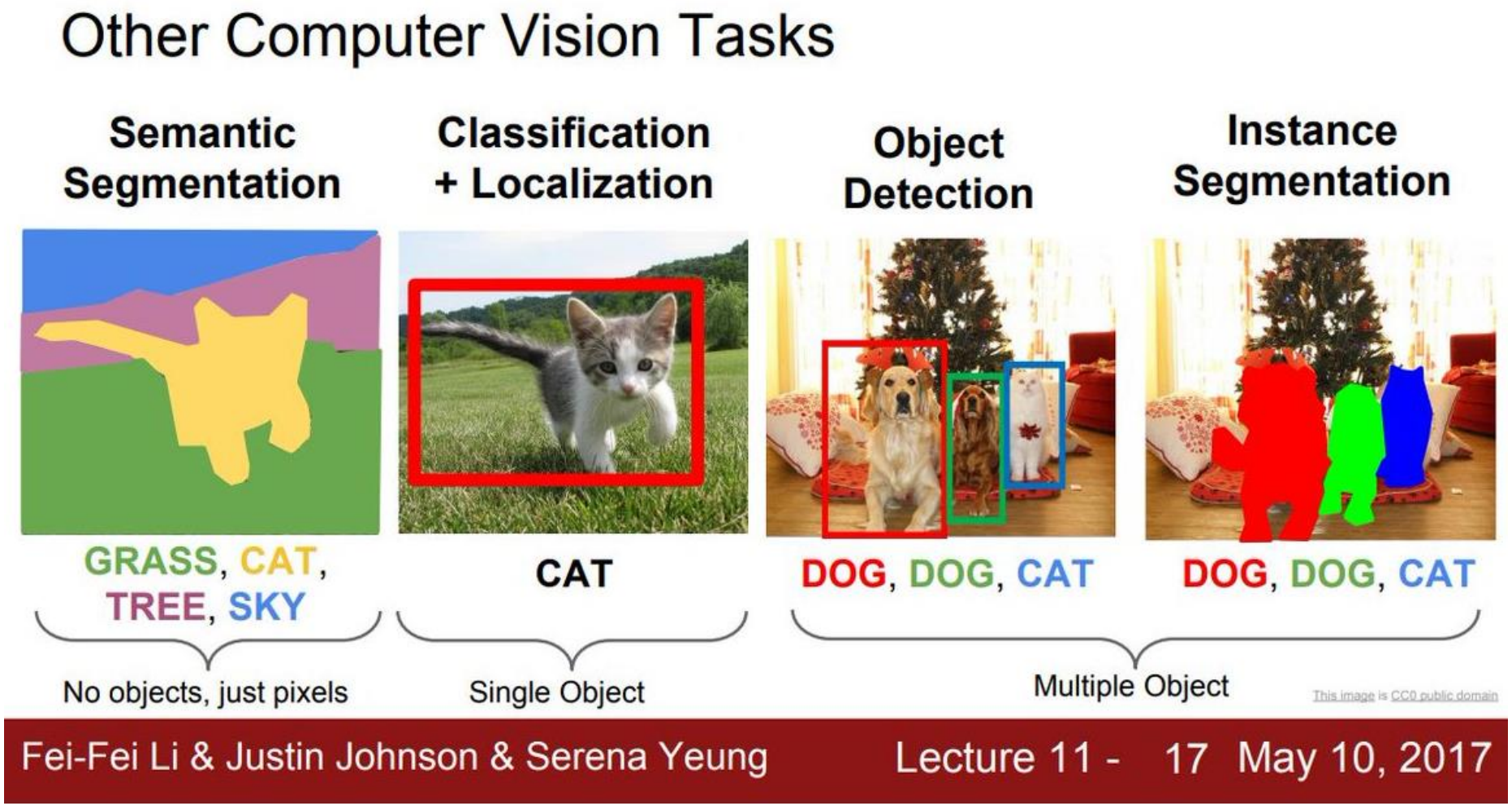
YOLOv4:
Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao

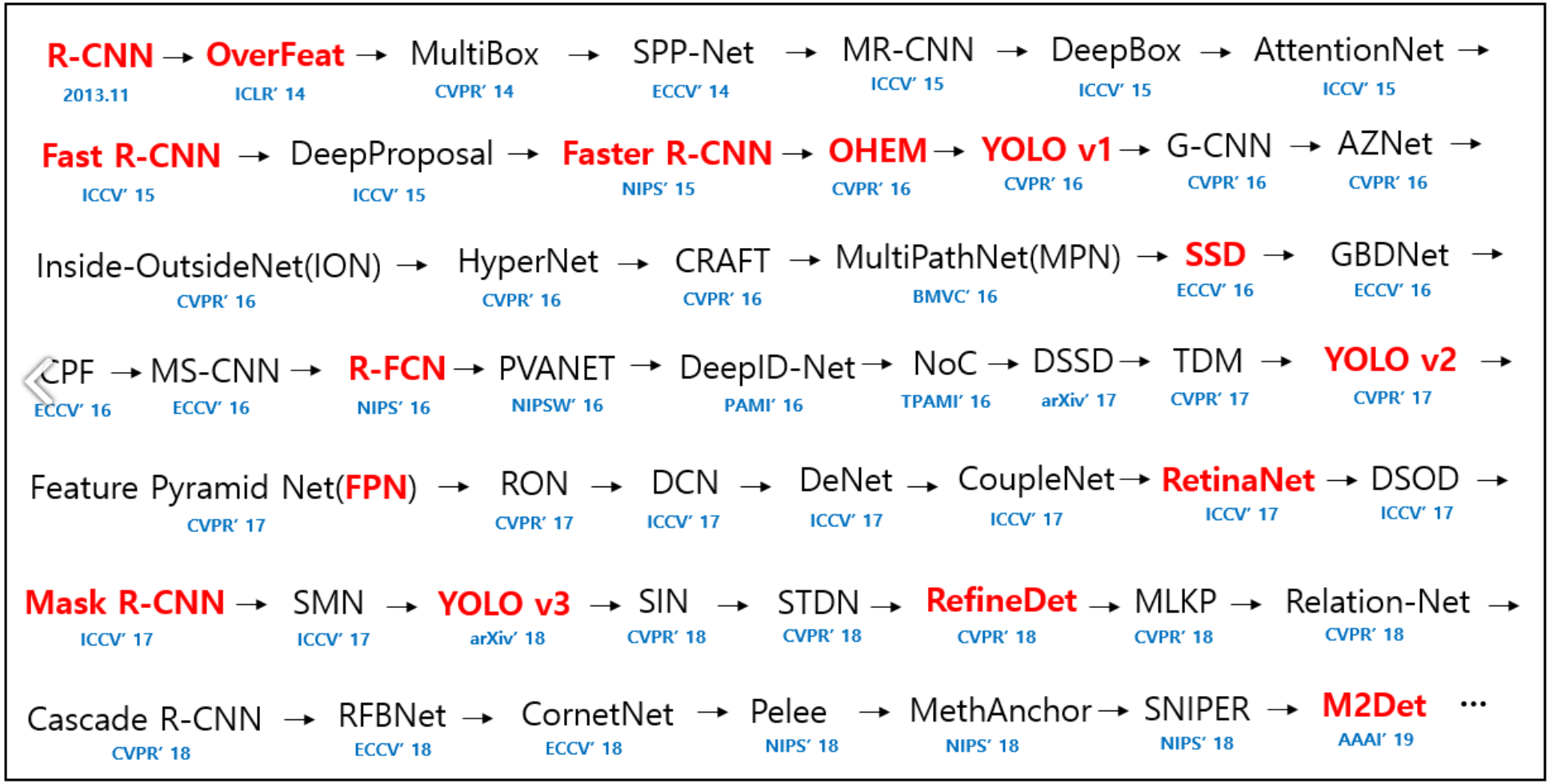
Wonbeom Jang

https://medium.com/@jonathan_hui/yolov4-c9901eaa8e61

Object Detection



Object Detection



Object Detection

Two Stage Detector

R-CNN, fast R-CNN, faster R-CNN

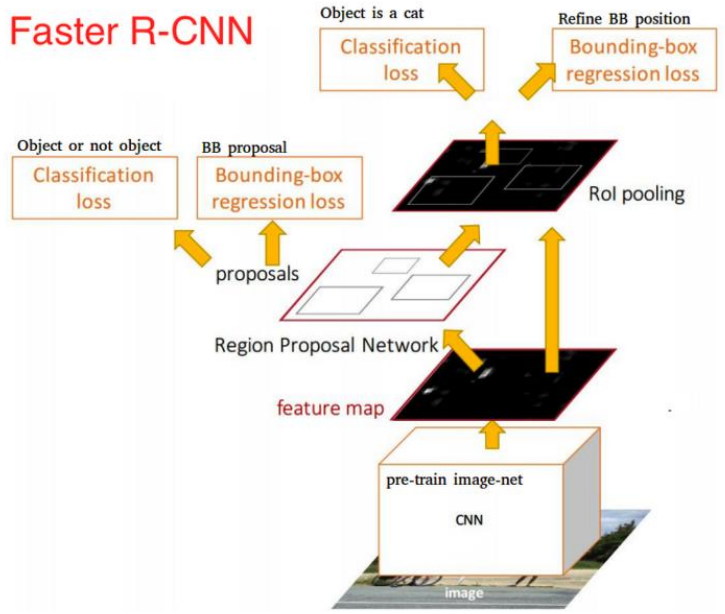
One Stage Detector

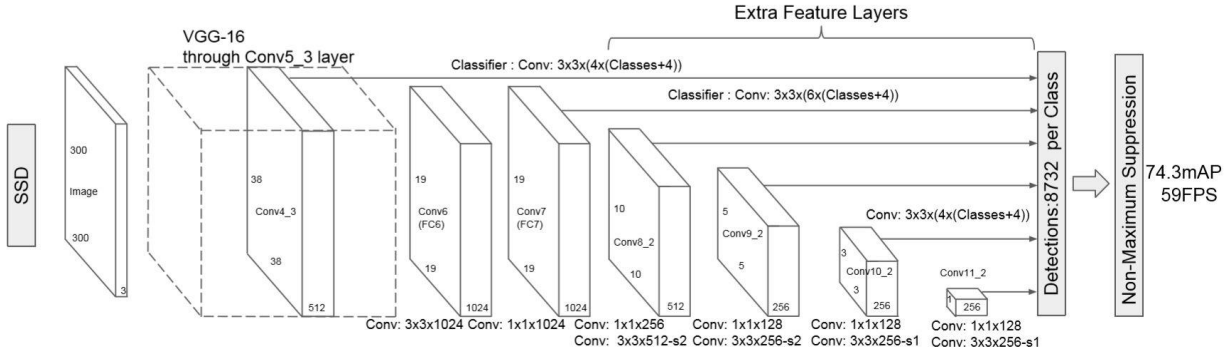
Anchor

YOLO, SSD, RetinaNet

Anchor-free

CornerNet, CenterNet



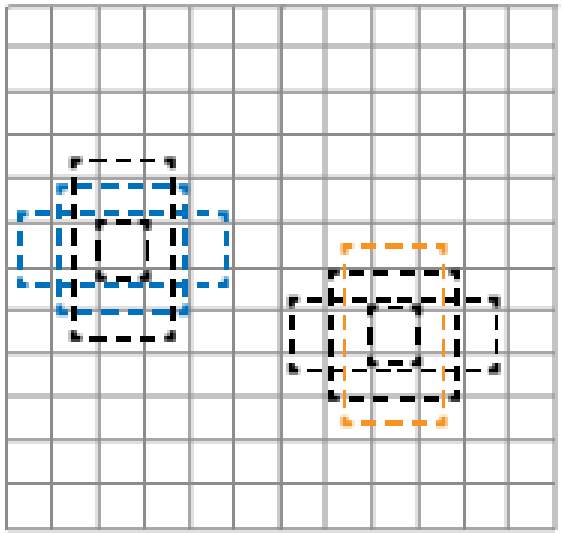


SSD

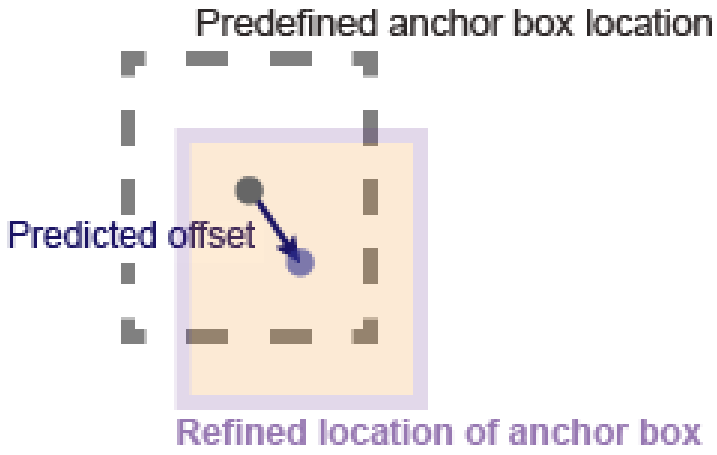
Object Detection

Anchor Box

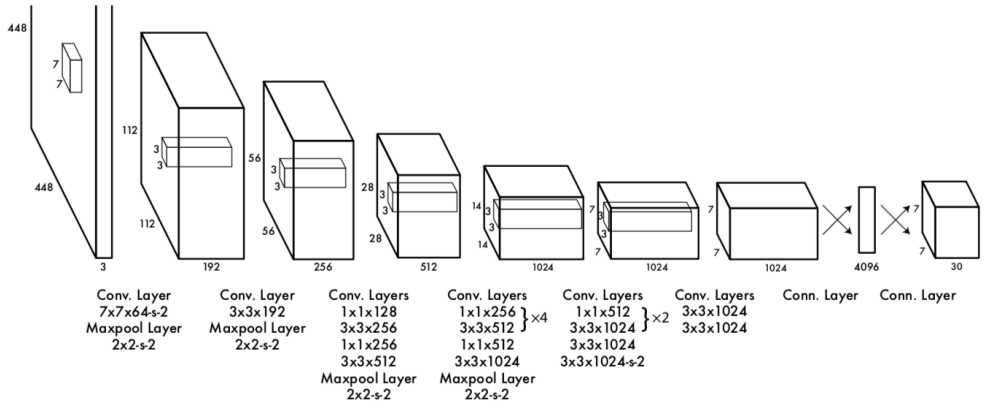
Ground truth image and bounding boxes



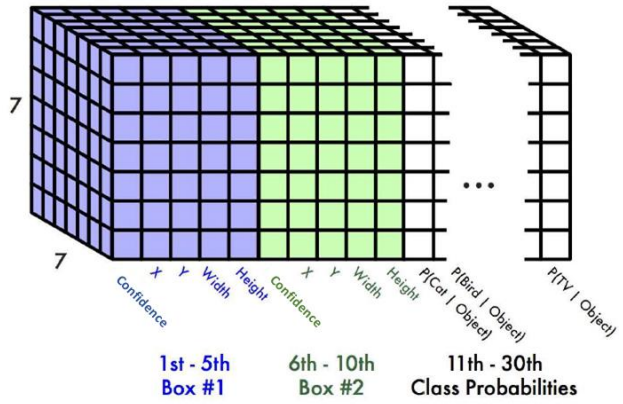
Anchor boxes at each predefined location in each feature map



