drones

MDPI

Article

A Real-Time UAV Target Detection Algorithm Based on Edge Computing

Qianqing Cheng *, Hongjun Wang *, Bin Zhu, Yingchun Shi and Bo Xie

School of Electronic Countermeasures, National University of Defense Technology, Hefei 230037, China
* Correspondence: chengqianqing@nudt.edu.cn (Q.C.); wanghongjun17@nudt.edu.cn (H.W.); Tel.: +86-1552-803-7356 (Q.C.)

Abstract: Small UAV target detection plays an important role in maintaining the security of cities and citizens. UAV targets have the characteristics of low-altitude flights, slow speeds, and miniaturization. Taking these characteristics into account, we present a real-time UAV target detection algorithm called Fast-YOLOv4 based on edge computing. By adopting Fast-YOLOv4 in the edge computing platform NVIDIA Jetson Nano, intelligent analysis can be performed on the video to realize the fast detection of UAV targets. However, the current iteration of the edge-embedded detection algorithm has low accuracy and poor real-time performance. To solve these problems, this paper introduces the lightweight networks MobileNetV3, Multiscale-PANet, and soft-merge to improve YOLOv4, thus obtaining the Fast-YOLOv4 model. The backbone of the model uses depth-wise separable convolution and an inverse residual structure to simplify the network's structure and to improve its detection speed. The neck of the model adds a scale fusion branch to improve the feature extraction ability and strengthen small-scale target detection. Then, the predicted boxes filtering algorithm uses the soft-merge function to replace the traditionally used NMS (non-maximum suppression). Soft-merge can improve the model's detection accuracy by fusing the information of predicted boxes. Finally, the experimental results show that the mAP (mean average precision) and FPS (frames per second) of Fast-YOLOv4 reach 90.62% and 54 f/s, respectively, in the workstation. In the NVIDIA Jetson Nano platform, the FPS of Fast-YOLOv4 is 2.5 times that of YOLOv4. This improved model performance meets the requirements for real-time detection and thus has theoretical significance and application value.

Keywords: unmanned aerial vehicle (UAV); object detection; non-maximum suppression (NMS); weighted boxes fusion (WBF); lightweight network; multiscale

Citation: Cheng, Q.; Wang, H.; Zhu, B.; Shi, Y.; Xie, B. A Real-Time UAV Target Detection Algorithm Based on Edge Computing. *Drones* **2023**, *7*, 95. https://doi.org/10.3390/drones7020095

Academic Editor: Anastasios Dimou

Received: 17 December 2022
Revised: 15 January 2023
Accepted: 24 January 2023
Published: 30 January 2023

1. Introduction

In recent years, the rapid development of drone technology has resulted in increased convenience in people's daily lives. However, invasive and disorderly flying of UAVs are increasing, posing a serious threat to public safety [1]. UAV targets have the characteristics of low flying altitudes, slow speeds, and miniaturization [2]. This makes their detection by traditional radar and radio methods difficult and costly [3,4]. Detecting UAVs based on sound is greatly disturbed by noise, so their detection is not obvious [5]. In addition to the above methods, the deep learning method based on two-dimensional image data is gradually being applied to UAV detection and has achieved good results [6,7]. Therefore, it is promising to study a new object detection method based on UAV images to address the lack of suitable detection methods.

With the development of deep learning and updates to graphics computing devices, image-based target detection methods have become a hotspot in the field of target detection [8]. An increasing number of engineering practices are being applied to facial recognition, pedestrian target detection, and autonomous driving. The representative algorithms in these technologies include R-CNN and Faster R-CNN based on two-stage

detection [9], as well as SSD (single-shot multi-box detector) and YOLO (you only look once) series algorithms based on one-stage detection [10]. The former methods are called two-stage algorithms because the target candidate regions are generated separately from the classification localization. The latter are called one-stage algorithms because the extraction of features and classification localization are carried out continuously. As the fourth version of the YOLO series of algorithms, YOLOv4 is the first to use CSPDarkNet-53 as the backbone network to extract image features [11], PANet (path aggregation network) as the feature fusion structure, and SPP (spatial pyramid pooling) to enhance feature extraction [12], which greatly improves the performance of the one-stage detection algorithm in model accuracy and reasoning speed.

At present, object detection applications are mostly divided into cloud computing and edge computing. Cloud computing is a mode of unifying the data collected on the edge side to the central processor for operation and then making decisions. This model has high latency, network instability, and low bandwidth problems, making it unsuitable for UAV detection tasks that require fast responses and have high error costs. On the contrary, edge computing can solve these problems well. Edge computing is a technology that provides cloud services and IT environment services for developers and service providers at the edge side of the network [13]. By integrating detection algorithms, edge computing platforms can partially replace the data processing functions of cloud devices and servers and directly store or detect the target data. This edge-side data processing method not only reduces the data transmission and communication time between devices but also saves on network bandwidth and energy consumption. Moreover, this method also frees target detection from its dependence on large servers and GPU devices and enables users to flexibly arrange edge computing devices to meet the needs of various detection tasks.

In target detection, accuracy and speed are the two most important metrics to measure a model's performance. Many scholars have also improved the detection algorithm based on these two points. In terms of accuracy, the proposed feature pyramid network and the application of various box-filtering algorithms enhance the detection capability of the model for multiscale targets. In terms of speed, various lightweight networks are proposed to greatly reduce the model parameters and accelerate the detection speed, so that the detection algorithm can be carried to the edge computing platform to complete the task. Aiming at the problems of slow detection speed and poor effect of multiscale target detection, we propose a UAV target detection algorithm Fast-YOLOv4 based on edge computing, which combines the lightweight network MobileNetV3, the improved Multiscale-PANet, and the soft-merge algorithm. To summarize, the innovations of the algorithm are as follows:

- Using the lightweight network MobileNetV3 to improve the backbone of YOLOv4 and introduce depth-wise separable convolution and inverse residual structures. These improvements can greatly reduce the number of model parameters and improve the detection speed.
- Adding the fourth feature fusion scale in the neck structure, PANet, of YOLOv4 and enhancing the flow superposition of high-dimensional image features and low-dimensional location features, improving the classification and localization accuracy of multiscale targets.
- Replacing NMS with the improved soft-merge algorithm. It is a predicted box-filtering algorithm and can fuse the predicted box information to obtain better-predicted boxes. The problems of missing detection and false detection are also reduced, enhancing the recognition effect.
- Combined with the edge-embedded platform NVIDIA Jetson Nano, the fast target detection algorithm is equipped to reduce the response time and energy consumption, and realize the real-time multi-scene accurate detection of UAV targets.

Through those improvements, Fast-YOLOv4 achieves high precision and fast detection of UAV targets on an edge computing platform, which provides a solution for research on UAV target detection.

2. Related Work

In this section, we mainly introduce work to four parts of our model: multiscale detection, box-filtering algorithms, lightweight networks, and edge computing about UAVs, as well as their related methods and applications. For a long time, multiscale detection has presented a major problem in target detection. From the initial image pyramid network and feature pyramid network method to the most commonly used Res2Net and PANet methods [12,14,15], multiscale detection is gradually maturing. For example, aiming at the significant differences in the scales of drone targets, Zeng et al. proposed a UAV detection network combining Res2net and a mixed feature pyramid structure, which achieved a more than 93% mAP in a self-built dataset [16]. However, the model is a little bloated and difficult to carry over to the edge platform. For mask target detection, Zhu et al. introduced a path aggregation network into the input feature layer of YOLOv4-Tiny, which improves the mAP of mask detection by 4.3% [17]. However, the method does not propose innovative improvements to the multiscale structure in that paper, which is slightly inadequate.

