YOLOv4: Optimal Speed and Accuracy of Object Detection

Chen Hui

Abstract

Abstract.In recent years, with the emergence of deep learning technology represented by convolutional neural networks, convolutional neural networks have achieved significant success in object detection. YOLO (You only look once) has the advantage of being fast while maintaining a certain level of accuracy, so YOLO is developing rapidly and has many applications in the field of object detection. YOLOv4 is an improved version of YOLOv3, which improves some details on the basis of YOLOv3 and makes a good improvement in speed and accuracy.

Keywords： Object detection, Data augmentation,Deep learning.

1 Introduction

The main goal of YOLOv4 is designing a fast operating speed of an object detector in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). YOLOv4 can be easily trained and used. For example, anyone who uses a conventional GPU to train and test can achieve real-time, high quality, and convincing object detection results, as the YOLOv4 results shown in Figure 1.This paper is based on the replication of YOLOv4, specifically on running through the YOLOv4 code, understanding the structure of YOLOv4, understanding the inputs and outputs, and improving on the features already implemented in YOLOv4.

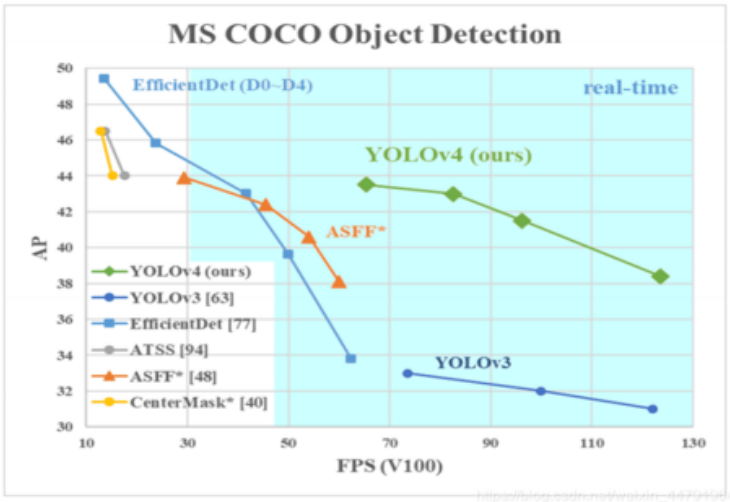

Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

2 Related works

2.1 Traditional object detection

A modern detector is usually composed of two parts, a backbone which is pre-trained on ImageNet and a head which is used to predict classes and bounding boxes of objects. For those detectors running on GPU platform, their backbone could be VGG[1], ResNet[2], ResNeXt[3], or DenseNet[4]. For those detectors running on CPU platform, their backbone could be SqueezeNet[5], MobileNet[6-9], or ShuffleNet[10-11]. As to the head part, it is usually categorized into two kinds, i.e., one-stage object detector and two-stage object detector. The most representative two-stage object detector is the R-CNN series, including fast R-CNN, faster R-CNN, R-FCN, and Libra R-CNN . It is also possible to make a two-stage object detector an anchor-free object detector, such as RepPoints. As for one-stage object detector, the most representative models are YOLO , SSD , and RetinaNet . In recent years, anchor-free one-stage object detectors are developed. The detectors of this sort are CenterNet , CornerNet , FCOS, etc. Object detectors developed in recent years often insert some layers between backbone and head, and these layers are usually used to collect feature maps from different stages. We can call it “the neck of an object detector.” Usually, a neck is composed of several bottom-up paths and several top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN), Path Aggregation Network (PAN), BiFPN, and NAS-FPN.

2.2 YOLOv4 improvements

YOLOv4 develope an efficient and powerful object detection model. It makes everyone can use a 1080 Ti or 2080 Ti GPU to train a super fast and accurate object detector. YOLOv4 verify the influence of state-of-the-art Bag-of-Freebies and Bag-of-Specials methods of object detection during the detector training.YOLOv4 modify state-of-the-art methods and make them more effecient and suitable for single GPU training, including CBN[12], PAN[13], SAM[14], etc.

