remote sensing

MDPI

Article

TPH-YOLOv5++: Boosting Object Detection on Drone-Captured Scenarios with Cross-Layer Asymmetric Transformer

Qi Zhao [1], Binghao Liu [1], Shuchang Lyu [1], Chunlei Wang [1] and Hong Zhang [2,*]

[1] Department of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2] School of Astronautics, Beihang University, Beijing 100191, China
* Correspondence: dmrzhang@buaa.edu.cn

Abstract: Object detection in drone-captured images is a popular task in recent years. As drones always navigate at different altitudes, the object scale varies considerably, which burdens the optimization of models. Moreover, high-speed and low-altitude flight cause motion blur on densely packed objects, which leads to great challenges. To solve the two issues mentioned above, based on YOLOv5, we add an additional prediction head to detect tiny-scale objects and replace CNN-based prediction heads with transformer prediction heads (TPH), constructing the TPH-YOLOv5 model. TPH-YOLOv5++ is proposed to significantly reduce the computational cost and improve the detection speed of TPH-YOLOv5. In TPH-YOLOv5++, cross-layer asymmetric transformer (CA-Trans) is designed to replace the additional prediction head while maintain the knowledge of this head. By using a sparse local attention (SLA) module, the asymmetric information between the additional head and other heads can be captured efficiently, enriching the features of other heads. In the VisDrone Challenge 2021, TPH-YOLOv5 won 4th place and achieved well-matched results with the 1st place model (AP 39.43%). Based on the TPH-YOLOv5 and CA-Trans module, TPH-YOLOv5++ can further increase efficiency while achieving comparable and better results.

Keywords: object detection; drone; transformer; tiny object; high-density scene

Citation: Zhao, Q.; Liu, B.; Lyu, S.; Wang, C.; Zhang, H. TPH-YOLOv5++: Boosting Object Detection on Drone-Captured Scenarios with Cross-Layer Asymmetric Transformer. *Remote Sens.* **2023**, *15*, 1687. https://doi.org/10.3390/rs15061687

Academic Editor: Pedro Melo-Pinto

Received: 16 January 2023
Revised: 16 March 2023
Accepted: 18 March 2023
Published: 21 March 2023

1. Introduction

Unmanned aerial vehicles (UAVs), also known as drones, are aircraft operated by radio-controlled devices or their own programmed controls and do not carry any people. Equipped with embedded devices and cameras, drones have been widely used in aerial photography, agriculture, express transportation, disaster relief, and many other fields [1–4] due to their advantages of small size, low cost, and convenient use. Meanwhile, research into object detection in drone-captured images has attracted attention in recent years.

Thanks to the emergence of large-scale benchmark datasets, such as MS COCO [5] and PASCAL VOC [6], deep convolutional neural networks (CNNs) [7–11] have made remarkable achievements in object detection tasks. However, most deep CNNs are designed for natural scene images, which do not work as well as expected on drone-captured images because of the significant differences between natural and drone-captured images.

Three typical problems are illustrated in Figure 1. First, as shown in row (a), the object scale and appearance vary considerably because the pitching angle and height of drones change greatly, and there are many extremely tiny objects in images. Second, we find in row (b) that objects in drone-captured images are usually in large quantities and often densely packed, which brings in occlusion between objects. Third, drone-captured images always contain confusing geographic elements because they cover a large area. As shown in row (c) of Figure 1, drone-captured images cover a wide range of scenes, including different weather scenes in different places at different times, leading to the diversity of object features. These three problems make object detection in drone-captured images very challenging.

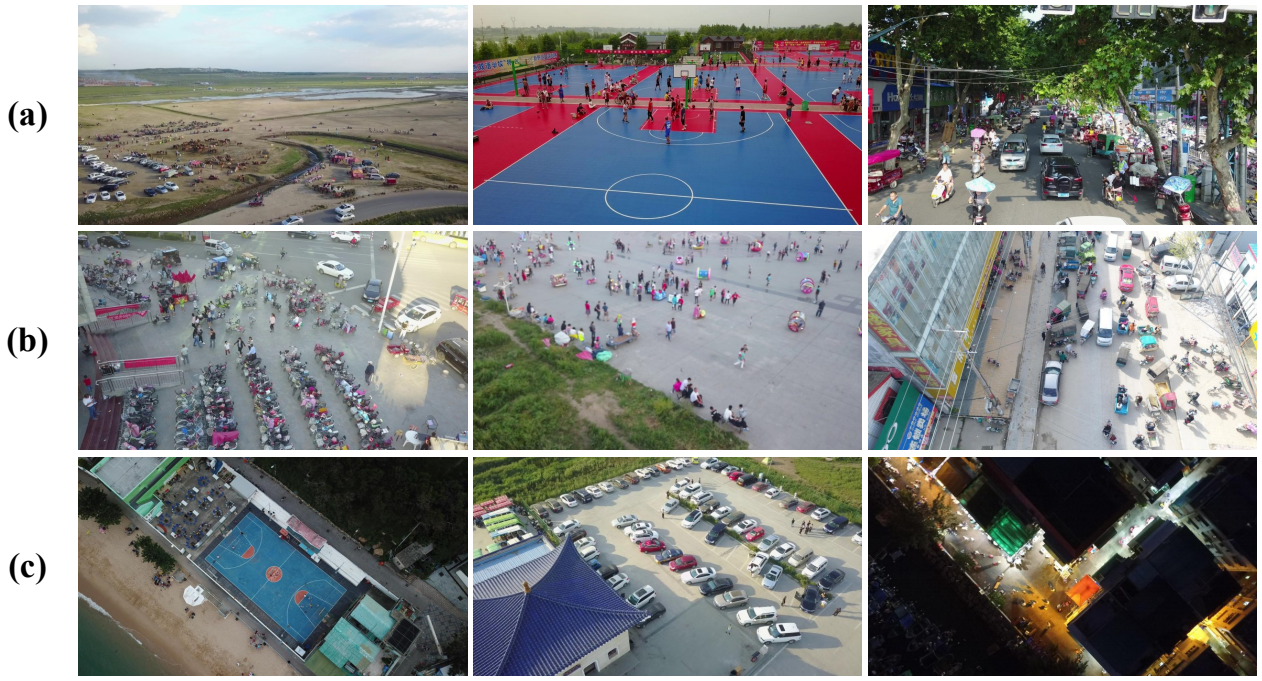


Figure 1. Intuitive cases to explain the three main problems in object detection in drone-captured images. The cases in row (**a**), (**b**), and (**c**), respectively, show the size variation, high-density, and large coverage of objects in drone-captured images.

In an object detection task, YOLO series [8,12–15] are representative methods for one-stage detectors. In this paper, we first propose an improved model, TPH-YOLOv5, to address the three main problems mentioned above. First of all, we find that YOLOv5 is insensitive to extremely small objects, so we add an additional head to YOLOv5 to utilize more information for tiny object detection. In total, our TPH-YOLOv5 has four detection heads separately used for the detection of tiny, small, medium, and large objects. Then, to make more effective use of context information, we replace the original CNN-based prediction heads with transformer-based heads named transformer prediction head (TPH) [16,17]. To further improve the performance of TPH-YOLOv5, we employ several tricks, including convolutional block attention module (CBAM) [18], data augmentation, multi-scale testing (ms-testing), multi-model ensemble inference strategies, and an auxiliary classifier for two confusing classes, "tricycle" and "awning-tricycle".

Although TPH-YOLOv5 can detect objects in drone-captured images with high performance, the additional prediction head introduces a large amount of computing resources and is time consuming. By analyzing the experiment results, compared to YOLOv5, we find TPH-YOLOv5 has more than a 60% increase in GPU memory usage and nearly a 50% decrease in frames per second (FPS). Meanwhile, TPH-YOLOv5 has negligible performance improvement on datasets that rarely have high-density scenes, such as UAVDT [19]. In TPH-YOLOv5, the prediction boxes generated by the additional head can account for nearly 80% of the total boxes, making it very suitable for dealing with high-density local areas. However, a non-negligible number of false positive prediction boxes hinder the performance of TPH-YOLOv5 on datasets like UAVDT [19].

To solve this problem and improve the generalization of our TPH-YOLOv5, we propose TPH-YOLOv5++ with a cross-layer asymmetric transformer (CA-Trans) to remove the additional prediction head while preserving the performance of detecting large-scale varying objects and high-density scenes. The overview of the working pipeline using TPH-YOLOv5++ is shown in Figure 2. By visualizing the results of the additional head and small object prediction head, we find the latter can detect a significant portion of the former's results. Due to the severe occlusion between objects, the small object prediction

head misses the objects with fewer features, while the additional head captures all the features. The CA-Trans introduces a transformer-based module between tiny and small paths, as shown in Figure 2, extracting asymmetric information between two paths to enrich the features of the small path. Because the resolutions of the two paths are extremely large for the transformer module and each pixel of features in the small path only needs the nearby pixels of ones in the tiny path, we propose a sparse local attention (SLA) to generate local self-attention and reduce the computational cost. The SLA generates a sparse relation matrix to explore the correlation between each pixel in small path features and its local area in tiny path features. Then, based on a relation reverse operation, SLA extracts the asymmetric information between two paths and fuses it to features of the small path. The obtained features cover the rich information of two paths and maintain the efficiency of the small path.

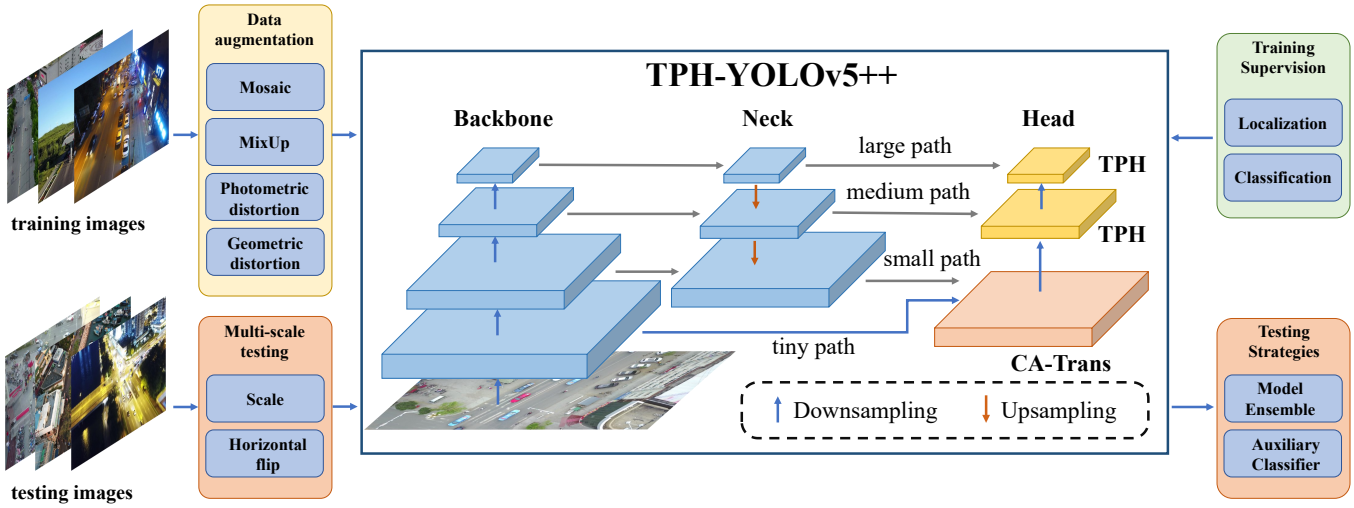


Figure 2. Overview of the working pipeline using TPH-YOLOv5++. Compared to the original YOLOv5, our TPH-YOLOv5 proposes an addition prediction head, four transformer prediction heads (TPH), and tricks to improve detection performance. Based on TPH-YOLOv5, TPH-YOLOv5++ designs a cross-layer asymmetric transformer (CA-Trans) to replace the tiny object detection head and transfer knowledge to the small object detection head.

