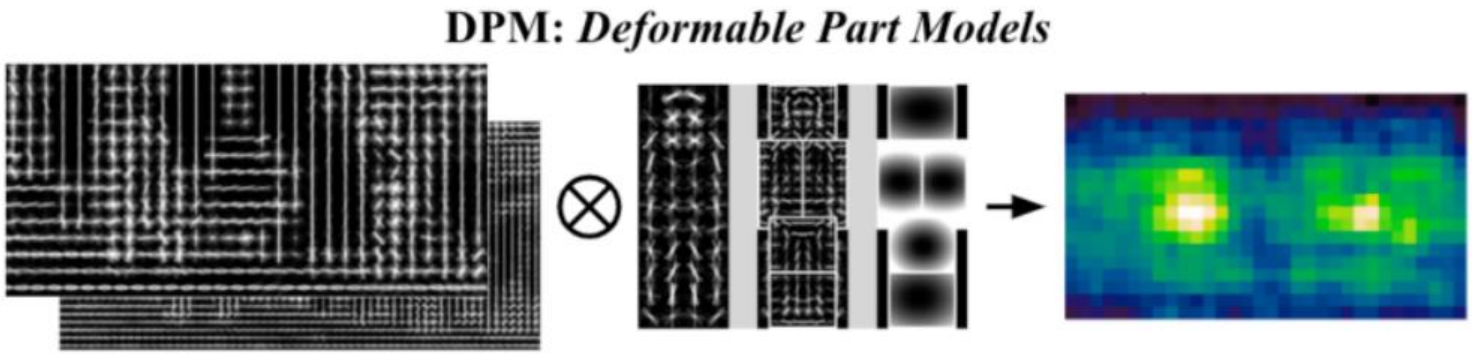
YOLO
You Only Look Once

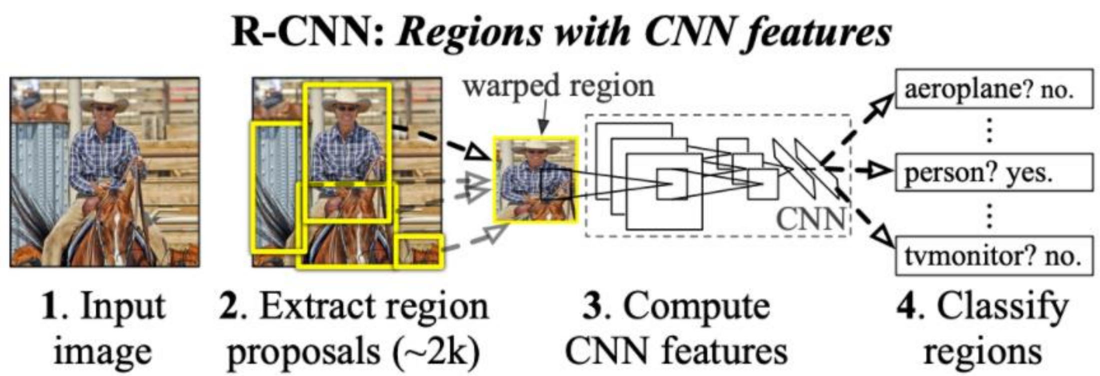
목차
CONTENTS

00

Why YOLO?

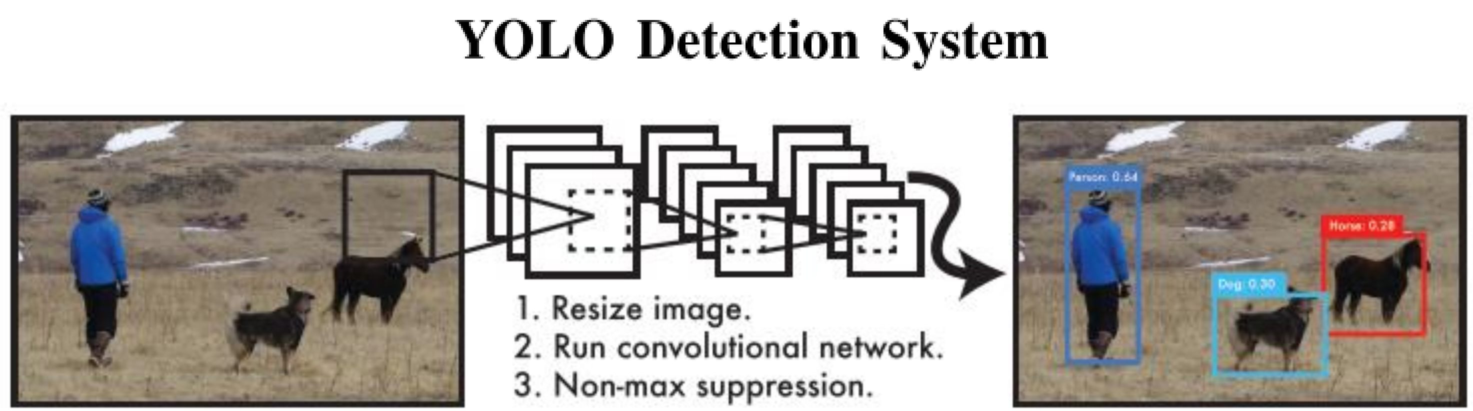
Object Detection





Object를 Detect하기 위한 기존의 Sliding window, DPM, R-CNN 같은 region-based classifiers는 Real-time object detect가 힘들다.

Object Detection



이미지 내의 Bounding Box(Bbox)와 class probability를 하나의 regression problem으로 묶는 one-stage detection 제안

이미지를 한번 보는 것으로
객체 종류와 위치 정보를 한번에 빠르게 추측

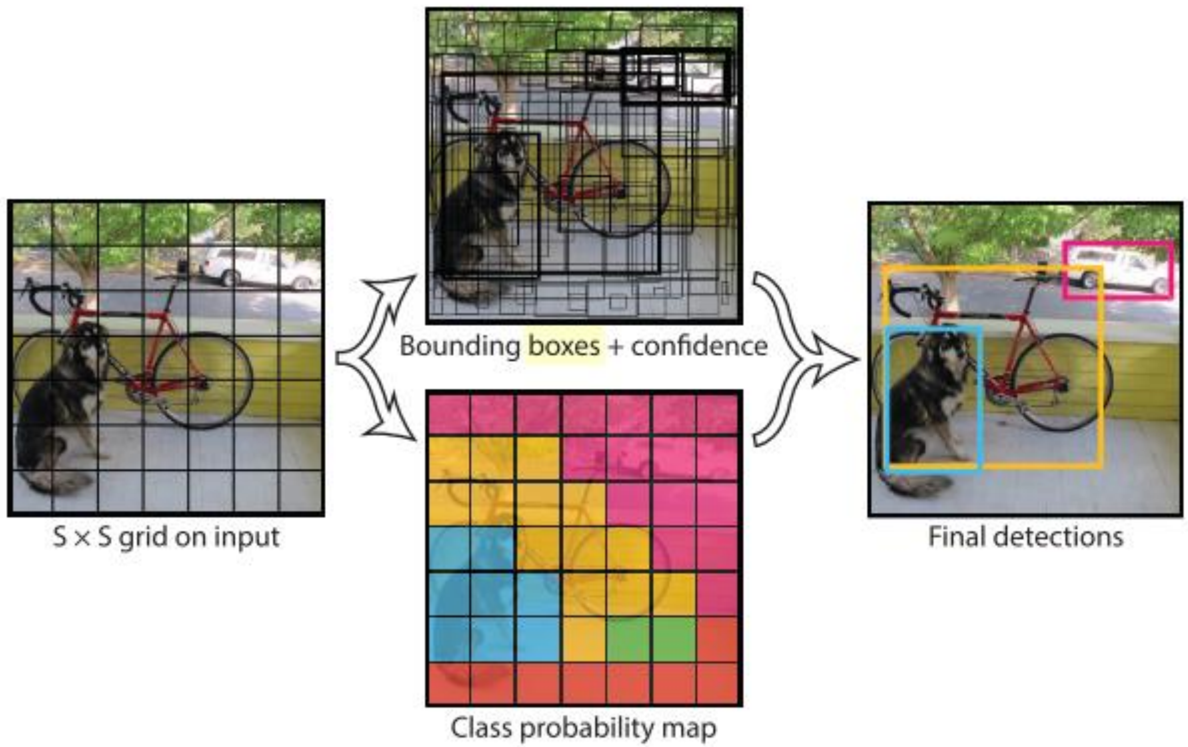
다른 도메인에 적용 가능

Unified detection 이 가능한 모델 YOLO!

