Unmanned Systems, Vol. 10, No. 1 (2022) 57–67

DOI: 10.1142/S2301385022500030

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us

Low-Latency Aerial Images Object Detection for UAV

Kai Feng[*,‡], Weixing Li[*,§], Jun Han[*,¶], Feng Pan[*,†,‖]

[*]*School of Automation, Beijing Institute of Technology, Beijing 100081, P. R. China*

[†]*Kunming-BIT Industry Technology Research Institute Co. Ltd, Kunming 650106, P. R. China*

Visual-based object detection has large applications in security and surveillance for unmanned aerial vehicles (UAVs). Meanwhile, the object detectors are required of low-latency and easy to be deployed on embedded onboard platforms. Aiming to address these problems, we present a PA-YOLOv3 aerial images object detector based on YOLOv3 and PANet algorithms, which can be deployed on embedded platforms. The PA-YOLOv3 model uses the dual-tower structure to improve the feature extraction and expression capabilities in feature fusion stage of the network. Besides, we propose a balanced pruning method to reduce the model size and the imbalance of different feature layers during pruning. After balanced pruning, the latency and size of the model are significantly decreased. Finally, we deploy and quantify the model on the embedded platform with TensorRT technology and apply the model on the UAV system for testing. The comprehensive experiments are executed on VisDrone2018 dataset and real-world scenarios. The experimental results show the inference speed of PA-YOLOv3 boost of about 10 × after model pruning and quantization, while maintaining high detection accuracy.

Keywords: Object detection; model pruning; embedded platform deployment; Unmanned Aerial Vehicle (UAV).

US

1. Introduction

Intelligent UAVs are the ideal devices for exploring complex environments and enabling unmanned operations, such as unmanned search and rescue, infrastructure inspections and military, etc. [1, 2]. Visual detection is essential for UAVs to complete the above tasks. Equipping with vision sensors to complete object detection is one of the important requirements of intelligent UAVs. For instance, combining with visual object detection can enable UAV to achieve autonomous navigation and following, using object detection can achieve unmanned search, rescue, security, etc. But UAV requires lower latency when implementing object detection and navigation control. Therefore, it is an urgent requirement for the deploying of detection algorithms on embedded platforms to reduce latency in practical applications of UAV.

Existing deep learning object detection algorithms have provided many high-precision object detectors, which can detect specific categories of objects in natural scenes (such as the 80 categories defined by the MS COCO [3] benchmark dataset). However, aerial objects and natural scene objects have large differences in sizes and shooting angles. Therefore, it is impossible to directly use the existing high-precision detection models, thus training on an aerial object dataset is required. At the same time, deep convolutional neural networks require powerful computing capabilities, due to a large number of computing resources consumed in model inference. It makes many high-performance detectors unusable on embedded platforms with limited resources. Therefore, it is very necessary to improve the inference speed of the model on embedded platforms.

Based on the above discussion, we focus on the aerial object detection for UAV and implement a low-latency

Received 23 September 2020; Revised 12 April 2021; Accepted 12 April 2021; Published 20 May 2021. This paper was recommended for publication in its revised form by editorial board member, Zhi Gao.
Email Address: [‡]*3120180881@bit.edu.cn*, [§]*liweixing@bit.edu.cn*, [¶]*3120190878@bit.edu.cn*, [‖]*andropanfeng@126.com*
[§]Corresponding author.

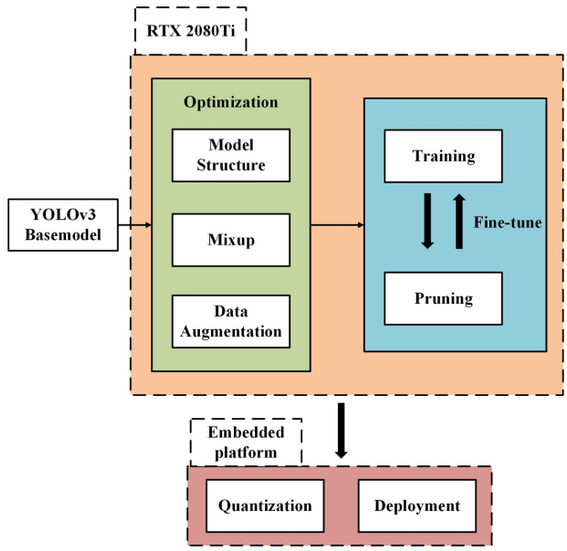

Fig. 1. Algorithm framework.

detection method which maintains high detection accuracy. The main contributions of this paper are as follows:

- Proposing a PA-YOLOv3 model. This model can enhance the feature extraction capabilities. Meanwhile, we modify the prediction layer to improve the detection accuracy of small objects by a down-sampling factor of 4, 8 and 16, respectively.
- Presenting a balanced pruning method. This method reduces the model size and the imbalance of different feature layers during pruning. After model pruning, the model size and latency have decreased significantly, which improves the inference speed of the detection model.
- The deployment of the detection model on three different performance embedded platforms: Jetson Nano, Jetson TX2 and Jetson Xavier. We build a UAV system for real-world scenarios testing and utilize TensorRT to complete model quantification and deployment. The experimental results achieve the minimum latency of the detection model less than 15 ms, which greatly improves the inference speed and meets the real-time detection requirements. The algorithm framework of the detection system is shown in Fig. 1.