Compared to TPH-YOLOv5, our TPH-YOLOv5++ greatly improves detection efficiency, maintains state-of-the-art (SOTA) performance, and even further improves the detection performance of some datasets. This is an extension of our previous work [20]. Our code is available at https://github.com/cv516Buaa/tph-yolov5.

Our contributions are summarized as follows:

- By adding one more prediction head and integrating a transformer, CBAM, and a bag of tricks, we propose TPH-YOLOv5 to effectively solve scale variation, high-density scenes, and large coverage of objects in drone-captured images.
- To improve efficiency while maintaining detection performance of TPH-YOLOv5, we further propose TPH-YOLOv5++ by designing the CA-Trans module to enrich the features of small path.
- On the VisDrone2021 test-challenge dataset, TPH-YOLOv5 achieves 39.18% (AP), won 4th place in the VisDrone2021 DET challenge, and has a minor gap compared to the 1st place models. Extensive experments show that our TPH-YOLOv5++ achieves SOTA results in VisDrone [21] and UAVDT [19] datasets and significantly reduces the computational and inference time costs of TPH-YOLOv5.

2. Related Work

2.1. Object Detection

Deep neural networks (DNNs) based models play a very important role in object detection tasks, and these methods can be divided into different types according to different criteria. Based on whether the model produces candidate boxes first for subsequent detection, we can divide models into one-stage detectors and two-stage detectors. YOLO series models [8,12–15] and SSD [9] are classical one-stage detectors, while Faster R-CNN [7] and Cascade R-CNN [22] are two-stage detectors. Based on whether there are predefined anchors in a working pipeline, we can divide models into anchor-based detectors and anchor-free detectors. The anchor-based methods family has Faster R-CNN [7] and YOLOv5 [15], while CornerNet [23], CenterNet [24], YOLOX [25], and RepPoints [26] are famous anchor-free detectors.

Faster R-CNN [7] proposes a region proposal network (RPN) promoting single-stage detectors as the preferred approach in early DNN-based object detection. YOLOv1 [8] and SSD [9] realize single-stage detection by matching each pixel on the feature map to specific anchors, achieving fast detection with competitive performance. FPN [27] uses multiple features with different resolutions to generate multiple prediction results. By using feature maps of different resolutions to detect objects of different sizes, FPN further improves the performance of previous methods and is widely used in object detection models. To mitigate the influence of foreground–background class imbalance on one-stage detectors, Lin et al. [10] designed the focal loss, which down-weights the loss assigned to well-classified examples, and they proposed RetinaNet to verify the effectiveness. To solve the problems caused by inference–time mismatch between the IoUs and overfitting during training, Cascade R-CNN [22] is proposed with a sequence of detectors trained with increasing IoU thresholds. In DNN-based detectors, there are always two choices in designing the structure of prediction head: a shared module for both localization and classification or two different modules for localization and classification. For the latter, to understand how these two head structures work for these two tasks, Wu et al. [28] performed a thorough analysis and proposed a new double-head method that uses a fully-connected head focusing on classification and a convolution head for bounding box regression. Revisiting the FPN architecture [27] used in most object detection methods, YOLOF [29] points out that the success of FPN is due to its divide-and-conquer solution rather than multi-scale feature fusion. By combining dilated encoder and uniform matching, YOLOF realizes a fast single-in-single-out method with comparable performance as multiple-in-multiple-out methods. To eliminate the network design difficulties caused by predefined anchor boxes, CornerNet [23] views the object detection task as finding a pair of keypoints for each object; these two keypoints are the top left and bottom right points of the object's bounding box. CenterNet [24] further develops the idea of CornerNet by adding a center point to detect each object as a triplet.

Due to the fact that the number of targets in an image is unknown, DNN-based detectors need to generate a lot of prediction boxes to ensure that all targets can be detected, which leads to a quantity of redundant invalid prediction boxes. Detectors should therefore use non-maximum suppression (NMS) to filter the overlapped boxes. However, if an object lies within the predefined overlap threshold, it leads to a miss. Soft-NMS [30] is proposed to decay the detection scores of boxes that are overlapped with other boxes with higher scores, so no box is eliminated in this process. However, WBF [31] aims to fuse different prediction boxes rather than eliminate part of them. By using information from all prediction boxes, WBF can obtain more reasonable results.

2.2. Object Detection in Drone-Captured Images

In recent years, drones have been widely used in plenty of application scenes, and object detection in drone-captured images has also attracted the attention of many researchers. Based on the emergence of large-scale standard drone-captured image datasets [19,21], DNN-based methods have made significant breakthroughs.

The design of object detection methods in drone-captured images is difficult due to the dynamic changes of flight height and angle, wide coverage scenes, and various types of objects involved in the image collection process of drones. Because one image covers a fairly large area, most of which does not have an object, the large background leads to inefficient detection. ClusDet [32] unifies object clustering and detection in an end-to-end framework by sequentially finding clustered areas and detecting objects in these areas. Similarly, Zhang et al. [33] proposed a difficult region estimation network to find a difficult high density area for further detection. Aiming to address vehicle detection challenges caused by diversity in drone-captured images, AdNet [34] seeks to align features between different viewpoints, illumination, weather, and background following the idea of domain adaptation. Tiny scale objects and unevenly distributed objects severely hinder the performance of detection models as discussed in ClusDet [32]. GLSAN [35] adds an efficient self-adaptive region, selecting an algorithm for the global–local detection network, finding high density areas, and detecting objects with large size variation accurately. Similar to ClusDet [32] and GLSAN [35], DMNet [36] proposes a novel crop strategy guided by a density map, removing the area without objects and balancing the information of the foreground and background. Yu et al. [37] analyzed the detection results of DMNet [36] and found that it has a explicit performance degradation on a long-tail scene. They designed a DSHNet [37] to handle head classes and tail classes separately by combining class-biased samplers and bilateral box heads. MDCT [38] designs a multi-kernel dilated convolution (MDC) block and transformer block to identify small objects in dense scenes. Gallo et al. [39] utilized the YOLOv7 model to solve the challenge caused by the existence of unstructured crop conditions and the high biological variation of weeds. RAANet [40] constructs a new residual ASPP by embedding the attention module and residual structure into the ASPP, to deal with the variability and complex background problems of land use in high-resolution imagery. HawkNet [41] proposes an up-scale feature aggregation framework to fully utilize multi-scale complementary information. CDMNet [42] formats density maps into coarse-grained form and designs a lightweight dual task density estimation network. FiFoNet [43] effectively selects a combination of multi-scale features for an object and block background interference, which further revitalizes the differentiability of the multi-scale feature representation. TPH-YOLOv5 [20] combines a transformer-based prediction head and the YOLOv5 detection model, realizing significantly performance improvement in large size variation and high density scenes.

In order to promote the application of a deep learning algorithm on drones in real-time scenes, many works focus on designing fast and lightweight models for high quality object detection in drone-captured images. UAV-Net [44] analyzes influences from different backbone architectures, prediction heads, and model pruning methods comprehensively and constructs a better combination to realize fast object detection. GDFNet [45] uses a global density model to jointly extract density information from multiple-level pyramid features, which is faster than most models based on pyramid feature fusion architecture. RHFNet [46] utilizes a bidirectional fusion architecture to fully use multi-layer features, efficiently realizing small object detection. By summarizing the disadvantages of the one-stage detectors as a mismatch of bounding box classification and the inadequate ability of only one-time regression, HSD [47] proposes a novel reg-offset-cls module and a stacked strategy implementing precision and speed at the same time. Integrating the specialized feature extraction and information fusion techniques, SODNet [48] effectively improves small object detection ability with high real-time performance. Dividing the high-resolution input image into a number of chips still introduces a heavy computational cost, so UFPMP-Det [49] merges sub-regions given by a coarse detector into a mosaic for a single inference, further promoting the efficiency of detection.

2.3. Vision Transformer

Transformer [16] has been proposed for machine translation tasks and has since become the state-of-the-art method in many NLP tasks. ViT [17] first introduced the

transformer to the computer vision field by taking the image as multiple patches. Each patch contains a certain number of pixels, and ViT does not regard pixels as a two-dimensional structure; instead it extracts attention between every pair of patches. By obtaining attention on a patch from all other patches, we can refine the features of this patch with the help of others.

However, as the input image resolution increases, the computational and storage demand of ViT increases a lot, which makes it hard to apply it to high-resolution computer vision tasks. To solve this issue, Swin–Transformer [50] uses a shifted windowing scheme to realize efficient computation of Transformer by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection. To further expand the cross-window connection, CSWin–Transformer [51] proposes cross-shaped window self-attention mechanisms for computing self-attention in the horizontal and vertical stripes, promoting the connection in the global perspective. The above-mentioned Transformer models can only be applied if query, key, and value have same shape, so CrossViT [52] proposes a novel Transformer-based module that can be used between features with different spatial sizes. CrossViT first utilizes two ViTs for two features separately. Then, CrossViT exchanges the class tokens of two features and extracts cross attention between two features. V2X-ViT [53] proposes V2X communication using a novel vision Transformer to achieve accurate 3D object detection. CoBEVT [54] designs a fused axial attention module (FAX) to realize bird's eye view semantic segmentation. MaxViT [55] consists of two aspects: blocked local and dilated global attention, which allows for global–local spatial interactions on arbitrary input resolutions with only linear complexity.

Transformer is also widely used in image object detection tasks. DETR [56] proposes an end-to-end architecture for object detection by regarding the task as a direct set prediction problem. However, in DETR, each object query will not focus on a specifc region. Anchor DETR [57] proposes a query design and an attention variant to make the object query focus on the objects near the anchor point. Based on the design idea of the YOLO series [8,12–15], YOLOS [58] proposes a series of Transformer-based object detection models.

3. Methodology

3.1. Overview of YOLOv5

YOLOv5 [15] uses CSPDarknet53 with a spatial pyramid pooling (SPP) module as the backbone, PANet as the neck, and a prediction head. To further improve the detection potential and robustness of YOLOv5, a bag of freebies and specials [14] are provided, such as mosaic data augmentation, multi-scale training strategy, and focal loss. Since it is the most notable and convenient one-stage detector, we select it as our baseline.

When we train the original YOLOv5 model on the VisDrone2021 dataset with multiple data augmentation (MixUp, Mosaic, Flip, Rotate, etc.), we find that YOLOv5x, the largest version of YOLOv5, has the best detection performance. The results of YOLOv5x are more than 1.5% better than YOLOv5s, YOLOv5m, and YOLOv5l (another, smaller version of YOLOv5) on AP value. We choose YOLOv5x as our baseline, even though it has more computational cost. In addition, we adjust the parameters of commonly used photometric distortions and geometric distortions.

