remote sensing

MDPI

Article

SCCMDet: Adaptive Sparse Convolutional Networks Based on Class Maps for Real-Time Onboard Detection in Unmanned Aerial Vehicle Remote Sensing Images

Qifan Tan [1], Xuqi Yang [2], Cheng Qiu [1,*], Yanhuan Jiang [3], Jinze He [2], Jingshuo Liu [2] and Yahui Wu [3]

[1] School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, China; tanqf@bjtu.edu.cn
[2] School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China; yxq_thu@163.com (X.Y.); he_jinze@163.com (J.H.); liujingshuo424@163.com (J.L.)
[3] Changcheng Institute of Metrology and Measurement, Beijing 100095, China; jiangyh010@avic.com (Y.J.); wuyh021@avic.com (Y.W.)
* Correspondence: chqiu@bjtu.edu.cn

Abstract: Onboard, real-time object detection in unmaned aerial vehicle remote sensing (UAV-RS) has always been a prominent challenge due to the higher image resolution required and the limited computing resources available. Due to the trade-off between accuracy and efficiency, the advantages of UAV-RS are difficult to fully exploit. Current sparse-convolution-based detectors only convolve some of the meaningful features in order to accelerate the inference speed. However, the best approach to the selection of meaningful features, which ultimately determines the performance, is an open question. This study proposes the use of adaptive sparse convolutional networks based on class maps for real-time onboard detection in UAV-RS images (SCCMDet) to solve this problem. For data pre-processing, SCCMDet obtains the real class maps as labels from the ground truth to supervise the feature selection process. In addition, a generate class map network (GCMN), equipped with a newly designed loss function, identifies the importance of features to generate a binary class map which filters the image for its more meaningful sparse features. Comparative experiments were conducted on the VisDrone dataset, and the experimental results show that our method accelerates YOLOv8 by 41.94% at most and increases the performance by 2.52%. Moreover, ablation experiments demonstrate the effectiveness of the proposed model.

Keywords: object detection; remote sensing; sparse convolution; onboard; real time; UAV-RS

Citation: Tan, Q.; Yang, X.; Qiu, C.; Jiang, Y.; He, J.; Liu, J.; Wu, Y. SCCMDet: Adaptive Sparse Convolutional Networks Based on Class Maps for Real-Time Onboard Detection in Unmanned Aerial Vehicle Remote Sensing Images. *Remote Sens.* **2024**, *16*, 1031. https://doi.org/10.3390/rs16061031

Academic Editor: Joaquín Martínez-Sánchez

Received: 24 November 2023
Revised: 1 March 2024
Accepted: 11 March 2024
Published: 14 March 2024

1. Introduction

With the increasing popularity of drones and advancements in computer vision technology, unmanned aerial vehicle (UAV)-based imaging technology has gained widespread application. Due to their flexibility and universal applicability, drones have significant advantages in low-altitude remote sensing applications. Neural network-based general object detection has achieved great success, with an increasing number of state-of-the-art (SOTA) methods improving the accuracy of object detection on various public datasets. In combination with object detection models, UAVs can be widely used in the fields of environmental monitoring, agricultural protection, power grid inspection, and disaster response. Due to limitations in edge resources, current methods primarily rely on cloud computing, with the drone transmitting images back to a cloud server for inference in pursuit of higher accuracy. However, cloud computation not only requires a good communication environment, but also has a communication delay, which is not suitable for real-time tasks. There are, therefore, still some challenges in the application of unmanned aerial vehicle remote sensing (UAV-RS), such as onboard real-time object detection, which requires achieving fast processing speeds for high-resolution images using the limited onboard computational resources. This paper is focused on the onboard detection of UAV-RS images.

Recently, many researchers have attempted to solve the problem of the cloud processing of aerial images from drones, aiming to address issues such as small objects or unevenly distributed objects, varying flight altitudes, high resolutions, and constantly changing perspectives in aerial images. These researchers have designed more complex models for higher accuracy in the detection of small objects in high-resolution aerial images; however, cloud detection cannot be used in all aspects. In some aspects, UAVs need onboard detection to provide feedback on their actions because cloud detection is not suitable for real-time communication tasks. Therefore, onboard detectors are needed to achieve fast inference and low latency in lightweight models deployed on the resource-constrained edge units equipped on UAVs. Many efforts have been made to address the dilemma of balancing accuracy and efficiency, focusing mainly on onboard detection. The authors of [1–3] mainly focus on reducing the complexity of the computational backbone or designing a backbone suitable for edge devices. In addition, several methods have been proposed to address this issue, including network pruning [4,5] and model distillation, which have been proven effective at accelerating inference. These methods are not novel methods and can be used to accelerate any model by sacrificing performance. In addition, the uniform down-sampling method [6,7] balances accuracy and speed by reducing the resolution of the original image and changing the size of meaningful areas. However, the reasonable scaling of meaningful areas is challenging, especially in drone images.

Sparse convolution [8] is another promising alternative that limits computational performance by employing convolutions only on sparsely sampled regions or channels via a learnable mask. While theoretically attractive, the practical results are highly dependent on selecting meaningful regions, since the focal region of the learning mask in sparse convolution tends to lie within the foreground. Furthermore, DynamicHead [9] and QueryDet [10] apply sparse convolutions to the detection head; unfortunately, their primary goal is to offset the increased computational cost when additional feature maps are jointly used for performance gain on general object detection. They both follow the traditional approach in original sparse convolutions of setting fixed mask ratios or focusing on the foreground only, and are thus far from reaching the optimal trade-off between accuracy and efficiency required by UAV detectors. CASEC [11] proposes an AMM approach to find a mask that strikes the optimal balance between accuracy and efficiency. However, this method only pays attention to the activity ratio of the mask, and disregards the activity position of the mask. In addition, this method is concerned with the inaccuracy of the foreground and proposes a context-enhanced (CE) method to recover part of the background, which increases the computation and inference time to some extent. The CE module appears to compensate for the loss of information from sparse convolution, which is a sub-optimal approach. As a result, it is particularly important to find an optimal approach to efficiently and completely extract meaningful regions in remote sensing images, and how to extract the entire meaningful region for sparse-convolution-based onboard detection for UAV-RS images is still and open question.

In this study, the proposed SCCMDet method is focused on how to best extract the entire meaningful region from an image. For the task of object detection, the most meaningful regions are the areas that contain objects. In this view, SCCMdet is a method designed to map the meaningful regions of real class maps that have the same size as the feature maps. The class maps comprise the masks used in CASEC and the query mask in QueryDet. We refer to them as class maps because they have the same use as a class token used in Vit. In class maps, the value of each point represents the proportion of meaningful regions, and the position represents the region's location. We will introduce the proposed method in detail. In addition, a structure called the generate class map network (GCMN) generates a class map under the supervision of a real class map. This study also designs a loss function for the class map to improve the accuracy of class map production, which is important for sparse convolution. These methods can be easily extended to various sparse-convolution-based detectors for improving the accuracy of the selection in meaningful areas. In summary, we make three main contributions in our work:

(1) This paper proposes the SCCMDet model for real-time, onboard sparse-convolution-based detection of UAV-RS images. We design real class maps from the ground truth images to supervise the selection of sparse features, thus accelerating the inference speed.
(2) For the GCMN structure, we propose a method for obtaining a real class map and design a loss function to evaluate the performance of the class map generated by the GCMN to feed back to the network. The GCMN has the ability to generate more meaningful sparse features for improving the performance of sparse convolution-based detectors.
(3) We evaluate the proposed approach on a major public dataset. The results show that the proposed approach significantly reduces computational costs. The performance on a VisDrone shows that our method accelerates the operation of the YOLOv8 model by 41.94% at most and increases the performance by 2.52%.