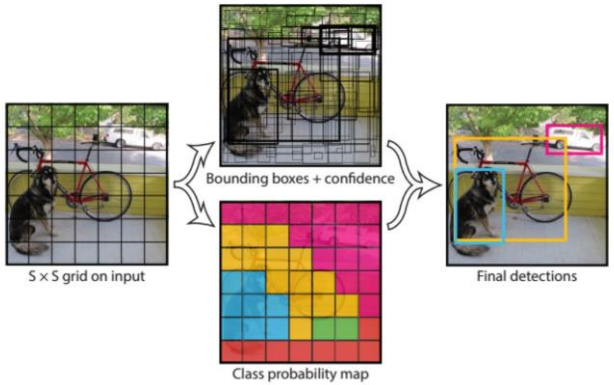
YOLO V1



Architecture



Output feature



Fast operation: 7x7 grid

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

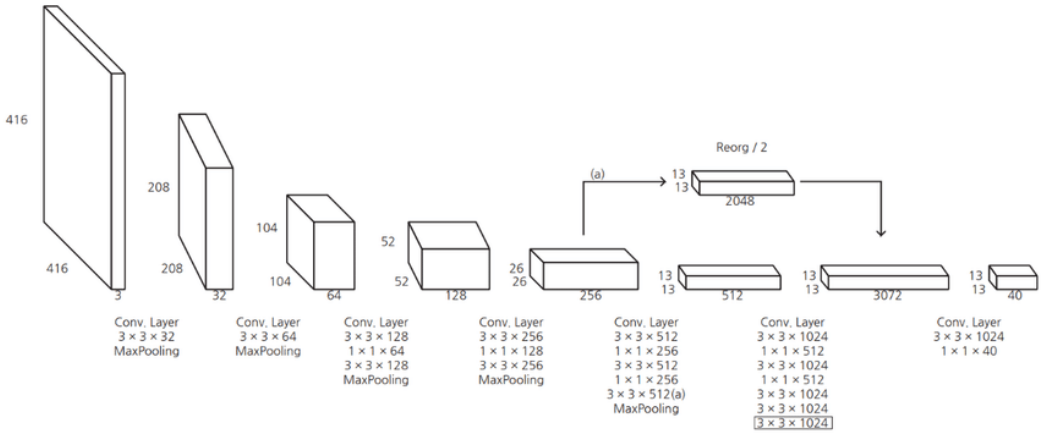
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

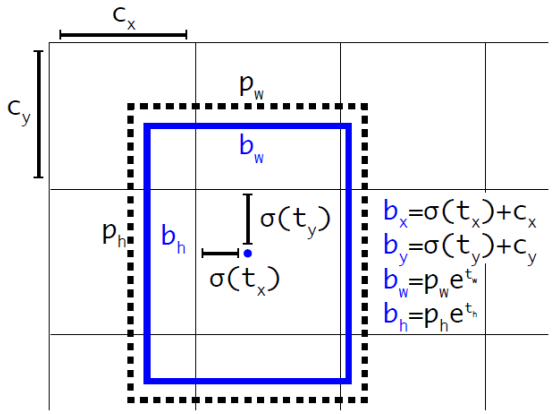
Loss Function

https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89

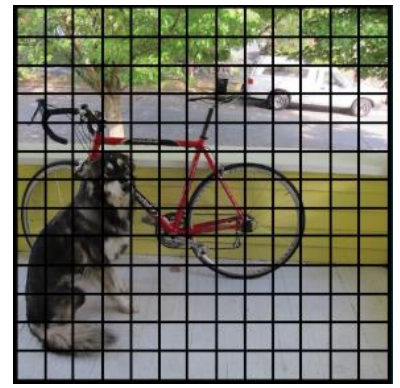
YOLO V2



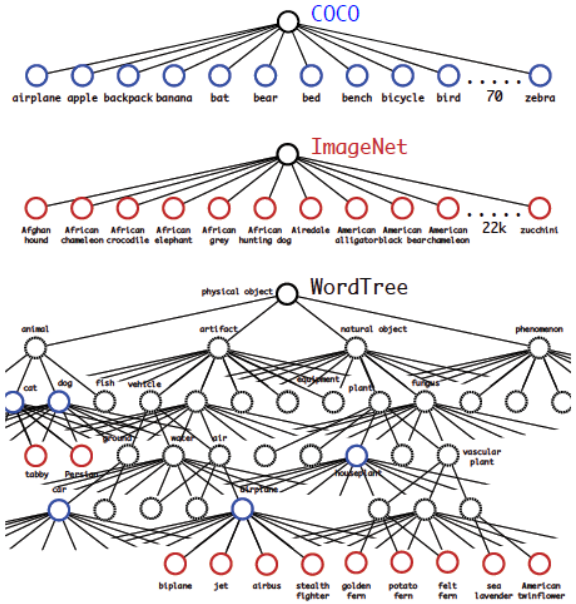
Architecture



Better Accuracy: 5 Anchor box

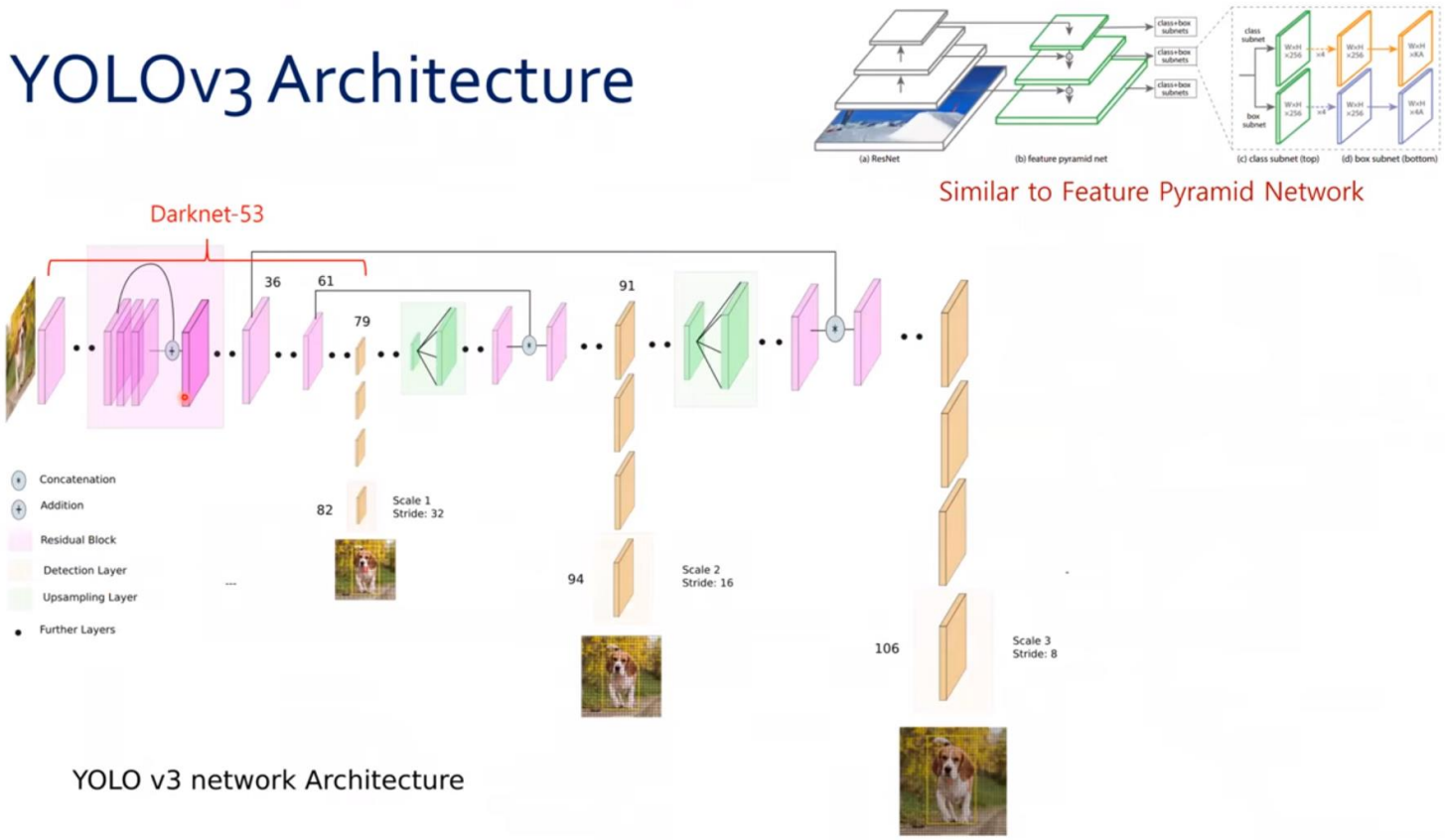


To find small object: 13x13 grid



Word Tree

YOLO V3



YOLO v3 network Architecture

We still use k-means clustering to determine our bounding box priors. We just sort of chose 9 clusters and 3 scales arbitrarily and then divide up the clusters evenly across scales. On the COCO dataset the 9 clusters were: $(10 \times 13), (16 \times 30), (33 \times 23), (30 \times 61), (62 \times 45), (59 \times 119), (116 \times 90), (156 \times 198), (373 \times 326)$.