3 Method

3.1 Overview

The structure of object detector is shown in Figure3：

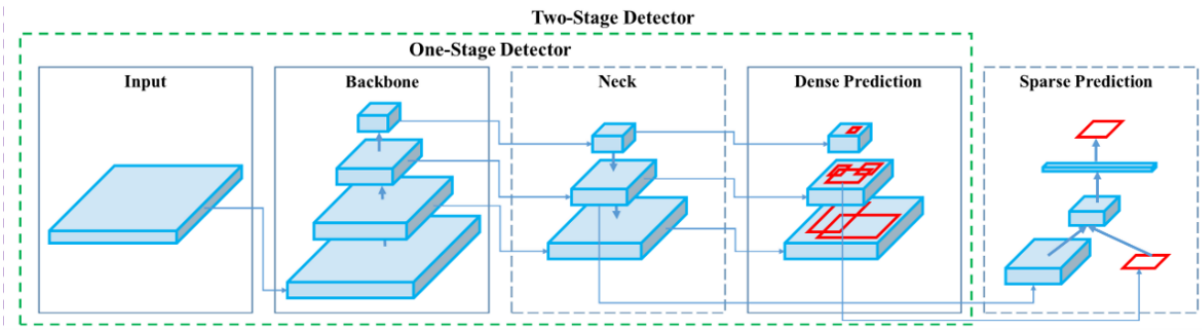

Figure 2: structure of object detector

3.2 Feature extraction

The backbone feature extraction network is shown in Figure3：

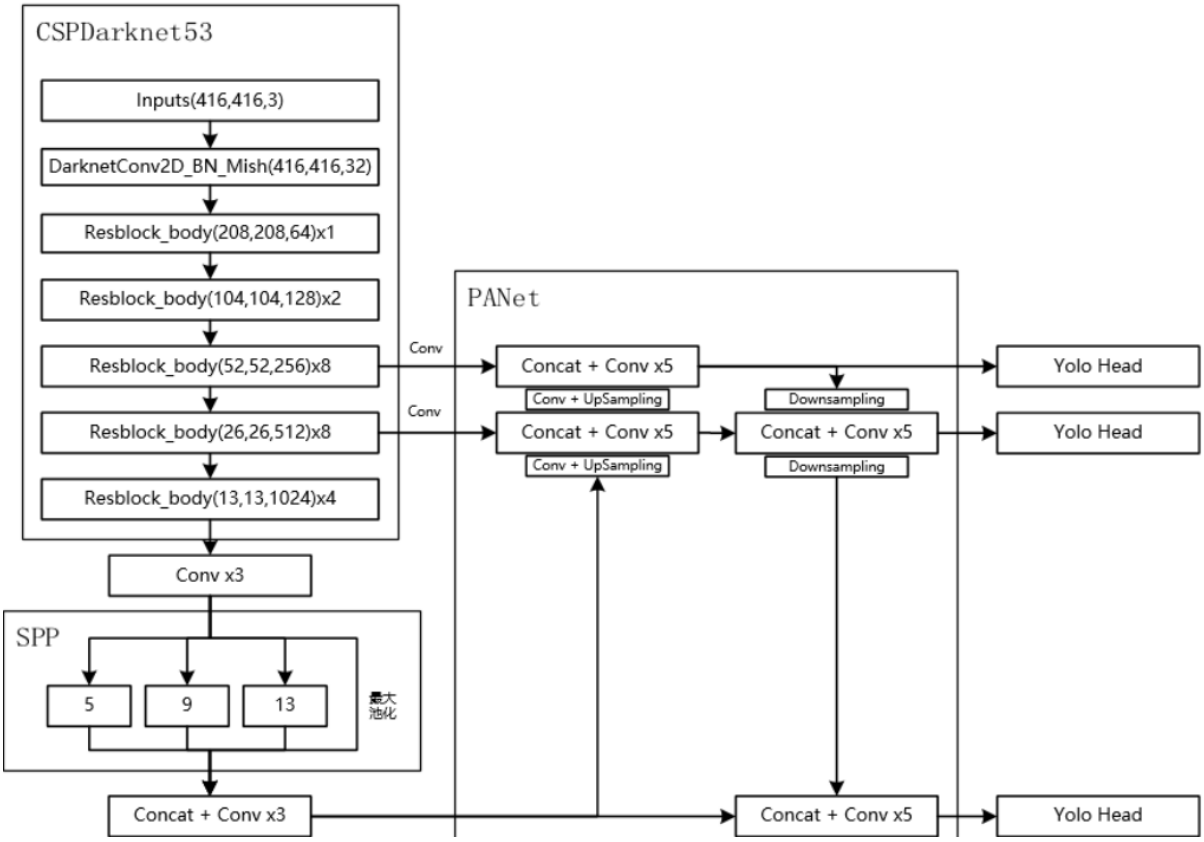

Figure 3: structure of backbone

3.3 Loss

$$\mathrm{LOSS_{CIOU}} = 1 - \mathrm{IOU} + \frac{\rho^2\left(\mathrm{b},\ \mathrm{b^{gt}}\right)}{\mathrm{c}^2} + \alpha \mathrm{v} \tag{1}$$

$$\rho^2\left(\mathrm{b},\ \mathrm{b^{gt}}\right) \tag{2}$$

$$\alpha = \frac{\mathrm{v}}{1 - \mathrm{IOU} + \mathrm{v}} \tag{3}$$

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

Equation 2 represents the Euclidean distance between the center points of the prediction frame and the real frame. c represents the diagonal distance of the smallest closed area that can contain both the predicted and real boxes.