2. Related Works

2.1. Object Detection

The main objective of object detection is to classify and localize target objects using bounding boxes. With the continuous development of convolutional neural networks (CNNs), researchers have divided object detection methods into two-stage and one-stage object detection. Two-stage object detection methods include two main steps: candidate box generation and object classification with position regression. In the first stage, a series of candidate boxes that may contain the target is generated. Then, in the object classification and position regression stage, these candidate boxes are analyzed to determine whether they actually contain targets, and the targets' positions are accurately determined. Representative algorithms in this category include the RCNN and its variants, which achieve high detection accuracy. However, these algorithms are still far from achieving real-time performance in terms of speed. Typical networks include Faster-RCNN and Mask RCNN [12–14].

On the other hand, one-stage object detection involves uniformly sampling different scales and aspect ratios directly at various positions in the image, followed by feature extraction using CNNs to perform classification and regression. Some popular algorithms in this category are the YOLO series models and the SSD algorithm [15,16]. However, one-stage algorithms still have various shortcomings, and even the improved SSD algorithm is surpassed by two-stage algorithms like Faster-RCNN. To improve the performance of one-stage algorithms, attempts have been made to insert a coordinate attention module into the backbone network, such as in the MCS-YOLO algorithm [17]. Other efforts include using deeper network architectures with a greater number of residual and convolutional blocks, as well as introducing more data augmentation methods like random cropping, random rotation, and updated activation functions, as demonstrated by the YOLO-V7 algorithm [18].

Recently, a new type of anchor-free detection method has emerged that does not rely on anchor-based methods and focuses on improving computational cost, training speed, and inference speed. The FCOS method is a pioneering work in this direction. By eliminating the complex computations involving predefined anchor boxes, FCOS completely avoids the need for anchor-based operations, such as overlap calculation during the training process, while reducing memory consumption. Furthermore, since only non-maximum suppression (NMS) is adopted during post-processing, FCOS offers a simpler alternative to anchor-based one-stage detectors [19]. A recent research achievement, Unbiased Teacher v2, includes semi-supervised object detection techniques that provide favorable improvements to the regression branch [20]. However, until the advent of Transformer [21], attention mechanisms were not fully utilized in single- and two-stage object detection methods, regardless of whether anchor boxes were used. The reason for this may not be difficult to guess: it is challenging to determine how to best model the relationships between objects, given the variations in scale, position, category, and quantity, all of which are different

due to changes between images. Most CNN-based methods currently in use are relatively simple, having regular network structures that cannot handle more complex scenarios.

In this context, Relation Net [22] and DETR [23] are methods that utilize transformer-based attention mechanisms for object detection. DETR has shown the best performance so far. It uses CNNs to extract features from images, adds position encoding, and then inputs the serialized data into an encoder that leverages attention mechanisms to extract features. In the decoder stage, N initialized vectors are input, and each object query focuses on a different position. Through decoding, N vectors corresponding to the detected objects are generated. Finally, the neural network outputs the classes and positions of the targets.

In practical applications, such as remote sensing detection using drones and satellites, the detection accuracy and precise localization of remote sensing objects are often challenging due to their small and densely distributed shapes. Some improvements have been made in this regard, including partial modifications to the YOLO-V5 network structure and the integration of coordinate attention mechanisms in the YOLO-extract algorithm [24]. Another approach mentioned earlier is the incorporation of Transformers into the feature extraction layer, such as in RAST-YOLO [25]. This method proposes using the Swin Transformer as the backbone and leveraging the region attention mechanism as the feature extractor and utilizing the C3D module to fuse deep and shallow semantic information to optimize the multi-scale problem in remote sensing target detection.

2.2. Sparse Convolution

Traditional convolution treats each position in the input equally. However, for sparse data, this approach results in a significant waste of computational resources. Thus, how to reduce unnecessary computations has become a focus of research. The authors of [26] were the first to introduce sparse convolution, which reduces the computational cost of convolution by discarding the computation of irrelevant locations in sparse input data. To address the reduction in sparsity caused by convolution, Ref. [27] proposed a new approach named the submanifold sparse convolutional network. The output location is activated only when the corresponding input location is valid. This method avoids the generation of too many active locations. Methods like [28–31] accept dense data and trim unimportant portions based on masks to achieve efficient inference.

Several approaches have been proposed in previous studies to address this issue. Adding a network to learn to extract the foreground region in the image is a common approach [29,32]. For instance, SACT [33] predicts the halting score of each position in the residual structure to decide whether to mark a position as inactive, and then skips the subsequent calculation at these positions. Ref. [9] selects suitable regions from different FPN scales for concatenation. Ref. [30] treats the mask matrix as a probability field. Each point is sampled and calculated based on its probability, and the points not sampled are interpolated by the network.

To reduce the computational cost of high-resolution images, QueryDet [10] uses low-resolution feature maps to guide the sparse convolution of adjacent high-resolution feature maps. To adapt to the dramatic variations in foreground regions in drone images, Ref. [11] has designed an adaptive multi-layer masking method to optimize the mask ratio of the foreground.

3. Proposed Method

3.1. Overall Framework

The overall framework for adaptive sparse convolutional networks based on class maps for real-time onboard detection (SCCMDet) is introduced in Section 3.1. The details on how the real class maps are generated are presented in Section 3.2. The generate class map network (GCMN) and its functions are illustrated in Sections 3.3 and 3.4.

As shown in Figure 1, taking YOLOv8 as its base detector, SCCMDet replaces the detection head with adaptive sparse convolutional networks based on class maps (SCCM) in each FPN layer. Note that our SCCM approach is not limited to RetinaNet, as it can be

applied to any one-stage detector and to the region proposal network (RPN) in two-stage detectors with FPNs.

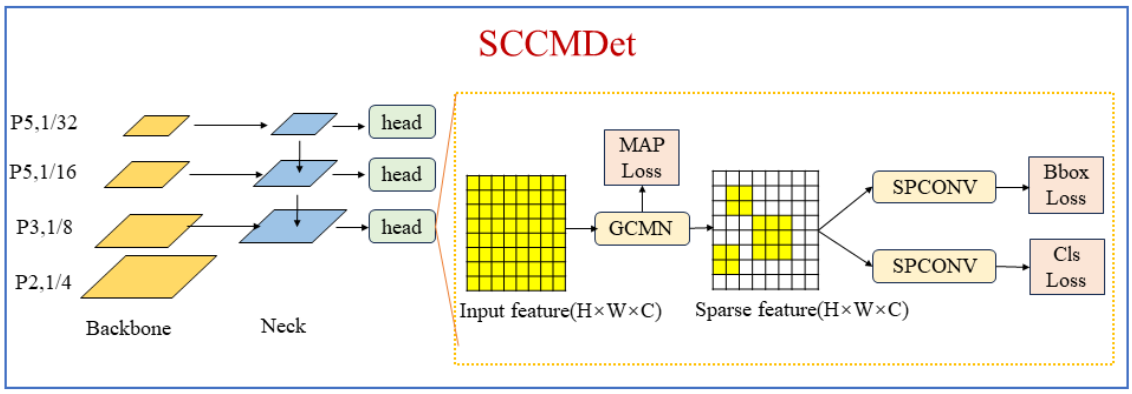


Figure 1. Framework of SCCMDet. Taking YOLOv8 a base detector, SCCMDet replaces the detection head with adaptive sparse convolutional networks based on class maps (SCCM) in each FPN layer. The generate class map network (GCMN) is designed to extract meaningful regions from input features under the supervision of a real class map.

In SCCM, the input is the features from the FPN. Through the GCMN, the input feature turns into a sparse feature, representing a spare meaningful region. The generate class map network (GCMN) is designed to extract the meaningful regions from input features under the supervision of a real class map. Class maps determine the sparse features, and therefore it is important to extract a reasonable class map from the input features. We define the pixels in the object as meaningful regions and define the real class map as a map that contains the activation ratio in each position. According to this definition, we can generate the real class map from the training dataset. The size of the real class map is different according to the feature maps of the different layers of the FPNs. For example, the real class map in the P5 layer is not the same size as the map of the same feature in the P4 layer. Moreover, the number of real class maps depends on the number of detector headers. Under the supervision of a real class map, the model can be trained to generate more reasonable class maps and sparse features to improve the accuracy of the sparse-convolution-based detectors. After generating the sparse features, sparse convolution replaces the YOLOv8's original convolution method, which can accelerate the execution of onboard UAV-RS detection tasks.