9 Anchor box

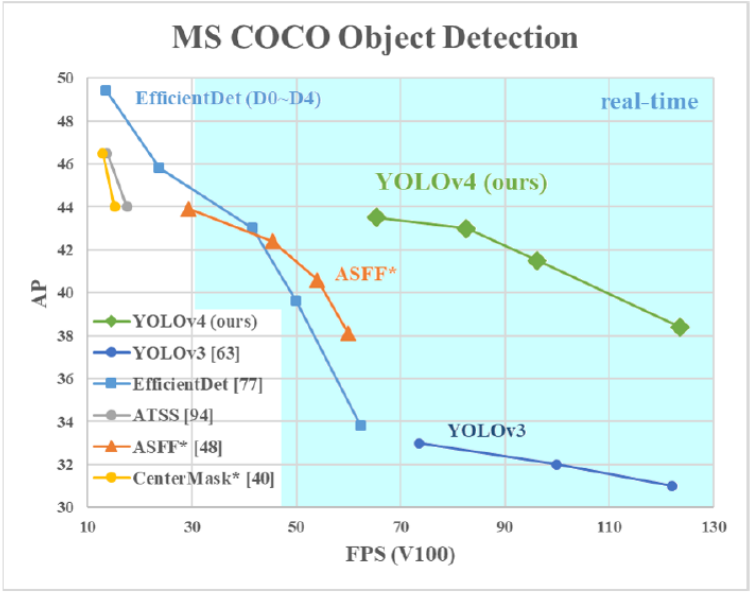
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 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			

Table 1. **Darknet-53.**

Contribution

1. We develop 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 efficient and suitable for **single GPU training**, including CBN [89], PAN [49], SAM [85], etc.

Bag of trick for YOLO V3



Bag of freebies

We call these methods that only change the training strategy or **only increase the training cost** as “bag of freebies.”

Data augmentation

Random erase, CutOut, grid mask, DropOut, DropConnect, DropBlock, MixUp, CutMix, style transfer GAN

Imbalanced data

Negative mining, focal loss

Relationship

Label smoothing

Bounding box regression

IoU loss, GIoU loss, CIoU loss

Bag of special

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”.

Module

SPP(Spatial Pyramid Pooling), ASPP, RFB

Attention

Squeeze-and-Excitation (SE), Spatial Attention Module (SAM)

Feature integration

skip connection, hyper-column

Activation function

ReLU, LReLU, PReLU, ReLU6, SELU, Swish, hard-Swish, Mish

NMS, (non-maximum suppression)

IoU NMS, DIoU NMS

Methodology

- For GPU we use a small number of groups (1 - 8) in convolutional layers: CSPResNeXt50 / **CSPDarknet53**
- For VPU - we use grouped-convolution, but we refrain from using Squeeze-and-excitement (SE) blocks specifically this includes the following models: EfficientNet-lite / MixNet [76] / GhostNet [21] / MobileNetV3

Selection of architecture

Our objective is to find the optimal **balance** among the input **network resolution**, the **convolutional layer number**, **the parameter number** (filter size2 * filters * channel / groups), and **the number of layer outputs** (filters).

The next objective is to **select additional blocks for increasing the receptive field** and the best method of parameter aggregation from different backbone levels for different detector levels: e.g. FPN, PAN, ASFF, BiFPN.

Selection of architecture

The CSPDarknet53 is better compared to CSPResNext50 in terms of detecting objects on the MS COCO dataset

The detector requires the following

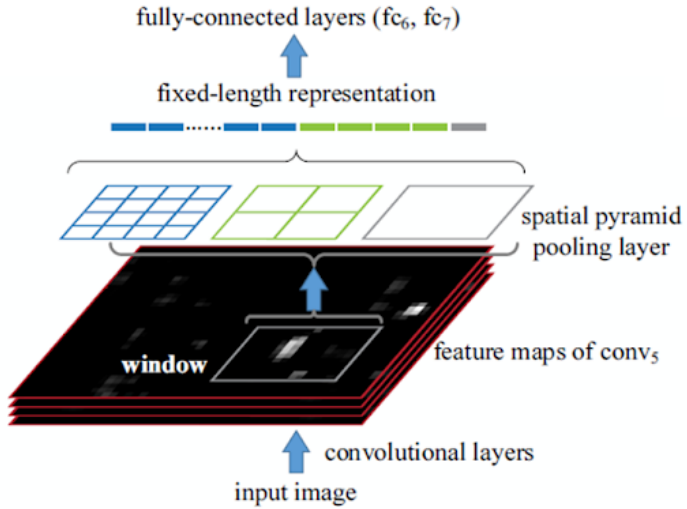
- Higher input network size (resolution) – for detecting multiple small-sized objects
- More layers – for a higher receptive field to cover the increased size of input network
- More parameters – for greater capacity of a model to detect multiple objects of different sizes in a single image

The influence of the receptive field with different sizes is summarized as follows:

- Up to the object size - allows viewing the entire object
- Up to network size - allows viewing the context around the object
- Exceeding the network size - increases the number of connections between the image point and the final activation

Selection of architecture

- We add the SPP block over the CSPDarknet53, since it significantly increases the receptive field,



- We use PANet as the method of parameter aggregation from different backbone levels for different detector levels, instead of the FPN used in YOLOv3.

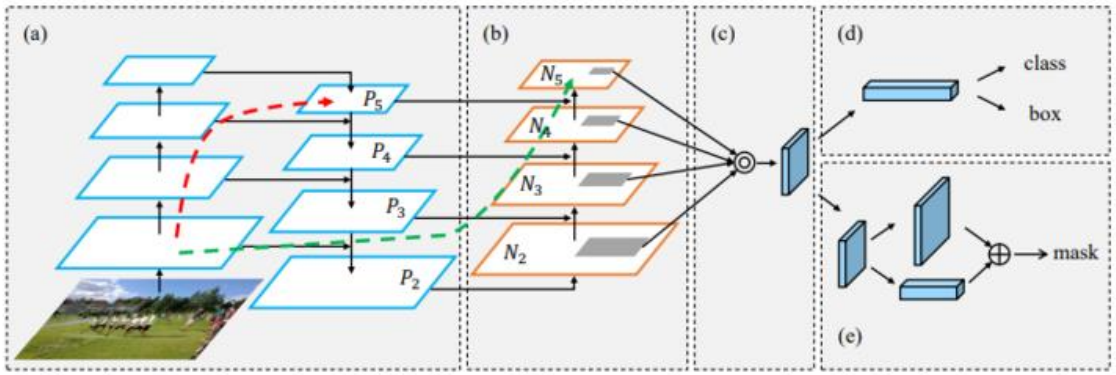
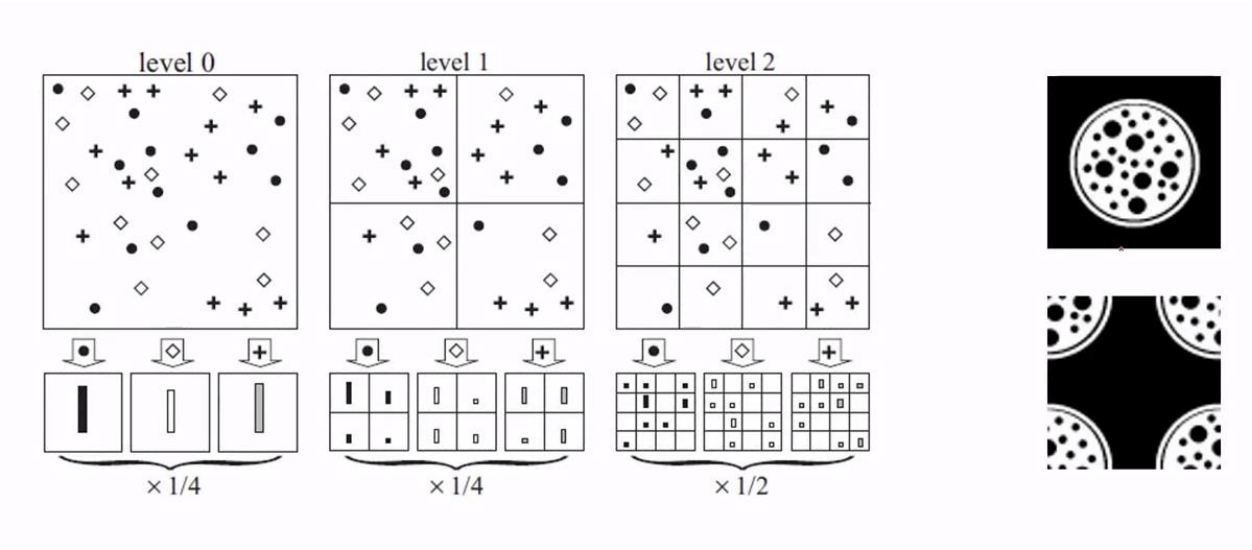


