drones

MDPI

Article

Small-Object Detection for UAV-Based Images Using a Distance Metric Method

Helu Zhou [1,2,†], Aitong Ma [1,†], Yifeng Niu [1] and Zhaowei Ma [1,*]

[1] College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410078, China
[2] Aerocraft Technology Branch of Hunan Aerospace Co., Ltd., Changsha 410200, China
* Correspondence: mazhaowei1989@126.com
† These authors contributed equally to this work.

Abstract: Object detection is important in unmanned aerial vehicle (UAV) reconnaissance missions. However, since a UAV flies at a high altitude to gain a large reconnaissance view, the captured objects often have small pixel sizes and their categories have high uncertainty. Given the limited computing capability on UAVs, large detectors based on convolutional neural networks (CNNs) have difficulty obtaining real-time detection performance. To address these problems, we designed a small-object detector for UAV-based images in this paper. We modified the backbone of YOLOv4 according to the characteristics of small-object detection. We improved the performance of small-object positioning by modifying the positioning loss function. Using the distance metric method, the proposed detector can classify trained and untrained objects through object features. Furthermore, we designed two data augmentation strategies to enhance the diversity of the training set. We evaluated our method on a collected small-object dataset; the proposed method obtained 61.00% mAP_{50} on trained objects and 41.00% mAP_{50} on untrained objects with 77 frames per second (FPS). Flight experiments confirmed the utility of our approach on small UAVs, with satisfying detection performance and real-time inference speed.

Keywords: small-object detection; backbone design; object positioning; object classification; UAV flight experiment

Citation: Zhou, H.; Ma, A.; Niu, Y.; Ma, Z. Small-Object Detection for UAV-Based Images Using a Distance Metric Method. *Drones* **2022**, *6*, 308. https://doi.org/10.3390/drones6100308

Academic Editor: Diego González-Aguilera

Received: 11 September 2022
Accepted: 17 October 2022
Published: 20 October 2022

1. Introduction

Nowadays, Unmanned aerial vehicles (UAVs) play an important role in civil and military fields, such as system mapping [1], low-attitude remote sensing [2], collaborative reconnaissance [3], and others. In many applications, reconnaissance tasks are mostly based on UAV airborne vision. In this case, the detection and recognition of ground targets is an important demand. However, when the UAV flies at high altitudes, the captured object occupies a relatively small pixel scale in UAV airborne images. It is a challenge to detect such small objects in complex large scenes. Additionally, due to the limited computing resources in UAVs, many large-scale detection models based on server and cloud computing are not suitable for online real-time detection of small unmanned aerial vehicles. In this case, achieving fast and accurate small-object detection using the onboard computer becomes challenging. This paper mainly focuses on the detection of small objects in UAV reconnaissance images.

Combining with the flight characteristics of small UAVs and the computing capability of onboard processors, this paper selects a neural-network-based model as the basic detection model. To the best of our knowledge, most of the current detection algorithms for UAVs use one-stage detectors [4]. One of the state-of-the-art detectors among the one-stage detectors is YOLOv4 [5]. The YOLOv4 object detector integrates various classic ideas [6–9] in the field of object detection and works at a faster speed and higher accuracy than other alternative detectors. We choose YOLOv4 as the benchmark detector. The YOLOv4 detector

was proposed and trained using a common dataset [10] covering various objects. However, the objects in the field of UAV reconnaissance that we are concerned with are limited in category, such as cars, aircraft, and ships. There are many instances of subdividing these limited categories of targets, but current object detection training sets rarely care about all types of objects. Therefore, objects that have not appeared in the training set are difficult to recognize in the UAV reconnaissance image during the inference stage, which is also a major challenge for UAV object detection. Generally, these objects are small in the vision of UAVs. When the flight altitude of the UAV is different, the image pixels of the same object are also different. Since YOLOv4 is a multi-scale detector, we improve YOLOv4 to make it more suitable for small-object detection in UAV reconnaissance images. Furthermore, the images of the same scene obtained by the UAV are different under different flight weather conditions.

To solve the above challenges, we proposed a small-object detection method that is applied to UAV reconnaissance. Our contributions are described as follows:

1. We propose two data augmentation methods to improve the generalization of the algorithm on the scene;

2. We design a backbone network that is suitable for small-object detection and modify the positioning loss function of the one-stage detector to improve detection accuracy;

3. We design a metric-based object classification method to classify objects into subclasses and detect objects that do not appear in the training phase, in other words, untrained objects.

The remainder of this manuscript is structured as follows. Section 2 introduces some related works of object detection algorithms. Section 3 formulates the detector structure for UAV untrained small-object detection and introduces the improved algorithm. Experimental results are presented in Section 4 to validate the effectiveness of our proposed method. Section 5 concludes this paper and envisages some future work.

2. Related Works

2.1. Small-Object Detection Algorithm

Most of the state-of-the-art detectors are based on deep-learning methods [11]. These methods mainly include two-stage detectors, one-stage detectors, and anchor-free detectors. Two-stage detectors first extract possible object locations and then perform classification and relocation. The classic two-stage detectors include spatial pyramid pooling networks (SPPNet) [9], faster region-CNN (RCNN) [12], etc. The one-stage detectors perform classification and positioning at the same time. Some effective one-stage detectors mainly include the single shot multi-box detector (SSD) [13], You Only Look Once (YOLO) series [5,8,14,15], etc. Anchor-free detectors include CenterNet [16], ExtremeNet [17], etc. These methods do not rely on predefined anchors to detect objects. In addition, some scholars have introduced transformers into the object detection field, such as detection with transformers (DETR) [18] and vision transformer faster RCNN (ViT FRCNN) [19], which have also achieved good results. However, the detection objects of the general object detectors are multi-scale. They are not designed for small-object detection specifically.

Small-object detection algorithms can be mainly divided into two kinds. One is to improve the detection performance of small objects with multiple scales in a video or image sequence. The other is to improve the detection performance of small objects with only a scale in an image. The improved detection methods of small objects with multiple scales mainly include feature pyramids, data augmentation, and changing training strategies. In 2017, Lin et al. proposed feature pyramid networks (FPN) [20], which improves the detection performance effect of small objects by fusing high-level and low-level features to generate multi-scale feature layers. In 2019, M. Kisantal et al. proposed two data augmentation methods [21] for small objects to increase the frequency of small objects in training images. B. Singh et al. designed scale normalization for image pyramids (SNIP) [22]. SNIP selectively backpropagates the gradients of objects of different sizes, and trains and tests images of different resolutions, respectively. The research object of

these methods is multi-scale objects, which cannot make full use of the characteristics of small objects. The approaches that only detect small objects with a scale are mainly of three kinds: designing networks, using context information, and generating super-resolution images. L. Sommer et al. proposed a very shallow network for detecting objects in aerial images [23]. Small-object detection based on network design is few and immature. J. Li et al. proposed a new perceptual GAN network [24] to improve the resolution of small objects to improve the detection performance. In this paper, we focus on algorithms that are suitable for small-object detection in UAV reconnaissance images.

2.2. Object Detection Algorithm for UAV

The object detection algorithms used in UAVs are mainly designed based on the requirements of the task scenarios. M. Liu et al. proposed an improved detection algorithm based on YOLOv3 [8]. The algorithm first optimizes the Res-Block and then improves the darknet structure by increasing the convolution operation. Y. Li et al. proposed a multi-block SSD (MBSSD) mechanism [25] for railway scene monitored by UAVs. MBSSD uses transfer learning to solve the problem of insufficient training samples and improve accuracy. Y. Liu et al. proposed multi-branch parallel feature pyramid networks (MPFPN) [26] and used a supervised spatial attention module (SSAM) to focus the model on object information. MPFPN conducted experiments on a UAV public dataset named VisDrone-DET [27] to prove its effectiveness. These algorithms are combined with UAV application scenarios, but are all based on classic methods. They cannot work well when inferencing against untrained small objects.