01
You Only Look Once: Unified, Real-Time Object Detection

Unified Detection

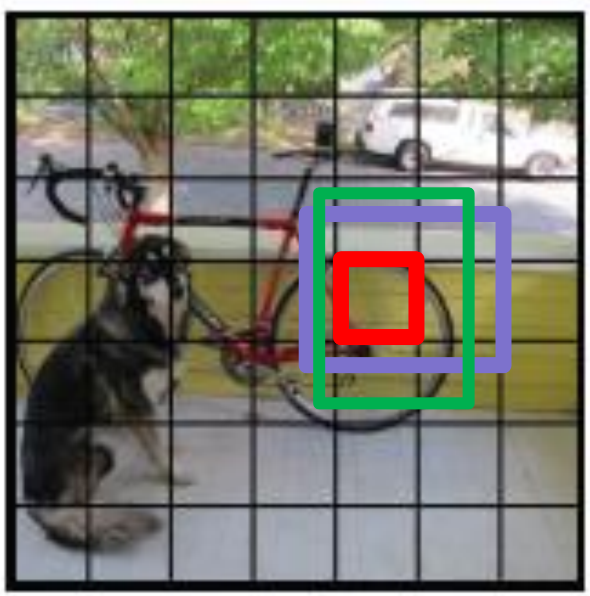
The Model



Unified Detection

YOLO on PASCAL VOC

S = 7
B = 2
C = 20 (PASCAL VOC has 20 labelled classes)



S × S grid on input

x_i
y_i
w_i
h_i
P_c

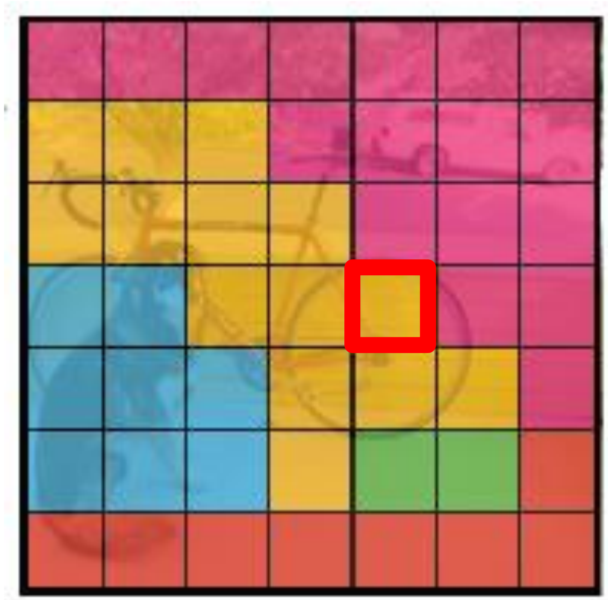
- x_i, y_i: Bbox 중심좌표 위치
- w_i, h_i: Input image W, H 로 normalize
- P_c → $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$

x_i
y_i
w_i
h_i
p_c

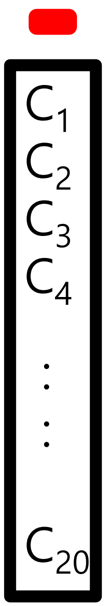
Unified Detection

YOLO on PASCAL VOC

S = 7
B = 2
C = 20 (PASCAL VOC has 20 labelled classes)



Class probability map



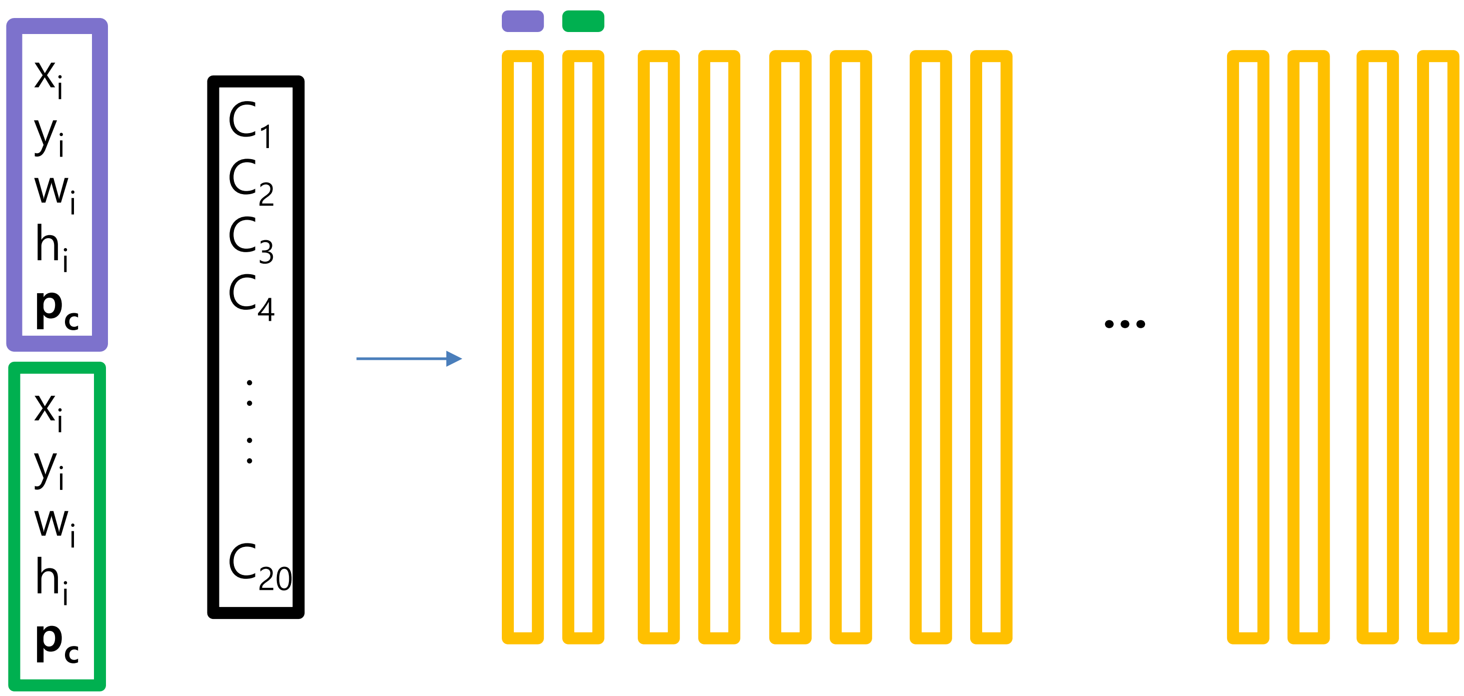
$$\Pr(\text{Class}_i|\text{Object})$$

물체가 Bbox 내에 있을때
Grid cell에 있는 object가
Class에 속할 확률

Unified Detection

class-specific confidence scores

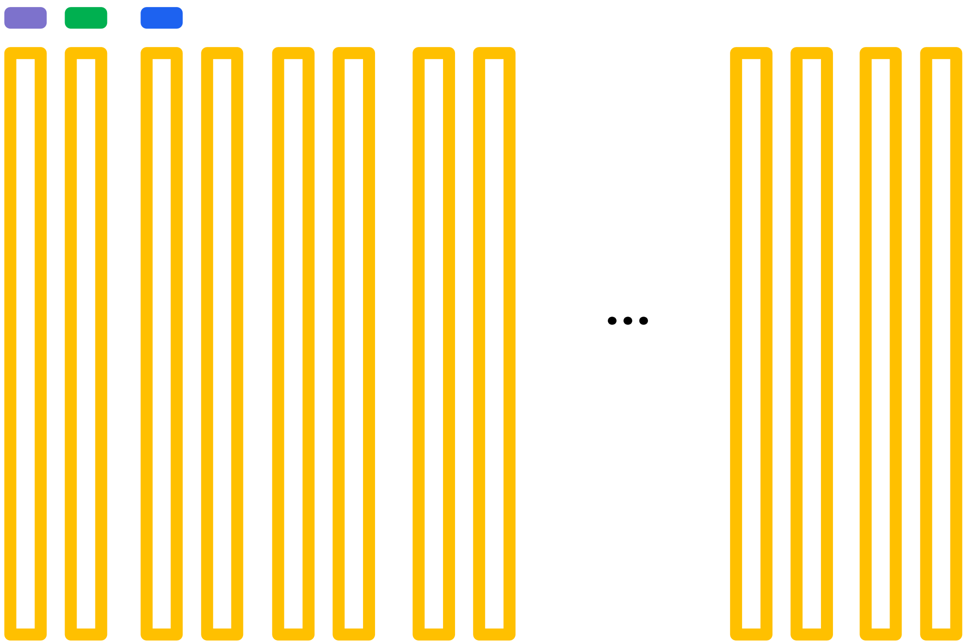
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

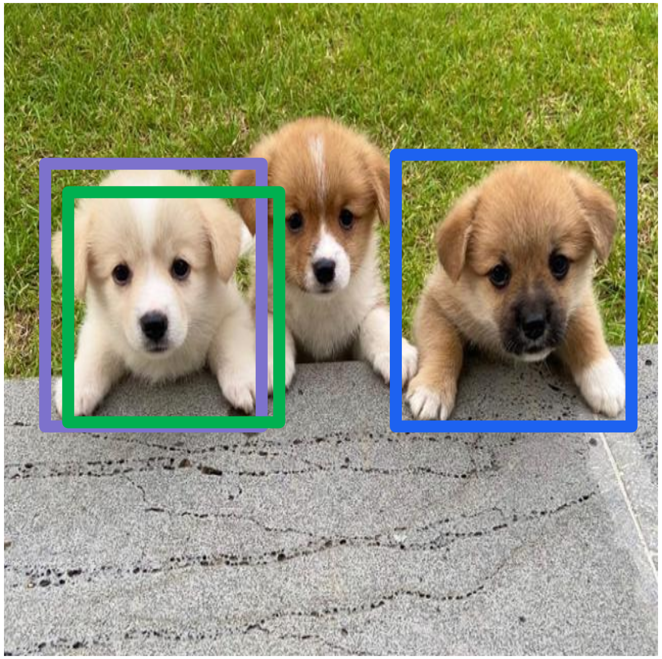


Unified Detection

NMS

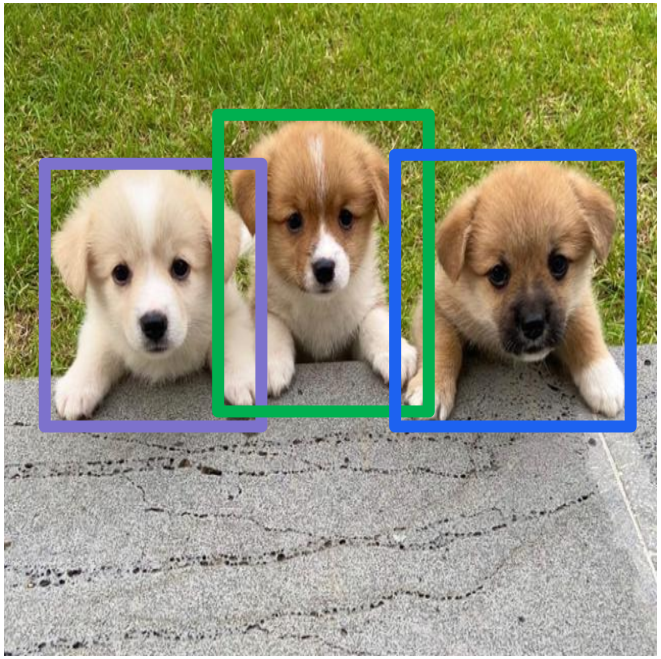
Class 별 진행하여 confidence 내림차순으로 정렬 후
가장 높은 confidence 를 갖는 박스를 기준으로
순차적으로 IOU를 구하여 특정치 이상이면 score 0로 변경





