Smart-YOLO: A Light-Weight Real-time Object Detection Network

Smart-YOLO: A Light-Weight Real-time Object Detection Network

Dongyang Zhang[1], Xiaoyan Chen[1,*], Yumeng Ren[1], Nenghua Xu[2], Shuangwu Zheng[2]

[1] College of Electronic Information and Automation, Tianjin University of Science & Technology, No.1038, Dagu South Road, Tianjin 300222, China

[2] Shenzhen Softsz Co., Ltd. Tianjin Branch, Rm.1701-1702, Shuangying Building, No.13, Heiniucheng Road, Tianjin 300222, China

*cxywxr@tust.edu.cn

Abstract. In the computer vision, YOLO has an important position in the field of object detection, but due to its speed limitation, it is not suitable for scenes that require extremely strict real-time performance, such as smart cameras or some mobile devices. However, inverted bottleneck layers, which are built upon depth-wise separable convolution, have been the predominant building blocks in state-of-the art object detection models on mobile devices. In this work, we propose a new lightweight algorithm Smart-YOLO based on the YOLO framework which uses inverted bottleneck blocks and deep-wise separable convolution. We also put forward a new loss function to make up for the loss of accuracy caused by the replacement of the backbone network. The results show that compared with YOLOv3, the accuracy of our model is reduced by about 21%, but achieved up to 4.5× speedup, and the model size is only about 1/8 of the original. This shows that our network is smaller, faster, and more suitable for scenarios that require higher speed and efficient

Keywords: Object Detection, CNN, YOLO, Real-time detection

1. Introduction

Human beings can easily detect and identify the surrounding objects in real time, and it takes almost time to react. However, using a computer to detect and recognize the same object requires a lot of calculation and processing, and consumes a lot of time to extract the information of the object. But in some edge devices (such as smart cameras), the speed and real-time requirements are extremely strict. Therefore, it is of great significance to accelerate and compress the object detection model.

At present, the development of lightweight models is divided into two directions. The first is to start from the model itself and directly develop high-efficiency networks. MobileNetV1 employs depth-wise separable convolution to substantially improve computation efficiency. MobileNetV2 introduces a resource-efficient block with inverted residuals and linear bottlenecks. Although MobileNets series can also be used for object detection, it is more used for classification tasks. EfficientDets combines Bi-FPN feature network structure and EfficientNet to perform efficient multi-

scale feature fusion and model expansion. Although EfficientDet has excellent performance, it is designed based on empirical rules and there are many parameters that are difficult to explain.

Another way is to reduce the high-precision models' parameters. Poly-YOLO uses a light SE-Darknet-53 backbone to solve the problem of a large amount of rewritten labels and inefficient distribution of anchors, which improves the detection speed but the amount of parameters is still huge. Tanvir Ahmad adds an inception model with a convolution kernel of 1×1 in YOLO, which reduces the number of weight parameters of the layers. The detection speed has been improved to a certain extent but not obviously enough.

Based on the YOLO framework, this paper replaces the backbone network with an inverted bottleneck structure, and modified loss functions. Finally, a real-time lightweight object detection model is obtained which can indeed strike a better trade-off between accuracy and efficiency.

2. Smart YOLO object detection model

In order to further optimize the network structure, decrease computational and memory cost, and speed up the algorithm detection, this paper proposes a lightweight object detection network Smart-YOLO based on YOLO. This section introduces the network structure of the Smart-YOLO model in detail, including the feature extraction layer, object detection layer, and loss layer.

2.1. Backbone Networks

Smart-YOLO uses the inverted residual with linear bottleneck as the backbone. The structure and convolution process are shown in Fig.1. First use a 1×1 convolution to increase the dimension, and then perform the convolution operation. It also has a residual branch. Through the point convolution of the size of 1×1 convolution kernel, the number of input channels can be greatly increased, and then a deep-wise separable convolution layer is adopted for feature extraction. The improvement of the channel will bring higher feature extraction ability, the depth-wise separable convolution reduces the amount of parameters and calculations less than original YOLO.