Another way to improve the model accuracy is to improve the box-filtering algorithm. Because the traditional NMS method is not effective in detecting occluded targets, many scholars have tried to study and improve it. To solve the problem of poor detection of occluded targets, Zhang et al. introduced position information into the NMS to adjust the final score of the box and proposed a distance-based intersection ratio loss function. The method was tested using classical datasets, and the accuracy was improved by 2% [18]. Ning et al. proposed an improved non-maximum suppression method called I-SSD (Inception-SSD) that can obtain weighted averages of the coordinates of the predicted boxes and improve the ability of the model's filtering results [19]. The proposed I-SSD algorithm achieves a 78.6% mAP on the Pascal VOC 2007 test. Roman et al. also proposed a weighted box fusion method by fusing predicted boxes of different detection models to obtain integration results. The method significantly improves the quality of the fused predicted rectangles for an ensemble and achieves the best results in dataset challenge competitions [20].

In addition to the detection accuracy of multiscale targets, the speed improvement in the detection model and the deployment of an edge platform are also current hot issues. To solve these problems, most scholars use lightweight networks to improve the target detection model. Zhou et al. proposed a lightweight YOLOv4 ship detection algorithm combined with MobileNetv2 to solve the problem that the large model could not run on the micro-platform and realized the high-precision and rapid detection of ship targets [21]. Liu et al. proposed a pruned-YOLOv4 model to overcome the problem of low detection speed and poor detection effect of small targets. The model achieves 90.5% mAP, and its processing speed is increased by 60.4%, which accomplishes the real-time detection of drones well [22]. However, the accuracy performance of the model is slightly inadequate. It can be seen from these papers that lightweight networks are very effective at improving target detection, which suggests ideas for their application to target detection in more ways.

Because of the broad application prospects of object detection combined with edge computing, many researchers try to carry the object detection algorithm onto the embedded platform to complete the detection task. Daniel et al. achieve an accurate detection of UAV targets by building YOLOv3 on the edge platform of NVIDIA Jetson TX2, with an average accuracy of 88.9% on the self-built dataset [23]. To solve the problem of dynamic obstacle avoidance in safe drone navigation, Adrian et al. propose a novel method of performing onboard drone detection and localization using depth maps. This method is integrated into a small quadrotor and reduces the maximum obstacle avoidance error distance to 10% [24]. In terms of expanding the extent of applicable mission scenarios of UAVs and coordinating the flight formations of fleets of UAVs, Roberto et al. proposed a YOLO object detection system integrated into an original processing architecture. This approach achieves a high level of accuracy and is robust against challenging conditions in terms of illumination, background, and target-range variability [25].

Combined with the ideas for improvement presented by related papers, the proposed Fast-YOLOv4 algorithm in this paper improves the detection accuracy in a multiscale

structure and predicted box algorithm and improves the speed of the lightweight network. In the following experiments, the proposed algorithm shows higher accuracy and faster speed than other algorithms on the edge computing platform and realizes a highly accurate and fast detection of multiscale UAV targets.

3. System Model

This section mainly introduces the design idea of the system model. Figure 1 shows the system architecture of target detection using the edge computing platform and workstation. The target detection system consists of three parts: a high-speed intelligent camera, an edge computing platform NVIDIA Jetson Nano, and a workstation. Firstly, the system will set up a high-speed intelligent camera in a detection scene or on a mobile platform such as a robot. Then, the camera will monitor the surrounding environment and transmit the collected image data to the Jetson Nano. The Jetson Nano integrates the target detection algorithm Fast-YOLOv4, which can analyze the image data in real time. If there are UAV targets in the image, the algorithm will identify and locate the UAV targets. Finally, the Jetson Nano will send the abnormal results detected in real time to the workstation through a wireless connection. Administrators at the workstation can alert and respond to abnormal results. These are the process of real-time UAV target detection by the system.

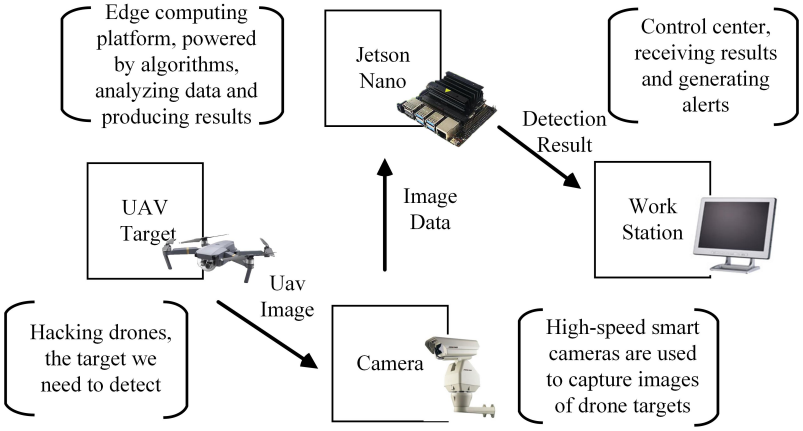

Figure 1. The object detection system model based on edge computing.

4. Algorithm Design

4.1. Fast-YOLOv4 Algorithm

This section mainly introduces the model architecture of Fast-YOLOv4, which is an improved algorithm based on YOLOv4. Figure 2 shows the architecture of Fast-YOLOv4 model and the structure of each module. The input image size of the model is $416 \times 416 \times 3$. The model architecture is mainly composed of MobileNet16, Multiscale-PANet, and YOLO-Head, including CBH module, MBN module, CBL module, and SPP module. CBH module is used as the initial layer of the backbone to process images, which is composed of a convolution block, normalization layer, and h-swish activation function. MBN module is a linear bottleneck inverted residual structure, which is mainly composed of a 1×1 convolution block, a 3×3 DW convolution block, and an h-swish activation function. When the input channel and output channel have the same number, the residual connection will be carried out to deepen the network. The CBL module is the basic module used for the detection and classification tasks, which is mainly composed of the convolution layer, normalization layer, and leaky ReLU activation function. The SPP module consists of the MaxPool layer and the full connection layer, which form a spatial pyramid pooling structure. This structure can obtain the same size feature vectors for any size of the input feature map, which can enhance the adaptability of the network to multiscale targets. MobileNet16 is stacked with CBH modules and MBN modules, with a total of 16 layers. Its function is to extract image features and output feature maps of different scales. Multiscale-PANet is

mainly built by the CBL module and SPP module. Its function is to fuse and superposition the input feature map and enrich the feature map information. YOLO-Head is mainly built by a CBL module and a convolution layer. Its function is to detect the location and class of the target and input it into the soft-merge algorithm to filter out the final result.

