Yolo-tiny-MSC: A tiny neural network for object detection

Dongsheng Li¹, Yujie Zhang¹,²*, Junping Xiang¹, Jianfei Li¹, and Yu Zou¹

¹Lianyungang JARI Electronics Co., Ltd. of CSIC, Lianyungang, China
²Jiangsu Automation Research Institute of CSIC, Lianyungang, China
*Corresponding author: zhangyujie@jari.cn

Abstract: Object detection is an important and active area of computer vision. However, the expensive computational cost and memory requirement are big challenges for deploying object detection networks in resource constraints embedded devices. To address this problem, we designed the multi-stage cascade module to reduce the computational cost of object detection networks. The proposed structure is used to build a compact and efficient object detection neural network. Experiments show that our multi-stage cascade module could significantly reduce the computational and parameter budgets meanwhile achieve outstanding object detection accuracy.

1. Introduction

Object detection is one of the most important and active areas of computer vision. In recent years, deep-learning-based object detection methods have been attracting increasing amounts of attention. Due to the technological breakthroughs in deep-learning made in recent years, many convolutional neural networks (CNNs) based object detection methods have been proposed. The advanced CNN-based methods have significantly improved the accuracy of object detection. These methods tend to adopt complex structures to enhance the representation ability of networks, such as R-CNN [1], SSD [2], Fast-RCNN [3], and Faster-RCNN [4]. However, even though these outstanding CNN-based detectors show state-of-the-art object detection accuracies, it is still challenging to deploy them on embedded mobile devices for practical applications. One of the major reasons is that these networks possess huge parameter sizes and require large amounts of computing resources, thus they are heavily dependent on the high-performance Graphics Processing Units (GPU) to achieve the inference process. Therefore, it is difficult to deploy these networks on embedded devices with limited computational ability and memory resources. This problem impedes the widespread application of these high-performance networks in stand-alone terminal detection devices that require almost real-time detection such as video surveillance and unmanned aerial vehicles. Thus it is necessary to design compact and lightweight CNN-based object detection neural networks that are suitable for edge devices with limited computational resources.

In the last few years, to reduce computation cost, many lightweight single-shot deep-learning-based detectors have been proposed. Among them, the tiny Yolo series is an attractive and interesting family that greatly reduces the model-size and computational cost of object detection networks. And it is extended into many variants [5], [6]. Yolov3-tiny [7], as one lightweight and efficient object detection network, is well-received in many areas. However, its performance needs to be further improved. Compared with yolov3-tiny, yolov4-tiny [8] achieves higher object detection accuracy, but it requires more computational operations than yolov3-tiny. In order to find a fine balance between detection accuracy and computational cost, in this paper, a multi-stage cascade (MSC) module is proposed. The MSC module composes of a series of convolutional layers and all features of convolutional layers are...
merged by a certain method. We replaced the CSP (cross spatial partial) module [[9]] in yolov4-tiny with our MSC module to build a compact network. Comparison experiments suggest that the MSC module could significantly reduce the computation operators and the number of parameters of the network, meanwhile achieve superior object detection accuracy.

2. Method

The basic aim of our multi-stage cascade module is to achieve a trade-off between object detection accuracy and computation complexity of the neural network. The main design strategy of our multi-stage cascade module is based on two thoughts: 1) use multi-stage cascade structure as the backbone, 2) use progressive aggregation to merge features in different stages.

2.1 Multi-stage cascade structure

Our multi-stage cascade structure is inspired by the Inception module [[10], [11]]. In Inception, input feature is processed by a combination of convolution layers with different kernel size, and the output of these convolution layers are concatenated into a single output. The naïve Inception module without pooling path is shown in Fig. 1(a). However, convolution with kernel size larger than 5 is computationally expensive and this structure can lead to a computational blow-up. Furthermore, in [[11]], it suggests that convolutions with large spatial filters can be reduced into a sequence of 3×3 convolutional layers to relieve the disproportionally expensive in terms of computation. In this condition, the reduced Inception is shown in Fig. 1(b).

As shown in Fig. 1(b), feature \( f_1 \) and \( f_2 \) are both generated by processing base input with a 3×3 convolution layer. Therefore, in terms of the receptive field, feature \( f_1 \) and \( f_2 \) are equivalent. It is kind of redundant to deploy two different convolutional layers to produce features with the same receptive field level. Based on this assumption, in our multi-stage cascade structure, we consider merging convolution operators with the same receptive level in different paths to reduce computational cost. The backbone of the two-stage cascade structure is shown in Fig. 1(c) (except parts marked with dashed lines). In fact, this structure could be further extended by cascading more convolutional layers behind the base structure and then concatenate the outputs together.