2. Related Work

2.1. *CNN-based object detection method*

Object detection algorithms based on deep learning can roughly be divided into two categories: the two-stage method and the one-stage method. R-CNN [4] is the first object detection algorithm based on deep learning. After that, researchers proposed a series of methods based on the R-CNN model to improve detection accuracy and speed, such as SPP-Net [5], Fast R-CNN [6], Faster R-CNN [7], etc. Faster R-CNN proposes a Region Proposal Network (RPN), which can replace the selective search [8] and quickly generate candidate regions. Lin *et al.* proposed the Feature Pyramid Network (FPN) [9] to improve the accuracy of small objects detection. Based on faster R-CNN, FPN uses the high resolution of low-level features and the semantic information of high-level features to improve the detection accuracy of a small object, but it also inevitably computation cost.

The two-stage object detection algorithm has achieved high detection accuracy on large-scale natural scene datasets (such as MS COCO) through twice classification and regression operations, but the detection speed is generally not real-time. Therefore, the one-stage object detection algorithm abandons the idea of the two-stage detection and completes the object detection through only one-time classification and regression. Such as YOLO [10–12], SSD [13], RefineDet [14], etc. The one-stage object detection algorithms achieved the real-time detection speed on GPU, but the detection accuracy still trails that of the two-stage methods. Lin [15] modified the traditional cross-entropy classification loss function to balance the foreground and background during training and achieved an improvement in detection accuracy.

2.2. *Object detection for UAV*

Visual perception is one of the necessary capabilities for intelligent UAVs. Many researchers study the visual perception systems of UAVs. Jung *et al.* [16] proposed the ADR-Net detection model to perform visual navigation and detect obstacles ahead for UAV. They replaced the VGG16 backbone network of SSD with a lightweight model, which improved the detection speed. Hsieh *et al.* [17] proposed the LPN for vehicle counting based on UAV. They increase the detection accuracy of dense vehicles by adding the spatial layout score, which makes vehicles surrounded by more objects have higher confidence. Cui *et al.* [18] used contour detection and color threshold to complete marker and digital detection, contributing to indoor navigation task for UAVs. Zhang *et al.* [19] used the Faster R-CNN method to achieve traffic flow monitoring. Compared with traditional methods, using UAV to complete statistics is more flexible. In summary, the deep learning object detection for UAV brings higher detection accuracy, but the detection speed also needs to be considered in practical applications. In this paper, we adopt a one-stage detector with high detection accuracy, and improve the speed of the network through

model pruning and quantification, while maintaining a balance between detection speed and accuracy.

2.3. *Model pruning and quantification*

In embedded platforms with limited resources, it is very difficult to run a deep convolutional neural network with huge parameters. Therefore, it is necessary to compress the number of model calculations. Model pruning and quantification are two effective methods to achieve the model compress. Many researchers have made important contributions to model pruning and quantification. Denton *et al.* [20] proposed a model parameter representation method using the singular value decomposition to perform low-rank decomposition on fully connected layers, yielding $\sim$3x model-size compression. However, since the calculations in deep networks are mostly convolutional layers, the model inference speed has not decreased significantly. Rastegari *et al.* [21] reduced the weights to binary or ternary, and restricted the weights to $\{-1, 1\}$ or $\{-1, 0, 1\}$. This weight quantization method reduces the model size and improves the model inference speed, but the low-bit approximation method usually comes with a large loss of accuracy. Han *et al.* [22] pruned some connections with a small weight in the trained network so that the weight of the final network was mostly zero thus the storage space can reduce by storing the network in a sparse format. Liu *et al.* [23] added the scale factor of the network channel to the sparsity training and then pruned the network through the global threshold. This method directly pruned the network structure and not used additional dedicated libraries and hardware. We are inspired by [23] and extend it to the object detection method.

In addition to the model pruning, model quantification is also an important method to improve the inference speed of the network. Gao *et al.* [24–26] discussed the application of float, half-float and 8-bit integer quantization techniques in model quantization. Nvidia TeneorRT [26] uses FP32, FP16 and INT8 to quantify the model, which can significantly reduce the model's float-point-operations (FLOPs) and increase the inference speed of the network. Besides, TensorRT uses the layer fusion and tensor fusion to merge the convolutional layer, BN [27] layer, and the activation layer into one structure, which reduces the CUDA core occupancy rate.

3. Method

3.1. *Network design*

In aerial images, the size of an object is relatively small due to the flying height of the UAV, so the detector needs to have stronger location and classification performance for small objects. In the deep convolutional neural network, the receptive field of the high-level feature layer is large, it responds strongly to the entire object and has good semantic information. The low-level feature layer has more local features suitable for localization and has more information about the small object. Based on this experience, it is an intuitive idea to fuse the information between different feature layers to improve the location and classification capabilities of small object detection. Through the top-down and bottom-up horizontal connection structure like [28], we combine the dual-tower structure with YOLOv3 to further improve the feature extraction capability of small aerial objects. So that each layer of the network has high-level semantic information and low-level location information. The network framework of PA-YOLOv3 is shown in Fig. 2.