3.2. Class Map

Class maps, which supervise the delineation of meaningful areas, are the core of our proposed method. This section will introduce how to generate a class map from an original feature map.

As previously mentioned, we define the pixels in the object as meaningful pixels and define the real class map as a map that contains the activation ratio in each position. The original input image's size is $(H \times W \times 3)$; after extracting features through the backbone and FPN layers, the original input image, whose size is $(H \times W \times 3)$, is split into three features, whose size is $(H/n, w/n, channels)$, by the three FPN layers. Here, n is the down-sampling multiplier of the feature map compared to the original image. Each value of the feature map represents a receptive field; for example, the value in $(1, 1, :)$ represents the areas in the upper left corner of the original image, and the value in $(1, W/n, :)$ represents the areas in the upper left corner of the original image. The value is most importantly impacted by the areas of $(1 : H/n, 1 : W/n, 3)$. The value is also influenced by other areas, and the closer the distance, the greater the influence.

To select more meaningful areas from feature maps, class maps should be the same size as feature maps. Like feature maps, the value of a class map also depends on the receptive field. Feature maps depend on the RGB value in the receptive field, while class

maps depend on the number of pixels in the receptive field. Considering the receptive field is large when processed through deep convolutional networks and the impact differs according to distance, we choose the patch $((p-1)H/n : pH/n, (q-1)H/n : qW/n, 3)$ of the original image as the receptive field of class map (p, q). Therefore, the value of the real class map of position (p, q) is the rate of meaningful pixels in the patch $((p-1)H/n : pH/n, (q-1)H/n : qW/n, 3)$. The formula is as follows:

$$RF(p,q) = image(\frac{(p-1)H}{n} : \frac{pH}{n}, \frac{(q-1)w}{n} : \frac{qW}{n}, 3) \tag{1}$$

$$RFM(p,q) = IfMeaningful(i) \text{ , for } i \text{ in } RF(p,q) \tag{2}$$

$$IfMeaningful(i) = \begin{cases} 0 \text{ , if } i \text{ is a pixel in object} \\ 1 \text{ , if } i \text{ is not a pixel in object} \end{cases} \tag{3}$$

$$RCM(p,q) = \sum_{(1,1)}^{(n,n)} (j) \text{ , for } j \text{ in } RFM(p,q) \tag{4}$$

where RF represents the receptive field and RFM represents the meaningful pixel in RF. RCM is the real class map, and (p,q) represents the position of the class map.

The pseudo-code for this process is introduced below (Algorithm 1).

Algorithm 1: The algorithms to generate real class map.

```
Data: original image: Img, object informations: Objects, the mutiple of
      down-sampling : n, width of image : width, height of image : height,
      judgement whether a block in object: conf, coordinate of
      object:xmin/xmax/ymin/ymax,
Result: RCM
initialization;
while Imgs not null do
    Img = getImg(Img);
    height,width=getImgInfo(Img);
    RCM = zeros(height/n,width/n);
    xmin,ymin,xmax,ymax= getObjInfo(Objects);
    while Objects not null do
        for obj in Objects do
            conf(xmin:xmax,ymin:ymax)=1;
            //If the pixel in objects,turn conf into 1;
        end
        for i in height do
            for j in width do
                block← conf(i,j);
                block_ sum← ∑_{i=1}^{m} block;
                i← i+n;
                j← j+n;
                RCM(i/n,j/n)← block_ sum/n^2;
            end
        end
        return RCM;
    end
end
```

As shown in Figure 2, the feature consists of eight multiple down-samplings through the backbone and FPN layers. In the sample patch (6,7), there are 8×8 pixels in total and

20 in the objects. As a result, position (6,7) of the real class map is $\frac{20}{64}$. After calculating the position of each real class map and each layer of the FPN, the process of generating real class maps is complete.

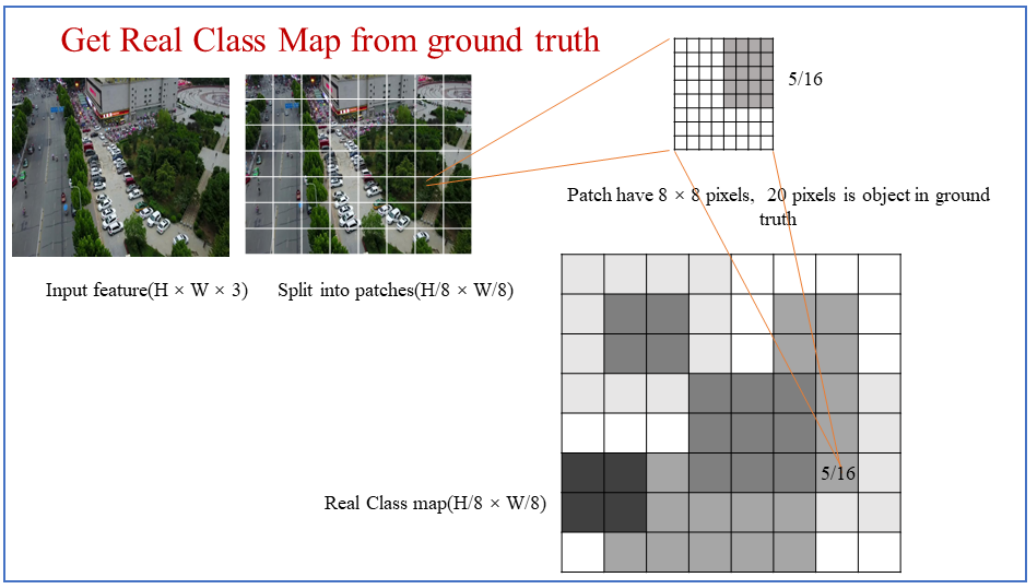


Figure 2. The process of generating a real class map from a ground truth.

3.3. The Generating Class Map Network

After obtaining the real class map, the generating class map network (GCMN) is designed to generate a class map under the supervision of the real class maps (Figure 3).

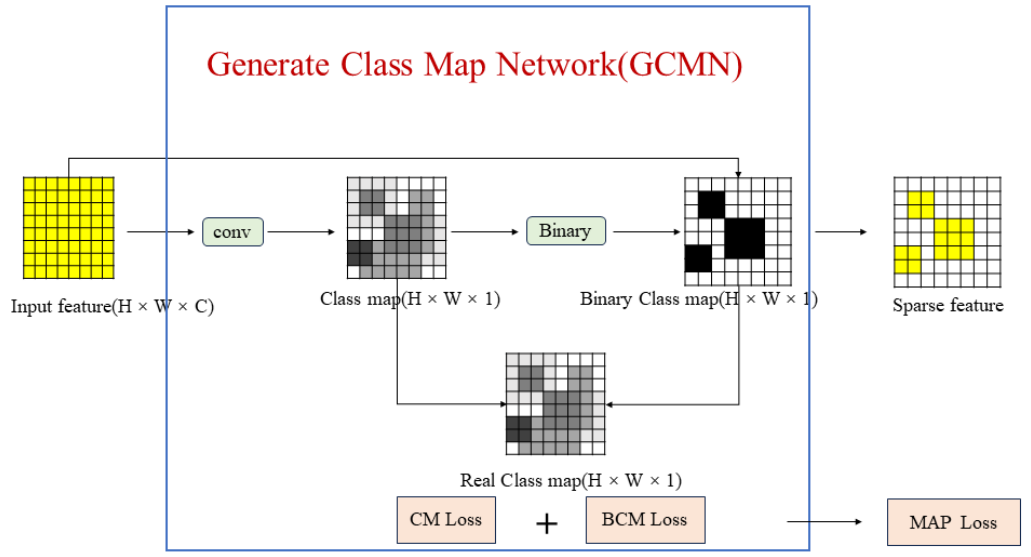


Figure 3. The process of generating a real class map from a ground truth.

