What is 'YOLOv4 '?

비트캠프 혁신 인공지능(서울) 4조

발표 및 팀장 : 전병준

팀원 : 강청순, 황채연, 김현수

Contribution

1. 효율적이고 강력한 model 개발

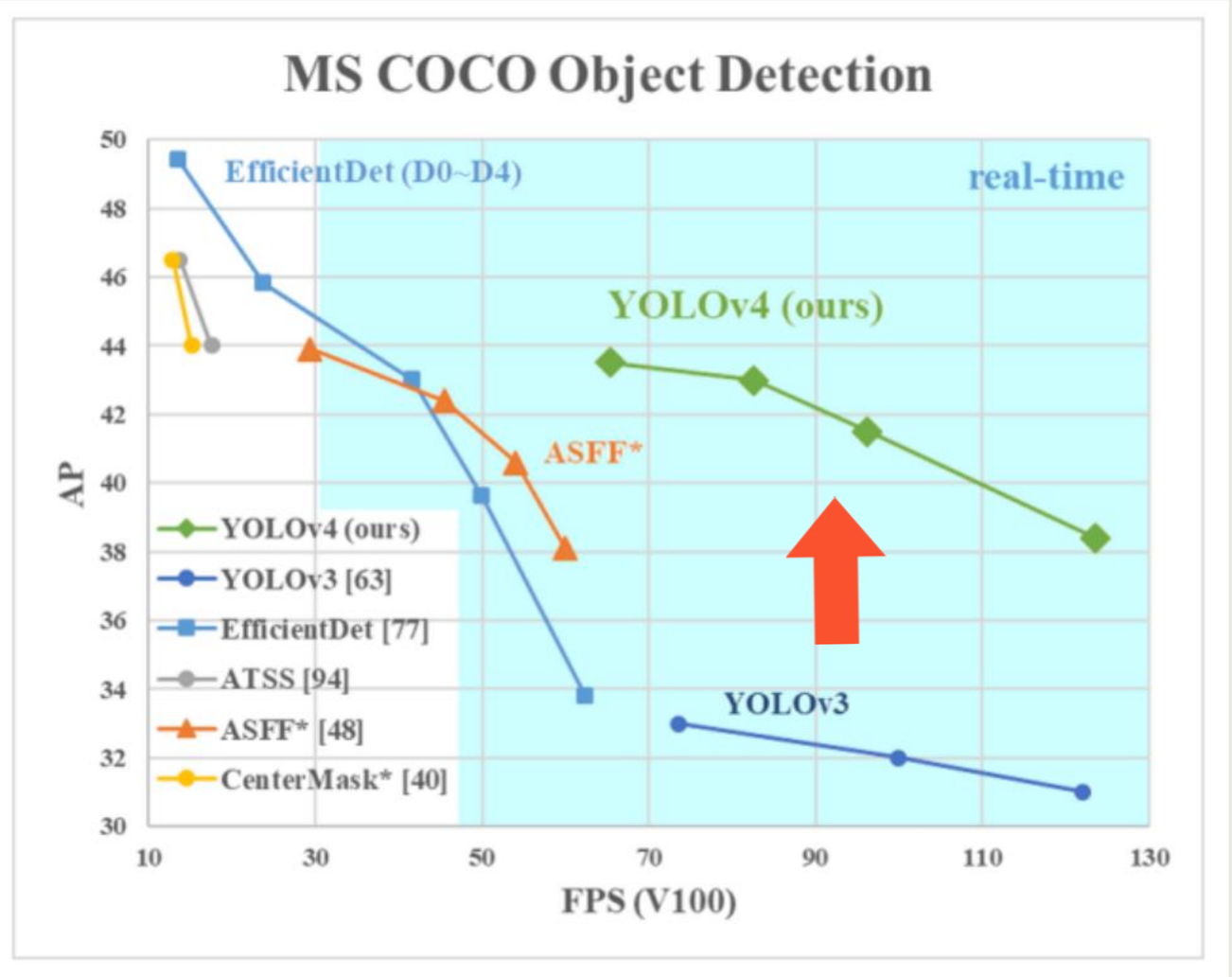
2. detector 훈련 시 최신기법(BoF, Bos)가 미치는 영향 분석

