Lightweight target detection model for embedded platform

Yuhuan Li *, Jie Wang, Baodai Shi

Air Defense and Missile Academy, Air Force Engineering University, Shaanxi, Xi’an 710051, China

*Corresponding author’s e-mail:xiao_wandering@163.com

Abstract: The detection speed of target detection algorithm depends on the performance of computer equipment. Aiming at the problems of slow detection speed and difficult trade-off between detection accuracy and detection speed when the target detection model is used in embedded devices, a lightweight target detection model based on the improved Tiny YOLO-V3 is proposed. Firstly, we analyze the time consumption of each layer structure in the convolutional neural network, and do a lot of experiments and tests. Then, we compress the time-consuming structure substantially. Secondly, we propose the segmentation and fusion module to improve the detection accuracy. Finally, we use the remote sensing data set of Wuhan University for experiments, and the experimental results show that compared with Tiny YOLO-V3, the detection speed is improved by 4 times, and the accuracy is improved by 2 percentage points.

1. INTRODUCTION

In recent years, the target detection algorithm has developed rapidly, and the accuracy and speed of detection have reached a new height. Yolov4 [1] achieved 43.5% Average Precision (AP) on MS Coco data set [2], and the detection speed can reach 65 Frames Per Second (FPS) on Tesla V100 GPU. However, the detection accuracy of target detection algorithm depends on a large number of network parameters, so it depends on the performance of computing equipment. Because most embedded processors are Central Processing units (CPUs) with a small number of arithmetic logic units, the detection speed of target detection model is greatly reduced. Tiny Yolov3, for example, can detect at 51.6FPS on GeForce GTX 1660 Ti(GPU), but only 6.2FPS on Intel(R) Core(TM) I7-9700 (CPU), making it difficult to detect objects in real-time on embedded devices. Therefore, the target detection model needs to be compressed.

At present, there are two methods to compress the target detection model: one is to compress the network parameters, the other is to design a new network structure. Han[3] et al. adopted pruning, weight sharing, weight quantization and Huffman coding methods to compress the parameters of the ALEXNET model by 35 times and the parameters of the VGG model by 49 times without loss of accuracy, and achieved a good improvement in the network speed and network energy consumption. Han[3] et al. adopted pruning, weight sharing, weight quantization and Huffman coding methods to compress the parameters of the ALEXNET model by 35 times and the parameters of the VGG model by 49 times without loss of accuracy, and achieved a good improvement in the detection speed and network energy consumption. LeCun[4] et al. proposed OBD algorithm and pruned it according to the contribution of parameters, so as to achieve the optimal balance between network complexity and training error. Molchanov et al. [5] proposed an iterative pruning method, in which pruning and fine-tuning are carried out alternately until the balance between precision and pruning...
target is reached, and the pruning effect is significantly improved. Luo [6] et al. prune the filter according to the statistical information calculated in the next layer, and improve the accuracy through fine-tuning, which can be easily transplanted to other models. Wen [7] et al. proposed the SSL method to make the network structure sparse, and the acceleration effect was obvious on MNIST data set, while the performance was poor on ImageNet [8] data set. However, this kind of method is mostly applicable to large networks, and the effect is not obvious for lightweight networks.

Iandola [9] et al. first opened up the direction of designing a new network by replacing the 3×3 convolution kernel with the 1×1 convolution kernel, which greatly reduced the parameters. However, due to the deeper network, the detection time was not greatly shortened. Howard [10] et al. used deep separable convolution and reduced the parameters of the convolution layer to about 1/9 of the original, but the network structure was too simple, resulting in a sharp decline in detection accuracy. Zhang [11] et al. rearranged the packet convolution according to channels to realize the information exchange between channels, which enhanced the feature expression ability of the network without increasing the computational cost. However, the 1×1 convolution layer still occupied a large amount of computational cost. Han [12] et al. proposed the GHOST module, which uses fewer parameters to generate more feature maps, and reduces the computational cost of the general convolutional layer while maintaining the detection accuracy. Most of the current research measures the complexity of the model by the number of Floating Point of Operations (FLOPS). It is believed that reducing the number of parameters and calculation cost can effectively improve the detection speed of the model. However, the experiment proves that the FLOPS of deep network and shallow network have different influence on the model detection speed.