The real class map of the FPN layer and the feature map of the same layer are the inputs of the GCMN. After some convolutional layers, the $(H \times W \times C)$ feature maps turn into a $(H \times W \times 1)$ class map. The values of the class map are between 0 and 1, which represents the meaningful area rate in this patch. In addition, a learnable threshold T, which depends on the learnable parameter l, is set to binarize the class map. The formula for this step is as follows:

$$T = \frac{2}{5}\frac{1}{1+e^{-p}} + \frac{1}{10} \tag{5}$$

$$Binary(i) = \begin{cases} 1 \text{ , if } i > T \\ 0 \text{ , if } i < T \end{cases} \tag{6}$$

$$CM(p,q) = Binary(CM(p,q)) \tag{7}$$

where T is the threshold to binarizing the class map with a value between 0.1 and 0.5, l is the learnable parameter, $Binary()$ is the binary function, and CM is the class map.

Then, the real class map is used to supervise the generation of further class maps based on the feature maps. Our method proposes a metric to evaluate the equality of the class map and design a loss function according to the metrics. This metric is simply the mean square error (MSE) between the generated class map and the real class map and this method can really represent the error between a class map and a real class map. However, this simple method presents a problem in that a learnable threshold T cannot be trained by a real class map. As a result, the binary class map can sometimes differ greatly from the real class map. This means that the binary class map should also be supervised by the real class map. The value of the binary class map is either 0 or 1, and the threshold T is between 0 and 0.5. The MSE is not a suitable metric for the two maps because the rate of meaningful areas is not high in drone images. To solve this class imbalance problem, our method introduces a focal loss function between BCM and RCM. In addition, there is a negative phenomenon that the binary class map is also likely to fix the real class map by setting it to 0 by learning a higher threshold T. The original purpose of introducing the real class map is not only to supervise the generated class map but also to supervise the binary class map. Therefore, another loss function to fit this situation is designed. The formula for this loss function is as follows:

$$Loss_{CM} = \sum (CM_{(p,q)} - RCM_{(p,q)})^2 \tag{8}$$

$$p_t = \begin{cases} RCM_{p,q} \text{ , if the position } (p,q) \text{ of } BCM = 1 \\ 1 - RCM_{p,q} \text{ , otherwise} \end{cases} \tag{9}$$

$$Loss_{BCM} = \sum_{0,0}^{n,n} (-(1-p_t)\cdot log(p_t)) \tag{10}$$

$$Loss_{map} = \alpha \cdot Loss_{CM} + (1-\alpha)\cdot Loss_{BCM} \tag{11}$$

where $Loss_{CM}$ and $Loss_{BCM}$ represent the loss of the class map and binary class map, respectively; $Loss_{map}$ is the total loss of the class map; and α is a manual parameter set between 0 and 1.

Finally, the $Loss_{total}$ can be defined as

$$Loss_{total} = w1\cdot Loss_{map} + w2\cdot Loss_{cls} + w3\cdot Loss_{reg}, \tag{12}$$

where $Loss_{cls}$ and $Loss_{reg}$ are same as the base detector and $w1$, $w2$, and $w3$ are the weights for each loss.

Compared with an adaptive multi-layer masking (AMM) module, our proposed method does not use the activation rate but rather the activation of areas. SCCMDet can adaptively choose the activation areas using the GCMN, and focuses on more meaningful areas in images to improve performance. The rate of the AMM module is the average rate of a dataset, and although this setting can meaningfully improve the performance of the method, it tends to predict the activation rate in an image as that of the dataset's. For example, if the rate in VisDrone is 37%, for an image with no objects, AMM will predict a mask with an approximately 37% activation rate, which causes unnecessary computation.

3.4. Sparse Convolution Networks

Most existing detection heads for UAV images convolve on the whole set of feature maps. Making full use of background information can be advantageous for small object detection. However, it will result in a significant computational cost, which is not applicable for resource-constrained UAV platforms. Moreover, the foreground regions occupy only a small portion of the image. Therefore, traditional convolution wastes a substantial amount of computational resources on useless information. This observation reveals the potential to accelerate the detection head's performance by focusing on the foreground regions.

Sparse convolution has only recently been proposed. Sparse convolution learns to focus on foreground regions by using sparse masks. A diagram of this process is shown in Figure 4. The use of sparse convolution can reduce the number of unnecessary computations performed on background regions and effectively speed up the inference phase on a variety of vision tasks. To understand sparse convolution, we first need to comprehend the computational formula for sparse convolution.

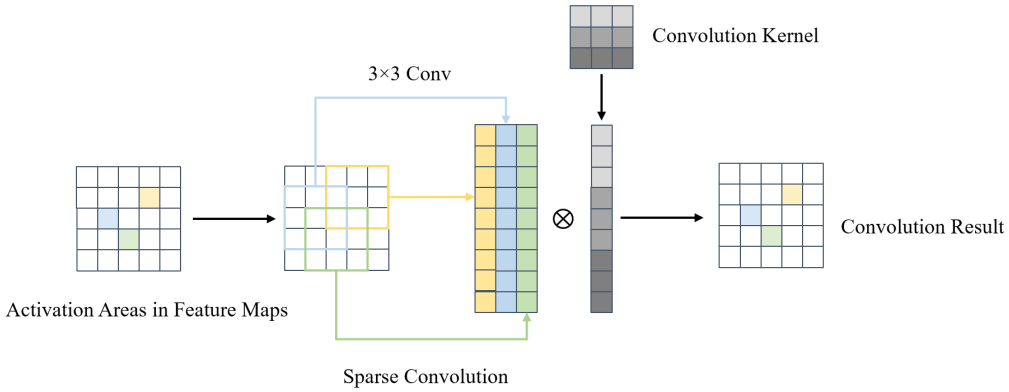


Figure 4. Activation areas considered as meaningful regions are put into sparse convolution. Only the meaningful regions and their neighboring regions need to be calculated; other regions are ignored.

An input feature s with input channels c_{in} and output channels c_{out} and data dimension d is processed by a kernel w of size $K(w \in R^{c_{in} \times K^d \times c_{out}})$. For sparse convolution, the input feature is restricted to P_{in}, where P_{in} represents the effective region containing the foreground. The output feature is P_{out}. At position p, the sparse convolution is represented as follows:

$$y_p = \sum_{k \in X_p} w_k \cdot s_{p_k} \tag{13}$$

$pk = p + k$ represents the positions around p, where k is an offset distance from p and k enumerates all discrete locations in the kernel space X. X_p is a subset of X, leaving out the empty positions of X, calculated as follows:

$$X_p = \{k \mid p + k \in P_{in}, k \in X\} \tag{14}$$

If P_{out} includes a union of all dilated positions around P_{in} and covered by X, this process is formulated as follows:

$$P_{out} = \bigcup_{p \in P_{in}} P_n \tag{15}$$

where P_n is

$$P_n = \{p + k \mid k \in X\} \tag{16}$$

Under these conditions, the formula represents conventional sparse convolution. However, this method extends the position containing the foreground region in subsequent layers, thus increasing the computational cost.

When $P_{in} = P_{out}$, this form of sparse convolution is called submanifold sparse convolution. Convolution in this method is performed when the center of the convolution kernel is in the foreground region. This design reduces the number of computations and is thus commonly used. However, this approach discards essential information from the background, which degrades the model's capabilities

4. Experiments and Results

4.1. Dataset

We adopt the VisDrone [34] dataset for training and evaluation in our experiments. VisDrone is designed for UAV platforms and consists of 7019 aerial images belonging to 10 categories, including pedestrians, people, cars, vans, buses, trucks, motorcycles, bicycles, canopies of tricycles, and tricycles. Of note, a person is classified as a pedestrian if they are standing or walking in the image; otherwise, they are classified as people. Meanwhile, objects with a truncation ratio greater than 50% are skipped during the computation. Images in VisDrone contain different scenes to ensure variety in the training data. The dataset is collected from a UAV's perspective. Therefore, VisDrone offers a broader field of view and enables researchers to conduct studies more effectively compared to other datasets. Moreover, VisDrone provides abundant annotation information, including occlusion ratios and truncation ratios, which assist in evaluating the conditions of the objects and in the adjustment of our algorithms and models.

