DFLM-YOLO: A Lightweight YOLO Model with Multiscale Feature Fusion Capabilities for Open Water Aerial Imagery

Chen Sun , Yihong Zhang * , Shuai Ma

Posted Date: 16 July 2024

doi: 10.20944/preprints202407.1302.v1

Keywords: lightweight; multiscale feature fusion; data augmentation; UAV; object detection

Article

GFLM-YOLO: A Lightweight YOLO Model with Multiscale Feature Fusion Capabilities for Open Water Aerial Imagery

Chen Sun, Yihong Zhang * and Shuai Ma

College of Information Science and Technology, Engineering Research Center of Digitized Textile & Fashion Technology, Ministry of Education, Donghua University, Shanghai 201620, China; 2232203@mail.dhu.edu.cn(C.S.); 2232075@mail.dhu.edu.cn(S.M.)

* Correspondence: zhangyh@dhu.edu.cn; Tel.: +86-138-1782-6259

Abstract: Object detection algorithms for open water aerial images present challenges such as small object size, unsatisfactory detection accuracy, numerous network parameters, and enormous computational demands. Current detection algorithms struggle to meet the accuracy and speed requirements while being deployable on small mobile devices. This paper proposes GFLM-YOLO, a lightweight small-object detection network based on the YOLOv8 algorithm with multiscale feature fusion. Firstly, to solve the class imbalance problem of the SeaDroneSee dataset, we propose a data augmentation algorithm called Small Object Multiplication (SOM). SOM enhance dataset balance by increasing the number of objects in specific categories, thereby improving model accuracy and generalization capabilities. Secondly, we optimize the backbone network structure by implementing Depthwise Separable Convolution (DSConv) and the newly designed FC-C2f, which reduces the model's parameters and inference time. Finally, we design the Lightweight Multiscale Feature Fusion Network (LMFN) to address the challenges of multiscale variations by gradually fusing the four feature layers extracted from the backbone network in three stages. In addition, LMFN incorporates the Dilated Re-param Block structure to increase the effective receptive field and improve the model's classification ability and detection accuracy. Experimental results on the SeaDroneSea dataset indicate that GFLM-YOLO improves mAP by 12.4% compared to the original YOLOv8s, while reducing parameters by 67.2%. This achievement provides a new solution for UAVs to conduct object detection missions in open water efficiently.

Keywords: lightweight; multiscale feature fusion; data augmentation; UAV; object detection

1. Introduction

Object detection in open waters is a significant branch of the field, with crucial implications for maritime navigation, marine search and rescue, marine environment monitoring, and national defense. The primary challenge in open water search and rescue is quickly identifying individuals in distress[1]. Traditional methods primarily rely on surface vessels and helicopters, which require significant manpower and resources. These methods also have low search and rescue efficiency and limited scope, resulting in delayed rescue and loss of life in maritime accidents. With the advancement of UAV technology, UAVs are extensively utilized in various fields such as agricultural production, wildfire prevention, disaster relief, and other fields due to their strong mobility, low cost, and wide angle of view[2–4]. By integrating object detection algorithms based on deep learning with UAVs, UAVs can achieve autonomous object detection, reducing the need for manpower and time in search and rescue operations and significantly improving rescue efficiency[5].

The COCO dataset distinguishes whether an object is small or not based on the absolute scale of the goal, where a goal with a resolution of less than 32 pixels by 32 pixels is considered to be small[6]. To expand the scope of search and rescue and enhance efficiency, UAVs are generally farther away

 doi:10.20944/preprints202407.1302.v1

from the sea surface when performing search and rescue missions, resulting in aerial images predominantly containing small objects. In addition, the complexity environment of open water, the variation of object scales, and the limited computational resources of the airborne computation device also impose higher demands on the object detection algorithms.

In recent years, deep learning-based object detection algorithms have rapidly developed and achieved significant results, successfully applied across various fields. For instance, the Transformer-based RT-DETR algorithm[7] and the YOLOv9[8] and YOLOv10[9] algorithms from the YOLO series proposed in 2024 have demonstrated strong performance in typical scenarios. However, these state-of-the-art general-purpose object detection algorithms struggle to maintain high detection performance in open water search and rescue missions. The primary reason is that the limited pixel information of the object and easily interfered with by the background, resulting in the model not being able to effectively extract features such as edges, texture, and color of small objects. For instance, YOLOv10s achieved an mAP50 score of only 63.9% on the SeaDroneSee validation set.

These general-purpose object detection algorithms have large parameters and complex model structure, demanding higher computational capabilities from the airborne computation device. However, during open water search and rescue missions, drones' endurance, detection speed, and cost constraints mean they cannot carry high-computation chips. Furthermore, object detection algorithms for open water must balance detection speed with accuracy. Therefore, it is necessary to further improve and optimize the existing object detection algorithms to enhance their applicability and performance.

To address the challenges encountered in open water search and rescue missions, this paper proposes a new lightweight multiscale feature fusion network, GFLM-YOLO. This network reduces model parameters and increases detection speed while enhancing the detection accuracy for small objects. Compared to the YOLOv8s, the improved model’s mAP50 increased by 12.4%, model parameters decreased from 11.17 million to 3.65 million (a 67.3% reduction), Floating Point of Operations (FLOPs) decreased from 28.7 to 23.3 (an 18.8% reduction), and detection time decreased from 2.6ms to 2.3ms (an 11.5% reduction). This model outperforms other advanced small object detection algorithms in both speed and accuracy. The main contributions of this paper are as follows:

1. The paper introduces a new data augmentation algorithm called SOM, which aims to expand the number of objects in specific categories without adding actual objects. This algorithm ensures that the characteristics of the added objects remain consistent with the original ones. Experiments demonstrate that this method enhances dataset balance and improves the model's accuracy and generalization capabilities.
2. Lightweight design of the backbone network:

a: Utilized depthwise separable convolutions as the feature extraction module in the backbone network, reducing model parameters, computation required for convolution operations, and network inference latency.

b: A new plug-and-play module, FC-C2f, was designed to optimize the backbone network structure, reduce computational redundancy, and lower the model's parameters and FLOPs.

3. Designed a new multiscale fusion network, LMFN, to address the accuracy issues in multiscale object recognition.

a: By gradually integrating features from different levels, the connections between layers are effectively increased, and the model’s feature fusion process is optimized. This enhances the model's capability to fuse multiscale features, improving detection accuracy for objects of various scales.

b: Combined dilated convolutions with small kernels and cascaded convolutions into a re-parameterized large kernel convolution. This approach retains the benefits of small kernels, such as reduced computational load and fewer parameters, while achieving the large effective receptive field of large kernels. Experimental results demonstrate that this structure reduces model parameters and increases the receptive field.

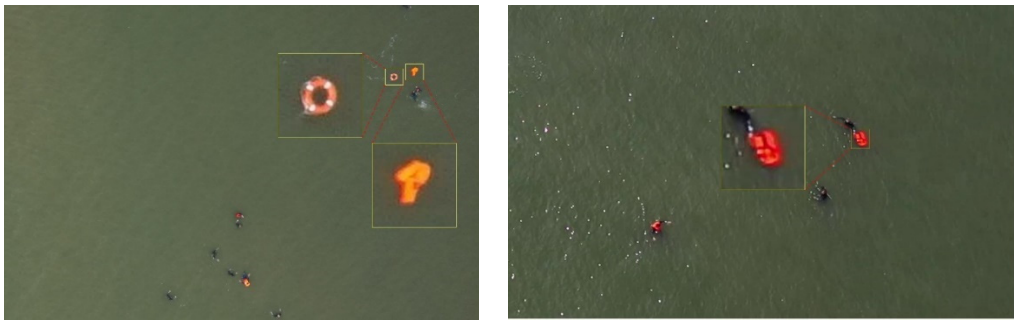
 doi:10.20944/preprints202407.1302.v1

2. Related Work

2.1. Data Augmentation

Depending on the application scenarios and the characteristics of different datasets, using appropriate data augmentation algorithms can enhance the model's detection accuracy and robustness. In some datasets, the number of samples in certain classes may be significantly higher than in others, a phenomenon known as class imbalance. Class imbalance adversely affects the model's convergence speed and detection accuracy, impacting both training convergence and generalization on the test set. To address class imbalance in the training set, the most common methods can be divided into two categories: data-level methods that directly manipulate the dataset to change class distribution and classifier-level methods. The most common data-level methods are oversampling and undersampling. Oversampling works by duplicating samples from minority classes, while undersampling involves removing samples from majority classes. Since the YOLO algorithm requires a large amount of training data to fit the network, this paper employs the oversampling method for data augmentation.

Chen et al. proposed a multi-sample data augmentation method for remote sensing images called SSMup[10], which integrates three data augmentation techniques: Mosaic, Mixup, and SSMOTE. This algorithm ensures uniform distribution of objects in the enhanced samples and provides rich background information. Zhang et al. proposed a novel grid-oversampling strategy[11]. This strategy first uses the OSTU algorithm for feature extraction before cropping, then crops the image using a sliding window, and finally retains only objects that occupy more than 80% of the sliding window's foreground. This approach accelerates the detection speed of sparse object images. In the SeaDroneSea dataset, the features of the objects may be very different, even for the same labeled categories. This strategy can speed up sparse object detection in sparse object images. In the SeaDroneSea dataset, even objects with the same label category can have vastly different characteristics. As shown in **Figure 1**, both images have objects labeled as "life_saving_appliances," but in **Figure 1(a)**, the objects are a lifebuoy and a float board, while in **Figure 1(b)**, the object is a life jacket. Therefore, simply using the Simple Copy-Paste[12] data augmentation algorithm to increase the number of objects is unlikely to yield satisfactory results. Additionally, the marine environment is complex and variable, with significant background differences between images. Copy-paste operations across multiple images can easily lead to image distortion due to large discrepancies between the pasted region and the original image background. Therefore, this paper proposes a data augmentation algorithm called Small Object Multiplication (SOM), which balances the dataset by increasing the number of small objects in specific categories, addressing the issue of class imbalance.



(**a**) (**b**)

Figure 1. Sample images for "life_saving_appliances": (a) a lifebuoy and a float board, (b) a life jacket.