Choi et al. [13] pointed out that the actual performance of the target detection model is affected by other factors such as computing power and bandwidth of the computing platform. Although the calculation amount and number of parameters of Tiny Yolov3 are significantly reduced compared with that of Tiny Yolov3 [14], the detection speed of the model fails to meet practical needs due to the weak computing power, bandwidth and other performance of embedded devices. To solve the above problems, we analyze the time consumption of Tiny Yolov3 blocks. The time-consuming parts are further compressed to reduce unnecessary computation while maintaining detection accuracy, so as to improve the detection speed of the target detection model running on embedded devices.

2. TINY YOLOV3

Target detection algorithms based on convolutional neural networks are mainly divided into two types: one-stage and two-stage. The single-stage target detection algorithm uses regression method to directly predict the target position and category, so the detection speed is faster and the accuracy is high. As a typical representative of single-stage target detection algorithm, YOLOV3 focuses more on improving detection speed and maintains high detection accuracy by using structures such as shortcut connection [15] and FPN [16] (Feature Pyramid Networks). Based on its lightweight version, Tiny Yolov3, we try to propose a smaller target detection model.

2.1 Algorithm principle

As shown in Figure 1, Tiny Yolov3 uses a lightweight network structure, using convolutional layers for feature extraction and pooling layers for down-sampling. The CBL(k, i, o) in the figure consists of three parts: the convolutional layer (the size of the convolutional kernel is k×k, and the number of input channels is i, and the number of output channels is o), Batch Normalization (BN) [17] and and the activation function Leaky ReLU. The convolution operation makes linear transformation to the feature map; BN makes the input of each layer keep the same distribution to prevent the gradient from vanishing and overfitting; Leaky ReLU enhances the nonlinear expression ability of the network and solves the gradient vanishing problem. The down-sampling uses the pooling layer instead of the convolutional layer with a step size of 2, because although the pooling layer will lose part of the characteristic information, it can reduce the amount of calculation and improve the detection speed. The frame of the target in the data set is classified into
two categories: large size and small size. The structure of FPN is adopted. Small targets are detected using a 32-fold subsampled detector head. The eighth layer is up-sampled and then fused with the fifth layer, which is used as a 16-fold down-sampled detection head to detect large-size targets. This improves the recall rate of large-size targets.

2.2 Time consuming analysis

It is generally believed that the number of parameters, time complexity and space complexity affect the lightweighting degree of the model. The number of parameters is the sum of the number of parameters to be trained for each layer of network. The calculation method is shown in Equation (1). The time complexity is FLOPs, and the calculation method is shown in Equation (2). The spatial complexity includes the total number of parameters of the model and the size of network output feature map of each layer. The calculation method is shown in Equation (3).

\[ N = D_k^2 \times C_{in} \times C_{out} \]  
\[ T = D_k^2 \times D_c^2 \times C_{in} \times C_{out} \]  
\[ O = (D_k^2 \times C_{in} \times C_{out} + D_c^2 \times C_{out}) \times 4 \]

In the formula, \( D_k \) is the size of the convolution kernel, \( C_{in} \) is the number of input channels, \( C_{out} \) is the number of output channels, and \( D_c \) is the resolution of the output feature map. But the experiment proves that the complexity of deep network and shallow network has different influence on the detection speed. We used the time consumption of the code of the convolutional layer and the pooling layer to represent its occupation of device resources. We used CPU to run the model 100 times and analyzed the time consumption of the entire network structure. The results are shown in Table 1:

