

Object Detection – 두개 이상의 Object 검출

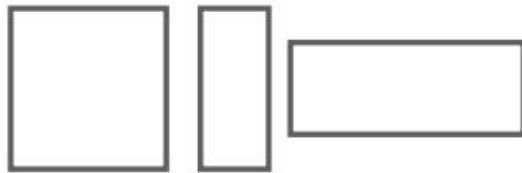
이미지의 어느 위치에서 Object를 찾아야 하는가?



Sliding Window 방식

Window를 왼쪽 상단에서 부터 오른쪽 하단으로 이동시키면서 Object를 Detection하는 방식

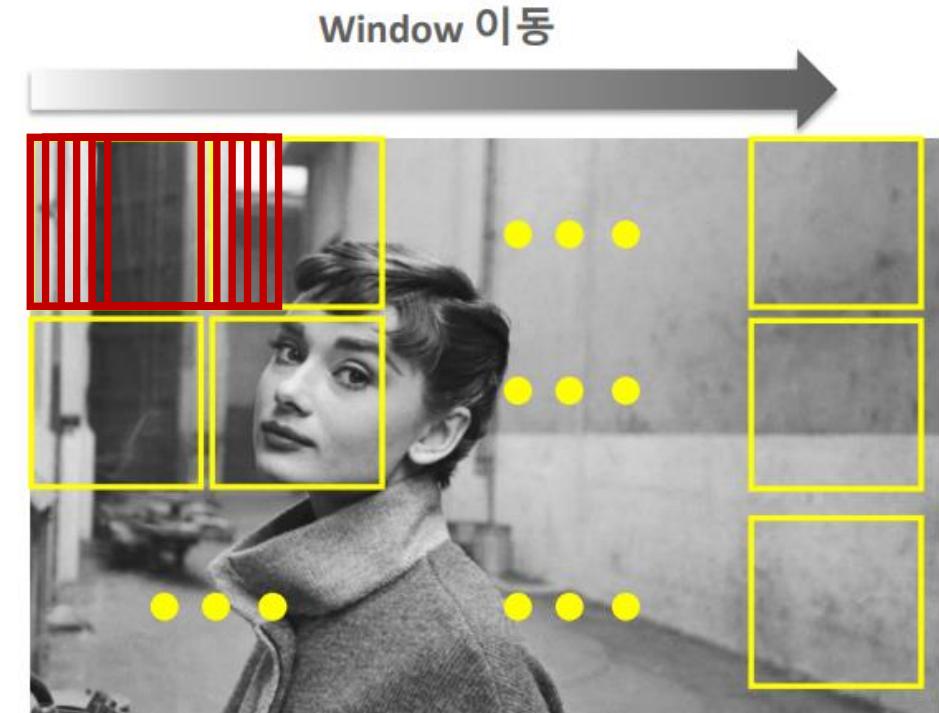
다양한 형태의 Window를 각각 sliding 시키는 방식



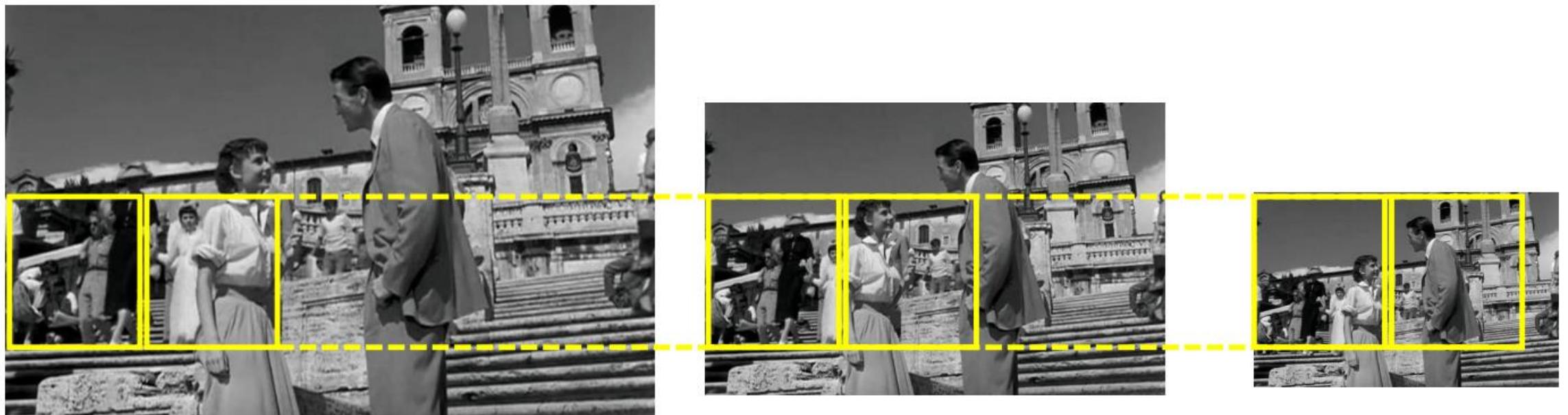
Window Scale은 고정하고 scale을 변경한 여러 이미지를 사용하는 방식



- Object Detection의 초기 기법으로 활용
- 오브젝트 없는 영역도 무조건 슬라이딩 하여야 하며 여러 형태의 Window와 여러 Scale을 가진 이미지를 스캔해서 검출해야 하므로 수행 시간이 오래 걸리고 검출 성능이 상대적으로 낮음
- Region Proposal(영역 추정) 기법의 등장으로 활용도는 떨어졌지만 Object Detection 발전을 위한 기술적 토대 제공

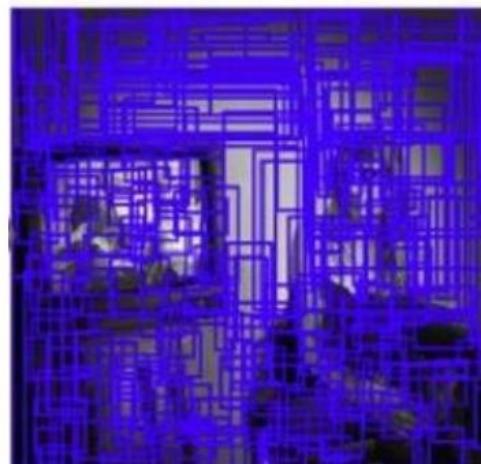
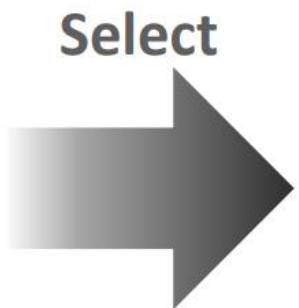


Sliding Window – Image Scale 조정으로 Object Detection

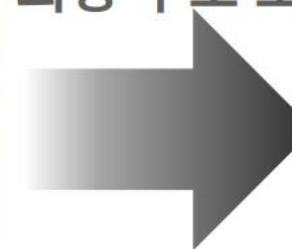


Region Proposal(영역 추정)

"Object가 있을 만한 후보 영역을 찾자"