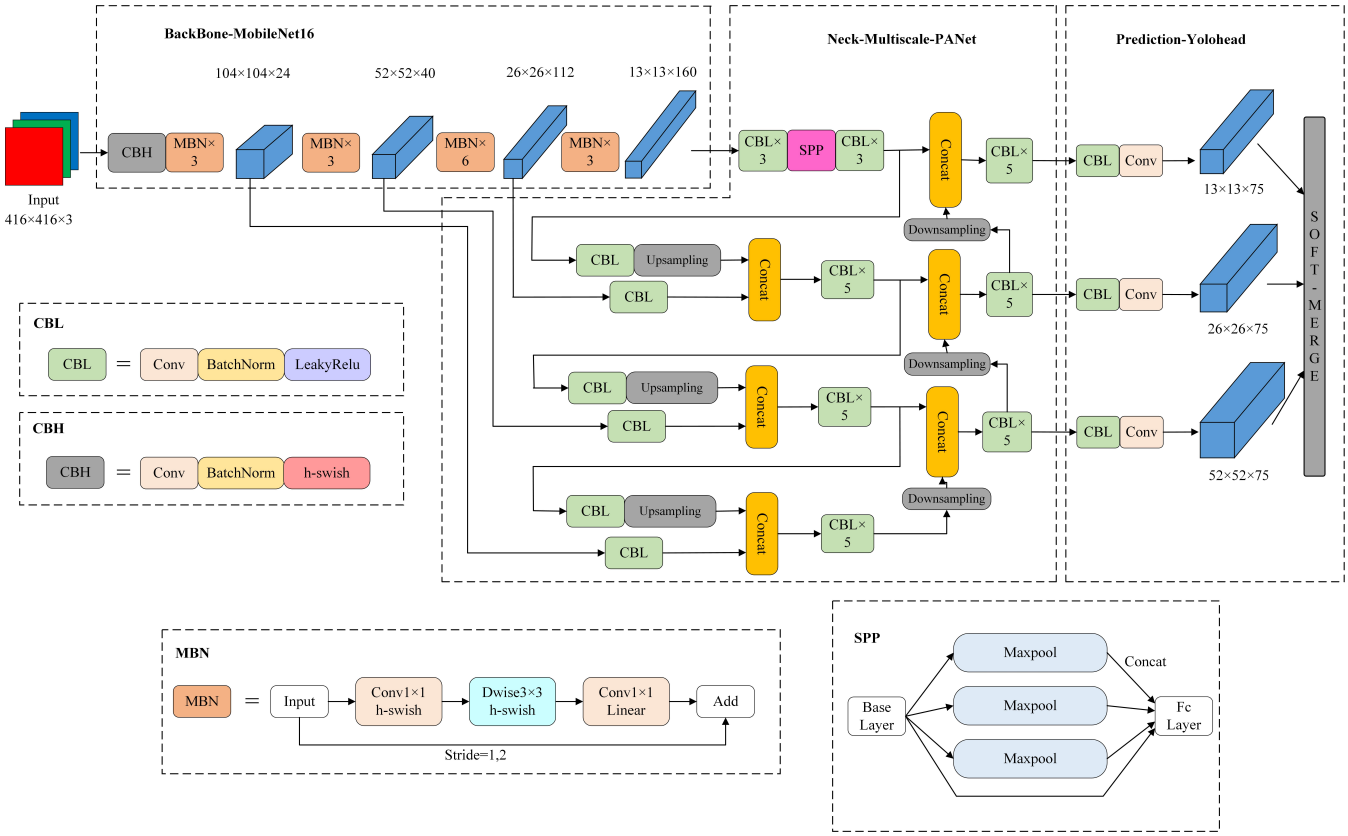

Figure 2. The architecture of the Fast-YOLOv4 model.

4.2. Lightweight Improvement of Backbone Network

The model inference speed determines the model detection speed. Therefore, reducing the number of model parameters and simplifying the network structure is the simplest way to speed up detection. Reducing the number of model parameters facilitates the integration of the algorithm into the edge computing platform and reduces energy consumption. Among the low-parameter models, the lightweight MobileNetV3 is a model with speed and accuracy advantages. MobileNetV3 is the third version of the lightweight network published by Google scientists such as Andrew Howard in 2019 [26]. It combines the deep separable convolution structure of V1 and the linear bottleneck inverse residual structure of V2 and introduces a new nonlinear activation function, h-swish, to save computational costs [27,28]. As a result, MobileNetV3 is a new lightweight network with higher performance.

Fast-YOLOv4 combines the lightweight structure of MobileNetV3 to transform the backbone network of YOLOv4. The backbone network uses the pruned MobileNetV3 as the starting part of the model, performs convolution operations on the input image, and extracts feature maps at four scales for feature fusion. The sizes of these feature maps are 104 × 104, 52 × 52, 26 × 26, and 13 × 13. These feature maps take into account the global and local features of various scale targets and provide sufficient information for subsequent classification and localization. The following Table 1 describes the backbone structure information of Fast-YOLOv4. In Table 1, the term "Operator" represents the module used at this layer and the size of the convolution kernel. The term "Exp size" represents the number of channels to be added to the MBN module. The term "Out size"

represents the number of output channels. The term "SE" represents whether to add an SE (squeeze-and-excite) structure to the depth-wise separable convolution block. The term "s" stands for step size.

Table 1. Table of the Fast-YOLOv4 backbone.

Floor	Input	Operator	Exp Size	Out Size	SE	s
1	$416^2 \times 3$	CBH, 3×3	-	16	-	2
2	$208^2 \times 16$	MBN, 3×3	16	16	-	1
3	$208^2 \times 16$	MBN, 3×3	64	24	-	2
4	$104^2 \times 24$	MBN, 3×3	72	24	-	1
5	$104^2 \times 24$	MBN, 5×5	72	40	✓	2
6	$52^2 \times 40$	MBN, 5×5	120	40	✓	1
7	$52^2 \times 40$	MBN, 5×5	120	40	✓	1
8	$52^2 \times 40$	MBN, 3×3	240	80	-	2
9	$26^2 \times 80$	MBN, 3×3	200	80	-	1
10	$26^2 \times 80$	MBN, 3×3	184	80	-	1
11	$26^2 \times 80$	MBN, 3×3	184	80	-	1
12	$26^2 \times 80$	MBN, 3×3	480	112	✓	1
13	$26^2 \times 112$	MBN, 3×3	672	112	✓	1
14	$26^2 \times 112$	MBN, 5×5	672	160	✓	2
15	$13^2 \times 160$	MBN, 5×5	960	160	✓	1
16	$13^2 \times 160$	MBN, 5×5	960	160	✓	1

The convolution layer used in the YOLOv4 structure is the standard 3×3 convolution, which will increase the number of model parameters and the computation time in the operation process. Therefore, Fast-YOLOv4 uses depth-wise separable convolution instead of the standard convolution. Depth-wise convolution is a convolution block proposed in MobileNetV1 to replace traditional convolution [27]. Its structure consists of a light depth-wise layer for spatial filtering and a heavy 1×1 point-wise layer for feature generation. Separating spatial filtering and feature generation greatly reduces the number of convolution operations and saves on computational costs. Figure 3 shows the structure of the standard convolution and depth-wise separable convolution.

