter in the realm of object detection. Ren et al. [9] introduced a method named Faster R-CNN, which substitutes a selective search algorithm with a Region Proposal Network (RPN) and achieves end-to-end training, making the object detection system more efficient and accurate. Reppoints [10] removes the process of predefined anchor boxes and introduces multiple representative points, where each point corresponds to a local region of the target. By predicting the positions of these representative points, the position and shape of the target can be determined more accurately, and fine-grained details can be captured.

Sparse R-CNN [11] is a fully sparse approach that eliminates the need for post-processing techniques like non-maximum suppression. Two-stage algorithms generally achieve higher accuracy but often suffer from slower detection speeds.

One-stage algorithms (such as YOLO [12] and RetinaNet [13]) directly obtain the category and position of objects through dense position grids or anchor boxes. They are faster but may have less accurate localization. YOLO utilizes a grid of size $S \times S$ on the input image. In each grid cell, multiple bounding boxes and their corresponding class probabilities are predicted. This method does not require complex candidate box generation, thus greatly improving the speed of detection.

However, due to the simultaneous prediction of multiple bounding boxes in a grid, there are issues of localization error and background false positives. To address these problems, various improved versions have been proposed, such as YOLOv3 [14], YOLOv5 [15], YOLOv6 [16], YOLOv7 [17], and YOLOv8 [18]. These improvements include using more complex network structures, introducing attention mechanisms, and using larger input resolutions.

RetinaNet introduced the Focal loss that automatically adjusts the weights based on the loss magnitude. This loss replaces the standard cross-entropy loss function and assists the model in focusing on samples that are hard to recognize during the training process. Additionally, RetinaNet constructs the Feature Pyramid Network (FPN) [19] with multiple scales to effectively handle objects of various sizes.

Transformer-based algorithms (such as DEtection TRansformer (DETR) [20] and DETR with Improved DeNoising Anchor Boxes (DINO) [21]) introduce self-attention mechanisms and enable interactions between different positions of images. DETR successfully combines transformer frameworks with convolutional neural networks for feature extraction. It achieves a detection accuracy comparable to Faster RCNN. DINO introduces a contrastive noise training method that adds positive and negative samples of the same ground truth simultaneously. Additionally, a hybrid query selection method is proposed, which improves on the efficiency and performance of previous DETR models. Transformer-based algorithms can extract feature representations with contextual dependencies using multi-head attention mechanisms. They possess strong model capabilities and exhibit robustness to data noise, deformations, and other variations.

Among these, algorithms belonging to the YOLO family have been widely developed for practical applications due to their advanced detection accuracy and speed. Given YOLOv8's high real-time performance, accuracy, and scalability, this research uses the YOLOv8 model as its basic model.

2.3. Data Augmentation

Data augmentation is a strategy for effectively solving problems such as insufficient and imbalanced samples. It modifies the original samples to generate similar yet different samples, increasing the number of datasets so that the model can receive sufficient training and avoid overfitting.

Common methods for image data augmentation can be categorized into two groups: image transformation and image synthesis. Image synthesis-based methods include mixed images (represented by mixup [22]), feature space expansion, and use generative adversarial networks (GANs [23]).

The methods based on image transformation include flipping, cropping, random erasing [24], rotation, and color jitter. Image synthesis-based methods have a long training time and high computational cost. Image transformation-based methods do not require new data but may have limited diversity in the augmented samples. Targeted data augmentation methods need to be selected for different tasks.

For camouflaged object detection tasks, it is necessary to ensure that the augmented samples do not hinder the model from learning meaningful semantic features from the original samples.

3. Materials and Methods

3.1. YOLOv8 Model

YOLOv8 consists of three components: a backbone for extracting image features, a neck for fusing multi-level features, and a head for outputting classification and localization predictions.

The backbone includes the Conv, C2f, and SPPF modules. Two-stacked Conv modules are used to extract the initial features. The C2f module is a residual feature learning module that enriches the gradient flow of the model through cross-layer connections, resulting in a neural network module with a stronger feature representation capability. The SPPF module is an improvement on the Spatial Pyramid Pooling (SPP) module [25], which uses a combination of serial and parallel maximum pooling operations to amplify different receptive fields and output feature maps with adaptive sizes.

The neck network adopts a Path Aggregation Feature Pyramid Network (PAFPN) structure. It first propagates semantic information from deep-level feature maps in a top-down manner through FPN. Then, it propagates texture and detail information from the feature maps from lower levels to higher levels through a Path Aggregation Network (PANet) [26]. Finally, it outputs three feature maps with varying scales to the head module.

The head module decouples the classification and localization processes. It mainly includes loss computation and bounding box filtering. YOLOv8 adopts the positive-negative sample matching strategy of Task Aligned Assigner [27]. This strategy selects

positive samples according to scores, combined with classification and regression. The loss function consists of both Binary Cross Entropy (BCE) Loss and bounding box regression loss, which integrates Distribution Focal Loss [28] and Complete Intersection over Union (CIoU). When inferring the model to obtain bounding boxes, YOLOv8 uses an NMS algorithm to filter the predicted boxes.

There are multiple versions of the YOLOv8 model, namely YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. These versions vary in their depth, parameter size, and computational requirements, with the smallest being YOLOv8n and the largest being YOLOv8x. The "s" in YOLOv8s stands for "small," and we chose this version because it provides a faster detection speed and higher detection accuracy with limited computing cost.

3.2. Improved Network Architecture

We improved YOLOv8 by constructing the Edge Feature Enhancement Module (EFM) and introducing the use of a multi-branch convolution module called Receptive Field Block (RFB) [29]. Figure 2 shows the designed network, and the EFM is represented as a blue dashed box. For the convenience of description, this designed network will be referred to as EC-Net in the following text.