4 Implementation details

4.1 Comparing with released source codes

The SPP structure, which is pooled by pooling nuclei of different sizes, can greatly increase the receptive field and isolate the most prominent context features.

Procedure 1 SPP structure

Input: pools of different sizes X,
Output: features Y
for *poolsize* ***in*** *poolsizes* **do**
| $maxpools$ = MaxPool2d(*poolsize*, 1, *poolsize*/2)
end
for *maxpool* ***in*** *maxpools[::-1]* **do**
| $features$ = maxpool(x) $features$ = torch.cat(*features* + [*x*], dim = 1)
end

4.2 Experimental environment setup

Install a requirements.txt compliant environment

4.3 Interface design

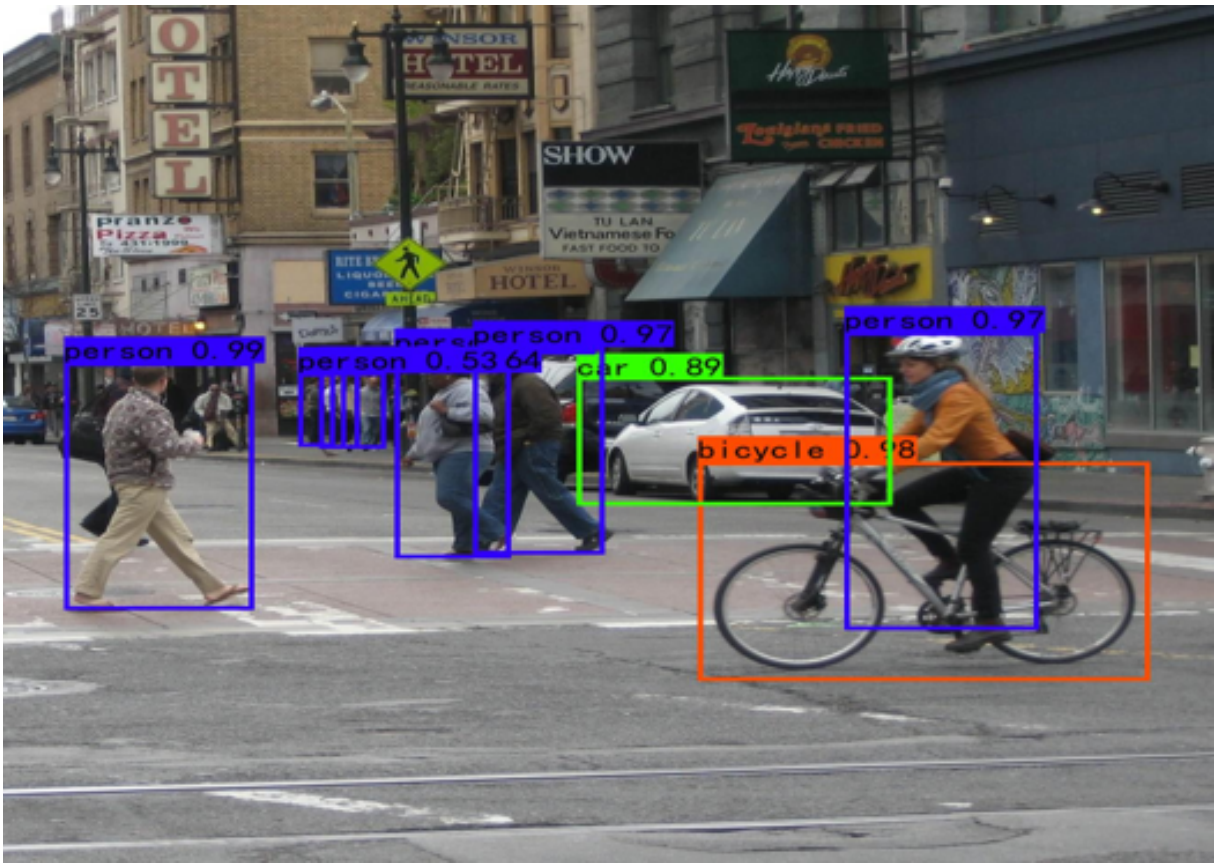

Figure 4: Interface

4.4 Main contributions

The data enhancement utilizes four images, which according to the paper has the great advantage of enriching the background of the detected objects and calculating the data of four images at once during the BN calculation.