3. Proposed Method

In this paper, we focus on small objects in images from the aerial view of UAVs. The proportion of object pixels in the image is less than 0.1% and the objects to be detected include objects that have not appeared in the training set. Current deep-learning-based object detection algorithms depend on a large amount of data. Therefore, we design an approach to expand the dataset in the proposed detection framework. In addition to the classic methods such as rotation, cropping, and color conversion, we propose two data augmentation methods—background replacement and noise adding—to improve the generalization of the detection algorithm. Background replacement gives the training images more negative samples to make the training set have a richer background. Noise adding can prevent the training model from overfitting the object.

After preprocessing the training images, the image features need to be obtained through the backbone network. We choose YOLOv4 [5], which has a good trade-off between speed and accuracy, as the benchmark algorithm for research. Common backbones in object detection algorithms are used to detect multi-scale objects. The receptive field of the backbone module is extremely large. In YOLOv4, the input of the backbone called CSPDarknet53 [7] is 725×725. However, the number of pixels of the small object in the image generally does not exceed 32×32. For small-object detection, the backbone does not need such a large receptive field. Therefore, the common backbone needs to be modified to apply to small object detection.

The YOLO series algorithms have three outputs, whether it is an object, which category it belongs to, and the bounding box coordinates of the object. These outputs are calculated by the same feature map. However, the convergence direction of the object positioning and object classification is not consistent. For example, the same object may have different coordinate positions, but the same coordinate position may have different objects. From this perspective, object positioning and object classification cannot use the same features. One of the research ideas of the proposed algorithm is to use different methods to deal with object positioning and object classification separately. This can avoid influence between positioning and classification.

As the input and output sizes of convolution neural networks (CNNs) are determined, the YOLO series algorithms can only detect a fixed number of objects. In order to recognize

untrained categories of objects, we extract the object features from feature maps and design a metric-based method to classify objects. When the algorithm is trained, the classification loss and the positioning loss are backpropagated at the same time.

The overall structure of the proposed detector is shown in Figure 1. The data augmentation module uses methods such as background replacement and noise adding to change the training data to obtain better training performance. The backbone network extracts the multi-layer features in the image for object positioning and object classification. The object positioning module uses high-level features to obtain the center point coordinates, width and height of the object. The object classification module uses the positioning results to extract object features and then judges the category of the object by distance measurement. Finally, the detection result are obtained by combining the positioning and classification calculation.

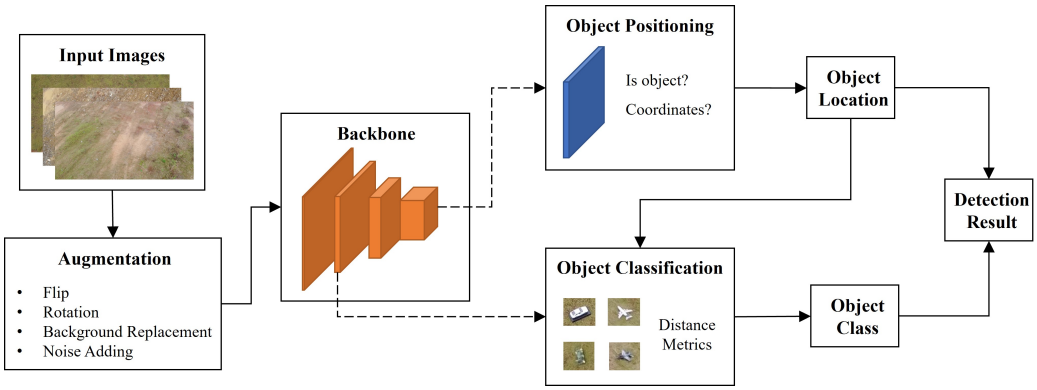


Figure 1. Structure of the proposed small-object detection method, including data augmentation, backbone network and object positioning and object classification modules.

3.1. Image Augmentation

We analyze two main reasons for insufficient datasets. One is that the background in the image is relatively monotonous and the other is that the object state is relatively monotonous. Aimed at these two reasons, we propose two methods to increase the image number of the database, as shown in Figure 2.

The purpose of background replacement is to reduce the impact of background singularity in UAV-based images. We randomly crop some areas in images that are not in the training set to cover areas in training images that do not contain the object. This can increase the diversity of negative samples, making the model eliminate the interference of irrelevant factors.

The output result of the object detection is a rectangular box surrounding the object. However, the object is generally not a standard rectangle, which means that the detected rectangular box will contain some information that does not belong to the object. If the object location does not change much, it is very likely to overfit the background information near the object, which is not conducive to the generalization of the detector. Generally speaking, invalid information will appear at the edge of the rectangular box. Therefore, we design a noise-adding augmentation strategy. We randomly select pixels in the image to cover the pixels near the edge of the rectangular box containing the object. Since we cannot accurately determine whether a pixel belongs to the object or background, we fill the pixels along the bounding box and the pixel block used as noise contains no more than 10 pixels, considering that the object is in the center of the detection box. The pixels near the object are changed by randomly adding noise to improve the generalization of the detector. The formula of background replacement is expressed as follows:

$$\begin{aligned} \tilde{x} &= M \odot x_A + (1 - M) \odot z_B \\ \tilde{y} &= y_A \end{aligned} \tag{1}$$

where $M \in \{0,1\}^{W \times H}$ represents the part of the image that needs to be filled and $\odot$ means pixel by pixel multiplication. x_A and x_B denote two samples in the training set. y_A and y_B represent the label corresponding to the training samples, and z_B is an image in the background image set.

Figure 2. Two data augmentation methods. The red line represents the background replacement and the yellow lines represent noise adding.

3.2. Backbone Design

YOLOv4 proves that CSPDarknet53 is the relatively optimal model. We modify CSPDarknet53 to make it suitable for small-object detection. The comparison between the modified backbone in this paper and the original backbone is shown in the Figure 3.

Compared with the original network, the modified network reduces the receptive field and improves the input image resolution without increasing the computational complexity. Since the research object focuses on small-scale objects, there is no need to consider large-scale objects. The modified backbone deletes the network layers used to predict large-scale objects, which reduces the network depth by half. We call it DCSPDarknet53.

In order to reduce the computational complexity of deep learning, the resolution of the input image is usually downsampled. However, low image resolution will make it difficult to correctly classify and locate small objects. Therefore, a convolutional layer is added in the front of the network to calculate higher-resolution images. We call it as ADCSPDarknet53. At the cost of a small amount of calculation speed, the detection accuracy is improved on a large scale. The specific network structure of our proposed backbone network ADCSPDarknet53 is shown in Figure 4.

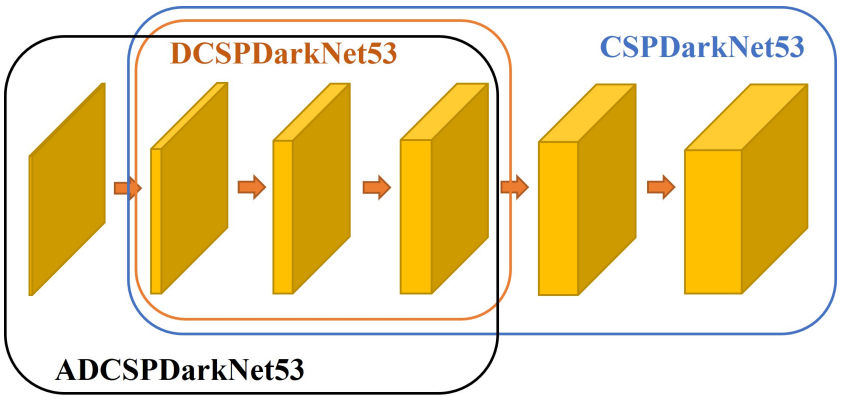


Figure 3. Comparison of backbones. CSPDarknet53 is the backbone of YOLOv4. DCSPDarknet53 is a backbone for small objects. ADCSPDarknet53 is a backbone that increases the downsampling network layer.