2.2. Lightweight Methods for Object Detection Networks Based on Deep Learning

In open water search and rescue missions, it is crucial to make lightweight improvements to existing object detection models due to the limited power consumption and computational resources available of UAVs. Lightweight network architectures originated from SqueezeNet[13] in 2016 and

 doi:10.20944/preprints202407.1302.v1

MobileNet[14] in 2017. These networks have since seen multiple improved versions, such as SqueezeNext[15] and MobileNetV2-4[16–18]. SqueezeNet introduces the Fire module, which uses multiple 1×1 convolution kernels connected instead of 3×3 convolution kernels. By adjusting the number of 1×1 convolutions, the number of channels in each layer can be flexibly controlled, reducing both model parameters and computational load. The MobileNet series algorithms proposed depthwise separable convolution, dividing standard convolution operations into two processes: depthwise convolutions followed by pointwise convolutions. This approach significantly reduces parameters and computational load, accelerating both model training and inference speed.

In addition to designing lightweight networks, many researchers have focused on creating lightweight designs for specific parts of existing object detection networks. Zhang et al. introduced a lightweight improvement to the detection head, proposing a Lightweight Asymmetric Detection Head (LADH-Head)[19]. LADH-Head utilizes depthwise separable convolution to optimize the Asymmetric Decouple Head (ADH), significantly reducing model parameters while improving detection accuracy. Wenkai Gong has proposed a lightweight feature extraction module called Dynamic Group Convolution Shuffle Transformer (DGST) to further enhance computational efficiency and performance[20]. DGST incorporates group convolution to reduce model parameters and computational load while preventing overfitting. It also integrates the channel shuffle technique from ShuffleNetV2 to promote effective inter-group feature information exchange. Additionally, DGST incorporates Vision Transformer, further improving computational efficiency and performance. Wang et al. introduced a plug-and-play feature upsampling module Content-Aware ReAssembly of FEatures (CARAFE)[21]. Instead of using a fixed kernel, CARAFE can dynamically generate adaptive kernels by performing content-aware processing on specific instances. In addition, CARAFE has a low computational overhead and can be easily integrated into other network architectures.

2.3. MultiScale Feature Fusion

In object detection algorithms, the neck network is commonly utilized to combine different layers of feature maps extracted from the backbone network. This fusion generates feature maps with multiscale information, which enable to enhances the accuracy of object detection. The introduction of Feature Pyramid Networks (FPN) in 2016 was a milestone, addressing the shortcomings of detection networks in handling multiscale variations[22]. Building on FPN, Liu et al. proposed the Path Aggregation Network (PANet)[23]. PANet enhances the feature hierarchy with a bottom-up path, leveraging precise positional information from lower-level feature layers to strengthen higher-level feature layers.

Xu et al. proposed an efficient Reparameterized Generalized-FPN (RepGFPN)[24]. The RepGFPN enhances feature interaction through queen-fusion and reduces the extra upsampling in post-fusion to decrease model complexity. It also introduces a reparameterization mechanism and efficient layer aggregation networks (ELAN) to upgrade the feature fusion module, achieving higher accuracy without additional computational burden. Tan et al. designed a Bi-Directional Feature Pyramid Network (BiFPN)[25] that achieves efficient bidirectional cross-scale connections and weighted feature fusion. Li et al. introduced a lightweight Context and Spatial Feature Calibration Network (CSFCN)[26]. CSFCN consists of two main parts: the Context Feature Calibration (CFC) module and the Spatial Feature Calibration (SFC) module. The CFC module calculates the similarity between pixels and their context, aggregating the context of each pixel's related semantics to achieve context feature calibration, thereby alleviating context mismatch issues. The SFC module groups channels into multiple sub-features along the spatial dimension and calibrates them separately to address feature misalignment problems.

3. Materials and Methods

The combination of object detection algorithms and UAV technology is significant for improving detection efficiency, ensuring regional safety, and reducing resource consumption. The YOLOv8 model comes in five versions, with parameters increasing in size from n, s, m, l, to x; larger models

 doi:10.20944/preprints202407.1302.v1

offer higher detection accuracy. Considering detection accuracy, model size, and practical application needs, this paper selects the YOLOv8s algorithm as the baseline model. Although the original YOLOv8s algorithm outperforms YOLOv8n in detection performance, its parameter count is 3.5 times greater, reaching 11.2M. Therefore, this paper proposes a lightweight improvement to the YOLOv8s algorithm, designing a lightweight object detection algorithm for aerial images in open waters, named GFCA-YOLO. Section 3.1 details the SOM data augmentation algorithm for optimizing and enhancing the dataset. Sections 3.2 and 3.3 focus on lightweight improvements to the backbone network. Section 3.4 discusses the redesign of the neck network. The network structure of the GFLM-YOLO model is shown in **Figure 2**.

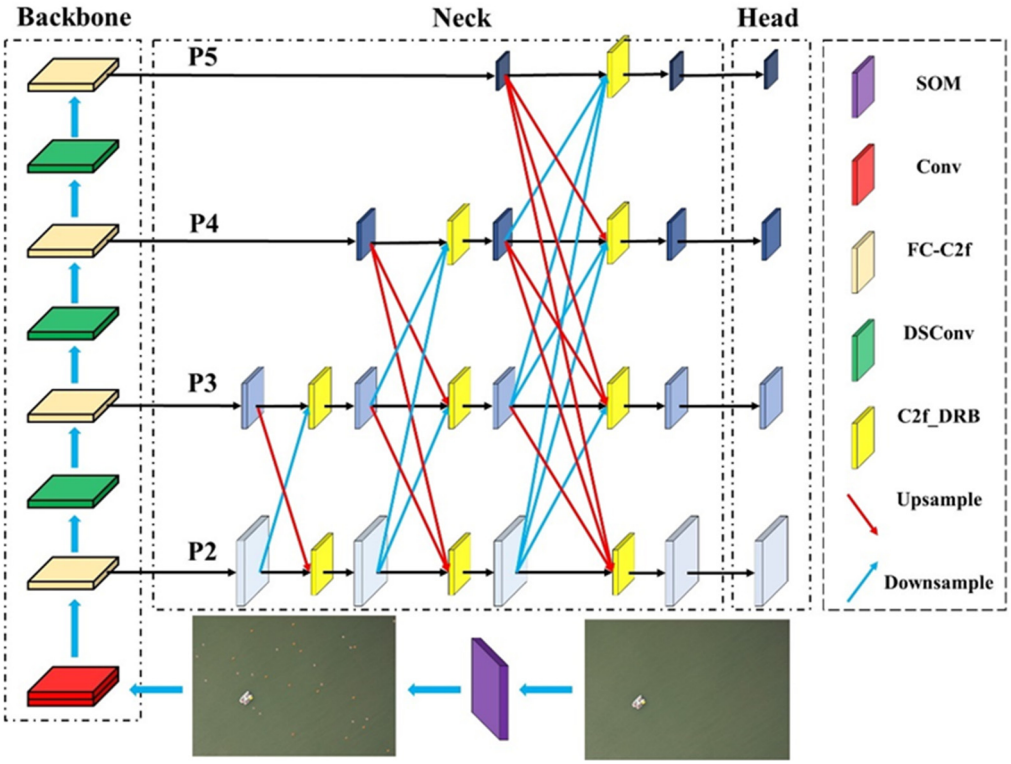
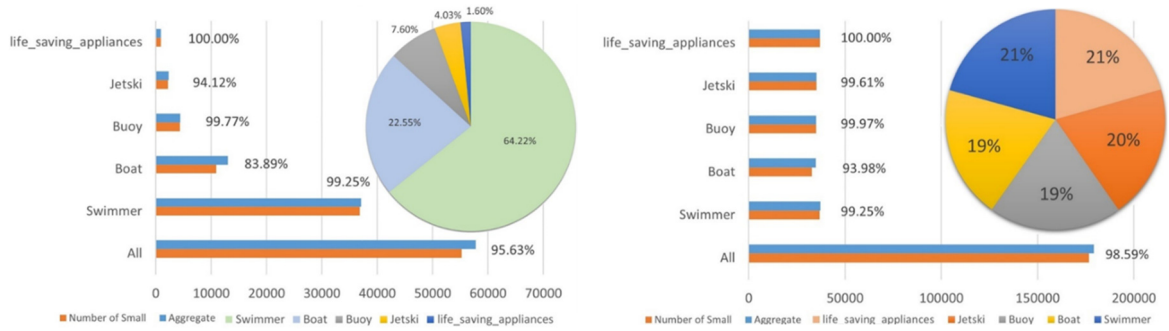


Figure 2. The structure of GFCA-YOLO.

3.1. Data Augmentation

In deep learning, the balance of the dataset is crucial for model training. If the number of samples in one class is significantly lower than in other classes, the model may become biased towards the more numerous classes during training, reducing its ability to classify the less represented classes. When drones perform object detection tasks in open waters, they are often far from the objects, resulting in generally smaller object sizes in the images and fewer features for the model to utilize. We conducted a statistical analysis of the training set of the SeaDroneSea dataset, with the results shown in **Figure 3(a)**. In the clustered bar chart, the blue portion represents the number of objects in each category, while the orange portion indicates the number of small objects within each category. The percentages at the top of the bars show the proportion of small objects within each category. The pie chart on the right in **Figure 3(a)** displays the proportion of objects in each category. From the pie chart, we can see that in the SeaDroneSea dataset, objects with an absolute size smaller than 32×32 pixels account for as much as 95.63% of all objects. Small objects labeled as "life_saving_appliances" make up 100% of that category, which itself represents only 1.6% of the total objects. Due to their small size and low number, the recall rate for "life_saving_appliances" objects in the YOLOv8s algorithm's detection results is only 14.5%, as shown in **Table 1**.

 doi:10.20944/preprints202407.1302.v1



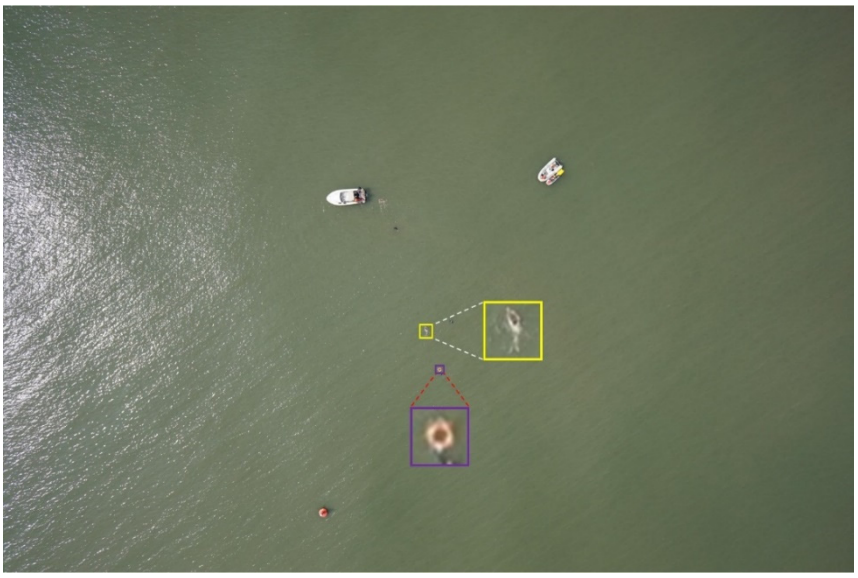
(**a**) (**b**)

Figure 3. The statistical tables of SeaDroneSee dataset. (a) original; (b) after data augmentation.