3. 단일 GPU 훈련에 적합하도록 최신 기법 수정

Introduction

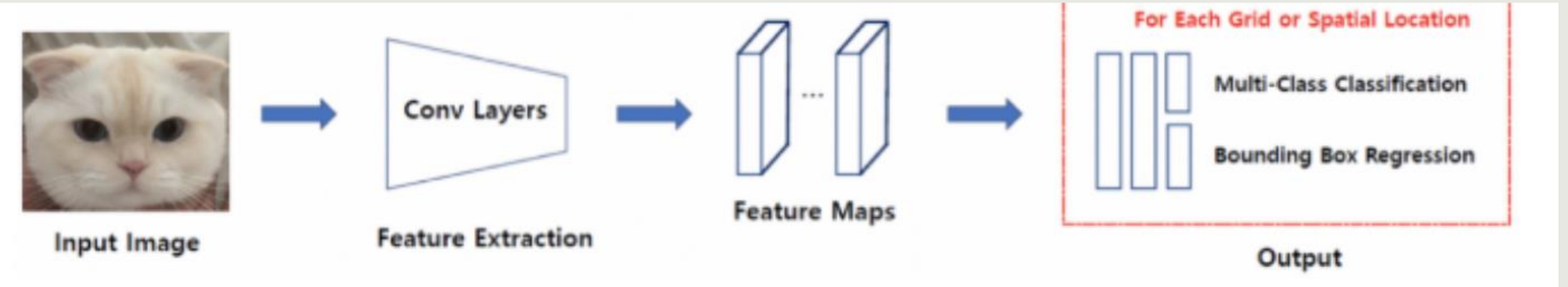
2020년 4월 23일 발표

- CNN기반의 대부분의 객체 탐지모델들은 느리고 정확한 상황에서만 사용이 가능한 것을 극복하여 정확도와 실시간 만족
- 기존의 Yolo v3에 비해서 높은 정확도를 보였다.
- 하나의 GPU만을 사용할 수 있도록 하였다

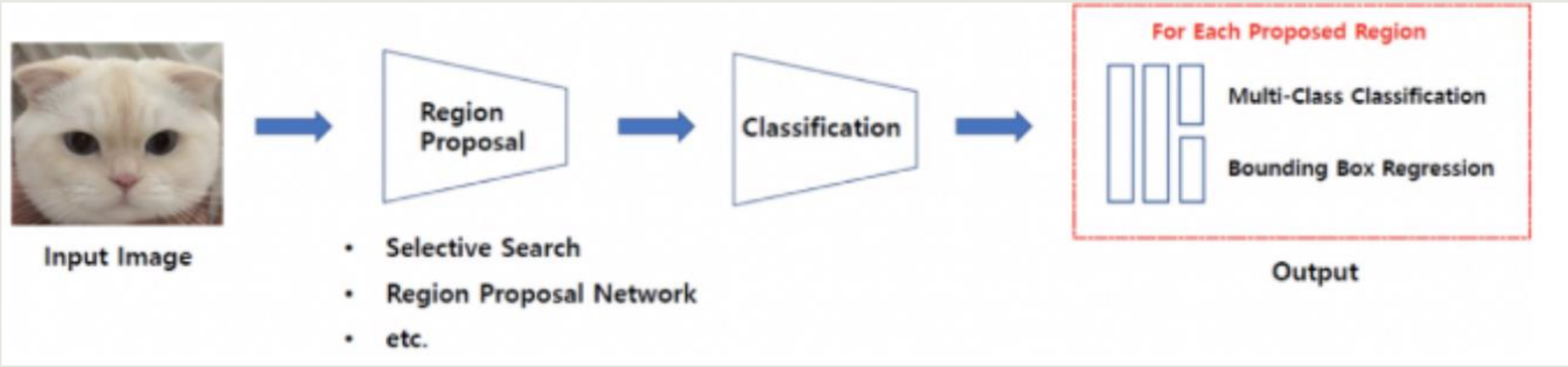


1-Stage Detector & 2-Stage Detector 차이점

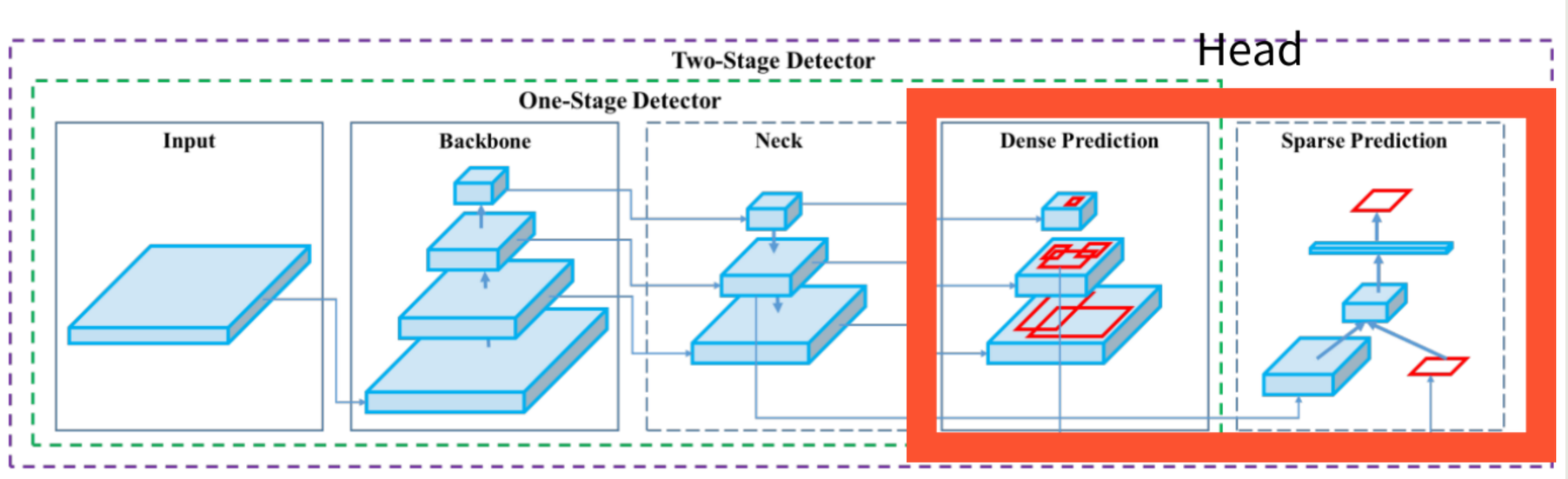
1-stage detector : Regional Proposal과 Classification이 동시에 이루어진다.