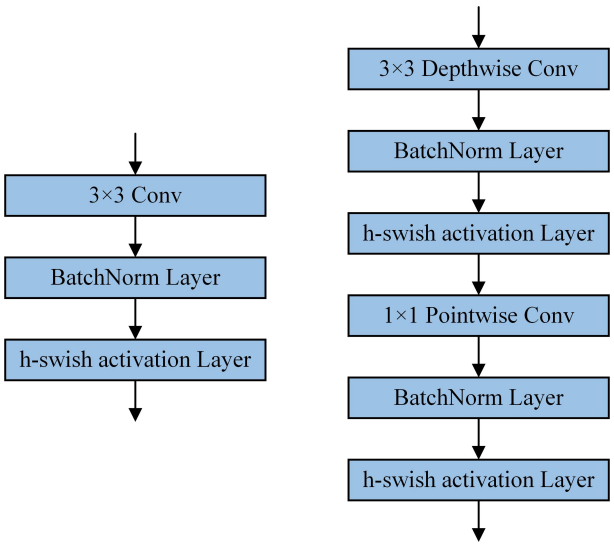

Figure 3. The structure of convolution.

The h-swish activation function is an approximate version of the swish activation function, which has a lower bound, no upper bound, and is not monotone, among other characteristics. The h-swish activation can make the neural network layer more expressive.

In order to adapt it to lightweight network use, the h-swish function is composed of common operators with small computational costs and is, thus, applicable to most frameworks. As a result, we use this function to improve the backbone network. The formula of the *h-swish* function can be written as:

$$h\text{-}swish[x] = x\frac{\mathrm{ReLU6}(x+3)}{6} \tag{1}$$

In Formula (1), x represents the value of the input activation function.

4.3. Multiscale Improvement of Neck Network

PANet is a path aggregation network applied to instance segmentation proposed by Shu [12]. Based on the top-down feature extraction of the FPN (feature pyramid network), PANet introduces a bottom-up path structure. Through feature fusion, the model can make full use of the local features of the shallow network and enrich the feature output of the deep network. An adaptive feature pooling layer is also added to the network to extract more features from the ROI (region of interest). Finally, the network uses the fully connected fusion layer to fuse the output of the convolution layer and the fully connected layer to obtain more accurate results.

The neck network of YOLOv4 adopts a PANet structure and extracts feature maps at three scales for fusion. This method enhances the network's ability of image feature extraction and integration but also results in losses of some features of small-scale targets. To strengthen the localization and feature extraction of small-scale targets, Fast-YOLOv4 improves the feature fusion structure of the neck network. Adding a feature output branch enriches the target feature extracted in the shallow network. By adding a feature output branch, the neck network can extract more target features from the shallow layer. Figure 4 shows the structure of the improved PANet.

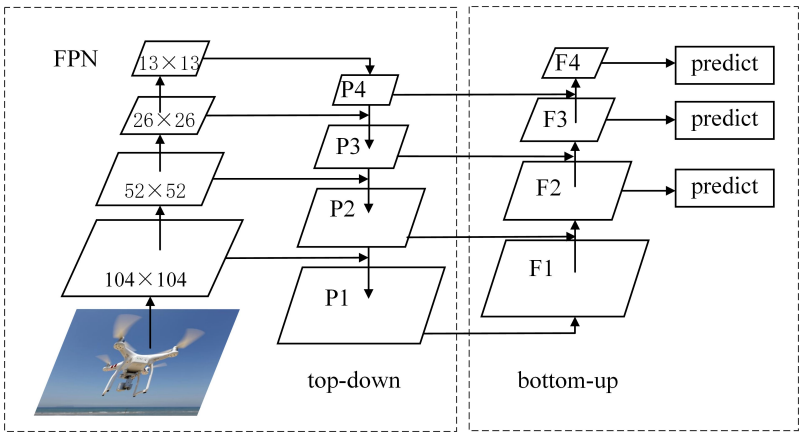

Figure 4. The diagram of the improved PANet structure.

4.4. Soft-Merge Algorithm

At present, NMS (non-maximum suppression) and Soft-NMS are commonly used in detection algorithms to filter predicted boxes [29]. These methods work well with a single model, but they can only filter relatively suitable predicted boxes and cannot effectively use the information of most predicted boxes. Different from NMS, Merge-NMS uses the confidence score and coordinate information of all predicted boxes to construct the fused predicted box, which is relatively closer to the real target boxes [21]. Figure 5 shows the difference in the results of Merge-NMS and NMS.

The operation of the traditional NMS is simple. In the filtering process, a predicted box larger than an IoU (intersection over union) threshold will be forcibly deleted, which may lead to missing detection or false detection in multi-UAV target scenarios. The IoU represents the area of the intersection ratio between two predicted bounding boxes, which

can be used to measure the degree of overlap between bounding boxes. Our improved soft-merge algorithm combines the idea of Soft-NMS weight assignment, can weight and fuse all predicted boxes. This approach can make full use of the coordinate and confidence information of the predicted boxes and reduce problems such as missing detection and false detection. The algorithm process of soft-merge can be outlined as Algorithm 1.

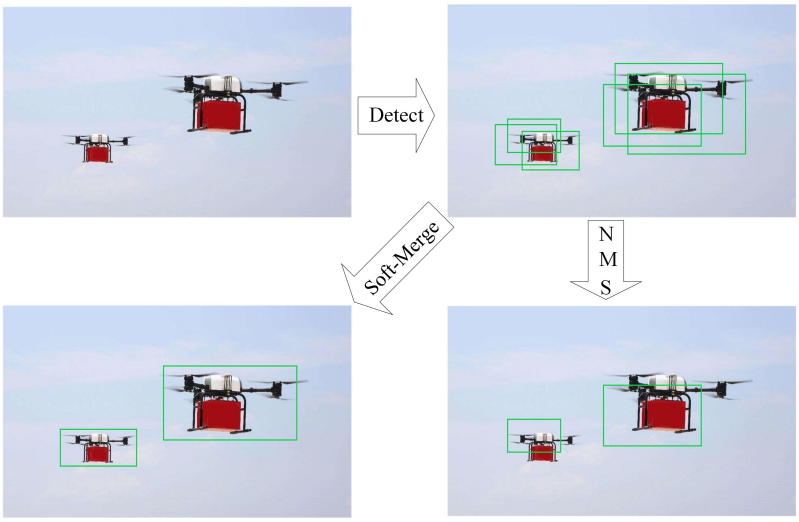

Figure 5. Comparison of Merge-NMS and NMS.

Algorithm 1: The algorithm process of soft-merge in Fast-YOLOv4

Input: Arrays $a_1, a_2, \ldots, a_m$, m represents the number of predicted boxes, a represents the information of predicted boxes, $a[confidence, X_1, X_2, Y_1, Y_2]$

Output: Array $F[f_1, f_2, \ldots, f_n]$, n represents the number of fusion boxes, and each f represents a fusion box result.

Assume that there are m predicted boxes in an image, and use soft-merge algorithm to obtain the fusion predicted box results. According to the confidence value of predicted box array a, all predicted boxes are sorted in descending order and added to array B to obtain $B[b_1, b_2, \ldots, b_m]$.;

for $i = 1$ *to* m **do**
  for $j = 1$ *to* n **do**
    if $IoU(a_i, f_j) >= Thre$ **then**
      Use the confidence update function for a_i and add it to the array L_j. L_j is the array corresponding to each fusion box in F, which is used to store the box to be fused;
    else
      $t \leftarrow t + 1$, t is used for counting, and the initial value is 0;
    end
  end
  if $t = n$ **then**
    $F[n + 1] \leftarrow a_i$;
    $n \leftarrow n + 1$;
  end
  for $k = 1$ *to* n **do**
    Use the weighted box fusion function to calculate the array L_k to obtain f_k;
    $F[k] \leftarrow f_k$;
  end
end

The formula of the confidence-updating function is:

$$s_i = \begin{cases} s_i, IoU < Thre \\ s_i e^{-\frac{iou(M,b_i)^2}{\sigma}}, IoU > Thre \end{cases} \quad (2)$$

In Formula (2), s_i represents the score of the prediction box i, $Thre$ represents the threshold, M represents the selected predicted box, and b_i represents other predicted boxes.