| Layer | Featuremap size       | Kernel size     | FLOPs    | Time(100hits) | Proportion/√% |
|-------|-----------------------|-----------------|----------|---------------|---------------|
| Conv1 | 416x416x3             | [16, 3, 3, 3]   | 1.50x10^8 | 1.97s         | 14.7          |
| Maxpool1 | 416x416x3            | [2, 2]           | 1.29x10^5 | 1.39s         | 10.4          |
| Conv2 | 208x208x16            | [32, 16, 3, 3]  | 3.98x10^8 | 1.52s         | 11.3          |
| Maxpool2 | 208x208x16           | [2, 2]           | 1.73x10^5 | 0.68s         | 5.1           |
| Conv3 | 104x104x32            | [64, 32, 3, 3]  | 3.98x10^8 | 1.05s         | 7.8           |
| Maxpool3 | 104x104x32           | [2, 2]           | 8.65x10^4 | 0.35s         | 2.6           |
| Conv4 | 52x52x64              | [128, 64, 3, 3] | 3.98x10^8 | 0.73s         | 5.4           |
| Maxpool4 | 52x52x64             | [2, 2]           | 4.33x10^4 | 0.17s         | 1.3           |
| Conv5 | 26x26x128             | [256, 128, 3, 3]| 3.98x10^8 | 0.56s         | 4.1           |
| Maxpool5 | 26x26x128            | [2, 2]           | 2.16x10^4 | 0.09s         | 0.7           |
| Conv6 | 13x13x256             | [512, 256, 3, 3]| 3.98x10^8 | 0.56s         | 4.2           |
| Maxpool6 | 13x13x256            | [1, 1]           | 1.08x10^4 | 0.18s         | 1.4           |
Lightweight target detection models for embedded devices should be analyzed and designed using embedded devices. Analyzing the data in Table 1, the following conclusions can be drawn: (1) The time consumption of the whole network is mainly concentrated in the first two layers; (2) When the size of the feature map is large, the time consumption of convolution and pooling is relatively large; (3) Conv7 and Conv12 used 3×3 convolution kernel and more channels, resulting in too large FLOPs and time consumption. For the above problems, there are the following improvement strategies: (1) Reduce the number of channels in the shallow layer network to reduce the time consumption in the first two layers; (2) Reduce the maximum number of channels in the whole network while ensuring the feature expression ability of the network does not decline; (3) For Conv7 and Conv12, a 1×1 convolution kernel can be used for dimensionality reduction first, and then a 3×3 convolution kernel can be used to extract features, so as to maintain a large receptive field while reducing the amount of computation.

### 3. IMPROVED MODEL

#### 3.1 Split concatenate module

In feature extraction, we proposed Split Concatenate Module (SCM). As shown in Figure 2, the size of the input feature graph is \( M \times M \) and the number of channels is \( 2C \). The input feature map is divided into C1 and C2 according to the number of channels. C2 is convolved with a 3×3 convolution kernel, and then fused with C1. On the one hand, SCM reduces the number of channels in the convolutional layer by half, greatly reducing the amount of calculation, and thus reducing the time consumption of the convolutional layer to improve the detection speed. On the other hand, C1 is directly fused with the convoluted feature map, so that the output feature map retains part of the features of the previous layer and contains more spatial information, which can effectively improve the detection effect of small targets.

![Figure 2. Split Concatenate Module Structure Diagram](image)

#### 3.2 Network structure

The improved network structure is shown in Figure 3. The size of the input image should match the magnitude of the model. If the size is too large, the calculation cost will increase greatly; if the size is too small, a lot of information will be lost and the detection accuracy will be reduced. Weighing the above two points, we chose 320pixel×320pixel as the size of the input image.
The feature map can be subsampled by pooling or convolution of step size 2. The input feature graph is subsampled in two ways respectively, and the time consumption analysis shows that the pooling time is smaller than the convolution time except for the first layer. Therefore, in the first layer, the convolution with a step size of 2 can significantly reduce the time consumption of the whole network and retain a large amount of image information, which can effectively improve the detection speed while maintaining the accuracy. The visualization results of feature map of each channel of F1 are shown in Figure 4. The SCM structure and the maximum pooling layer were alternately used for F1, and the 16-fold down-sampling layer F5 was obtained. Due to the large number of F5 channels, the first use of 1×1 convolution dimensionality reduction, continuous use of two 3×3 convolution to fully extract features and obtain a larger receptive field, which can effectively improve the detection effect of large-size targets. Due to the large number of F5 channels, the first use of 1×1 convolution dimensionality reduction, continuous use of two 3×3 convolution to fully extract features and obtain a larger receptive field, which can effectively improve the detection effect of large-size targets. And then increase the dimension. In this way, the feature expression ability of the network is improved and the computing cost is reduced.
The number of parameters in the improved network is $1.16 \times 10^5$, which is reduced by 98.68% compared with the original network. FLOPs is $2.29 \times 10^6$, which is 98.31% less than the original network. The space complexity is $1.64 \times 10^5$, which is 98.19% less than that of the original network.