PA-YOLOv3 is designed with adding top-down and bottom-up horizontal connection structures for feature fusion. We use Darknet-53 as the backbone and use $\{C_2, C_3, C_4, C_5\}$ to construct a dual-tower structure. The top-down FPN structure is formed by up-sampling and horizontal connection and the bottom-up structure is formed by a convolutional layer stride of 2 with 3×3 kernel size. We use the shortcut operation to fuse the layers of horizontal connections. Then a convolutional layer with 1×1 kernel size is used to change the output channel. The down-sampling factor is 32 of the top feature layer, which makes it difficult to predict small size objects effectively on this feature. Considering this situation, PA-YOLOv3 uses three feature maps $\{N_2, N_3, N_4\}$ with down-sampling factors of 4, 8 and 16 as the detection feature maps. We can obtain the classification and regression results with the added YOLO layer. Benefiting from the optimization of the network structure, PA-YOLOv3 has a much more satisfying ability to detect and locate small objects.

3.2. *Training*

The scenes of aerial images usually have many objects, such as dense vehicles on the road and pedestrians on the street. To improve the detection effect on dense objects, we add an anchor to each of prediction layer in PA-YOLOv3, so that each prediction layer has four preset anchors, which is enhanced the matching effect of the detector on dense objects. Before training PA-YOLOv3, the dataset labels clustered by the Kmeans algorithm to form 12 initial anchors with different sizes and aspect ratios, so that the detector can adapt to different sizes of aerial objects. During training, we use data augmentation to adapt the changes of geometry and lighting of objects, such as random image flipping, random adjustment of saturation, exposure and hue of the image.

We adopt the mixup method [29] to enhance the robustness of the detector and prevent overfitting. The key

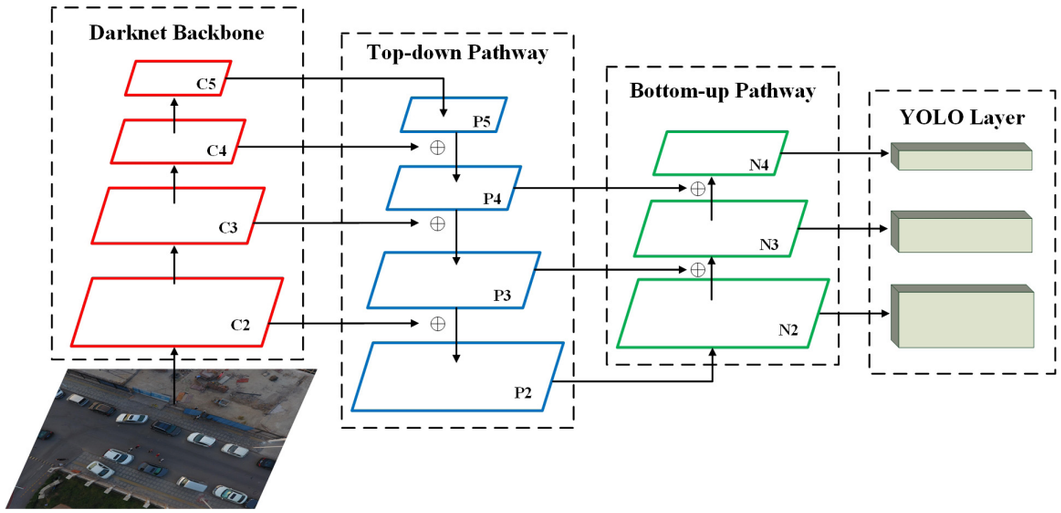

Fig. 2. Architecture of PA-YOLOv3.

idea of mixup is to regularize the model with a simple linear behavior by mixing up pixels as interpolations between pairs of training images. Meanwhile, the one-hot image labels are mixed up using the same ratio. Through the operation of the linear input and the linear output, the model moves toward the linear function during training, thus reducing the nonlinear overfitting ability of the deep convolutional neural network and improving the generalization ability of the network. We refer to the settings in [30] and add mixup to PA-YOLOv3 to enhance the robustness of the model.

3.3. *Pruning and quantization*

Model pruning is a method to achieve model compression. Liu *et al.* [23] proposed a channel scale factor γ to determine the importance of a layer of the model. They set a global pruning scale threshold τ according to the channel scale factors of all layers, and the convolutional layer with the scale factor γ below the global threshold τ will be pruned. This is a simple but effective model pruning method without adding any training costs and hardware.

Based on the research in [23], we extend the model pruning method into PA-YOLOv3. PA-YOLOv3 adds a batch normalization layer after each convolutional layer to speed up the model convergence. The expression form of the BN layer is shown in the following equation:

$$y = \gamma \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta, \tag{1}$$

where B is the current mini-batch, μ_B and σ_B are the mean and standard deviation values of the input activation over B, x and y are the inputs and outputs of the BN layer, and γ, β are the learnable scale factors and offsets factor, respectively. ε is a small positive number that prevents the denominator from being 0. Because the scale factor in the BN layer describes the mapping importance of a certain layer, the channel scale factor γ in the model pruning is replaced by the scale factor in the BN layer.