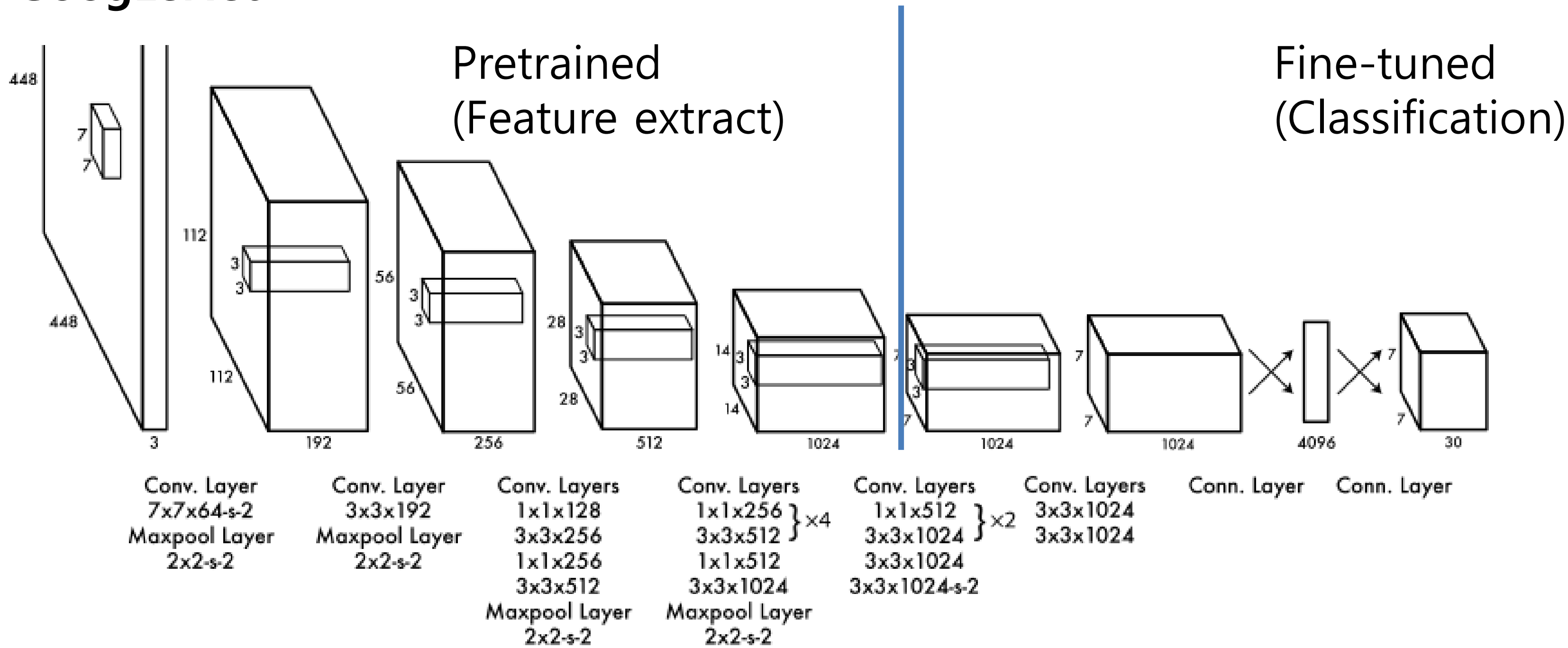
Unified Detection

Final Detection Result



Training : Network Architecture

GoogLeNet

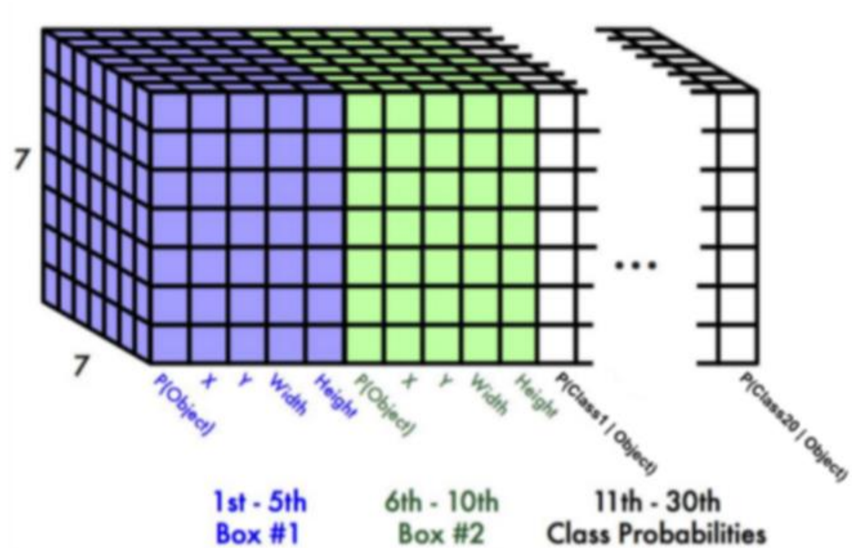


Pretrained : 20 Conv layers
Fine-tuned : 4 Conv layers + 2 fc layers

1x1 convolution을 통한 parameter 개수 감소

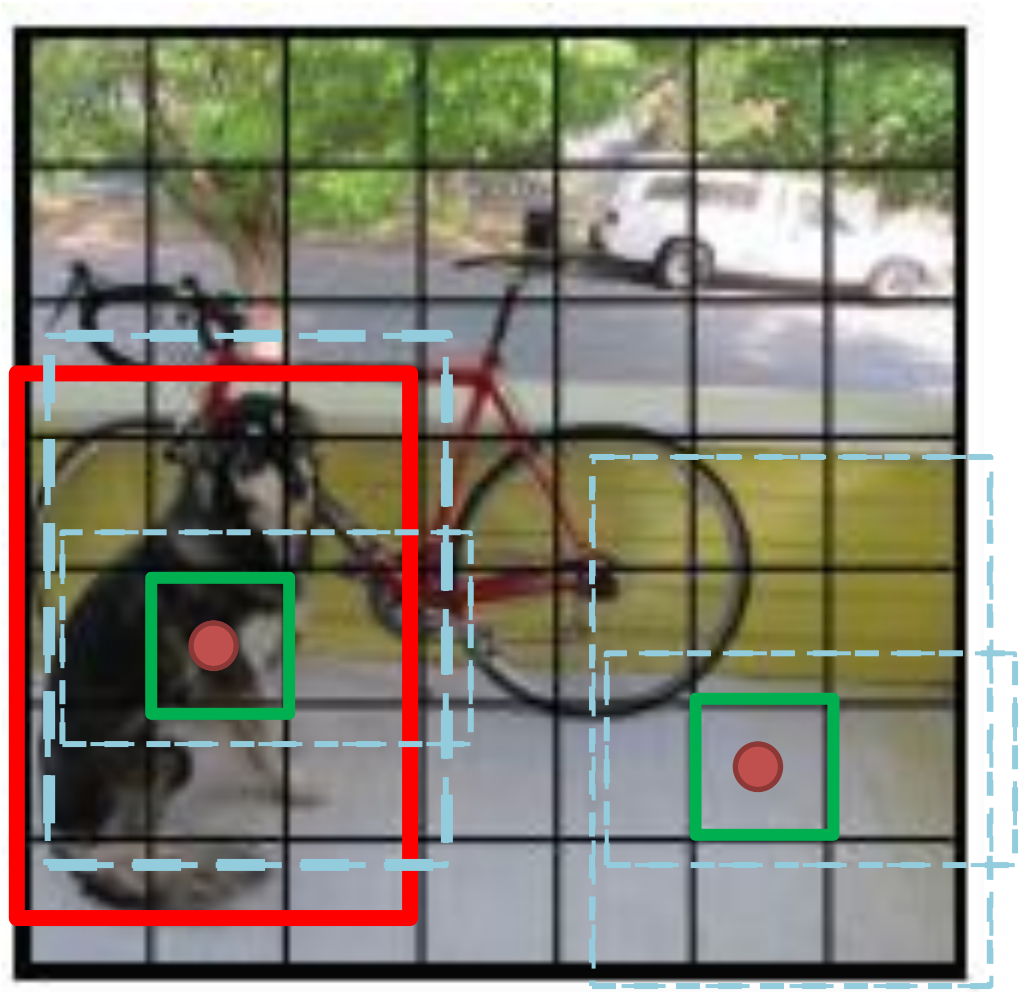
Detection Input size 224x224 -> 448 x 448

Training



Output Tensor

Training



1. Match gt label to appropriate grid cell

2. Class prediction

3. Bbox 생성 후 confidence 조정

가장 큰 confidence score를 가진 bbox를 predictor로 한다.

