YOLOv4

YOLOv4: Optimal Speed and Accuracy of Object Detection

A

OWOP season1 04
조유민

Introduction

YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP_{50}) for the MS COCO dataset at a real-time speed of ~65 FPS on Tesla V100. Source code is at https://github.com/AlexeyAB/darknet.

1. Introduction

The majority of CNN-based object detectors are largely applicable only for recommendation systems. For example, searching for free parking spaces via urban video cameras is executed by slow accurate models, whereas car collision warning is related to fast inaccurate models. Improving the real-time object detector accuracy enables using them not only for hint generating recommendation systems, but also for stand-alone process management and human input reduction. Real-time object detector operation on conventional Graphics Processing Units (GPU) allows their mass usage at an affordable price. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size. We address such problems through creating a CNN that operates in real-time on a conventional GPU, and for which training requires only one conventional GPU.

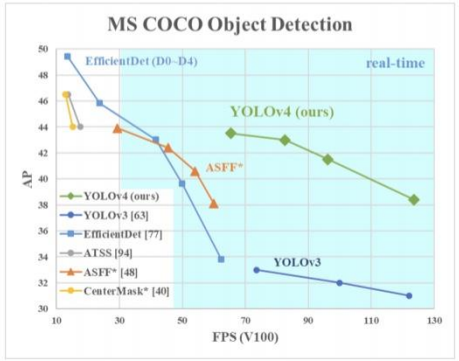
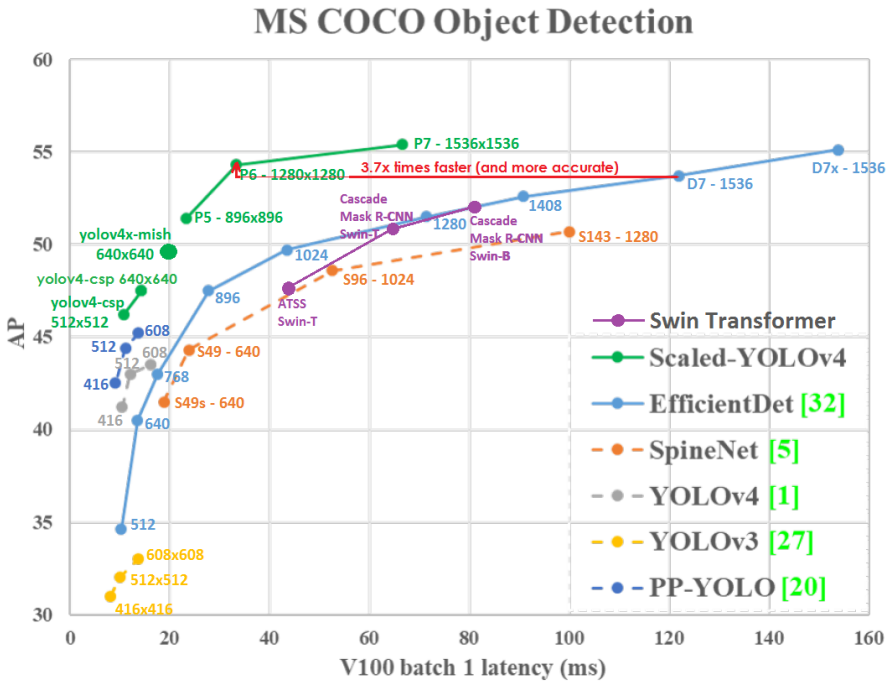


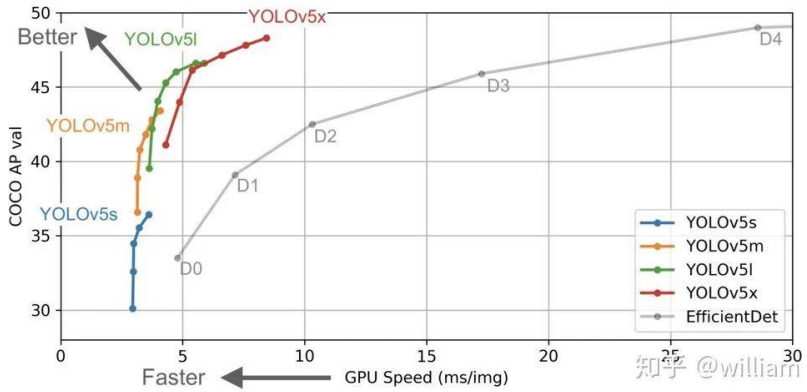
Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

The main goal of this work 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). We hope that the designed object 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. Our contributions are summarized as follows:

1. We 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.
2. We verify the influence of state-of-the-art Bag-of-Freebies and Bag-of-Specials methods of object detection during the detector training.
3. We modify state-of-the-art methods and make them more effecient and suitable for single GPU training, including CBN [89], PAN [49], SAM [85], etc.

YOLOv4 vs YOLOv5





Architecture

YOLOv4 = CSPDarknet53 (BackBone) + SPP + PANet (Neck) + YOLOv3 (Head)

BoF

Regularization - DropBlock

Data Augmentation – CutMix, Mosaic

Imbalance Sampling – Class label smoothing

BoS

Activation - Mish

Skip Connections– CSP, MiWRC

BoF

Bbox regression loss - CIoU-loss

Normalization - CmBN

Regularization - DropBlock

Data Augmentation – Mosaic, SAT

Hyperparameter tuning– Genetic algorithms

Etc.

BoS

Activation - Mish

Receptive Field – SPP

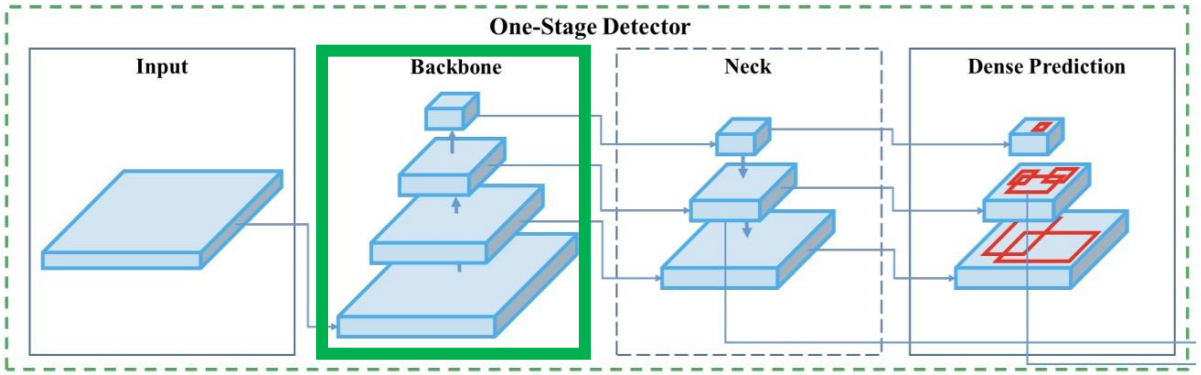
Attention – SAM

Feature Integration - PAN

Post processing– DIoU-NMS

Architecture - Backbone

YOLOv4 = **CSPDarknet53 (BackBone)** + **?** + **?**



Backbone

Feature-extraction

VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50, **CSPDarknet53**

Neck

Feature-extraction from different stages of Backbone(Refinement and Reconfiguration)

Additional blocks - SPP, ASPP, RFB, SAM

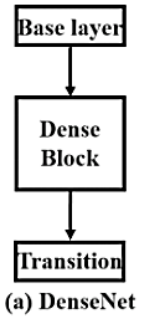
Path-aggregation blocks - FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM