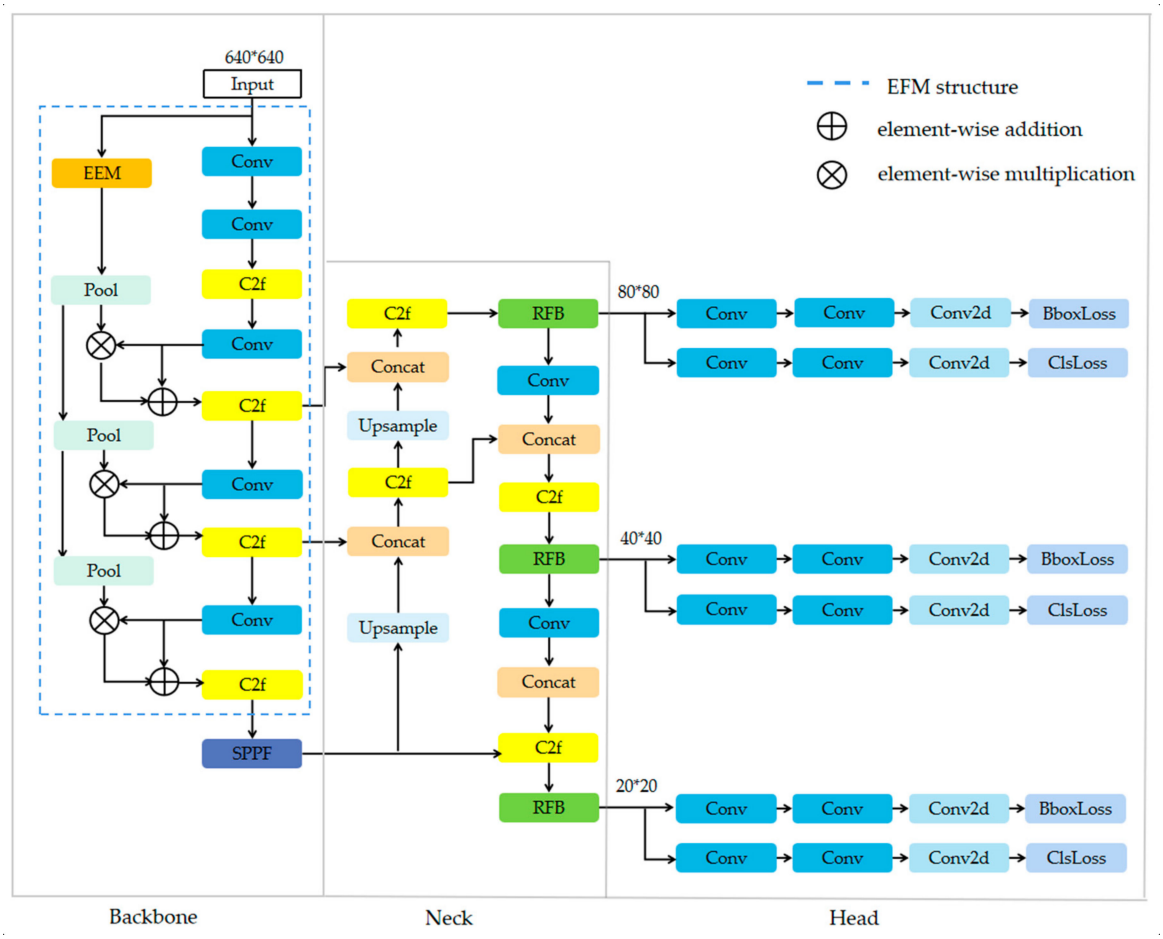


Figure 2. EC-Net.

3.2.1. Edge Feature Enhancement Module

The boundaries separating camouflaged objects from the background are blurry. To address this, edge features have been enhanced to highlight the object's boundaries, thereby improving the model's localization accuracy. The Scharr operator is employed to extract

the edges of camouflaged objects, and these extracted edges are then fused with the convolutional features.

The Scharr operator, also known as the Scharr filter, is an efficient and fine-grained edge detection operator. It calculates the image differences in the x or y direction to detect points with significant grayscale changes. The Scharr operator is calculated as follows:

$$f(i,j) = \sum_{i=0}^{k} \sum_{j=0}^{k} g(i,j) \otimes h(k-i, k-j) \quad (1)$$

where k represents the kernel size. To better capture edge features in various directions, we expanded the Scharr operator to four directions: 0°, 45°, 90°, and 135°. By performing gradient differencing using these directional templates within a 3×3 neighborhood, we can extract the edge information along each direction and then combine them. The templates used for these operators are depicted in Figure 3.

−3	0	3	−10	−3	0	−3	−10	−3	10	3	0
−10	0	10	−3	0	3	0	0	0	3	0	−3
−3	0	3	0	3	10	3	10	3	0	−3	−10
Scharr Operator in the 0° direction			Scharr Operator in the 45° direction			Scharr Operator in the 90° direction			Scharr Operator in the 135° direction		

Figure 3. Scharr operator kernel template.

Figure 4 shows the Edge Extraction Module (EEM). Within the EEM, the "Grayscale" operation averages the RGB image channels, reducing them to a single-channel matrix. The "SConv" operation convolves the image with the directional operator templates from Figure 3. The "S_fuse" operation employs a trainable convolutional layer to intelligently fuse the edge feature matrices by assigning appropriate weights.

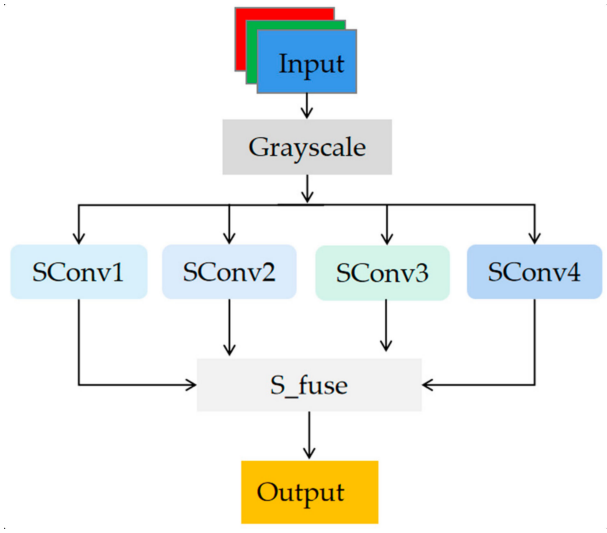


Figure 4. Edge Extraction Module (EEM).