The formula of weighted box fusion is:

$$X1,2 = \frac{\sum_{i=1}^{T} s_i \times X1,2_i}{\sum_{i=1}^{T} s_i}, \quad (3)$$

$$Y1,2 = \frac{\sum_{i=1}^{T} s_i \times Y1,2_i}{\sum_{i=1}^{T} s_i} \quad (4)$$

In Formulas (3) and (4), s_i represents the score of the prediction frame i, T represents the number of predicted boxes participating in the fusion, $X1,2$ and $Y1,2$ represent the coordinates of the fused box, and $X1,2_i$ and $Y1,2_i$ represent the coordinates of the predicted box i. During the soft-merge calculation, the confidence scores of other boxes will be suppressed according to the IoU of the selected box. Boxes with high confidence are given a higher weight in the fusion formula, and the influence of the low-confidence boxes will be reduced in the final result.

5. Dataset and Experimental Preparation

5.1. Dataset Information

The datasets used in the experiment include the PASCAL VOC 07+12 dataset and UAV dataset. Firstly, the algorithm is trained on the VOC 07+12 dataset, and then the algorithm is trained and tested on the UAV dataset via transfer learning. The PASCAL VOC 07+12 dataset is a public competition dataset used by the PASCAL VOC Competition in 2007 and 2012. The dataset contains 20 classes of targets with a total of 21,504 images, of which 16,551 are training set images and 4952 are test set images for target detection. UAV dataset is constructed by combining the public dataset [30] and autonomously taking images of UAV targets. It includes three scales of large, medium, and small UAV targets, with 3698 images and 3719 targets. The division of the experimental dataset first randomly selects 20% of the images as the test set and then selects 90% of the remaining 80% of images as the training set and the rest as the validation set. Table 2 shows detailed information on the UAV dataset.

Table 2. The table of UAV dataset information.

Class	Number	Large	Medium	Small
train	2662	1621	547	509
validation	296	176	60	63
test	740	434	171	138
total	3698	2231	778	710

5.2. Experimental Environment and Hyperparameter

The experimental environment includes the workstation and the edge computing platform NVIDIA Jetson Nano. The workstation is used to train and validate the algorithm, and Jetson Nano is used to test the final performance of the algorithm. The configuration of the workstation is Intel (R) Core(TM) I9-10900F CPU @ 2.80 GHz 2.81 GHz 64 G memory based on Windows 10 operating system. The GPU is an NVIDIA Quadro P4000 with 8 G of video memory. The configuration of NVIDIA Jetson Nano is a 128-core CUDA Maxwell GPU, a quad-core ARM A57 1.43 GHz CPU with 4 GB LPDDR4 memory and 472 GFLOPS

of processing power. The deep learning framework used in the experiment is PyTorch 1.8.1, and the GPU acceleration library is CUDA10.2.

Hyperparameters are a set of parameters related to model training. They are also related to the length of training time and training effect. Table 3 shows the hyperparameter information of model training.

Table 3. The table of model training hyperparameter setting.

Type	Parameter	Note
Image size	416 × 416	Image input size
Epoch	200	Total training times
Batch size	16	Number of images per iteration
Learning rate1	0.01	Initial learning rate
Learning rate2	0.0001	Minimum learning rate
Momentum	0.937	Momentum of optimizer
Weight decay	0.0005	Decay of weights

5.3. Evaluation Metrics

The evaluation metrics of the experiment include object detection accuracy AP (average precision), mAP (mean average precision), IoU, detection speed FPS (frames per second), model parameters, and model volume size.

The mAP can comprehensively evaluate the localization and classification effect of the model for multi-class and multi-target tasks. Calculating the mAP requires calculating the AP for each class in the recognition task and then taking its average. The formula is as follows:

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C}, \tag{5}$$

In Formula (5), C represents the number of total classes, and AP_i represents the AP value of class i.

Calculating AP requires knowing the values of P (precision) and R (recall). The formulas for these three metrics are as follows:

$$P = \frac{TP}{TP + FP}, \tag{6}$$

$$R = \frac{TP}{TP + FN}, \tag{7}$$

$$AP = \int_0^1 P(R)dR, \tag{8}$$

In Formulas (6)–(8), TP (true positive) means that the input is a positive sample and the predicted result is also a positive sample; FP (false positive) means that the input is a negative sample and the predicted result is a positive sample; FN (false negative) means that the input is a positive sample and the prediction result is a negative sample; and TN (true negative) means that the input is a negative sample and the prediction result is a negative sample.

The IoU metric is used to calculate the ratio of the intersection and union of two bounding boxes. In essence, it converts the accuracy of object detection division into a comparison of the area between detection results and the true values. Under different IoU conditions, we can calculate different detection accuracies to comprehensively measure the accuracy of the model. Assuming that the areas of the two bounding boxes are A and B, the formula of the IoU metric can be written as:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{A \cap B}{A \cup B} \tag{9}$$

The FPS metric is the time that a model takes to detect a picture or the number of pictures detected in one second. The larger the FPS, the faster the model is detecting, which can be used to measure the detection speed of the model. The model parameters and model volume size are both metrics of model complexity. They all represent the size of the model, which can directly reflect the model size.

6. Experimental Results and Analysis

6.1. Performance Experiment on PASCAL VOC 07+12

To investigate the improvements of Fast-YOLOv4, we performed a comparative experiment on the PASCAL VOC 07+12 dataset. This dataset contains 20 categories of large-, medium-, and small-scale targets, which can comprehensively reflect the detection performance of the algorithm. The comparative algorithms used in the experiment include Faster R-CNN, YOLOv4, YOLOv3, SSD, YOLOv4-MobileNetV3, and Fast-YOLOv4. All algorithms will be trained and validated in the workstation. In addition, the metrics used in this experiment are mAP (IoU = 0.5), FPS, parameter, and volume. Table 4 shows the detection results of each algorithm on PASCAL VOC 07+12.

Table 4. The performance comparison of algorithms on PASCAL VOC 07+12 (in the workstation).

Model	Backbone	Input Size	mAP (%)	FPS (f/s)	Parameter ($\times 10^6$)	Volume (MB)
Faster R-CNN	ResNet50	416 × 416	77.02	12	137.10	522.99
YOLOv4	CSPDarkNet53	416 × 416	84.29	29	63.94	243.90
YOLOv3	DarkNet53	416 × 416	80.24	35	61.63	235.08
SSD	VggNet16	300 × 300	78.27	43	26.29	100.27
YOLOv4-MobileNetV3	MobileNetV3	416 × 416	79.75	60	11.30	43.51
Fast-YOLOv4	MobileNetV3	416 × 416	81.45	54	13.47	51.39

As can be seen from Table 4, in terms of detection accuracy, our approach has good performance compared with the other algorithms. Its mAP reaches 81.45%, second only to YOLOv4. Compared with YOLOv4-MobileNetV3 and YOLOv3, Fast-YOLOv4's accuracy increased by 1.7% and 1.21%, respectively. In terms of the detection speed, FPS, and model parameters, the performance of Fast-YOLOv4 is slightly worse than YOLOv4-MobileNetV3, but better than the other algorithms. The FPS metric reaches 55 f/s, and the model parameters and volume are only 13.47 M and 51.39 MB, which are better than the other models. In addition, compared with the low detection speed and the large number of parameters in YOLOv4 and Faster R-CNN, the proposed Fast-YOLOv4 has greater real-time detection performance and more lightweight parameters.