2-stage detector : Regional Proposal과 Classification이 순차적으로 이루어진다.



Object Detector의 구조



Backbone : 이미지의 피처를 추출해 내는 pre-trained network를 이용하여 파처 맵 생성

Neck : 피라미드 형태를 통해 검출된 객체에 다양한 스케일 변화를 주는 단계

Head : 실질적인 검출이 이루어지는 곳으로 1Stage Detector 와 2-Stage Detector 로 나뉜다

Bag of Freebies

Bag of Freebies (pre-processing + training strategy)

Training Phase

- Call methods that only change the **training** strategy or only increase the **training cost** as "BoF"

Data Augmentation	Regularization	Loss Function
• Random erase	• DropOut	• MSE
• CutOut	• DropPath	• IoU
• MixUp	• Spatial DropOut	• GIoU **Generalized**
• CutMix	• DropBlock	• CIoU **Complete**
• Style transfer GAN		• DIoU **Distance**

전처리나 학습단계에서 훈련 비용을 증가시켜 Detector의 정확도를 높이는 방법들을 의미

Bag of Specials

후처리 단계에서 훈련 비용을 약간 증가시켜 Detection의 정확도를 높이는 방법

architecture related

Bag of Specials (plugin modules + post-processing)

Inference Phase

- Call methods that only increase the **inference cost** but can improve the accuracy as "BoS"

Enhancement of receptive field

- Spatial Pyramid Pooling
- ASPP (dilated conv)
- Receptive Field Block (RFB)

Feature Integration

- Skip-connection
- Feature Pyramid Network
- SFAM (Scale-wise Feature Aggregation Module)
- ASFF (adaptively spatial feature fusion)
- BiFPN

Activation function

- ReLU
- Leaky ReLU
- Parametric ReLU
- ReLU6
- Swish
- Mish

Attention Module

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

Normalization

- Batch Norm (BN)
- Cross-GPU Batch Norm (CGBN or SyncBN)
- Filter Response Normalization (FRN)
- Cross-Iteration Batch Norm (CBN)

Post Processing

- NMS
- Soft NMS
- DIoU NMS

Methodology : Selection of architecture

1. 기존 yolo의 small_object를 잘 찾지 못하는 것을 개선 하기 위해 이미지 크기를 키웠다.

2. Receptive Field를 높여주기 위해 레이어 층을 쌓았다.

3. 하나의 이미지에서 여러 크기의 오브젝트를 잘 찾기 위해 파라미터 수를 늘렸다.