3.2. TPH-YOLOv5

In order to make YOLOv5 deal better with the three main problems existing in drone-captured images mentioned above, we first modify the YOLOv5 and propose the TPH-YOLOv5, a general detector for object detection in drone scenes. The architecture of TPH-YOLOv5 is shown in Figure 3.

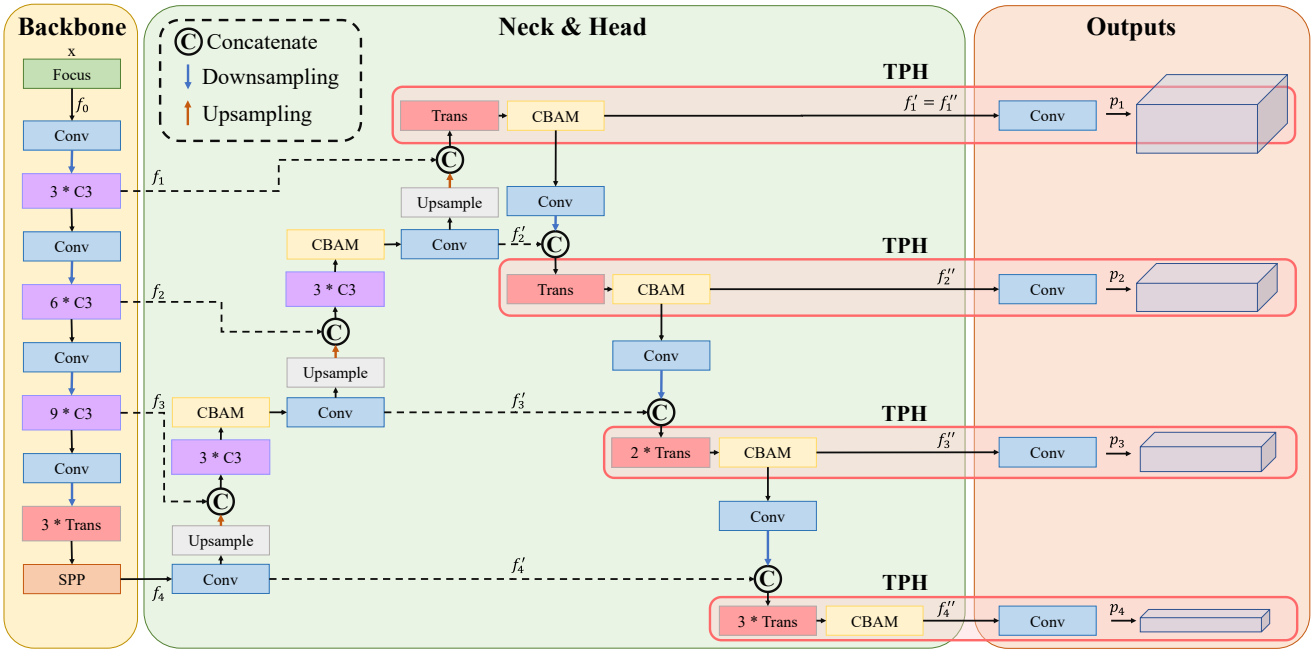


Figure 3. The architecture of TPH-YOLOv5. TPH-YOLOv5 introduces an additional head, transformer prediction head (TPH), and convolutional block attention module (CBAM). Four prediction heads are named tiny, small, medium, and large heads, and the branches before these heads are tiny, small, medium, and large paths.

3.2.1. Additional Prediction Head for Tiny Objects

Because the pitching angle and height of drones change greatly, the object scale and appearance vary considerably. On the other hand, it is very frequent for objects in the image to appear in high-density clusters. The experimental results show that the original YOLOv5 with three prediction heads can not work effectively in cases where many tiny objects appear as clusters.

We therefore add an additional prediction head to YOLOv5, promoting the performance of tiny object detection. As shown in Figure 3, the additional head takes low-level high-resolution features as input and fuses them with multi-layer high-level features, which is more sensitive to tiny objects. Although the additional prediction head introduces quite a lot of computation and memory costs, the performance of tiny object detection is greatly improved.

3.2.2. Transformer Prediction Head

Due to the confusing characteristics of objects in drone-captured images, objects of different categories can be quite similar in some very confusing scenes. It is necessary to adequately extract the long-range relationship between objects and other instances in the scene. Inspired by the vision transformer [17], we replace some CSPbottleneck modules with transformer encoder modules to utilize contextual information by generating attentions between every pixel pairs. The transformer encoder module increases the ability to capture different local information. It can also explore the feature representation potential with its self-attention mechanism [16].

However, the computation and memory costs required by transformer modules greatly increase as the input resolution increases. On the other hand, the transformer lacks some critical inductive biases inherent to CNNs, such as translation equivariance and locality. To solve the problems mentioned above, we only replace the last CSPbottleneck module of the backbone and four prediction heads. When enlarging the resolution of input images, we can reduce the number of replaced modules. For example, when we use 1996 $\times$ 1996 resolution for VisDrone2021 test-challenge datasets, we only replace the last one or two prediction

heads with transformer modules; as there are only two transformer blocks in each module, our TPH-YOLOv5 is flexible.

Compared to the original vision transformer module [17], the Swin-Transformer [50] reduces memory cost while increasing computation cost slightly. Therefore, we also utilize the Swin-Transformer module to construct our transformer prediction head (TPH), and all four prediction heads can be replaced by TPH simultaneously when the image resolution is 1996×1996.

3.2.3. Other Strategies

To efficiently take advantage of channel information and spatial information within the features, we utilize convolutional block attention module (CBAM) [18] after CSPbottleneck and transformer encoder modules, as shown in the light green part of Figure 3. In drone-captured images, large coverage of scenes always leads to confusing geographical elements. By introducing the CBAM module, our TPH-YOLOv5 can mitigate the influence of confusing information and focus on useful information that can clearly identify objects.

To further improve the detection performance on multi-scale objects, we implement a multi-scale testing (ms-testing) strategy during the inference phase. Our ms-testing first resizes the width and height of input images to 1, 0.83, and 0.67 times the original resolution, and then each image is flipped horizontally. After obtaining six different prediction results, we fuse them by weighted boxes fusion (WBF) [31] to get the final prediction result.

After analyzing the detection results of our TPH-YOLOv5 on the VisDrone2021 test-dev dataset, we find that there are some easily confusing categories, such as tricycle and awning-tricycle. To identify these two categories efficiently, we train an auxiliary classifier to distinguish tricycle and awning-tricycle in particular.

3.2.4. Overview of TPH-YOLOv5

To make the discussion easier, we define the model structure mathematically. As shown in Figure 3, defining the input images as x, we get $f_0 = Focus(x)$. The four-feature output from the backbone can be denoted as f_i, where $i = 1, \cdots, 4$. These four features can obtained as in Equation (1):

$$f_i = B_i(f_{i-1}), \quad i = 1, \cdots, 4 \tag{1}$$

where $B_i(\cdot)$ denotes the different blocks in the backbone; $B_1(\cdot)$, $B_2(\cdot)$, and $B_3(\cdot)$ are combinations of a convolutional (Conv) layer; there are 3, 6, or 9 CSPbottleneck modules; and $B_4(\cdot)$ is a combination of a Conv layer, three transformer modules, and an SPP module.

In the neck part, there are also four features, $f_i^{'}$, where $i = 1, \cdots, 4$. These four features can be formulated as follows:

$$f_i^{'} = \begin{cases} N_i\left(f_i, f_{i+1}^{'}\right), & i = 1, 2, 3 \\ Conv(f_4), & i = 4 \end{cases} \tag{2}$$

where $N_i(\cdot, \cdot)$ represents different blocks; $Conv(\cdot)$ denotes Conv layers; and $N_i(\cdot, \cdot)$ is formulated as in Equation (3):

$$N_i\left(f_i, f_{i+1}^{'}\right) = UpBlock\left(Concat\left(f_i, Upsampling\left(f_{i+1}^{'}\right)\right)\right) \tag{3}$$

where $Concat(\cdot, \cdot)$ and $Upsampling(\cdot)$ are concatenate and upsampling operations, respectively; $UpBlock(\cdot)$ denotes the combination of different modules; in $N_2(\cdot, \cdot)$ and $N_3(\cdot, \cdot)$, the $UpBlock(\cdot)$ consists of three CSPbottleneck modules, a CBAM module, and a Conv layer; and the $UpBlock(\cdot)$ in $N_1(\cdot, \cdot)$ is composed of a transformer module and a CBAM module.

We define the features before the last four Conv. layers as $f_i^{''}$ where $i = 1, \cdots, 4$. Therefore, we obtain Equation (4):

$$f_i^{''} = \begin{cases} f_1^{'}, & i = 1 \\ H_i\left(f_i^{'}, f_{i-1}^{''}\right), & i = 2, 3, 4 \end{cases} \tag{4}$$

where $H_i(\cdot, \cdot)$ represents different blocks, similar to $N_i(\cdot, \cdot)$.

$$H_i\left(f_i^{'}, f_{i-1}^{''}\right) = DownBlock\left(Concat\left(f_i^{'}, Conv\left(f_{i-1}^{''}\right)\right)\right) \tag{5}$$

The $DownBlock(\cdot)$ also denotes the combination of different modules. The $DownBlock(\cdot)$ in $H_2(\cdot, \cdot)$, $H_3(\cdot, \cdot)$, and $H_4(\cdot, \cdot)$ consists of a CBAM module and 1, 2, or 3 transformer modules, respectively. After obtaining the $f_i^{''}$, we can obtain the final predictions as in Equation (6):

$$p_i = Conv\left(f_i^{''}\right), \ i = 1, \cdots, 4 \tag{6}$$

where p_i denotes the four output predictions from different prediction heads.

3.3. TPH-YOLOv5++

Based on the TPH-YOLOv5 model, to improve the efficiency while maintaining detection performance, we propose TPH-YOLOv5++. We will discuss the overall architecture of TPH-YOLOv5++ and the designed cross-layer asymmetric transformer (CA-Trans) module in detail.

3.3.1. Analysis of TPH-YOLOv5

Although TPH-YOLOv5 obtains significant performance improvement on drone-captured images, there are some obvious weaknesses hindering the application of this model.

To analyze the influence of the additional prediction head, we conduct two visualization experiments. In the first experiment shown in Figure 4, we match the prediction boxes produced by each head to the ground truth individually and split prediction results into the correct set and the wrong set. We sort all bounding boxes in the prediction set from largest to smallest according to their confidence values. By setting the IoU threshold as 0.5, for every ground truth box, the bounding box satisfying this requirement with the highest confidence is seen as the correct one, and other boxes are assigned to the wrong set. We process all the bounding boxes predicted in the VisDrone2021 test-dev and draw the distribution of confidence values in Figure 4. We can see that the additional prediction head (the "Tiny Prediction Head" in Figure 4) produces plenty of wrong boxes with relatively large confidence, especially between 0.2 and 0.6. These wrong boxes hamper the detection performance on datasets where high-density scenes are not frequent; for example, the UAVDT [19].