To illustrate the detection performance of the proposed algorithm on different categories of targets, we select Fast-YOLOv4 and YOLOv4 to compare their AP scores in six categories of targets in the PASCAL VOC 07+12 dataset. Figure 6 shows the comparison results.

It can be seen from Figure 6 that compared with YOLOv4, Fast-YOLOv4 improved the detection accuracy in most categories, especially for small-scale targets, such as potted plants, boats, and birds. The mAP has improved in these categories by 20%, 15%, and 7%, respectively. There are also some improvements in categories of medium-scale targets, such as dog, sheep, and cow, with increased mAPs of 8%, 5%, and 4%, respectively. This indicates that the proposed algorithm has enhanced the detection ability of small targets after adding scale fusion branches from shallow feature maps.

To sum up, Fast-YOLOv4 combines MobileNetV3's lightweight network, the improved Multiscale-PANet structure, and the soft-merge box fused algorithm. This combination greatly reduces the number of parameters in the model, speeds up the detection, and obtains good detection results on the PASCAL VOC 07+12 dataset, especially for small-scale targets. Compared with common one-stage and two-stage detection algorithms, such as YOLOv3, YOLOv4, SSD, and Faster R-CNN, Fast-YOLOv4 achieves better performance, reflecting the feasibility of the algorithm and its effectiveness in small-scale target detection.

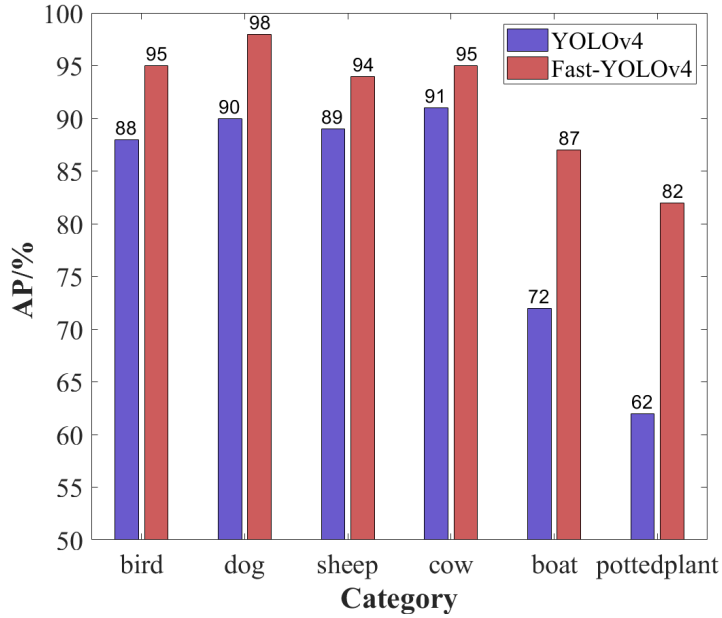

Figure 6. The comparison of AP of different categories.

6.2. Performance Experiment on the UAV Dataset

To demonstrate the effectiveness of Fast-YOLOv4 in UAV target detection, we conduct an experiment on the UAV dataset using Faster R-CNN, SSD, YOLOv3, YOLOv4, YOLOv4-MobileNetV3, and Fast-YOLOv4. All algorithms are trained and validated in the workstation. The metrics used in this experiment are mAP (IoU = 0.5), AP_S, AP_M, and AP_L (IoU = 0.5:0.95), where AP_S, AP_M, and AP_L are the average accuracies of the three scales of the targets. Table 5 shows the detection results of each algorithm on the UAV dataset. Figure 7 shows the comparison of the partial detection results of YOLOv4 and Fast-YOLOv4.

Table 5. The performance comparison of algorithms on the UAV dataset (in the workstation).

Model	mAP	AP_S	AP_M	AP_L
Faster R-CNN	88.12	12.89	38.06	61.68
YOLOv4	90.18	20.70	39.03	57.28
YOLOv3	88.92	17.93	33.18	53.41
SSD	86.88	8.36	38.82	65.50
YOLOv4-MobileNetV3	88.81	16.73	34.92	55.33
Fast-YOLOv4	90.62	23.88	37.53	59.71

From the experimental results in Table 5, the mAP (IOU = 0.5) of the proposed approach reaches 90.62%, which is superior to the other algorithms. Meanwhile, Fast-YOLOv4 performs well on the AP accuracy metrics of three scales, with the accuracy of small-scale targets to large-scale targets being 23.88%, 37.53%, and 59.71%, respectively. In particular, compared with YOLOv4 and YOLOv4-MobileNetV3, the APs of small-scale targets have increased by 3.18% and 7.15%, respectively, reflecting improvements in the multiscale structure and soft-merge brought to small target detection. Furthermore, our approach performs fairly well in medium-scale target detection. Compared with YOLOv3 and YOLOv4-MobileNetV3, Fast-YOLOv4 has a certain improvement. However, due to model size limitations, the AP is slightly inferior to YOLOv4, SSD, and Faster R-CNN. In large-scale target detection, our approach performs slightly worse than SSD and Faster R-CNN. The main reason is that Fast-YOLOv4 has a big gap in model volume compared with the other algorithms, and the large-scale model has more advantages in feature extraction and feature fusion, so it performs better in large-scale target detection.

Moreover, it can be seen from Figure 7 that Fast-YOLOv4 detected more small-scale UAV targets than the other models. It also reduces the number of missing detection and false detection, which shows that our approach has better detection performance than the original YOLOv4. In view of all performance metrics, the proposed Fast-YOLOv4 model combines the advantages of high accuracy, fast speed, and fewer model parameters, making it very suitable for integration with edge computing platforms for improving UAV target detection.