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

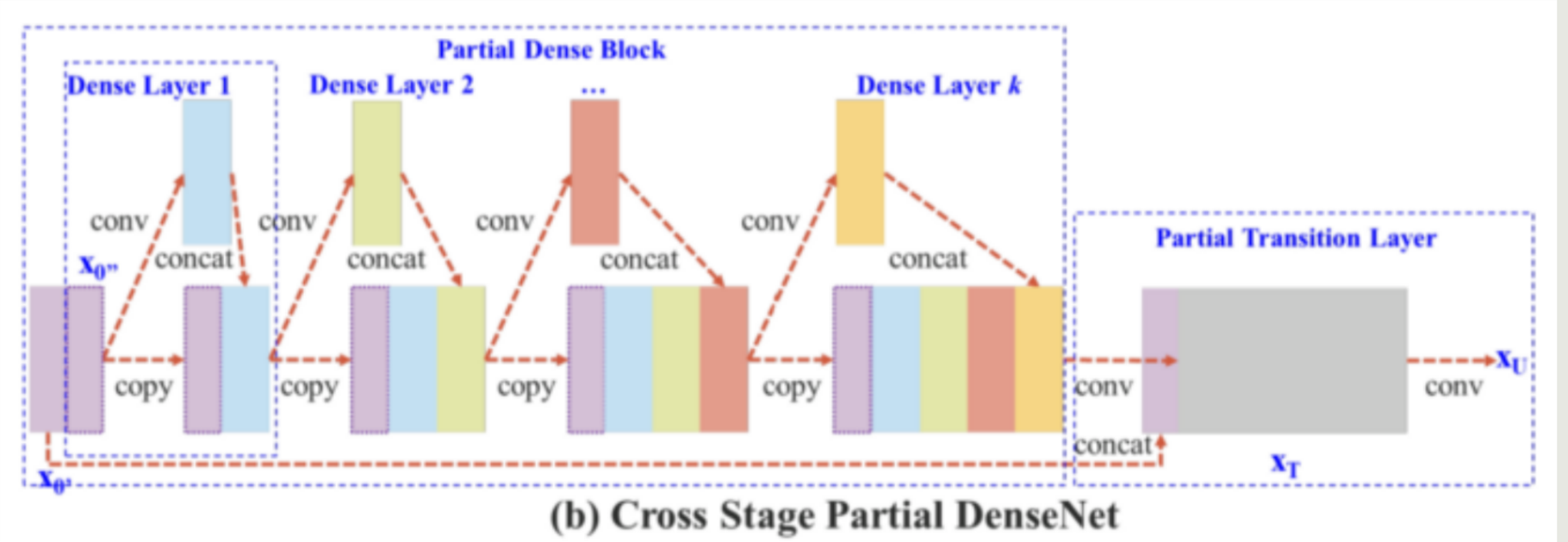
CSP Darknet53

● 기존 Darknet53의 Cross Stage Partial DenseNet 적용

● 인풋 레이어를 두 파트로 나누어 절반만 연산하고 나머지 절반은 맨 뒤에서 합쳐준다

Densenet의 출력값 연결을 통한 재사용을 유지하면서도 Gradient 정보가 많아지는 것을 방지한다

● 네트워크의 계산 비용 및 메모리 사용량을 획기적으로 낮추었고 정확도를 향상시켰다



Methodology : Selection of BoF & 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 [36], Spatial DropOut [79], or DropBlock
- **Normalization of the network activations by their mean and variance**: Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
- **Skip-connections**: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

→ PReLU, SELU : 학습이 어려워 제외
ReLU6는 quantiztion network를 위해 고안되어 제외

→ DropBlock의 저자가 직접 실험을 하여 가장 좋은 결과임을 입증

→ 하나의 GPU를 이용한 훈련 전략에 중점을 두어 syncBN은 제외

Methodology : Additional improvements

Bag of Freebies

Mosaic agumentation : 4개의 이미지를 1개의 이미지로 합쳐 한개의 인풋으로 4개의 이미지를 배울 수 있다. 배치사이즈를 적게가져가도 큰 배치사이즈를 가져가는 효과를 얻을 수있다.

SAT : 2단계로 나누어 1단계에서 neural network는 network의 weight 대신에 원본 이미지를 변경하고(adversarial attack) 2단계에선 neural network은 변경된 이미지에서 정상적인 방식으로 object를 검출