최종 후보 도출



원본 이미지

후보 Bounding Box 선택

최종 Object Detection

Region Proposal(영역 추정) – Selective Search

빠른 Detection과 높은 Recall 예측 성능을 동시에 만족하는 알고리즘

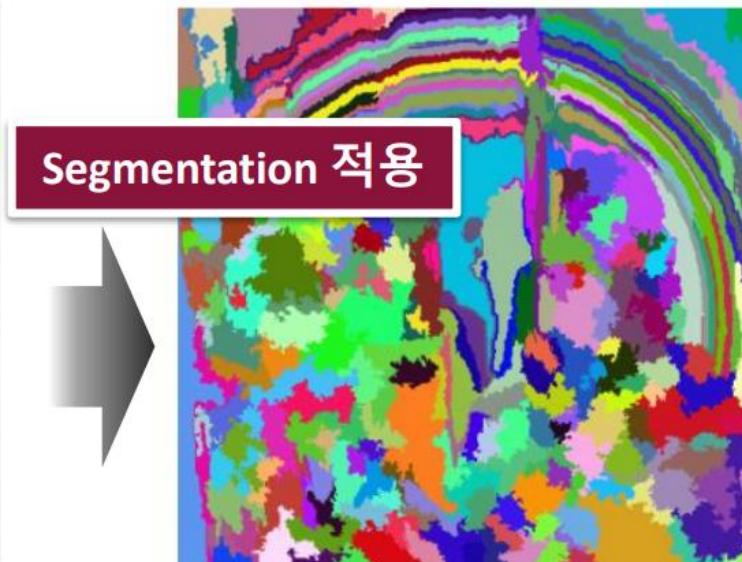
컬러, 무늬(Texture), 크기(Size), 형태(Shape)에 따라 유사한 Region을 계층적 그룹핑 방법으로 계산

Selective Search는 최초에는 Pixel Intensity기반한 graph-based segment 기법에 따라 Over Segmentation을 수행

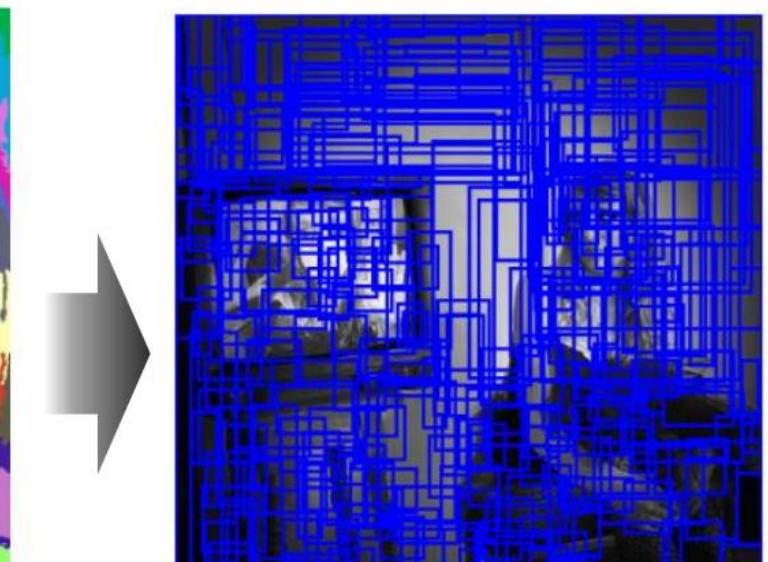
(각각의 object들이 1개의 개별 영역에 담길 수 있도록 많은 초기 영역을 생성 by Felzenszwalb and Huttenlocher 2004)



원본 이미지



최초 Segmentation
(Over Segmentation)



후보 Objects

Region Proposal(영역 추정) – Selective Search

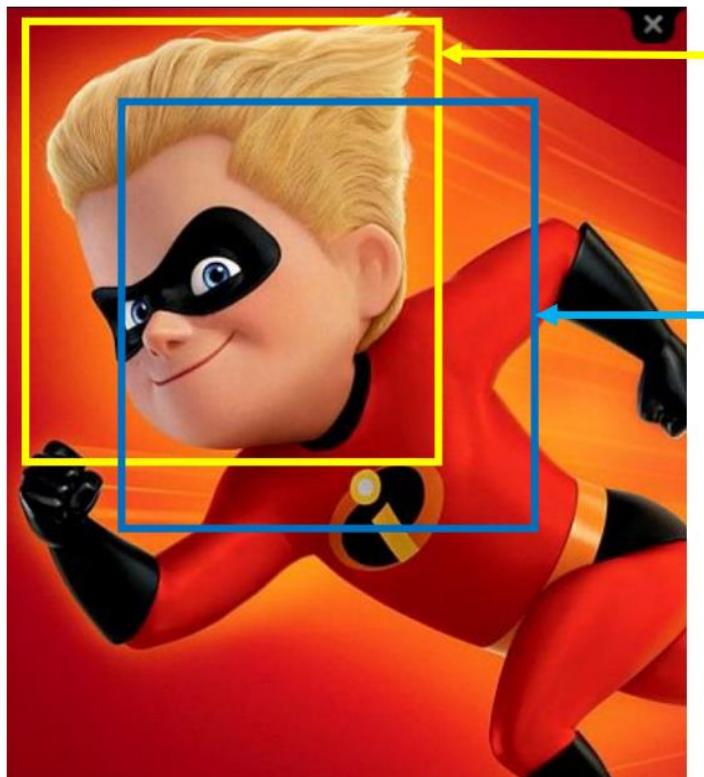
1. 개별 Segment된 모든 부분들을 Bounding box로 만들어서 Region Proposal 리스트로 추가
2. 컬러, 무늬(Texture), 크기(Size), 형태(Shape)에 따라 유사도가 비슷한 Segment들을 그룹핑함.
3. 다시 1번 Step Region Proposal 리스트 추가, 2번 Step 유사도가 비슷한 Segment들 그룹핑을 계속 반복하면서 Region Proposal을 수행



Object Detection 성능평가 - IoU

IoU: Intersection over Union

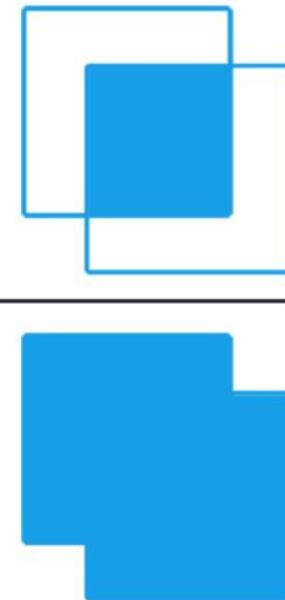
모델이 예측한 결과와 실측(Ground Truth) Box가 얼마나 정확하게 겹치는가를 나타내는 지표



실측(Ground Truth)
Bounding box

예측(Predicted)
Bounding box

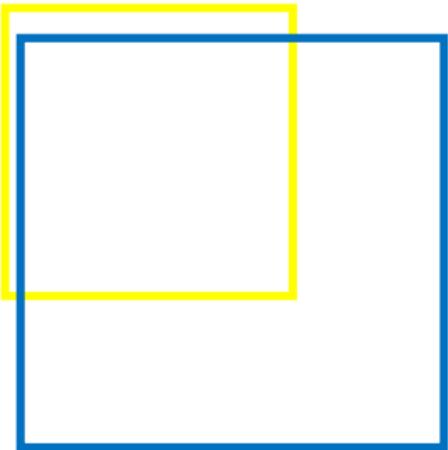
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