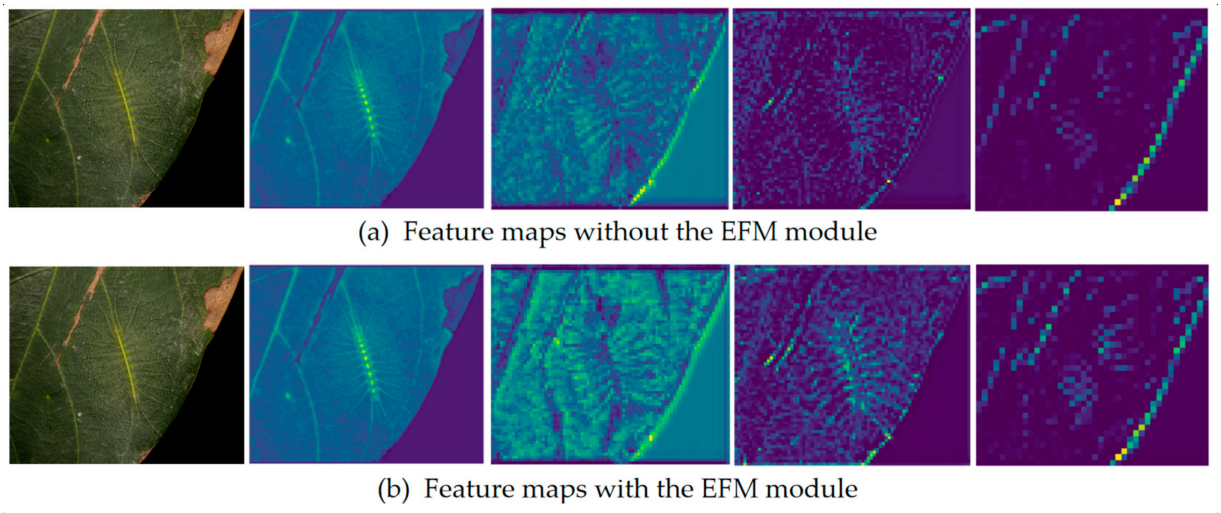
To enhance the learning of edge information within intermediate layers of the network, we constructed the Edge Feature Enhancement Module (EFM). We can integrate the edge images extracted by the EEM with the features acquired by the deep network.

We can then resize the edge images to match the size of the intermediate feature maps by downsampling them at different scales. The channel number of the edge images is

adjusted using a convolutional module, and then the values of the edge images are scaled to fall within the 0 to 1 range.

We can calculate the gradient and generate a weight map, which is element-wise multiplied by the intermediate feature maps of the network. The resulting map is added to the current intermediate feature map. Finally, the edge-enhanced feature map is passed through a residual module for further feature extraction.

Figure 5 visualizes the intermediate feature maps before and after edge enhancement. Column 1 shows the input image, Column 2 represents the input feature map, and Columns 3, 4, and 5 correspond to the feature maps obtained after passing through the second, third, and fourth C2f modules, respectively. Row (a) displays the maps of the model without edge enhancement, while row (b) shows the maps of the model after applying the proposed method for edge enhancement.



(a) Feature maps without the EFM module

(b) Feature maps with the EFM module

Figure 5. Visualization of the feature maps.

We can see that after incorporating the EFM into the network, the learned edge features of the feature maps are brighter in color, indicating that the edge features have been effectively enhanced.

3.2.2. Multi-Branch Convolution

Due to the variable scale and aspect ratio differences in camouflaged objects, single-scale features often fail to capture the details and contextual information of the targets comprehensively. The PAFPN network integrates features from different levels through top-down and bottom-up pathways.

However, simple feature fusion might not fully utilize the multi-scale information from different levels. To address this issue, a multi-branch convolution named the RFB module was introduced during the feature fusion stage. The RFB module improves the matching between the convolutional kernels and camouflaged objects by incorporating irregular convolutional kernels. Moreover, by employing dilated convolutions with different rates, the RFB module achieves a branch structure with diversified receptive fields. This hierarchical feature representation contributes to a more accurate expression of the characteristics of camouflaged objects, enhancing the model's adaptability to targets with various scales and aspect ratios.

Figure 6 shows the RFB module. Feature maps obtained from the previous network are fed into five distinct branches. In the beginning, four branches undergo 1×1 convolutions to reduce the number of channels. Subsequently, they undergo dilated convolutions and asymmetric convolutions to extract features at higher levels. Both the $\text{Conv}_{1\times3}$ and $\text{Conv}_{1\times5}$ asymmetric convolutions have a stride of 1, and padding is applied to ensure consistent feature map sizes. Ultimately, the features are merged along the channel dimensions.

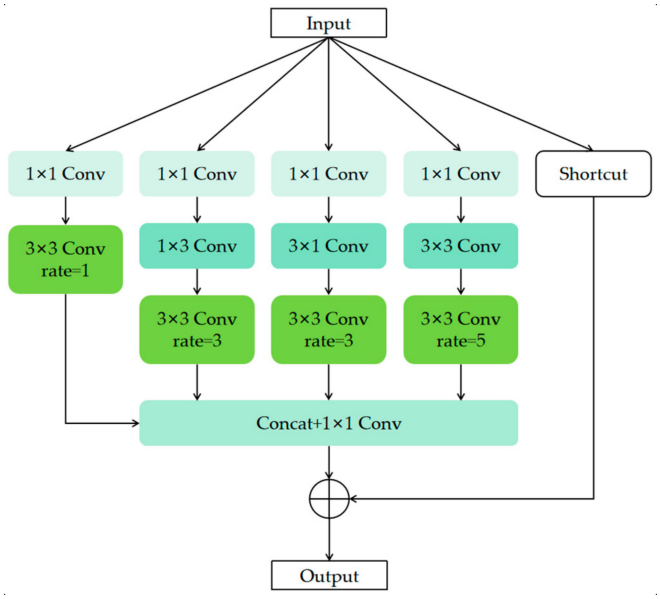


Figure 6. Receptive Field Block (RFB).