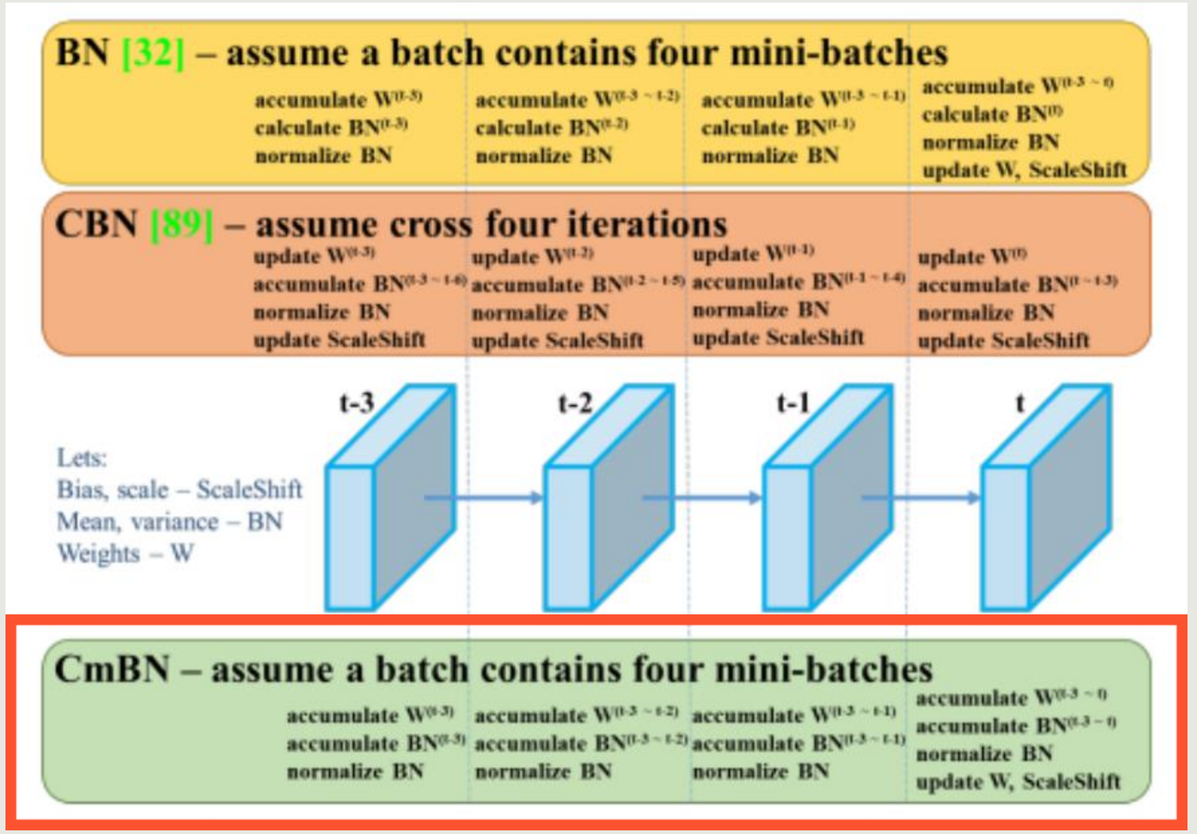
Methodology : Additional improvements

Bag of Specials

BN: batch size가 작을 경우 examples들에 대한 정확한 추정이 어려우므로 효율성이 저하

CBN: 추정 정확도의 향상을 위해, 이전 iteration들의 통계량을 함께 활용

CmBN : CBN에 변경된 버전으로, 단일 batch 내에서 mini-batches 사이에 대한 통계량 수집



Methodology : Additional improvements

Bag of Specials

Modified SAM : Max-Pooling, Average-Po oling -> Convlution

Modified PAN : shortcut connection을 concatenation으로 변경

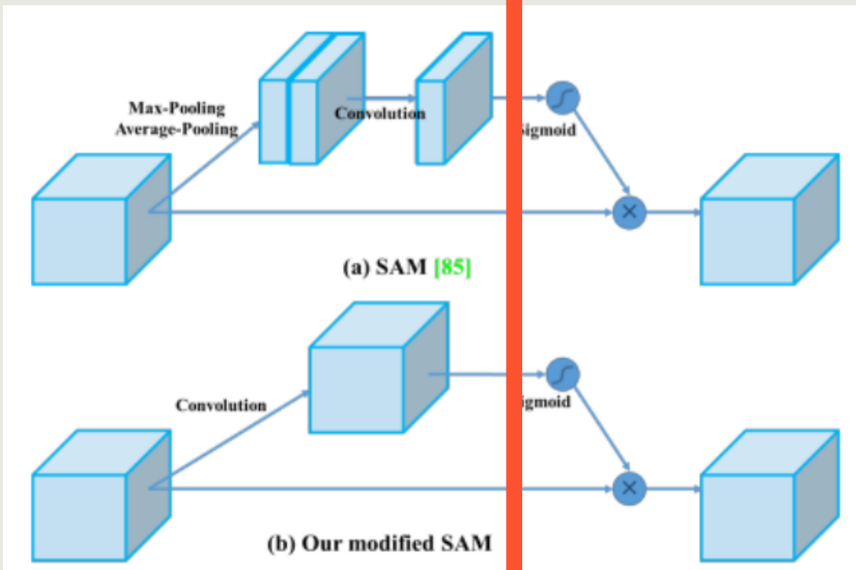


Figure 5: Modified SAM.

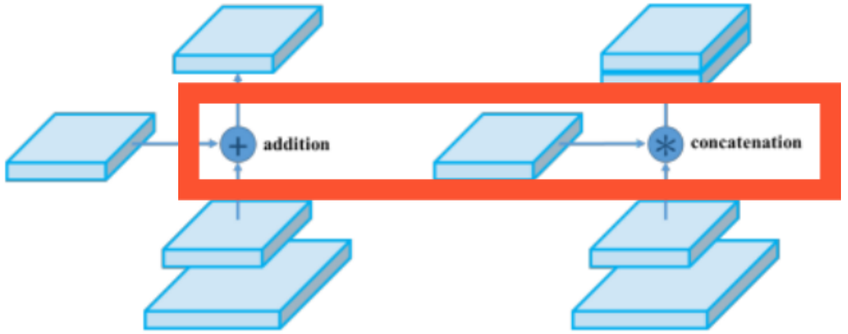
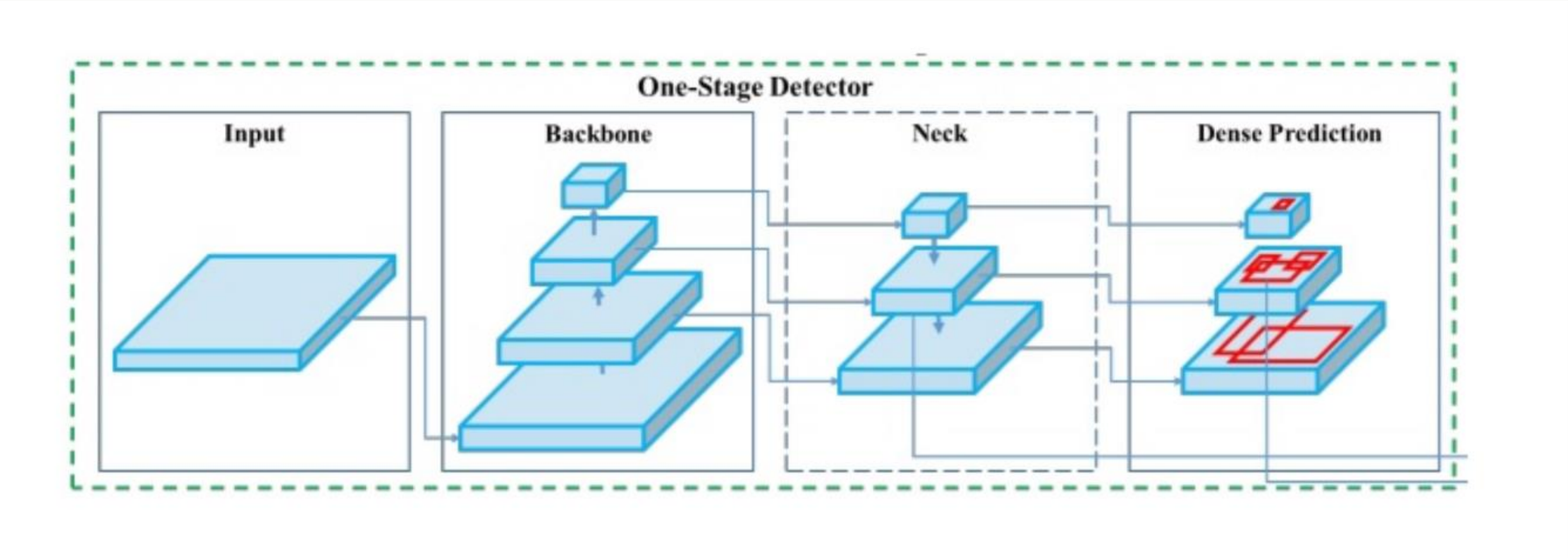