CSPNet

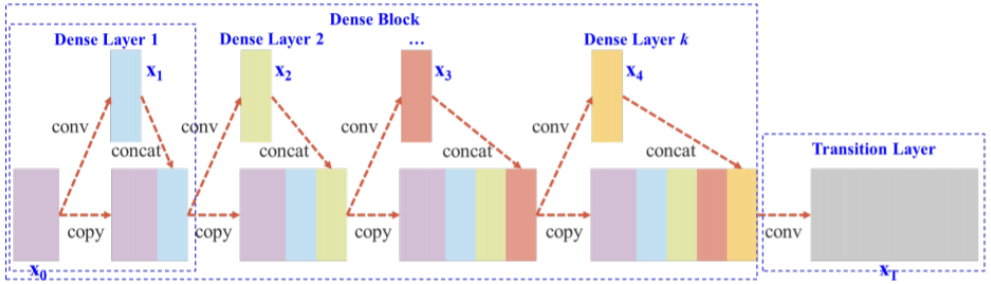
Problem

Duplicate gradient information within optimization

It requires heavy inference computations



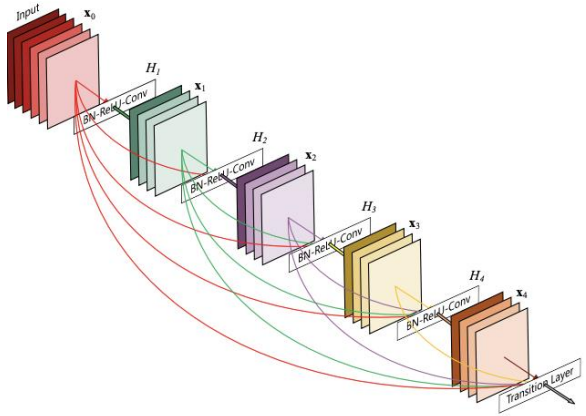
(a) DenseNet



(a) DenseNet

$$x_1 = w_1 * x_0$$
$$x_2 = w_2 * [x_0, x_1]$$
$$\vdots$$
$$x_k = w_k * [x_0, x_1, \ldots, x_{k-1}]$$

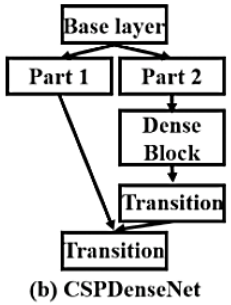
$$w_1' = f(w_1, g_0)$$
$$w_2' = f(w_2, g_0, g_1)$$
$$w_3' = f(w_3, g_0, g_1, g_2)$$
$$\vdots$$
$$w_k' = f(w_k, g_0, g_1, g_2, \ldots, g_{k-1})$$



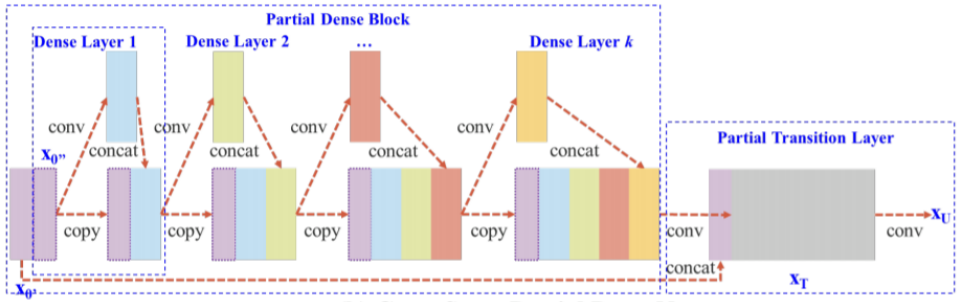
Solution

Integrate feature maps from the beginning and the end

It prevents the duplicate problem by truncating the gradient flow



(b) CSPDenseNet



(b) Cross Stage Partial DenseNet

$$x_k = w_k * [x_{0''}, x_1, \ldots, x_{k-1}]$$
$$x_T = w_T * [x_{0''}, x_1, \ldots, x_k]$$
$$x_U = w_U * [x_{0'}, x_T]$$

$$w_k' = f(w_k, g_{0''}, g_1, g_2, \ldots, g_{k-1})$$
$$w_T' = f(w_T, g_{0''}, g_1, g_2, \ldots, g_k)$$
$$w_U' = f(w_U, g_{0'}, g_T)$$

cf.) CSPResNet

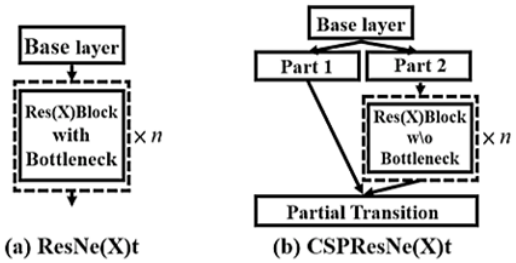
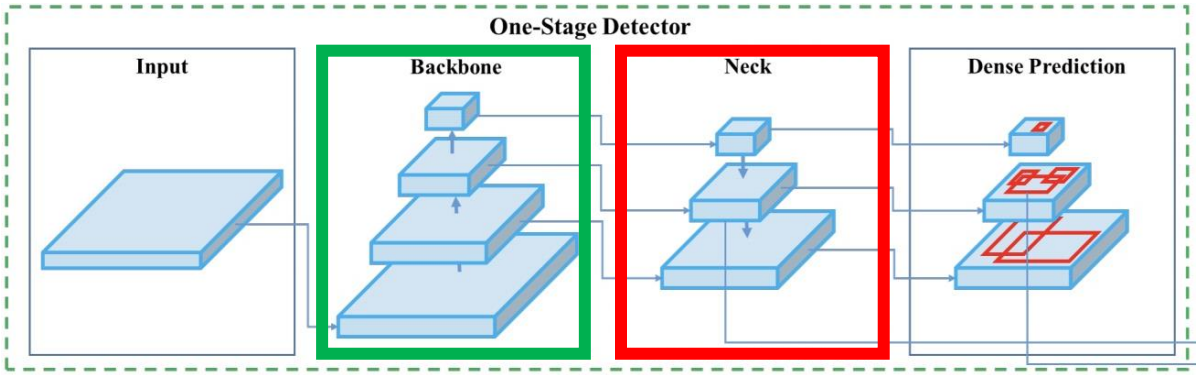


Figure 5: Applying CSPNet to ResNe(X)t.

Architecture - Neck

YOLOv4 = **CSPDarknet53 (BackBone)** + **SPP + PANet (Neck)** + **?**



Backbone

Feature-extraction

VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50,

Neck

Feature-extraction from different stages of Backbone(Refinement and Reconfiguration)

Additional blocks - **SPP**, ASPP, RFB, SAM

Path-aggregation blocks - FPN, **PAN**, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM

Selection of Backbone and Neck

Problem

To detect multiple objects of different sizes in a single image, model needs larger receptive field and many parameters

Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

Solution

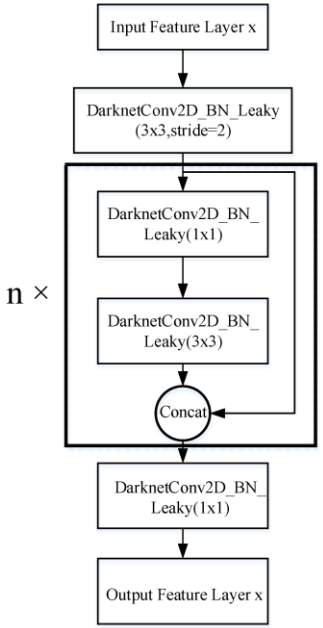
1) We can choose Backbone which has higher resolution, more layers and parameters

YOLOv2 - Darknet19

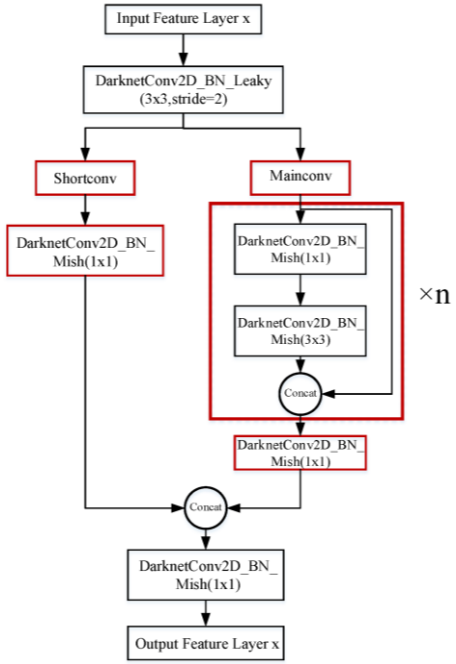
Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