Inspired by SSMup, Simple Copy-Paste, and Augmentation for small object detection[27], this paper proposes a data augmentation algorithm called Small Object Multiplication (SOM). This algorithm balances the dataset by increasing the number of small objects in specific categories, addressing the low recall and precision rates caused by the lack of small object samples and inconsistent shape features. Unlike the Simple Copy-Paste method, the SOM algorithm uniformly performs Copy-Paste operations on small objects of specified categories with pixel sizes smaller than 32×32 in the original images. The object category index and the number of Copy-Paste operations are set as hyperparameters, allowing users to apply different numbers of Copy-Paste operations for different categories.

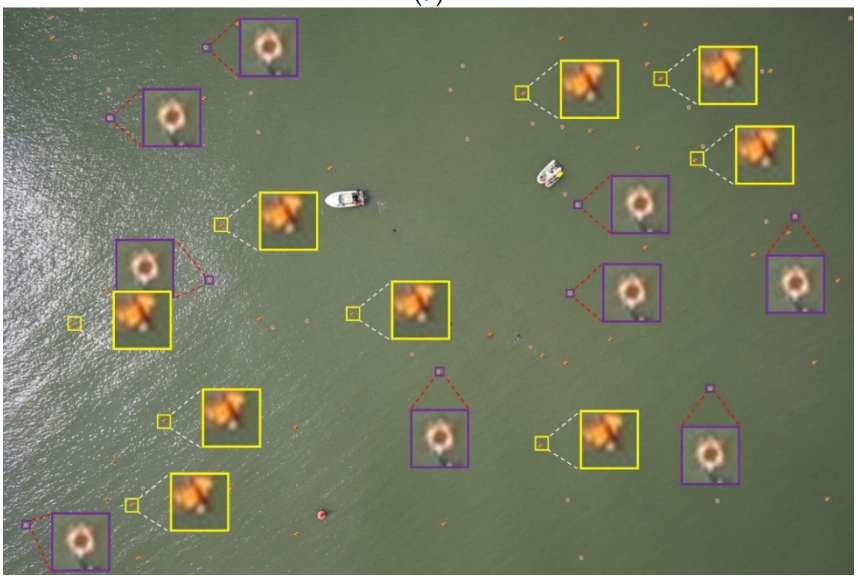
In this paper, the small objects labeled as "boat," "jetski," "life_saving_appliances," and "buoy" were subject to Copy-Paste operations 2, 7, 39, and 15 times, respectively. **Figure 4** displays an example image before and after applying the SOM data augmentation algorithm, with some areas of the figure are enlarged. From the figure, it's evident that two different "life-saving appliances" were duplicated 39 times and evenly pasted onto the original image. Additionally, **Figure 3(b)** presents the statistics of the number of objects in each category of the dataset after data augmentation. It can be seen from the figure that the dataset is nearly balanced across categories after data augmentation.

The experimental results for the SeaDroneSee dataset are shown in **Table 1**. Combining the results from **Figure 4** and **Table 1**, it can be seen that the more Copy-Paste operations performed for a category, the greater the mAP increase. The "life_saving_appliances" category shows the most significant improvement, with the accuracy rate increasing from 78.2% to 81%, the recall rate increasing from 14.5% to 25.5%, and the mAP increasing from 28.2% to 35.4%. Therefore, the SOM data augmentation algorithm can effectively address the low recall and accuracy rates caused by the lack of object samples, resolving the class imbalance problem in the dataset.



Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1302.v1

(**a**)



(**b**)

Figure 4. Sample images before and after data augmentation. (a) original; (b)after data augmentation.

Table 1. The influence of the SOM data augmentation algorithm on SeaDroneSee-val.

Classes	P(%)	R(%)	mAP(%)	P^{SOM}(%)	R^{SOM}(%)	mAP^{SOM}(%)
swimmer	78.7	66.5	69.6	80.1(+1.4)	64.8(-1.7)	70.3(+0.7)
boat	89.8	86	91.6	89.9(+0.1)	87.4(+1.4)	91.2(+0.4)
jetski	76.8	82.2	83.7	86.3(+9.5)	82.5(+0.3)	84.6(+0.9)
life_saving_appliances	78.2	14.5	28.2	81(+2.8)	25.5(+11)	35.4(+6.9)
buoy	77 .7	50.5	57.2	88.5(+10.8)	51.6(+1.1)	61.4(+4.2)
All	80.2	59.9	66.1	85.2(+5)	62.4(+2.5)	68.6(+2.5)

3.2. Depthwise Separable Convolution

The backbone network of the YOLOv8s algorithm accounts for approximately 45% of the overall parameters, with its complex convolutional structures leading to an excessive number of model parameters. This increases the demand for hardware computational power and memory, posing challenges for resource-limited mobile computing terminals to perform object detection tasks. To address this problem, inspired by the classic MobileNet lightweight network family, this paper uses DSConv[28] to make lightweight improvements to the YOLOv8s backbone network. The detailed structure of DSConv is shown in **Figure 5**.

We will now analyze and compare the computational requirements for performing a convolution operation using standard convolution and DSConv. Let's assume that the input feature map size is $h \times w \times c$, the size of the convolution kernel is $k \times k \times c$, and the total number of kernels is N. The spatial dimension of the feature map contains a total of $h \times w$ points. The computation required to perform the convolution operation for each point is equivalent to the size of the convolution kernel. If we perform one convolution operation for each point at every spatial location in the feature map, then a single convolution would require a total $h \times w \times c \times k^2$ of computations. Analogous analysis, the number of calculations required in the DWConv stage is also $h \times w \times c \times k^2$. The number of calculations required in the PWConv process is $h \times w \times c \times N$ of DSConv is: $h \times w \times c \times k^2 + h \times w \times c \times N$. From this, the total number of calculations for the N standard convolutions is $h \times w \times c \times k^2 \times N$. We can derive the ratio of the total number of calculations for the DSConv and the standard convolutions as follows: $\frac{1}{N} + \frac{1}{k^2}$. In summary, the computational efficiency of DSConv is far superior to that of standard convolutions, which can reduce the model

 doi:10.20944/preprints202407.1302.v1

parameters and reduce the network inference delay, which is conducive to the lightweight design of the model. computational efficiency is much better than standard convolution, which can reduce the model parameters, reduce the network inference delay, and is conducive to the lightweight design of the model.

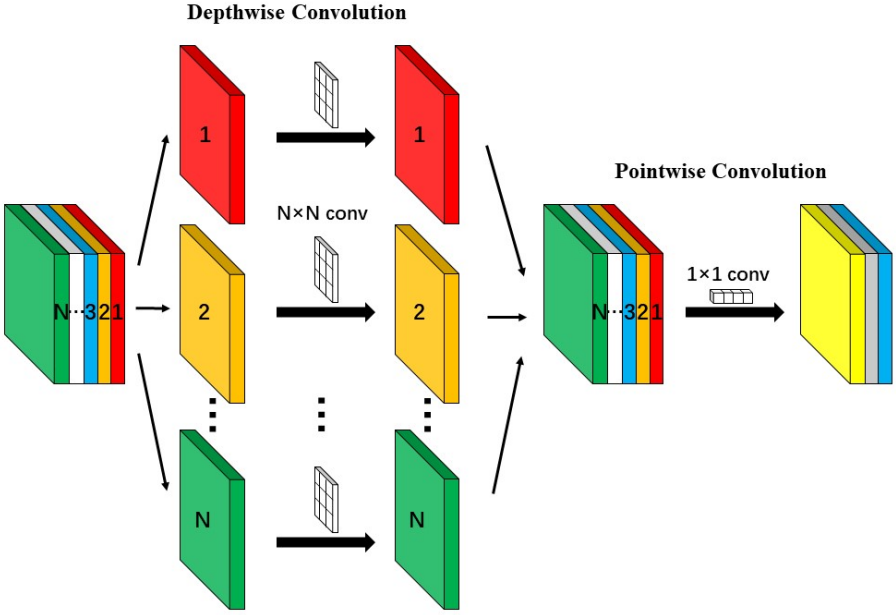


Figure 5. The structure of DSConv.

The experimental results for the SeaDroneSee dataset before and after lightening the backbone network using DSConv are presented in **Table 2**. The improved backbone network with the DSConv module reduces the model's parameters by 13.9%, FLOPs by 10.5%, and decreases the inference time from 2.6ms to 1.2ms, resulting in a 53.85% improvement. Additionally, there is a slight improvement in detection accuracy. These results indicate that DSConv can enhance the computational efficiency of the model, decrease the model's inference time, and improve detection accuracy.

Table 2. The influence of DSConv on SeaDroneSee-val.