Additionally, one branch serves as a direct connection to the input. After merging the other branches, the merged features are element-wise added to the features from the direct connection branch.

3.3. Improved NMS Algorithm

The NMS algorithm is a commonly used post-processing technique for object detection. When parsing the output of the model into bounding boxes, there are often multiple overlapping boxes that localize the same object. The NMS algorithm first chooses the bounding box that has the highest classification score. Then, it compares the Intersection over Union (IoU) values of the other boxes with this box and deletes the boxes that have an IoU value higher than the threshold obtained with this box.

In camouflaged object detection, the rigid strategy used by NMS results in the erroneous removal of boxes for overlapping objects, thus increasing the model's missed detection rate. Furthermore, IoU does not consider either the size of the bounding box or the position information between borders, leading to the incorrect suppression of targets.

To address this issue, we replaced IoU with Efficient-IoU (EIoU) [30] when handling bounding boxes, and we adopted a soft strategy from Soft-NMS [31] to decrease the confidence scores of the overlapping boxes instead of outright discarding them. This approach allows for the preservation of more detection boxes containing overlapping objects. For ease of explanation, the enhanced algorithm is referred to as E-Soft-NMS.

E-Soft-NMS is calculated as:

$$s_i = \begin{cases} s_i, EIoU(M, b_i) < t \\ s_i e^{-\frac{EIoU(M,b_i)^2}{\sigma}}, EIoU(M, b_i) \geq t \end{cases} \quad (2)$$

In this formula, s_i means the confidence score, M is the box with maximum confidence, b_i is the other bounding box, and t is the threshold. When the EIoU value between b_i and M exceeds the threshold, we multiply the score of b_i with the decay function. If only the confidence score is lowered, there may be redundant boxes predicting the same target, which should be removed. To prevent its removal, a Gaussian decay function $e^{-\frac{EIoU(M,b_i)^2}{\sigma}}$ related to EIoU is used. This means that detection boxes far from M will not be affected, while those close to M will be penalized more. As a result, boxes predicting different targets are preserved, while redundant boxes predicting the same target are removed.

The equation for EIoU is given by:

$$EIoU = 1 - IoU + \frac{\rho^2(o, o*)}{c^2} + \frac{\rho^2(w, w*)}{{C_w}^2} + \frac{\rho^2(h, h*)}{{C_h}^2} \tag{3}$$

where o represents the center of b_i, $o*$ is the center of M, ρ is the Euclidean distance between o and $o*$, and w and $w*$ are the widths of b_i and M, respectively, while h and $h*$ represent the heights of b_i and M, respectively. c is the diagonal length of the minimum box B that encloses b_i and M. C_w and C_h are the width and height of B.

3.4. Shape-Enhanced Data Augmentation Method

Most objects that employ camouflage rely on deceptive textures that closely resemble their surroundings, resulting in weakened shape features. Hence, this study designed a shape-enhanced data augmentation approach called SPatch to enhance images. This method involves dividing the target within its mask into multiple texture patches and, subsequently, shuffling the spatial arrangement order of these patches.

Image scrambling is a technique used in information hiding. It includes Pixel scrambling [32] and block scrambling [33], typically applied to the entire image. We scrambled the foreground area of an image. Moreover, a dynamic adjustment method for the scrambling region size is proposed, where the scale of scrambling is dynamically determined based on the number of target pixels.

By utilizing segmentation annotations provided by the dataset, we obtained the target mask and segmented the image within the mask. The foreground region can be flexibly divided into texture patches of size N, depending on the ratio of target pixels in the entire image. N is calculated as follows:

$$N = \left\lceil \sqrt{\frac{a}{s}} * 100 * k \right\rceil \tag{4}$$

where a is the pixel count in the foreground, s is the total pixels in the image, k is the dynamic scrambling magnitude (experimentally, k is set to 0.1 for the best result), and "$\lceil\ \rceil$" denotes rounding up.

During the dividing process, accurately cutting the regions involving edges poses a challenge. To address this, a method for calculating the edge proportion is proposed. The specific implementation is as follows: based on the coordinates of the patch in the entire image, a region of the same size and position within the mask is selected. Then, we calculate the proportion of non-zero pixels in the current region:

$$T = \frac{\sum\limits_{m \in R} m | m > 0}{W_R * H_R} \tag{5}$$

where m represents the pixel in the region R, $m > 0$ indicates the current pixel belongs to the foreground region, H_R denotes the height of R, and W_R is the width of region R. If T exceeds the threshold of 0.96, the current patch is added to the scrambling pool, and the position of each patch is recorded. Subsequently, the order of patches in the pool is shuffled, and then the patches are extracted from the scrambled pool and placed into the target region. Figure 7 depicts an example of SPatch augmentation, and the specific procedure is detailed in Algorithm 1, where I means pixels and (i, j) denotes the top left coordinate of the current patch.

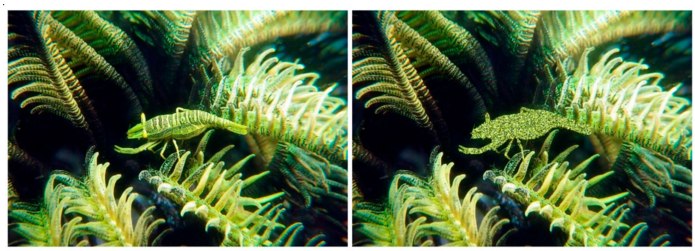


Figure 7. Example of the SPatch augmentation method.