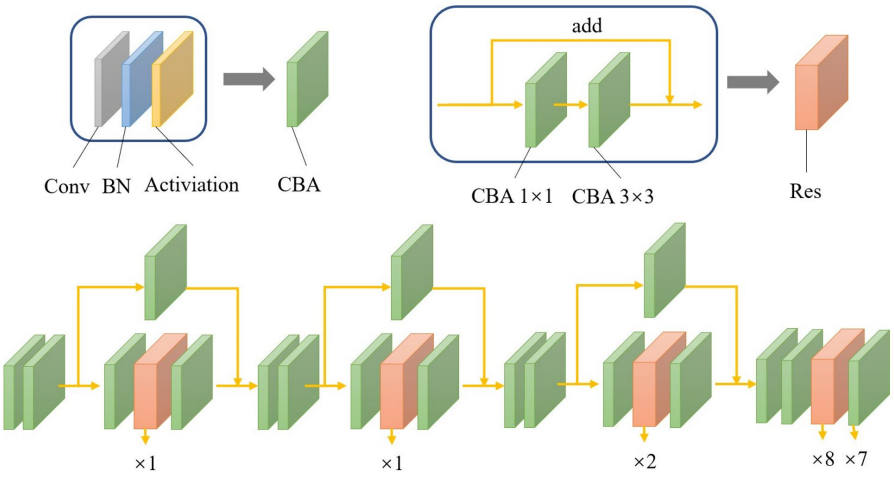


Figure 4. The structure of backbone network ADCSPDarknet53. Conv means convolution layer. BN denotes batch normalization. Activation layer uses ReLU function.

3.3. Object Positioning

The object positioning algorithm is improved based on the YOLO series. YOLOv5 [28] uses a positive sample expansion method. In addition to the original positive sample, two anchor points close to the object center are also selected as positive samples. The calculation formula is expressed as follow:

$$P = \left\{ \begin{array}{c} p, \\ \text{if } (p \cdot x - \lfloor p \cdot x \rfloor \leq 0.5) : p + (-1, 0) \\ \text{if } (p \cdot x - \lfloor p \cdot x \rfloor > 0.5) : p + (1, 0) \\ \text{if } (p \cdot y - \lfloor p \cdot y \rfloor \leq 0.5) : p + (0, -1) \\ \text{if } (p \cdot y - \lfloor p \cdot y \rfloor > 0.5) : p + (0, 1) \end{array} \right\} \tag{2}$$

where P represents the expanded positive sample coordinate set and p means the original positive sample coordinate. For example, in Figure 5, the gray plane is predicted by the grid where the gray plane's center point is located. After the expansion, the gray plane is also predicted by the grid where the red dots are located.

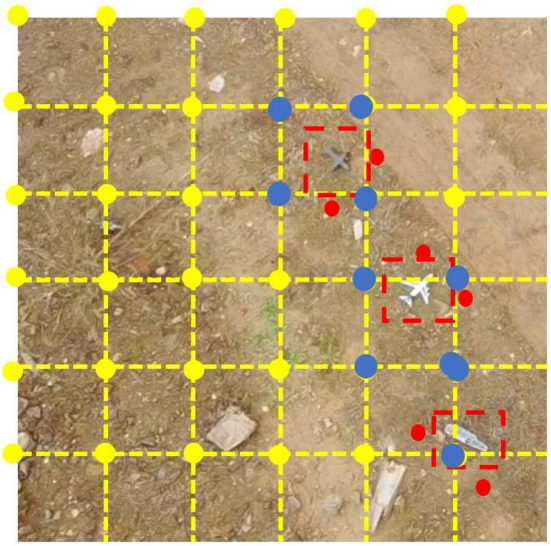


Figure 5. Selection of positive sample. The yellow dots are anchor points. The red dots are positive samples expanded by YOLOv5. The blue dots are positive samples proposed in this article.

The distance between each anchor point is not far in the last feature layer, which means that the object may contain multiple anchor points. It is not appropriate to select the closest anchor point as a positive sample and define other anchor points as negative samples. Therefore, we revise the selection method of positive samples. We calculate the four anchor points around the object center point as positive samples, as shown by the blue dots in Figure 5.

The YOLO series algorithms define the probability that the anchor contains the object as 1. Since each anchor point has a different distance from the object, the strategy of defining all as 1 cannot reflect the difference between different anchor points. Therefore, we use the Euclidean distance between the anchor point and the object center point as the metric for the probability that the anchor contains the object. As the positive sample must contain the object, the probability of containing objects of the positive sample anchor point cannot be 0. According to the design principle, the function of calculating the probability is shown as follows:

$$p_{obj} = 1 - \left((x_a - x_t)^2 + (y_a - y_t)^2 \right) / 4 \tag{3}$$

3.4. Untrained Sub-Class Object Detection

Generally speaking, top-level features are beneficial to object positioning and bottom-level features are beneficial to object classification. In order to separate the process of object localization and object classification to reduce the distractions between these two and make better use of the features extracted from the backbone network, we select features from the middle layers of the backbone for object classification. Through object positioning, we can obtain the coordinates of the object. Using these coordinates, the feature vector of the object can be extracted from feature maps and then can be used to classify the object.

In UAV airborne images, it is common to think of objects in terms of large classes, such as aircraft, cars, buildings, pedestrians, etc. However, specific objects such as black cars and white cars are difficult to determine. To address this sub-class classification problem, we divide the object classification process into the rough classification process and the fine classification process. Rough classification mainly distinguishes objects with large differences in appearance, such as aircraft and cars. Fine classification mainly distinguishes objects that have similar appearance characteristics, but belong to different classes, such as black cars, white cars, etc.

In this paper, a measurement method based on Euclidean distance is used to classify the object. The advantage is that it does not need to fix the object class. By using this metric learning, the algorithm can identify the potential objects in the scene that do not appear in training process, in other words, untrained object. The training goal of object classification is to make the object features of the same class as close as possible and to make the object features of different classes as far as possible. After extracting the object features, three objects are randomly selected from all objects, in which two classes are the same. We use the triple loss [29] as the loss function of object classification. The loss function is defined as:

$$\text{loss}_{cls} = \max(d(a_1, a_2) - d(a_1, b) + \text{ thre }, 0) \tag{4}$$

where a_1 and a_2 represent objects that belong to the same class. b means the object that is different from a_1 and a_2 and *thre* is the expected distance between objects of different classes. The rough classification calculates the loss value between the object classes, while the fine classification calculates the loss value between the object sub-classes. The *thre* of the fine classification process is lower than the *thre* of the rough classification.

In the testing process, we input the labeled images with all objects into the trained model to obtain image features and then extract the feature vector of the object according to the object position to construct a classification database. The classification database is used to classify the object in the test image. The flowchart of object classification is shown in Figure 6. First, the rough classification database is used to determine the object class and then the fine classification database is used to determine the object sub-class.

The principle of object classification is to classify the object into the class closest to the object. If the distance between the object and any category in the database is greater than the threshold, the object is considered to belong to an unknown class that has not appeared in the database. If the distance between the object and the class closest to it is less than the threshold, the object is considered to belong to that class.

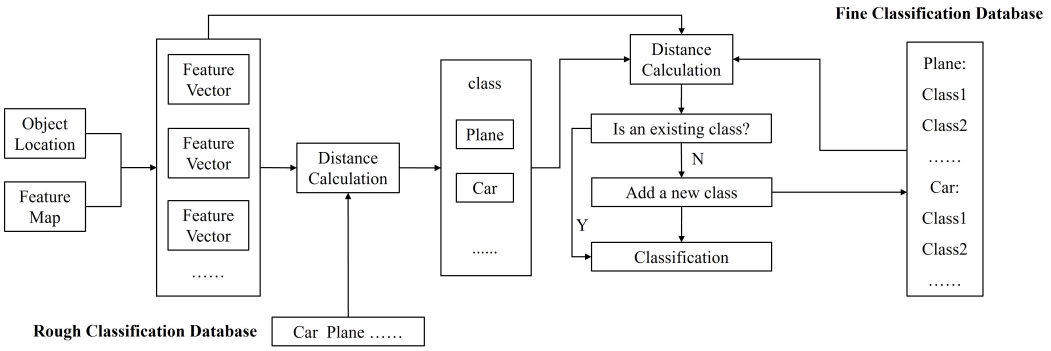


Figure 6. The flowchart of object classification for fine and rough classification.