In [23], by setting the global pruning threshold to control the pruning of the model, it is easy to cut most of the low-level convolutional layers of the model, and keep most of the high-level convolutional layers. Since the high-level feature layers have more semantic information, and the image classification network only uses the top feature layer for classification, so the global pruning method has achieved good results in the image classification task. But the object detection task not only need the classification information of the high-level feature layers, but also need to add the local positioning information of the low-level feature layers. Therefore, the global pruning makes most of the low-level convolutional layers cropped, leading to the low-level information that cannot be better retained. Furthermore, the performance of the feature fusion will decline, and the detection accuracy of the model will be damaged.

In PA-YOLOv3, the dual-tower structure fully integrates the high-level classification information and the low-level location information. We propose a balance pruning method to reduce the loss of the low-level features by global pruning. In balance pruning, the scale factors of the BN layer of the same down-sampling module are sorted, and then we prune the model in different down-sampling modules. So, the pruning ratios of the feature layers with different down-sampling ratios are guaranteed to be the same, and the problem of global pruning is alleviated, which

is beneficial for the feature fusion stage of the dual-tower module of PA-YOLOv3.

In the balance pruning of the PA-YOLOv3, the up-sample layer, pooling layer and the YOLO prediction layer are ignored, and the other convolutional layers are pruned according to the balance pruning threshold τ. To keep the connection relationship between the layers of the network, a local minimum pruning threshold is set to prevent over-pruning. The route layer and the shortcut layer in PA-YOLOv3 are specially processed. The route layer is pruned after concatenating operation, and the number of shortcut layer selects the minimum number of the two pruned convolution layers when concatenated. After model pruning, the model needs to be fine-tuned on the dataset to compensate for temporary degradation, and the accuracy will gradually recover until it is stable. In the quantification stage, we use TensorRT to quantify the model to improve the inference speed. The TensorRT framework supports model quantization in three modes, namely Float32, Float16 and INT8. We deploy detection models on three embedded platforms: Jetson Nano, Jetson TX2 and Jetson Xavier and uses TensorRT to perform inference acceleration and reduce the model's consumption.

4. Experiments

4.1. *Experiment settings*

Training data. The experimental training data uses VisDrone2018 [31] aerial object detection benchmark dataset, which contains about 7000 images and 11 categories. To compare with the existing categories in MS COCO dataset, we select eight sub-categories in VisDrone2018 as training categories, including people, pedestrian, bicycle, car, van, truck, bus, motorbike. We remap people and pedestrian labels to person and van labels to the car to match the same subcategory of the MS COCO dataset.

Training parameters. We use darknet framework to train PA-YOLOv3. Since the MS COCO dataset contains 6 categories of the remapped VisDrone2018, the PA-YOLOv3 uses the pre-trained model that has been trained on the MS COCO dataset to initialize and then trained on the VisDrone2018 dataset. All experiments were trained and tested on an RTX 2080Ti. The number of model training iterations is 20k, minibatch is 64, the initial learning rate is 0.001 and decreases by 0.1 when the number of iterations is 16k and 18k. The model optimization method is Stochastic Gradient Descent (SGD), where weight decay is 0.0005 and momentum is 0.9. To maintain a similar ratio to the dataset image (16/9), the width and height of the input image are resized during training. The $w \in [480, 960]$, $h \in [288, 544]$, where width and height are multiples of 32.

4.2. *Experiment study*

According to the dataset and training parameters in 4.1, we train YOLOv3 and PA-YOLOv3 on the VisDrone2018 dataset, respectively. The training results were evaluated on the Visdrone2018 validation set. The evaluation metric is mean average accuracy (%mAP, IoU $= 0.5$). We also test the YOLOv3 model that trained on the MS COCO dataset, and compare it with the model pre-trained on MS COCO and then perform on Visdrone2018. Table 1 shows the experiments results of PA-YOLOv3.

Fine-tone Result. As can be seen from Table 1, YOLOv3 has poor detection accuracy on the VisDrone2018 without transfer training (i.e. 24.51%), especially the categories of bus, bicycle and truck. This shows that although MS COCO is an excellent dataset with a large amount of data and labels (MS COCO2014 has about 120k images, including 80 categories). The perspective of natural scenes and aerial scenes are quite different. Therefore, we use the YOLOv3 model trained on MS COCO as a pre-trained model and fine-tone on VisDrone2018. It can be seen from the results that after fine-tuning, the accuracy of the YOLOv3 has greatly

Table 1. Detection results on Visdrone2018.