Algorithms	**P** (%)	**R** (%)	**mAP50**^{val} (%)	**Params** (M)	**FLOPs** (G)	**Speed** RTX4090 b16 (ms)
YOLOv8s	80.2	59.9	66.1	11.14	28.7	2.6
YOLOv8s+DSConv	79.8(-0.4)	60.5(+0.6)	66.6(+0.5)	9.59(-1.55)	25.7(-3)	1.2(-1.4)

3.3. Improved C2f Module Based on Convolutional Gated Linear Unit and Faster Block

In the original YOLOv8s' backbone network, the convolution module only accounts for 30% of the total parameters, while the C2f module accounts for 56%. Therefore, this paper draws inspiration from the FasterBlock module in FasterNet[29] to make lightweight improvements to the C2f module in the backbone network. FasterNet is a fast neural network proposed by Jierun Chen et al. in 2023. The model delay is calculated as $Latency = \frac{FLOPs}{FLOPS}$. Chen et al. discovered that many researchers focus on accelerating models by reducing FLOPs, such as ShuffleNets[30] and GhostNet[31]. However, this reduction in FLOPs can lead to increased memory access, resulting in higher network latency and affecting computational speed. This explains why some models have low FLOPs but still exhibit slow inference and run times. We encountered a similar situation in our experiments when replacing the original YOLOv8s backbone with HGNetV2[7]. Although FLOPs decreased by 19%, there was almost no change in inference speed, as shown in **Table 3**.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1302.v1

Table 3. The influence of replacing YOLOv8s backbone with HGNetV2 on SeaDroneSee-val.

Algorithms	FLOPs (G)	Pre-Process (ms)	Inference (ms)	NMS (ms)
YOLOv8s	28.8	0.2	1.6	0.8
YOLOv8s +HGNetV2	23.3	0.2	1.6	0.7

To reduce memory access and computational redundancy, Chen et al. proposed a new convolutional structure called Partial Convolution (PConv) and designed FasterNet block based on PConv. The working principle of PConv and the model structure of FasterNet block are illustrated in **Figure 6**. As depicted in **Figure 6**, PConv utilizes standard convolution to extract spatial features from a portion of the input channel while leaving the rest of the channel unchanged. This process significantly reduces computational redundancy and enhances the inference speed. We use the same method as in the previous subsection to calculate the computation of PConv, assuming that the size of the input feature map is $h\times w\times c$, c_p is the number of feature map channels participating in the convolution operation, and the size of the convolution kernel is $k\times k\times c$, and it is the number of c_p. We can derive the total computation amount of PConv as $h\times w\times c_p^2\times k^2$. In summary, the ratio of the total computational load of PConv to that of standard convolution is $\frac{c_p^2}{c\times N}$

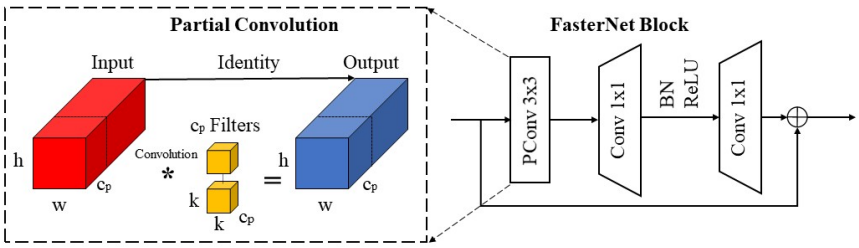


Figure 6. The structure of FasterNet Block.

Although the improved C2f module reduces the model parameters by 13%, inference is still slow because Conv extracts features through serial operations. The Gated Linear Unit (GLU)[32] was proposed in 2016 by Yann N. Dauphin et al. The GLU consists of two linear projections, one of which is controlled by a gating function, and the two projections are multiplied elementwise. The GLU can accelerate computation speed and reduce model's complexity through parallel processing structure. Subsequently, Dai Shi proposed an improved GLU call Convolutional GLU (CGLU)[33]. This new model innovatively combines depthwise convolution using parallel processing structures to enhance computation speed and reduce model complexity. Additionally, the depthwise convolution structure provides positional information and efficient feature extraction capabilities. Inspired by FasterBlock and CGLU, this paper improves the C2f structure in the YOLOv8s backbone and proposes the FasterBlock-CGLU-C2f (FC-C2f) module. The structure of the FC-C2f module is shown in **Figure 7**.

The experimental results on the SeaDroneSee dataset are shown in **Table 4**, where FB-C2f is the improved C2f module using only FasterBlock. From **Table 4**, we can conclude that the FC-C2f module improved with CGLU can further optimize the network structure compared to the FB-C2f module. It not only enhances detection accuracy but also makes the network lighter, increasing the model's detection speed and proving the effectiveness of the CGLU improvement. The improved FC-C2f module reduces the model's parameters by 17.5%, FLOPs by 17.1%, and inference time from 2.6ms to 1.4ms, a reduction of 46.2%, with a slight improvement in detection accuracy. The proposed FC-C2f module can increase detection accuracy while reducing model parameters, lowering model complexity, and significantly improving the model's running speed.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1302.v1

Table 4. The influence of FasterBlock and FasterBlock-CGLU-C2f on SeaDroneSee-val.

Algorithms	P (%)	R (%)	mAP50^{val} (%)	Params (M)	FLOPs (G)	Speed RTX4090 b16 (ms)
YOLOv8s	80.2	59.9	66.1	11.14	28.7	2.6
YOLOv8s+FB-C2f	80.4(+0.2)	61.2(+1.3)	66.1(+0)	9.69(-1.45)	24.4(-4.3)	2.2(-0.4)
YOLOv8s +FC-C2f	82.8(+2.6)	59.5(-0.4)	66.5(+0.4)	9.48(-1.66)	23.8(-4.9)	1.4(-1.2)

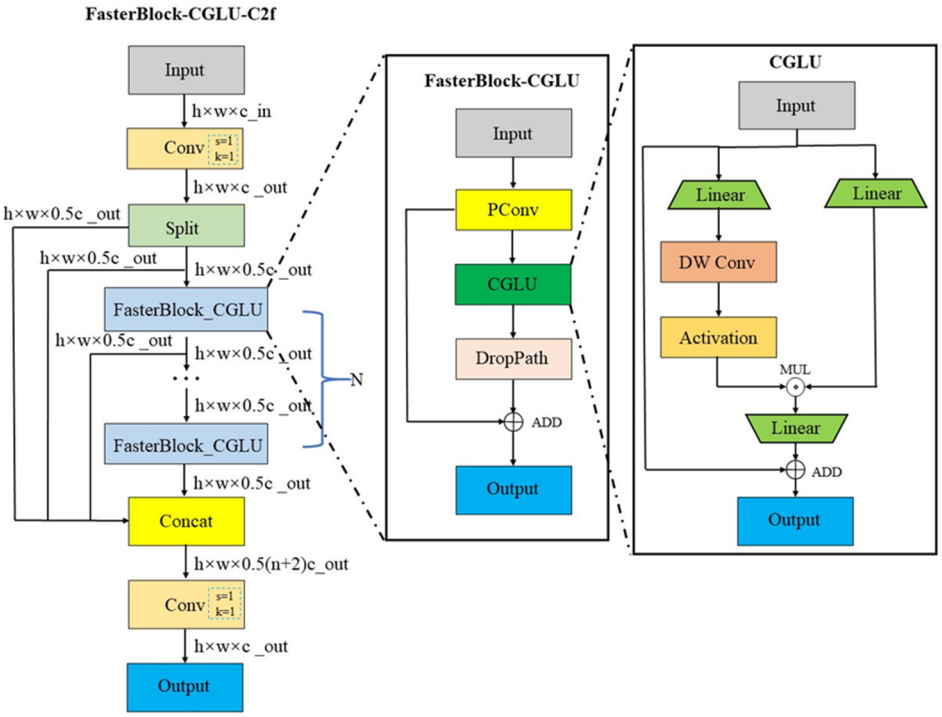


Figure 7. The structure of FasterBlock-CGLU-C2f.

3.4. Lightweight Multiscale Feature Fusion Network

The uncertainty of the object size in aerial images can lead to the loss of information during feature extraction, which affects the model's detection performance. The YOLOv8 algorithm utilizes FPN and PAN structures for downsampling and upsampling, respectively. These structures have large parameter counts, high computational redundancy, significant conflicts between different feature levels, and relatively limited effectiveness in detecting small objects. The neck network of the YOLOv8s algorithm has approximately 6.06M parameters, making it challenging to deploy on airborne computation device. Therefore, this paper designs a lightweight multiscale fusion network (LMFN). The architecture of LMFN is shown in **Figure 8**.

LMFN adopts the progressive fusion strategy of the Asymptotic Feature Pyramid Network (AFPN)[34], which integrates shallow and deep features extracted by the backbone network in three stages. This fusion method weights the semantic and positional information of deep and shallow features according to different feature levels, avoiding significant semantic gaps between different feature layers and preventing information loss and degradation caused by multi-level transmission.

The receptive field can be expanded by connecting multiple small-kernel convolutions or by using a single large-kernel convolution. Researchers typically prefer the first method because using multiple small-kernel convolutions has three advantages over a single large-kernel convolution. Firstly, multiple small-kernel convolution structures include more non-linear activation layers, thereby enhancing the discriminative ability of the model. Secondly, it reduces network parameters. For example, using three 3×3 cascade convolutions instead of a single 7×7 large kernel convolution

 doi:10.20944/preprints202407.1302.v1

can reduce $7\times7-3\times3\times3=22$ parameters, cutting the network parameters by 45%. Lastly, it also decreases the computational load. Similarly, using three 3×3 cascade convolutions instead of a single 7×7 large kernel convolution can reduce the number of computations by $7\times7\times C-3\times3\times3\times C=22\times C$, reducing the computational load by 45%.

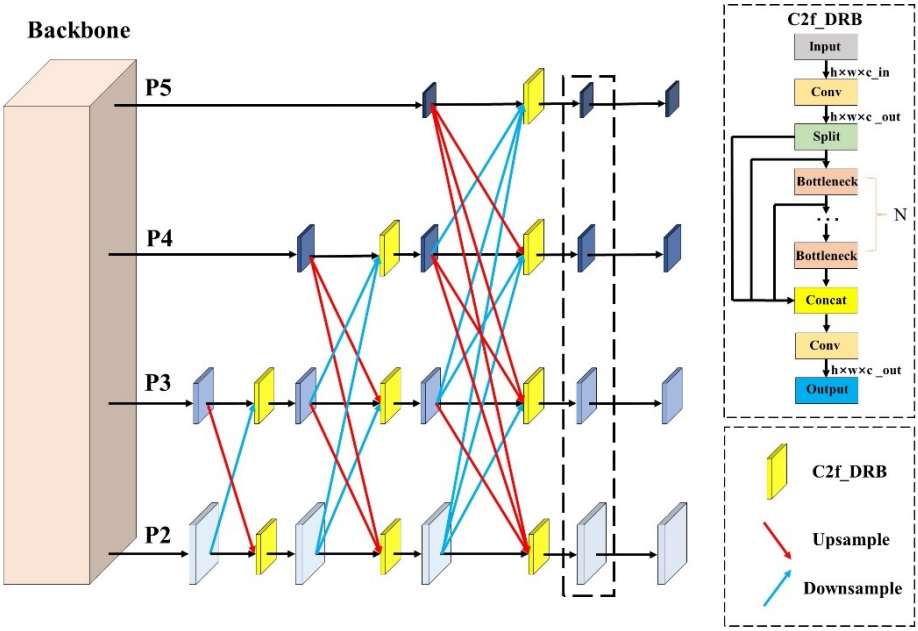


Figure 8. The structure of LMFN.