In summary, we designed two data augmentations—background replacement and noise adding—to increase the background diversity of the dataset. Based on the information flow of small objects through convolution layers, we modified the detector backbone CSPDarkNet53 to ADCSPDarknet53 to obtain a larger feature map for small-object detection as well as to reduce the computation cost. For object positioning, we selected the four anchor points around the object center point as positive samples and modified the function for calculating the objectness probability, which can increase the positioning accuracy of small objects. For object classification, we combined information from shallow feature maps and positioning results to perform rough and fine classification processes to obtain more accurate classification results and identity untrained sub-class small objects.

4. Experiments

To evaluate the small-object detection and classification algorithm proposed in this paper, we first constructed a dataset consists of small objects. Then, we performed experiments on trained and untrained small objects to compare localization and classification performance. Finally, we conducted flight experiments to test the detection performance and real-time inference of the proposed algorithm on small UAVs.

4.1. Dataset of Small Objects and Evaluation Metrics

We choose to detect small objects in the visual field of UAVs to evaluate our algorithm. In order to obtain as much target data as possible, we built a scaled-down experimental environment to collect our dataset. The UAV we used to collect the dataset was a DJI Mini2. To obtain various data, we used some small target models as objects to be detected. The target models were between 15 cm–25 cm in length and 5 cm–20 cm in width. When taking the image data, the flight altitude of the UAV was controlled between 8–10 m to simulate the dataset captured at high altitudes. As the resolution of captured images is 1920×1080 pixels, the pixel ratio of the object to be detected in the image is less than 0.1% as shown in Figure 7a. Eight types of objects were selected to construct the dataset. There are two object classes in the dataset, car and plane. Each class has four sub-classes, which are listed in Figure 7b.

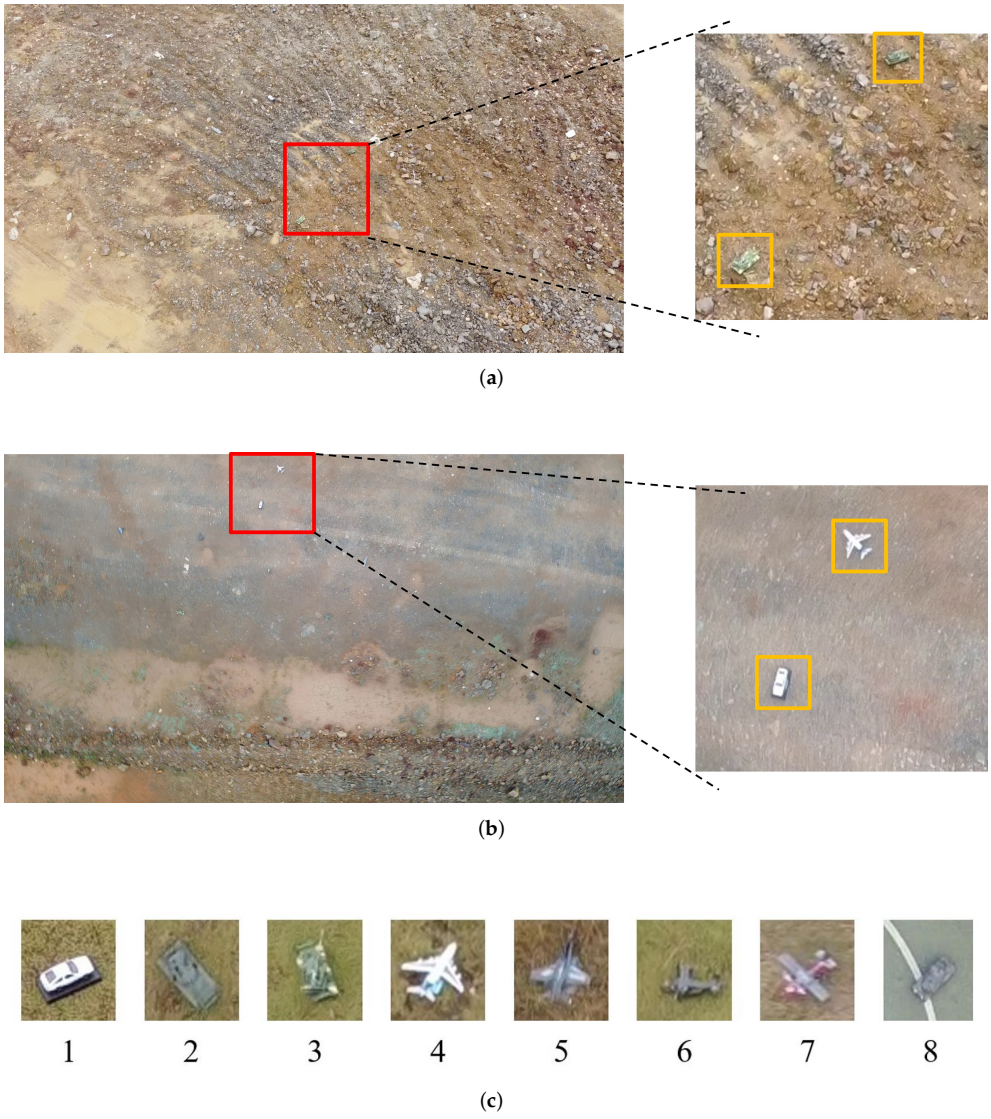


Figure 7. (**a**,**b**) are example images in the collected small-object dataset. The images on the left show the pixel proportion of the object to be detected in the whole image, and the patches on the right are a zoomed-in version of the red box on the left images. (**c**) shows eight objects in the dataset. 1, 2 and 3 are three trained cars belonging to different sub-classes. 4, 5 and 6 are three trained planes belonging to different sub-classes. 7 is an untrained plane and 8 is an untrained car. 7 and 8 are objects that did not appear in the training set and validation set.

The collected dataset are split into training set, validation set and testing set, with 977 images, 100 images and 195 images, respectively. For the object-detection task, the label file contains four object positioning coordinates and an object class label. For the sub-class object classification task, the label file contains two object class labels, in which the first label denotes the object class and the second one is the sub-class. In order to evaluate the detection performance of the proposed algorithm on untrained small objects, the designs of the testing set are slightly different from the training set and validation set. The training set and the validation set contain six types of objects, including three types of cars and three types of planes. In addition to these six types of objects, one type of car and one type of plane are added to the testing set.

For evaluation, we use the general indicators in the field of object detection [30] to evaluate the performance of the proposed algorithm, including mean average precision (mAP), mean average precision at IoU = 0.50 (mAP_{50}) and frames per second (FPS). Intersection over union (IoU) evaluates the overlap between the ground truth bounding

boxes and the predicted bounding box. Based on IoU and the predicted object category confidence scores, mAP is applied to measure the detection performance with classification and localization. mAP_{50} is calculated with IoU ≥ 0.50. These two metrics are performed under the COCO criteria [10]. FPS is used to evaluate the running speed of the algorithm on certain computation platforms. The FPS calculation method is the same as in YOLOv4.