To further analyze the difference in prediction results produced by the additional head and the small prediction head, we visualize the spatial distribution of correct bounding boxes. As shown in Figure 5, we calculate the average confidence value of each pixel. First, for each pixel, we find all the correct bounding boxes that cover this pixel. Second, we average the confidence values of these correct bounding boxes as the visualization value of this pixel. Finally, the visualization results are shown in Figure 5. If the average confidence is high, the color tends towards red, otherwise it tends towards blue. We can obviously find that the additional head indeed improves the performance for high-density scenes and large size variations. However, the small prediction head also captures objects of considerable proportions that are contained by the results predicted by the additional head.

Based on the analysis of the two visualization experiments, we find there is asymmetric information between the additional head and the small prediction head. Therefore, we design the cross-layer asymmetric transformer (CA-Trans) to enrich the feature of small paths with the help of tiny paths. By introducing the CA-Trans, we propose the TPH-

YOLOv5++ significantly reduces the computational cost and improves the detection speed of TPH-YOLOv5.

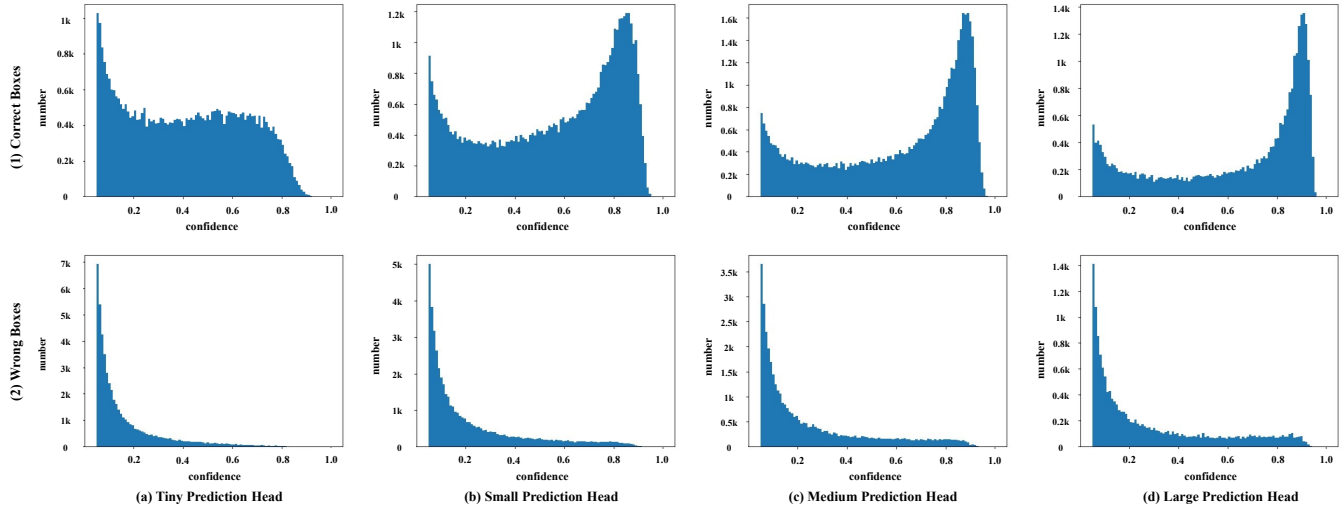
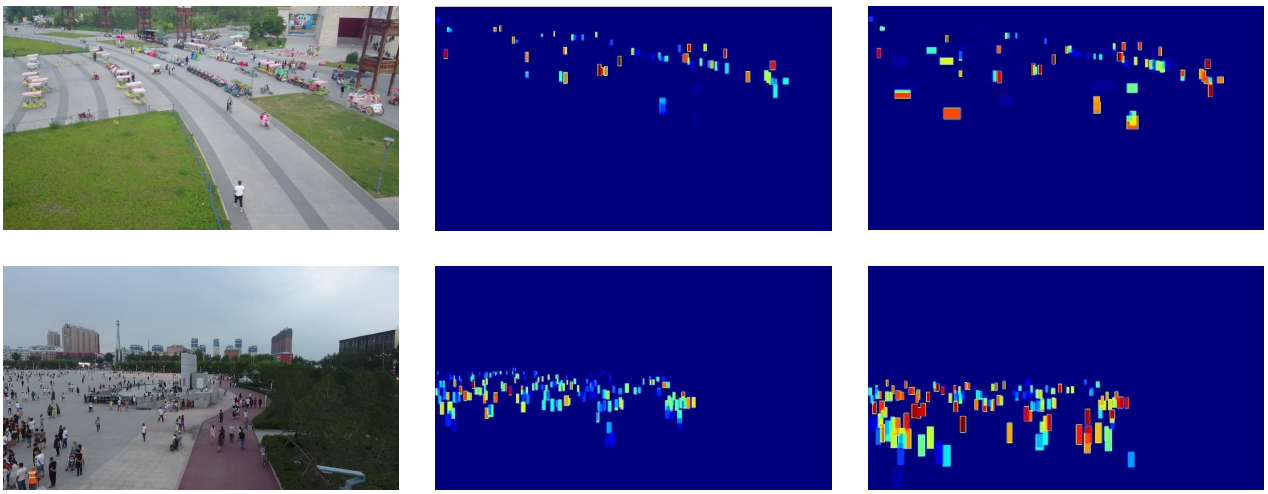


Figure 4. The distributions of confidence of each prediction bounding box generated by four prediction heads. The bounding boxes are split into correct boxes and wrong boxes. The first and second row are the distributions of correct and wrong boxes, respectively.



(a) Original Image **(b) Tiny Prediction Head** **(c) Small Prediction Head**

Figure 5. Two cases of spatial distributions of correct bounding boxes generated by tiny and small heads. The color of each pixel denotes the average confidence of correct boxes covering the pixel. If the confidence is high, the color tends towards red, otherwise it tends towards blue.

3.3.2. Overview of TPH-YOLOv5++

Based on the TPH-YOLOv5, we remove the additional prediction head and introduce CA-Trans between tiny paths and small paths to predict the final $f_2^{''}$, as shown in Figure 6. As discussed earlier, the additional prediction head leads to a significant increase in computation and memory costs. However, the additional prediction head indeed captures more information than the small prediction head, which greatly helps object detection in high-density scenes and for tiny objects. Aiming to transfer the asymmetric information between two heads to the small prediction head, we design CA-Trans, taking f_1 and $f_2^{'}$ as inputs and getting $f_2^{''}$.

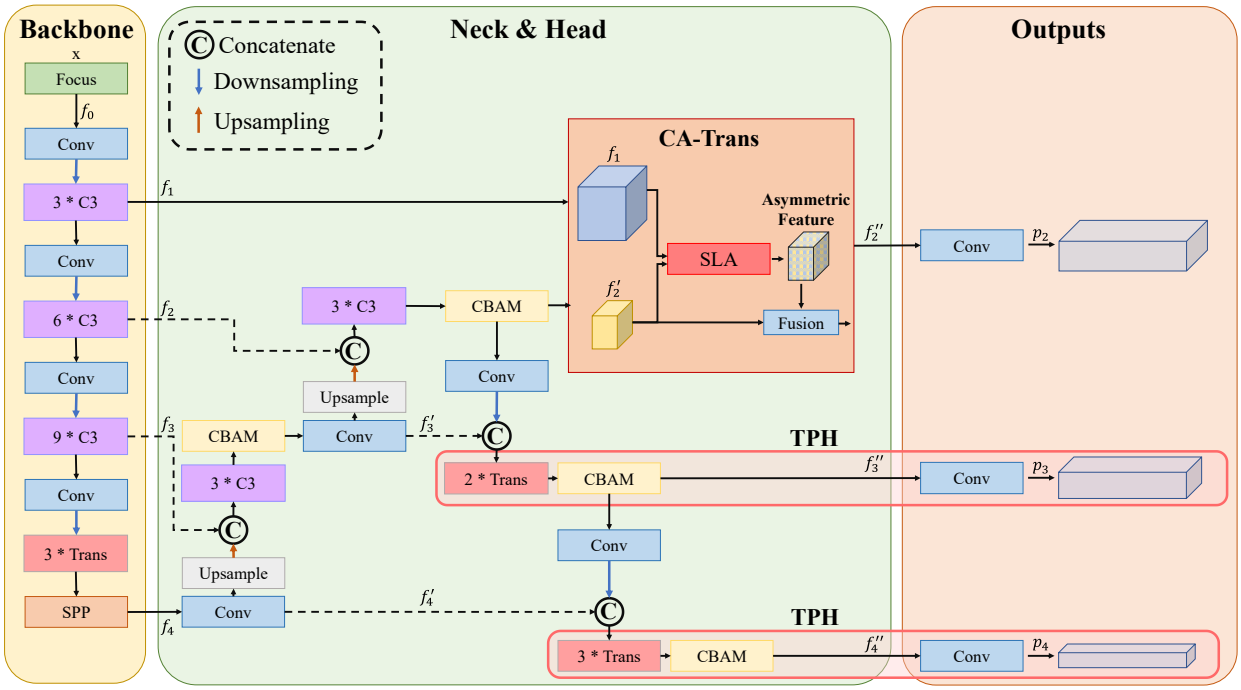


Figure 6. The overall architecture of TPH-YOLOv5++. Different from the TPH-YOLOv5, TPH-YOLOv5++ removes the additional head and introduces the CA-Trans to enrich the features of small paths. The CA-Trans takes f_1 and $f_2^{'}$ as inputs and outputs $f_2^{''}$. After a Conv. layer, a new p_2 is predicted to absorb the best of the original p_1 and p_2.

Compared to TPH-YOLOv5, there are no $f_1^{'}$ or $f_1^{''}$ in TPH-YOLOv5++. Otherwise, the new p_2 is obtained as in Equation (7):

$$p_2 = Conv\left(CA - Trans\left(f_1, f_2^{''}\right)\right) \tag{7}$$

where $CA - Trans(\cdot, \cdot)$ denotes the cross-layer asymmetric transformer (CA-Trans) module.

3.3.3. Cross-Layer Feature Enrichment Transformer

To extract the asymmetric information, which means the information that the additional head can capture but the small prediction head ignores, we design the CA-Trans as shown in Figure 7.

CA-Trans takes f_1 and $f_2^{'}$ as inputs and generates $f_2^{''}$. Then, K and Q are generated from f_1, which can be formulated as $K = LN(Conv(f_1))$ and $V = LN(Conv(f_1))$, where $LN(\cdot)$ denotes the layer norm. Similar to K and V, Q is generated from $f_2^{'}$ by $Q = LN\left(Conv\left(f_2^{'}\right)\right)$. After obtaining Q, K, and V, we use the sparse local attention (SLA) module to calculate sparse attentions between features of two different layers: 1) the feature of tiny paths represented by f_1; and 2) the feature of small paths represented by $f_2^{'}$.