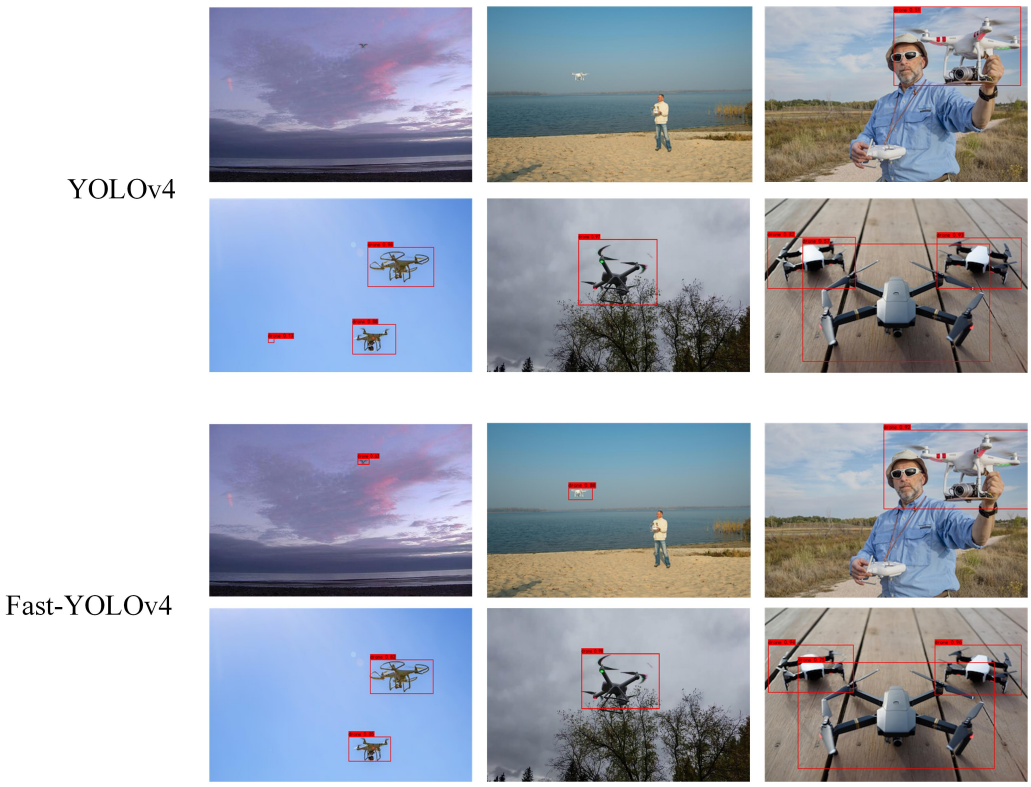

Figure 7. The partial detection results comparison of YOLOv4 and Fast-YOLOv4.

6.3. Ablation Experiment

To directly observe the improvement of Fast-YOLOv4, we conduct an ablation experiment in the workstation platform. The experimental method will train and validate each improved part of Fast-YOLOv4 on the UAV dataset and record its mAP (IoU = 0.5), APS, APM, APL (IoU = 0.5:0.95), FPS, and model parameters to check the improvement effect. Table 6 shows the performance results of the original YOLOv4 and Fast-YOLOv4 improvements.

From the experimental results in Table 6, improvement can be seen in each step of Fast-YOLOv4. Among all improvements, the lightweight network based on the backbone has significantly improved the performance of the model. Compared with YOLOv4, the YOLOv4-MobileNetV3 model improves the FPS to 60 f/s and reduces the number of parameters to 11.30M at the cost of only a 1.37% reduction in mAP. Multiscale-PANet improves the mAP and APS of the model by 1.37% and 4.05%, respectively, but the FPS decreases by 3 f/s, and the number of parameters increases by 3.64M. The soft-merge method improves the mAP of the YOLOv4 model by 0.86%, and other AP metrics are improved to varying degrees; however, the FPS decreases by 2 f/s. Finally, the proposed approach, Fast-YOLOv4, which combines these three methods, achieves an mAP of 90.62%, an APS of 23.88%, an FPS of 54 f/s, and 11.54M parameters. Compared with YOLOv4, this is an overall improvement and reflects the success of the improved method in UAV target detection tasks.

Table 6. The performance comparison of YOLOv4 and Fast-YOLOv4's improvements based on the UAV dataset (in the workstation).

Model	mAP	AP_{Small}	AP_{Medium}	AP_{Large}	FPS	Parameter
YOLOv4	90.18	20.70	39.03	57.28	29	63.94
YOLOv4-MobileNetV3	88.81	16.73	34.92	55.33	60	11.30
YOLOv4-Multiscale	91.55	24.75	39.40	56.19	26	64.89
YOLOv4-Soft-Merge	90.73	20.94	39.66	59.76	27	63.94
Fast-YOLOv4	90.62	23.88	37.53	59.71	54	11.54

6.4. Edge Computing Application

To investigate the real-time detection performance of Fast-YOLOv4 in the edge computing platform, all detection models are trained in the Quadro P4000 GPU and integrated into the Jetson Nano edge platform for target detection of UAVs in a video. The experiment tests the real-time detection effect of the model by comparing the speed and confidence of detection of each model in the edge platform. Meanwhile, we monitor the CPU temperature and memory footprint of the embedded platform to observe their changes when running different algorithms. Figure 8 shows the running results of different algorithms in the Jetson Nano. Figure 9 shows the comparison of Fast-YOLOv4's and YOLOv4's performances in detecting UAV targets in the videos.

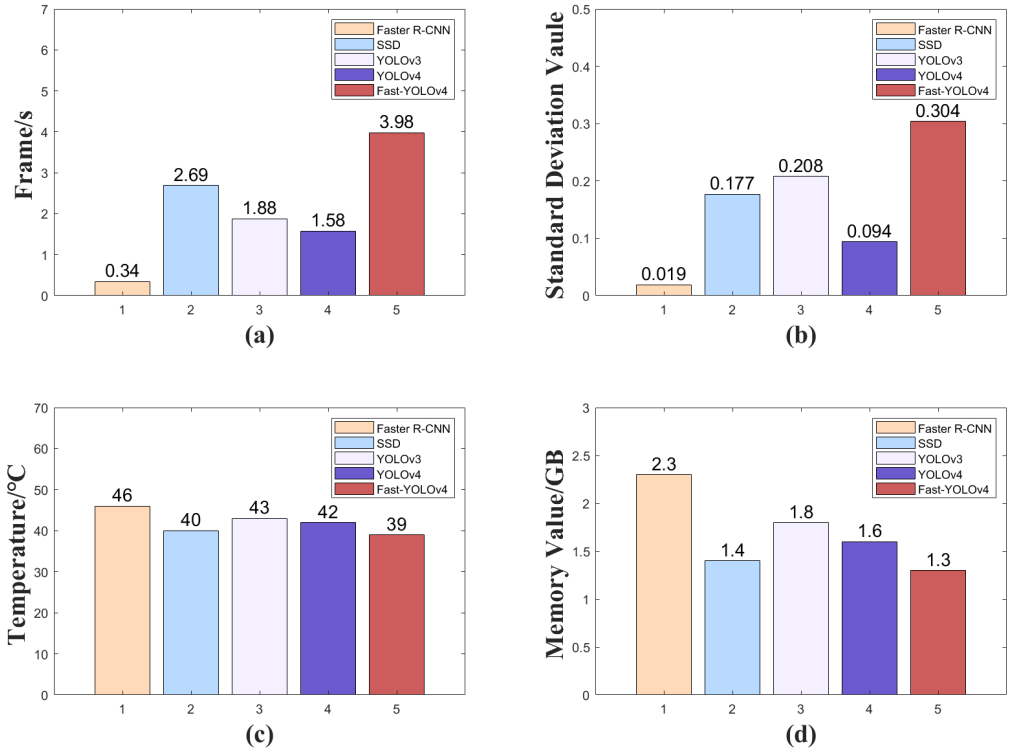

Figure 8. The comparison of running results of different algorithms (in the Jetson Nano). (**a**) The first panel is the average FPS. (**b**) The second panel is the standard deviation of FPS. (**c**) The third panel is CPU temperature. (**d**) The fourth panel is memory footprint.