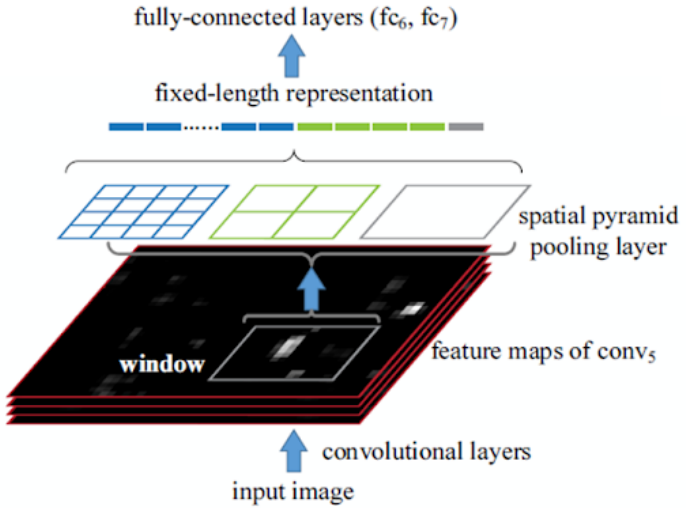
Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Selection of architecture

SPM

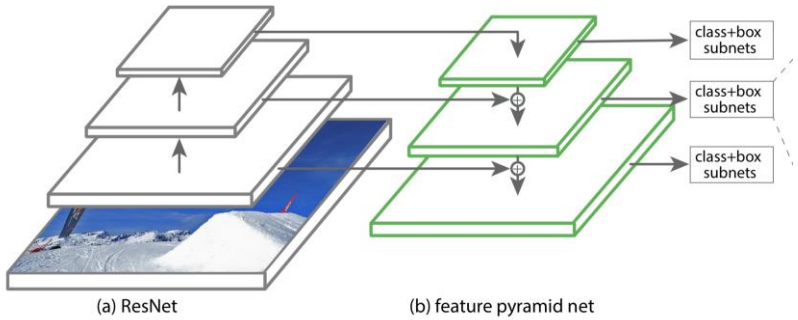


SPP



Selection of architecture

FPN



PANet

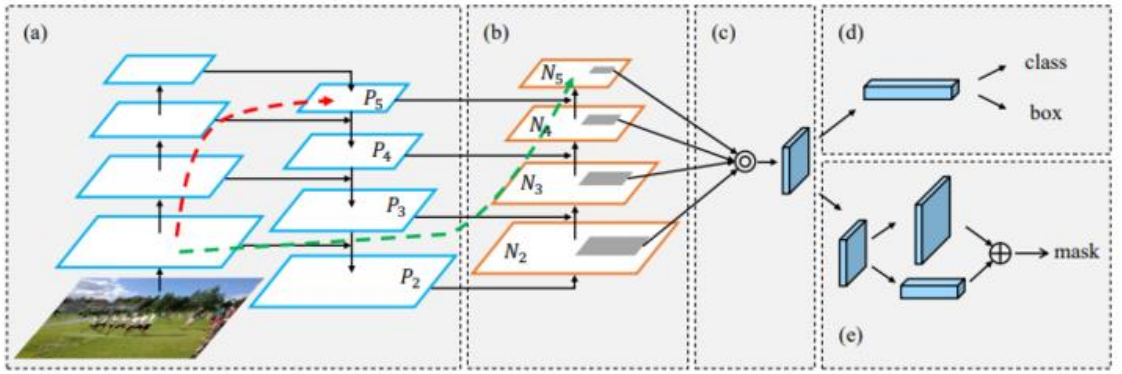


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Selection of BoF and BoS

- Activations: ReLU, leaky-ReLU, ~~parametric-ReLU~~, ~~ReLU6~~, ~~SELU~~, Swish, or Mish
- Bounding box regression loss: MSE, IoU, GIoU, CIoU, DIoU
- Data augmentation: CutOut, MixUp, CutMix
- Regularization method: DropOut, DropPath, Spatial DropOut,or DropBlock
- Normalization of the network activations by their mean and variance: Batch Normalization (BN), Cross-GPU Batch Normalization (CGBN or SyncBN), Filter Response Normalization (FRN), or Cross-Iteration Batch Normalization (CBN)
- Skip-connections: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

Selection of BoF and BoS

- We introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)
- We select optimal hyper-parameters while applying genetic algorithms
- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)

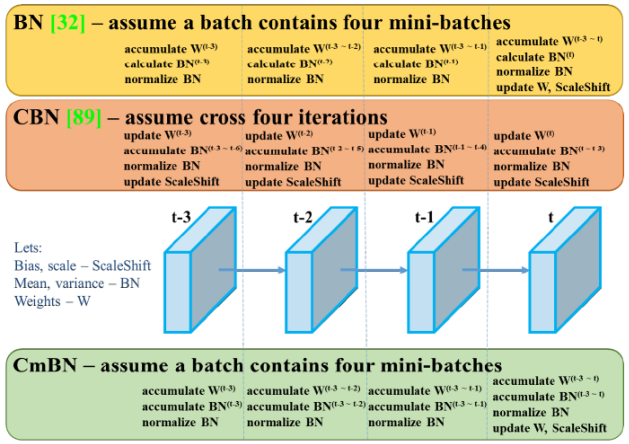


Figure 4: Cross mini-Batch Normalization.

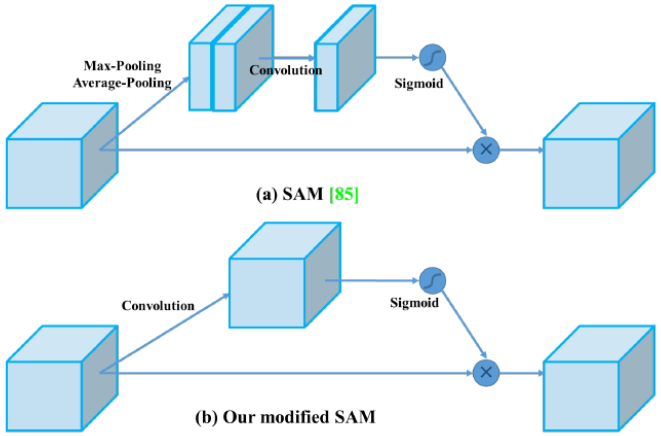


Figure 5: Modified SAM.

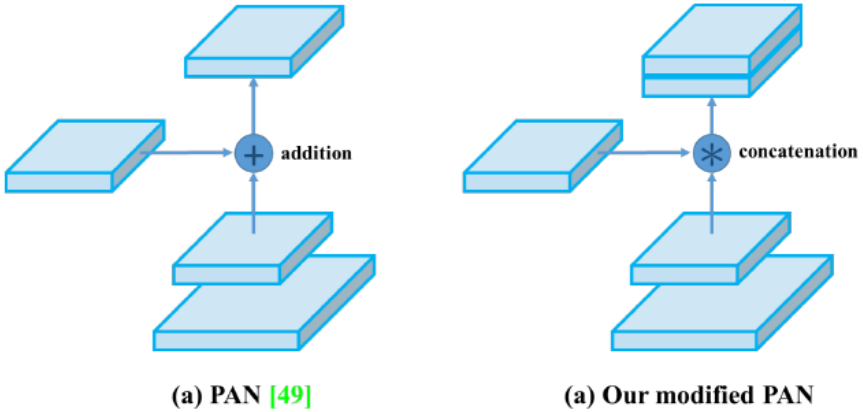


Figure 6: Modified PAN.