Model	Input	mAP (%)	Person	Bicycle	Car	Truck	Bus	Motorbike
YOLOv3 (train on MS COCO)	608×352	24.51	29.82	9.76	53.46	15.01	20.01	19.02
YOLOv3 (train on MS COCO)	960×544	31.85	42.73	12.21	64.24	18.65	28.11	25.17
RetinaNet_R50	1333×800	38.90	42.70	11.34	74.99	34.01	31.12	35.45
SSD	512×512	33.17	30.86	10.35	68.70	25.22	36.04	27.85
Tiny-YOLOv3	608×352	19.79	16.39	9.62	41.71	15.29	21.12	14.59
YOLOv3	608×352	41.40	41.02	16.18	73.35	33.81	47.16	36.90
YOLOv3	960×544	48.73	51.03	21.41	81.62	38.66	54.95	44.73
PA-YOLOv3	608×352	49.86	52.20	19.95	82.18	40.07	56.77	47.96
PA-YOLOv3	960×544	55.51	60.23	27.48	87.34	41.53	62.89	53.59

improved only 20k rounds of training (i.e. 41.40%). The accuracy has greatly improved in easily confusing such as a person, car and bus. The experimental results show that using the training results of big data for transfer training can cut down the training time and improve the detection accuracy.

YOLOv3 vs. PA-YOLOv3. Taking YOLOv3 as the baseline, we trained PA-YOLOv3 with the same parameters. The experimental results are shown in Table 1. It can be seen that compared to YOLOv3, the mAP of PA-YOLOv3 improves 8.46% (i.e. 41.40% vs. 49.86%) when the input image size is 608×352, and mAP improves 6.78% (i.e., 48.73% vs. 55.51%) when the input image size is 960×544. Moreover, the accuracy of small objects (such as person, bicycle, etc.) has greatly improved. It is shown that PA-YOLOv3 has not only greatly improved the mean detection accuracy, but also been beneficial to the detection of small objects. The accuracy of large objects (i.e. bus) has also been greatly improved (i.e. 47.16% vs. 56.77%). This shows that the dual-tower structure fully integrates the semantic information of high-level features into different feature layers, so that the feature layer with lower down-sampling factor maintains the ability to detect large objects. Compared with other one-stage detection algorithms, such as SSD512 and RetinaNet-R50, PA-YOLOv3 also has high accuracy. In Fig. 6, we show some detection results of YOLOv3 and PA-YOLOv3 on the VisDrone2018 (The objects not detected by YOLOv3 are drawn in the red dotted box).

Ablation Experiments. To demonstrate the effectiveness of different components in PA-YOLOv3, we use YOLOv3 as baseline and construct four variants and evaluate them on VisDrone2018, shown in Table 2. The image input size is 608×352 and all training parameters are the same as 4.1.

To demonstrate the effectiveness of adding one anchor, we add one anchor to the prediction layer. The mAP increased by 1.61% (i.e. 43.01% vs. 41.40%). The reason is that in dense aerial object detection, adding an anchor helps to increase the matching capability of the model, which improves the detection accuracy. When we add a mixup module on the previous network (see the fourth and fifth column in Table 2). The mAP is increased from 43.01% to 43.82%. When we add a mixup module with dual-tower (see the second and third column in Table 2), the mAP is increased from 48.94% to 49.86%. This increase (i.e. 0.81% and 0.92%) demonstrates that mixup helps to improve the accuracy of model detection and enhance the generalization ability. To demonstrate the effectiveness of the dual-tower structure, we replace the FPN structure with a dual-tower structure (see the third and fifth column in Table 2). We find that mAP is increased from 43.01% to 48.94%. The significant increase (i.e. 5.93%) demonstrates that the dual-tower structure can help promote the performance of the ability to integrate large-scale context information, further to enhance the accuracy of small aerial objects. Finally, we use both mix-up and dual-tower structure. Compared to baseline (see the second and sixth column in Table 2), the mAP is increased by 8.46% (i.e. 49.86% vs. 41.40%). The results have shown that combining different components can effectively improve model performance and accuracy.

Table 2. Effectiveness of different components.

Component	PA-YOLOv3				
Dual-tower	✓	✓			
Mixup	✓		✓		
+1 Anchor	✓	✓	✓	✓	
Map(%)	49.86	48.94	43.82	43.01	41.40

4.3. *Pruning study*

In this section, we perform model pruning on PA-YOLOv3. To demonstrate the performance of the global pruning and balance pruning methods, we conduct three model pruning experiments. The thresholds of the global pruning and the balance pruning are set to the same value and the thresholds for three consecutive pruning experiments are set to 0.3, 0.3, 0.4, respectively. The model is fine-tuned on the dataset when each pruning operation is completed. After three pruning experiments, the cumulative proportions of the pruned model are about 30%, 50% and 70%. Table 3 shows the statistics of PA-YOLOv3 after prune operations. We use Billion Floating Point Operations (BFLOPs) to measure the computational complexity of the model and use Frames Per Second (FPS) to measure the inference speed. The FPS testing GPU environment is RTX 2080Ti.

As can be seen from Table 3 that after three pruning experiments, the inference speed of the model has been improved well, the size of the model and the BFLOPS have also been greatly reduced. Comparing the global pruning and balanced pruning method, it is shown that under the same pruning threshold, the model size and BFLOPS of the

Table 3. Model pruning experiment.