YOLOv3 - Darknet53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			



YOLOv4 - CSPDarknet53

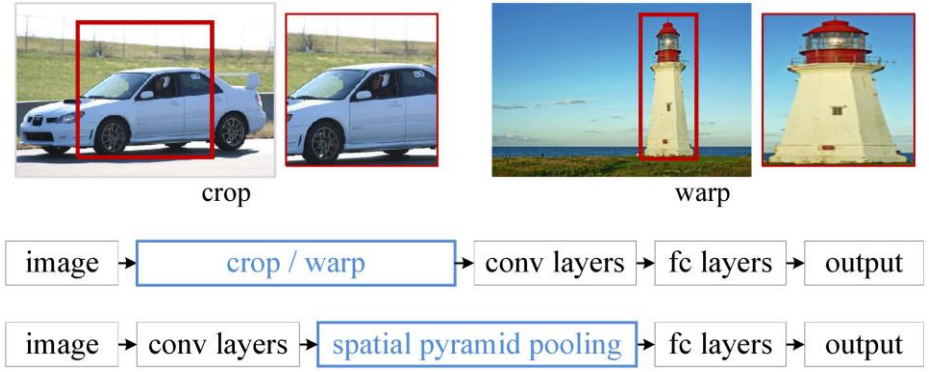


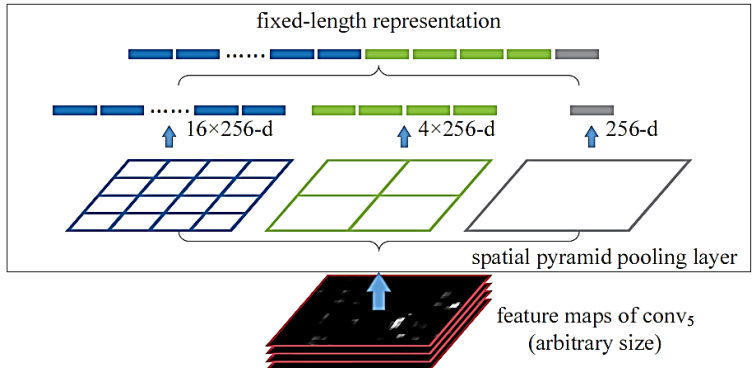
SPP(Spatial Pyramid Pooling)

Solution

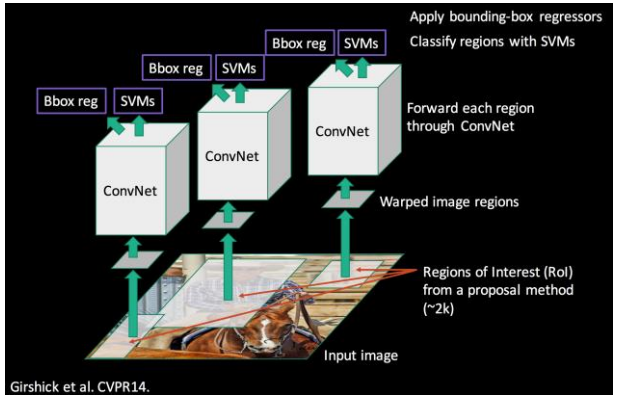
2) Add SPP-blocks on backbone to increase receptive field

SPP(Spatial Pyramid Pooling)