Figure 6: Modified PAN.

Yolov4 architecture



CSPDarknet53 SPP + PAN YOLOv3

Yolov4 = CspDarknet53 + SPP + PAN(+ SAM) + BOF + BOS

Experiments

In ImageNet image classification experiments, the default hyper-parameters are as follows: the training steps is 8,000,000; the batch size and the mini-batch size are 128 and 32, respectively; the polynomial decay learning rate scheduling strategy is adopted with initial learning rate 0.1; the warm-up steps is 1000; the momentum and weight decay are respectively set as 0.9 and 0.005. All of our BoS experiments use the same hyper-parameter as the default setting, and in the BoF experiments, we add an additional 50% training steps. In the BoF experiments, we verify MixUp, CutMix, Mosaic, Bluring data augmentation, and label smoothing regularization methods. In the BoS experiments, we compared the effects of LReLU, Swish, and Mish activation function. All experiments are trained with a 1080 Ti or 2080 Ti GPU.

In MS COCO object detection experiments, the default hyper-parameters are as follows: the training steps is 500,500; the step decay learning rate scheduling strategy is adopted with initial learning rate 0.01 and multiply with a factor 0.1 at the 400,000 steps and the 450,000 steps, respectively; The momentum and weight decay are respectively set as 0.9 and 0.0005. All architectures use a single GPU to execute multi-scale training in the batch size of 64 while mini-batch size is 8 or 4 depend on the architectures and GPU memory limitation. Except for using genetic algorithm for hyper-parameter search experiments, all other experiments use default setting. Genetic algorithm used YOLOv3-SPP to train with GIoU loss and search 300 epochs for min-val 5k sets. We adopt searched learning rate 0.00261, momentum 0.949, IoU threshold for assigning ground truth 0.213, and loss normalizer 0.07 for genetic algorithm experiments. We have verified a large number of BoF, including grid sensitivity elimination, mosaic data augmentation, IoU threshold, genetic algorithm, class label smoothing, cross mini-batch normalization, self-adversarial training, cosine annealing scheduler, dynamic mini-batch size, DropBlock, Optimized Anchors, different kind of IoU losses. We also conduct experiments on various BoS, including Mish, SPP, SAM, RFB, BiFPN, and Gaussian YOLO [8]. For all experiments, we only use one GPU for training, so techniques such as syncBN that optimizes multiple GPUs are not used.

Classifiction은 imagenet으로 Detection은 CoCo데이터셋으로 성능 평가

Experiments - Classifier(BoF)

CSPResNext을 backbone으로 이용했을 때,
BoF와 Mish가 Classifier의 정확도에 주는 영향

BoF중 cutmix, mosaic, label smoothing 이 성능이 좋앗고 활성화 함수인 Mish를 사용하였을때 보다 향상된 정확도를 보여주었다

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%
						✓	[illegible]	[illegible]
	✓	✓		✓			**78.5%**	**94.8%**
	✓	✓		✓		✓	**79.8%**	**95.2%**

CSPDarknet을 backbone으로 이용했을 때,
BoF와 Mish가 Classifier의 정확도에 주는 영향

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%**

Experiments - Detector(BoF)