Training : Loss Function

Sum-squared error coordinate , confidence, class probability loss구함

x_i
y_i
w_i
h_i
P_c

C_1
C_2
C_3
C_4
:
C_{20}

predictor box에 해당

Bbox size 차이 보정

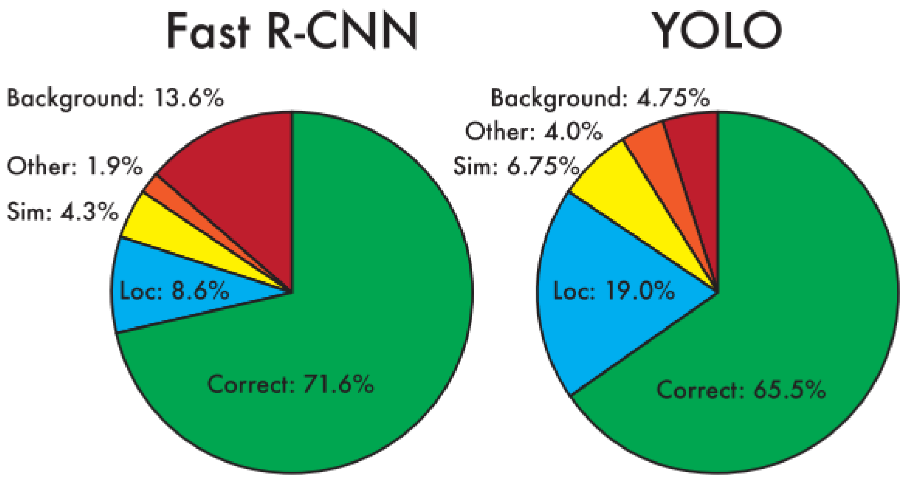
$$\boxed{\lambda_{\textbf{coord}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \boxed{\lambda_{\textbf{coord}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[\boxed{\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2} \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \boxed{\lambda_{\textbf{noobj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \boxed{\mathbb{1}_{i}^{\text{obj}}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

$$\boxed{\lambda_{\textbf{coord}} = 5 \quad \lambda_{\textbf{noobj}} = .5.}$$

bbox coordinate, no object
에 가중치를 부여하여 loss 반영

grid cell에 object가 존재

Experiments



VOC

Localization error가 높음
correct 비율이 낮음

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

다른 detector 과 비교 하였을 때
높은 성능과 속도를 보임

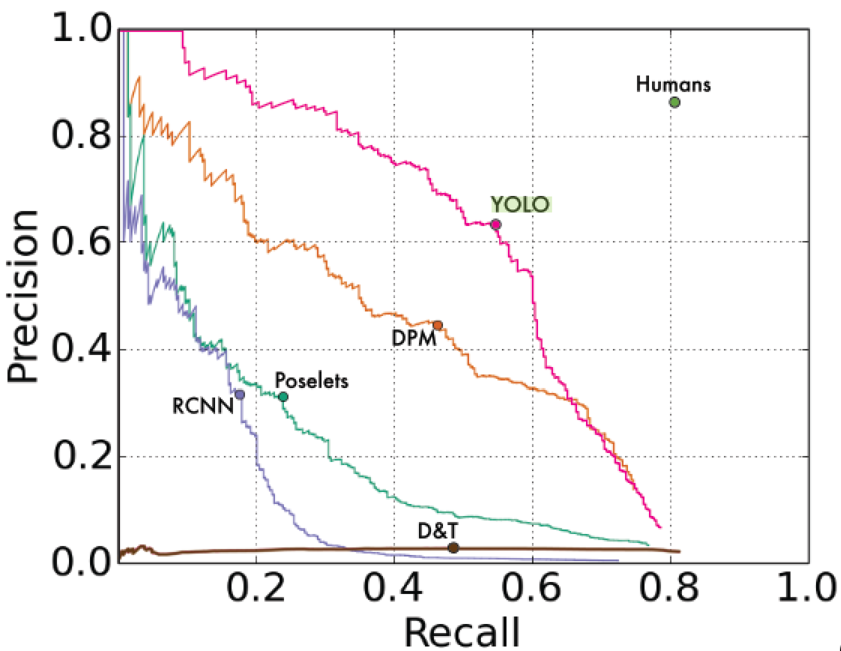
Experiments

Combine to Fast R-CNN

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Experiments



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.

02
YOLO9000: Better, faster, stronger.

Better, faster, stronger

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

Faster	Darknet – 19
	Training for classification
	Training for detection

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

Better, faster, stronger

YOLO v1에서 필요한 개선점들

다른 SOTA detection에 비해 성능이 낮음

Localization errors

낮은 recall 로 object를 전부 찾아내지 못함

Detection class 개수 너무 작고 개수를 늘리기 힘듦

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

Conv layers에 Batch Normalization 추가
-> mAP 2% 상승

Overfitting 없이 dropout 제거

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

Image-net data set @224x224 pretrained

YOLO v1
Detection @448x448로 detection
커진 input size에 적응하면서 성능 저하 예상

YOLO v2
Fine-tuning Classifier network @448x448
for 10epoches
-> almost mAP 4% 상승

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

YOLO v1

- Prior anchor boxes 사용하지 않음
- Use fc layers
- Detection Input image size @448x448
  Down sample 시 가운데에 4개 cell을 가짐
- 각 gird cell 마다 class prediction
  -> mAP 69.5 , recall 81%

YOLO v2

- Anchor boxes 개념 도입
- Use conv layers
- Detection Input image size @416x416
  Down sample 시 가운데에 1개 cell을 가짐
- anchor box 마다 class prediction
  -> mAP 69.2 , recall 88% 성능 개선 여지를 찾음!

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

How many anchor box??

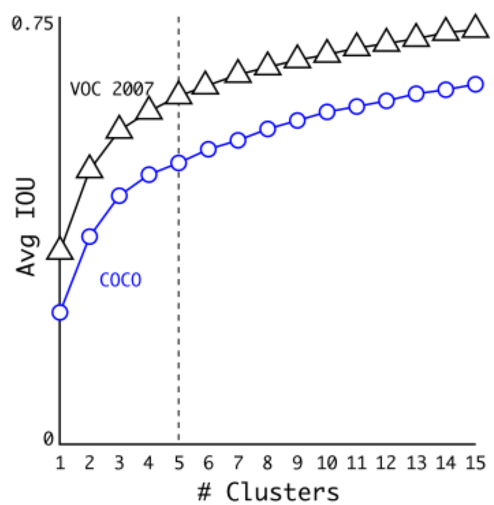
Data set 에 맞는 anchor box 찾자!

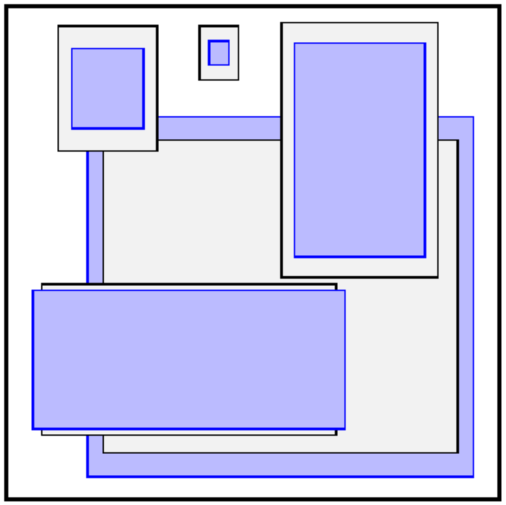
GT Bbox를 **k-means clustering** **하여 Avg IOU 구함**

Centroid와의 Euclidian distance -> IOU

K = 5 is chosen

(good trade off between complexity and high recall)





YOLO v2 : Better

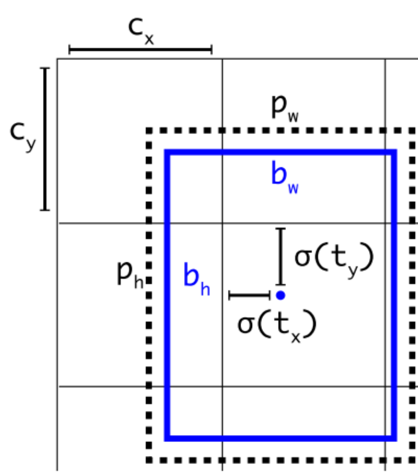
Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

Anchor box가 왼쪽 위에서 정의 되어도 regression 식

$$\hat{G}_x = P_w d_x(P) + P_x$$

$$\hat{G}_y = P_h d_y(P) + P_y$$

에 의하여 Image의 어느 위치로도 이동이 가능한 **Instability** 문제 해결하기 위한 방법 제안



$$b_x = \boxed{\sigma(t_x) + c_x}$$

$$b_y = \boxed{\sigma(t_y) + c_y}$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

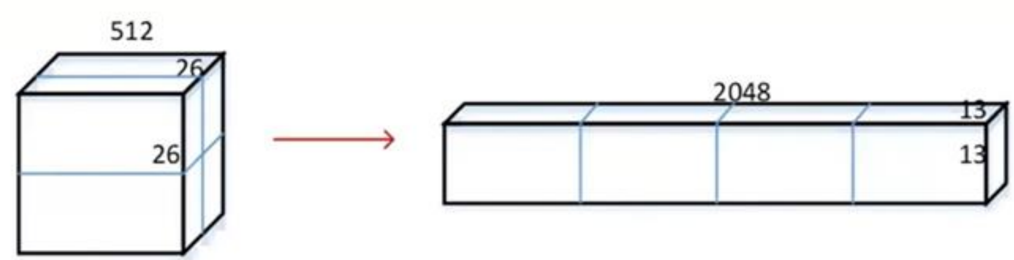
Almost 5% improvement over the version with anchor boxes