In SLA, we only calculate attentions between pixels in Q and their $a \times b$ neighborhoods in K, as shown in Figures 7c and 8. The a and b must satisfy the restriction condition that $a \geq 2$ and $b \geq 2$. If a and b equal 2, then the neighborhood areas of all pixels in Q can cover the whole space of K just fine without overlapping. However, if a or b is larger than 2, the cover space of all the neighborhood areas will be beyond the space of K, which means we need to pad some zeros in K to make the calculation work well. Defining the spatial size of Q as (w, h), then the spatial size of K is $(2w, 2h)$. If the size of neighborhood area is (a, b), in which $a \geq 2$ and $b \geq 2$, then we need to pad K with zeros with $\left(\left\lfloor \frac{a-2}{2} \right\rfloor, \left\lfloor \frac{b-2}{2} \right\rfloor\right)$ in horizontal and vertical dimensions, respectively, to get the K_p, where $\lfloor \cdot \rfloor$ denotes the rounding down operation. If a equals an odd number, for example, $a = 3$, then we have

$\lfloor \frac{3-2}{2} \rfloor = \lfloor \frac{2-2}{2} \rfloor$. Therefore, for each pixel in Q, the pixels in K_p covered by the neighborhood area under the 3×3 condition are the same as those under the 2×2 condition. Therefore, we only use even number as a and b in experiments, and we set $a = b$ because there is no obvious scale difference between horizontal and vertical image features. The whole working pipeline of SLA can be seen in Algorithm 1.

Algorithm 1: Sparse Local Attention

Input: input feature maps Q, K_p, V
Output: output F_{out}

for *i from 0 to* $a - 1$ **do**
  for *j from 0 to* $b - 1$ **do**
    select the (i, j) point in neighborhood of each pixel in Q;
    build the neighborhood feature M^{ij};
  end
end
concatenate all neighborhood features to generate K_{sparse};
$R \longleftarrow \frac{K_{sparse} \cdot Q}{\sqrt{d_Q}}$;
for *each row r of* R **do**
  calculate the average $\bar{R}_r$;
  $A(r, l) \longleftarrow 2 \cdot \bar{R}_r - R(r, l)$;
end
build the asymmetric map A;
$F_a \longleftarrow softmax(A) \cdot V$;
return F_a;

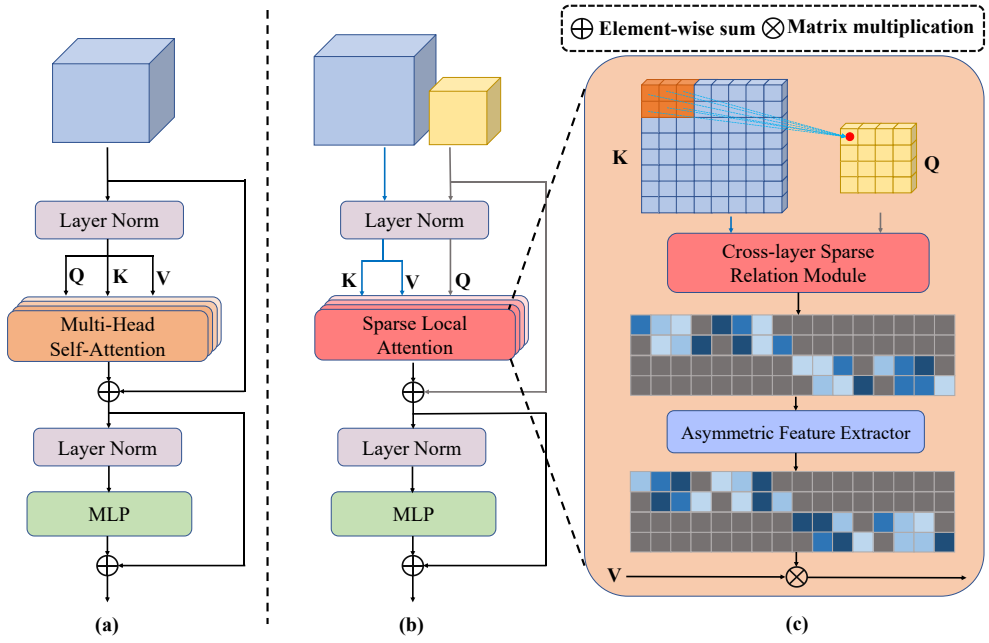


Figure 7. Overview of the CA-Trans module. (**a**) is the vanilla ViT module that generates Q, K, and V from a single feature map and uses multi-head self-attention (MHSA) to obtain attentions. (**b**) shows the architecture of our CA-Trans, where K and V are generated from f_1, while Q is generated from f_2'. Otherwise, we replace the MHSA with the sparse local attention (SLA) to extract attentions between two different layers. (**c**) is the SLA. By introducing the cross-layer sparse relation module (CSRM) and asymmetric feature extractor (AFE), our CA-Trans can efficiently extract asymmetric information between two paths and enrich the features of small paths.

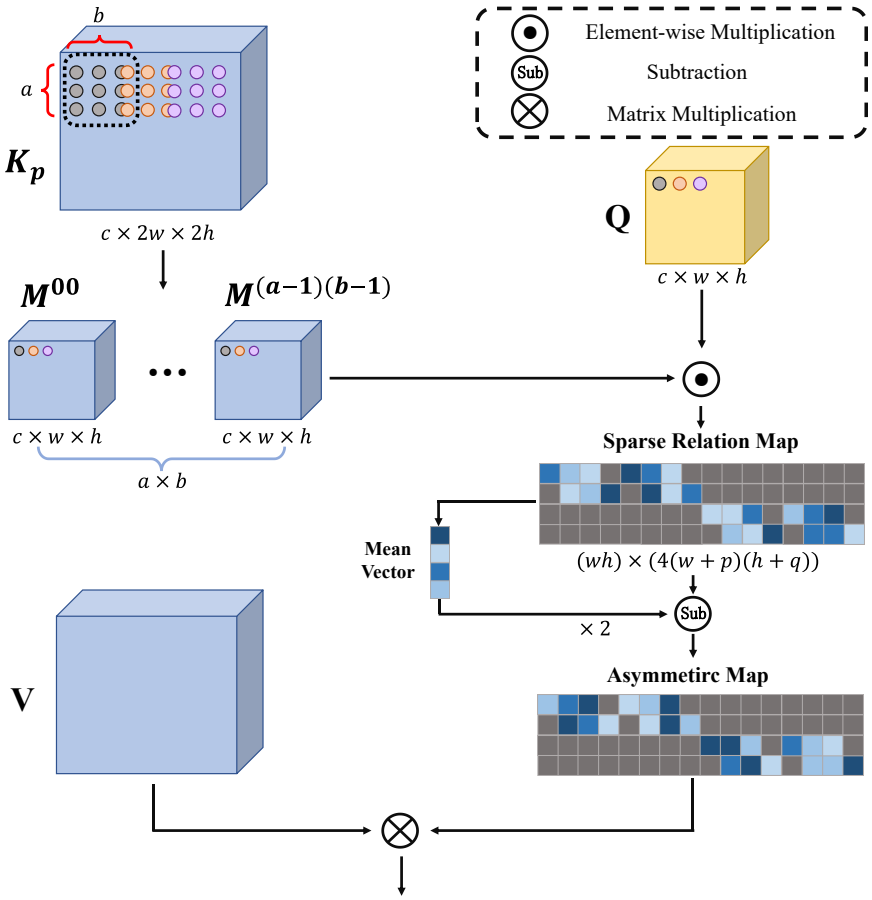


Figure 8. The architecture of SLA with three inputs: K_p, Q, and V. The K_p is generated from K after padding with zeros. After using a sparse relation extraction and an inverse method, the V values are multiplied by the asymmetric map to obtain the final output.

As shown in Figure 8, we can get neighborhood area of K_p for each pixel in Q. Therefore, we have a neighborhood set for Q as formulated in Equation (8):

$$M = \left\{ M^{ij} \mid neighborhood\ feature\ of\ pixel\ (i,j),\ i = 0, \cdots, a-1,\ \ j = 0, \cdots, b-1 \right\} \quad (8)$$

where M is the set of neighborhood features of all pixels in Q, and M^{ij} contains the (i,j) neighborhood feature of each pixel in Q. The transformation of each pixel from K_p to M^{ij} can be calculated by Equation (9):

$$M_{ij}^{(u,v)} = K_p(2u+i, 2v+j) \quad (9)$$

where u and v are the coordinates of pixel in M^{ij}, and they satisfy $u = 0, 1, \cdots, w-1$ and $v = 0, 1, \cdots, h-1$. Then we concatenate all the neighborhood features as in Equation (10):

$$K_{sparse} = Concat\left(M^{00}, M^{01}, \cdots, M^{(a-1)(b-1)}\right) \quad (10)$$

where K_{sparse} is the feature generated by concatenating all elements in M. Then, as shown in Figure 8, we get the sparse relation map by following Equation (11):

$$R = \frac{K_{sparse} \cdot Q}{\sqrt{d_Q}} \quad (11)$$

R denotes the sparse relation map, and d_Q is the channel dimension of Q. The dot between between K_{sparse} and Q denotes the element-wise multiplication. Then we use asymmetric feature extractor (AFE) to obtain the asymmetric map:

$$A = AFE(R) \tag{12}$$

A denotes the asymmetric map, and $AFE(\cdot)$ denotes the AFE module. Specifically, the value at each location of A can be calculated by Equation (13):

$$A(i,j) = 2 \cdot \bar{R}_i - R(i,j) \tag{13}$$

where $\bar{R}_i$ denotes the average of the i-th row of R. Therefore, we invert values in R along each row to find the features that are least relative to features in Q.

We apply the asymmetric map to V to extract the features with the asymmetric information that the original small prediction head ignores.

$$F_a = softmax(A) \cdot V \tag{14}$$

where F_a denotes the asymmetric feature output from the SLA and $softmax(\cdot)$ is the softmax operation along each row of A. Finally, the output F_{out} of our CA-Trans can be formulated as:

$$F_{out} = f_2'' + F_a + MLP\Big(LN\Big(f_2'' + F_a\Big)\Big) \tag{15}$$

4. Experiments

4.1. Implementation Details

4.1.1. Datasets

The VisDrone2021 [21] and UAVDT [19] datasets are used in experiments to evaluate our methods. VisDrone2021 contains four parts: a training set, validation set, test-dev set, and test-challenge set, in which the maximal resolution of images is 2000 × 1500. The training set has 6471 images and their corresponding annotations for training models. The validation set and the test-dev set have 548 and 1610 images, respectively, all of which have corresponding annotations, and these two sets are used to evaluate the performance of models. The test-challenge set has 1580 images and provides no annotations because this set is applied as the test set of the VisDrone Challenge 2021. The UAVDT dataset consists of 50 videos that altogether have 40,376 images, of which 24,778 images are for training and 15,598 images are for testing; all of the images have 1024 × 540 resolution. Following previous works [32,33,49], the training set and testing set are from different videos, and all images in the same video can only be included in one of these two sets. Therefore, the training set consists of images from 31 videos and the testing set contains images of the other 19 videos.

4.1.2. Experimental Setting

We implement our models in Pytorch. All of our models use an NVIDIA RTX3090ti GPU for training and testing. In the training phase, we use part of a pre-trained model from YOLOv5x; because our models and YOLOv5 share most of the backbone and some part of the head, there are many weights that can be transferred from YOLOv5x to TPH-YOLOv5 and TPH-YOLOv5++. By using these weights, we can save a lot of training time.

We train our TPH-YOLOv5 and TPH-YOLOv5++ on VisDrone2021 for 65 epochs, the first 2 of which are used for warm-up. The Adam optimizer is used, and we set the initial learning rate as 3×10^{-4} with the cosine learning rate (cosine lr) schedule. The learning rate of the last epoch decays to 0.12 of the initial learning rate. To avoid missing important information, we set the width and height of input as 1536 due to the large image size. Limited by the GPU memory, the batch size is only 2. On UAVDT, we use 30 epochs and set the batch size as 4. Because the image sizes of UAVDT are smaller than those of VisDrone2021, we use 1024 as the input size.