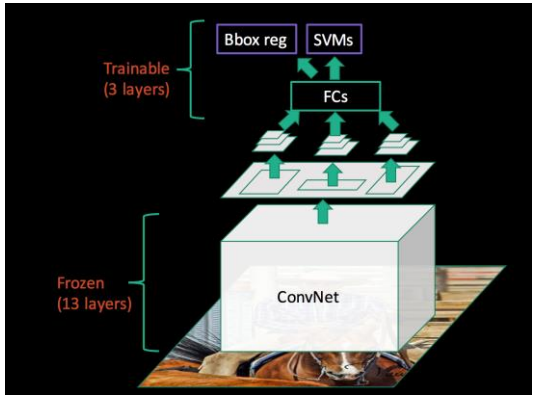




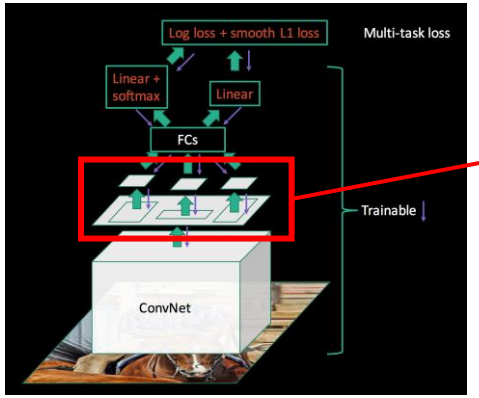
R-CNN



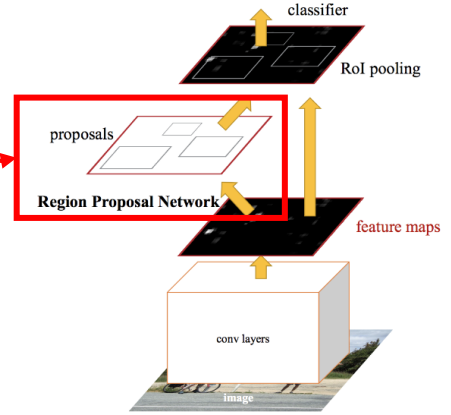
SPPNet



Fast R-CNN

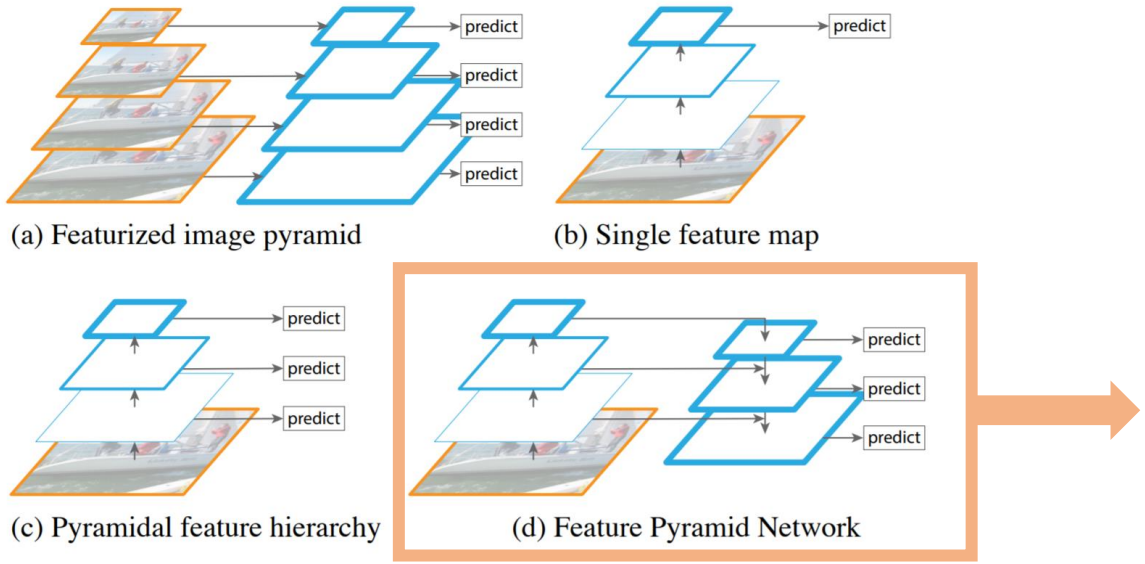


Faster R-CNN

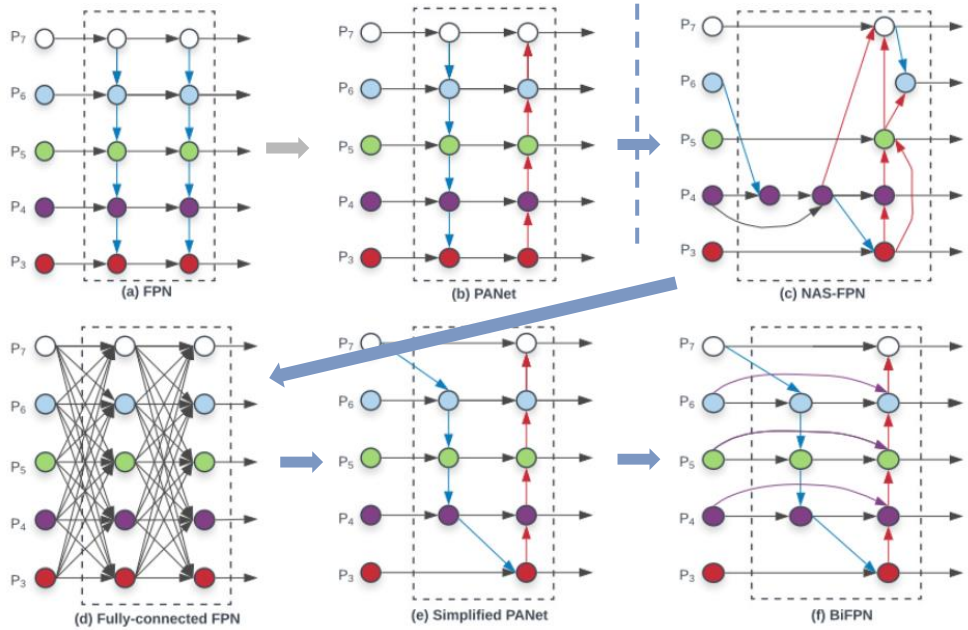


Multi-scale prediction

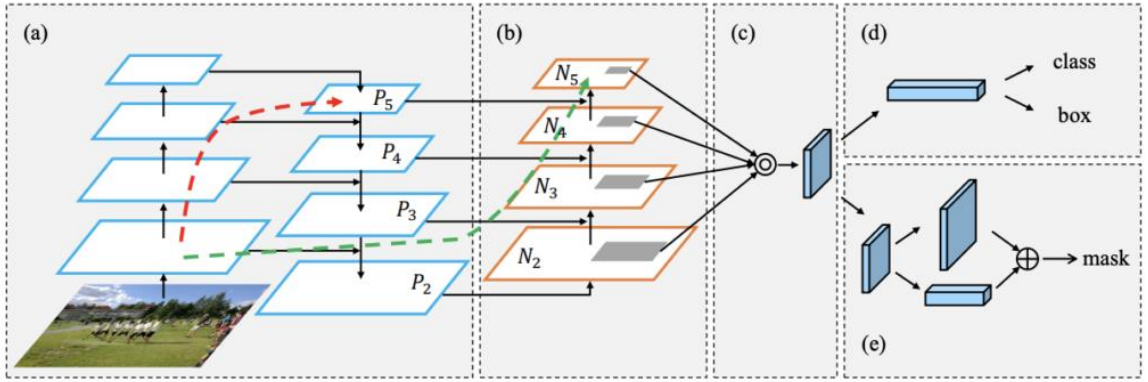
FPN



Cross-scale Connection / Feature Fusion



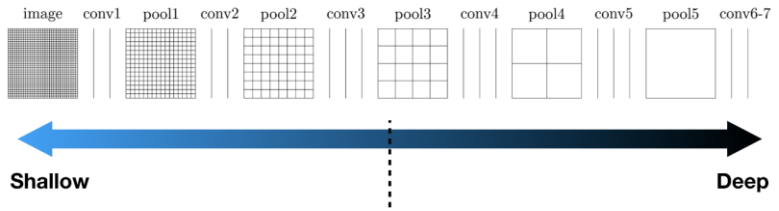
PAN(PANet)



Feature Integration

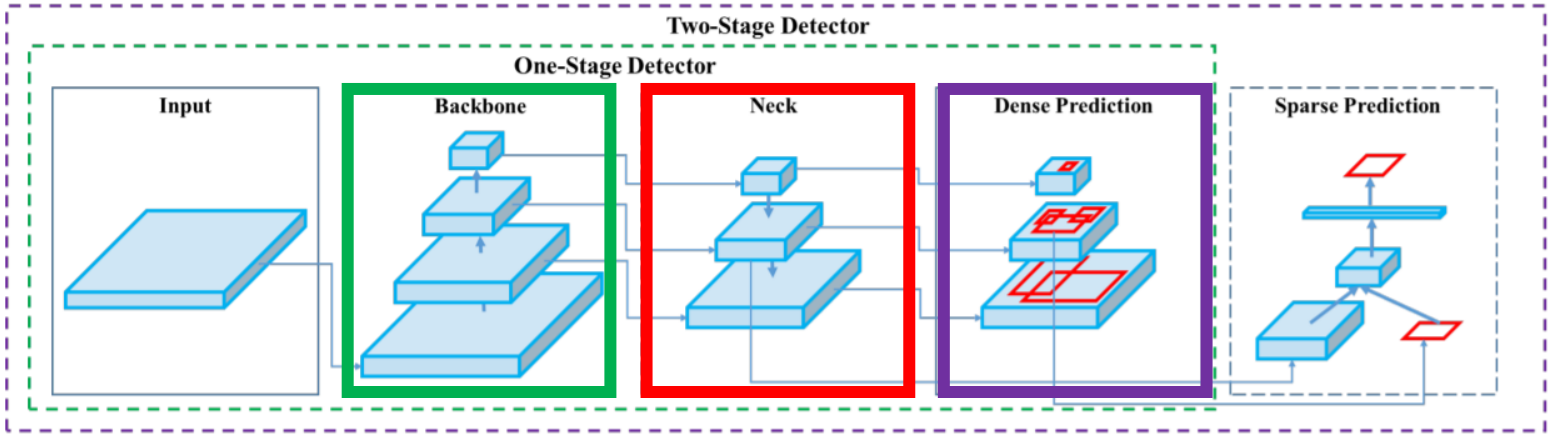
Integrate different feature pyramid

low level physical feature + high-level semantic feature



Architecture - Head

YOLOv4 = **CSPDarknet53 (BackBone)** + **SPP + PANet (Neck)** + **YOLOv3 (Head)**



Sparse Prediction Head

2-stage detector

RCNN series, RedPoints

Dense Prediction Head

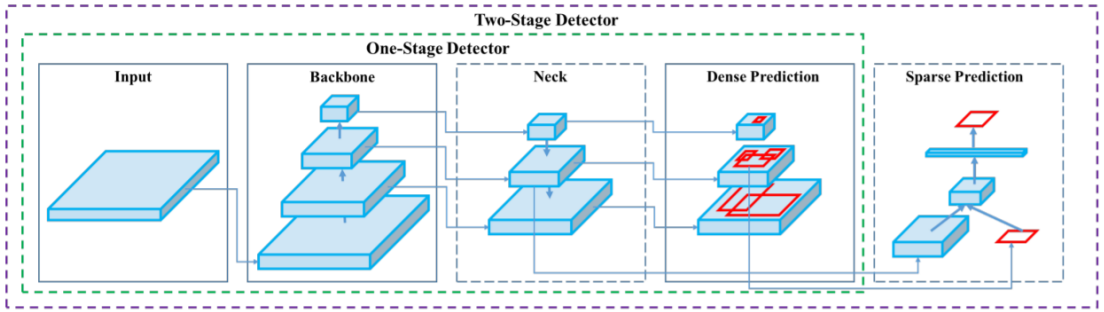
1-stage detector

SSD, YOLO series, RetinaNet, CenterNet, CornerNet, FCOS

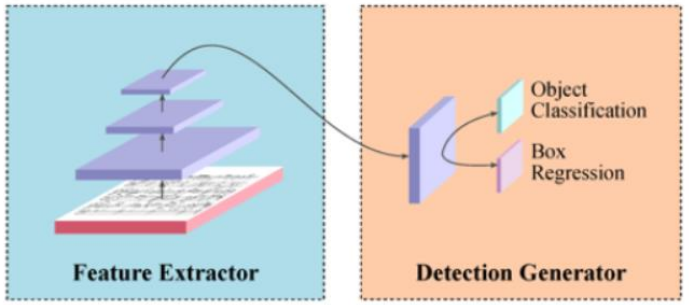
Object Detection Model

Task of Object Detection

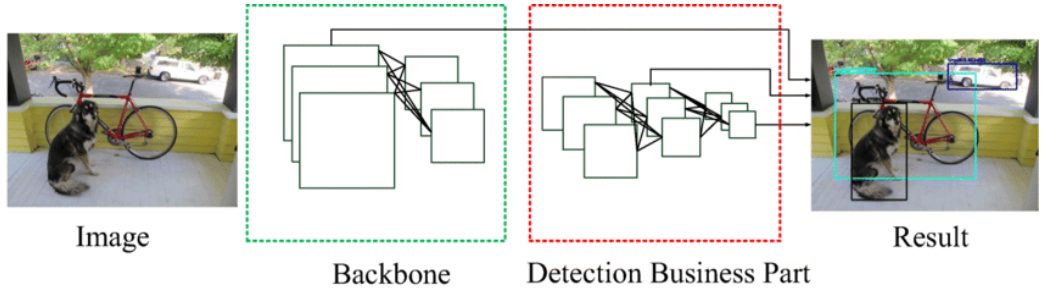
1) Localization – Region proposal & Bbox regression
2) Classification – Predict classes