4.2. Implementation Details

We chose YOLOv4 as our baseline model since the proposed network is improved based on YOLOv4. For comparison of object detection, we implemented several object-detection models on the collected dataset, including Faster RCNN (with VGG16 [31] as backbone), SSD [13], FCOS [31], PPYOLO [32], PPYOLOv2 [33], PPYOLOE [34] and PicoDet [35]. Among them, Faster RCNN is a two-stage object detector and the rest are one-stage detectors. FCOS, PPYOLOE and PicoDet are anchor-free detectors, and PicoDet is designed for mobile devices. All the detectors have the same size of input image (608×608) and were trained from the pretrained model on the COCO dataset [10] to obtain faster and better convergence. The training epochs were set to 300 with initial learning rate 0.001 and Adam optimizer. Then, learning rate decayed by 0.1 times at the 150th epoch and 250th epoch. The batch size was set to be 16. For the special sub-class classification part of our method, the *thre* of the rough classification was 10 and the *thre* of the fine classification was 2. All the training and testing experiments were conducted on one NVIDIA Quadro GV100 GPU.

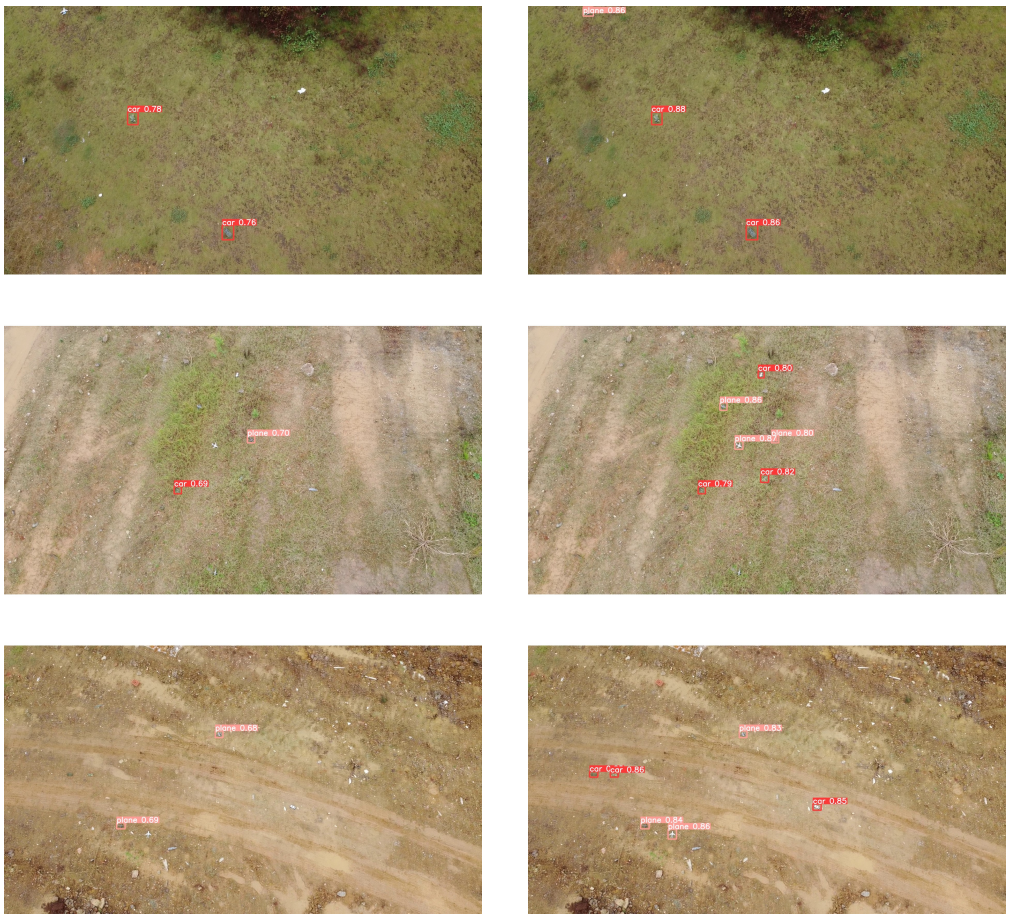
4.3. Experiment Results

4.3.1. Small Object Detection

We compare the detection performance on small objects with several existing object detectors. The results are listed in Table 1, from which we can see that our proposed method gives the highest average precision (33.80%) compared to the others, as well as running at the highest speed (77 FPS). It also achieves the third highest mAP_{50}, with 61.00% among ten models. These improvements can be attributed to the following aspects: (1) the modified backbone network focuses more on the small objects and discards the deep layers, which have little effect on detecting small objects. Meanwhile, the interference of extra parameters on network learning is reduced. (2) Metric-based learning improves the ability of the network to classify objects. (3) The proposed object positioning method increase the number of positive samples and thus improves small-object localization abilities. (4) The modified backbone network reduces computation significantly, which makes the network run at a faster speed and achieves the performance of real-time detection. Figure 8 shows some detection results of the collected dataset using YOLOv4 and our algorithm. Our algorithm has stronger ability to detect small objects.

Table 1. Experiment results of small object detection.

Method	FPS	mAP_{50}	mAP
PPYOLOE_s	56	33.40%	12.70%
PPYOLOE_l	34	52.50%	22.40%
PicoDet_s	68	21.70%	7.50%
FasterRCNN	18	29.40%	10.80%
SSD	51	34.80%	10.80%
PPYOLO	54	63.70%	27.50%
PPYOLOv2	36	62.00%	25.60%
FCOS	26	21.20%	8.80%
YOLOv4	71	39.60%	21.70%
Ours(ADCSPDarknet53)	**77**	61.00%	**33.80%**



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

Figure 8. Small-object detection experiment with other algorithms. (**a**) YOLOv4 algorithm detection results; (**b**) our algorithm detection results.

4.3.2. Untrained Sub-Class Object Classification

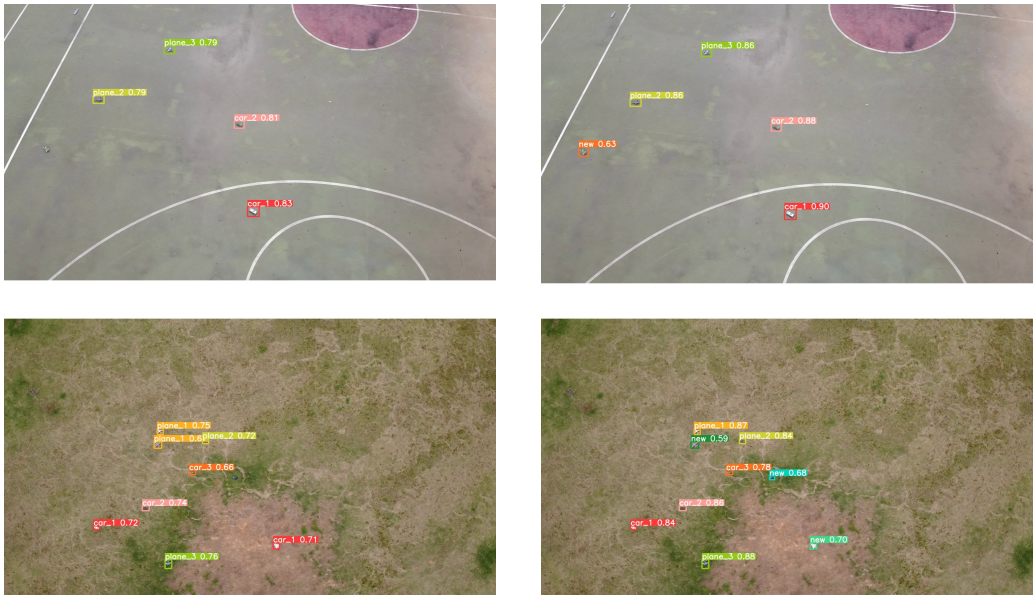
In our proposed method, object classification includes the process of rough object classification and fine object classification. The designed classification method has two advantages. One is that it can classify untrained objects, and the other is that it avoids the mutual influence of classification and positioning. To illustrate both points, we conduct multiple comparative experiments. The experimental results of object classification are shown in Table 2. Experiments 1, 2 and 3 represent detection results of YOLOv4 under different conditions. Experiments 4 and 5 are the detection results of the same rough classification model under different test categories. Experiments 6, 7 and 8 are the detection results of the same fine classification model under different test categories.