Ding et al. found that although connecting multiple small-kernel convolutions can theoretically expand the model's maximum receptive field, the actual effective depth is not significant, resulting in a small Effective Receptive Field (ERF)[35]. The ERF is proportional to $O(K\sqrt{L})$, where K represents the kernel size and L is the model depth[36]. This indicates that the ERF is more sensitive to changes in K. Although the optimization problems brought by increasing model depth have been addressed by ResNet, increasing the depth of the model is not as effective as increasing the kernel size.

To achieve a large ERF and fully utilize deep features while avoiding the increase in network parameters and computational load caused by large-kernel convolutions, this paper designs a new C2f module named DRB-C2f. The Dilated Re-param Block (DRB)[37] module utilizes parallel dilated convolution in addition to the large kernel convolution. Using the concept of reparameterization, the entire module can be considered a non-dilated large kernel convolution. The operational diagram is depicted in **Figure 9**. The ignored pixels in dilated convolution can be considered as adding extra zero terms to the convolution kernel. Therefore, a dilated convolution layer with a small convolution kernel can be seen as a non-dilated convolution layer with a larger but sparser kernel. The process of adding zeros can be carried out using transposed convolution with a step size r and a unit kernel $I\in ¡^{1\times1}$. If the original convolution kernel is $I\in ¡^{1\times1}$, the convolution kernel after the interpolation operation is $W'\in ¡^{((k-1)r+1)\times((k-1)r+1)}$. The Re-parameterize module consists of a non-diluted small convolution kernel and multiple diluted small convolution kernels to complement the non-diluted large convolution kernel.

 doi:10.20944/preprints202407.1302.v1

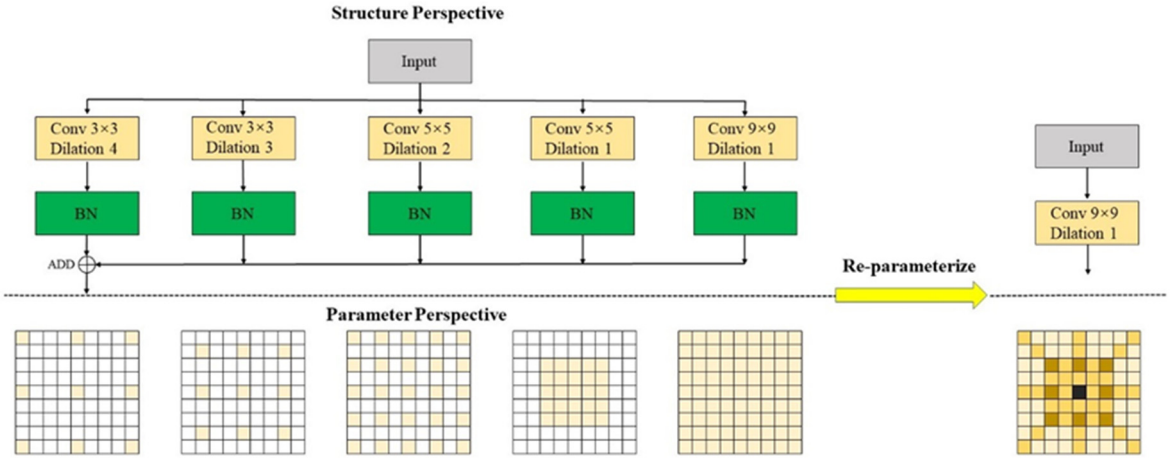


Figure 9. The computational process of the DRB module.

We utilized the visualization tool Zetane to display the feature maps in the dashed section of **Figure 8**. **Figure 10** illustrates the changes in the four feature layers before and after implementing DRB. The left side displays the size of the feature maps without DRB enhancement, while the right side shows the size of the feature maps after incorporating DRB enhancement. The figure indicates that using DRB significantly increases the size of the model's receptive field. The mean receptive field size increased from 0.305, 1.63, 1.77, and 6.28 to 0.879, 3.15, 2.7, and 12.7 for P2, P3, P4, and P5, respectively.

The experimental results on the SeaDroneSee dataset are shown in **Table 5**. AFPN_C2f is the neck network that is not improved with DRB. From the table, we can see that AFPN_C2f achieves lightness at the expense of accuracy. The DRB module introduction improves the multiscale feature fusion network, increasing the model's effective receptive field and detection accuracy, while reducing the model's parameters and FLOPs. Compared to YOLOv8s, mAP50 is improved by 10.6%, and the number of parameters is reduced by 38.4%. However, due to the extensive multi-scale fusion operations, FLOPs increased by 7.7%. Compared to AFPN, the detection accuracy of LMFN did not decrease but improved by 0.4%, and the number of parameters and FLOPs decreased by 21.7% and 20%, respectively. Overall, the enhanced LMFN network exhibits significant superiority.

Table 5. The influence of AFPN, AFPN_C2f and LMFN on SeaDroneSee-val.

Algorithms	**P** (%)	**R** (%)	**mAP50**^{val} (%)	**Params** (M)	**FLOPs** (G)	**Speed** RTX4090 b16 (ms)
YOLOv8s	80.2	59.9	66.1	11.14	28.7	2.6
AFPN	83.2(+3.0)	71.4(+11.5)	76.3(+10.2)	8.76(-2.38)	38.9(+10.2)	3.1(+0.5)
AFPN_C2f	84.5(+4.3)	69.6(+9.7)	76.0(+9.9)	7.09(-4.05)	34.2(+5.5)	2.8(+0.2)
LMFN	86.8(+6.6)	70.5(+10.6)	76.7(+10.6)	6.86(-4.28)	31.1(+2.4)	2.3(-0.3)

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1302.v1

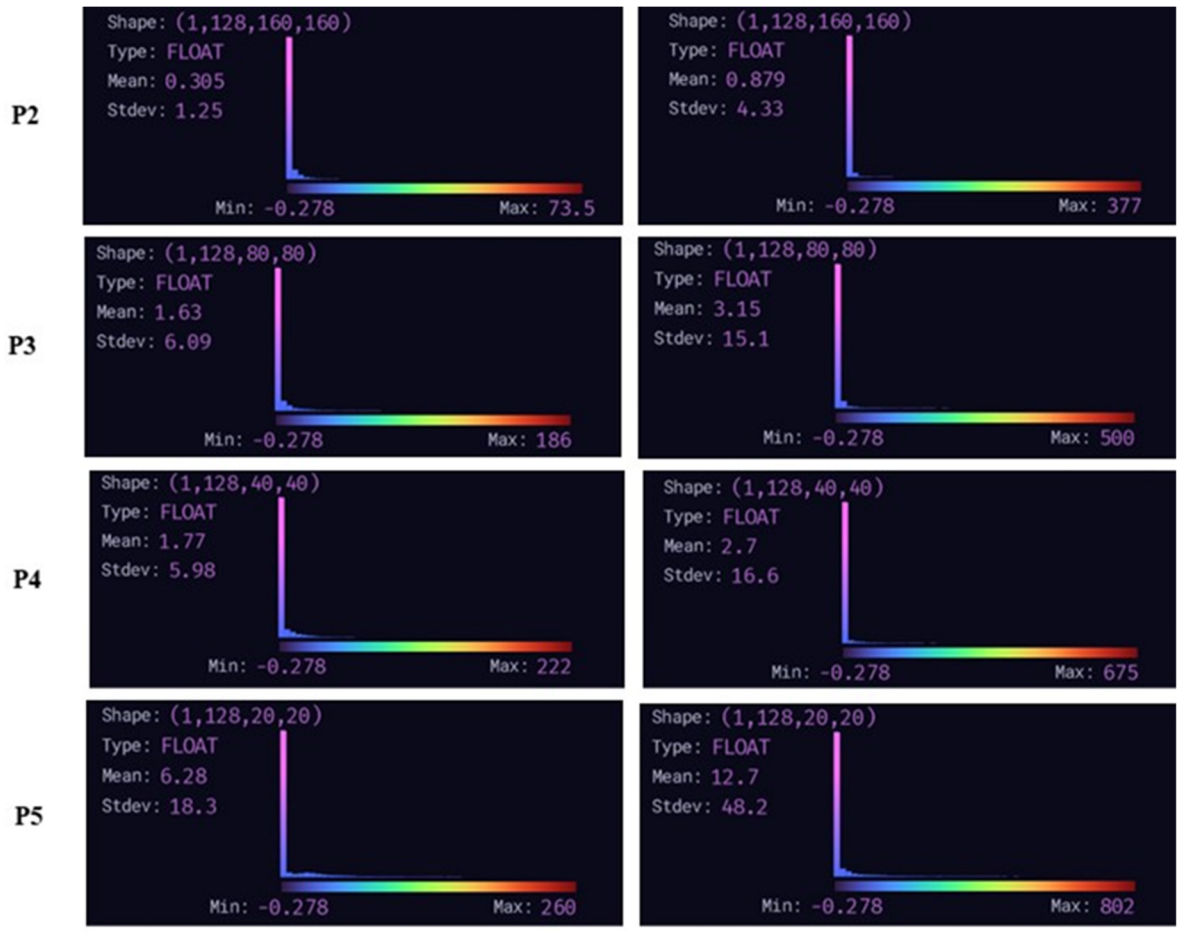


Figure 10. The feature map dimensions for P2-P5.