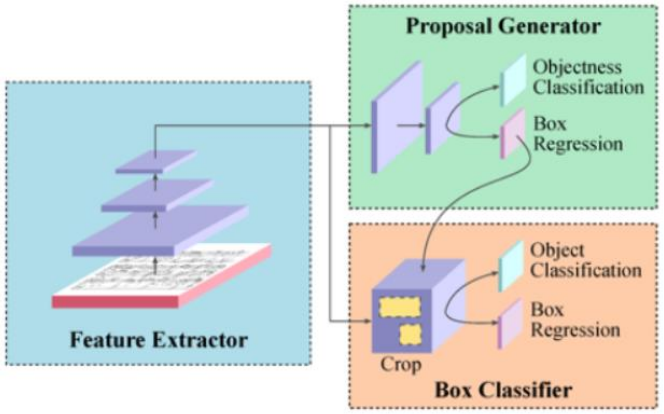
1-stage detector



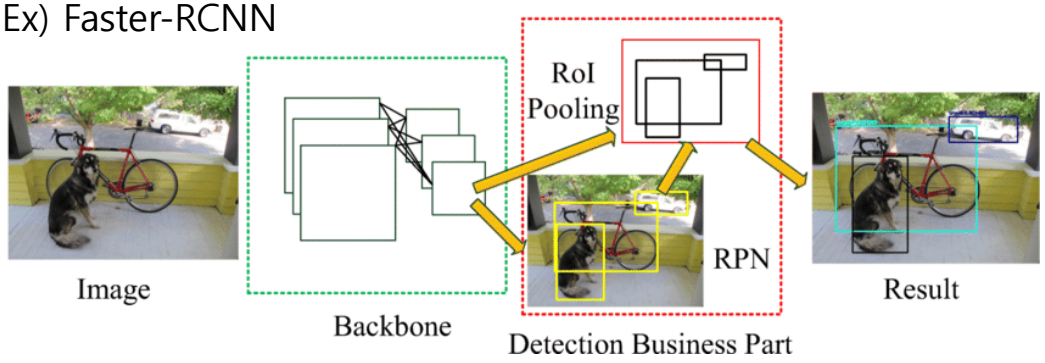
Ex) SSD



2-stage detector

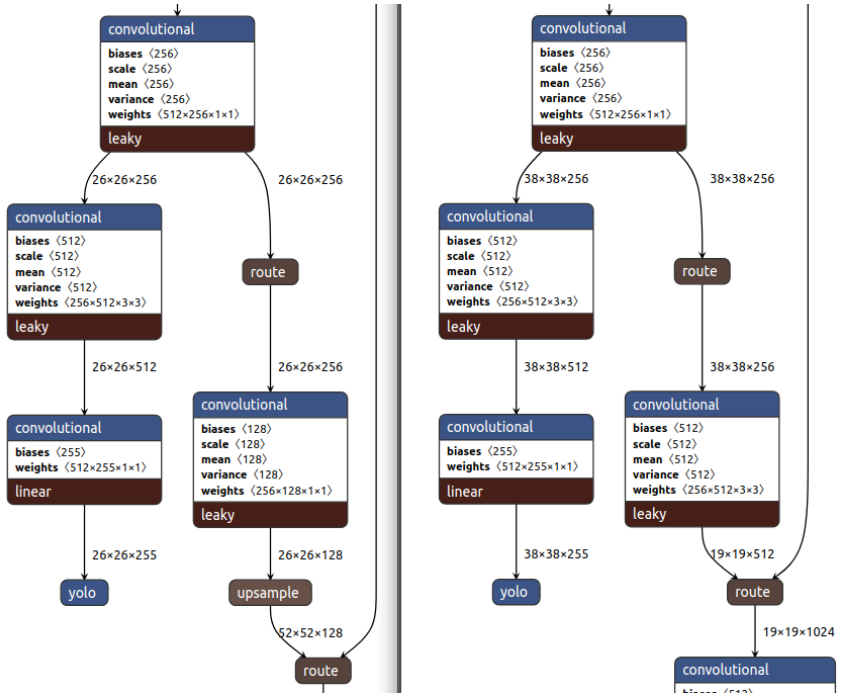


Ex) Faster-RCNN



Selection of Head

YOLOv3 = YOLOv4 = YOLOv5



YOLOv4 = CSPDarknet53 (BackBone)

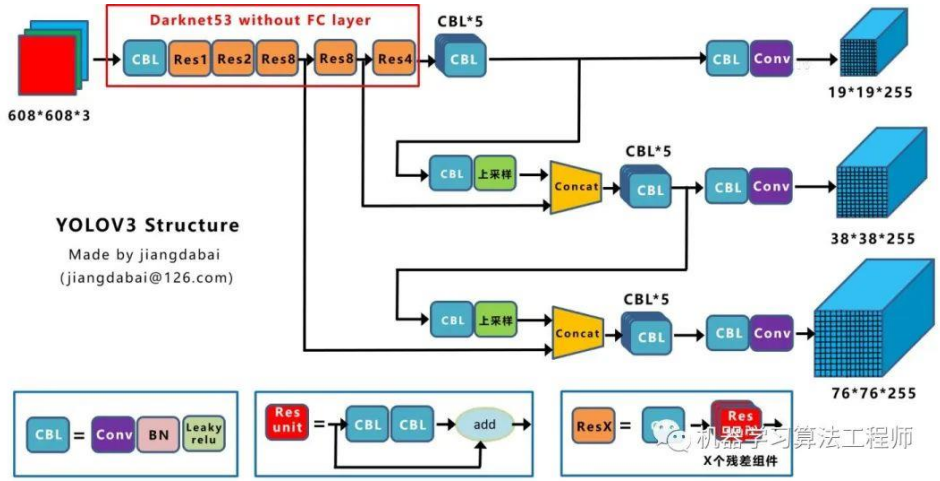
+

SPP + PANet (Neck)

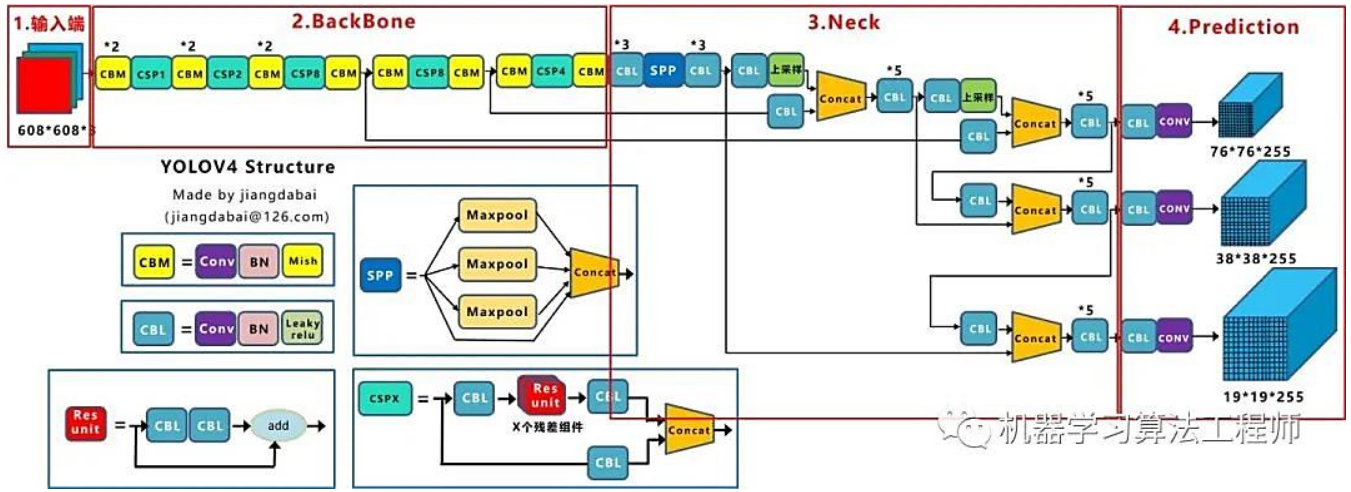
+

YOLOv3 (Head)

YOLOv3



YOLOv4



Bag of freebies

2.2. Bag of freebies

Usually, a conventional object detector is trained offline. Therefore, researchers always like to take this advantage and develop better training methods which can make the object detector receive better accuracy without increasing the inference cost. We call these methods that only change the training strategy or only increase the training cost as "bag of freebies." What is often adopted by object detection methods and meets the definition of bag of freebies is data augmentation. The purpose of data augmentation is to increase the variability of the input images, so that the designed object detection model has higher robustness to the images obtained from different environments. For examples, photometric distortions and geometric distortions are two commonly used data augmentation method and they definitely benefit the object detection task. In dealing with photometric distortion, we adjust the brightness, contrast, hue, saturation, and noise of an image. For geometric distortion, we add random scaling, cropping, flipping, and rotating.