4.2. Evaluation Metrics

We employ the mean average precision (mAP) as the metric for evaluating accuracy, as well as FPS as the metric for efficiency.

TP is defined as a positive sample that is classified as true; FP is a negative sample classified as true; TN is a negative sample classified as false; and FN is positive sample classified as false. Moreover, *Precision* and *Recall* are calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

where *Precision* means the rate of accuracy in our prediction and *Recall* means the rate of accuracy of identifying positive samples. When we adjust the IOU threshold from 0 to 1, we obtain a series of values for *Precision* and *Recall*, and these values can be used to form a curve.

The AP is the integral of the curve. $AP_{IOU=0.50}$ means that the IOU larger than 0.5 is a TP. $AP_{IOU=0.50,0.05,0.95}$ means that the intersection to union ratio starts from 0.5 and increases at an interval of 0.05, taking a final value of 0.95, and the mean is calculated from this.

The mAP is obtained by averaging the $AP_{IOU=0.50,0.05,0.95}$ across all classes and is used to measure the accuracy of the model. It is a comprehensive metric that can evaluate the performance across multiple classes of the model. A higher mAP means a better performance in object detection. The $mAP_{@50}$ is obtained by averaging the $AP_{IOU=0.50}$.

FPS (frames per second) represents the number of image frames processed by the model per second and is used to measure the model's efficiency at handling image data. It can be calculated as follows:

$$FPS = \frac{num}{t} \tag{19}$$

where num is the number of images collected in time t.

4.3. Implementation Details

We implement our approach based on PyTorch [35] and a Python package called spconv. All models are trained on NVIDIA 4090 GPUs and tested on an NVIDIA 3090 GPU. For VisDrone, the batch size is set to 8. The optimizer used is the stochastic gradient

descent (SGD) with an initial learning rate of 0.01. The weights of each loss are set to 0.2, 0.4, and 0.4. The value of α in $Loss_{map}$ is set to 0.6.

In particular, the real class maps are used as the input labels when training, and each image has three maps, one for each FPN layer.

4.4. Baseline

To evaluate the performance of the model, we used eight methods as a baseline: Cascade RCNN, RetinaNet, YOLOV5, YOLOV8, YOLOX, DETR, QueryDet, and CEASC (RetinaNet). The following is a brief description of the eight methods:

Cascade RCNN [36]: Cascade R-CNN is an object detection algorithm that features a cascade structure. Its fundamental idea is to progressively improve detection accuracy through multiple stages of cascaded detectors. Each stage of the cascade builds upon the previous stage's output, further processing and refining it to arrive at more precise detection results. This cascade structure effectively reduces the number of false positives and enhances the detector's accuracy.

RetinaNet [37]: RetinaNet is a state-of-the-art object detection algorithm that addresses the class imbalance problem between the foreground and background in object detection. It introduces a novel loss function called focal loss, which effectively down-weights the contribution of a large number of easily classified negative samples during training and focuses more on the hard-to-classify samples.

YOLOV5n: Compared to previous versions, YOLOv5 employs a more efficient network architecture, allowing for real-time detection across a range of hardware devices. Additionally, it utilizes sophisticated training techniques and data augmentation strategies, greatly enhancing the model's overall performance.

YOLOV8n: YOLOv8 represents the latest iteration in the real-time object detection algorithm series known as YOLO (You Only Look Once). This version combines the strengths of its predecessors with new technological advancements, resulting in a significant improvement in both speed and accuracy.

YOLOX [38]: YOLOX is a cutting-edge object detection algorithm that builds upon the YOLO series, introducing a range of innovations and optimizations. It leverages an efficient network architecture that integrates techniques such as CSPNet, a SiLU activation function, and PANet, enabling it to excel in real-time object detection tasks.

DETR [39]: DETR, short for Detection Transformer, is a groundbreaking model for object detection tasks. It marks the first time that the Transformer architecture has been adapted for object detection, enabling a truly end-to-end approach.

QueryDet [10]: QueryDet is a novel algorithm designed for object detection tasks. It employs a query-based approach to detect objects, utilizing a set of predicted query vectors to represent targets' locations and category information. Compared to traditional object detection methods, QueryDet offers greater flexibility and efficiency.

CEASC (RetinaNet) [11]: A context-enhanced adaptive sparse convolutional network (CEASC) that first develops a context-enhanced group normalization layer.

4.5. Comparisons of Performance

Figures 5 and 6 illustrate the performance of our method and benchmark methods on the VisDrone dataset. The detail is shown in Table 1. The results show that SCCMDET performs the best. We can draw five primary conclusions from the table:

(1) Cascade RCNN is a two-step detector, and DETR is a transformer-based detector. Cascade RCNN is the best method and the DETR is the second best method in terms of accuracy due to its detector-type long latency. SCCMDet is a single-step detector, so its accuracy is a bit lower than that of Cascade RCNN and DETR. Although the performances of these models are good, they are are too slow to infer an image in real time and are thus not suitable for UAV-RS tasks.

(2) YOLO-based detectors always achieve a balance between accuracy and inference speed. As a result, SCCMDet chooses YOLOv8 as a base detector.

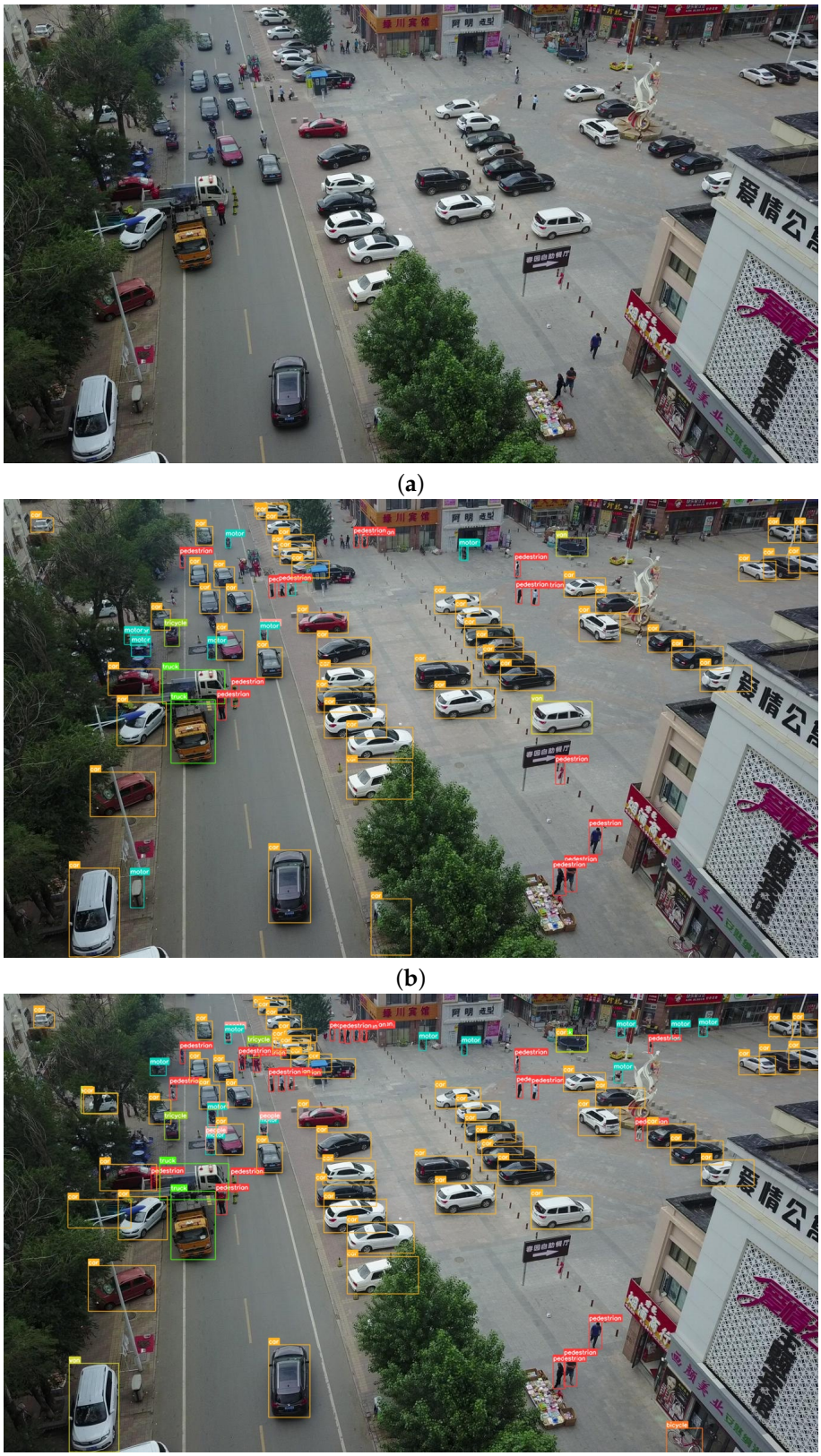
(3) Compared with sparse-convolution-based detectors like QueryDet and CEASC, in VisDrone, our method is better because the other networks use RetinaNet as their base detectors. More experiments comparing the details of these methods are carried out in the ablation study.
(4) Compared with other methods, our method's use of YOLOv8 as a base detector in VisDrone results in a performance enhanced by 2.52% in terms of accuracy and by 41.94% in terms of inference speed.