Method	mAP (%)	BFLOPs	Size (MB)	FPS
No pruned	49.86	92.130	239.5	38.9
Global pruning				
30% pruned	47.13	62.881	186.1	48.2
50% pruned	44.30	48.485	143.6	65.9
70% pruned	41.23	32.249	90.6	78.6
Balance pruning				
30% pruned	48.98	44.875	117.1	60.8
50% pruned	46.36	21.834	64.3	88.7
70% pruned	43.04	11.693	32.0	101.4

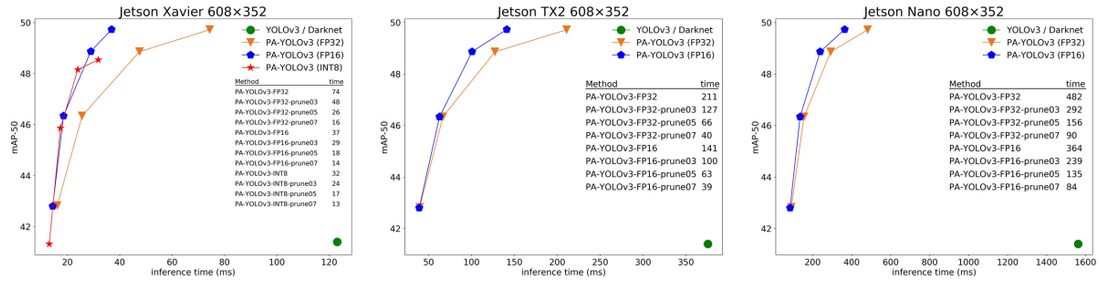

Fig. 3. Speed (ms) versus accuracy (mAP) on the Xavier, TX2 and Nano platforms.

balance pruning have a greater decrease. This experimental results show that balanced pruning can better remove high-level redundancy features and maintain the overall balance of the model. It reduces the high calculation amount caused by the retention of high-level feature channels, which greatly reduces the model size and BFLOPs while increasing FPS. Compared with global pruning, the accuracy recovery effect of balanced pruning is better. This shows that balance pruning can retain different feature layers in a more balanced manner, which is conducive to feature fusion of feature layers with different down-sampling ratios. So that the balanced pruning method is more effective when the model is fine-tuned. In the balance pruning, after the first pruning (30% Pruned), mAP has a small decrease compared to the no-pruned, which verifies the rationality and effectiveness of pruning the unimportant convolutional layer according to the scaling factor of the BN layer. After the third pruning (70% Pruned), mAP decreased by 6.82% compared to no-pruned, but the model inference speed was greatly improved (i.e. 38.9FPS vs. 101.4FPS). This is also a tradeoff between detection accuracy and speed for low-performance embedded processors with limited resources. Compared with a lightweight version of YOLOv3, Tiny-YOLOv3 (in Table 1), the 70% balance pruned model still has a high accuracy advantage (i.e. 19.79% vs. 43.04%) while maintaining a fast detection speed.

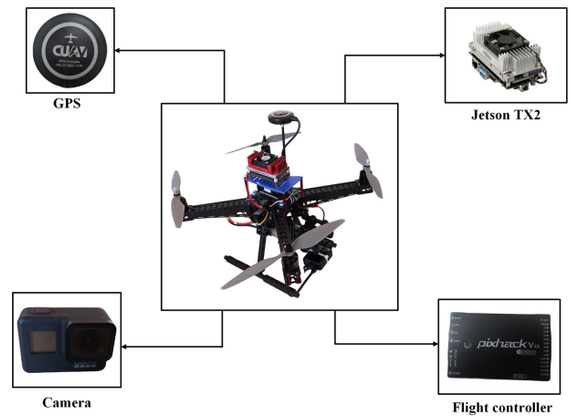

Fig. 4. The UAV system of object detection.

Fig. 5. Test results in the IMAV2019 outdoor competition (a) and the road scene (b).

Fig. 6. Comparison of detection results by YOLOv3 (left) and PA-YOLOv3 (right) on VisDrone2018. The objects not detected by YOLOv3 are drawn in the red dotted box.

4.4. *Deployment*

In this section, we deploy PA-YOLOv3 and prune the model on three embedded platforms with different performances: Jetson Nano, Jetson TX2 and Jetson Xavier. Besides, we use TensorRT technology for model quantization. On the Xavier platform, the quantization method supports three quantization modes, including fp32, fp16 and int8. On TX2 and

Nano platforms, the quantization method supports fp32 and fp16 mode. Because the TensorRT does not support the darknet format, we convert the darknet model to the Caffe model for inference. Figure 3 shows the inference speed results on three embedded platforms.

In Fig. 3, the inference speed of PA-YOLOv3-int8 can reach 32 ms on Jetson Xavier, which meets the requirements of low latency and real-time detection onboard. Moreover, it is worth noting that PA-YOLOv3-prune07 reached 13 ms which is a very short latency in int8 mode. On Jetson TX2, the PA-YOLOv3-prune07 inference speed also reaches 40 ms in fp32 mode, which can run in real-time. On the Jetson Nano with lower-performance, the inference speed of PA-YOLOv3-prune07 also reached 84 ms in fp16 mode. In contrast, the YOLOv3 model requires too much virtual memory to complete the model's inference, and can only be used for image testing because of the speed limit (1582 ms). In summary, after pruning and quantifying of PA-YOLOv3, the inference speed is greatly improved.