4.1.3. Evaluation Metrics

We adopt AP, AP50, and AP75 as our evaluation metric. Specifically, AP is computed by averaging over all 10 IoU thresholds, i.e., in the range [0:50:0:95] with uniform step size 0:05 of all categories. AP50 and AP75 are computed at the single IoU thresholds 0:5 and 0:75, respectively. The VisDrone Challenge 2021 also provides AR1, AR10, AR100, and AR500 as evaluation metrics. The AR1, AR10, AR100, and AR500 scores are the maximum recalls given 1, 10, 100, and 500 detections per image.

4.2. Comparison with State-of-the-Art Methods

We conduct experiments to compare our models with current state-of-the-art (SOTA) methods on the test-challenge set, validation set, and test-dev set of the VisDrone2021 dataset and UAVDT test set.

The results in Table 1 are provided by the organizer of VisDrone Challenge 2021 [59]. On the AP metric, which is also the ranking criterion of this challenge, our TPH-YOLOv5 achieves 39.18% and wins 4th place. On the other hand, on AP75, AR1, AR10, AR100, and AR500, we achieve the highest performance, which means TPH-YOLOv5 can accurately detect most objects but produces quite a few false positive boxes at the same time. The bold values in the tables denote the best result on that metric.

Table 1. Results of our TPH-YOLOv5 and other SOTA methods in the VisDrone Challenge 2021 test-challenge dataset, provided by the organizer of this challenge [59].

Method	AP [%]	AP50 [%]	AP75 [%]	AR1 [%]	AR10 [%]	AR100 [%]	AR500 [%]
DBNet(A.1)	**39.43**	**65.34**	41.07	0.29	2.03	12.13	55.36
SOLOer(A.2)	39.42	63.91	40.87	1.75	10.94	44.69	55.91
Swin-T(A.3)	39.40	63.91	40.87	1.76	10.96	44.65	56.83
TPH-YOLOv5(A.4)	39.18	62.83	**41.34**	**2.61**	**13.63**	**45.62**	**56.88**
VistrongerDet(A.5)	38.77	64.28	40.24	0.77	8.10	43.23	55.12
cascade++(A.6)	38.72	62.92	41.05	1.04	6.69	43.36	43.36
DNEFS(A.7)	38.53	62.86	40.19	1.42	9.38	43.10	54.87
EfficientDet(A.8)	38.51	63.25	39.54	1.82	11.12	43.89	55.12
DPNet-ensemble	37.37	62.05	39.10	0.85	7.96	42.03	53.78
DroneEye2020	34.57	58.21	35.74	0.28	1.92	6.93	52.37
Cascade R-CNN	16.09	31.94	15.01	0.28	2.79	21.37	28.43

We also conduct experiments on the VisDrone2021 validation set. As shown in Table 2, our TPH-YOLOv5 achieves 42.1% and 45.7% for AP and AP75, respectively, which are better than previous SOTA methods. Specifically, TPH-YOLOv5 surpasses UFPMP-Net [49] by 2.9% and 5.5% for AP and AP75, and has 2.2% AP50 lower than UFPMP-Net. Compared to TPH-YOLOv5, TPH-YOLOv5++ is about 1% lower on three metrics but still gets higher AP and AP75 than other methods.

Table 2. Results of our methods and current SOTA methods on the VisDrone2021 validation set.

Method	AP [%]	AP50 [%]	AP75 [%]
ClusDet [32]	28.4	53.2	26.4
Zhang et al. [33]	30.3	58.0	27.5
GLSAN [35]	32.5	55.8	33.0
DMNet [36]	29.4	49.3	30.6
DSHNet [37]	30.3	51.8	30.9
HawkNet [41]	25.6	44.3	25.8
CDMNet [42]	31.9	52.9	33.2
DCRFF [60]	35.0	57.0	29.5
UFPMP-Net [49]	39.2	**65.3**	40.2
TPH-YOLOv5	**42.1**	63.1	**45.7**
TPH-YOLOv5++	41.4	61.9	45.0

Table 3 shows the results of different methods on the VisDrone2021 test-dev set. We can see that the Vit-YOLO [61], which also wins 2nd place in the VisDrone Challenge 2021, achieves the SOTA results. Compared to other methods, our TPH-YOLOv5 and TPH-YOLOv5++ have comparable results.

Table 3. Results of our methods and current SOTA methods on the VisDrone2021 test-dev set.

Method	AP [%]	AP50 [%]	AP75 [%]
GDFNet [45]	18.7	31.7	19.4
VistrongerDet [62]	33.85	57.27	34.81
ViT-YOLO [61]	**38.5**	**63.2**	**40.5**
TPH-YOLOv5	34.4	54.5	36.5
TPH-YOLOv5++	33.5	52.5	35.7

The experiments on UAVDT are also conducted, as shown in Table 4. Compared to current methods, our TPH-YOLOv5 achieves new SOTA results on all three metrics, which are 26.9%, 41.3%, and 32.7% for AP, AP50, and AP75, respectively. The TPH-YOLOv5 obtains more than a 2% performance gain at least, and is even 4.7% higher than UFPMP-Net [49] on AP75. Based on TPH-YOLOv5, our TPH-YOLOv5++ further improves the performance to 30.1%, 43.5%, and 34.3% on these three metrics, which mean 3.2%, 2.2%, and 1.5% gains. As we discussed above, for datasets with less high-density scenes like UAVDT, TPH-YOLOv5++ can significantly overcome the shortages of TPH-YOLOv5.

Table 4. Results of our methods and current SOTA methods on UAVDT.

Method	AP [%]	AP50 [%]	AP75 [%]
ClusDet [32]	13.7	26.5	12.5
Zhang et al. [33]	17.7	-	-
GDFNet [45]	15.4	26.1	17.0
GLSAN [35]	19.0	30.5	21.7
DMNet [36]	14.7	24.6	16.3
DSHNet [37]	17.8	30.4	19.7
CDMNet [42]	20.7	35.5	22.4
SODNet [48]	17.1	29.9	18.0
UFPMP-Net [49]	24.6	38.7	28.0
TPH-YOLOv5	26.9	41.3	32.7
TPH-YOLOv5++	**30.1**	**43.5**	**34.3**

4.3. Ablation Studies

4.3.1. Ablation Study on VisDrone2021 Test-Dev Set

We analyze the importance of each proposed component on a local test-dev set as we cannot test them on the VisDrone2021 competition server. As shown in Table 5, we evaluate six different models from multiple perspectives. For analyzing the detection performance, we use AP, AP50, and AP75 as metrics. Additionally, GPU memory cost during testing phase, giga floating point of operations (GFLOPs), and frames per second (FPS) are utilized to evaluate the computation cost and operation efficiency of these models.

The additional prediction head significantly increases the detection performance. Compared to YOLOv5x, it improves the AP, AP50, and AP75 by 2.1%, 3.3%, and 2.1%, respectively. However, it also introduces non-negligible memory and computation costs. Specifically, the GPU memory cost increases from 4279 M to 4667 M, and GFLOP increases from 200.2 to 241.2. The FPS of 'YOLOv5x+p2' is 10.89, which is lower than the 13.68 of YOLOv5x. Similar to the additional prediction head, the transformer modules (both ViT and Swin-Transformer) and the CBAM lead to higher memory and computation costs when further improving the detection performance. The 'TPH-YOLOv5 (SwinTrans+CBAM)'

achieves 34.0%, 53.2%, and 35.8% on AP, AP50, and AP75, but its GPU memory cost, GFLOPs, and FPS are 4977M, 315.4, and 7.36, which are inferior to these of YOLOv5x.

Table 5. Ablation study of the importance of each proposed component on the VisDrone2021 test-dev set. 'p2' denotes the additional prediction head. 'ViT' denotes the ViT module. 'previous' denotes the 'YOLOv5x+p2+ViT', and 'TPH-YOLOv5 (SwinTrans+CBAM)' denotes the model replacing all ViT modules with Swin-Transfomer.

Methods	AP [%]	AP50 [%]	AP75 [%]	GPU Memory	GFLOPs	FPS
YOLOv5x	28.9	45.4	30.8	4279 M	200.2	13.68
YOLOv5x+p2	31.0	48.7	32.9	4667 M	241.2	10.89
YOLOv5x+p2+ViT	32.8	52.0	34.8	5103 M	244.5	9.51
TPH-YOLOv5 (previous+CBAM)	33.6	53.2	35.8	5105 M	245.1	8.22
TPH-YOLOv5 (SwinTrans+CBAM)	34.0	53.2	35.8	4977 M	315.4	7.36
TPH-YOLOv5++	33.1	52.1	35.1	4715 M	207.0	11.86

Based on the TPH-YOLOv5, we propose the TPH-YOLOv5++. As shown in Table 5, TPH-YOLOv5++ decreases 0.9%, 1.1%, and 0.7% on three detection performance metrics. In addition, the GFLOP of TPH-YOLOv5++ is 207.0, which is substantially lower than that of TPH-YOLOv5 and slightly higher than YOLOv5x. Meanwhile, for the FPS metric, TPH-YOLOv5++ improves 4.50 compared to TPH-YOLOv5.

4.3.2. Ablation Study on UAVDT

On the UAVDT dataset, we also evaluate the detection performance and efficiency of TPH-YOLOv5 and TPH-YOLOv5++. As shown in Table 6, different from the results in the VisDrone2021 test-dev set, TPH-YOLOv5++ is better than TPH-YOLOv5 on detection performance. For AP50 and AP75, TPH-YOLOv5++ gains 2.2% and 1.6% respectively. For AP, TPH-YOLOv5++ even improves the performance from 26.9% to 30.1%, which is also the SOTA result. Additionally, TPH-YOLOv5++ decreases GPU memory cost from 3631 M to 3361 M. The GFLOP of TPH-YOLOv5++ is 293.2, which is 47.3% less than that of TPH-YOLOv5. For FPS, TPH-YOLOv5++ achieves 42.19, evenly 68.0% higher than that of TPH-YOLOv5.

Table 6. Ablation study of TPH-YOLOv5 and TPH-YOLOv5++ on UAVDT.

Methods	AP [%]	AP50 [%]	AP75 [%]	GPU Memory	GFLOPs	FPS
TPH-YOLOv5	26.9	41.3	32.7	3631 M	556.6	25.12
TPH-YOLOv5++	30.1	43.5	34.3	3361 M	293.2	42.19

4.3.3. Ablation Study on Each Category

To further analyze the performance of different models on all categories, we calculate the AP of each category as shown in Table 7. "TPH-YOLOv5+ms-testing" means utilizing multi-scale testing on TPH-YOLOv5. In the VisDrone2021 images, 'pedestrian' means humans walking or standing while 'people' means humans sitting, lying, or driving cars, so pedestrians are more likely to appear in high-density scenes. The additional prediction head improves the AP of pedestrians by 2.1%, but the AP of people only increases 0.6%. This result is indicative of the better performance of additional detection heads for high-density scenes.