BoF의 다양한 Detector 피처

기법	설명
S: Eliminate grid sensitivity	• YOLOv3에서 object의 좌표 계산식을 변경하여, object가 검출되지 않는 grid에 대한 영향을 제거
M: Mosaic data augmentation	• 훈련 동안에 단일 이미지 대신 4개의 image mosaic을 이용
T: IoU threshold	• IoU(truth, anchor) > IoU_threshold 인 경우에 single ground truth에 대해 multiple anchors를 사용
GA: Genetic algorithms	• 전체 시간 중 처음 10% 기간 동안 network 학습 시 최적의 hyper-parameters 선택을 위해 Genetic Algorithms을 적용
LS: Class label smoothing	• sigmoid activation을 이용하여 Class label smoothing을 수행
CBN: CmBN	• 단일 mini-batch 내 statistic 수집이 아닌, 전체 batch 내 statistic 수집을 위해 Cross mini-Batch Normalization을 이용
CA: Cosine annealing scheduler	• 훈련 시 learning rate를 변경하는 Cosine annealing scheduler를 적용
DM: Dynamic mini-batch size	• Random training shapes을 이용하여 작은 해상도의 훈련 중에는 mini-batch 크기를 자동으로 확대
OA: Optimized Anchors	• 512 x 512 크기의 network 해상도를 이용한 훈련 시 최적화된 anchors를 사용
GIoU, CIoU, DIoU, MSE	• bounded된 box를 regression 시 서로 다른 종류의 loss 알고리즘들을 사용

Experiments - Detector(BoF)

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%**
✓									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%**

CSPResNet-PANet-SPP 를 Detector 로 이용하였을때

Loss를 MSE로 고정시켯을때 실험결과 M, GA, CBN, CA를 포함하는 것이 정확도 향상에 유리하였다

S, M, IT, GA를 함께 적용하고 Loss를 CIoU로 주었을때 정확도 향상에 유리

Experiments - Detector(BoF)

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%**

CSPResNet-PANet-SPP 를 Detector 로 이용하였을때

OA를 적용하는 것도 정확도 향상의 유리하다

GIoU와 CIoU loss를 적용하는 것이 정확도 향상에 유리하고 GIoU와 CIoU는 같은 결과를 보였다

Experiments - Detector(BoS)

BoS의 다향한 피처들이 Detector훈련에 주는 영향

PANet-SPP-SAM을 같이 사용하였을 경우 가장 우수한 정확도를 보였다.

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%

SPP : 컨보루션레이어를 통해 추출된 피처맵을 n개의 피라미드를 이용하여 고정된 길이의feature representation생성

SAM : 채널 어텐션에서 피처맵을 입력받아 maxpool과 averagepool을 개별적용시켜 2개의 피처맵 생성

Experiments - Detector

서로 다른 Backbone model들이 정확도에 주는 영향

Image Classification 정확도는 CSPResNet가 높았으나/ Object Detection 정확도는 CSPDarknet이 더 좋았다.

--> Classification에서 좋은 정확도를 보여도 Detection에서도 항상 우수한것은 아니다

	✓	✓	✓	78.5% 94.8%
	✓	✓	✓	✓ 79.8% 95.2%

Resnet

	✓	✓	✓	77.8% 94.4%
	✓	✓	✓	✓ 78.7% 94.8%

DarkNet

detector 훈련을 위해 서로 다른 classifier로 pre-trained된 W 사용

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

Experiments - Detector

서로 다른 Backbone model들이 정확도에 주는 영향

CSPResNeXt는 BoF와 Mish사용시 Classification에선 좋은 결과를 보였지만 Object Detection에선 오히려 정확도가 저하되었다

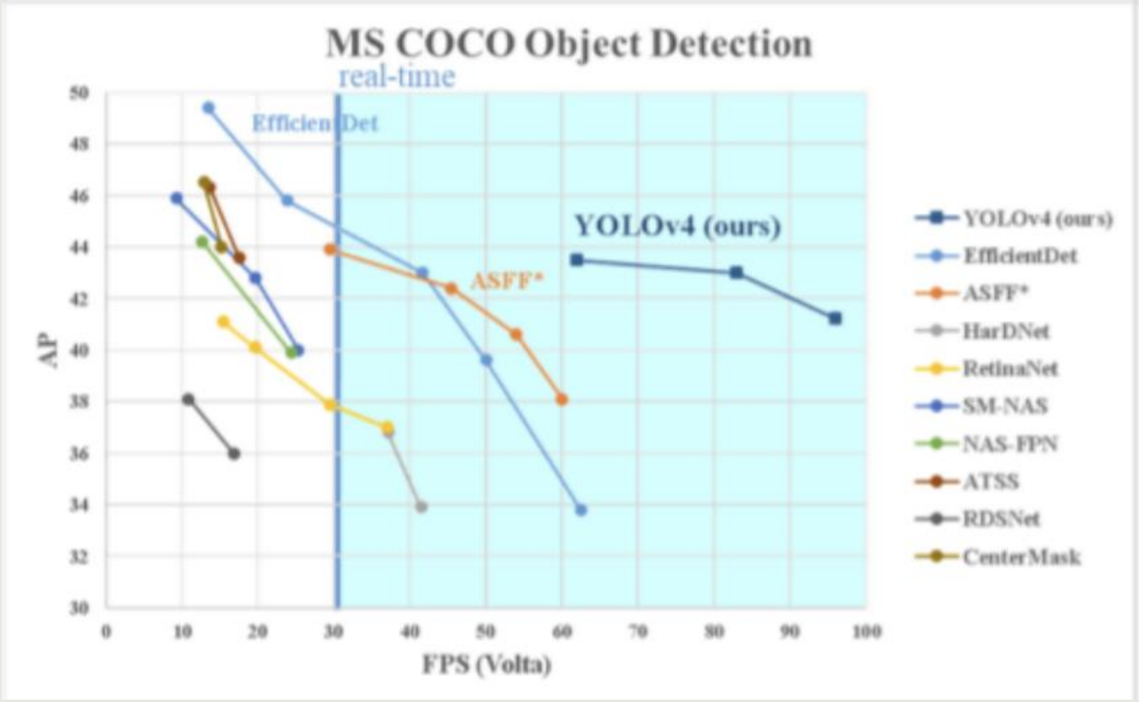
하지만 CSPDarknet에선 Classification정확도 뿐만 아니라 Object Detection에서도 좋은 성능을 보였다