The data augmentation methods mentioned above are all pixel-wise adjustments, and all original pixel information in the adjusted area is retained. In addition, some researchers engaged in data augmentation put their emphasis on simulating object occlusion issues. They have achieved good results in image classification and object detection. For example, random erase [100] and CutOut [11] can randomly select the rectangle region in an image and fill in a random or complementary value of zero. As for hide-and-seek [69] and grid mask [6], they randomly or evenly select multiple rectangle regions in an image and replace them to all zeros. If similar concepts are applied to feature maps, there are DropOut [71], DropConnect [80], and DropBlock [16] methods. In addition, some researchers have proposed the methods of using multiple images together to perform data augmentation. For example, MixUp [92] uses two images to multiply and superimpose with different coefficient ratios, and then adjusts the label with these superimposed ratios. As for CutMix [91], it is to cover the cropped image to rectangle region of other images, and adjusts the label according to the size of the mix area. In addition to the above mentioned methods, style transfer GAN [15] is also used for data augmentation, and such usage can effectively reduce the texture bias learned by CNN.

Different from the various approaches proposed above, some other bag of freebies methods are dedicated to solving the problem that the semantic distribution in the dataset may have bias. In dealing with the problem of semantic distribution bias, a very important issue is that there is a problem of data imbalance between different classes, and this problem is often solved by hard negative example mining [72] or online hard example mining [67] in two-stage object detector. But the example mining method is not applicable to one-stage object detector, because this kind of detector belongs to the dense prediction architecture. Therefore Lin *et al.* [45] proposed focal loss to deal with the problem of data imbalance existing between various classes. Another very important issue is that it is difficult to express the relationship of the degree of association between different categories with the one-hot hard representation. This representation scheme is often used when executing labeling. The label smoothing proposed in [73] is to convert hard label into soft label for training, which can make model more robust. In order to obtain a better soft label, Islam *et al.* [33] introduced the concept of knowledge distillation to design the label refinement network.

The last bag of freebies is the objective function of Bounding Box (BBox) regression. The traditional object detector usually uses Mean Square Error (MSE) to directly perform regression on the center point coordinates and height and width of the BBox, i.e., $\{x_{center}, y_{center}, w, h\}$, or the upper left point and the lower right point, i.e., $\{x_{top\_left}, y_{top\_left}, x_{bottom\_right}, y_{bottom\_right}\}$. As for anchor-based method, it is to estimate the corresponding offset, for example $\{x_{center\_offset}, y_{center\_offset}, w_{offset}, h_{offset}\}$ and $\{x_{top\_left\_offset}, y_{top\_left\_offset}, x_{bottom\_right\_offset}, y_{bottom\_right\_offset}\}$. However, to directly estimate the coordinate values of each point of the BBox is to treat these points as independent variables, but in fact does not consider the integrity of the object itself. In order to make this issue processed better, some researchers recently proposed IoU loss [90], which puts the coverage of predicted BBox area and ground truth BBox area into consideration. The IoU loss computing process will trigger the calculation of the four coordinate points of the BBox by executing IoU with the ground truth, and then connecting the generated results into a whole code. Because IoU is a scale invariant representation, it can solve the problem that when traditional methods calculate the l_1 or l_2 loss of $\{x, y, w, h\}$, the loss will increase with the scale. Recently, some researchers have continued to improve IoU loss. For example, GIoU loss [65] is to include the shape and orientation of object in addition to the coverage area. They proposed to find the smallest area BBox that can simultaneously cover the predicted BBox and ground truth BBox, and use this BBox as the denominator to replace the denominator originally used in IoU loss. As for DIoU loss [99], it additionally considers the distance of the center of an object, and CIoU loss [99], on the other hand simultaneously considers the overlapping area, the distance between center points, and the aspect ratio. CIoU can achieve better convergence speed and accuracy on the BBox regression problem.

Bag of Freebies

Change the training strategy or only increase the training cost as "bag of freebies."

Data augmentation / Data Imbalance / Loss function of Bbox regression

YOLOv4 BoF

Data Augmentation – CutMix, Mosaic , SAT

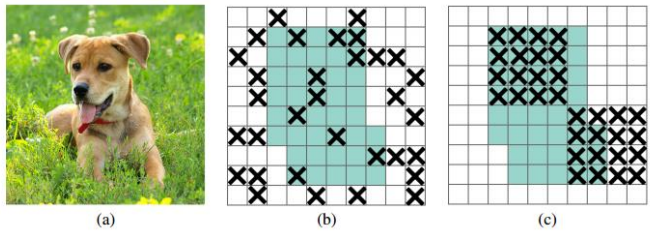
Imbalance Sampling – Class label smoothing

Bbox regression loss - CIoU-loss

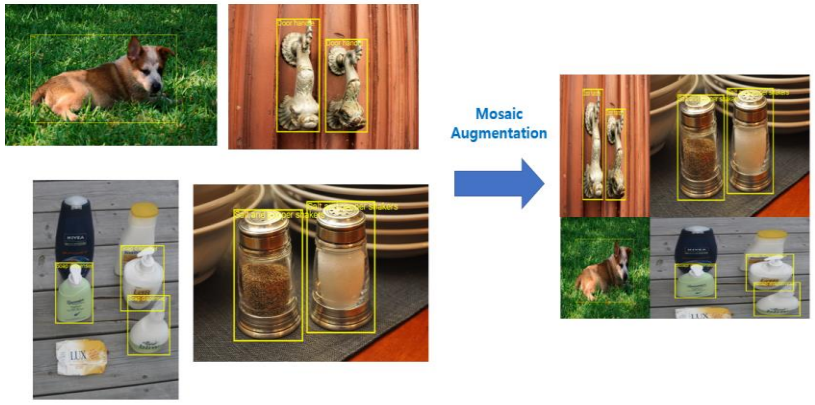
Regularization - DropBlock

Normalization - CmBN

Dropblock



Mosaic



Bag of specials

2.3. Bag of specials

For those plugin modules and post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection, we call them "bag of specials". Generally speaking, these plugin modules are for enhancing certain attributes in a model, such as enlarging receptive field, introducing attention mechanism, or strengthening feature integration capability, etc., and post-processing is a method for screening model prediction results.

Common modules that can be used to enhance receptive field are SPP [25], ASPP [5], and RFB [47]. The SPP module was originated from Spatial Pyramid Matching (SPM) [39], and SPMs original method was to split feature map into several $d \times d$ equal blocks, where d can be $\{1, 2, 3, ...\}$, thus forming spatial pyramid, and then extracting bag-of-word features. SPP integrates SPM into CNN and use max-pooling operation instead of bag-of-word operation. Since the SPP module proposed by He *et al.* [25] will output one dimensional feature vector, it is infeasible to be applied in Fully Convolutional Network (FCN). Thus in the design of YOLOv3 [63], Redmon and Farhadi improve SPP module to the concatenation of max-pooling outputs with kernel size $k \times k$, where $k = \{1, 5, 9, 13\}$, and stride equals to 1. Under this design, a relatively large $k \times k$ max-pooling effectively increase the receptive field of backbone feature. After adding the improved version of SPP module, YOLOv3-608 upgrades AP_{50} by 2.7% on the MS COCO object detection task at the cost of 0.5% extra computation. The difference in operation between ASPP [5] module and improved SPP module is mainly from the original $k \times k$ kernel size, max-pooling of stride equals to 1 to several 3×3 kernel size, dilated ratio equals to k, and stride equals to 1 in dilated convolution operation. RFB module is to use several dilated convolutions of $k \times k$ kernel, dilated ratio equals to k, and stride equals to 1 to obtain a more comprehensive spatial coverage than ASPP. RFB [47] only costs 7% extra inference time to increase the AP_{50} of SSD on MS COCO by 5.7%.