In order to balance the dimension of input and output, we adopt two strategies. First, decreasing the number of filters stage-by-stage. Second, employing 1×1 convolutional layers to adjust the channels of features in different paths. Through these two strategies, the dimension of input is kept to be equal to the dimension of output. As an example, the detailed structure of the basic four-stage cascade module is shown in Fig. 2(a). The parameter budget of the MSC module with \( n \) stages can be formulated as:

\[
P = \sum_{i=1}^{n} \left( 1^2 + 3^2 \right) \frac{c}{2^i-1} \frac{c}{2^i} = \sum_{i=1}^{n} \frac{10c^2}{2^{2i-1}} = \sum_{i=1}^{n} \frac{20c^2}{4^i} = \frac{20c^2 4^n-1}{3 4^n} \quad (1)
\]

where, \( c \) is the channels of input feature. The limitation of Eq. (1) is \( \lim(P) = 20c^2/3 = 6.67c^2 \). For comparison, the parameter budget of commonly used 3×3 convolutional layer with the same input and output channels is 9\( c^2 \), which is 1.35 times of MSC module.
2.2 Progressive aggregation

In the basic structure of the multi-stage cascade module, features in different stages are concatenated directly. However, due to the channel decreasing strategy, the number of features at high level is much less than the numbers of features at low levels. In this condition, the proportion of features generated in each stage is not balanced in the output.

In order to solve this problem, we design PA (progressive aggregation) structure to fuse features in different stages from the bottom layer gradually. In each fusion, only outputs in two adjacent stages are concatenated together. Thus the dimensions of the two features are equal in every fusion operation. And then a 1×1 convolutional layer is applied to further integrate the fused features. As an example, the structure of four-stage cascade module with PA is shown in Fig. 2(b).

3. Experiment Results and Discussion

Based on the architecture of the yolo-tiny series, we designed a lightweight object detection network by replacing CSP modules in yolov4-tiny with our MSC module. To study the efficacy of our method, we examine its computational resource cost and object detection accuracy on COCO [[12]] dataset. The yolov3-tiny and yolov4-tiny are used as the baseline models since they are both classical and popular compact neural networks for object detection.

All baselines and our methods are trained from scratch on COCO datasets by the Darknet neural network framework. All evaluation is conducted on the condition that the input size is 416×416. The final results are shown in Table 1. And the best results are highlighted in bold.

| models            | #Params (×10^6) | #FLOPs (×10^9) | mAP   |
|-------------------|-----------------|----------------|-------|
| Yolov3-tiny       | 8.845           | 5.571          | 30.8% |
| Yolov4-tiny       | 6.050           | 6.910          | 38.4% |
| Yolo-tiny+MSC     | **5.546**       | **5.353**      | 38.1% |
| Yolo-tiny+MSC+PA  | 5.574           | 5.440          | **38.6%** |

As shown in Table 1, in terms of the number of parameters, our method possesses a parameter size of about 5.5×10^6, which is 0.62 times smaller than yolov3-tiny and is also smaller than yolov4-tiny. And for FLOPs, our methods require the fewest computational operations, which is 0.78 times of yolov4-tiny. It indicates that our method could significantly reduce the computation and parameter budget of the network, which is important for resource constraints embedded devices.

In addition, although leveraging the fewest parameters and calculations, our methods achieve outstanding object detection accuracy. The mAP of our method is about 8% higher than yolov3-tiny. The network with progressive aggregation strategy even performs slightly better than yolov4-tiny.
4. Conclusion

In this paper, a multi-stage cascade module is proposed for compact object detection neural networks. Based on the Inception module, the MSC module reuses convolution operators with the same receptive field level to remove redundant calculations. Compared with the traditional convolution layer, the MSC module requires fewer parameter budgets and computation budgets, which is of great significance for embedded platforms with constrained resources. Based on the MSC module, a tiny-size object detection neural network is constructed, and then the tiny network is trained and evaluated on the COCO dataset. Experiments demonstrate that our method achieves almost equivalent object detection accuracy as yolov4-tiny with a fewer computational cost.

Reference

[1] Girshick Ross, Donahue Jeff, Darrell Trevor, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.
[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. SSD: Single Shot MultiBox Detector [C]. European conference on computer vision. Springer, Cham, 2016: 21-37
[3] Ross Girshick. Fast R-CNN [C]. Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
[4] Shaoqing Ren, Kaiming He, Ross Girshick, et al. Faster R-CNN: Towards real-time object detection with region proposal networks [J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(6): 1137-1149.
[5] Zhou L, Wei S, Cui Z, et al. YOLO-RD: A lightweight object detection network for range doppler radar images[C]. IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2019, 563(4): 042027.
[6] Rani E. LittleYOLO-SPP: A delicate real-time vehicle detection algorithm [J]. Optik, 2021, 225:165818.
[7] J. Redmon. YOLO: Real-time object detection. https://pjreddie.com/darknet/yolo/, 2016
[8] Alexey Bochkovskiy. Yolov4-tiny. https://github.com/AlexeyAB/darknet. 2020
[9] Wang C Y, Liao H Y M, Wu Y H, et al. CSPNet: A new backbone that can enhance learning capability of CNN[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 390-391.
[10] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C].Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
[11] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.
[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. Microsoft COCO: Common Objects in Context [C]. European conference on computer vision. Springer, Cham, 2014: 740-755.