In Figure 8, we select the average FPS, the standard deviation of FPS, CPU temperature, and memory footprint metrics to compare the algorithms. It can be seen that the average FPS of Fast-YOLOv4 is 3.98 f/s, which is the best result in all algorithms. In CPU temperature and memory footprint metrics, Fast-YOLOv4 shows lower heat dissipation requirements and memory usage. Although Fast-YOLOv4's FPS stability is slightly inadequate, its performance in Jetson nano is better than other algorithms on the whole. In addition, Figure 9 also shows the detection effect of YOLOv4 and Fast-YOLOv4 on UAV videos. In

Figure 9, the FPS of the improved model is 2.5 times higher than that of YOLOv4, and the confidence is comparable. Compared with other algorithms, our approach greatly improves the detection speed and accuracy. In terms of the results, the application results show that Fast-YOLOv4 has good performance in the edge computing platform and can achieve high-precision detection in real time.

Figure 9. The comparison of YOLOv4 and Fast-YOLOv4 detection results (in the Jetson Nano).

7. Conclusions

Aiming to address the problems in the detection of the multiscale UAV target, we present a novel real-time target detection algorithm called Fast-YOLOv4 based on edge computing. The proposed algorithm is based on YOLOv4 and uses the lightweight network MobileNetV3 to improve the backbone, which can reduce structural complexity and model parameters to speed up detection. Then, our approach combines the Multiscale-PANet to enhance the use and extraction of shallow features, strengthen the detection of small-scale targets, and improve the model's accuracy. In view of missing detection and false detection of multiscale targets, the proposed approach adopts an improved weighted boxes fusion algorithm, called soft-merge, to perform the weighted fusion of the predicted boxes instead of the traditional NMS. Finally, Fast-YOLOv4 achieves the best results in the fast detection of UAV targets in the edge computing platform Jetson Nano. To summarize, Fast-YOLOv4 provides a practical and feasible research method for real-time UAV target detection based on edge computing, which is worthy of further research.

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

Funding: This research was funded by the Theoretical Research Project (KY20S011), and the Fusion Special Project of Anhui Province (KY21C008), China.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Acknowledgments: The author would like to thank all contributors to this study.

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

References

1. Yuan, C.X.; Li, G.; Su, D.C. Development status of anti-low-slow and small UAV systems. *Electron. World* **2021**, *23*, 138–140. (In Chinese) [CrossRef]
2. Jiang, R.Q.; Bai, R.K.; Peng, Y.P. Review of low and slow small UAV target detection technology. *Maneuverable Missile* **2020**, *49*, 100–105. (In Chinese) [CrossRef]
3. Mendis, G.J.; Randeny, T.; Wei, J.; Madanayake, A. Deep learning based doppler radar for micro UAS detection and classification. In Proceedings of the Military Communications Conference, Baltimore, MD, USA, 1–3 November 2016; pp. 924–929. MILCOM.2016.7795448. [CrossRef]
4. Bisio, I.; Garibotto, C.; Lavagetto, F.; Sciarrone, A.; Zappatore, S. Unauthorized Amateur UAV Detection Based on WiFi Statistical Fingerprint Analysis. *IEEE Commun. Mag.* **2017**, *56*, 106–111. [CrossRef]
5. Bougaiov, N.; Danik, Y. Hough Transform for UAV's Acoustic Signals Detection. *Adv. Sci. J.* **2015**, *6*, 65–68. [CrossRef]
6. Briese, C.; Guenther, L. Deep learning with semi-synthetic training images for detection of non-cooperative UAVs. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 981–988. [CrossRef]
7. Mejias, L.; McFadyen, A.; Ford, J J. Sense and avoid technology developments at Queensland University of Technology. *IEEE Aerosp. Electron. Syst. Mag.* **2016**, *31*, 28–37. [CrossRef]
8. Xie, F.; Zhu, D J. Survey on Deep Learning Object Detection. *Comput. Syst. Appl.* **2022**, *2*, 1–12. . (In Chinese) [CrossRef]
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
10. 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; pp. 21–37. [CrossRef]
11. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020.
12. 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, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768. [CrossRef]
13. Ai, Y.; Peng, M.; Zhang, K. Edge computing technologies for internet of things: A primer. *Digit. Commun. Netw.* **2018**, *2*, 77–86. [CrossRef]
14. Lin, T.; 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. [CrossRef]
15. Gao, S.; Cheng, M.M.; Zhao, K.; Zhang, X.Y.; Yang, M.H.; Torr, P. Res2net: A new multiscale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 652–662. [CrossRef] [PubMed]
16. Zeng, Z.; Wang, Z.; Qin, L.; Li, H. Drone Detection Based on Multi-scale Feature Fusion. In Proceedings of the 2021 International Conference on UK-China Emerging Technologies (UCET), Chengdu, China, 4–6 November 2021; pp. 194–198. [CrossRef]
17. Zhu, J.; Wang, J.L.; Wang B. Lightweight mask detection algorithm based on improved YOLOv4-tiny. *Chin. J. Liq. Cryst. Disp.* **2021**, *36* 1525–1534. (In Chinese) [CrossRef]
18. Zhang, C.; Zhang, C.; Wang, H.; He, Q.; Liu, Y. Research on non-maximum suppression based on attention mechanism in object detection. *Electron. Meas. Technol.* **2021**, *44*, 82–88. (In Chinese) [CrossRef]
19. Ning, C.; Zhou, H.; Song, Y.; Tang, J. Inception single shot multibox detector for object detection. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), Hong Kong, China, 10–14 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 549–554. [CrossRef]
20. Solovyev, R.; Wang, W.; Gabruseva, T. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image Vis. Comput.* **2021**, *107*, 104117. [CrossRef]
21. Zhou, L. Piao, J. A Lightweight YOLOv4 Based SAR Image Ship Detection. In Proceedings of the 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 13–15 August 2021; pp. 28–31. [CrossRef]
22. Liu, H.; Fan, K.; Ouyang, Q.; Li, N. Real-Time Small Drones Detection Based on Pruned YOLOv4. *Sensors* **2021**, *21*, 3374. [CrossRef] [PubMed]
23. Xun, D.T.W.; Lim, Y.L.; Srigrarom, S. Drone detection using YOLOv3 with transfer learning on NVIDIA Jetson TX2. In Proceedings of the 2021 Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 20–22 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
24. Carrio, A.; Tordesillas, J.; Vemprala, S.; Saripalli, S.; Campoy, P.; How, J.P. Onboard detection and localization of drones using depth maps. *IEEE Access* **2020**, *8*, 30480–30490. [CrossRef]
25. Opromolla, R.; Inchingolo, G.; Fasano, G. Airborne visual detection and tracking of cooperative UAVs exploiting deep learning. *Sensors* **2019**, *19*, 4332. [CrossRef] [PubMed]

26. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 1314–1324. [CrossRef]
27. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Wey, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
28. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510—4520. [CrossRef]
29. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Improving object detection with one line of code. *arXiv* **2017**, arXiv:1704.04503.
30. Hao, Y.J.; Teck, L.K.; Xiang, C.Y.; Jeevanraj, E.; Srigrarom, S. Fast Drone Detection using SSD and YoloV3. In Proceedings of the 2021 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Repulic of Korea, 12–15 October 2021; pp. 1172–1179. [CrossRef]