IoU는 개별 Box가 서로 겹치는 영역/ 전체 Box의 합집합 영역

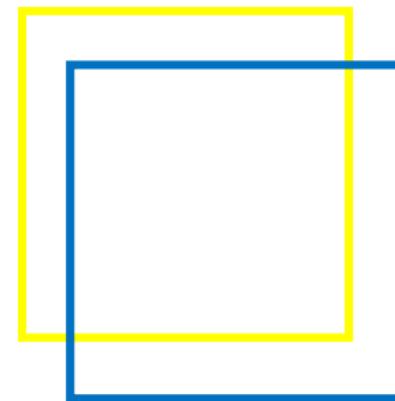
Object Detection 성능평가 - IoU

IoU: 0.403



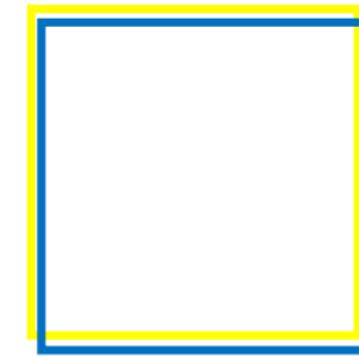
Poor

IoU: 0.740



Good

IoU: 0.936



Excellent

NMS(Non Max Suppression)

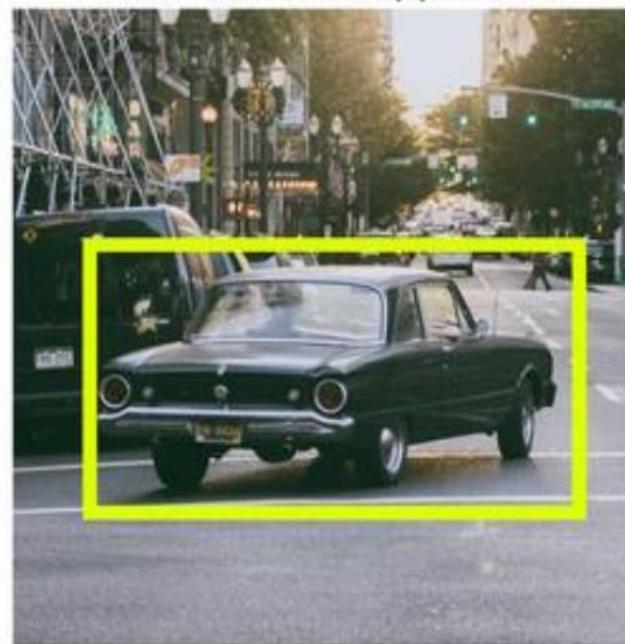
Before non-max suppression



Non-Max
Suppression



After non-max suppression

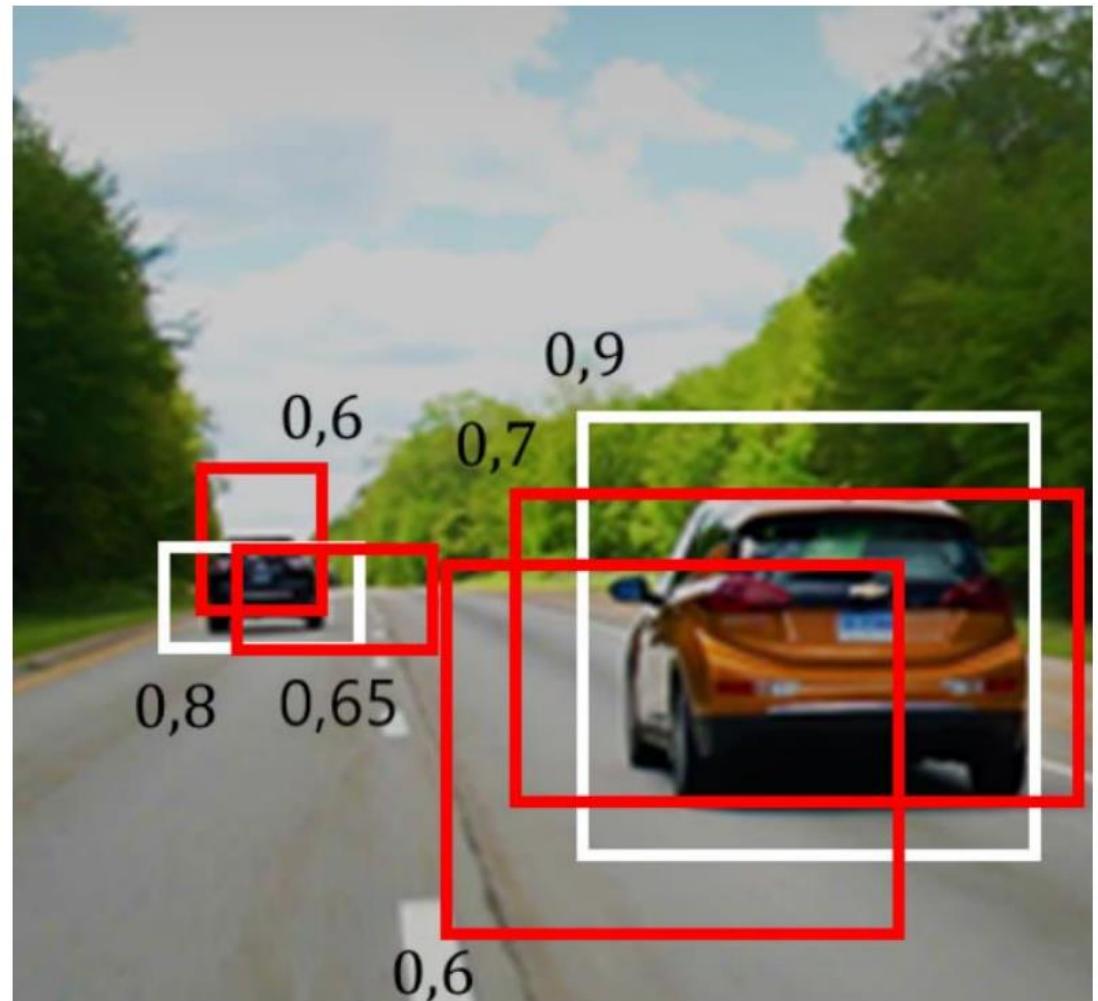


- Object Detection 알고리즘은 Object 가 있을 만한 위치에 많은 Detection을 수행하는 경향이 강함.
- NMS는 Detected 된 Object의 Bounding box중에 비슷한 위치에 있는 box를 제거하고 가장 적합한 box를 선택하는 기법

NMS(Non Max Suppression)

1. Detected 된 bounding box별로 특정 Confidence threshold 이하 bounding box는 먼저 제거(confidence score < 0.5)
2. 가장 높은 confidence score를 가진 box 순으로 내림차순 정렬하고 아래 로직을 모든 box에 순차적으로 적용.
 - 높은 confidence score를 가진 box와 겹치는 다른 box를 모두 조사하여 IOU가 특정 threshold 이상인 box를 모두 제거(예: IOU Threshold > 0.4)
3. 남아 있는 box만 선택