The attention module that is often used in object detection is mainly divided into channel-wise attention and point-wise attention, and the representatives of these two attention models are Squeeze-and-Excitation (SE) [29] and Spatial Attention Module (SAM) [85], respectively. Although SE module can improve the power of ResNet50 in the ImageNet image classification task 1% top-1 accuracy at the cost of only increasing the computational effort by 2%, but on a GPU usually it will increase the inference time by about 10%, so it is more appropriate to be used in mobile devices. But for SAM, it only needs to pay 0.1% extra calculation and it can improve ResNet50-SE 0.5% top-1 accuracy on the ImageNet image classification task. Best of all, it does not affect the speed of inference on the GPU at all.

In terms of feature integration, the early practice is to use skip connection [51] or hyper-column [22] to integrate low-level physical feature to high-level semantic feature. Since multi-scale prediction methods such as FPN have become popular, many lightweight modules that integrate different feature pyramid have been proposed. The modules of this sort include SFAM [98], ASFF [48], and BiFPN [77]. The main idea of SFAM is to use SE module to execute channel-wise level re-weighting on multi-scale concatenated feature maps. As for ASFF, it uses softmax as point-wise level re-weighting and then adds feature maps of different scales. In BiFPN, the multi-input weighted residual connections is proposed to execute scale-wise level re-weighting, and then add feature maps of different scales.

In the research of deep learning, some people put their focus on searching for good activation function. A good activation function can make the gradient more efficiently propagated, and at the same time it will not cause too much extra computational cost. In 2010, Nair and Hinton [56] propose ReLU to substantially solve the gradient vanish problem which is frequently encountered in traditional tanh and sigmoid activation function. Subsequently, LReLU [54], PReLU [24], ReLU6 [28], Scaled Exponential Linear Unit (SELU) [35], Swish [59], hard-Swish [27], and Mish [55], etc., which are also used to solve the gradient vanish problem, have been proposed. The main purpose of LReLU and PReLU is to solve the problem that the gradient of ReLU is zero when the output is less than zero. As for ReLU6 and hard-Swish, they are specially designed for quantization networks. For self-normalizing a neural network, the SELU activation function is proposed to satisfy the goal. One thing to be noted is that both Swish and Mish are continuously differentiable activation function.

The post-processing method commonly used in deep-learning-based object detection is NMS, which can be used to filter those BBoxes that badly predict the same object, and only retain the candidate BBoxes with higher response. The way NMS tries to improve is consistent with the method of optimizing an objective function. The original method proposed by NMS does not consider the context information, so Girshick *et al.* [19] added classification confidence score in R-CNN as a reference, and according to the order of confidence score, greedy NMS was performed in the order of high score to low score. As for soft NMS [1], it considers the problem that the occlusion of an object may cause the degradation of confidence score in greedy NMS with IoU score. The DIoU NMS [99] developers way of thinking is to add the information of the center point distance to the BBox screening process on the basis of soft NMS. It is worth mentioning that, since none of above post-processing methods directly refer to the captured image features, post-processing is no longer required in the subsequent development of an anchor-free method.

Bag of Specials

Increase the inference cost a little bit but improve the accuracy significantly

Plugin modules / Post-processing / Activation function

YOLOv4 BoS

Activation – Mish

Receptive Field – **SPP**

Skip Connections– **CSP**, MiWRC

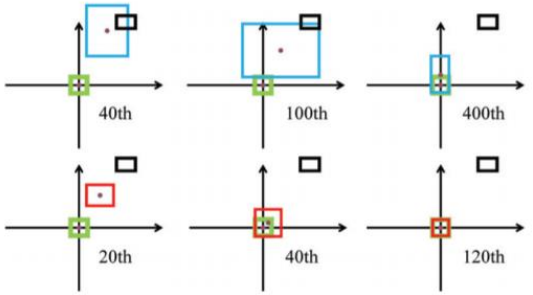
Attention – SAM

Feature Integration - **PAN**

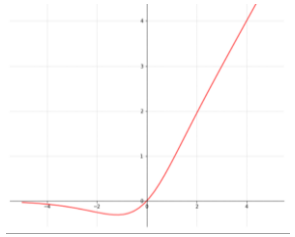
Post processing– DIoU-NMS

DIoU-NMS

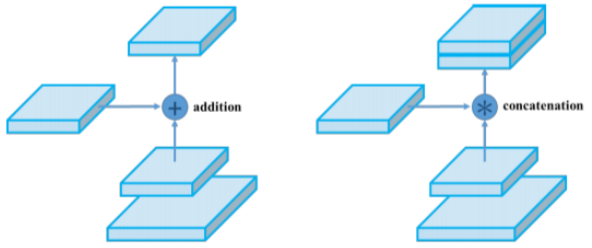
$$s_i = \begin{cases} s_i, IoU - \mathcal{R}_{DIoU}(\mathcal{M}, B_i) < \epsilon \\ 0, IoU - \mathcal{R}_{DIoU}(\mathcal{M}, B_i) \geq \epsilon \end{cases}$$

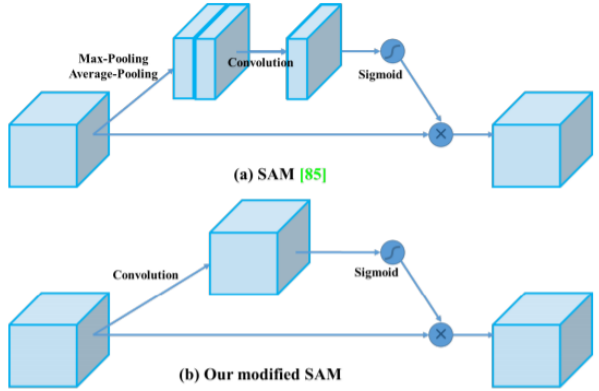


Mish



Modified PAN and SAM





Architecture

YOLOv4 = CSPDarknet53 (BackBone) + SPP + PANet (Neck) + YOLOv3 (Head)

BoF

Regularization - DropBlock

Data Augmentation – CutMix, Mosaic

Imbalance Sampling – Class label smoothing

BoS

Activation - Mish

Skip Connections– CSP, MiWRC

BoF

Bbox regression loss - CIoU-loss

Normalization - CmBN

Regularization - DropBlock

Data Augmentation – Mosaic, SAT

Hyperparameter tuning– Genetic algorithms

Etc.

BoS

Activation - Mish

Receptive Field – SPP

Attention – SAM

Feature Integration - PAN

Post processing– DIoU-NMS

Experiments

4.2. Influence of different features on Classifier training

First, we study the influence of different features on classifier training; specifically, the influence of Class label smoothing, the influence of different data augmentation techniques, bilateral blurring, MixUp, CutMix and Mosaic, as shown in Fugure 7, and the influence of different activations, such as Leaky-ReLU (by default), Swish, and Mish.

Table 2: Influence of BoF and Mish on the CSPResNeXt-50 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	**94.0%**
	✓						**78.0%**	**94.3%**
		✓					**78.1%**	**94.5%**
			✓				77.5%	93.8%
				✓			**78.1%**	**94.4%**
					✓		64.5%	86.0%
						✓	**78.9%**	**94.5%**
	✓	✓		✓			**78.5%**	**94.8%**
	✓	✓		✓		✓	**79.8%**	**95.2%**

4.3. Influence of different features on Detector training

Further study concerns the influence of different Bag-of-Freebies (BoF-detector) on the detector training accuracy, as shown in Table 4. We significantly expand the BoF list through studying different features that increase the detector accuracy without affecting FPS:

Table 3: Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			**77.8%**	**94.4%**
	✓	✓		✓		✓	**78.7%**	**94.8%**

Table 5: Ablation Studies of Bag-of-Specials. (Size 512x512).