When applying shape enhancement, the camouflage characteristics of the target should not be compromised, as it may cause the model to be biased toward recognizing salient objects. In this regard, we validated the augmented samples on a model trained on the original training sets. For the COD dataset, the model attained a detection accuracy of 62% on the augmented samples and 62.2% on the original samples. As for the CAMO dataset, the detection accuracy on the augmented samples and original samples was 54.2% and 54.9%, respectively. The difference in accuracy before and after augmentation was less than 1% for both datasets. Hence, the samples generated by the SPatch augmentation method exhibit similar features to the original camouflaged objects. Using this method does not introduce poisonous samples.

Algorithm 1. SPatch

Input: Image X, Array E for storing patches, Array P for recording positions, *mask*, Amplitude k
Output: $\widetilde{X}$
1. $E = P = \varnothing$, $\widetilde{X} = X$, $Q = X \odot mask$
2. $a = \sum_{I \in mask} I | I > 0$, $s = W_X * H_X$
3. $N = \left\lceil \sqrt{\frac{a}{s}} * 100 * k \right\rceil$
4. while patch (i, j) $\in$ Q
5. $\quad R = mask(i, j)$
6. $\quad T = \frac{\sum_{m \in R} m | m > 0}{W_R * H_R}$
7. $\quad$ While T > 0.96
8. $\quad E = E \cup \left\{ patch_{(i,j)} \right\}$
9. $\quad P = P \cup \{(i, j)\}$
10. end while
11. end while
12. E = Shuffle (E)
13. while t $\in$ E and (i, j) $\in$ P do
14. $\widetilde{X}_{i,j} = t$
15. end while
16. return $\widetilde{X}$

4. Experiment

4.1. Setup

4.1.1. Datasets

The experiment utilized two major publicly available camouflaged object datasets, COD10K [1] and CAMO [34], for training and testing. These datasets contain a large number of sample categories, with a severe class imbalance where some categories have fewer than 10 samples. Therefore, we divided each dataset into two categories: cam and background. Additionally, the segmentation annotations were converted into bounding box annotations. The distribution of the datasets is shown in Table 1.

Table 1. Sample division of the datasets.

Dataset	Samples	Category	Training Samples	Testing Samples
COD10K	5050	Cam/Background	4035	1015
CAMO	1250	Cam/Background	1000	250

4.1.2. Evaluation Criteria

We utilized the mean Average Precision (mAP) as the evaluation metric, which reflects the average accuracy of the model's detection across different categories.

The detection model's predictions can be categorized as True Positive (TP), True Negative (TN), False Positive (FP), or False Negative (FN) based on their correctness. The model's precision p for a category is the ratio of TP to the sum of TP and FP, and the recall r is the ratio of TP to the sum of TP and FN.

We employed the 11-point interpolation [35] method to calculate the AP for a specific category:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\widetilde{r}: \widetilde{r} > r} p(\widetilde{r}) \tag{6}$$

Given 11 recall values r, we need to identify the maximum precision value p for all $\widetilde{r}$ values greater than r, and then calculate the average of these values. The mAP value is the ratio of the sum of the AP values for each category to the number of categories.

In addition, we used three other metrics, including the Frames Per Second (FPS), Parameters (Params), and Floating Point Operations Per Second (FLOPs) to measure the detection speed, network size, and computational cost of the algorithm, respectively.

4.1.3. Details

We implemented all experiments under the PyTorch 1.8.1 framework. The workstation was configured with Nvidia GeForce RTX 2080Ti 12G GPU. During the training process, we set the batch size to 32. The momentum of SGD was 0.937, and the weight decay was 5×10^{-4}. The learning rate was initialized to 0.01. Mosaic-4 data augmentation was turned on during training. We ran 500 epochs in each training session and took the average results from 5 runs as the outcome.

4.2. Comparison with the State-of-the-Arts

We evaluated our method by comparing it with other advanced approaches in object detection, including Faster R-CNN [9], RepPoints [10], RetinaNet [13], YOLOv5s [15], Sparse R-CNN [11], YOLOv6s [16], YOLOv7 [17], DINO [21], and YOLOv8s [18].

Figure 8 compares the detection results of our approach with existing mainstream methods on camouflaged data. It shows that the original YOLOv8 model tends to miss small objects (Column 6) and objects with blurry edges (Column 3). Moreover, it has inaccurate localization for camouflaged objects (Columns 1, 2, 4, and 5).

Meanwhile, the proposed method improved the localization accuracy by enhancing edge and shape features. It introduced a multi-branch convolutional module to enhance adaptability to objects of different sizes. The improved post-processing algorithm effectively reduced the false-negative rate.

Table 2 presents a comparison of mAP values across different models. Our model achieved the highest mAP value on CAMO, and it achieved joint first place with Sparse R-CNN in terms of detection performance on COD10K. We compared the parameters of our model with those of Sparse R-CNN, which achieved the second-best detection performance. Our model had less than 1/7 of the parameter size of Sparse R-CNN.