5 Results and analysis

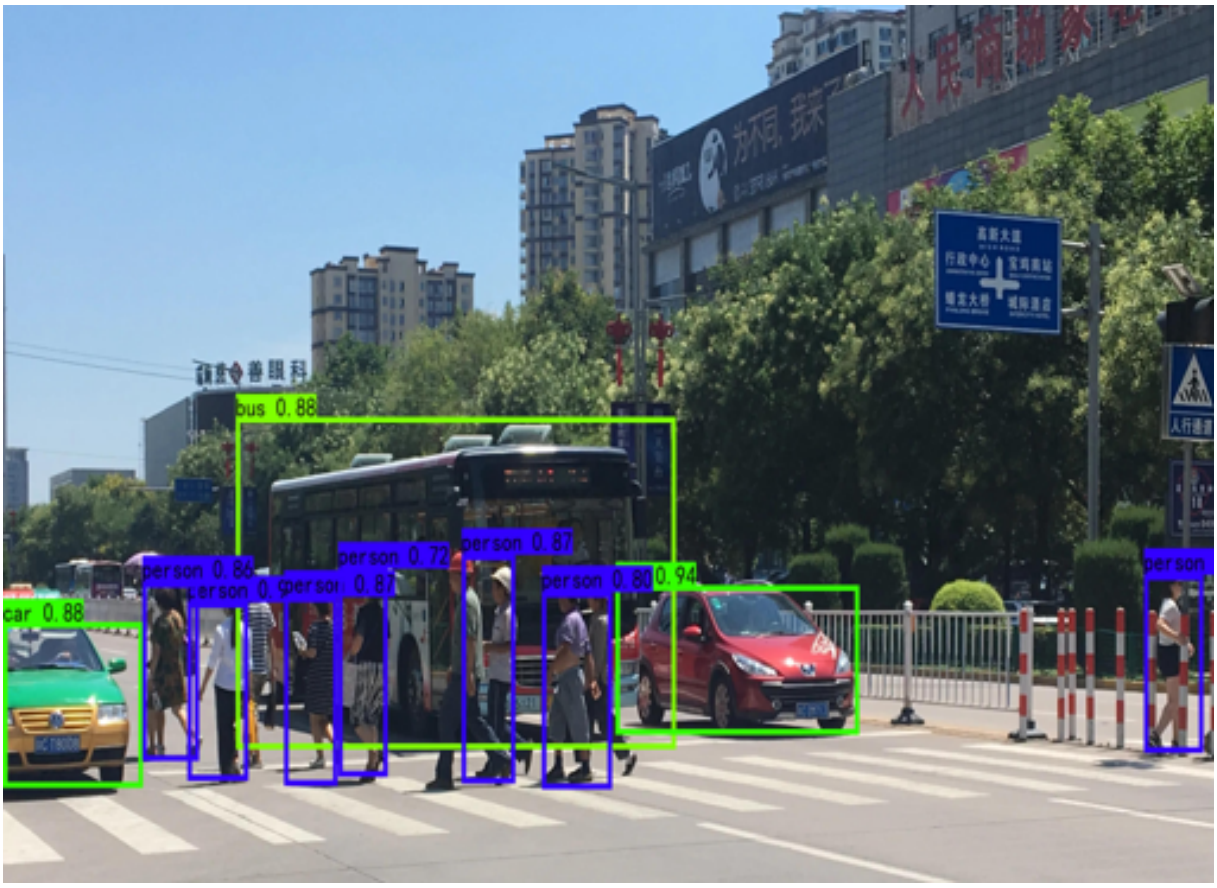

Figure 5: Experimental results

```
Get map.
94.41% = aeroplane AP   ||  score_threhold=0.5 : F1=0.90 ; Recall=84.17% ; Precision=96.19%
86.33% = bicycle AP     ||  score_threhold=0.5 : F1=0.73 ; Recall=58.73% ; Precision=96.10%
84.65% = bird AP    ||  score_threhold=0.5 : F1=0.69 ; Recall=53.08% ; Precision=97.87%
72.36% = boat AP    ||  score_threhold=0.5 : F1=0.42 ; Recall=27.13% ; Precision=94.59%
70.38% = bottle AP  ||  score_threhold=0.5 : F1=0.57 ; Recall=42.75% ; Precision=87.20%
87.81% = bus AP     ||  score_threhold=0.5 : F1=0.77 ; Recall=63.46% ; Precision=97.06%
92.45% = car AP     ||  score_threhold=0.5 : F1=0.87 ; Recall=79.32% ; Precision=96.45%
95.52% = cat AP     ||  score_threhold=0.5 : F1=0.90 ; Recall=85.33% ; Precision=95.15%
74.56% = chair AP   ||  score_threhold=0.5 : F1=0.67 ; Recall=54.25% ; Precision=88.93%
87.50% = cow AP     ||  score_threhold=0.5 : F1=0.60 ; Recall=42.86% ; Precision=97.67%
67.20% = diningtable AP     ||  score_threhold=0.5 : F1=0.45 ; Recall=30.30% ; Precision=85.71%
91.27% = dog AP     ||  score_threhold=0.5 : F1=0.84 ; Recall=76.25% ; Precision=93.37%
93.01% = horse AP   ||  score_threhold=0.5 : F1=0.88 ; Recall=80.28% ; Precision=97.44%
93.09% = motorbike AP   ||  score_threhold=0.5 : F1=0.84 ; Recall=73.24% ; Precision=97.20%
91.81% = person AP  ||  score_threhold=0.5 : F1=0.87 ; Recall=82.37% ; Precision=91.84%
58.78% = pottedplant AP     ||  score_threhold=0.5 : F1=0.19 ; Recall=10.68% ; Precision=89.29%
84.96% = sheep AP   ||  score_threhold=0.5 : F1=0.52 ; Recall=35.62% ; Precision=96.30%
70.11% = sofa AP    ||  score_threhold=0.5 : F1=0.46 ; Recall=31.18% ; Precision=87.88%
94.82% = train AP   ||  score_threhold=0.5 : F1=0.86 ; Recall=79.61% ; Precision=94.25%
87.45% = tvmonitor AP   ||  score_threhold=0.5 : F1=0.60 ; Recall=44.44% ; Precision=93.15%
mAP = 83.92%
Get map done.
```

Figure 6: Experimental results

6 Conclusion and future work