(**a**)

(**b**)

(**c**)

Figure 5. Comparison 1 of original image (**a**), YOLOv8n detector (**b**), and SCCMDet (**c**) based on YOLOv8n.



(**a**)

(**b**)

(**c**)

Figure 6. Comparison 2 of original image (**a**), YOLOv8n detector (**b**), and SCCMDet (**c**) based on YOLOv8n.

Table 1. Comparison of baseline methods with SCCMDet.

Method	$mAP_{@50}$	FPS
Cascade RCNN [36]	**54.81**	5
RetinaNet [37]	40.24	11
YOLOV5	49.84	33
YOLOV8 (base detector)	50.66	31
YOLOX [38]	50.37	34
DETR [39]	53.74	3
QueryDet [10]	48.14	13
CEASC + RetinaNet [11]	46.27	15
Ours	51.94	**44**
improve%	2.52	41.94

4.6. Ablation Study

4.6.1. GCMN, CSQ, and AMM

To prove the usefulness of the GCMN, we compared its performance to that of CSQ and AMM. CSQ was first proposed in paper and AMM was first proposed in paper. All of these methods are used to generate masks to select meaningful regions in feature maps to input sparse convolution layers. As shown in Table 2, we have observed that the accuracy of the YOLOv8 detector is higher than that of RetinaNet due to YOLOv8 being more powerful.

Table 2. Ablation learning on resolution, mask methods, and base detector.

Base Detector	Method	Image Size	$mAP_{@50}$	mAP	FPS
YOLOv8	Baseline	640	35.56	20.72	62
	CSQ (QuerryDet)		34.49	20.31	66
	AMM (CEASC)		34.85	20.41	70
	Ours		33.06	18.41	73
RetinaNet	Baseline		26.48	15.88	17
	CSQ (QuerryDet)		25.13	15.03	20
	AMM (CEASC)		24.37	14.83	23
	Ours		23.53	14.62	23
YOLOv8	Baseline	1280	50.67	32.59	31
	CSQ (QuerryDet)		52.01	33.34	36
	AMM (CEASC)		50.97	31.63	45
	Ours		51.94	33.17	44
RetinaNet	Baseline		40.28	21.76	11
	CSQ (QuerryDet)		41.05	21.85	13
	AMM (CEASC)		40.21	14.8	16
	Ours		41.47	22.09	16

In addition, the resolution of the image has a great influence on the sparse-convolution-based detector. Compared with baseline, the accuracy of the three methods all decreased on the image with a resolution of 640×640, while they all increased on the higher-resolution 1280×1280 image. To explain this phenomenon, we posit that the sparse convolution method can effectively extract the foreground and eliminate the noise from the background in the high-resolution image. However, sparse convolution also loses some important information when the resolution is low. Overall, sparse convolution is suitable for onboard detection in UAV-RS images, and we will analyze the results on an image with a 1280×1280 resolution.

Compared with the AMM method, the GCMN has nearly the same inference speed but performs better in terms of accuracy. The AMM method is a simple method used to generate masks without real labels. As a SOTA approach, CAESC mainly depends on the context-enhanced (CE) module to increase its accuracy. In contrast, the CSQ method has

the highest accuracy using the YOLOv8 detector and the second highest accuracy using the RetinaNet detector, albeit with the slowest inference speed. The CSQ method employs full convolution in the P5 layer and needs to perform inferences in a certain order.

In conclusion, the GCMN, which simply makes use of the label from the ground truth, shows better performance in both accuracy and inference speed.

4.6.2. On BCM and RCM

Figures 7 and 8 show a comparison between the RCM from the ground truth and the BCM from the GCMN. The white regions in the BCM represent the meaningful regions which will be input into the sparse convolution layer. When there are fewer regions to be computed, the computational demand will evidently be decreased. As we expected, the inference time is significantly shorter using sparse convolution, and this method has the potential to be used onboard UAV platforms for remote sensing detection at low heights.

The loss function is designed to reduce the number of errors between the BCM and RCM in order to find meaningful regions. The BCM should be the same as the RCM in each layer. However, there is a significant gap in the BCM in each layer, especially the P5 layer, in which there is not even a region selected to input into the next layer. In addition, the P3 and P4 layers tend to select regions with objects whose size is suitable to predict in the P3 or P4 layer. We suppose that the CM in different layers not only has the ability to find meaningful regions, but also has the ability to identify the scale of the objects in these selected regions. There are almost no large objects in the VisDrone dataset, and as a result, the BCM in the P5 layer has fewer white regions.