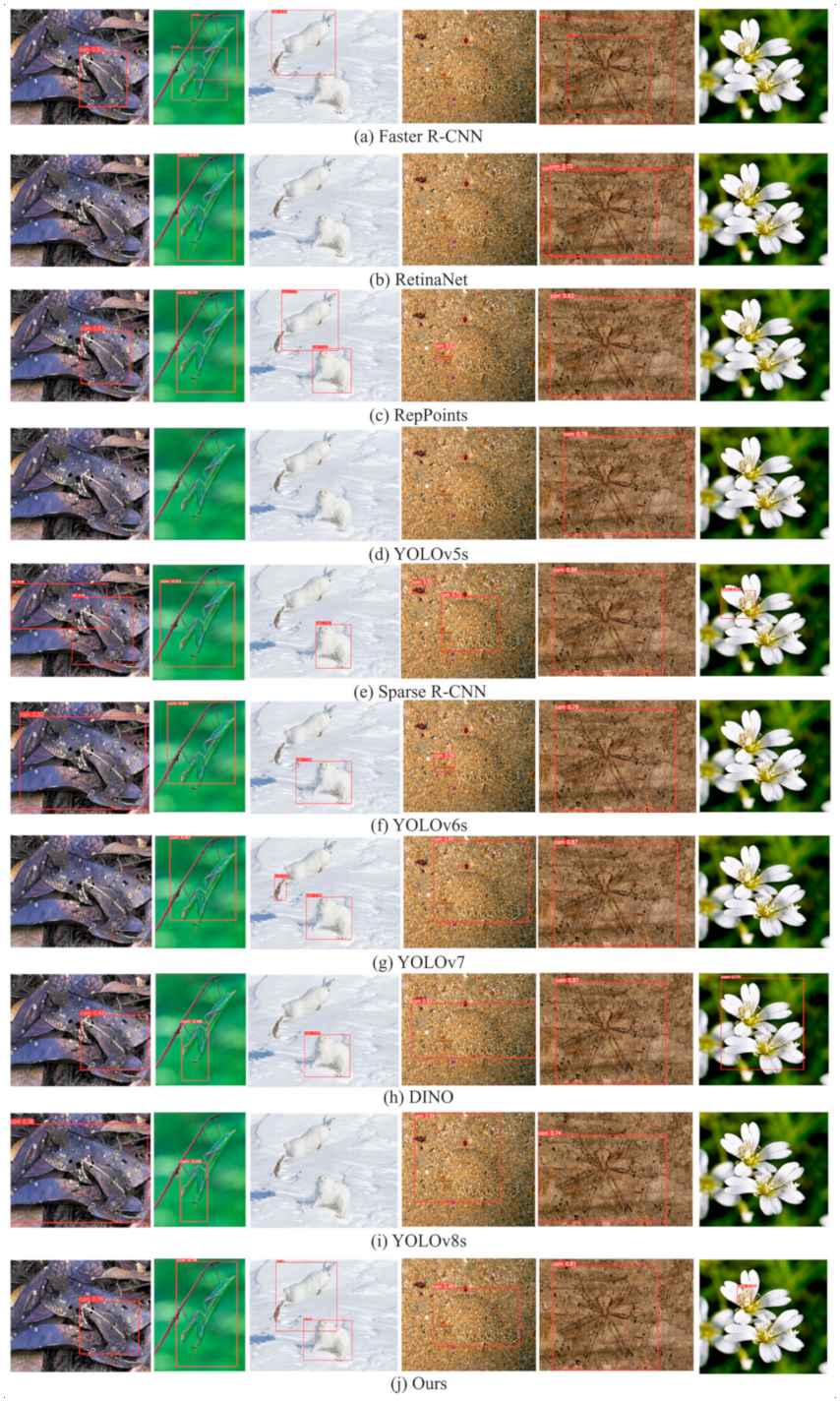


Figure 8. Detection results of different models.

From the metrics of the FLOPs and FPS, it can be observed that YOLOv5s requires the least computational power and exhibits the fastest inference speed. YOLOv8s follows in terms of computational cost and running speed. Our approach has slightly increased computational cost and running speed compared to the original YOLOv8s, but it still maintains comparable performance with other models.

Table 2. Quantitative results of various models.

Model	Year	COD10K	CAMO	Params (M)	FLOPs (G)	FPS
Faster R-CNN	NeurIPS2015	0.579	0.50	41.8	270	20
RetinaNet	ICCV2017	0.55	0.465	32.2	254	23
RepPoints	ICCV2019	0.628	0.523	35.6	152	19
YOLOv5s	2020	0.576	0.511	7.2	15.8	136
Sparse R-CNN	CVPR2021	0.705	0.617	98.2	128.2	23
YOLOv6s	2022	0.614	0.533	16.4	44.7	104
YOLOv7	2022	0.628	0.579	37.2	103.1	72
DINO	ICLR2023	0.667	0.596	45.1	238	24
YOLOv8s	2023	0.622	0.549	11.2	28.8	108
Ours		0.705	0.64	13.1	32.3	75

4.3. Ablation Study

To validate each crucial component, we designed several ablation experiments. We report the mAP of the models in Table 3, where "√" indicates the use of a certain module. We used YOLOv8s as the baseline model.

Table 3. Evaluation in ablation studies of the two datasets.

EC-Net		E-Soft-NMS	SPatch	COD10K	CAMO
RFB	EFM				
				0.622	0.549
√				0.64	0.583
	√			0.637	0.565
		√		0.671	0.603
√	√			0.65	0.594
	√	√		0.682	0.617
√		√		0.676	0.62
√	√	√		0.686	0.632
√	√	√	√	0.705	0.64

When the EFM and RFB modules were respectively incorporated into the baseline model during training, the mAP of the model increased. Furthermore, after applying the E-Soft-NMS algorithm, the mAP value improved significantly. When combining the EFM, RFB, and E-Soft-NMS, the detection performance on the COD10K and CAMO datasets improved by 6.4% and 8.3%, respectively. Finally, the training sets were augmented with the SPatch method to enhance the shape features of the camouflaged objects, further improving the model performance.

4.3.1. Network

To valid the effectiveness of EC-Net, comparative experiments were conducted on the COD10K and CAMO datasets with the existing YOLOv8 algorithm. We compared the mAP, recall rate (r), and parameter count (params) of different networks. We report them in Table 4.

Table 4. Ablation study using various networks.

Net	COD10K		CAMO		Params (M)
	mAP	r	mAP	r	
YOLOv8s	0.622	0.555	0.549	0.526	11.2
YOLOv8m	0.668	0.61	0.571	0.568	25.9
EC-Net	0.65	0.604	0.594	0.569	13.1

The comparative results indicate that YOLOv8s has fewer parameters, while YOLOv8m has more parameters, and the performance of YOLOv8m surpasses that of YOLOv8s. The proposed EC-Net achieves performance comparable to YOLOv8m while utilizing only half the number of parameters.