By adding the ViT modules, we can see that AP of each category has a significant improvement. For people, the AP also increases from 14.9% to 16.0%. Therefore, the introduction of transformer can utilize more context information and explore the prediction potential.

Table 7. Ablation study for each category on the VisDrone2021 test-dev set. The AP of each category is used as the result.

Methods	All	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	Awning-Tricycle	Bus	Motor
YOLOv5x	28.9	23.5	14.3	13.5	51.8	35.4	38.0	20.2	19.9	48.6	23.8
YOLOv5x+p2	31.0	25.6	14.9	14.3	56.2	37.4	40.1	22.0	21.5	52.5	25.3
YOLOv5x+p2+ViT	32.8	26.7	16.0	15.5	59.1	40.0	42.7	23.4	22.2	55.4	27.1
TPH-YOLOv5 (previous+CBAM)	33.6	27.4	16.3	15.9	61.4	41.9	43.3	23.9	21.5	56.9	27.8
TPH-YOLOv5 (SwinTrans+CBAM)	34.0	27.5	16.1	15.9	61.7	41.9	43.9	24.2	24.0	56.4	28.5
TPH-YOLOv5+ms-testing	34.9	28.8	16.3	15.0	65.9	44.3	43.8	25.7	22.8	59.0	27.1

4.3.4. Ablation Study of Neighborhood Size

In TPH-YOLOv5++, the SLA plays a very important role by transferring information from tiny paths to small paths. In SLA, the neighborhood size (a, b) affects how large the area in the feature map of tiny paths is needed by each pixel on the small path. To analyze the best neighborhood size, we conduct experiments on the VisDrone2021 test-dev set. Because there is no obvious difference between information along the horizontal and vertical axes in an image, we set $a = b$ in experiments. Additionally, $a = 0, b = 0$ means only remove the additional prediction head without adding SLA.

As shown in Table 8, when $a = 0, b = 0$, the AP is only 31.9%, which is substantially lower than TPH-YOLOv5. By adding the SLA with the smallest neighborhood size $(2, 2)$, TPH-YOLOv5++ increases the AP to 33.1%, higher than the TPH-YOLOv5 without CBAM. Meanwhile, TPH-YOLOv5++ is also efficient, with 207 GFLOPs and 11.86 FPS. During enlargement of the neighborhood size, the detection performance can obtain a slight increase. However, the efficiency and memory cost of TPH-YOLOv5++ becomes worse at the same time. The GPU memory costs are 7475 M and 12185 M if the neighborhood sizes are $(4, 4)$ and $(6, 6)$, respectively. At the same time, the FPS also decreases significantly when the neighborhood size increases. For the $(4, 4)$ and $(6, 6)$, TPH-YOLOv5++ decreases the FPS to 10.14% and 7.67%, respectively. When the neighborhood size is $(8, 8)$, the GPU memory cost is very high, so we do not continually enlarge the neighborhood size. In conclusion, we choose $(2, 2)$ as the best neighborhood size in other experiments.

Table 8. Ablation study for neighborhood size (a, b) of SLA on the VisDrone 2021 test-dev dataset.

	Neighborhood Size (a, b)			
	$a = 0, b = 0$	$a = 2, b = 2$	$a = 4, b = 4$	$a = 6, b = 6$
AP [%]	31.9	33.1	33.5	33.6
AP50 [%]	51.7	52.1	52.5	52.6
AP75 [%]	33.5	35.1	34.9	35.0
GPU Memory	4299 M	4715 M	7475 M	12,185 M
GFLOPs	204.5	207	214.9	228.1
FPS	12.01	11.86	10.14	7.67

4.4. Visualization Analysis

4.4.1. Qualitative Visualization of Detection Results

To analyze the performance of our TPH-YOLOv5++ intuitively, we visualize the final prediction results in Figures 9 and 10. The results show that TPH-YOLOv5++ can accurately detect tiny objects, dense objects, and objects blurred by motion.

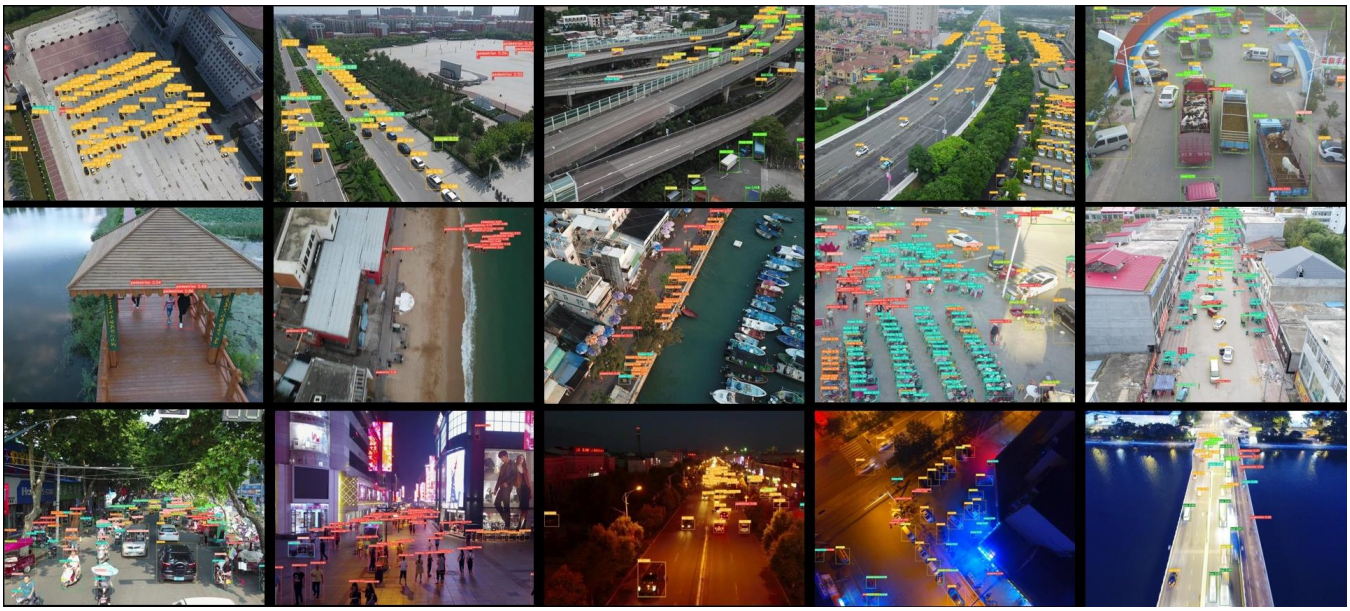


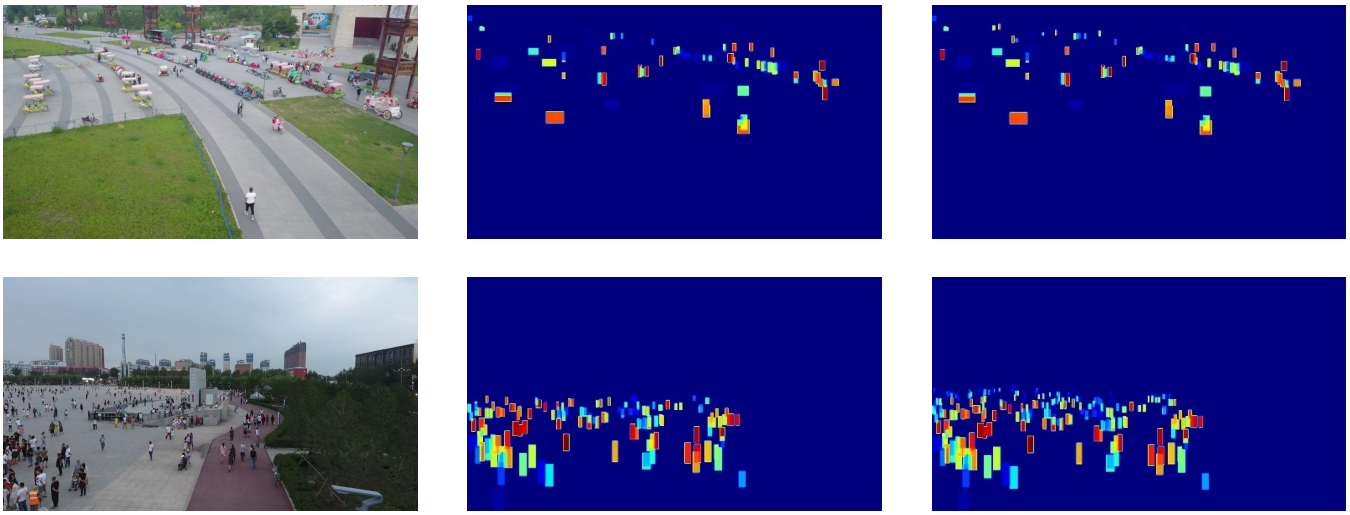
Figure 9. Qualitative results of the proposed TPH-YOLOv5++ on the VisDrone2021 test-challenge set. Different categories use bounding boxes with different colors. The performance is good at localizing tiny objects, dense objects, and objects blurred by motion.

Figure 10. Qualitative results of the proposed TPH-YOLOv5++ on the UAVDT dataset.

4.4.2. Visualization of Correct Bounding Boxes

We visualize the spatial distribution of correct bounding boxes generated by the small prediction heads of TPH-YOLOv5 and TPH-YOLOv5++, as shown in Figure 11. Similar to Figure 5, the color of each pixel denotes the average confidence of correct boxes covering the pixel. If the confidence is high, the color tends towards red, otherwise it tends towards blue.

The second column of Figure 11 represents the results of TPH-YOLOv5, and the third column represents the results of TPH-YOLOv5++. We can obviously find that TPH-YOLOv5++ can detect many tiny objects that are ignored by TPH-YOLOv5. In addition, this difference is more pronounced in high-density areas.



(a) Original Image **(b) TPH-YOLOv5** **(c) TPH-YOLOv5++**

Figure 11. Spatial distributions of correct bounding boxes generated by the small prediction heads of TPH-YOLOv5 and TPH-YOLOv5++.

Although TPH-YOLOv5++ can significantly improve the detection performance of the small prediction head with the help of the tiny head, the bounding boxes that can be re-detected have relatively low confidence, as shown in Figure 11. Due to this phenomenon, the improvement that TPH-YOLOv5++ achieves has an upper limit, and there is also a drop in some small object density scenes.

In conclusion, TPH-YOLOv5++ can effectively transfer information from tiny paths to enrich features of small paths. By introducing SLA, the detection performance of the small prediction head gets a great promotion, detecting tiny objects, especially in high-density scenes, more accurately. Otherwise, the performance on the neglected objects of the small head is weaker than the original tiny head, so it will have a performance decrease in small object density scenes.

4.4.3. Visualization between TPH-YOLOv5 and TPH-YOLOv5++

As discussed above, TPH-YOLOv5++ achieves better results than TPH-YOLOv5 on the UAVDT dataset. To intuitively analyze the performance gap between two models, we visualize the prediction results of two models on UAVDT, as shown in Figure 12.

There are three cases displayed in Figure 12: on the left of the black dotted line are the results of TPH-YOLOv5, while on the right are the results of TPH-YOLOv5++. Meanwhile, each area that can highlight the performance differences between the two models is zoomed in on the right side of the corresponding image.