4. EXPERIMENT AND ANALYSIS

4.1 Model training and testing

The dataset used in the experiment is RSOD-Dataset[20] provided by Wuhan University, which contains 936 images in total, including 446 images of aircraft, 165 images of oil tank, 176 images of overpasses, and 149 images of playground. Some of the images are shown in Figure 5. There are a large number of aircraft and oil tank images in the data set, and the target size in the images is small and the samples are rich. The number of images of overpass and playground is small, and the size of the target in the images is large and the samples are rare. The dataset contains a small number of images, rare samples and complex background, so it increases the difficulty of learning the model. In the experiment, random flipping, clipping, translation and other methods are used to enhance the data. In addition, due to a certain error in the frame of the target, increasing or decreasing its width and height by 0-5% randomly can reduce overfitting and enhance data to a certain extent. 85% of the data set was randomly selected as the training set and the remaining 15% as the test set. The k-means algorithm is used to cluster the boxes in the data set into six categories according to the size, which are respectively: (8,8), (13,13), (18,19), (25,27), (37,42), (112,138).

![Figure 5. Partial data set sample diagram](image)

The configuration of experimental equipment during training is as follows: CPU, Intel(R)Core(TM) i7-9700 CPU 3.00GHz, RAM 16GB; The graphics card is GeForce GTX 1660TI with 6GB of memory; The operating system is Windows10; The deep learning framework is Pytorch 1.1.0. Adopting ADAM optimization algorithm, the learning rate increases linearly in the preheating stage, with a maximum value of $1 \times 10^{-4}$, and then decreases by cosine annealing. The learning rate change curve in the training process is shown in Figure 6. The batch size was set as 6, and each round included 133 batch. After 10020 iterations, the loss value converged to 3.798. The change curve of loss value in training is shown in Figure 7.
During the test, the model should be run on an embedded device. In the experiment, CPU is used to replace the embedded device, and video card is not used. The targets in the 139 images of the test set are detected, and the results are shown in Figure 8. Most of the large and small size objects in the images are correctly identified, but there are some missed and false detection due to the high lightweight degree of the model.

### 4.2 Comparative experiment of the models

In order to analyze the performance improvement effect of the improved model in terms of detection accuracy and detection speed, the improved model was compared with Tiny Yolov3. The main comparison indexes include Precision, mean Average Precision (mAP) and FPS.

Precision represents the proportion of correctly identified targets among identified targets in each category. Recall represents the proportion of targets are correctly identified in all targets of each category. The calculation method is shown in the formula:
Precision_A = \frac{TP_A}{TP_A + FP_A} \tag{4}

Recall_A = \frac{TP_A}{TP_A + FN_A} \tag{5}

In the formula, \( TP_A \) represents the number of correctly identified targets in category A, \( FP_A \) represents the number of wrongly identified targets in category A, and \( FN_A \) represents the number of unidentified targets in category A. There is a negative correlation between Precision and Recall. The area below the precision-recall curve of category A is taken as the Average Precision (AP) of category A, and the average value of AP of each category is taken as mAP, which represents the average accuracy of the model for targets of each category. FPS is the number of frames detected by the model per second. Compared with the number of parameters, FLOPs and spatial complexity, FPS can better reflect the lightweight degree of the model. The 139 images in the test set were detected and relevant data were recorded at the same time, and the MAP comparison results of the model were obtained as shown in Table 2:

| Model               | AP(aircraft) | AP(oiltank) | AP(overpass) | AP(playground) | mAP   |
|---------------------|--------------|-------------|--------------|----------------|-------|
| Tiny YOLO-V3        | 0.608        | 0.397       | 0.007        | 0.023          | 0.253 |
| The improved network| 0.675        | 0.380       | 0.046        | 0.017          | 0.279 |