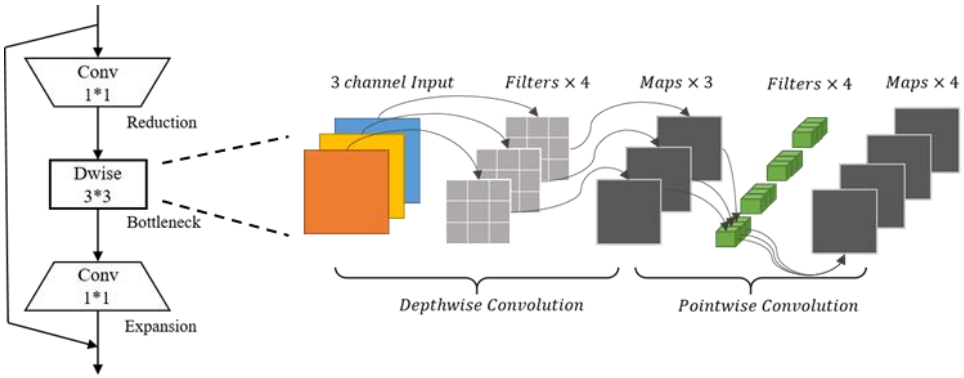

Fig.1 Inverted residual block structure and depth-wise separable convolution process

2.2. Loss layer

Bounding Box Loss

The intersection over union *(IoU)* is used to represent the ratio of the intersection of the object bounding box and the prediction bounding box to the union. *IoU* is the common measurement method for the evaluation index of object detection. Based on the invariance of the *IoU* scale, *CIoU* integrates the distance between the object and prediction bounding boxes and the degree of overlap, and uses the ratio of the length and width of the prediction bounding box as a penalty item. It makes the prediction result more stable. The schematic diagram of *CIOU* is shown in Fig.2

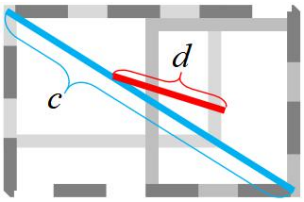

Fig.2 The schematic diagram of *CIoU*

CIoU formula：

$$CIoU = IoU - \frac{\rho^2(b,b^{gt})}{c^2} - \alpha v \tag{1}$$

$$\alpha = \frac{v}{1-IOU+v} \tag{2}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \tag{3}$$

Where $d = \rho^2(b,b^{gt})$ is the Euclidean distance between the center point of the object and prediction bounding box, $b = (x,y,w,h)$ represents the object bounding box (ground truth), $b^{gt} = (x^{gt},y^{gt},w^{gt},h^{gt})$ is the information of the prediction bounding box. c is the diagonal distance of the smallest area containing the object and prediction bounding boxes.

The modified bounding box loss:

$$Loss_{CIoU} = \sum_{i=0}^{S\times S}\sum_{j=0}^{k} I_{ij}^{obj}\left[1 - IoU + \frac{\rho^2(b,b^{gt})}{c^2} + \alpha v\right] \tag{4}$$

Confidence Loss

This paper introduces the focal loss instead of the original binary cross-entropy loss to solve the problem of unbalanced training samples. By reducing the weight of negative samples, the model is trained to focus more on samples that are difficult to classify, avoiding the positive sample despised in the loss function.

The calculation of confidence loss is divided into two steps: the first is to determine which of the $S \times S \times 3$ prediction bounding boxes are counter examples; the second is to calculate the focal loss.

Focal loss is modified on the basis of the cross-entropy loss function, and the expression is as follows:

$$Loss_{conf} = -ylny^{'} - (1-y)\,ln\left(1-y^{'}\right) = \begin{cases} -lny^{'}, & m=1 \\ -l\left(1-y^{'}\right), & m=0 \end{cases} \tag{5}$$

m represents whether the bounding box is a counterexample (“0” for true, “1” for false).

Class loss

The class loss uses Sigmoid cross entropy. Where y_{ij} represents the true classification probability value, if it is a certain class, it is 1, otherwise it is 0. p_{ij} is the predicted value.

$$p_{ij} = sigmoid\left(logits_{ij}\right) = \frac{1}{1+e^{-logits_{ij}}} \tag{6}$$

$$Loss_{class} = -\sum_{i=0}^{S\times S}\sum_{j=0}^{k}\left[y_{ij}\,ln\,p_{ij} + \left(1-y_{ij}\right)ln\left(1-p_{ij}\right)\right] \tag{7}$$

In summary, the total loss is the sum of the above three.

$$Loss = Loss_{CIoU} + Loss_{conf} + Loss_{class} \quad (8)$$

3. Experimental results and analyses

Here, we describe the experiments and results which we realized. Under the same initial conditions, we tested multiple models for comparison. All training is performed on GTX1080Ti but the inference is run on RTX2060.