4. Experimental and Analysis

4.1. Dataset and Experimental Environment and Parameter Settings

The experimental data for this paper comes from the SeaDroneSee public dataset. SeaDroneSee was collected by the University of Tuebingen team to assist in the development of search and rescue systems using UAVs in maritime scenarios. The dataset comprises 8,930 training images, 1,547 validation images, and 3,750 test images. These images include various lighting conditions, different shooting distances, and angles, meeting factors such as small object size and complex environments. There are five categories in the dataset: Swimmer, Boat, Buoy, Jetski, and life_saving_appliances. This dataset provides a valuable data resource for maritime search and rescue in open waters.

The hardware configurations and software versions involved in this experiment are shown in **Table 6**. The hyperparameter settings of the model are shown in **Table 7**.

Table 6. Hardware model and software version used for the experiment.

Experimental Environment	Parameter/Version
Operating System	Ubuntu20.04
GPU	NVIDIA Geforce RTX 4090
CPU	Intel(R) Xeon(R) Gold 6430
Cudn	11.3
Pytorch	1.10.0
Python	3.8

 doi:10.20944/preprints202407.1302.v1

Table 7. Network model hyperparameter settings.

Parameter	Setup
Image size	640×640
Momentum	0.937
BatchSize	16
Epoch	200
initial learning rate	0.01
final learning rate	0.0001
Weight decay	0.0005
Warmup epochs	3
IoU	0.7
Close Mosaic	10
Optimizer	SGD

4.2. Experimental Metrics

To objectively evaluate the model's performance, this paper measures and analyzes it from two aspects: detection performance and detection speed. The quantitative metrics for detection performance include Precision, Recall, mAP50, Parameters, and FLOPs. The quantitative metric for detection speed is the total inference time for a single image with a batch size set to 16.

The Precision measures the ratio of correctly predicted objects to the total number of predicted objects using the following formula (1):

$$P = \frac{TP}{TP + FP}, \tag{1}$$

Recall measures the ratio of correctly predicted objects to the number of actual objects. The calculation formula is as follows (2):

$$R = \frac{TP}{TP + FN}, \tag{2}$$

In the formulas, TP (True Positives) is the number of objects correctly identified by the model, FP (False Positives) is the number of other class objects incorrectly identified as this class, and FN (False Negatives) is the number of this class objects incorrectly identified as other classes. Thus, $TP + FP$ represents the number of objects predicted by the model, while $TP + FN$ represents the actual number of such objects.

The mean average precision (mAP) is the average of the average precision (AP) values for multiple categories and is used to measure the overall performance of the model. mAP50 is the mAP at an IoU threshold of 50%. The calculation formula (3) for mAP is as follows:

$$mAP = \frac{\sum_{n=1}^{N} AP_n}{N}, \tag{3}$$

Where AP_n refers to the average precision of the nth class of objects, used to evaluate the model's detection performance for that specific class. The calculation formula (4) is as follows:

$$AP = \int_0^1 PRdR, \tag{4}$$

The parameter count represents the total number of parameters in the model, which is a key indicator of the model's size and spatial complexity. It directly impacts the size of the weight file and indirectly affects the storage space required for model deployment. FLOPs measure the computational load required during model training and inference, reflecting the model's computational efficiency.

 doi:10.20944/preprints202407.1302.v1

4.3. Ablation Experiments

In this study, we propose four improvement methods for YOLOv8s aimed at reducing model parameters, increasing detection speed, and enhancing the detection accuracy for small object. The four improvement methods include: (a) utilizing the SOM data enhancement algorithm to improve dataset balance, (b) using DSConv instead of standard convolutions in the backbone network for lightweight design, (c) improving the C2f module in the backbone network using FasterBlock and GSLU to reduce parameters and latency, and (d) redesigning the neck network with the LMFN network to increase the effective receptive field and multiscale feature fusion capability. To study and analyze the effectiveness of each improvement method in depth, we conducted ablation experiments on the SeaDroneSee dataset. The results are shown in **Table 8**. **Figure 11** compares the mAP curves of eight sets of experimental results, and **Figure 12** shows the scatter plot of model parameters and inference time for the eight sets of experiments. Since Chapter 3 already provides detailed analyses of the experimental results for each individual improvement, this chapter focuses on discussing and analyzing the combined effects of multiple improvements on the baseline model.

Table 8. The influence of different enhancements evaluated on SeaDroneSee-val.

Class	Algorithms	P (%)	R (%)	mAP50^{val} (%)	Params (M)	FLOPs (G)	Speed RTX4090 b16 (ms)
1	YOLOv8s	80.2	59.9	66.1	11.14	28.7	2.6
2	a	84.1	62.2	67.7	11.14	28.7	2.6
3	b	79.8	60.5	66.6	9.59	25.7	1.2
4	c	82.8	59.5	66.5	9.48	23.8	1.4
5	d	86.8	70.5	76.7	6.86	31.1	2.4
6	b+c	81	59.4	66.1	7.93	20.9	1.1
7	b+c+d	82.6	71.3	76.6	3.65	23.3	2.1
8	a+b+c+d(our)	85.5	71.6	78.3	3.64	22.9	2.1

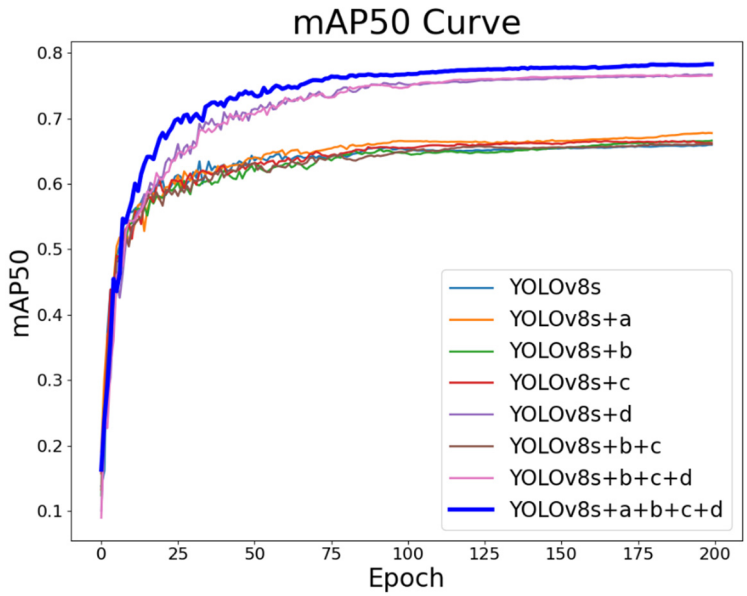


Figure 11. The mAP curves of different enhancements and GFLM-YOLO (blue).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1302.v1

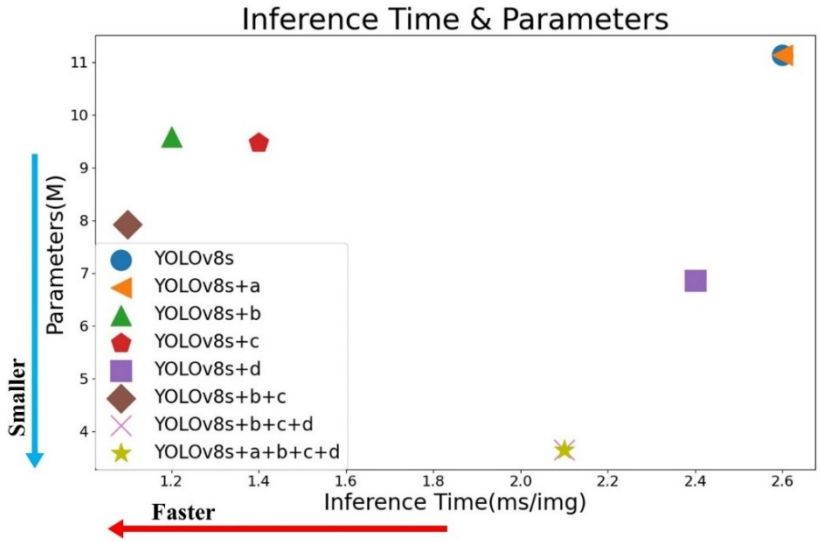


Figure 12. The scatter plot of different enhancements and GFLM-YOLO (yellow star).

1. Lightweight improvements to the backbone network: From the results of Group 6 experiments in **Table 8**, we can see that compared with the baseline model, using DSConv and FC-C2f for lightweight improvements to the baseline model's backbone network significantly reduces model parameters and FLOPs while substantially increasing detection speed. Specifically, the model parameters decreased from 11.14 to 7.93, FLOPs reduced from 28.7 to 20.9, and detection time per image dropped from 2.6ms to 1.1ms. In addition, the change in detection accuracy before and after the lightweight improvements was minimal. **Figure 11** and **Figure 12** visually confirm these findings, showing that the lightweight improved model has similar mAP curves to the YOLOv8s model but with fewer parameters and shorter inference time. The experiments demonstrate that DSConv and FC-C2f can improve the computational efficiency and reduce the complexity of the model without affecting detection accuracy.

2. LMFN: Comparing the experiments from Group 6 and Group 7, we can see that the proposed LMFN network can enhance the model's Precision, Recall, and mAP, while reducing the model's parameters. However, LMFN increases the model's FLOPs due to its gradually fusion of multiple feature maps from different layers. This extensive cross-layer fusion mitigates information discrepancy between different feature levels, enhancing the model's ability to recognize objects at various scales. However, this complex fusion process also leads to an increase in the model's FLOPs and network delay. **Figure 11** and **Figure 12** show that the mAP curve of the LMFN improved model is higher than that of the baseline model, although the inference time is increased. The experiments demonstrate that the LMFN proposed in this paper can effectively improve the detection accuracy of the model and reduce model parameters. Although the FLOPs increased from 20.9 to 23.3, this increase is negligible compared to the significant improvement in detection accuracy.

3. GFLM-YOLO: Comparing the group 8 of experiments with the baseline model, we find that the proposed model improves precision by 6.4%, recall by 11.1%, and mAP50 by 12.4%. The model parameters are reduced from 11.17 to 3.65, a reduction of 67.2%, and FLOPs decrease from 28.7 to 23.3, a reduction of 18.8%. Inference time is reduced from 2.6ms to 2.3ms, a reduction of 11.5%. **Figure 11** and **Figure 12** demonstrate that GFLM-YOLO has the highest mAP curve compared to other models, showing excellent detection accuracy while meeting real-time requirements. This model achieves a balance between detection accuracy and speed.