4.5. *The object detection system for UAV*

In this section, we design a UAV system to test the PA-YOLOv3 model. The UAV system architecture is shown in Fig. 4. The UAV system are equipped with flight controller, embedded computing platforms and vision sensors. We use the Jetson TX2 as an onboard computing platform for object detection. The vision sensor's camera provides 60fps, 1920 $\times$ 1440 color images. The system consists of a PixHack flight controller as the under-level control module, which contains a set of sensors, such as accelerometers, gyroscopes and magnetometers that are necessary to stabilize the attitude of the UAV.

In real-world scenarios testing, we use the UAV system to test in the IMAV2019 outdoor competition and the road scene. Figure 5 (left) depicts the result of the object detection mission in IMAV2019. The competition requires UAV to complete the detection of cars, houses, packages and color-marked mailboxes. We fine-tuned the PA-YOLOv3-prune07 on 300 images collected by UAVs. Detailed experimental parameters are shown in Table 4. The test results prove that the model designed in this paper can be used in practical applications and has a good accuracy.

Table 4. The parameter settings of IMAV2019.

Fine-tune model	PA-YOLOv3-prune07
Number of training images	250
Number of testing images	50
Image size	640 $\times$ 480
Max iter	2000
Batch size	64
mAP	98.05%

Figure 5 (right) shows the detection results using PA-YOLOv3-prune07 trained on VisDrone2018. The detection results show that the detection model has a generalization ability and can be used in object detection tasks in actual scenes.

5. Conclusions

Aiming at the aerial object detection task for UAV, firstly, we proposed a low-latency and high accuracy model named PA-YOLOv3. The detection accuracy and generalization ability benefit from the dual-tower structure and mixup method of our designed model. The results on the VisDrone2018 dataset show that the proposed model has 8.46% improvement in mAP, and has a better detection performance on small objects compared with YOLOv3. Secondly, we proposed balance pruning method to reduce the model size and improve the inference speed. The experimental results show that the balance pruning method can significantly reduce the model size and maintain a competitive accuracy. In the end, we deployed the model to three embedded platforms equipped with TensorRT, including Jetson Nano, Jetson TX2 and Jetson Xavier. The experimental results of deployment show that the inference speed has greatly improved and reached 25FPS on Jetson TX2 and Jetson Xavier, which satisfies the low-latency aerial object detection task. In order to verify the effectiveness of the detection algorithm designed in this paper, we built a UAV system and tested it in IMAV2019 outdoor competition and real-world scenarios. The experimental results show that the detection model has a reliable generalization ability and can be used in object detection tasks in actual scenes. In future work, we will continue to work on small object detection algorithms for aerial images, and explore the arbitrary-oriented object detection method to adapt to the rotation of the UAVs.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (Nos. 61603040 and 61973036), Yunnan Applied Basic Research Project of China (No. 201701CF00037), Guangdong Province Science and Technology Innovation Strategy Special Fund Project (No. skjtdzxrwqd2018001) and Yunnan Provincial Science and Technology Department Key Research Program, China (Engineering) (No. 2018BA070).

References

[1] G. Skorobogatov, C. Barrado and E. Salamí, Multiple UAV systems: A Survey, *Unmanned Syst.* **8**(2) (2020) 149–169.