3.1. Parameter settings

We use 416×416 images as input for both training and evaluation. In order to ensure the accuracy and stability of the results, we conduct multiple training sessions on a GPU with 4 GTX1080Ti. The initial batch size is set to 16, the learning rate is 1e-5, after 10 epochs of warm-up, learning rate is adjusted to 1e-3 to accelerate training. We also add the automatic learning rate decay and early stop mechanism to monitor the training loss at any time to improve the over-fitting problem to a certain extent. All the models discussed in this paper are trained without any pre-training weights.

3.2. Dataset

The dataset comes from 39,437 video surveillance images provided by Shenzhen Softsz Co., Ltd., of which 35044 pictures are used for training, 3893 pictures are used for verification, and 500 pictures are used for effect evaluation. The dataset mainly contains two types of marked object, pedestrians and vehicles. The real scenes come from the security surveillance deployed in an urban area in China, including open scenes such as highways and streets, and complex scenes with multiple pedestrians such as markets, schools, and hospital entrances. It also includes night, rain, snow weather and other complex background scenes. The original image size is 1920×1080, adjusted to 416×416 during training. In addition, random filling, cropping, translation, and horizontal flipping methods are added during training to expand the dataset.

3.3. Result

Fig.3 shows the detection effect and loss curve of Smart-YOLO. It can be seen that basically all objects can be detected and the loss function optimization speed is faster than YOLOv3.

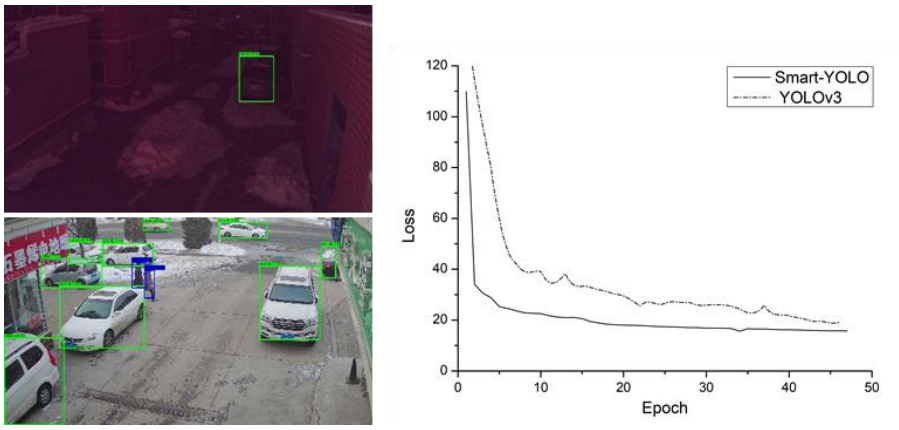

Fig.3 Detection results and loss function curve

Tab.1 shows the model size, mAP and inference speed obtained by Smart-YOLO and other methods (YOLOv3, YOLOv3-Tiny, YOLOv4, YOLOv4-Tiny, YOLOv3-prune and YOLOv3+Mobilenet) using the same training set and test set.

(1) Compared with the lightweight models with similar model weights, the model size of Smart-YOLO (28.9MB) is slightly better than YOLOv3-Tiny, but the mAP of Smart-YOLO is 20.14% higher than that of YOLOv3-Tiny, and it is 10.67% higher than YOLOv3-prune with a model size of 79.1MB. By improving the structure and loss function of YOLOv3, Smart-YOLO is basically close to the detection accuracy and speed of YOLOv4-Tiny, which almost has the best performance in the current lightweight network.

(2) Compared with larger models Smart-YOLO has lost about 10-20% of mAP, and the model size and detection speed have been qualitatively improved. Smart-YOLO is about 8.1 times smaller in model and almost 4.5 times faster than YOLOv3. Compared with YOLOv4, the model is about 8.4 times smaller and the speed is increased by about 2.8 times. By contrast, Smart-YOLO is almost 3.2 times smaller and 2.3 times faster than YOLOv3+Mobilenet.