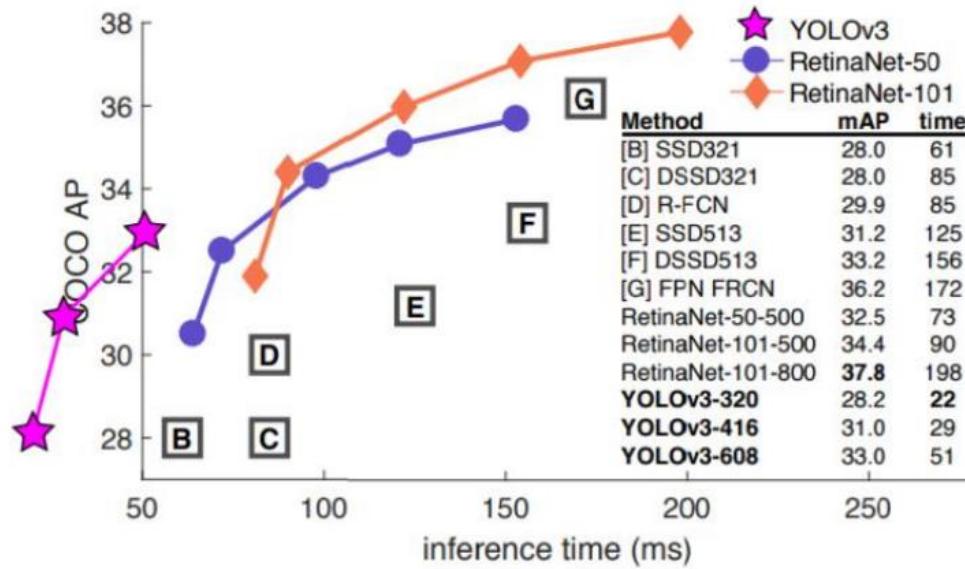
Confidence score가 **높을 수록**,
IOU Threshold가 **낮을 수록** 많은 Box가 제거됨.



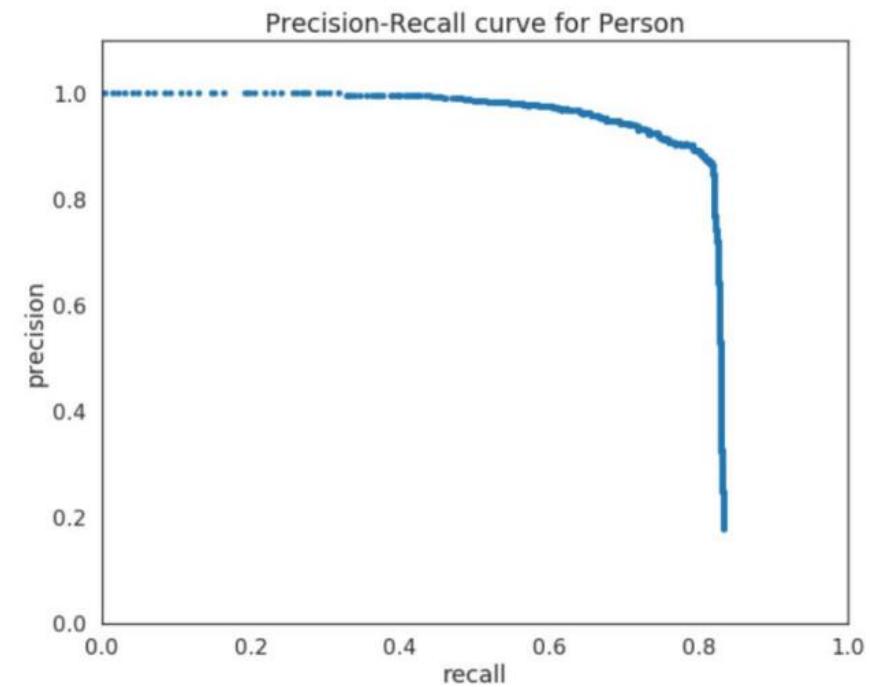
Object Detection 성능평가 - mAP

실제 Object가 Detected된 재현율(Recall)의 변화에 따른 정밀도(Precision)의 값을 평균한 성능 수치

mAP
(mean Average Precision)



- IOU
- Precision-Recall Curve, Average Precision
- Confidence threshold



Object Detection 성능평가 – Precision(정밀도) vs Recall(재현율)

정밀도(Precision)과 재현율(Recall)은 주로 이진 분류(Binary Classification)에서 사용되는 성능 지표입니다.

- 정밀도(Precision)는 예측을 Positive로 한 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율을 뜻합니다. Object Detection에서는 검출 알고리즘이 검출 예측한 결과가 실제 Object들과 얼마나 일치하는지를 나타내는 지표입니다.
- 재현율(Recall)은 실제 값이 Positive인 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율을 뜻합니다. Object Detection에서는 검출 알고리즘이 실제 Object들을 빠뜨리지 않고 얼마나 정확히 검출 예측하는지를 나타내는 지표입니다.

검출 예측을 정확히 Bird로 함
Precision=100%



무슨 소리?
오른쪽의 새는 아예 예측하지 못함.
Recall = 50%

Object Detection 성능평가 – IoU값에 따른 성공 결정

Object Detection에서 개별 Object에 대한 검출(Detection) 예측이 성공하였는지의 여부를 IoU로 결정.

일반적으로 PASCAL VOC Challenge에서 사용된 기준을 적용하여 IoU가 0.5 이상이면 예측 성공으로 인정.

(그러나 COCO challenge에서는 여러 개의 IoC 기준을 변경해 가면서 예측 성공을 적용)



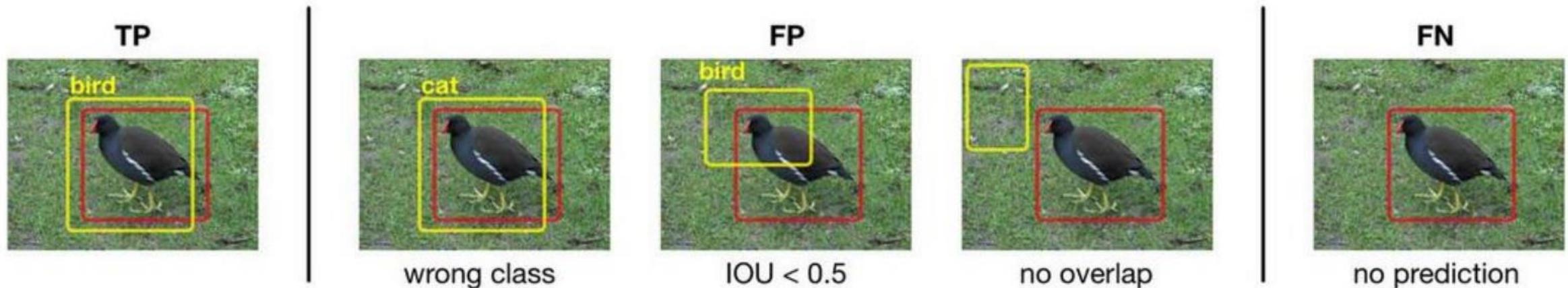
오차 행렬(Confusion Matrix)

오차 행렬은 이진 분류의 예측 오류가 얼마인지와 더불어 어떠한 유형의 예측 오류가 발생하고 있는지를 함께 나타내는 지표입니다

		예측 클래스(Predicted Class)	
		Negative(0)	Positive (1)
		검출되지 말아야 할 것이 검출되지 않음	틀린결정
실제 클래스 (Actual Class)	Negative(0)	Negative Negative (True Negative)	Negative Positive (False Positive)
	Positive(1)	Positive Negative (False Negative)	Positive Positive (True Positive)
		TN	FP
		FN	TP
		검출 되어야 할 것이 검출되지 않음	옳은결정