Model	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	42.4%	64.4%	45.9%
CSPResNeXt50-PANet-SPP-RFB	41.8%	62.7%	45.1%
CSPResNeXt50-PANet-SPP-SAM	**42.7%**	**64.6%**	**46.3%**
CSPResNeXt50-PANet-SPP-SAM-G	41.6%	62.7%	45.0%
CSPResNeXt50-PANet-SPP-ASFF-RFB	41.1%	62.6%	44.4%

Table 4: Ablation Studies of Bag-of-Freebies. (CSPResNeXt50-PANet-SPP, 512x512).

S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP_{50}	AP_{75}
									MSE	38.0%	60.0%	40.8%
✓									MSE	37.7%	59.9%	40.5%
	✓								MSE	**39.1%**	**61.8%**	**42.0%**
		✓							MSE	36.9%	59.7%	39.4%
			✓						MSE	**38.9%**	**61.7%**	**41.9%**
				✓					MSE	33.0%	55.4%	35.4%
					✓				MSE	**38.4%**	**60.7%**	**41.3%**
						✓			MSE	**38.7%**	**60.7%**	**41.9%**
							✓		MSE	35.3%	57.2%	38.0%
✓									GIoU	**39.4%**	59.4%	**42.5%**
✓									DIoU	**39.1%**	58.8%	**42.1%**
✓									CIoU	**39.6%**	59.2%	**42.6%**
✓	✓	✓	✓						CIoU	**41.5%**	**64.0%**	**44.8%**
	✓		✓					✓	CIoU	36.1%	56.5%	38.4%
✓	✓	✓	✓					✓	MSE	**40.3%**	**64.0%**	**43.1%**
✓	✓	✓	✓					✓	GIoU	**42.4%**	**64.4%**	**45.9%**
✓	✓	✓	✓					✓	CIoU	**42.4%**	**64.4%**	**45.9%**

4.4. Influence of different backbones and pre-trained weightings on Detector training

Table 6: Using different classifier pre-trained weightings for detector training (all other training parameters are similar in all models) .

Model (with optimal setting)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP	512x512	42.4	64.4	45.9
CSPResNeXt50-PANet-SPP (BoF-backbone)	512x512	42.3	64.3	45.7
CSPResNeXt50-PANet-SPP (BoF-backbone + Mish)	512x512	42.3	64.2	45.8
CSPDarknet53-PANet-SPP (BoF-backbone)	512x512	42.4	64.5	46.0
CSPDarknet53-PANet-SPP (BoF-backbone + Mish)	512x512	43.0	64.9	46.5

4.5. Influence of different mini-batch size on Detector training

Finally, we analyze the results obtained with models trained with different mini-batch sizes, and the results are shown in Table 7. From the results shown in Table 7, we found that after adding BoF and BoS training strategies, the mini-batch size has almost no effect on the detector's performance. This result shows that after the introduction of BoF and BoS, it is no longer necessary to use expensive GPUs for training. In other words, anyone can use only a conventional GPU to train an excellent detector.

Table 7: Using different mini-batch size for detector training.

Model (without OA)	Size	AP	AP_{50}	AP_{75}
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 4)	608	37.1	59.2	39.9
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 8)	608	38.4	60.6	41.6
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 4)	512	41.6	64.1	45.0
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 8)	512	41.7	64.2	45.2

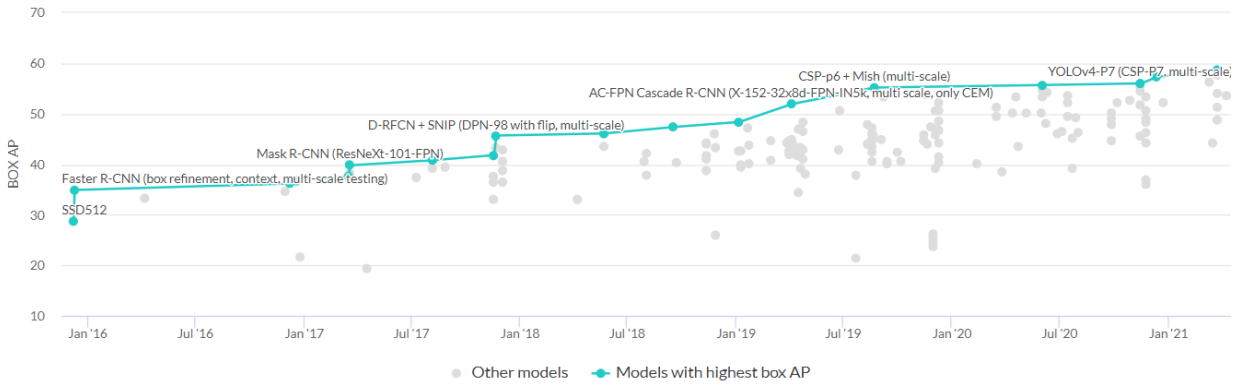
Conclusions

1. We 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.

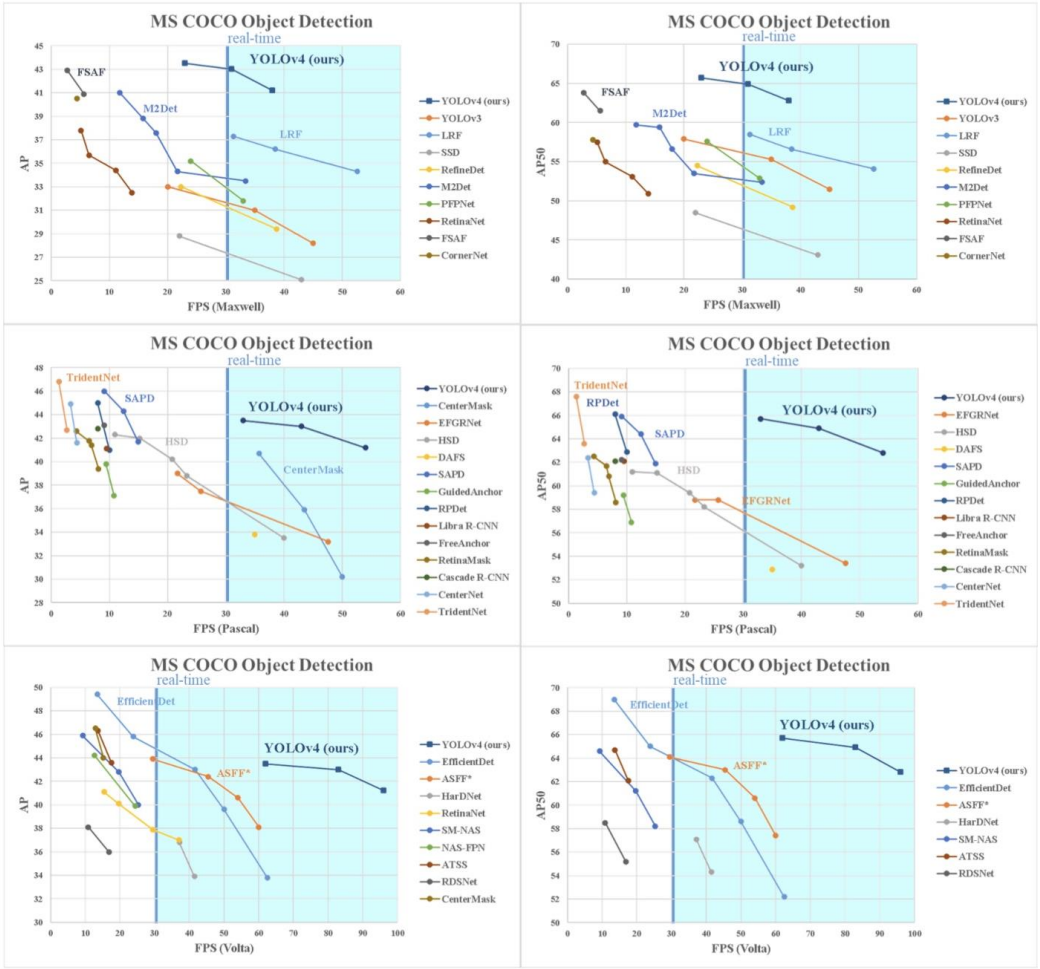
2. We verify the influence of state-of-the-art **BoF and BoS** methods of object detection during the detector training.

3. We **modify** state-of-the-art methods and make them more efficient and suitable for **single GPU training**, including CBN, PAN, SAM, etc.

<Paper with Code>



Object Detection on COCO test-dev



References

[Paper]

https://arxiv.org/pdf/2004.10934.pdf

[Code]

https://github.com/pjreddie/darknet