EC-Net incorporates the EFM module, which effectively enhances the edge features of camouflaged objects and enables the model to capture more detailed information. The introduced RFB module improves the model's perception ability for objects with different aspect ratios at a lower parameter cost by using multiple irregular convolution branches.

4.3.2. Non-Maximum Suppression Algorithm

To evaluate the effectiveness of our enhanced NMS algorithm, we utilized YOLOv8s as the base model and compared the performance of different NMS algorithms while maintaining the same network architecture. The comparison results can be found in Table 5.

Table 5. Ablation study using different NMS algorithms.

Algorithm	COD10K		CAMO	
	mAP	**r**	**mAP**	**r**
NMS	0.622	0.555	0.549	0.526
EIoU-NMS	0.644	0.571	0.578	0.534
Soft-NMS	0.662	0.599	0.59	0.572
E-Soft-NMS	0.671	0.607	0.603	0.57

The comparison results indicate that using EIoU instead of IoU or Soft-NMS improved the model. The proposed E-Soft-NMS algorithm achieved better performance compared to both methods. This is because EIoU takes into account the size of the bounding box and the position information between the borders, which improves the localization accuracy of the model. Soft-NMS uses a decay function to decrease the confidence scores of overlapping boxes instead of directly discarding them, preserving more detection boxes of overlapping objects and significantly reducing the miss rate. The proposed E-Soft-NMS algorithm combines the advantages of both methods, resulting in further improvement of the model performance.

4.3.3. Data Augmentation

Since the SPatch algorithm belongs to traditional methods based on image transformation, it can be compared with traditional augmentation methods. We used the SPatch augmentation method and traditional augmentation methods to augment datasets separately. We trained the YOLOv8s on the augmented training sets and then evaluated the well-trained model on the original test sets. Table 6 shows the mAP of models trained with different methods, where "-" represents the original model, "flip" denotes horizontal flipping, "rotate" represents random rotation, "crop" signifies random cropping, "swap" indicates channel swapping, "erasing" represents random erasing [22], and "cj" denotes color jitter.

According to Table 6, the SPatch augmentation method achieved the highest mAP. It improved the mAP values on the COD10K dataset by 2.2% and on the CAMO dataset by 1.9%. Horizontal flipping facilitates the model in learning features related to the symmetry and invariance of camouflaged objects. Color jitter introduces random color variations to the images, helping the model learn invariance to lighting changes. Both of these augmentations improved the mAP of the model on camouflaged objects. Channel shuffling introduces significant differences between augmented samples and the original samples, which leads to a decrease in accuracy. Random rotation may make it hard for the model to recognize the shape and structure of the target, especially for targets with complex or elongated shapes, resulting in a decline in detection performance. Random erasing simulates

occlusion situations. However, since camouflaged objects are inherently concealed, this augmentation method has little impact on model performance.

Table 6. Ablation study using different augmentation methods.

Method	COD10K	CAMO
-	0.622	0.549
SPatch	0.644	0.568
flip	0.64	0.552
rotate	0.617	0.538
crop	0.63	0.534
swap	0.611	0.538
erasing	0.629	0.525
cj	0.63	0.552

The proposed SPatch augmentation method preserves the same contours as the original samples while changing the internal structure of the camouflaged objects by partitioning and combining patches. This effectively alters the highly similar textures between the targets and backgrounds, highlights the shape boundaries, and increases the distinguishability between the targets and backgrounds, leading to significant improvement in model performance.

4.4. Limitations

Our method offers a significant improvement in detecting camouflaged objects. However, it still has several limitations. As depicted in Figure 9, if the targets lack distinctive edges and their texture features closely resemble those of the background, our method may experience localization errors and omissions. Additionally, our method tends to mistake local salient textures within the target as separate objects (Column 1 and Column 5). We also encountered difficulties in inferring the global shape of occluded targets (Column 2) and distinguishing between real targets and their reflections on a water surface (Column 3). Another issue we faced is that, due to edge enhancement of feature maps in the middle layers of the network, our method may incorrectly identify background regions with stronger edge features as targets (Column 4).

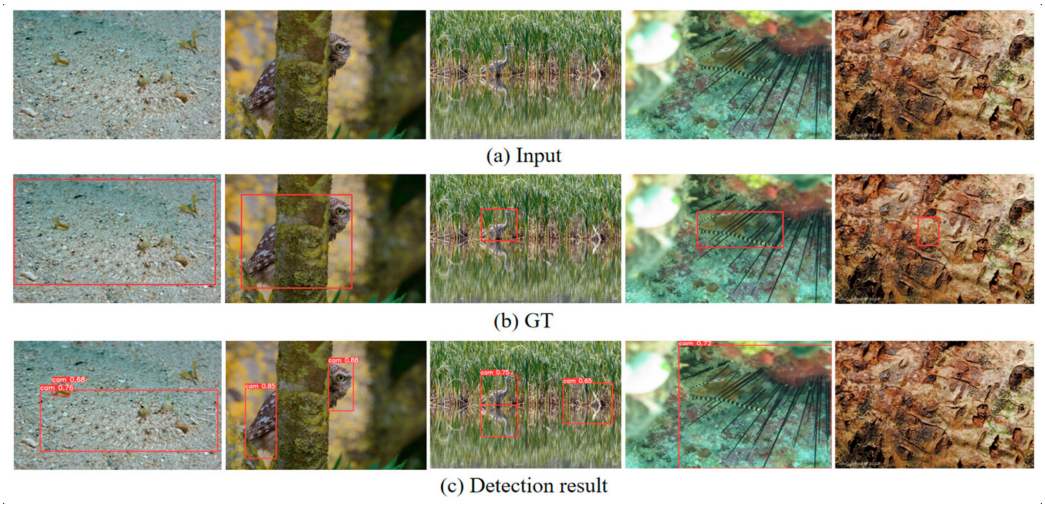


Figure 9. Failure cases.