Object Detection 성능평가 – Precision(정밀도) vs Recall(재현율)

Object Detection에서 TP, FP, FN



- 정밀도 = $TP / (FP + TP)$
- 재현율 = $TP / (FN + TP)$

		예측 클래스(Predicted Class)	
		Negative(0)	Positive (1)
실제 클래스 (Actual Class)	검출되지 말아야 할 것이 검출되지 않음 Negative(0)	TN (True Negative)	틀린결정 FP (False Positive)
	검출 되어야 할 것이 검출되지 않음 Positive(1)	FN (False Negative)	TP (True Positive)

Object Detection 성능평가 – Precision(정밀도) vs Recall(재현율)

- 재현율이 상대적으로 더 중요한 지표인 경우는 실제 Positive 양성인 데이터 예측을 Negative로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우 : 암 진단, 금융사기 판별
- 정밀도가 상대적으로 더 중요한 지표인 경우는 실제 Negative 음성인 데이터 예측을 Positive 양성으로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우: 스팸 메일

Object Detection 성능평가 – Precision(정밀도) vs Recall(재현율)의 맹점

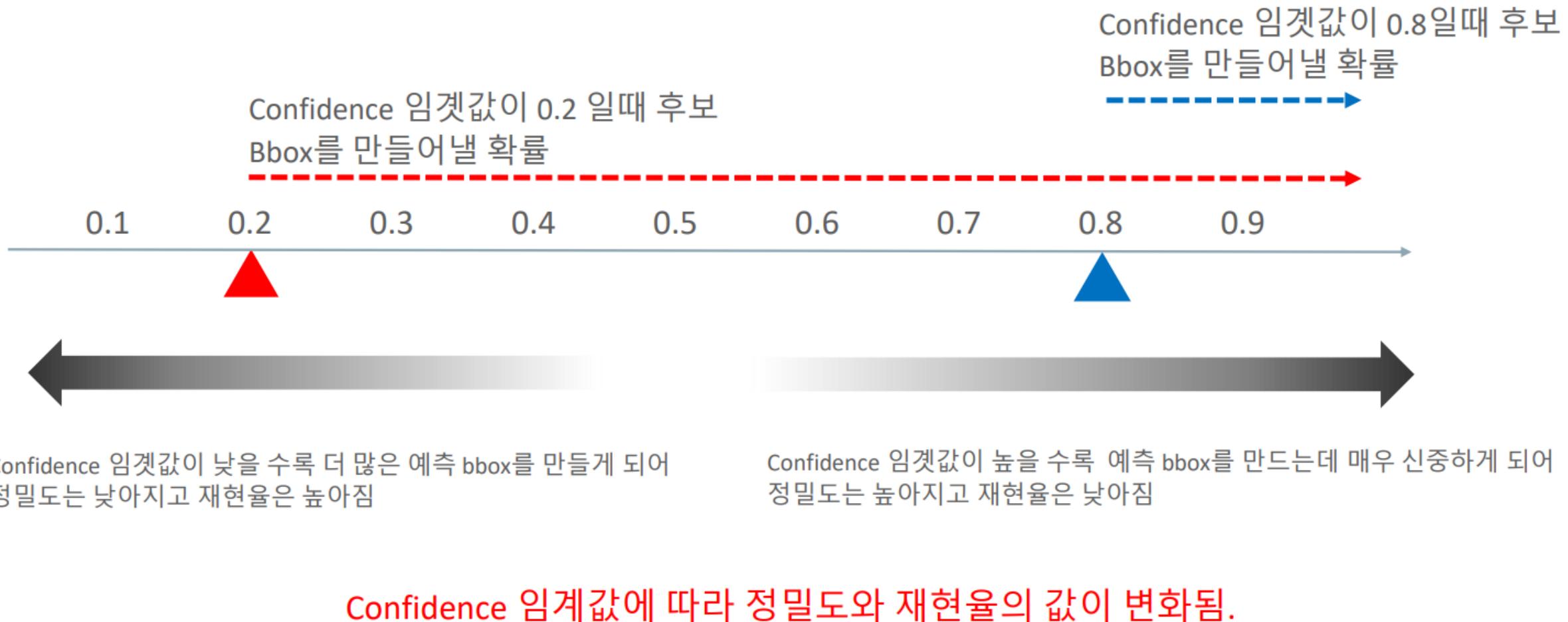
정밀도를 100% 만드는 법

- 확실한 기준이 되는 경우만 Positive로 예측하고 나머지는 모두 Negative로 예측합니다. 정밀도 = $TP / (TP + FP)$ 입니다. 전체 환자 1000명 중 확실한 Positive 징후만 가진 환자는 단 1명이라고 하면 이 한 명만 Positive로 예측하고 나머지는 모두 Negative로 예측하더라도 FP는 0, TP는 1이 되므로 정밀도는 $1/(1+0)$ 으로 100%가 됩니다

재현율을 100% 만드는 법

- 모든 환자를 Positive로 예측하면 됩니다. 재현율 = $TP / (TP + FN)$ 이므로 전체 환자 1000명을 다 Positive로 예측하는 겁니다. 이 중 실제 양성인 사람이 30명 정도라도 TN이 수치에 포함되지 않고 FN은 아예 0이므로 $30/(30 + 0)$ 으로 100%가 됩니다.