[2] G. W. Cai, J. Dias, L. Seneviratne, A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends, *Unmanned Syst.* **2**(2) (2014) 1–26.
[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, Microsoft coco: Common objects in context, in *Proc. 13th European Conf. Computer Vision*, Zurich, Switzerland, 2014, pp. 740–755.
[4] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proc. IEEE Computer Vision and Pattern Recognition*, Columbus, USA, 2014, pp. 580–587.
[5] K. M. He, X. Y. Zhang, S. Q. Ren and J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in *Proc. 13th European Conf. Computer Vision*, Zurich, Switzerland, 2014, pp. 346–361.
[6] R. Girshick, Fast R-CNN, in *Proc. IEEE Int. Conf. Computer Vision*, Boston, USA, 2015, pp. 1440–1448.
[7] S. Q. Ren, K. M. He, R. B. Girshick and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in *Proc. Advances in Neural Information Processing Systems*, Montréal, Canada, 2015, pp. 91–99.
[8] K. E. Van de Sande, J. R. Uijlings, T. Gevers and A. W. Smeulders, Segmentation as selective search for object recognition, in *Proc. IEEE Int. Conf. Computer Vision*, Springs, Colorado, USA, 2011, pp. 1879–1886.
[9] T.-Y. Lin, P. Dollár, R. Girshick, K. M. He, B. Hariharan and S. J. Belongie, Feature pyramid networks for object detection, in *Proc. IEEE Computer Vision and Pattern Recognition*, Honolulu, USA, 2017, pp. 936–944.
[10] J. Redmon and A. Farhadi, Yolov3: An incremental improvement, preprint (2018), arXiv:1804.02767.
[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You only look once: Unified, real-time object detection, in *Proc. IEEE Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 779–788.
[12] J. Redmon and A. Farhadi, Yolo9000: Better, faster, stronger, in *Proc. IEEE Computer Vision and Pattern Recognition*, Honolulu, USA, 2017, pp. 6517–6525.
[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, SSD: Single shot multibox detector, in *Proc. 14th European Conf. Computer Vision*, Amsterdam, Netherlands, 2016, pp. 21–37.
[14] S. Zhang, L. Wen, X. Bian, Z. Lei and S. Z. Li, Single-shot refinement neural network for object detection, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Salt Lake City, USA, 2018, pp. 4203–4212.
[15] T. Lin, P. Goyal, R. Girshick, K. M. He and P. Dollár, Focal loss for dense object detection, in *Proc. IEEE Int. Conf. Computer Vision*, Venice, Italy, 2017, pp. 2999–3007.
[16] S. Jung, S. Hwang, H. Shin and D. H. Shim, Perception, guidance, and navigation for indoor autonomous drone racing using deep learning, *Robot. Autom. Lett.* **3**(3) (2018) 2539–2544.
[17] M. Hsieh, Y. Lin and W. H. Hsu, Drone-based object counting by spatially regularized regional proposal network, in *Proc. IEEE Int. Conf. Computer Vision*, Venice, Italy, 2017, pp. 4165–4173.
[18] J. Q. Cui *et al.*, Search and rescue using multiple drones in post-disaster situation, *Unmanned Syst.* **4**(1) (2016) 83–96.
[19] J. Zhang, J. Cao and B. Mao, Application of deep learning and unmanned aerial vehicle technology in traffic flow monitoring, in *Proc. Int. Conf. Machine Learning and Cybernetics*, Ningbo, China, 2017, pp. 189–194.
[20] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun and R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, in *Proc. Advances in Neural Information Processing Systems*, Montréal, Canada, 2014, pp. 1269–1277.
[21] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in *Proc. 14th European Conf. Computer Vision*Amsterdam, Netherlands, 2016, pp. 525–542.
[22] S. Han, J. Pool, J. Tran, and W. Dally, Learning both weights and connections for efficient neural network, in *Proc. of the Advances in Neural Information Processing Systems*, Montréal, Canada, 2015, pp. 1135–1143.
[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan and C. Zhang, Learning efficient convolutional networks through network slimming, in *Proc. IEEE Int. Conf. Computer Vision*, Venice, Italy, 2017, pp. 2755–2763.
[24] H. Gao, W. Tao, D. Wen, T. W. Chen, K. Osa and M. Kato, IFQ-Net: Integrated fixed-point quantization networks for embedded vision, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Salt Lake City, USA, 2018, pp. 607–615.
[25] J. Gong, H. Shen, G. Zhang, X. Liu, S. Li and G. Jin, Highly efficient 8-bit low precision inference of convolutional neural networks with intelcaffe, in *Proc. 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning*, 2018.
[26] M. Szymon, Nvidia 8-bit inference with tensorrt, in *Proc. 2017 GPU Technology Conf.*, San Jose, USA, 2017.
[27] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, preprint (2015), arXiv:1502.03167.
[28] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, Path aggregation network for instance segmentation, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Salt Lake City, USA, 2018, pp. 8759–8768.
[29] H. Y. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, mixup: Beyond empirical risk minimization, in *Proc. Int. Conf. Learning Representations*, 2018.
[30] Z. Zhang, T. He, H. Zhang, Z. Y. Zhang, J. Y. Xie and M. Li, Bag of freebies for training object detection neural networks, preprint (2019), arXiv:1902.04103.
[31] P. F. Zhu *et al.*, VisDrone-VDT2018: The vision meets drone video detection and tracking challenge results, in *Proc. 15th European Conf. Computer Vision Workshops*, Munich, Germany, 2018, pp. 437–468.

January 5, 2022 1:27:51pm WSPC/284-US 2250003 ISSN: 2301-3850 2nd Reading

Kai Feng is a postgraduate student majoring in control science and engineering in Beijing Institute of Technology (BIT), China. He received his B.S. degree from BIT in 2018. His research interests include computer vision, deep learning and robotics.

Weixing Li is an associate professor at School of Automation in Beijing Institute of Technology, China. He received his M.S. degree in pattern recognition and intelligent control from Beijing Institute of Technology, China. His research interests include pattern recognition, video analysis and machine learning.

Jun Han is a postgraduate student majoring in control science and engineering in Beijing Institute of Technology (BIT), China. He received his B.S. degree from BIT in 2019. His research interests include computer vision and multi-object tracking.

Feng Pan is an associate professor at School of Automation in Beijing Institute of Technology (BIT), China. He received his M.S. and Ph.D. degrees in pattern recognition and intelligent control from BIT. His research interests include intelligent computing, robotics and control, and computer vision.