The experimental results show that the four proposed improvements to the baseline model not only improve detection accuracy but also achieve a lightweight design, simplifying the network structure, enhance computational efficiency, and reducing network latency. These improvements ensure the model meets the real-time and accuracy requirements for drone-based maritime object detection tasks.

 doi:10.20944/preprints202407.1302.v1

Figure 13 shows the detection results of GFLM-YOLO and YOLOv8s models in various sea maritime environments (some smaller areas are enlarged). Group 1 shows detection in dense small object scenarios. YOLOv8s has instances of missed small objects, while GFLM-YOLO correctly identifies all small objects. From the Group 2's detection results, it is evident that GFLM-YOLO not only has higher accuracy than YOLOv8s but also detects more small objects. The third group of results shows that due to the similar pixel features of "swimmer" and "life_saving_appliances", YOLOv8s incorrectly identifies "life_saving_appliances" as "swimmer," whereas the improved model correctly identifies "life_saving_appliances". The detection results of the fourth group indicate that GFLM-YOLO has significantly higher detection accuracy for small objects than YOLOv8s, and also exhibits a higher recall rate. Therefore, the GFLM-YOLO algorithm proposed in this paper offers superior small object detection performance compared to the baseline model and provides a new solution for maritime object detection.

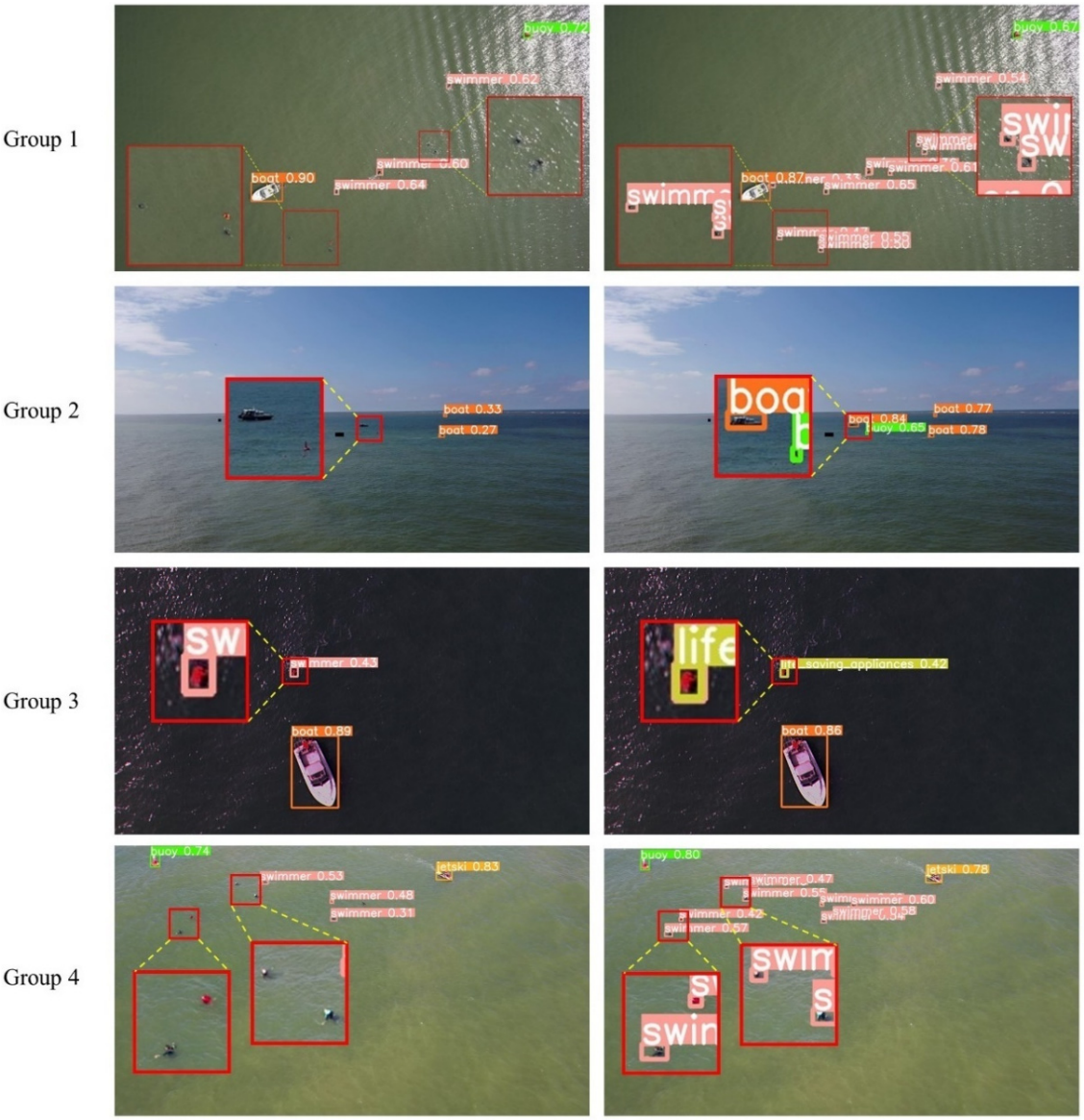


Figure 13. Partial comparison object detection results of the SeaDroneSee dataset,YOLOv8(left) and GFLM-YOLO(right).

4.4. Comparative Experiment

To further validate the performance advantages of the GFLM-YOLO algorithm for small object detection with UAVs, comparative experiments are conducted against seven popular detection

 doi:10.20944/preprints202407.1302.v1

algorithms: YOLOv5s, YOLO-OW, YOLOv8n, YOLOv9t, YOLOv10s, YOLO-OW, and DETR. Among these, YOLO-OW is the top-ranking detection model on the official leaderboard[38]. The experimental results of each model are shown in **Table 9**.

Table 9. The influence of different models evaluated on SeaDroneSee-val.

Class	Algorithms	P (%)	R (%)	mAP50^{val} (%)	Params (M)	FLOPs (G)	Speed RTX4090 b16 (ms)
1	YOLOv5s	82.7	57.9	65.4	9.11	23.8	2.1
2	YOLOv6n	79.5	57.7	60.6	4.23	11.8	1.7
3	YOLOv8n	79.0	58.8	63.6	3.0	8.1	1.6
4	YOLOv9t	74.1	58.5	62.3	2.62	10.7	4.5
5	YOLOv10s	82.3	59.3	63.8	8.04	24.5	1.0
6	YOLO-OW	82.4	76.2	73.1	42.1	94.8	4.6
7	RT-DETR-R18	88.4	82.6	83.6	20.0	57.0	3.8
8	GFLM-YOLO (our)	85.5	71.6	78.3	3.64	22.9	2.1

According to **Table 9**, the proposed GFLM-YOLO algorithm significantly outperforms other algorithms with similar parameter levels in terms of detection accuracy. The parameter counts of YOLOv6n, YOLOv8n, and YOLOv9t are similar to that of GFLM-YOLO algorithm, but the mAP scores of the GFLM-YOLO algorithm are higher by 17.7%, 14.7%, and 16%, respectively. The mAP curve in **Figure 14** is indicates that the GFLM-YOLO algorithm outperforms the other three algorithms. **Figure 15** illustrates that YOLOv6n and YOLOv8n have inference times 0.4ms and 0.5ms faster than GFLM-YOLO, respectively, while YOLOv9t has an inference time of 4.5ms, due to the auxiliary reversible branches used.

YOLOv5s and YOLOv10s have FLOPs similar to GFLM-YOLO, but their parameter counts are 2.5 times and 2.2 times higher, respectively, and their mAP scores are 12.9% and 14.5% lower than GFLM-YOLO. As shown in **Figure 15**, YOLOv5s has a similar inference time to GFLM-YOLO, while YOLOv10s has an inference time of only 1ms. This is because YOLOv10s uses a non-NMS training continuous dual assignment strategy, which significantly reduces inference time.

The mAP of the YOLO-OW algorithm is 73.1%, which is 5.2% lower than GFLM-YOLO. Its parameter count is 11.5 times that of GFLM-YOLO, its FLOPs are 4.1 times higher, and its inference time is as high as 4.5ms. In repeated experiments under the same conditions, we observed that the mAP metric for YOLO-OW did not improve during the early stages of training. In repeated experiments under the same conditions, we observed that the mAP metric for YOLO-OW did not improve during the early stages of training. The mAP of RT-DETR-R18 is 5.3% higher than GFLM-YOLO, but its parameter count is 5.5 times higher, its FLOPs are 2.5 times greater, and its inference time is approximately twice that of GFCL-YOLO.

In summary, the GFLM-YOLO algorithm proposed in this paper demonstrates significant advantages in detection accuracy compared to current mainstream object detection algorithms with similar parameter levels and FLOPs. The GFLM-YOLO maintains low parameter count and FLOPs while meeting the requirements for high detection accuracy and real-time performance, providing a new solution for object detection in open water.

 doi:10.20944/preprints202407.1302.v1

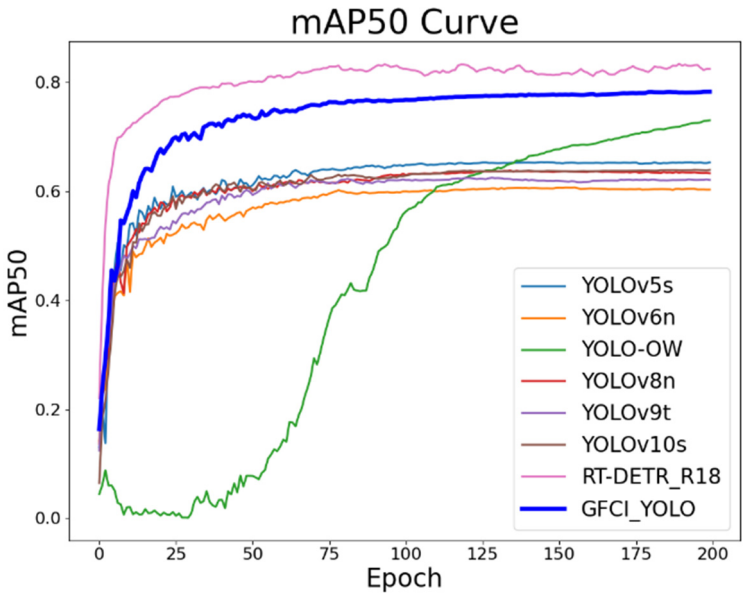


Figure 14. The mAP curves of other models and GFLM-YOLO (blue).