Comparing Experiments 1 and 3, the accuracy of object detection decreases from 88.30% to 65.60% when objects are classified during training, which shows the interference between object classification and localization. Through Experiments 3 and 4, it can be found that metric-based learning is beneficial to improve the result of object detection, as the mAP_{50} increases from 65.60% to 88.30%. It can be demonstrated by Experiments 1, 3 and 6 that the detection performance of the object is worse for more categories. In Experiment 8, we can find that the untrained car and the untrained plane can be detected with 49.4% mAP_{50} and 34.1% mAP_{50}, respectively. Although the metric-based method is not very accurate in detecting untrained objects, it can still locate untrained objects and distinguish them from trained objects.

Table 2. Experiment results of untrained object classification.

Experiment	Model	Number of Training Classes	Number of Testing Classes	mAP_{50} (Trained)	mAP_{50} (Untrained)
1	YOLOv4	1	1	88.30%	-
2	YOLOv4	2	2	51.10%	-
3	YOLOv4	2	1	65.60%	-
4	rough classification	2	1	83.80%	-
5	rough classification	2	2	64.70%	-
6	fine classification	9	1	78.30%	-
7	fine classification	9	2	57.70%	-
8	fine classification	9	9	41.900%	aircraft: 34.10% car: 49.40%

Figure 9 shows the visualization results of untrained object classification using YOLOv4 and our algorithm. For the untrained sub-class objects, YOLOv4 will give incorrect classification results, but the proposed algorithm will add these untrained objects to new sub-classes using metric-based learning.



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

Figure 9. Untrained object classification experiment with other algorithms. (**a**) YOLOv4 algorithm detection results; (**b**) our algorithm detection results. Different colors of bounding boxes mean different sub-classes, and the recognized untrained sub-class objects are labeled with 'new'.

4.3.3. Ablation Study

To analyze the effectiveness of our proposed method, we conducted an ablation study on data augmentation, backbone design and object positioning.

Data Augmentation. Based on YOLOv4, we analyze the results of two data augmentation methods. It can be seen from Table 3 that the two types of data augmentation can improve detection performance. The results show that replacing part of the image background can lead the network to learn more combinations of patterns and effectively increase the diversity of the dataset. Adding noise around the object reduces the overfitting to special backgrounds. However, the effect of running the two methods at the same time

is not as good as the effect of running the two methods separately. This is mainly because the data augmentation method we propose introduces significant noise while increasing the diversity of the dataset. Applying both methods at the same time may cause too much noise and cannot obtain better performance.

Table 3. Experiment results of data augmentation.

Image Input Size	Replace Background	Add Noise	mAP_{50}	mAP
608 × 608	N	N	36.50%	19.10%
608 × 608	Y	N	44.60%	24.90%
608 × 608	N	Y	41.70%	20.70%
608 × 608	Y	Y	39.60%	22.40%

Backbone Design. The backbone design consists of two steps. First, the DCSPDarknet53 deletes the deep network layers used to detect large-scale objects in the original detection backbone CSPDarknet53 to reduce the influence of the deep network on the detection of small objects. Then, ADCSPDarknet53 adds a network layer for downsampling at the front end of the network, so as to obtain better detection results while increasing the computational complexity as little as possible. The experiment results are shown in Table 4. Compared to the CSPDarknet53-based detector, there is little increase in mAP_{50} and mAP of the DCSPDarknet53-based detector in small-object detection, but the calculation speed is more than double, which proves that small-object detection can use a high-resolution network with fewer layers. As for the ADCSPDarknet53-based detector, the mAP_{50} is increased by 18% and the mAP is increased by 11% compared to the original detector. Although the FPS drops, it still runs faster than the CSPDarknet53-based detector. In the actual scene, the image size and backbone can be adjusted as needed.

Table 4. Experiment results of backbone design.

Backbone	Image Input Size	FPS	mAP_{50}	mAP
CSPDarknet53	608 × 608	71	39.60%	21.70%
DCSPDarknet53	608 × 608	166	42.90%	23.90%
ADCSPDarknet53	1216 × 1216	77	57.60%	32.70%

Object Positioning. We modified the loss function of object positioning; Table 5 shows the detection evaluation of detectors with and without loss function modification. After modifying the loss function of object positioning, the mAP_{50} of the CSPDarknet53-based detector is increased by 10.7% and the mAP is increased by 2.9%. For the ADCSPDarknet53-based detector, the mAP_{50} increased by 3.4% and the mAP increased by 1.1%. This proves the effectiveness of the modified loss function, which can select positive samples that can represent the ground truth more accurately from many candidate samples. The training process of object positioning is shown in Figure 10. With the modified object positioning loss function, the training process is more stable.

Table 5. Experiment results of object positioning.

Backbone	Loss Function Modification	mAP_{50}	mAP
CSPDarknet53	N	39.60%	21.70%
CSPDarknet53	Y	50.30%	24.60%
ADCSPDarknet53	N	57.60%	32.70%
ADCSPDarknet53	Y	61.00%	33.80%

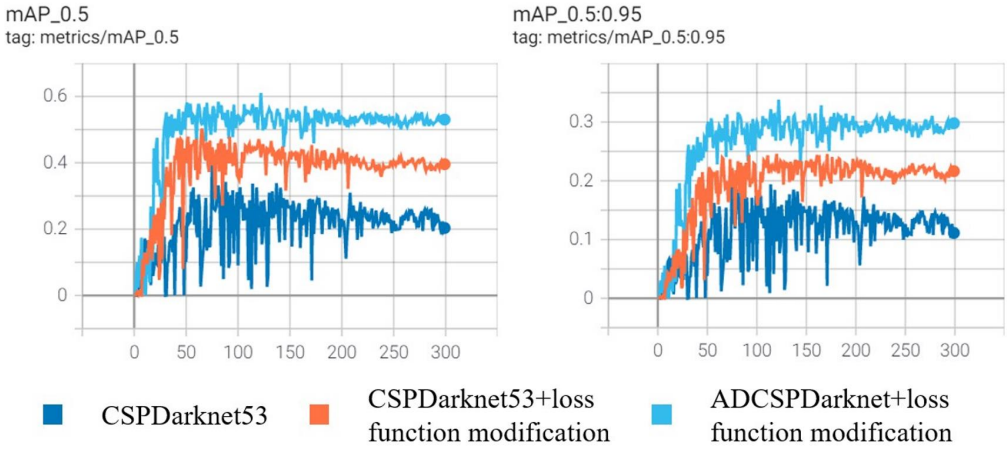


Figure 10. Training process of object positioning.

4.4. Flight Experiments

In order to verify the effectiveness of the proposed algorithm in the actual UAV application scenario, we built a UAV flight experiment system and deployed the proposed algorithm on a small drone.

4.4.1. Experiment Settings

The drone we used as the experiment platform was a Matrice M210v2 drone manufactured by DJI Innovations. Its overall dimensions are 883 × 886 × 398 mm with a maximum takeoff weight 6.14 kg. The onboard optical camera was a DJI innovation company Chansi X5s camera equipped with a DJI MFT 15 mm/1.7 ASPH lens. It was fixed to the drone body through a 3-DOF gimbal and its posture can be controlled with a remote controller. The resolution of the images taken by the camera was set to be 1920 × 1080 pixels. To implement our proposed algorithm on the drone, we deployed a Nvidia Jetson Xavier NX processor on the drone for real-time processing. In order to ensure the safe outdoors flight of the drone, we also set the real-time kinematic (RTK) global navigation satellite system on the drone for the positioning. The flight experiment system is shown in Figure 11.

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

Figure 11. Hardware system for flight experiments. (**a**) DJI Matrice M210v2 drone; (**b**) DJI camera; (**c**) Nvidia Jetson Xavier NX.