The enhancement of edge features also increases background noise to some extent, so further research is needed to suppress irrelevant edge features. As many camouflaged objects possess periodic textured structures, the periodicity of texture can be utilized to

discover camouflaged objects hidden in the background. In future research, exploring methods to enhance texture features can be considered.

5. Conclusions

This study utilized rectangle-based object detection techniques for the first time to achieve camouflaged object detection. To address the challenges in camouflaged object detection, improvements were made to YOLOv8. By designing a network that includes an Edge Feature Enhancement Module and a Receptive Field Block, feature enhancement was achieved. The NMS algorithm was improved to address the issue of missed detections for overlapping targets. Additionally, an augmentation method is proposed to enhance the model's ability to learn the shapes of camouflaged objects. The experimental results indicate a substantial enhancement of the performance of the proposed method relative to the original model.

Author Contributions: Conceptualization, T.H.; methodology, T.H.; validation, Y.Z. and B.F.; writing—original draft preparation, T.H.; writing—review and editing, T.C., Y.Z., L.C. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No.61801512, No.62071484), and the Natural Science Foundation of Jiangsu Province (BK20180080).

Data Availability Statement: This study used open data sources.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fan, D.P.; Ji, G.P.; Sun, G.; Cheng, M.M. Camouflaged object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 2777–2787.
2. Qin, X.; Fan, D.P.; Huang, C.; Diagne, C.; Zhang, Z.; Sant'Anna, A.C.; Suarez, A.; Jagersand, M.; Shao, L. Boundary-aware segmentation network for mobile and web applications. *arXiv* **2021**, arXiv:2101.04704.
3. Zhai, Q.; Li, X.; Yang, F.; Chen, C.; Cheng, H.; Fan, D.P. Mutual graph learning for camouflaged object detection. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 12997–13007.
4. Li, A.; Zhang, J.; Lv, Y.; Liu, B.; Zhang, T.; Dai, Y. Uncertainty-aware Joint Salient Object and Camouflaged Object Detection. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10066–10076.
5. Yan, J.; Le, T.N.; Nguyen, K.D.; Do, T.T.; Nguyen, T.V. Mirrornet: Bio-inspired camouflaged object segmentation. *IEEE Access* **2021**, *9*, 43290–43300. [CrossRef]
6. Mei, H.; Ji, G.P.; Wei, Z.; Yang, X.; Wei, X.; Fan, D.P. Camouflaged Object Segmentation with Distraction Mining. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 8768–8777.
7. Lin, J.; Tan, X.; Xu, K.; Ma, L.; Lau, R.W. Frequency-aware camouflaged object detection. *ACM Trans. Multimed. Comput. Commun. Appl.* **2023**, *19*, 1–16. [CrossRef]
8. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
9. Ren, S.Q.; He, K.M.; GIRSHICK, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
10. Yang, Z.; Liu, S.; Hu, H.; Wang, L.; Lin, S. RepPoints: Point Set Representation for Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9656–9665.
11. Sun, P.; Zhang, R.; Jiang, Y.; Kong, T.; Xu, C.; Zhan, W.; Tomizuka, M.; Li, L.; Yuan, Z.; Wang, C. Sparse R-CNN: End-to-End Object Detection with Learnable Proposals. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 14449–14458.
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
13. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *42*, 2999–3007.
14. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
15. YOLOv5 Code. Available online: https://github.com/ultralytics/yolov5 (accessed on 1 May 2023).

16. Li, C.; Li, L.; Jiang, H. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
17. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
18. YOLOv8 Code. Available online: https://github.com/ultralytics/ultralytics (accessed on 1 August 2023).
19. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
20. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
21. Zhang, H.; Li, F.; Liu, S.; Zhang, L.; Su, H.; Zhu, J.; Ni, L.M.; Shum, H.-Y. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv* **2022**, arXiv:2203.03605.
22. Zhang, H.Y.; Cissé, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–13.
23. Goodfellow, I.J.; Pouget-ABADIE, J.; Mirza, M. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 8–13 December 2014; pp. 2672–2680.
24. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random Erasing Data Augmentation. *Proc. AAAI Conf. Artif. Intell.* **2017**, *34*, 13001–13008. [CrossRef]
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
26. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
27. Feng, C.; Zhong, Y.; Gao, Y.; Scott, M.R.; Huang, W. Tood: Task-aligned one-stage object detection. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 3490–3499.
28. Li, X.; Lv, C.; Wang, W.; Li, G.; Yang, L.; Yang, J. Generalized focal loss: Towards efficient representation learning for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 3139–3153. [CrossRef]
29. Liu, S.; Huang, D.; Wang, Y. Receptive Field Block Net for Accurate and Fast Object Detection. In Proceedings of the Computer Vision—ECCV 2018: 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 404–419.
30. Zhang, Y.F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and efficient IOU loss for accurate bounding box regression. *Neurocomputing* **2022**, *506*, 146–157. [CrossRef]
31. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Soft-NMS—Improving Object Detection with One Line of Code. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5562–5570.
32. Li, S.; Li, C.; Chen, G.; Bourbakis, N.G.; Lo, K.T. A general quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks. *Signal Process. Image Commun.* **2008**, *23*, 212–223. [CrossRef]
33. Qu, L.; Chen, F.; He, H.Y.Y.; Yuan, Y. Security Analysis of Image Encryption Algorithm Based on Bit Plane-Pixel Block Scrambling. *J. Appl. Sci.* **2019**, *37*, 631–642.
34. Le, T.N.; Nguyen, T.V.; Nie, Z.; Tran, M.-T.; Sugimoto, A. Anabranch network for camouflaged object segmentation. *Comput. Vis. Image Underst.* **2019**, *184*, 45–56. [CrossRef]
35. Padilla, R.; Netto, S.L.; Da, S.E. A survey on performance metrics for object-detection algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and IMAGE Processing (IWSSIP), Niterói, Brazil, 1–3 July 2020; pp. 237–242.