Compared with Tiny Yolov3, the detection accuracy of the improved model for aircraft and overpasses is improved by about 7% and 4% respectively, but the detection accuracy for oil tank and playground is slightly decreased, and MAP is improved by about 2.6%. Because there are few images of oil tank, overpasses and playground in the data set, and there are few samples in the images of overpasses and playground, the detection accuracy is low, which can be improved by expanding the data set. In addition, the detection speed of the improved model is 34.97 FPS, which is about 463% higher than that of the Tiny Yolov3. This model can be directly applied to the embedded equipment such as UAV platform to detect and recognize ground remote sensing image targets.

5 CONCLUSION

Firstly, we illustrated the importance of model compression for the application of target detection algorithm to embedded devices, analyzed the advantages and disadvantages of various model compression methods, and pointed out the limitations of the factors affecting the lightweight of the model. Secondly, we analyze and improve Tiny Yolov3 from the perspective of time consumption, compress it by about 98%, and propose a split fusion module to maintain the detection accuracy. Finally, the experimental results show that the detection accuracy of the improved model is improved by about 2%, and the detection speed is improved to 34.97 FPS when running on CPU. The model can be deployed to the UAV platform and other embedded devices to detect and locate the ground remote sensing target. However, due to the small number of samples in the data set, the detection accuracy is not high. Data set can be expanded by generating data, manual annotation, and so on. When the target detection model is run on different devices, the occupation of time consumption of each layer is different, so the model should be designed for the used device. We provide a new idea for model compression, but new network structure and optimization algorithm are still needed to improve the detection accuracy and speed.

References

[1] Bochkovskiy A, Wang C Y and Liao H Y M. 2020. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.
[2] Lin T Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollar P and Zitnick C L. 2014. Microsoft coco: Common objects in context. European conference on computer vision. Springer, Cham: 740-755.
[3] Han S, Mao H and Dally W J. Deep compression: Compressing deep neural networks with
pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[4] LeCun Y, Denker J S and Solla S A. Optimal brain damage[C]//Advances in neural information processing systems. 1990: 598-605.

[5] Molchanov P, Tyree S, Karras T, Aila T and Kautz J. Pruning convolutional neural networks for resource efficient inference[J]. arXiv preprint arXiv:1611.06440, 2016.

[6] Luo J H, Wu J and Lin W. Thinet: A filter level pruning method for deep neural network compression[C]//Proceedings of the IEEE international conference on computer vision. 2017: 5058-5066.

[7] Wen W, Wu C, Wang Y, Chen Y and Li H. Learning structured sparsity in deep neural networks[J]. arXiv preprint arXiv:1608.03665, 2016.

[8] Deng J, Dong W, Socher R, Li L J, Li K and Fei L F. Imagenet: A large-scale hierarchical image database[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255.

[9] Iandola F N, Han S, Moskewicz M W, Ashraf K, Dally W J and Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

[10] Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W J, Weyand T, Andreetto M and Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[11] Zhang X, Zhou X, Lin M, Sun J and Inc M. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.

[12] Han K, Wang Y, Tian Q, Guo J, Xu C J and Xu C. Ghostnet: More features from cheap operations[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 1580-1589.

[13] Choi J W, Bedard D, Fowler R, et al. A roofline model of energy[C]//2013 IEEE 27th International Symposium on Parallel and Distributed Processing. IEEE, 2013: 661-672.

[14] Redmon J and Farhadi A. Yolov3: An incremental improvement[EB/OL]. (2018 - 04 - 08) [2019 - 09 - 19]. https//arxiv.org/abs/1804.02767.

[15] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Las Vegas:IEEE Computer Society 2016: 770-778.

[16] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.

[17] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. PMLR, 2015: 448-456.

[18] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[19] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR), July 21-26, 2017. Honolulu, HI. IEEE, 2017:6517-6525.

[20] Xiao Z, Liu Q, Tang G, et al. Elliptic Fourier transformation-based histograms of oriented gradients for rotationally invariant object detection in remote-sensing images[J]. International Journal of Remote Sensing, 2015, 36(2): 618-644.