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

마지막 단의 feature map (큰 object detect)
Input과 가까운 쪽의 feature map (작은 object detect)

13x13 feature map pooling 전의
26x26 feature map 통째로 13x13 feature map 에
concatenate 후 Bbox 뽑는 방식



1% performance increase

YOLO v2 : Better

Better	Batch Normalization
	High resolution Classifier
	Convolution with Anchor boxes
	Dimension Clusters
	Direct Location prediction
	Fine-grained features
	Multi-scale training

Fc layer -> conv layer

Input size의 강건화

10 batches 마다 randomly image dimension choose

{320,352, … ,608}

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

YOLO v2 : Faster

<table>
<tr><td rowspan="3">Faster</td><td>Darknet – 19</td></tr>
<tr><td>Training for classification</td></tr>
<tr><td>Training for detection</td></tr>
</table>

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	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		$2 \times 2/2$	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			

ImageNet dataset

Darknet
5.58 billion operator :
72.9% top-1
91.2% top-5 accuracy

VGG-16
30.69 billion operator :
90.0% top-5 accuracy

YOLO v2 : Faster

Faster	Darknet – 19
	Training for classification
	Training for detection

Classification

ImageNet 1000 classes for 160 epochs

Standard data augmentation 사용
: random crops, rotations, hue, saturation 등

Initial training @224x224 이후
Fine tuning @448x448 for 10 epochs
-> top-5 acc of 93.3%

Detection

Remove Global avg pooling layer
Add 3x3 conv with 1024 filters + 1x1 conv layer

For PASCAL VOC (20 classes)
5 boxes with 5 coordinates (x, y, w, h, confidence)
-> 125 filters

Epochs: 160, lr : 10^-3 (60, 90 epoch에 lr decay)

YOLO v2 : Experiment

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

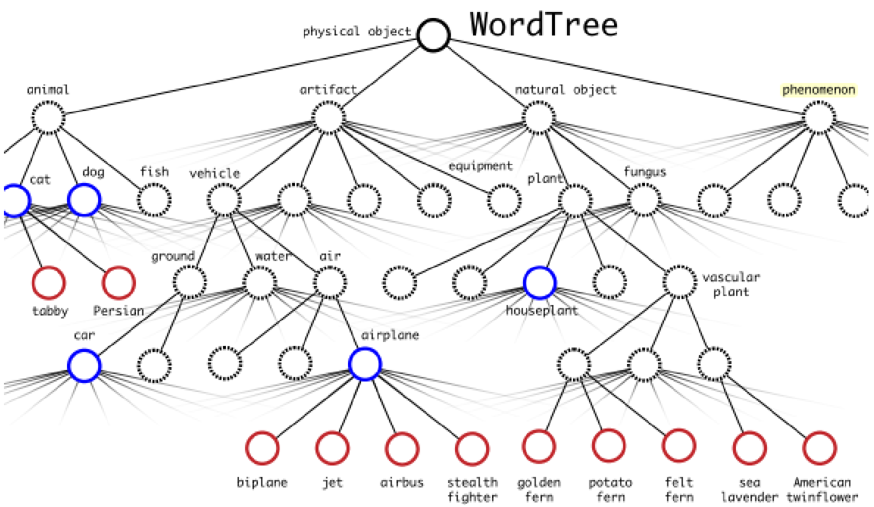
Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

WordNet 의 구조를 Shorter path만을 남기고 나머지 가지 치기하여 Tree 형태로 변형

Root node는 Physical Object



YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

Conditional probability 구하기

$$Pr(\text{Norfolk terrier}|\text{terrier})$$
$$Pr(\text{Yorkshire terrier}|\text{terrier})$$
$$Pr(\text{Bedlington terrier}|\text{terrier})$$

Norfolk terrier 확률 구하기

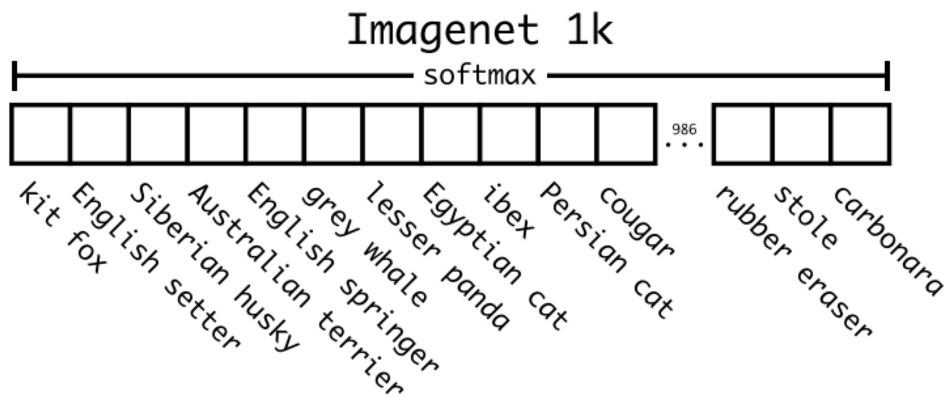
$$\begin{aligned} Pr(\text{Norfolk terrier}) = \; & Pr(\text{Norfolk terrier}|\text{terrier}) \\ & *Pr(\text{terrier}|\text{hunting dog}) \\ & *\ldots* \\ & *Pr(\text{mammal}|Pr(\text{animal}) \\ & *Pr(\text{animal}|\text{physical object}) \end{aligned}$$

YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

Build WordTree 1k (**add intermediate nodes**)
-> 1369개 label
정답 label 을 위로 propagate
자손 node label(Norfolk terrier) 값이 1이면
모든 조상 node label(Dog, mammal, …) 1로 변경

Multiple softmax 방식으로 classification
- 같은 level 끼리 softmax



71.9% top-1 accuracy
90.4% top-5 accuracy

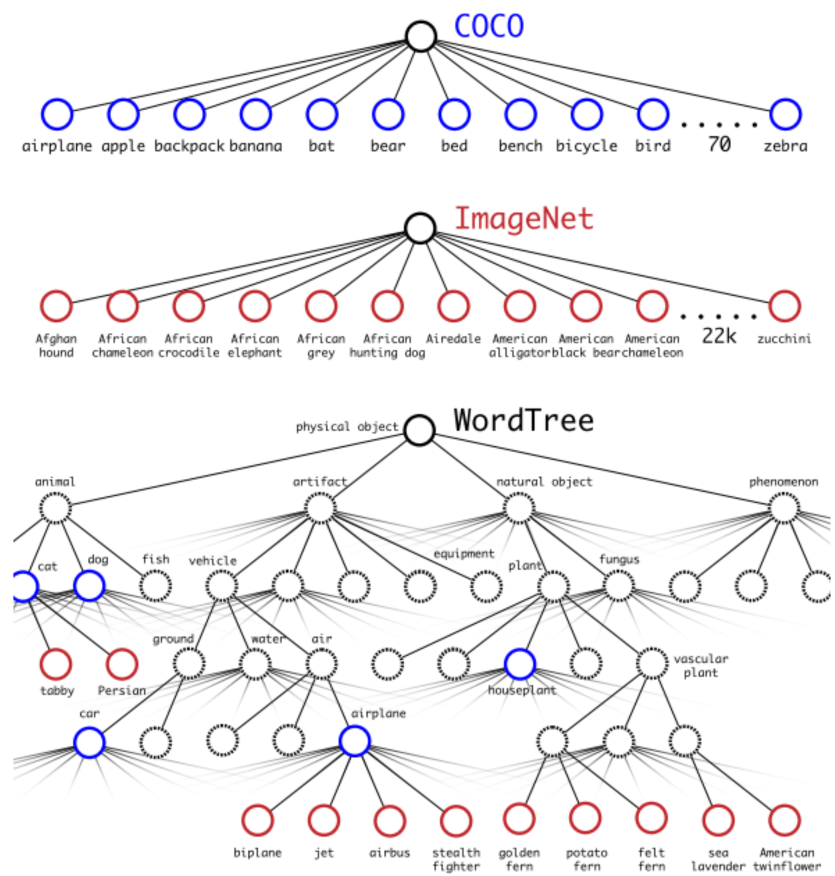
YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

Concept

General
(Higher node)

Specific
(Lower node)



YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

Create New dataset!

COCO detection dataset
\+ Top 9000 classes from full ImageNet release
\+ ImageNet detection challenge dataset
-> 9418 classes

Anchor Box k = 5 -> 3 으로 제한

Detection dataset image
backpropagation 하여 loss 구함

Classification dataset image
Image class에 맞는 level로부터 classification loss만 backpropagate

Bbox를 통해 class를 예측한 것 중 가장 높은 것을 계산

YOLO9000 : Stronger

Stronger	Hierarchical classification
	Dataset Combination With World-tree
	Joint classification and detection

ImageNet(detection data set)과 COCO 는 44 object categories를 공유

한번도 보지못한 disjoint 156 object classes 에 대하여 19.7mAP overall with 16.0mAP

Coco data set의 특성 으로 인하여

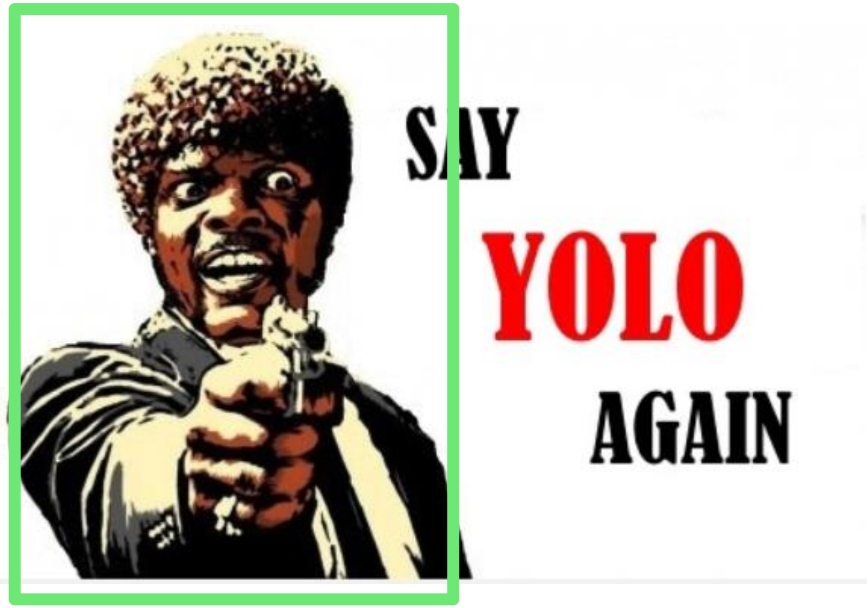
diaper	0.0
horizontal bar	0.0
rubber eraser	0.0
sunglasses	0.0
swimming trunks	0.0
...	
red panda	50.7
fox	52.1
koala bear	54.3
tiger	61.0
armadillo	61.7

옷, 장비 category 학습 불가능

새로운 동물 category 학습 가능

03
YOLO v3
： An Incremental Improvement

Better, NotFaster, Stronger (?)