The proposed algorithm is implemented by PyTorch 1.10 on Nvidia Jetson Xavier NX's GPU with a computational capacity of 7.2. All programs run on Robot Operating System (ROS) systems. While the drone is flying, we use the Rosbag tool to record the on-board processing data, such as the real-time detection image results. Once the drone is back on the ground, we can use the Rosbag's playback function to check how well the algorithm works.

We set up two flight scenarios to validate our algorithm with trained and untrained objects. In one detection scenario, the objects to be detected were the small models used for creating the above dataset, but with new backgrounds. In this case, the flight altitude of the drone was set to 10 m to stay consistent with the dataset. The purpose of this detection

scenario is to verify the generalization performance of the learned model in practical application scenarios. In another detection scenario, the model detected real vehicles with flight altitude of 95 m. We used seven different types of vehicles to test the classification and localization ability of the model for object classes with high inter-class similarity. In this case, we collected new data but only six types were labeled and appear in the training set. Then the model is retrained and tested for detection performance and speed. The two scenarios settings are shown in Figure 12.

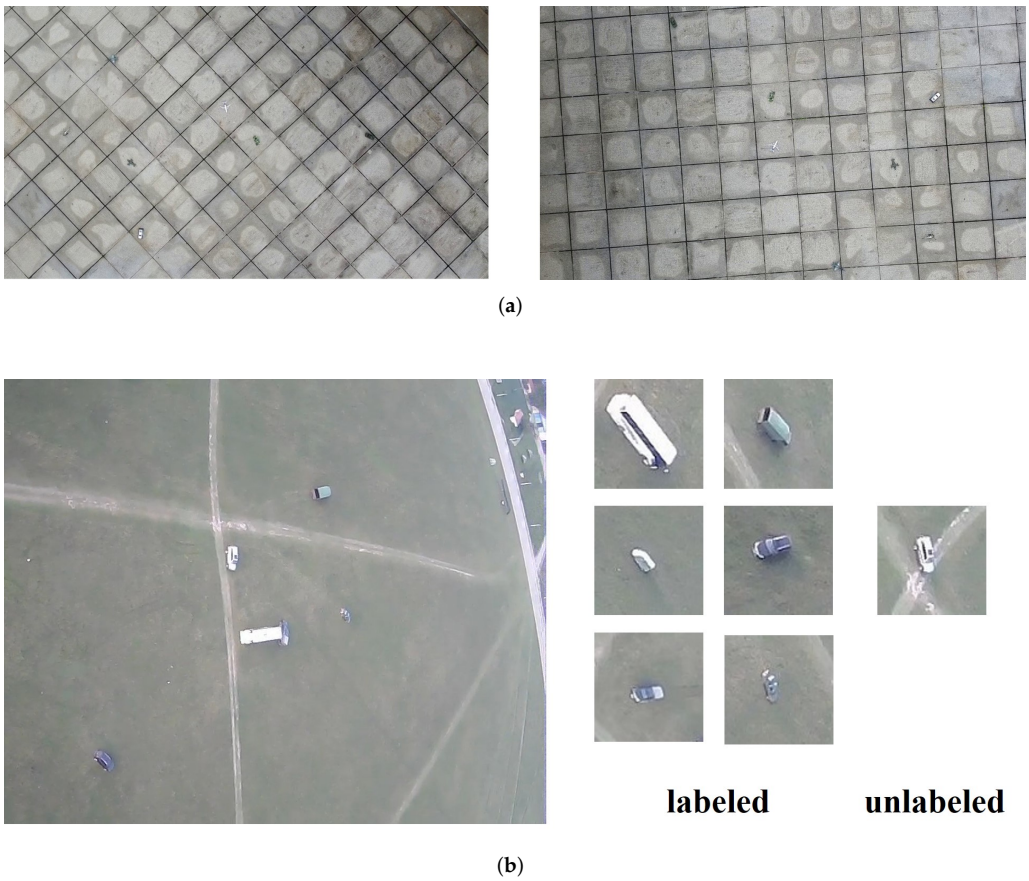
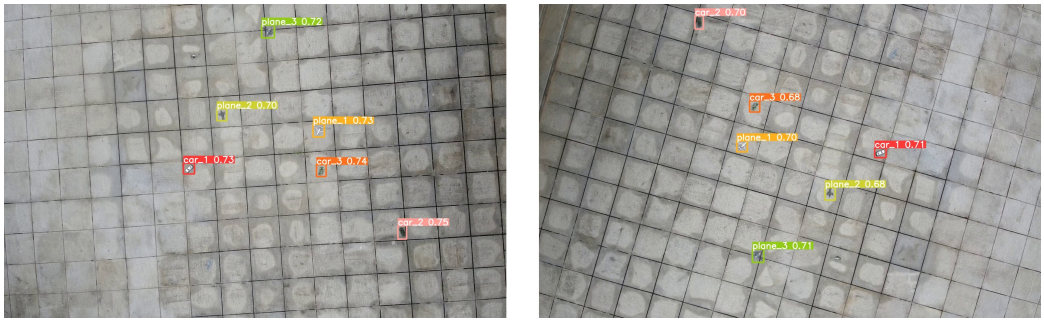


Figure 12. (**a**) The first detection scenario is set to have the same objects as the collected dataset, but with different background. (**b**) The second detection scenario uses real vehicles as objects. The figure on the left shows part of the drone's field of view, and the right images show different types of vehicles, with six labeled types and one unlabeled.

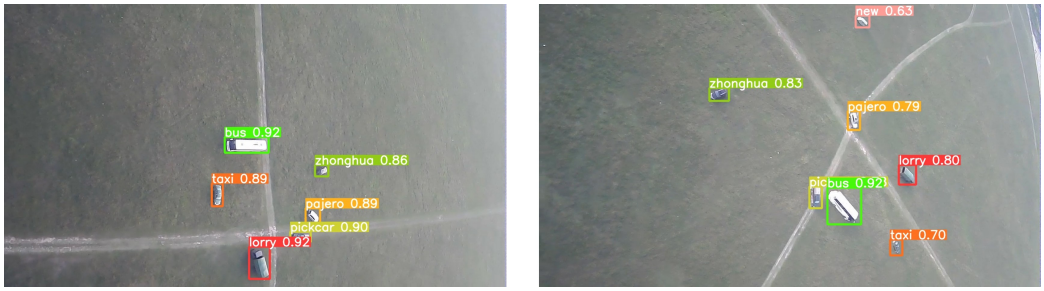
4.4.2. Results

Some qualitative detection results are shown in Figure 13. In Figure 13a, our proposed algorithm can detect small objects in the visual field without being influenced by the changing background. This is because the data enhancement method we used effectively prevents the model from overfitting to the background during the training process. In Figure 13b, the learned objects are detected by the proposed detector. In addition, the potential target (the unlabeled one) is also identified by the network and classified into a new class according to metric learning.

It is worth noting that small-object detection during flight faces additional challenges, such as camera vibration and motion caused by flight. Camera motion in the imaging process leads to the blurring of objects and damages the features. In this case, our algorithm can still detect small objects in the airborne visual field accurately. Our proposed algorithm not only extracts the features of small objects with CNNs, but also distinguishes the inter-class and intra-class differences of objects by measuring the distance metric. This more powerful feature extraction method helps reduce the effect of motion blur.



(**a**)



(**b**)

Figure 13. (**a**) Detection results on small model objects with different backgrounds. (**b**) Detection results on seven types of vehicles.

We also checked the real-time performance of our proposed method. We computed the runtime of the algorithm on the edge GPU (Nvidia Xavier NX) using different input image resolutions, including the pre-processing phase, model inference phase, and post-processing phase. The results are listed in Table 6. For images with input sizes of 416 × 416, 608 × 608 and 640 × 640, our algorithm reaches an average speed of 22 FPS. The small runtime difference mainly comes from the pre-processing phase and post-processing phase, since these two phases are run on the CPU. The parallel computing capability of the GPU makes the inference time of the model almost the same. However, for images with input size of 1216 × 1216, it takes more than twice as long to process a single frame. As there are four times as many pixels in the image, more time is needed to perform normalization for each pixel and permute the image channels in the pre-processing stage. For model inference, the larger input image size makes the feature map in the network larger, with more activation needs to compute, which accounts for the increase of time usage [36]. During the post-processing phase, more candidate detection boxes need to be computationally suppressed.