Selection of BoF and BoS

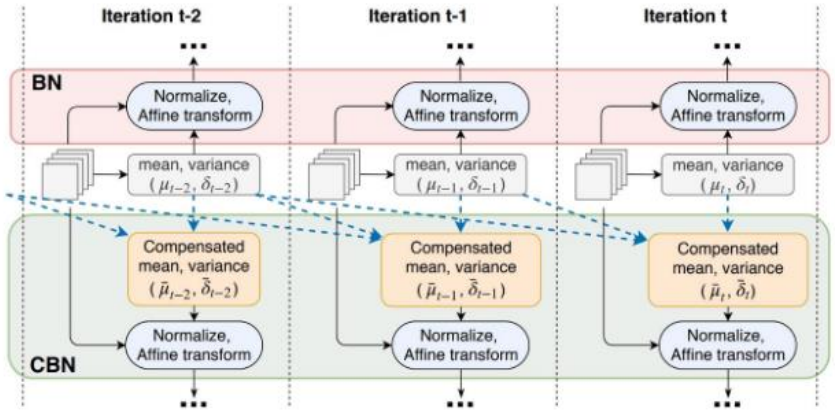
- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)

as weights are changing in each iteration, the statistics collected under those weights may become inaccurate under the new weight

$$\bar{\mu}^l_{t,k}(\theta_t) = \frac{1}{k}\sum_{\tau=0}^{k-1}\mu^l_{t-\tau}(\theta_t),$$

$$\bar{\nu}^l_{t,k}(\theta_t) = \frac{1}{k}\sum_{\tau=0}^{k-1}\max\left[\nu^l_{t-\tau}(\theta_t), \mu^l_{t-\tau}(\theta_t)^2\right],$$

$$\bar{\sigma}^l_{t,k}(\theta_t) = \sqrt{\bar{\nu}^l_{t,k}(\theta_t) - \bar{\mu}^l_{t,k}(\theta_t)^2},$$



Cross-Iteration Batch Normalization

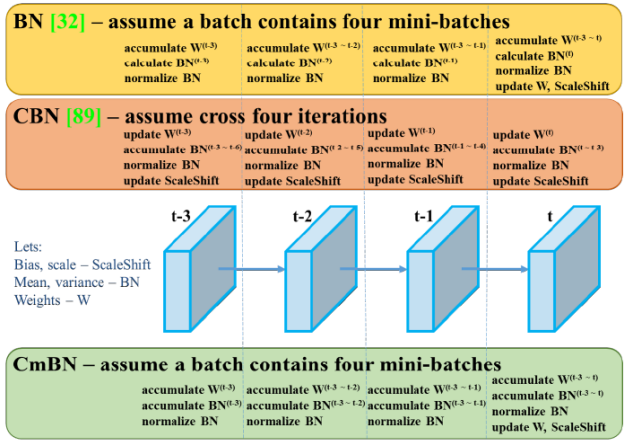
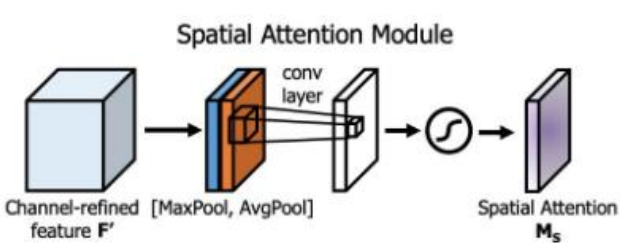


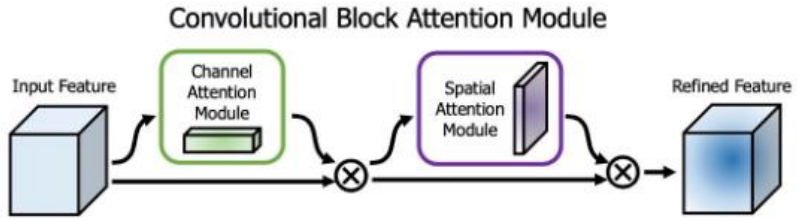
Figure 4: Cross mini-Batch Normalization.

Cross mini-Batch Normalization

Selection of BoF and BoS

- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)





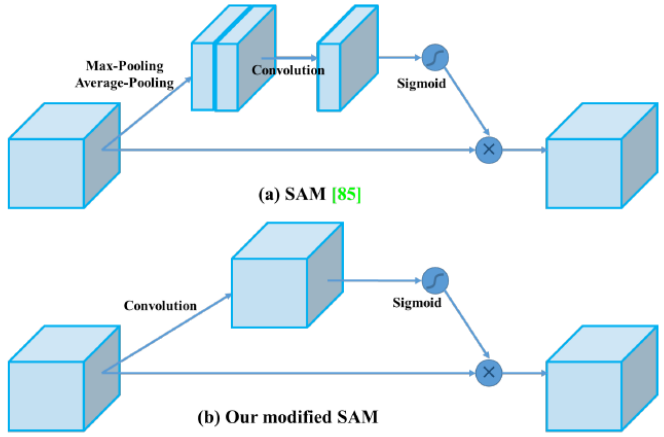
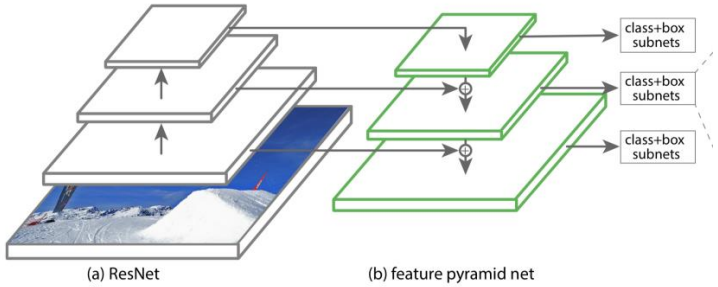


Figure 5: Modified SAM.

Selection of BoF and BoS

- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)



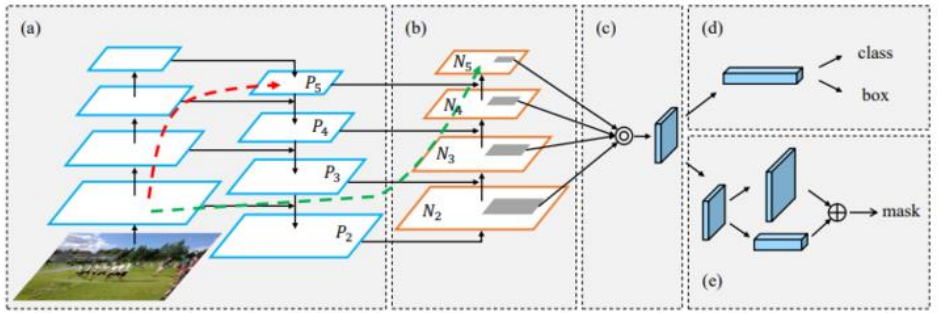


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

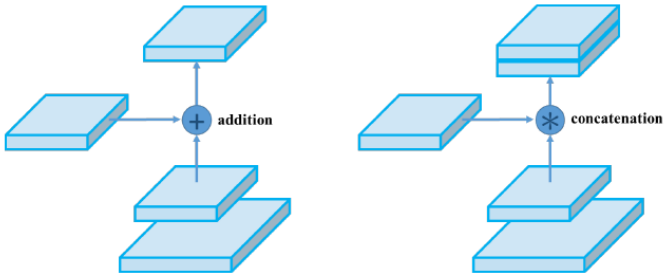


Figure 6: Modified PAN.