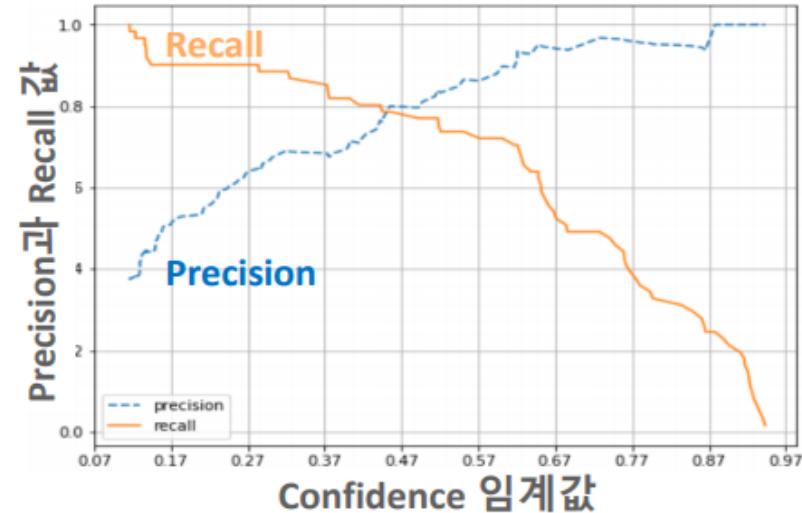
Confidence 임계값에 따른 정밀도 – 재현율 변화



Confidence 임계값에 따른 정밀도 – 재현율 변화

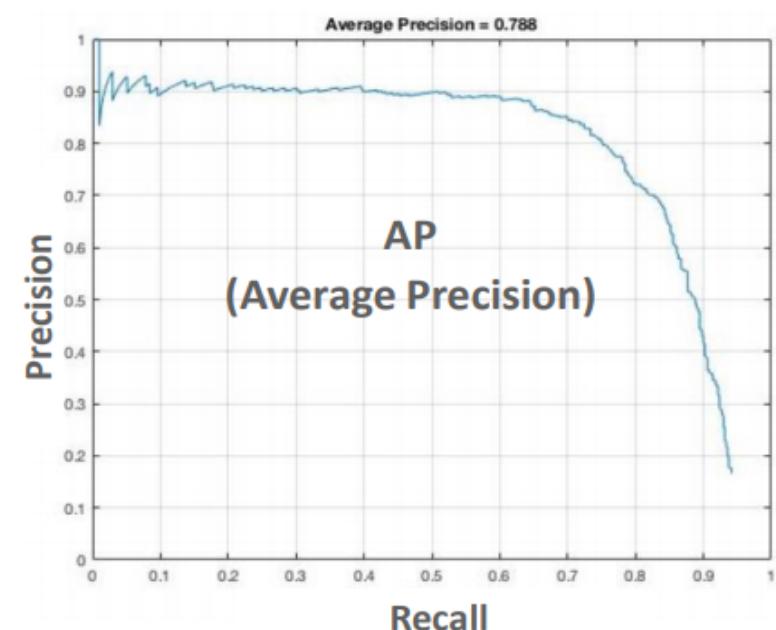
정밀도 재현율 트레이드 오프 (Precision Recall Trade-off)

- Confidence 임계값(Threshold)을 조정하면 정밀도 또는 재현율의 수치를 높일 수 있습니다. 하지만 정밀도와 재현율은 상호 보완적인 평가 지표이기 때문에 어느 한쪽을 강제로 높이면 다른 하나의 수치는 떨어지기 쉽습니다. 이를 정밀도/재현율의 트레이드오프(Trade-off)라고 부릅니다.



정밀도 재현율 곡선 (Precision-Recall Curve)

- Recall 값의 변화에 따른(Confidence 값을 조정하면서 얻어진) Precision 값을 나타낸 곡선을 정밀도 재현율 곡선이라고 합니다. 그리고 이렇게 얻어진 Precision 값의 평균을 AP라고 하며, 일반적으로 정밀도 재현율 곡선의 면적 값으로 계산됩니다.



Confidence에 따른 정밀도 – 재현율 변화

Confidence: 0. 9

confidence	정확?	Precision	Recall
0.9	True	1.0	0.2

Confidence: 0. 8

confidence	정확?	Precision	Recall
0.9	True	1.0	0.2
0.8	True	1.0	0.4

Confidence : 0. 7

confidence	정확?	Precision	Recall
0.9	True	1.0	0.2
0.8	True	1.0	0.4
0.7	False	0.67	0.4

Confidence: 0. 6

confidence	정확?	Precision	Recall
0.9	True	1.0	0.2
0.8	True	1.0	0.4
0.7	False	0.67	0.4
0.6	False	0.5	0.4

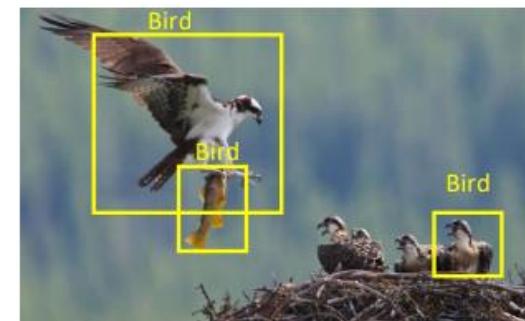
Precision = 1/1 , Recall = 1/5



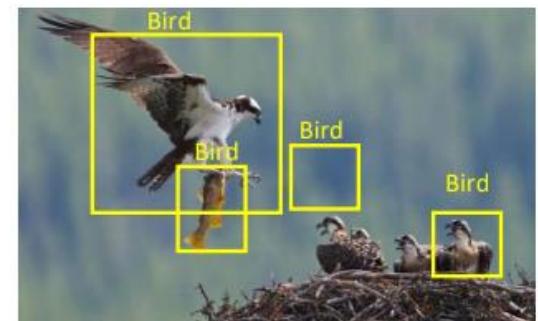
Precision = 2/2 , Recall = 2/5



Precision = 2/3 , Recall = 2/5



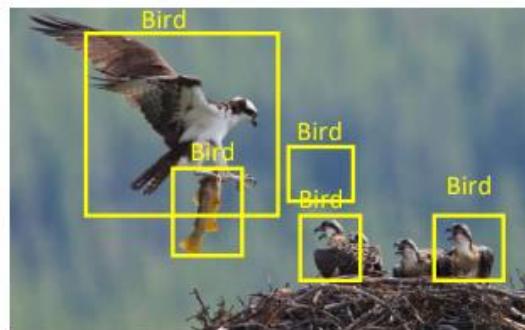
Precision = 2/4 , Recall = 2/5



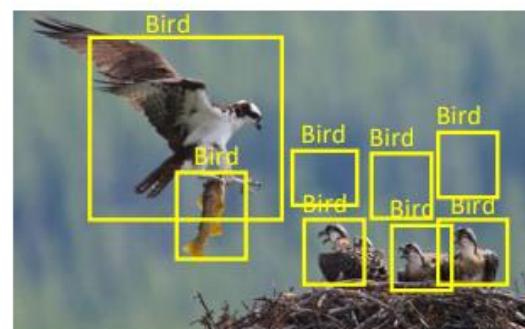
Confidence에 따른 정밀도 – 재현율 변화

confidence	정확?	Precision	Recall
0.9	True	1.0	0.2
0.8	True	1.0	0.4
0.7	False	0.67	0.4
0.6	False	0.5	0.4
0.5	True	0.6	0.6
0.4	False	0.5	0.6
0.3	True	0.57	0.8
0.2	False	0.5	0.8
0.1	False	0.44	0.8
0.05	True	0.5	1.0

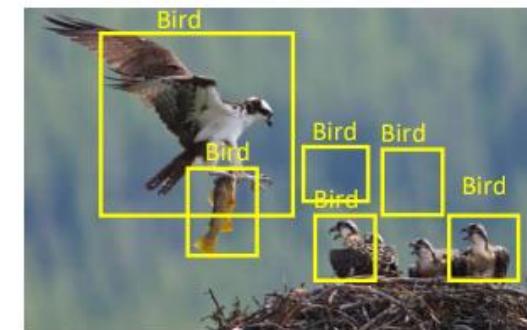
Confidence: 0.5
Precision = 3/5, Recall = 3/5



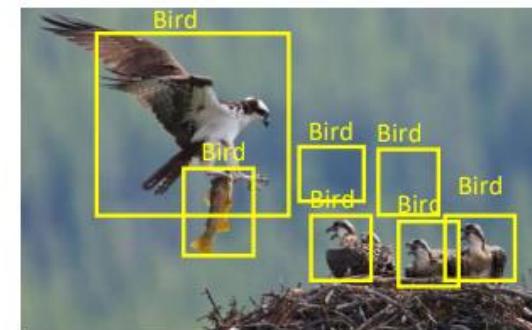
Confidence: 0. 2
Precision = 4/8, Recall = 4/5



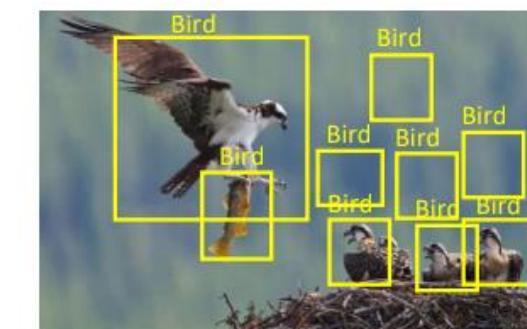
Confidence: 0. 4
Precision = 3/6, Recall = 3/5