In our flight test, we used an image input size of 608 × 608. Without any acceleration library, our algorithm can achieve real-time performance, which is sufficient for reconnaissance missions on UAVs. Some runtime optimizations can be made, for example, using the TensorRT [37] library to accelerate the model inference, or by improving the code efficiency in pre-processing and post-processing stages.

Table 6. Experiment results of real-time performance.

Image Input Size	Average Pre-Processing Time (ms)	Average Inference Time (ms)	Average Post-Processing Time (ms)	Average Total Process Time (ms)	FPS
416 × 416	1.6	40.8	1.3	43.7	22.88
608 × 608	2.1	41.0	1.3	44.4	22.52
640 × 640	2.2	41.3	1.4	44.9	22.27
1216 × 1216	3.1	51.9	49.3	104.3	9.59

5. Conclusions

Aimed at challenges such as small-scale objects, untrained objects during inference, and real-time performance requirements, we designed a detector to detect small objects in UAV reconnaissance images. To conduct our research, we collected a small object dataset from the perspective of a high-flying UAV. We proposed two data augmentation methods, background replacement and noise adding, which improve the background diversity of the collected dataset. For the backbone design, we designed ADCSPDarkent53 based on the characteristics of small objects and evaluated the improved backbone on accuracy and speed. For object positioning, we modified the positioning loss function, which greatly improved detection accuracy. For object classification, a metric-based classification method was proposed to solve the problem of untrained sub-class object classification. Experiments on UAV-captured images and flight tests show the effectiveness and applicable scope of the proposed small-object detector. In the next step, improvements can be made in terms of dataset construction, feature selection and metric function design.

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

Funding: This paper was supported by National Natural Science Foundation of China (No. 61876187) and Natural Science Foundation of Hunan Province (No. S2022JJQNJJ2084 and No. 2021JJ20054).

Acknowledgments: Thanks to the Unmanned Aerial Vehicles Teaching and Research Department, Institute of Unmanned Systems, College of Intelligence Science and Technology, National University of Defense Technology for providing the experimental platform.

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

References

1. Belmonte, L.M.; Morales, R.; Fernández-Caballero, A. Computer Vision in Autonomous Unmanned Aerial Vehicles—A Systematic Mapping Study. *Appl. Sci.* **2019**, *9*, 3196. [CrossRef]
2. Zhang, H.; Wang, L.; Tian, T.; Yin, J. A Review of Unmanned Aerial Vehicle Low-Altitude Remote Sensing (UAV-LARS) Use in Agricultural Monitoring in China. *Remote Sens.* **2021**, *13*, 1221. [CrossRef]
3. Zheng, Y.J.; Du, Y.C.; Ling, H.F.; Sheng, W.G.; Chen, S.Y. Evolutionary Collaborative Human-UAV Search for Escaped Criminals. *IEEE Trans. Evol. Comput.* **2020**, *24*, 217–231. [CrossRef]
4. Wu, X.; Li, W.; Hong, D.; Tao, R.; Du, Q. Deep Learning for Unmanned Aerial Vehicle-Based Object Detection and Tracking: A survey. *IEEE Geosci. Remote Sens. Mag.* **2022**, *10*, 91–124. [CrossRef]
5. Alexey, B.; Wang, C.; Mark Liao, H. Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
6. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
7. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 14–19 June 2020.
8. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.

9. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
10. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Proceedings of the Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755.
11. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
12. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [CrossRef] [PubMed]
13. 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: Cham, Switzerland, 2016; pp. 21–37.
14. 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, 26 June–1 July 2016; pp. 779–788.
15. 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.
16. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850.
17. Zhou, X.; Zhuo, J.; Krahenbuhl, P. Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 850–859.
18. 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: Cham, Switzerland, 2020; pp. 213–229.
19. Beal, J.; Kim, E.; Tzeng, E.; Park, D.H.; Zhai, A.; Kislyuk, D. Toward transformer-based object detection. *arXiv* **2020**, arXiv:2012.09958.
20. 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.
21. Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; Cho, K. Augmentation for small object detection. *arXiv* **2019**, arXiv:1902.07296.
22. Singh, B.; Davis, L.S. An analysis of scale invariance in object detection snip. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3578–3587.
23. Schumann, A.; Sommer, L.; Klatte, J.; Schuchert, T.; Beyerer, J. Deep cross-domain flying object classification for robust UAV detection. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–6.
24. Li, J.; Liang, X.; Wei, Y.; Xu, T.; Feng, J.; Yan, S. Perceptual generative adversarial networks for small object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1222–1230.
25. Yundong, L.; Han, D.; Hongguang, L.; Zhang, X.; Zhang, B.; Zhifeng, X. Multi-block SSD based on small object detection for UAV railway scene surveillance. *Chin. J. Aeronaut.* **2020**, *33*, 1747–1755.
26. Liu, Y.; Yang, F.; Hu, P. Small-object detection in UAV-captured images via multi-branch parallel feature pyramid networks. *IEEE Access* **2020**, *8*, 145740–145750. [CrossRef]
27. Du, D.; Zhu, P.; Wen, L.; Bian, X.; Lin, H.; Hu, Q.; Peng, T.; Zheng, J.; Wang, X.; Zhang, Y.; et al. VisDrone-DET2019: The vision meets drone object detection in image challenge results. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 27–28 October 2019.
28. Glenn, J.; Ayush, C.; Jirka, B.; Alex, S.; Yonghye, K.; Jiacong, F.; Tao, X.; Kalen, M.; Yifu, Z.; Colin, W.; et al. Ultralytics/Yolov5. Available online: https://github.com/ultralytics/yolov5 (accessed on 1 April 2021).
29. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.
30. Padilla, R.; Netto, S.L.; Da Silva, E.A. A survey on performance metrics for object-detection algorithms. In Proceedings of the 2020 international conference on systems, signals and image processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020; pp. 237–242.
31. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos: Fully convolutional one-stage object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 9627–9636.
32. Long, X.; Deng, K.; Wang, G.; Zhang, Y.; Dang, Q.; Gao, Y.; Shen, H.; Ren, J.; Han, S.; Ding, E.; et al. PP-YOLO: An effective and efficient implementation of object detector. *arXiv* **2020**, arXiv:2007.12099.
33. Huang, X.; Wang, X.; Lv, W.; Bai, X.; Long, X.; Deng, K.; Dang, Q.; Han, S.; Liu, Q.; Hu, X.; et al. PP-YOLOv2: A practical object detector. *arXiv* **2021**, arXiv:2104.10419.
34. Xu, S.; Wang, X.; Lv, W.; Chang, Q.; Cui, C.; Deng, K.; Wang, G.; Dang, Q.; Wei, S.; Du, Y.; et al. PP-YOLOE: An evolved version of YOLO. *arXiv* **2022**, arXiv:2203.16250.
35. Yu, G.; Chang, Q.; Lv, W.; Xu, C.; Cui, C.; Ji, W.; Dang, Q.; Deng, K.; Wang, G.; Du, Y.; et al. PP-PicoDet: A Better Real-Time Object Detector on Mobile Devices. *arXiv* **2021**, arXiv:2111.00902.

36. Dollar, P.; Singh, M.; Girshick, R. Fast and Accurate Model Scaling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 924–932.
37. Rajeev, R.; Kevin, C.; Xiaodong, H.; Samurdhi, K.; Shuyue, L.; Ryan, M.; Gwena, C.; Vinh, N.; Boris, F.; Paul, B.; et al. TensorRT Open Source Software. Available online: https://github.com/NVIDIA/TensorRT (accessed on 2 October 2022).