Selection of BoF and BoS

- We introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)
- We select optimal hyper-parameters while applying genetic algorithms
- We modify some exsiting methods to make our design suitble for efficient training and detection – modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)

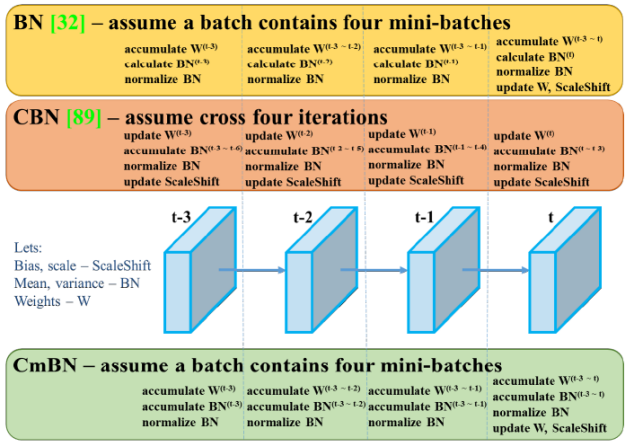


Figure 4: Cross mini-Batch Normalization.

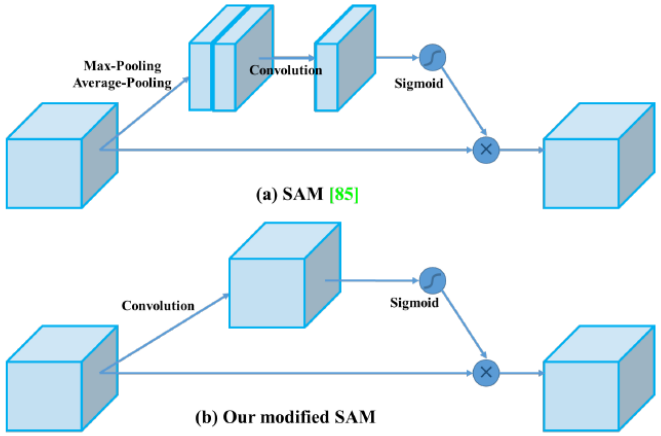


Figure 5: Modified SAM.

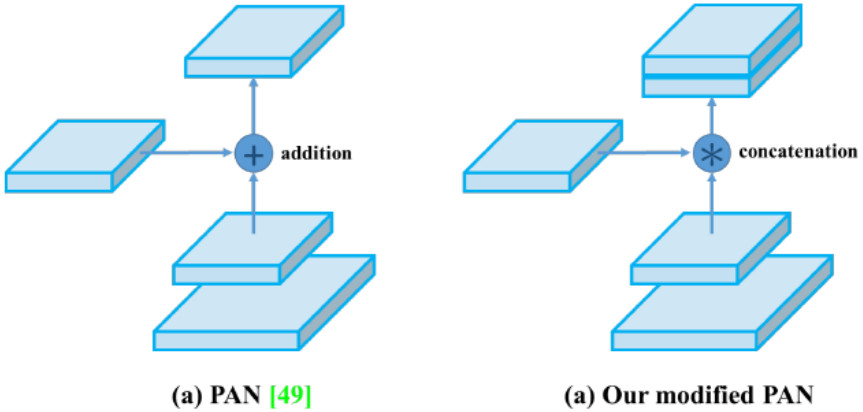


Figure 6: Modified PAN.

YOLOv4

- Backbone: CSPDarknet53
- Neck: SPP, PAN
- Head: YOLOv3

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing

• Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC)

• Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes

• Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

YOLOv4

- Backbone: CSPDarknet53
- Neck: SPP, PAN
- Head: YOLOv3

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing

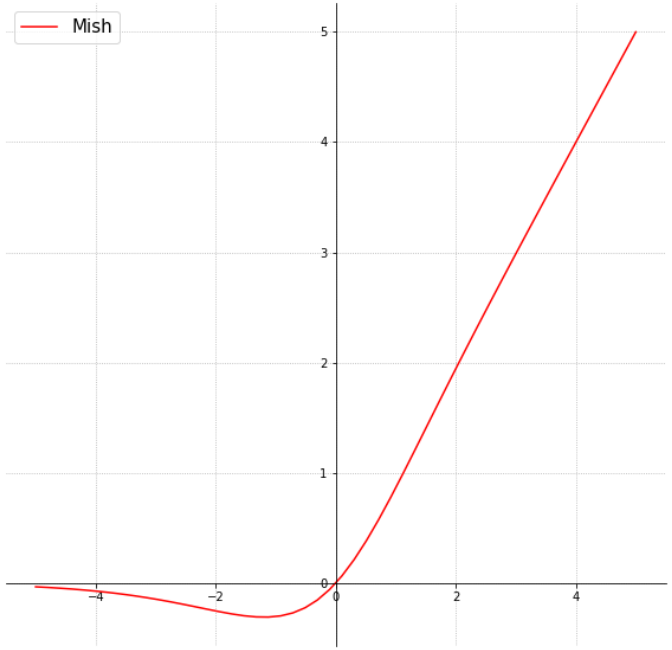
• Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multi input weighted residual connections (MiWRC)

• Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes

• Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

YOLOv4

Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multi input weighted residual connections (MiWRC)

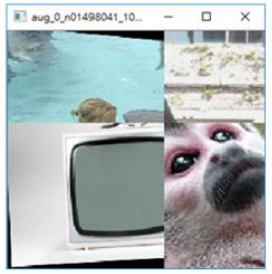


x * tf.nn.tanh(tf.nn.softplus(x))

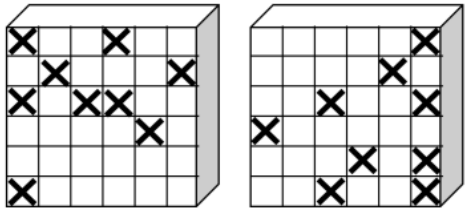
YOLOv4

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing

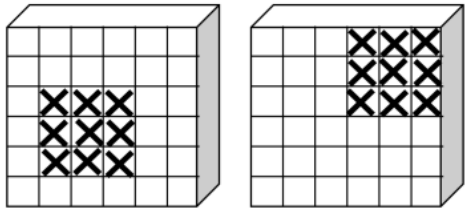
(c) CutMix



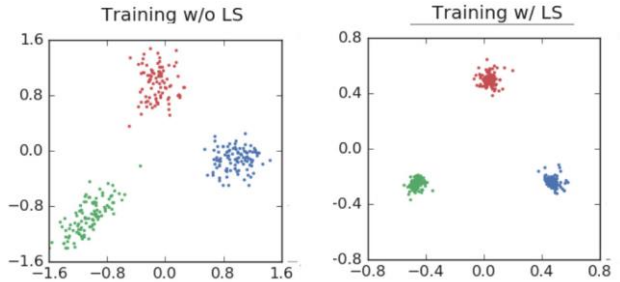
(d) Mosaic



(a) Dropout



(c) DropBlock



[1, 0, 0, 0] → [0.8, 0.2, 0.1, 0]

Class label smoothing

YOLOv4

- Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes

$$\mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2},$$

where $\mathbf{b}$ and $\mathbf{b}^{gt}$ denote the central points of B and B^{gt}, $\rho(\cdot)$ is the Euclidean distance, and c is the diagonal length of the smallest enclosing box covering the two boxes.

$$\mathcal{R}_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v,$$

where α is a positive trade-off parameter, and v measures the consistency of aspect ratio,

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2.$$

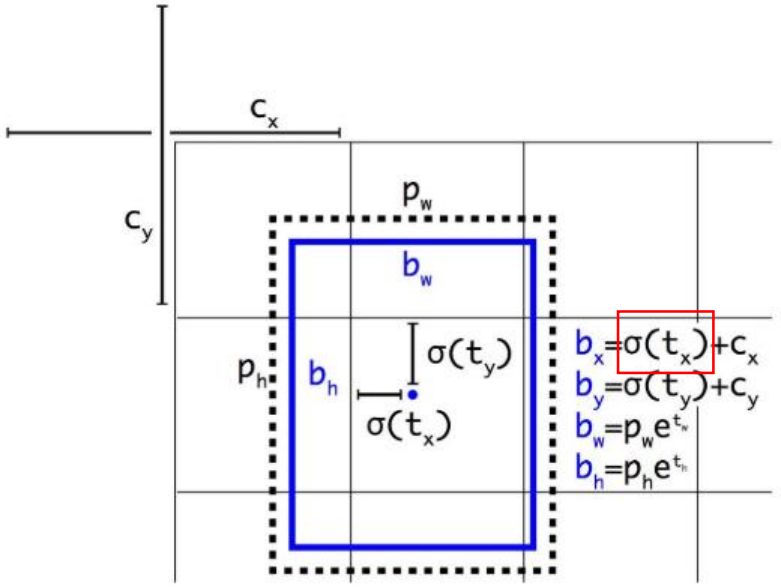
Then the loss function can be defined as