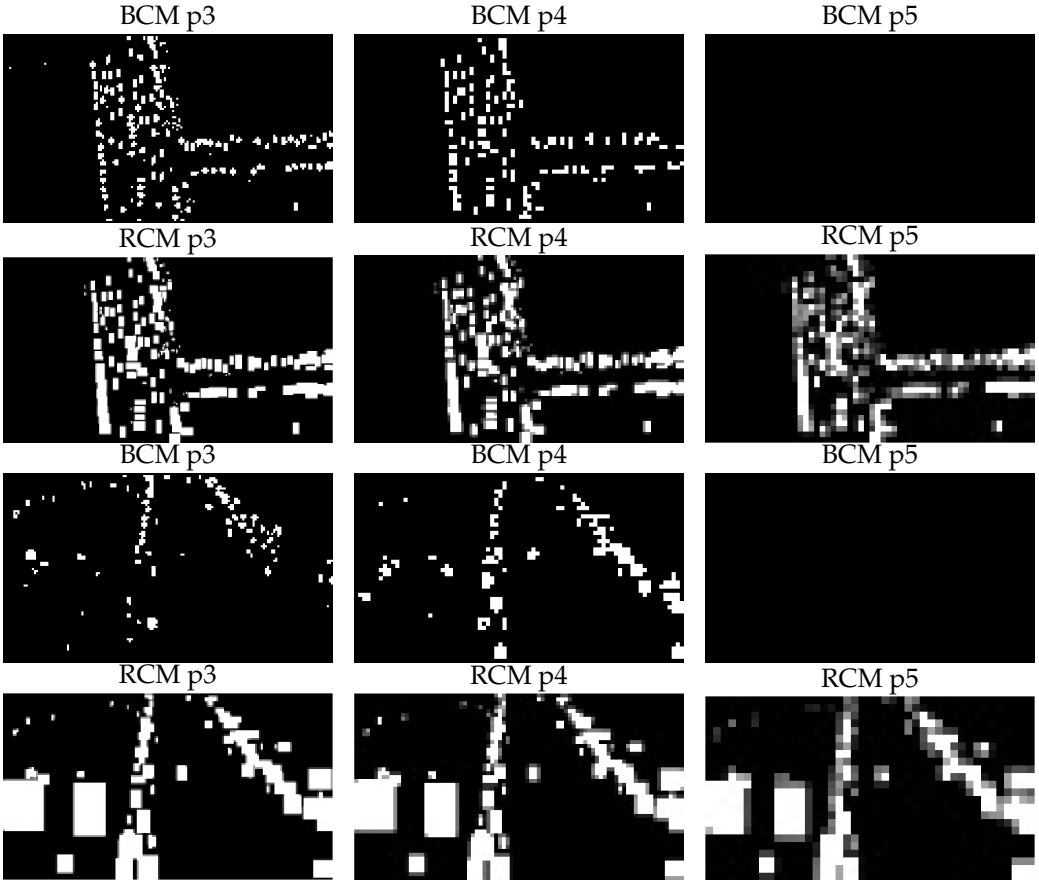


Figure 7. Comparison 1 of real class map with binary class map.

In addition, the RCM in each layer is still different from the BCM layer. We inspect the loss function to ensure it is convergent and find that the total loss has decreased to a low level and is stable. However, the $Loss_{map}$ is decreased to a moderate level. If we set a large

weight for $Loss_{map}$, the $Loss_{map}$ will be decreased to a low level, but the $Loss_{reg}$ becomes incorrect. To explain this phenomenon, we posit that if the weight for $Loss_{map}$ is too large, the backbone and FPN layers tend to be a classifying network, which is not suitable for the final remote sensing detection task.

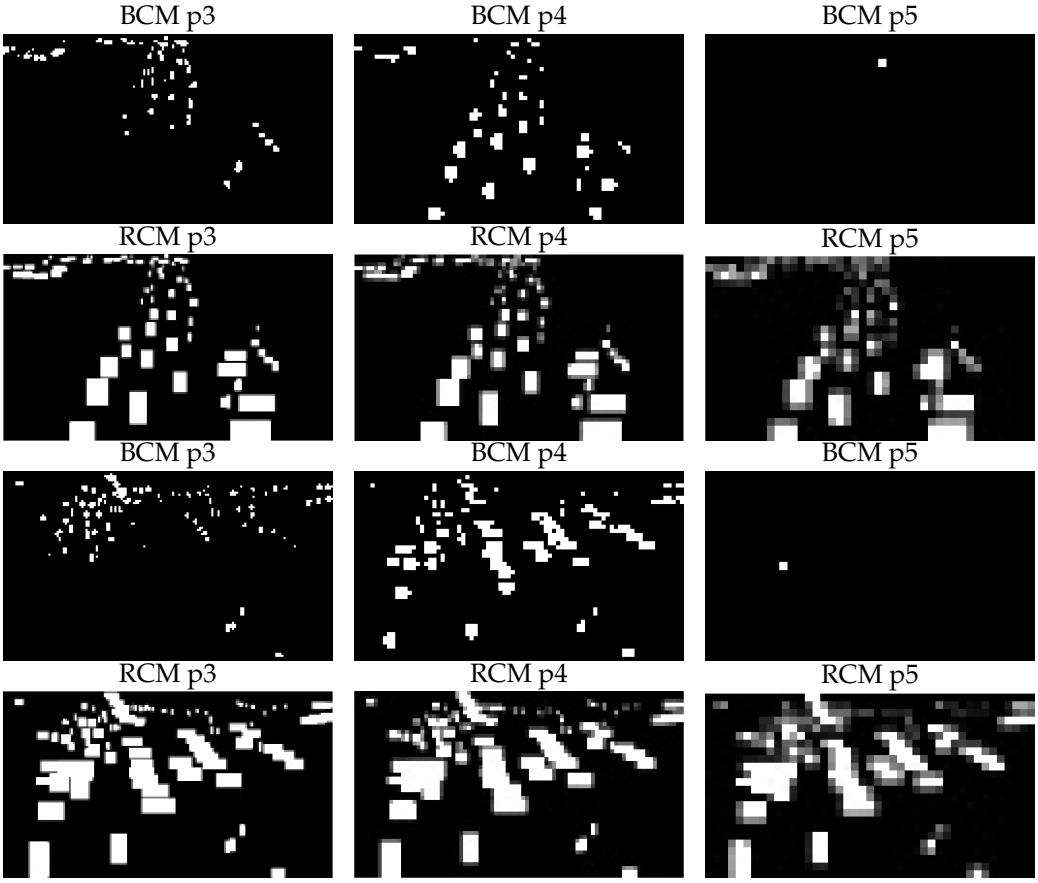


Figure 8. Comparison 2 of real class map with binary class map.

4.6.3. Class Performance

To gain a better understanding of the model's performance for different kinds of classes within the Visdrone dataset, the performance table for various classes is shown in Table 3. If the $mAP_{@50}$ of our method is 5% higher than YOLOv8's, we define that as high. If the $mAP_{@50}$ of our method is 5% lower than YOLOv8's, we define that as low. Otherwise, the mAP is considered moderate.

Compared with YOLOv8, SCCMDet's APs are high in the car, van, truck, and bus categories and low in the bicycle and motorcycle categories. SCCMDet can improve the accuracy of the detection of large objects while maintaining the accuracy of small objects like people and pedestrians. However, in the bicycle and motorcycle classes, the accuracy is very low.

Table 3. Performance table for various classes.

Method	$mAP_{@50}$	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	Awning-Tricycle	Bus	Motor
RetinaNet	54.81	55.4	32.5	9.9	82.6	74.4	52.4	31.3	17.6	80.1	42.5
Cascade RCNN	40.24	43.2	24.3	10.6	69.1	53.5	39.9	22.6	14.0	60.5	34.5
YOLOv5	49.84	48.0	30.5	9.2	76.2	67.0	44.5	26.7	17.0	76.1	40.2
YOLOv8	50.66	51.4	39.2	23.1	79.5	44.3	42.0	35.9	18.1	53.1	52.4
QueryDet	48.14	45.5	28.6	8.2	71.0	65.9	46.7	26.6	14.7	63.9	38.5
CEASC	46.27	44.1	27.2	7.9	74.4	62.9	44.3	26.0	15.1	63.7	37.1
Ours	51.94	49.5	35.5	8.8	85.5	70.6	49.8	29.2	17.0	71.5	35.7
Compared with YOLOv8	moderate	moderate	moderate	low	high	high	high	moderate	moderate	high	low

5. Discussions and Conclusions

When comparing our results to those of previous studies, it must be pointed out that our method makes use of the ground truth to train the activation regions and uses sparse convolution to accelerate the inference speed, and this makes it complex to train. The performance of our method is enhanced by 2.52% in terms of accuracy and by 41.94% in terms of inference speed. From the ablation study, we can learn that the GCMN is more efficient at activating the meaningful regions. The sparse convolution method and the backbone are the main reasons behind the accelerated inference speed in our proposed model. In addition, SCCMDet can improve the accuracy of detecting large objects while still maintaining the accuracy of detecting small objects like people and pedestrians. However, why the accuracy of detecting motors and bicycles is so poor remains an open question. It is possible that it is a problem of data distribution.

Moreover, future iterations of edge deployment may in fact demonstrate even greater potency. For example, if we deploy our method using TensorRT or ONNX, the inference speed will be accelerated greatly. However, employing the TensorRT operator on a GPU is not suitable for performing sparse convolution. Designing a TensorRT operator suitable for sparse convolution on a GPU is a direction of our next work. For now, SCCMDet is suitable for platforms which have restricted resources but need a real-time detector, especially for UAV-RS.