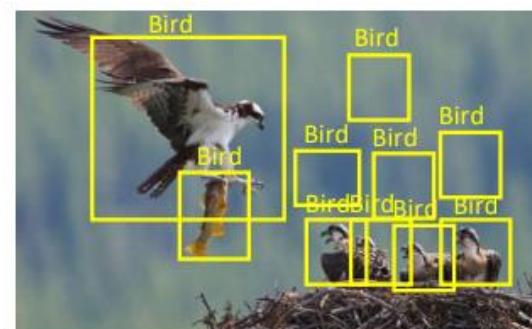
Confidence: 0. 3
Precision = 4/7, Recall = 4/5



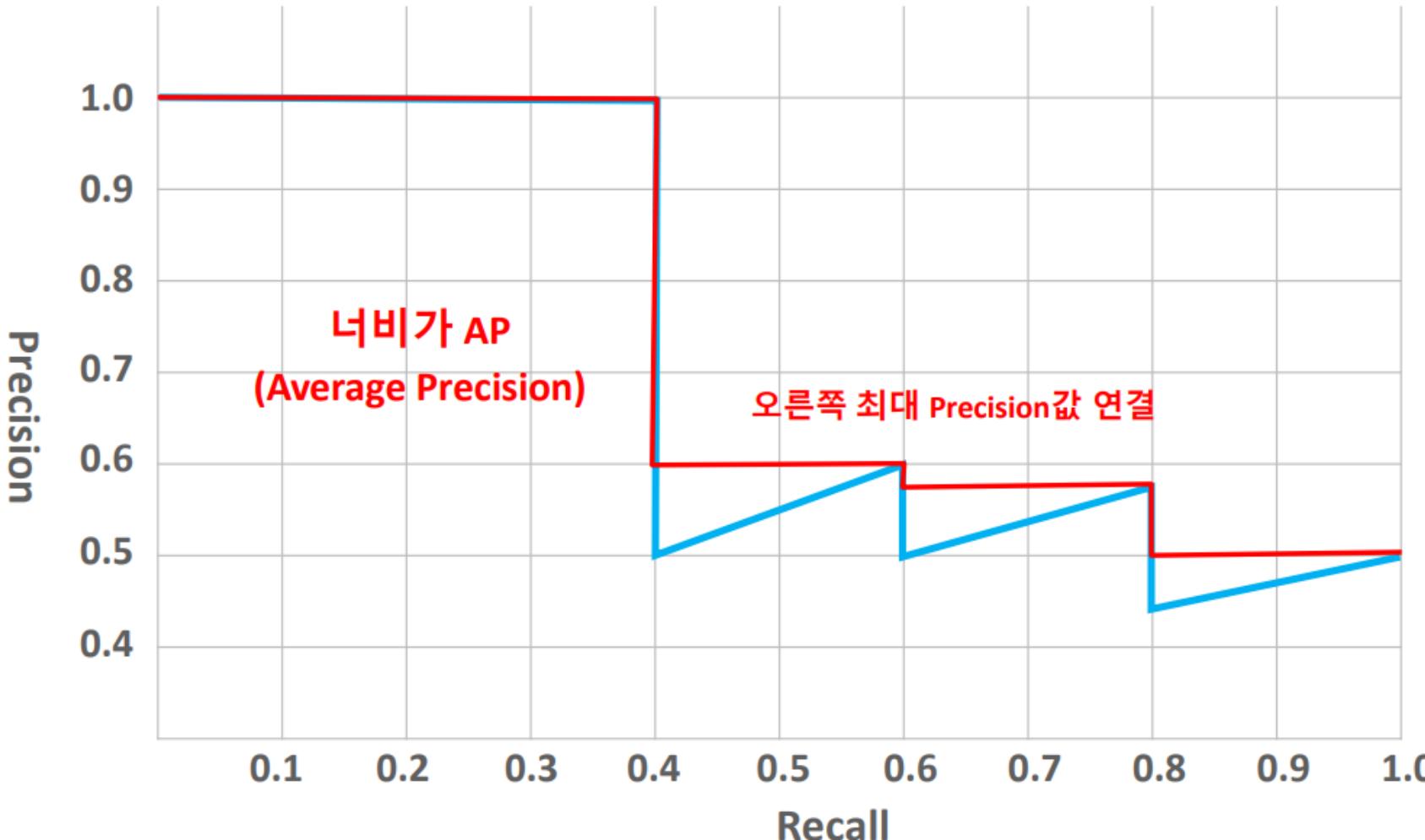
Confidence: 0. 1
Precision = 4/9, Recall = 4/5