Brand NEW!

1. Bonding Box Prediction
2. Class Prediction
3. Predictions Across Scales
4. Darknet-53

Bonding Box Prediction

- 기존 YOLO 처럼 각 Bbox의 objectness score를 logistic regression으로 예측
- YOLO v3 에서 Ground truth 와 IOU가 높은 Bbox의 confidence score을 1로 지정
- R-CNN : GT에 가장 높은 IOU 가진 Bbox + thresh hold 이상인 Bbox다 할당
- YOLO v3 : 각 GT 마다 1개의 Bbox 할당

Class Prediction

Multilabel classification

COCO dataset의 80 class에 각각 sigmoid 취해서
binary multiclass classification 진행

Google Open Image : dataset 계층적인 600 classes 존재
ex) person, women 둘 다 1되는 경우가 필요

Logistic regression 으로 예측 모든 class에 대하여 classification 진행

Predictions Across Scales

3가지 Scale에 3가지 Bbox : total 9 box

Tensor size : $N \times N \times [3 * (4 + 1 + 80)]$

$\{\#\text{of Anchors} * [(t_x, t_y, t_w, t_h, P_o) + (C_{1 \text{ to } 80})]\}$

YOLO v2에서 사용한 k-means clustering 사용

COCO dataset 9 clusters :
(10x13) (33x23) (30x61) / small Scale
(30x61) (62x45) (59x119) / medium Scale
(116x90) (156x198) (373x326) / large Scale

YOLO v2 와 비교하면 @416x416 기준 x10배 이상의 box를 사용한다

Darknet – 53

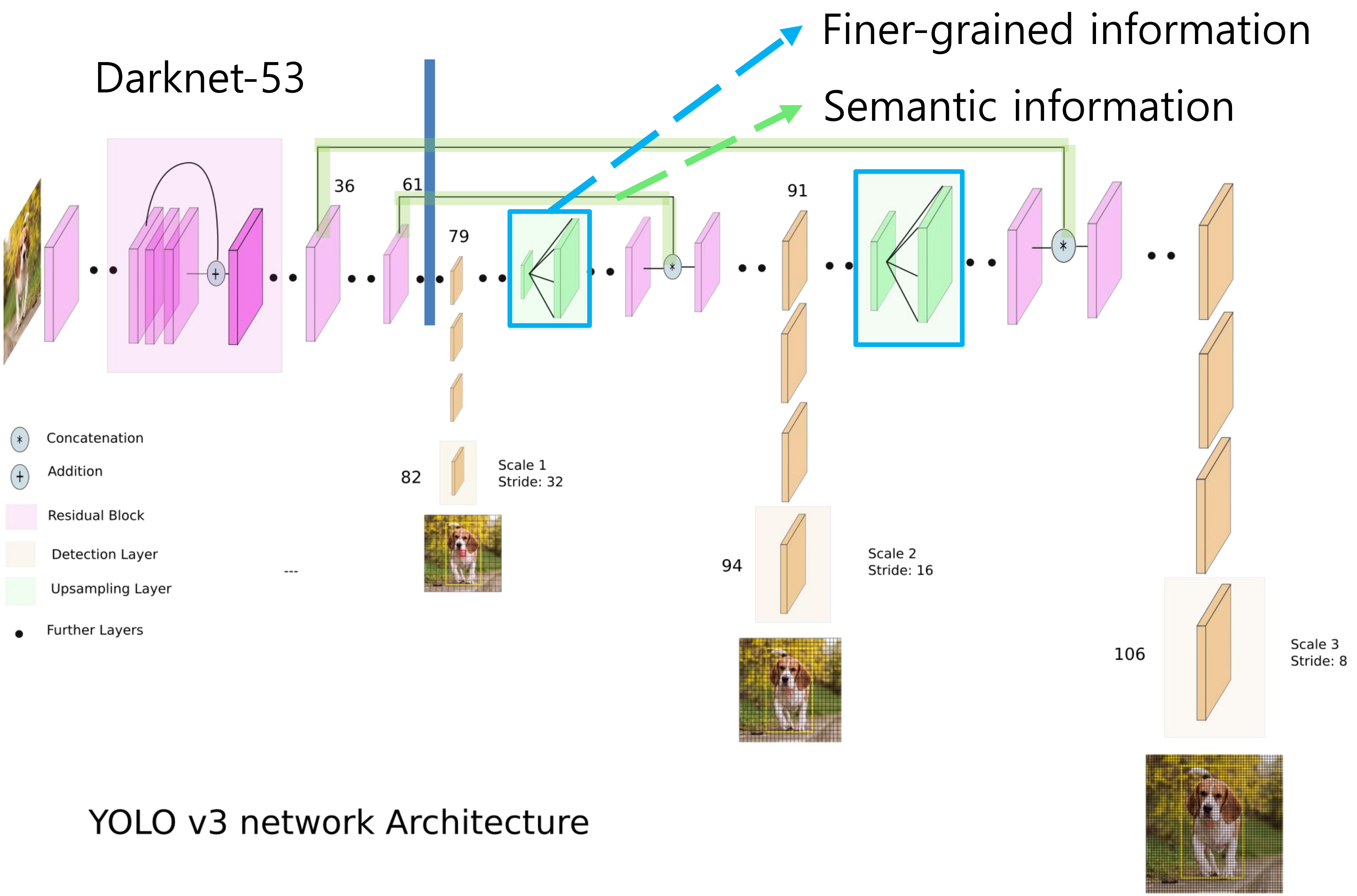
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3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

1초 연산량

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Better utilize GPU

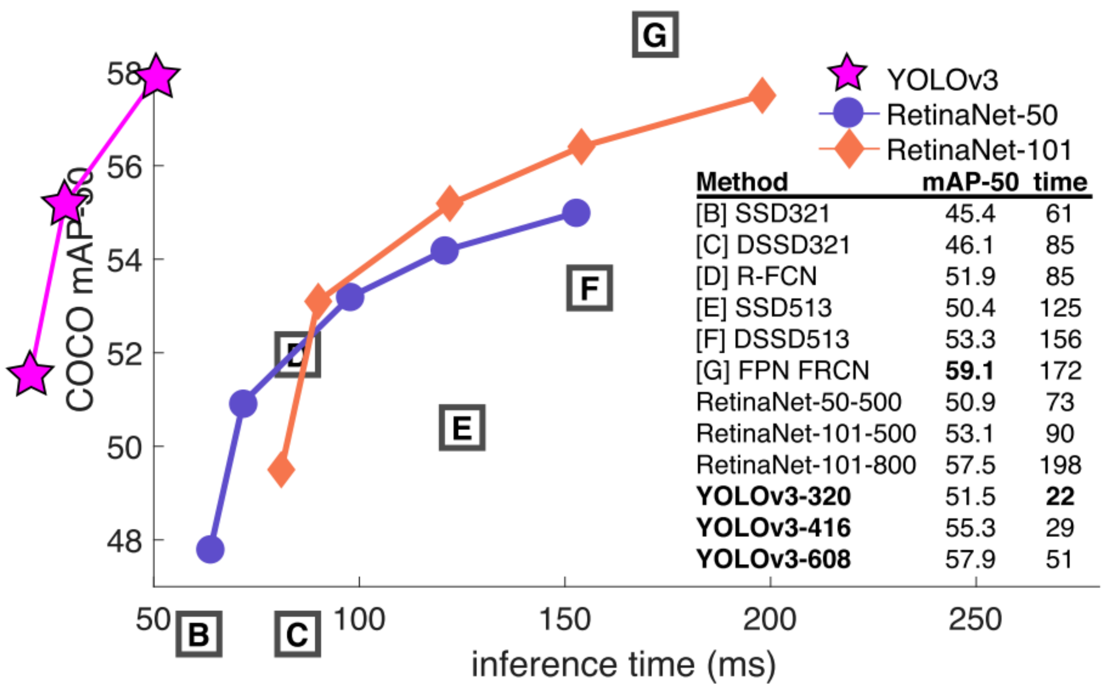
YOLO v3 Architecture



YOLO v3 network Architecture

Experiment

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 $\times$ 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9



04
YOLOv4
: Optimal Speed and Accuracy of Object Detection

Introduction

YOLO v3에 학계에서 발표된 성능을 올리는 다양한 기법들을 적용

YOLO v3 + Bag of Freebies

Bag of Specials

Selection of Architecture

Bag of Freebies (pre-processing + training strategy)