For example, on the first row, the TPH-YOLOv5 detects the bus as multiple objects, while TPH-YOLOv5 can significantly decrease these wrong bounding boxes. On the second row, TPH-YOLOv5 detects a bounding box across the top two cars. By contrast, TPH-YOLOv5++ correctly predicts two cars.

Based on the above analysis, the additional prediction head of TPH-YOLOv5 indeed introduces many false positive boxes in dense scenes. With the help of the removal of the additional head and the SLA, TPH-YOLOv5++ can significantly avoid these wrong bounding boxes and improve the efficiency.

(a) TPH-YOLOv5 **(b) TPH-YOLOv5++**

Figure 12. Visualization of TPH-YOLOv5 and TPH-YOLOv5++ on the UAVDT dataset.

5. Conclusions

Object detection on drone-captured images has three main challenges: size variation, high-density, and large coverage of objects. Based on the YOLOv5, we add some cutting-edge techniques, i.e., transformer encoder block, CBAM, and some experienced tricks to improve the detection performance in drone-captured scenarios. Then, to alleviate the computational and inference time costs while maintaining performance, we design a novel cross-layer asymmetric transformer module, constructing the TPH-YOLOv5++ model. By replacing the original multi-head self-attention in vision transformer with sparse local attention, the cross-layer asymmetric transformer module can enrich the feature of small paths with the help of tiny paths. Our TPH-YOLOv5 won 4th place in the VisDrone challenge 2021. Extensive experiments are conducted on two benchmark datasets, which show that our two models achieve the new SOTA results and that TPH-YOLOv5++ can significantly reduce computation and memory costs while achieving comparable or better performance than TPH-YOLOv5.

Author Contributions: Conceptualization, Q.Z. and B.L.; methodology, Q.Z., B.L. and S.L.; validation, Q.Z. and B.L.; formal analysis, Q.Z. and B.L.; writing—original draft preparation, Q.Z. and B.L.; writing—review and editing, H.Z. and C.W.; visualization, Q.Z. and B.L.; supervision, C.W.; funding acquisition, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62072021.

Data Availability Statement: Not applicable.

Acknowledgments: We sincerely thank the authors of YOLOv5 and ViT for providing their algorithm codes to facilitate the comparative experiments.

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

References

1. Audebert, N.; Le Saux, B.; Lefèvre, S. Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. *ISPRS J. Photogramm. Remote. Sens.* **2018**, *140*, 20–32. [CrossRef]
2. Gu, J.; Su, T.; Wang, Q.; Du, X.; Guizani, M. Multiple moving targets surveillance based on a cooperative network for multi-UAV. *IEEE Commun. Mag.* **2018**, *56*, 82–89. [CrossRef]
3. Hird, J.N.; Montaghi, A.; McDermid, G.J.; Kariyeva, J.; Moorman, B.J.; Nielsen, S.E.; McIntosh, A.C. Use of unmanned aerial vehicles for monitoring recovery of forest vegetation on petroleum well sites. *Remote Sens.* **2017**, *9*, 413. [CrossRef]
4. Kellenberger, B.; Marcos, D.; Tuia, D. Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sens. Environ.* **2018**, *216*, 139–153. [CrossRef]
5. 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 European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
6. Everingham, M.; Eslami, S.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the NIPS 2015, Advances in Neural Information Processing Systems 28, Montreal, QC, Canada, 7–12 December 2015.
8. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
9. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
10. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
11. Zhang, H.; Wang, Y.; Dayoub, F.; Sunderhauf, N. Varifocalnet: An iou-aware dense object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8514–8523.
12. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
13. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
14. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. Jocher, G.; Stoken, A.; Borovec, J.; Chaurasia, A.; Changyu, L.; Laughing, A.; Hogan, A.; Hajek, J.; Diaconu, L.; Marc, Y.; et al. ultralytics/yolov5: V5. 0-YOLOv5-P6 1280 models AWS Supervise. ly and YouTube integrations. *Zenodo* **2021**, *11*.
16. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the NIPS 2017, Advances in Neural Information Processing Systems 30, Long Beach, CA, USA, 4–9 December 2017.
17. Alexey, D.; Lucas, B.; Alexander, K.; Dirk, W.; Xiaohua, Z.; Thomas, U.; Mostafa, D.; Matthias, M.; Georg, H.; Sylvain, G.; et al. An imageis worth 16x16 words: Transformers for image recognition at scale. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Vienna, Austria, 3–7 May 2021.
18. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
19. Du, D.; Qi, Y.; Yu, H.; Yang, Y.; Duan, K.; Li, G.; Zhang, W.; Huang, Q.; Tian, Q. The unmanned aerial vehicle benchmark: Object detection and tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 370–386.
20. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2778–2788.
21. 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]
22. Cai, Z.; Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6154–6162.
23. Law, H.; Deng, J. Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.
24. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. Centernet: Keypoint triplets for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 6569–6578.
25. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
26. Yang, Z.; Liu, S.; Hu, H.; Wang, L.; Lin, S. Reppoints: Point set representation for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 9657–9666.
27. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
28. Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; Fu, Y. Rethinking classification and localization for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10186–10195.

29. Chen, Q.; Wang, Y.; Yang, T.; Zhang, X.; Cheng, J.; Sun, J. You only look one-level feature. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13039–13048.
30. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Soft-NMS–improving object detection with one line of code. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5561–5569.
31. Solovyev, R.; Wang, W.; Gabruseva, T. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image Vis. Comput.* **2021**, *107*, 104117. [CrossRef]
32. Yang, F.; Fan, H.; Chu, P.; Blasch, E.; Ling, H. Clustered object detection in aerial images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 8311–8320.
33. Zhang, J.; Huang, J.; Chen, X.; Zhang, D. How to fully exploit the abilities of aerial image detectors. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27–28 October 2019.
34. Zhang, R.; Newsam, S.; Shao, Z.; Huang, X.; Wang, J.; Li, D. Multi-scale adversarial network for vehicle detection in UAV imagery. *ISPRS J. Photogramm. Remote Sens.* **2021**, *180*, 283–295. [CrossRef]
35. Deng, S.; Li, S.; Xie, K.; Song, W.; Liao, X.; Hao, A.; Qin, H. A global-local self-adaptive network for drone-view object detection. *IEEE Trans. Image Process.* **2020**, *30*, 1556–1569. [CrossRef] [PubMed]
36. Li, C.; Yang, T.; Zhu, S.; Chen, C.; Guan, S. Density map guided object detection in aerial images. In Proceedings of the proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 13–19 June 2020; pp. 190–191.
37. Yu, W.; Yang, T.; Chen, C. Towards resolving the challenge of long-tail distribution in UAV images for object detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Virtual, 5–9 January 2021; pp. 3258–3267.
38. Chen, J.; Hong, H.; Song, B.; Guo, J.; Chen, C.; Xu, J. MDCT: Multi-Kernel Dilated Convolution and Transformer for One-Stage Object Detection of Remote Sensing Images. *Remote Sens.* **2023**, *15*, 371. [CrossRef]
39. Gallo, I.; Rehman, A.U.; Dehkordi, R.H.; Landro, N.; La Grassa, R.; Boschetti, M. Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images. *Remote Sens.* **2023**, *15*, 539. [CrossRef]
40. Liu, R.; Tao, F.; Liu, X.; Na, J.; Leng, H.; Wu, J.; Zhou, T. RAANet: A Residual ASPP with Attention Framework for Semantic Segmentation of High-Resolution Remote Sensing Images. *Remote Sens.* **2022**, *14*, 3109. [CrossRef]
41. Lin, H.; Zhou, J.; Gan, Y.; Vong, C.M.; Liu, Q. Novel up-scale feature aggregation for object detection in aerial images. *Neurocomputing* **2020**, *411*, 364–374. [CrossRef]
42. Duan, C.; Wei, Z.; Zhang, C.; Qu, S.; Wang, H. Coarse-grained Density Map Guided Object Detection in Aerial Images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2789–2798.
43. Xi, Y.; Jia, W.; Miao, Q.; Liu, X.; Fan, X.; Li, H. FiFoNet: Fine-Grained Target Focusing Network for Object Detection in UAV Images. *Remote Sens.* **2022**, *14*, 3919. [CrossRef]
44. Ringwald, T.; Sommer, L.; Schumann, A.; Beyerer, J.; Stiefelhagen, R. UAV-Net: A fast aerial vehicle detector for mobile platforms. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–17 June 2019.
45. Zhang, R.; Shao, Z.; Huang, X.; Wang, J.; Li, D. Object detection in UAV images via global density fused convolutional network. *Remote Sens.* **2020**, *12*, 3140. [CrossRef]
46. Chen, P.Y.; Hsieh, J.W.; Wang, C.Y.; Liao, H.Y.M. Recursive hybrid fusion pyramid network for real-time small object detection on embedded devices. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 402–403.
47. Cao, J.; Pang, Y.; Han, J.; Li, X. Hierarchical Regression and Classification for Accurate Object Detection. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–15. [CrossRef]
48. Qi, G.; Zhang, Y.; Wang, K.; Mazur, N.; Liu, Y.; Malaviya, D. Small Object Detection Method Based on Adaptive Spatial Parallel Convolution and Fast Multi-Scale Fusion. *Remote Sens.* **2022**, *14*, 420. [CrossRef]
49. Huang, Y.; Chen, J.; Huang, D. Ufpmp-det: Toward accurate and efficient object detection on drone imagery. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2022; Volume 36, pp. 1026–1033.
50. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10012–10022.
51. Dong, X.; Bao, J.; Chen, D.; Zhang, W.; Yu, N.; Yuan, L.; Chen, D.; Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12124–12134.
52. Chen, C.F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 357–366.
53. Xu, R.; Xiang, H.; Tu, Z.; Xia, X.; Yang, M.H.; Ma, J. V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 107–124.
54. Xu, R.; Tu, Z.; Xiang, H.; Shao, W.; Zhou, B.; Ma, J. CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers. *arXiv* **2022**, arXiv:2207.02202.

55. Tu, Z.; Talebi, H.; Zhang, H.; Yang, F.; Milanfar, P.; Bovik, A.; Li, Y. Maxvit: Multi-axis vision transformer. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 459–479.
56. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
57. Wang, Y.; Zhang, X.; Yang, T.; Sun, J. Anchor DETR: Query Design for Transformer-Based Object Detection. *arXiv* **2021**, arXiv:2109.07107.
58. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You only look at one sequence: Rethinking transformer in vision through object detection. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26183–26197.
59. Cao, Y.; He, Z.; Wang, L.; Wang, W.; Yuan, Y.; Zhang, D.; Zhang, J.; Zhu, P.; Van Gool, L.; Han, J.; et al. VisDrone-DET2021: The vision meets drone object detection challenge results. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2847–2854.
60. Mittal, P.; Sharma, A.; Singh, R.; Dhull, V. Dilated convolution based RCNN using feature fusion for Low-Altitude aerial objects. *Expert Syst. Appl.* **2022**, *199*, 117106. [CrossRef]
61. Zhang, Z.; Lu, X.; Cao, G.; Yang, Y.; Jiao, L.; Liu, F. ViT-YOLO: Transformer-based YOLO for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2799–2808.
62. Wan, J.; Zhang, B.; Zhao, Y.; Du, Y.; Tong, Z. VistrongerDet: Stronger Visual Information for Object Detection in VisDrone Images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2820–2829.