Method	Resolution	Model Size	mAP	FPS
YOLOv3	416×416	235MB	60.03	15
YOLOv3-Tiny	416×416	33.2MB	27.23	72
YOLOv3+Mobilenet	416×416	92.6MB	58.59	30
YOLOv4	416×416	244mb	69.83	25
YOLOv4-Tiny	416×416	22.6mb	47.51	77
YOLOv3-prune	416×416	79.1mb	36.69	22
Smart-YOLO	416×416	28.9mb	47.36	68
Smart-YOLO	608×608	28.9mb	60.31	40
Smart-YOLO	320×320	28.9mb	36.07	91

Tab.1 The results of 7 models on the same test set

Comparing the size and speed of the model, it can be seen that Smart-YOLO has an obvious lightweight effect, and the detection speed is significantly better than high-precision models such as YOLOv3 and YOLOv4. Combined with actual needs, the detection accuracy achieved by Smart-YOLO can basically meet the needs of on-site object detection. In summary, Smart-YOLO is more suitable for edge devices with less memory, less computing resources and high real-time requirements because of its smaller model, fewer parameters, and faster detection speed.

4. Conclusion

This paper proposes a new real-time object detection algorithm Smart-YOLO based on the YOLO framework. It is small in model magnitude, fast in detection speed, and more suitable for promotion to some edge devices or mobile end devices. As an improvement under the YOLO framework, Smart-YOLO replaces the original backbone network with deep-wise separable convolutions and inverted bottleneck blocks, which reduces the model size by more than 8 times and increases the detection speed by more than 4 times. Then, in order to optimize the prediction bounding box regression and solve the problem of unbalanced training samples, we also use *CIoU* measure and focal loss based on the original loss function of YOLOv3, and add some data enhancement methods to expand the training data to a certain extent. The above methods can speed up the training process, reduce the difficulty of model training, and improve the accuracy of detecting objects. Finally, the experimental results show that Smart-YOLO has a faster detection speed and a smaller model size than other methods. It can indeed strike a better trade-off between accuracy and efficiency in certain scenarios with high real-time requirements.

Acknowledgments

This work was supported by the Tianjin Municipal Science and Technology Bureau under Grant No. 18YFZCGX00360.

References

[1] Andrew G, Menglong Z, Bo C, Dmitry K, Weijun W, Tobias W, Marco A and Hartwig A 2017 Mobilenets: Efficient convolutional neural networks for mobile vision applications *CoRR*. abs/1704.04861

[2] Mark S, Andrew G, Menglong Z, Andrey Z and Liang-Chieh C 2018 Mobilenetv2: Inverted residuals and linear bottlenecks. mobile networks for classification, detection and segmentation *CoRR*. abs/1801.04381

[3] Yunyang X, Hanxiao L, Suyog G, Berkin A, Gabriel B, Pieter-Jan K, Mingxing T, Vikas S and Bo C 2020 MobileDets: Searching for Object Detection Architectures for Mobile Accelerators arXiv:2004.14525

[4] Hurtik P, Molek V, Hula J, Vajgl M, Vlasanek P and Nejezchleba T 2020 Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3 *arXiv e-prints.* arXiv:2005.13243

[5] Xiaoyan C, Jianyong C, Xiaoguang H, Chundong Z, Dongyang Z, Kuifeng Z and Yanjie S 2020 A Light-Weighted CNN Model for Wafer Structural Defect Detection *IEEE Accessp* **8** pp24006-24018

[6] Tanvir A, Yinglong M, Muhammad Y, Belal A, Shah N and Amin ul H 2020 Object Detection through Modified YOLO Neural Network *Scientific Programming*

[7] Joseph R, Divvala S, Girshick R and Farhadi A You only look once: unified, real-time object detection *2016 USA CVPR* pp779–788

[8] Joseph R and Farhadi 2018 A Yolov3: An incremental improvement arXiv:1804.02767

[9] Tsung-Yi L, Goyal P, Girshick R, Kaiming H and Dollar P 20018 Focal Loss for Dense Object Detection *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42(2)** pp318-327

[10] Zhaohui Z, Ping W, Dongwei R, Wei L, Rongguang Y, Qinghua H and Wangmeng Z 2020 Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation arXiv:2005.03572