YOLOv4 is a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO $AP_50...95$ and AP_50 than all available alternative detectors. The detector described can be trained and used on a conventional GPU with 8-16 GB-VRAM this makes its broad use possible. The original concept of one-stage anchor-based detectors has proven its viability. We have verified a large number of features, and selected for use such of them for improving the accuracy of both the classifier and the detector. These features can be used as best-practice

for future studies and developments.

References

[1] SIMONYAN K, ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. arXiv e-prints, 2014, arXiv:1409.1556: arXiv:1409.1556. arXiv: 1409.1556 [cs.CV]. DOI: 10.48550/arXiv.1409.1556.

[2] 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 (CVPR). 2016.

[3] XIE S, GIRSHICK R, DOLLAR P, et al. Aggregated Residual Transformations for Deep Neural Networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[4] HUANG G, LIU Z, van der MAATEN L, et al. Densely Connected Convolutional Networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[5] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size[J]. arXiv e-prints, 2016, arXiv:1602.07360: arXiv:1602.07360. arXiv: 1602.07360 [cs.CV]. DOI: 10.48550/arXiv.1602.07360.

[6] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J]. arXiv e-prints, 2017, arXiv:1704.04861: arXiv:1704.04861. arXiv: 1704.04861 [cs.CV]. DOI: 10.48550/arXiv.1704.04861.

[7] SANDLER M, HOWARD A, ZHU M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[8] HOWARD A, SANDLER M, CHU G, et al. Searching for MobileNetV3[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2019.

[9] TAN M, CHEN B, PANG R, et al. MnasNet: Platform-Aware Neural Architecture Search for Mobile [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[10] ZHANG X, ZHOU X, LIN M, et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[11] MA N, ZHANG X, ZHENG H T, et al. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[12] YAO Z, CAO Y, ZHENG S, et al. Cross-Iteration Batch Normalization[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021: 12331-12340.

[13] LIU S, QI L, QIN H, et al. Path Aggregation Network for Instance Segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[14] WOO S, PARK J, LEE J Y, et al. CBAM: Convolutional Block Attention Module[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018.