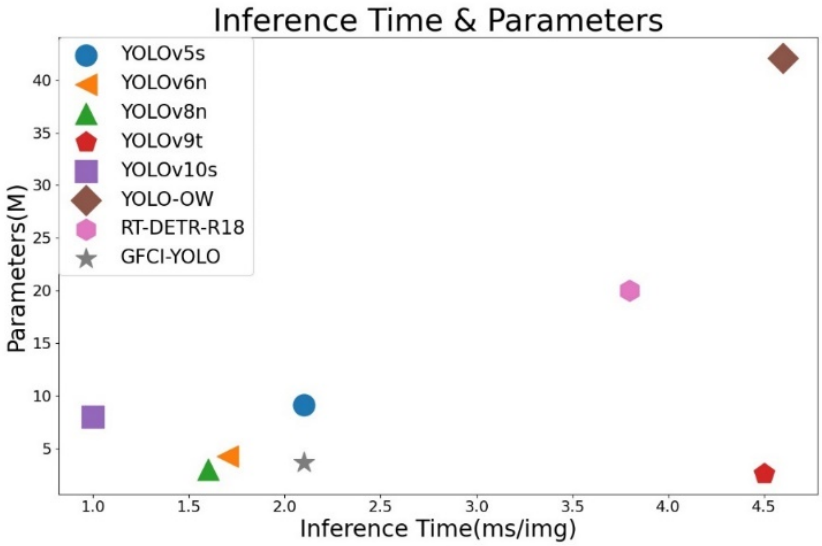


Figure 15. The scatter plot of other models and GFLM-YOLO (grey star).

5. Conclusions

Object detection in open water is crucial for maritime rescue, resource management, and navigation. In this paper, we propose a lightweight object detection algorithm based on multiscale fusion, which improves the detection accuracy while satisfying the deployment conditions and real-time requirements of airborne computation device.

This paper proposes a new data augmentation algorithm called SOM to address the class imbalance problem in the SeaDroneSee dataset. The algorithm increases the number of specified class objects through Copy-Paste without adding actual objects, thereby resolving the class imbalance problem. To meet the requirements of deploying airborne computation device, while maintaining the detection accuracy. Firstly, we lightweight the backbone of the original YOLOv8s network, introducing the lightweight feature extraction module FC-C2f, which significantly reduces model parameters and FLOPs, thereby shortening inference time. Additionally, DSConv was used to replace standard convolutions in the original network, further reducing the parameters and FLOPs of the backbone network. Finally, a lightweight multi-scale feature fusion network, LMFN, was proposed as the neck network. The LMFN effectively reduces the contradiction between different feature layers

 doi:10.20944/preprints202407.1302.v1

by gradually fusing multiple feature layers extracted from the backbone network and improves the model's ability to recognize multiscale objects. This paper introduces the DRB module into LMFN, which uses a Re-parameterize module to equate multiple small kernel dilated convolutions to a single large kernel non-dilated convolution. The Re-parameterize module consists of one non-dilated small convolution kernel and several dilated small convolution kernels. Research shows that the DRB module can significantly increase the effective receptive field of LMFN, enhancing the model's detection performance and improving the recall rate for small objects. However, while LMFN improves detection accuracy and reduces parameter count, the extensive fusion between different feature layers increases the model's complexity.

In future research, we will collect and analyze image data under extreme weather conditions to enhance the model's adaptability and generalization in practical applications. Additionally, we will address the adverse effects of sea surface glare and waves on small object detection, and improve network structure and feature fusion strategies to further reduce the model's FLOPs and increase detection speed.

Author Contributions: Conceptualization, C.S.; methodology, C.S.; software, C.S.; validation, C.S. and S.M.; formal analysis, C.S.; investigation, C.S.; resources, C.S.; data curation, C.S.; writing—original draft preparation, C.S.; writing—review and editing, C.S. and S.M.; visualization, C.S.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Shanghai Industrial Collaborative Innovation Project, grant number XTCX-KJ-2023-2-18.

Data Availability Statement: The SeaDroneSee dataset used in this study is publicly accessible and can be obtained from the official repository: https://seadronessee.cs.uni-tuebingen.de/dataset

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

References

1. Yang, T.; Jiang, Z.; Sun, R.; Cheng, N.; Feng, H. Maritime Search and Rescue Based on Group Mobile Computing for Unmanned Aerial Vehicles and Unmanned Surface Vehicles. *IEEE Trans. Ind. Inform.* **2020**, *16*, 7700–7708, doi:10.1109/TII.2020.2974047.
2. Tang, G.; Ni, J.; Zhao, Y.; Gu, Y.; Cao, W. A Survey of Object Detection for UAVs Based on Deep Learning. *REMOTE Sens.* **2024**, *16*, 149, doi:10.3390/rs16010149.
3. Bouguettaya, A.; Zarzour, H.; Kechida, A.; Taberkit, A.M. Deep Learning Techniques to Classify Agricultural Crops through UAV Imagery: A Review. *NEURAL Comput. Appl.* **2022**, *34*, 9511–9536, doi:10.1007/s00521-022-07104-9.
4. Zhao, C.; Liu, R.W.; Qu, J.; Gao, R. Deep Learning-Based Object Detection in Maritime Unmanned Aerial Vehicle Imagery: Review and Experimental Comparisons. *Eng. Appl. Artif. Intell.* **2024**, *128*, 107513, doi:10.1016/j.engappai.2023.107513.
5. Yang, Z.; Yin, Y.; Jing, Q.; Shao, Z. A High-Precision Detection Model of Small Objects in Maritime UAV Perspective Based on Improved YOLOv5. *J. Mar. Sci. Eng.* **2023**, *11*, 1680, doi:10.3390/jmse11091680.
6. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the Computer Vision – ECCV 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, 2014; pp. 740–755.
7. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-Time Object Detection 2024.
8. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information 2024.
9. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection 2024.
10. Chen, G.; Pei, G.; Tang, Y.; Chen, T.; Tang, Z. A Novel Multi-Sample Data Augmentation Method for Oriented Object Detection in Remote Sensing Images. In Proceedings of the 2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP); September 2022; pp. 1–7.

11. Zhang, Q.; Meng, Z.; Zhao, Z.; Su, F. GSLD: A Global Scanner with Local Discriminator Network for Fast Detection of Sparse Plasma Cell in Immunohistochemistry. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP); September 2021; pp. 86–90.
12. Ghiasi, G.; Cui, Y.; Srinivas, A.; Qian, R.; Lin, T.-Y.; Cubuk, E.D.; Le, Q.V.; Zoph, B. Simple Copy-Paste Is a Strong Data Augmentation Method for Instance Segmentation 2021.
13. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size 2016.
14. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications 2017.
15. Gholami, A.; Kwon, K.; Wu, B.; Tai, Z.; Yue, X.; Jin, P.; Zhao, S.; Keutzer, K. SqueezeNext: Hardware-Aware Neural Network Design 2018.
16. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks 2019.
17. Qin, D.; Leichner, C.; Delakis, M.; Fornoni, M.; Luo, S.; Yang, F.; Wang, W.; Banbury, C.; Ye, C.; Akin, B.; et al. MobileNetV4 -- Universal Models for the Mobile Ecosystem 2024.
18. Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3 2019.
19. Zhang, J.; Chen, Z.; Yan, G.; Wang, Y.; Hu, B. Faster and Lightweight: An Improved YOLOv5 Object Detector for Remote Sensing Images. *Remote Sens.* **2023**, *15*, 4974, doi:10.3390/rs15204974.
20. Gong, W. Lightweight Object Detection: A Study Based on YOLOv7 Integrated with ShuffleNetv2 and Vision Transformer 2024.
21. Wang, J.; Chen, K.; Xu, R.; Liu, Z.; Loy, C.C.; Lin, D. CARAFE: Content-Aware ReAssembly of FEatures. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV); October 2019; pp. 3007–3016.
22. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection 2017.
23. 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; June 2018; pp. 8759–8768.
24. Xu, X.; Jiang, Y.; Chen, W.; Huang, Y.; Zhang, Y.; Sun, X. DAMO-YOLO : A Report on Real-Time Object Detection Design 2023.
25. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection 2020.
26. Li, K.; Geng, Q.; Wan, M.; Cao, X.; Zhou, Z. Context and Spatial Feature Calibration for Real-Time Semantic Segmentation. *IEEE Trans. Image Process.* **2023**, *32*, 5465–5477, doi:10.1109/TIP.2023.3318967.
27. Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; Cho, K. Augmentation for Small Object Detection 2019.
28. Guo, Y.; Li, Y.; Feris, R.; Wang, L.; Rosing, T. Depthwise Convolution Is All You Need for Learning Multiple Visual Domains Available online: https://arxiv.org/abs/1902.00927v2 (accessed on 16 May 2024).
29. Chen, J.; Kao, S.; He, H.; Zhuo, W.; Wen, S.; Lee, C.-H.; Chan, S.-H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks 2023.
30. Ma, N.; Zhang, X.; Zheng, H.-T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proceedings of the Computer Vision – ECCV 2018; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, 2018; pp. 122–138.
31. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features From Cheap Operations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); June 2020; pp. 1577–1586.
32. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks 2017.
33. Shi, D. TransNeXt: Robust Foveal Visual Perception for Vision Transformers 2024.
34. Yang, G.; Lei, J.; Zhu, Z.; Cheng, S.; Feng, Z.; Liang, R. AFPN: Asymptotic Feature Pyramid Network for Object Detection 2023.
35. Ding, X.; Zhang, X.; Zhou, Y.; Han, J.; Ding, G.; Sun, J. Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs 2022.
36. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks 2017.

37. Ding, X.; Zhang, Y.; Ge, Y.; Zhao, S.; Song, L.; Yue, X.; Shan, Y. UniRepLKNet: A Universal Perception Large-Kernel ConvNet for Audio, Video, Point Cloud, Time-Series and Image Recognition 2024.
38. Xu, J.; Fan, X.; Jian, H.; Xu, C.; Bei, W.; Ge, Q.; Zhao, T. YoloOW: A Spatial Scale Adaptive Real-Time Object Detection Neural Network for Open Water Search and Rescue From UAV Aerial Imagery. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–15, doi:10.1109/TGRS.2024.3395483.