Change Training strategy or Only Increase the training costs

- **Data Augmentation**
  - Photometric distortion, geometric distortion
  - Simulating occlusions: random erase, CutOut, hide-and-seek, grid mask
  - To feature maps: DropOut, DropConnect, DropBlock
- **Biased data (imbalacne)**
  - Hard negative example mining, online hard example mining, focal loss
  - Label smoothing
- **Objective function**
  - MSE, IoU loss, GIoU loss, DIoU loss

Bag of Specials (plugin modules + post-processing)

Plugin modules and post-processing methods

Inference cost가 조금 올라가는 것에 비해 성능 향상

- **Enhancement of receptive field**
  - SPP, ASPP, RFB
- **Attention**
  - Squeeze-and-Excitation(SE), Spatial Attention Module (SAM)
- **Feature integration**
  - FPN, SFAM, ASFF, BiFPN
- **Activation functions**
  - LReLU, PReLU, ReLU6, SELU, Swish, hard-Swish, Mish
- **Post processing**
  - NMS, soft NMS, DIoU NMS

Selection of Architecture

Detector Requirement

Higher input network size (resolution) - small object 탐지 가능

More layers - Higher Receptive field

More parameters - 여러 size의 object를 탐지 가능

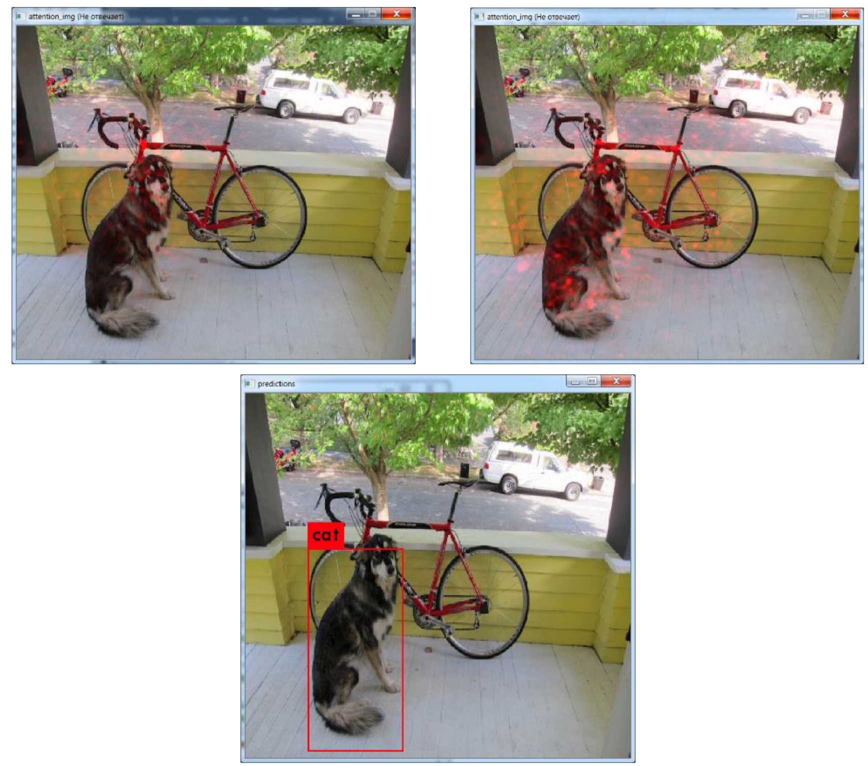
Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3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

Additional Improvements

New data augmentation method

Mosaic

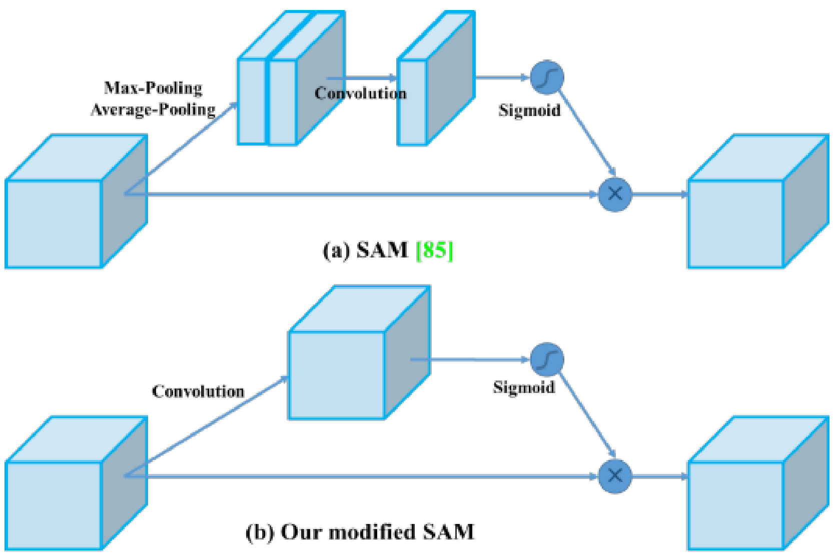
SAT (Self-Adversarial Training)



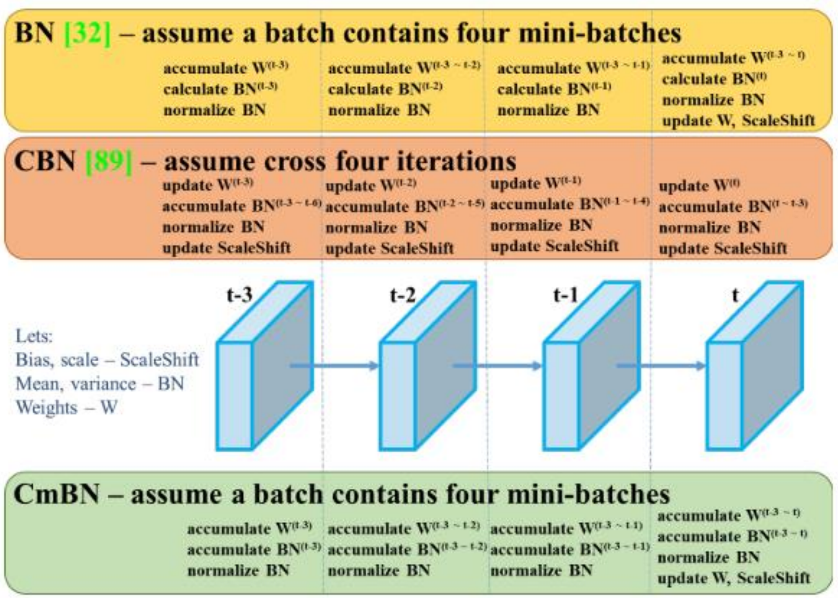
Additional Improvements

Modify existing methods

SAM



CmBN(Cross mini-batch Normalization)



YOLO v4

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

Backbone

- **Bag of Freebies** : CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing

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

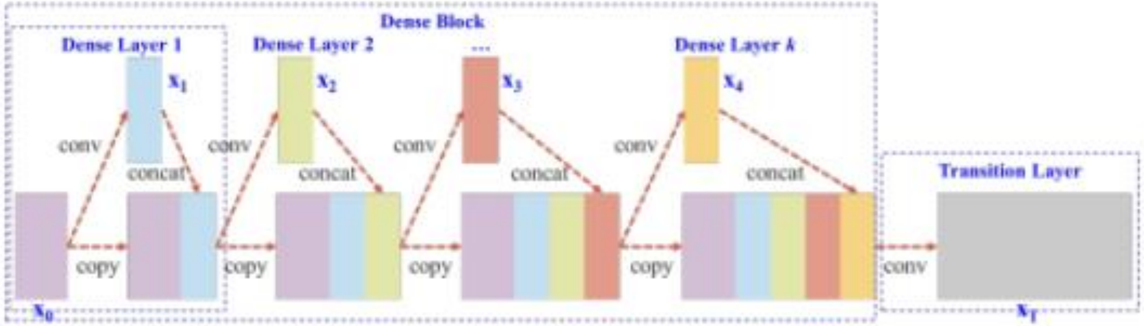
Detector

- **Bag of Freebies** : 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 hyper- parameters, Random training shapes

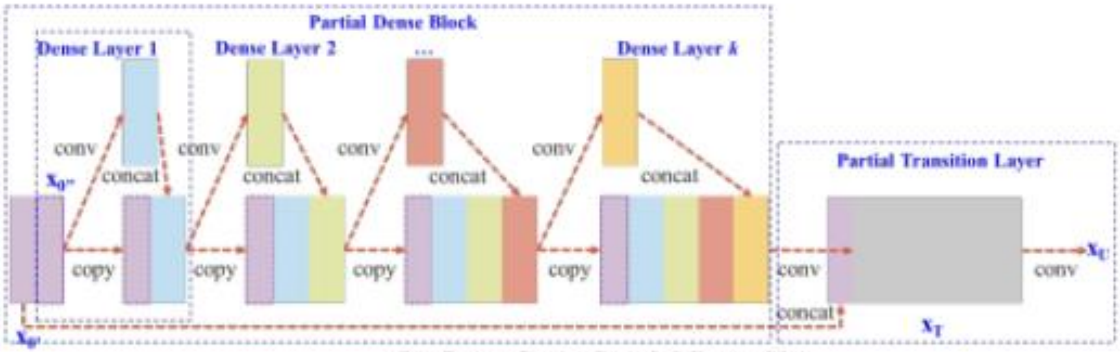
- **Bag of Specials** : Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

Network Architecture

Cross Stage Partial network (CSPNet)
첫 layer feature 절반을 나눈 후
하나는 dense block에 통과, 나머지는 transition layer에 concatenate
Computational bottleneck 제거, memory cost 절감, Inference 속도 향상