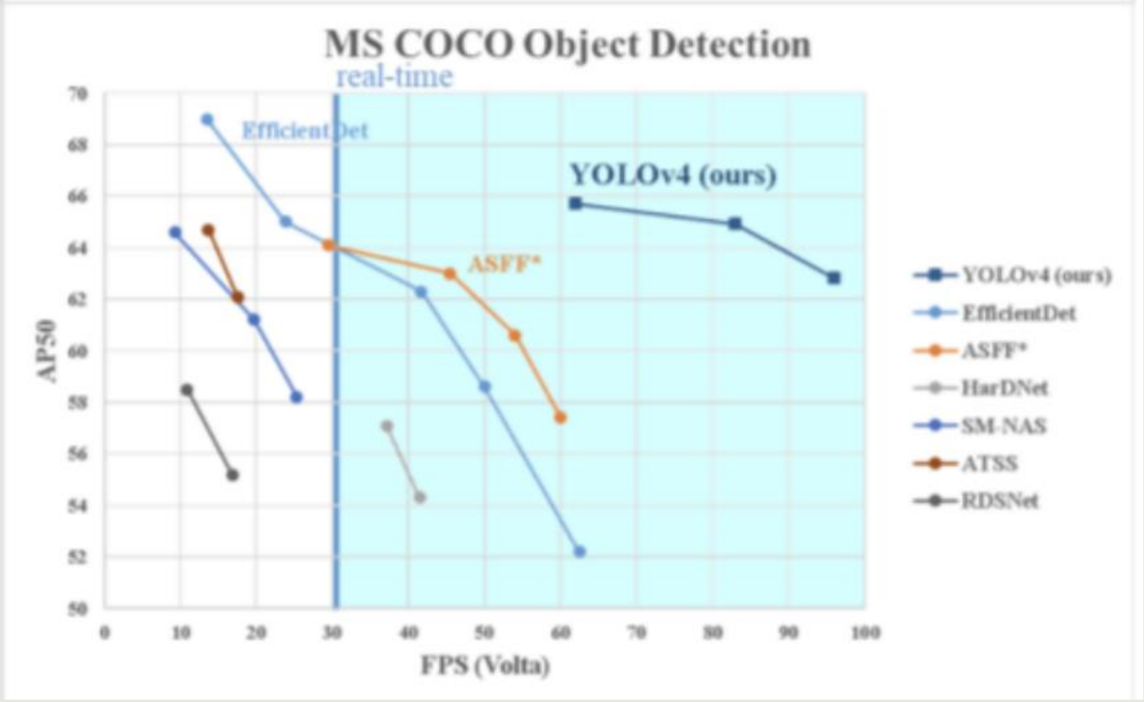
detector 훈련을 위해 서로 다른 classifier로 pre-trained된 W 사용

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

->CSPDarknet이 CSPResNeXt보다 Detector에 적합한 Backbone 이라고 할 수 있다.

Result





다른 Detection 모델들 보다 속도 및 정확도 측면에서 가장 빠르고 정확 한 것을 확인

Conclusion

1. 학계에서 다뤄진 다양한 기법들(Bof ,Bos)를 Yolo에 적용하여 빠르고 정확한 Detector 제안

2. 모든 연구자들이 사용할 수 있도록 1개의 GPU만을 이용할 수 있도록 하였다

3. 제안한 Detector는 vram 8g ~ 16g 로 기존 GPU에서 사용가능, 광범위하게 적용가능

4. 정확도 개선을 위해 많은 feature들을 검증하여 선택하였고 이러한 feature들을 향 후 연구개발을 위한 모범 사례로 사용이 가능하다


Q & A