This paper focuses on enhancing the selection of meaningful parts of RS images by convolving them via sparse convolution and accelerating the inference speed. This study proposes an adaptive sparse convolutional network based on class maps for real-time onboard detection in UAV-RS images (SCCMDet), which solves the selection problem for sparse convolution. From the data pre-processing perspective, SCCMDet obtains the real class maps as labels from the ground truth to supervise the selection process. In addition, the generated class map network (GCMN), equipped with a newly designed loss function, explores the importance of features to generate a binary class map that filters for more meaningful features. Comparative experiments show that our method is suitable for a platform that has restricted resources but needs a real-time detector.

Author Contributions: Methodology, Q.T.; Software, J.H.; Validation, X.Y.; Resources, Y.J.; Writing—original draft, X.Y.; Writing—review & editing, J.L. and Y.W.; Visualization, J.H.; Project administration, C.Q.; Funding acquisition, C.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by The Fund of the Science and Technology on Metrology and Calibration Laboratory (Grant No. JLKG2021001B001) and The Talent Fund of Beijing Jiaotong University (Grant No. 2022RC014).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

References

1. 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. *arXiv* **2017**, arXiv:1704.04861.
2. Sandler, M.; Howard, A.G.; Zhu, M.; Zhmoginov, A.; Chen, L. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *arXiv* **2018**, arXiv:1801.04381.
3. Howard, A.; Pang, R.; Adam, H.; Le, Q.V.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.; Tan, M.; Chu, G.; et al. Searching for MobileNetV3. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Republic of Korea, 27 October–2 November 2019; IEEE: Hoboken, NJ, USA, 2019; pp. 1314–1324. [CrossRef]
4. Tessier, H. Convolutional Neural Networks Pruning and Its Application to Embedded Vision Systems (Élagage de Réseaux de Neurones Convolutifs et son Application aux Systèmes Embarqués de Vision par Ordinateur). Ph.D. Thesis, IMT Atlantique Bretagne Pays de la Loire, Brest, France, 2023.
5. Wang, J.; Cui, Z.; Zang, Z.; Meng, X.; Cao, Z. Absorption Pruning of Deep Neural Network for Object Detection in Remote Sensing Imagery. *Remote Sens.* **2022**, *14*, 6245. [CrossRef]

6. Thavamani, C.; Li, M.; Cebron, N.; Ramanan, D. FOVEA: Foveated Image Magnification for Autonomous Navigation. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, BC, Canada, 10–17 October 2021; IEEE: Hoboken, NJ, USA, 2021; pp. 15519–15528. [CrossRef]
7. Bejnordi, B.E.; Habibian, A.; Porikli, F.; Ghodrati, A. SALISA: Saliency-Based Input Sampling for Efficient Video Object Detection. In Proceedings of the Computer Vision-ECCV 2022—17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13670, pp. 300–316. [CrossRef]
8. Yan, Y.; Mao, Y.; Li, B. SECOND: Sparsely Embedded Convolutional Detection. *Sensors* **2018**, *18*, 3337. [CrossRef] [PubMed]
9. Song, L.; Li, Y.; Jiang, Z.; Li, Z.; Sun, H.; Sun, J.; Zheng, N. Fine-grained dynamic head for object detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11131–11141.
10. Yang, C.; Huang, Z.; Wang, N. QueryDet: Cascaded sparse query for accelerating high-resolution small object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 13668–13677.
11. Du, B.; Huang, Y.; Chen, J.; Huang, D. Adaptive Sparse Convolutional Networks with Global Context Enhancement for Faster Object Detection on Drone Images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, USA, 18–22 June 2023; pp. 13435–13444.
12. Girshick, R.B.; 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, CVPR 2014, Columbus, OH, USA, 23–28 June 2014; IEEE Computer Society: Hoboken, NJ, USA, 2014; pp. 580–587. [CrossRef]
13. Ren, S.; He, K.; 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] [PubMed]
14. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 386–397. [CrossRef] [PubMed]
15. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976.
16. Yan, C.; Zhang, H.; Li, X.; Yuan, D. R-SSD: Refined single shot multibox detector for pedestrian detection. *Appl. Intell.* **2022**, *52*, 10430–10447. [CrossRef]
17. Cao, Y.; Li, C.; Peng, Y.; Ru, H. MCS-YOLO: A Multiscale Object Detection Method for Autonomous Driving Road Environment Recognition. *IEEE Access* **2023**, *11*, 22342–22354. [CrossRef]
18. Wang, C.; Bochkovskiy, A.; Liao, H.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, CVPR 2023, Vancouver, BC, Canada, 17–24 June 2023; IEEE: Hoboken, NJ, USA, 2023; pp. 7464–7475. [CrossRef]
19. Wang, T.; Zhu, X.; Pang, J.; Lin, D. FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, 11–17 October 2021; IEEE: Hoboken, NJ, USA, 2021; pp. 913–922. [CrossRef]
20. Liu, Y.; Ma, C.; Kira, Z. Unbiased Teacher v2: Semi-supervised Object Detection for Anchor-free and Anchor-based Detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022; IEEE: Hoboken, NJ, USA, 2022; pp. 9809–9818. [CrossRef]
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
22. Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P.H.S.; Hospedales, T.M. Learning to Compare: Relation Network for Few-Shot Learning. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; Computer Vision Foundation/IEEE Computer Society: Hoboken, NJ, USA, 2018; pp. 1199–1208. [CrossRef]
23. Choi, H.K.; Paik, C.K.; Ko, H.W.; Park, M.; Kim, H.J. Recurrent DETR: Transformer-Based Object Detection for Crowded Scenes. *IEEE Access* **2023**, *11*, 78623–78643. [CrossRef]
24. Liu, Z.; Gao, Y.; Du, Q.; Chen, M.; Lv, W. YOLO-Extract: Improved YOLOv5 for Aircraft Object Detection in Remote Sensing Images. *IEEE Access* **2023**, *11*, 1742–1751. [CrossRef]
25. Jiang, X.; Wu, Y. Remote Sensing Object Detection Based on Convolution and Swin Transformer. *IEEE Access* **2023**, *11*, 38643–38656. [CrossRef]
26. Graham, B. Spatially-sparse convolutional neural networks. *arXiv* **2014**, arXiv:1409.6070.
27. Graham, B.; Van der Maaten, L. Submanifold sparse convolutional networks. *arXiv* **2017**, arXiv:1706.01307.
28. Su, H.; Jampani, V.; Sun, D.; Gallo, O.; Learned-Miller, E.; Kautz, J. Pixel-adaptive convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 11166–11175.
29. Verelst, T.; Tuytelaars, T. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2320–2329.

30. Xie, Z.; Zhang, Z.; Zhu, X.; Huang, G.; Lin, S. Spatially adaptive inference with stochastic feature sampling and interpolation. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 531–548.
31. Qin, R.; Huangfu, L.; Hood, D.; Ma, J.; Huang, S. Kernel Inversed Pyramidal Resizing Network for Efficient Pavement Distress Recognition. In Proceedings of the 29th International Conference, ICONIP 2022, Virtual Event, 22–26 November 2022; Springer: Berlin/Heidelberg, Germany, 2022; Volume 1793, pp. 302–312. [CrossRef]
32. Chen, Y.; Li, Y.; Zhang, X.; Sun, J.; Jia, J. Focal sparse convolutional networks for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5428–5437.
33. Figurnov, M.; Collins, M.D.; Zhu, Y.; Zhang, L.; Huang, J.; Vetrov, D.; Salakhutdinov, R. Spatially adaptive computation time for residual networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1039–1048.
34. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Fan, H.; Hu, Q.; Ling, H. Detection and tracking meet drones challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 7380–7399. [CrossRef] [PubMed]
35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv* **2019**, arXiv:1912.01703.
36. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving Into High Quality Object Detection. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; Computer Vision Foundation/IEEE Computer Society: Hoboken, NJ, USA, 2018; pp. 6154–6162. [CrossRef]
37. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002.
38. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv* **2021**, arXiv:2107.08430.
39. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. *arXiv* **2020**, arXiv:2005.12872.