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v.$$

And the trade-off parameter α is defined as

$$\alpha = \frac{v}{(1 - IoU) + v},$$

CIoU-loss

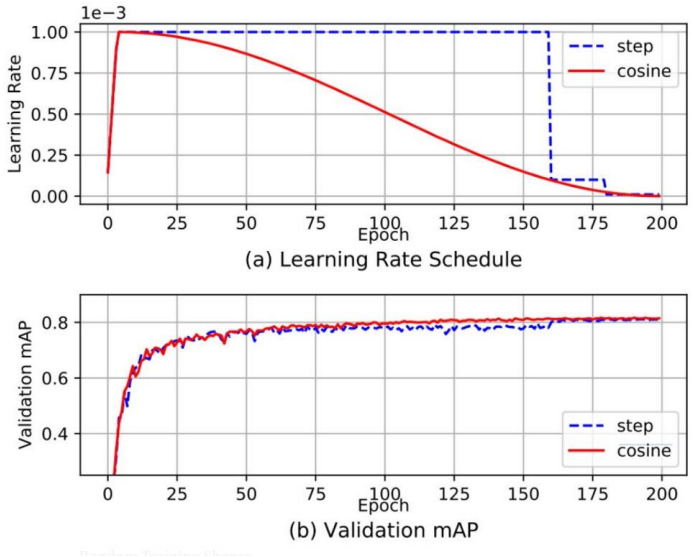


Eliminate grid sensitivity

Multiply scaling factor > 1

YOLOv4

- Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes



Cosine annealing scheduler

Genetic algorithm

Optimal hyperparameters

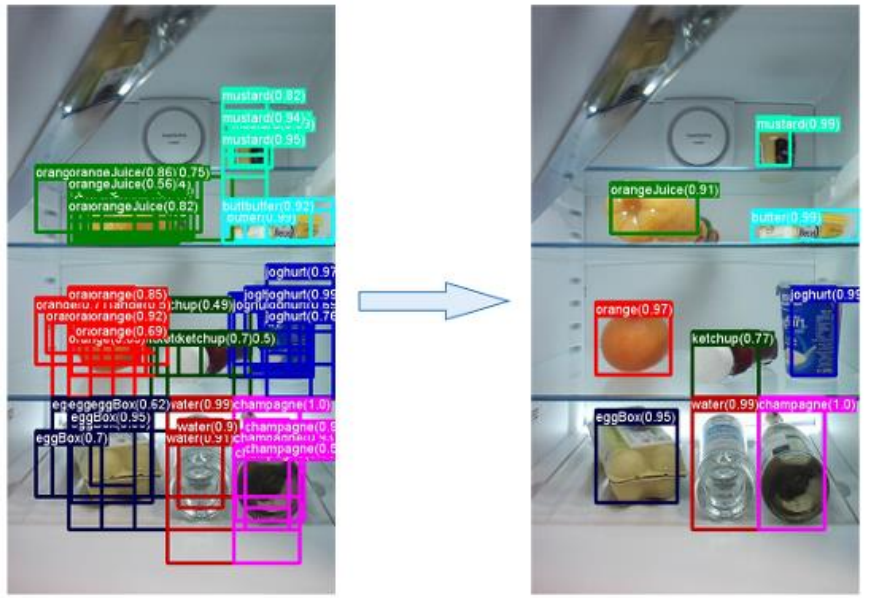
Multi-scale

Random training shapes

YOLOv4

- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$



DIoU-NMS

Influence of different features on Classifier training

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%**

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%**

Influence of different features on Detector training

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%**

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%

Influence of different backbones and pretrained weightings on Detector training

First, although classification accuracy of CSPResNeXt-50 models trained with different features is higher compared to CSPDarknet53 models, the **CSPDarknet53 model shows higher accuracy in terms of object detection.**

However, using BoF and Mish for the CSPDarknet53 classifier training increases the accuracy of both the classifier and the detector

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

Influence of different mini-batch size on Detector training

The mini-batch size has almost no effect on the detector's performance.

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

Result

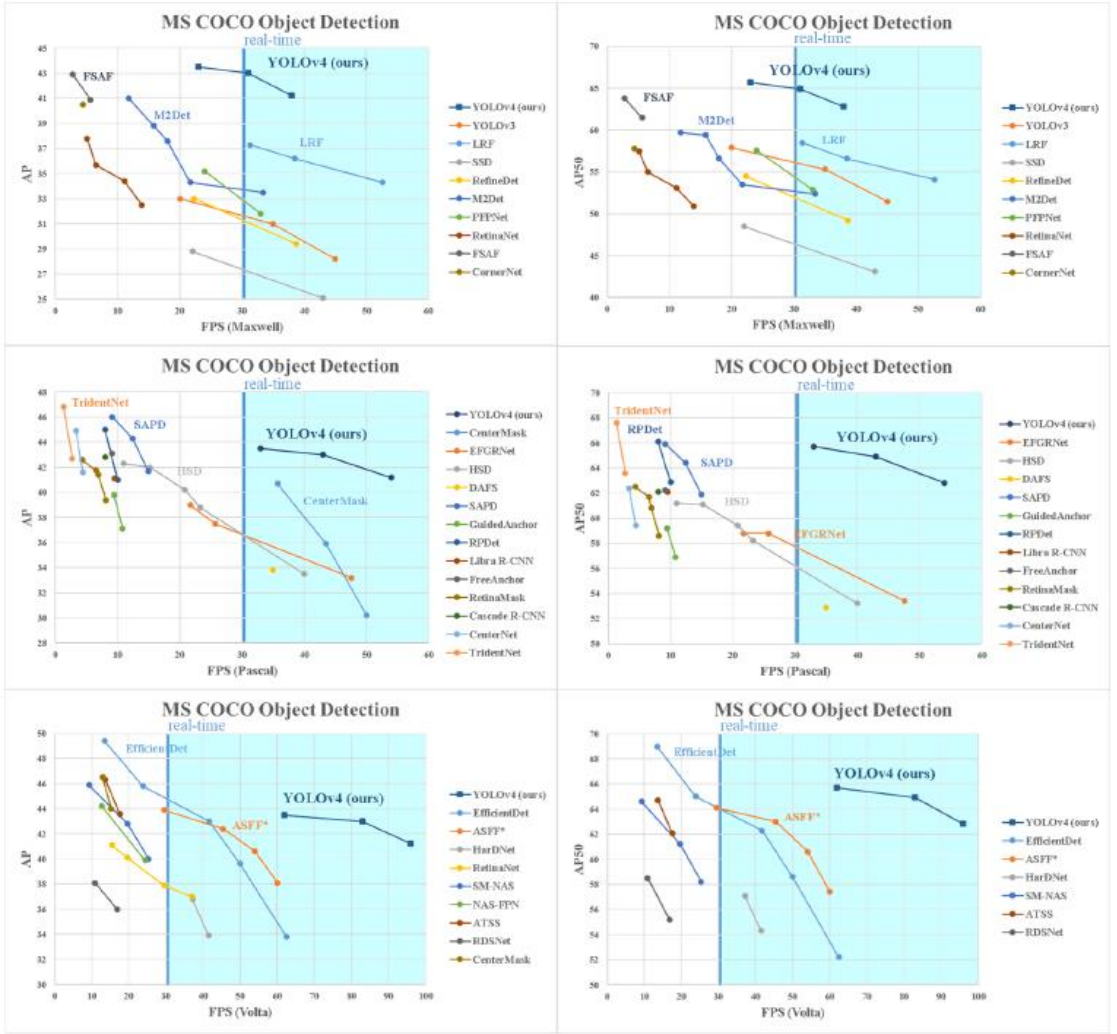


Figure 8: Comparison of the speed and accuracy of different object detectors. (Some articles stated the FPS of their detectors for only one of the GPUs: Maxwell/Pascal/Volta)