(a) DenseNet

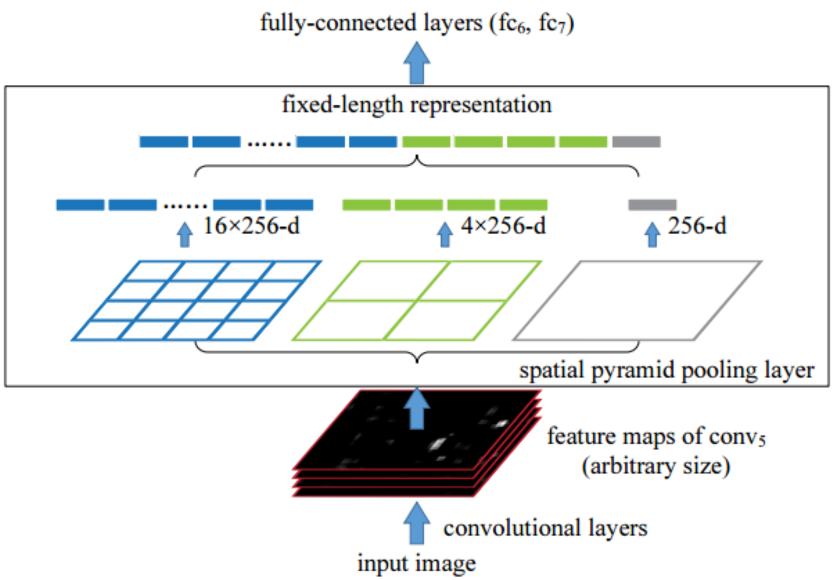


(b) Cross Stage Partial DenseNet

Network Architecture

SPP Block (Spatial Pyramid Pooling)

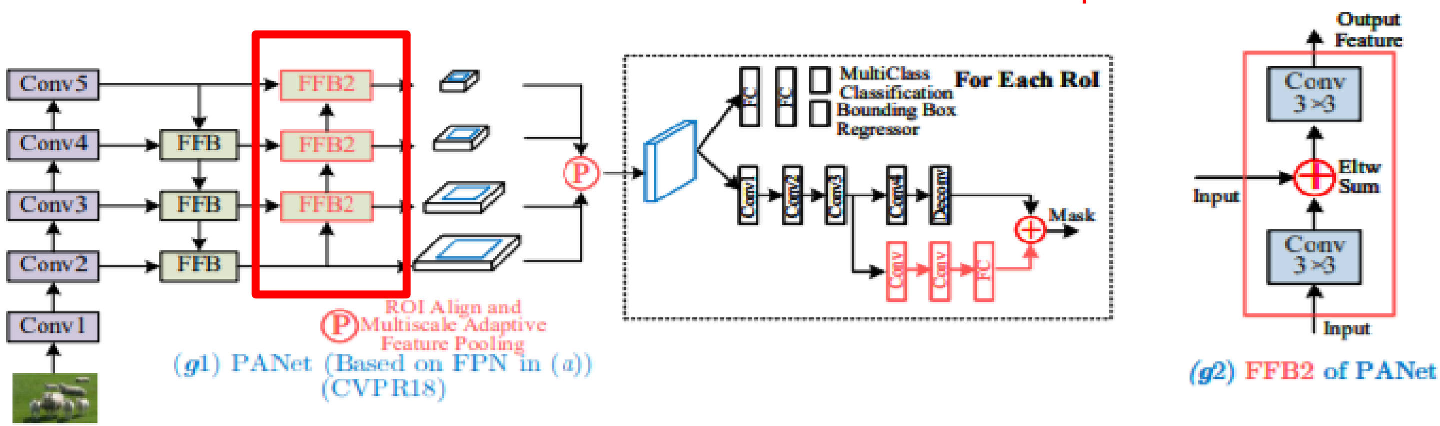
SPP Block to increase the receptive field



Network Architecture

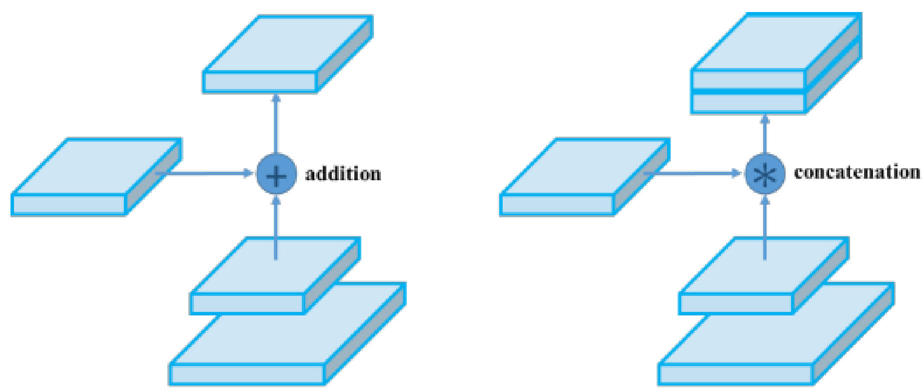
PANet(Path Aggregation Network)

기존 FPN 구조에서 정보 전달이 잘 되지 않는 점을 보안하기위해
추가적인 Short cut을 추가하여 shorten the information path



(g1) PANet (Based on FPN in (a)) (CVPR18)
(g2) FFB2 of PANet

Replace addition shortcut connection of PAN to concatenation



(a) PAN [49]
(a) Our modified PAN

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%**

CutMix

Mosaic

Label Smoothing

Mish

Influence of different features on Detector training

BoF

S: Eliminate grid sensitivity

M: Mosaic data augmentation

IT: IoU threshold

GA: Genetic algorithms

LS: Class label smoothing

CBN: CmBN

CA: Cosine annealing scheduler

DM: Dynamic mini-batch size

OA: Optimized Anchors

Loss : GIoU, CIoU, DIoU, MSE

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%**

Influence of different features on Detector training

BoS

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%

Other Influences on Detector training

Influence of different backbones and pretrained 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

Other Influences on Detector training

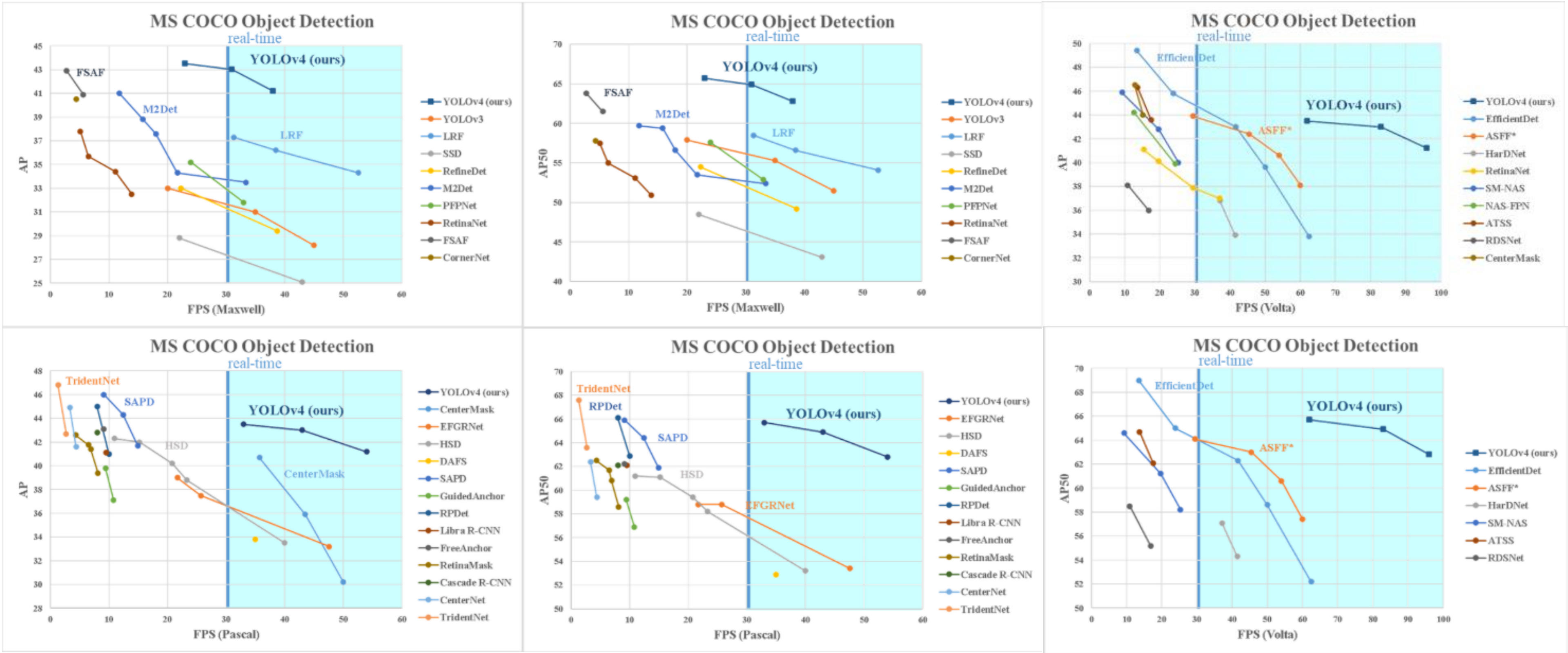
Influence of different mini-batch size on Detector training

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

Results

다른 state-of-the-art object detectors 와 비교했을 때 성능이 좋음을 보임



Conclusions

Offer a state-of-the-art detector which is faster (FPS) and more accurate

YOLO v4 can be Trained and used on a conventional GPU with 8-16 GB-VRAM

this makes its broad use possible

verified a large number of features, and selected for use such of them

for improving the accuracy of both the classifier and the detector