Confidence: 0. 05
Precision = 5/10, Recall = 5/5

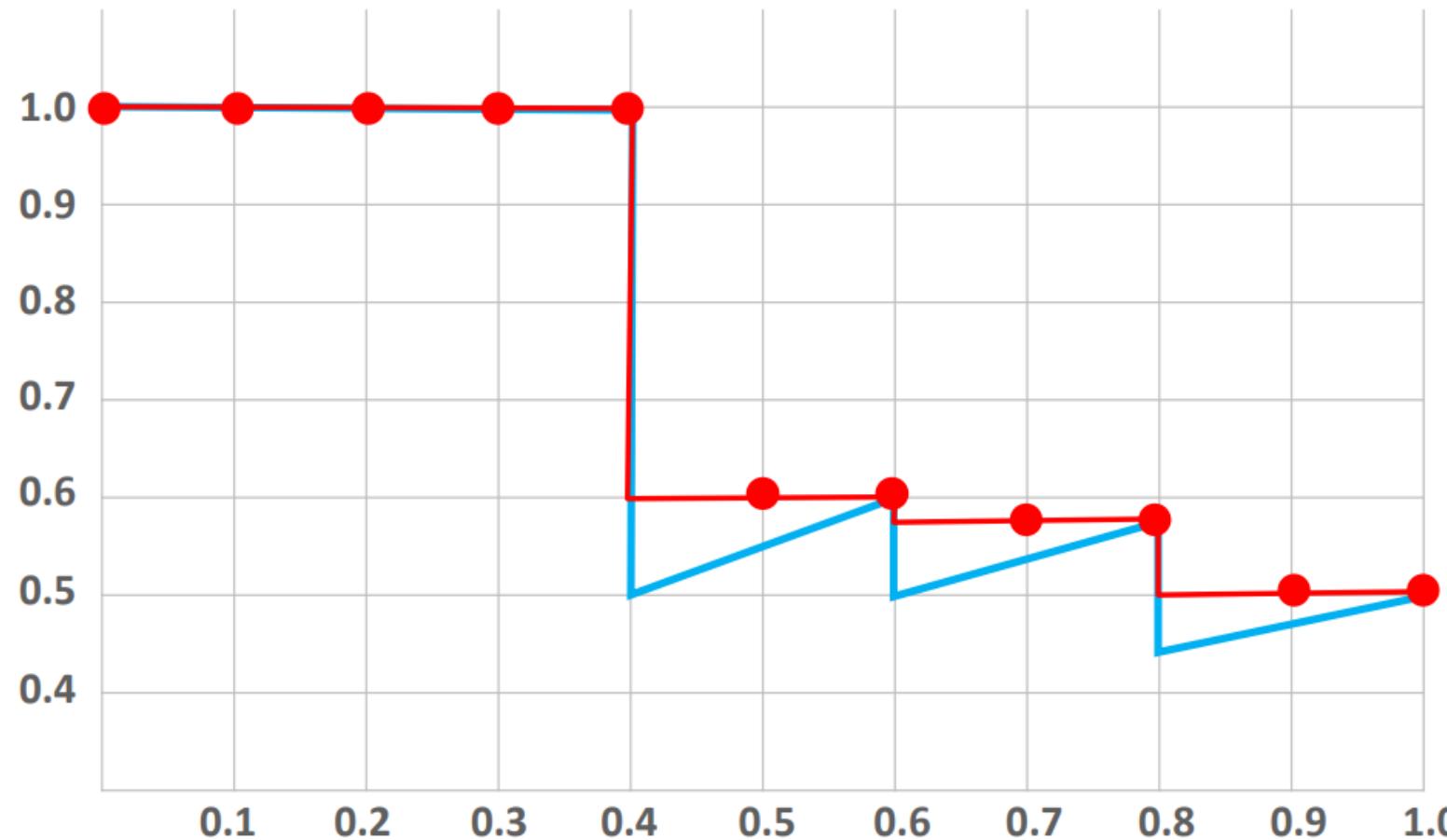


Confidence에 따른 정밀도 – AP(Average precision) 계산



confidence	정확?	Precision	Recall
0.9	True	1.0	0.2
0.8	True	1.0	0.4
0.7	False	0.67	0.4
0.6	False	0.5	0.4
0.5	True	0.6	0.6
0.4	False	0.5	0.6
0.3	True	0.57	0.8
0.2	False	0.5	0.8
0.1	False	0.44	0.8
0.05	True	0.5	1.0

Confidence에 따른 정밀도 – AP(Average precision) 계산



개별 11개(0.0 ~ 1.0 까지) Recall 포인트
별로 최대 Precision의 평균 값을 구함

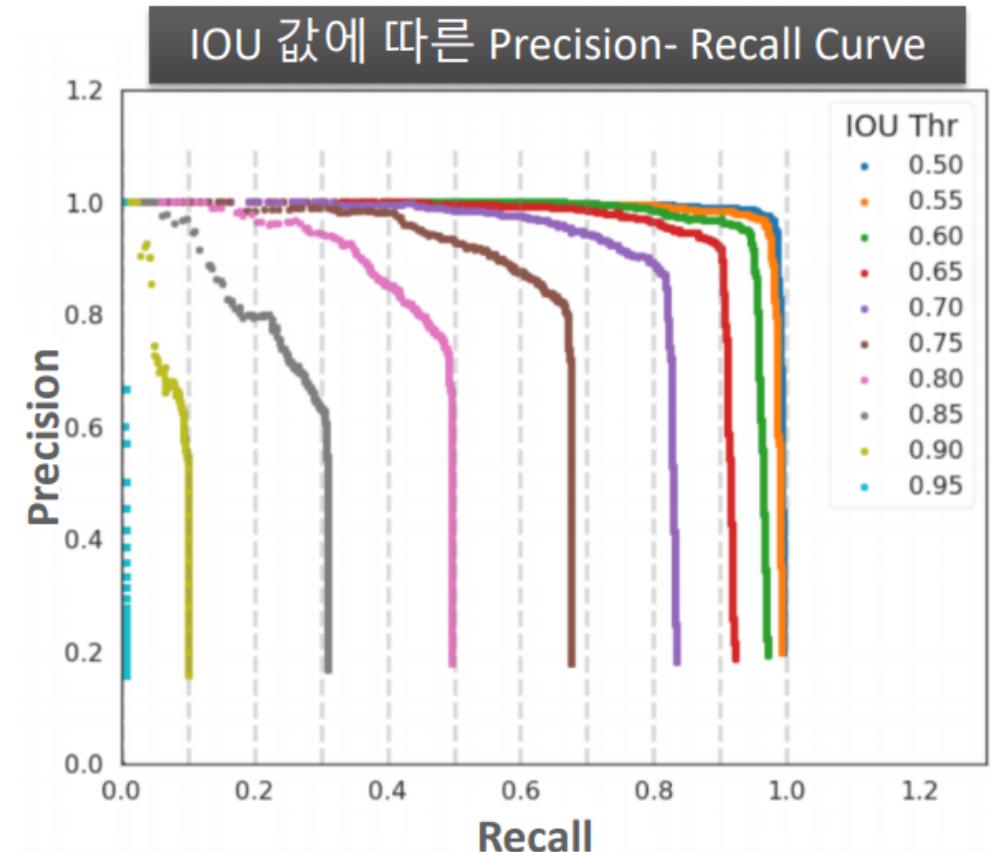
$$\begin{aligned} AP &= \frac{1}{11} \times (mP(r=0) + mP(r=0.1) + \dots + mP(r=1)) \\ &= \frac{1}{11} \times (1.0 + 1.0 + 1.0 + 1.0 + 1.0 + \\ &\quad 0.6 + 0.6 + 0.57 + 0.57 + 0.5 + 0.5) \\ &= \frac{1}{11} \times (5 \times 1.0 + 0.6 \times 2 + 0.57 \times 2 + 0.5 \times 2) \\ &= 0.758 \end{aligned}$$

Confidence에 따른 정밀도 – mAP(mean Average precision) 계산

- AP는 한 개 오브젝트에 대한 성능 수치
- mAp(mean Average Precision) 은 여러 오브젝트들의 AP를 평균한 값

COCO Challenge – mAP(mean Average precision) 계산

- 예측 성공을 위한 IOU를 0.5 이상으로 고정한 PASCAL VOC와 달리 coco Challenge는 IOU 를 다양한 범위로 설정하여 예측 성공 기준을 정함.
 - IOU 0.5 부터 0.05 씩 값을 증가 시켜 0.95까지 해당하는 IOU별로 mAP를 계산(AP@[.50:.05:.95] 는 시작 IOU기준: 0.5, 증가 Step: 0.05, 최종 IOU: 0.95 에 따른 AP 값을 의미)
 - 또한 크기의 유형(대/중/소)에 따른 mAP도 측정



COCO Challenge – mAP(mean Average precision) 계산

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

VOC 2007 YOLO V2

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

COCO YOLO V2

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

COCO YOLO V3

YOLO

(You Only Look Once)



yolo homepage



전체

이미지

뉴스

동영상

지도

더보기

설정

도구

검색결과 약 5,400,000개 (0.42초)

[pjreddie.com](#) > yolo ▾

YOLO: Real-Time Object Detection - Joseph Redmon

You only look once (**YOLO**) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% ...

Darknet

[YOLO - Installing Darknet -](#)
[Nightmare - Coq Tactic -](#) ...

YOLO: Real-Time Object ...

You only look once (**YOLO**) is a
state-of-the-art, real-time object ...

[pjreddie.com](#) 검색결과 더보기 »

Publications

My publications.

Pascal VOC Dataset Mirror

Here is a mirror for the Pascal
VOC files in case, you know, you ...

home darknet



coq tactics



publications

Projects

résumé

YOP

YOLO: Real-Time Object Detection

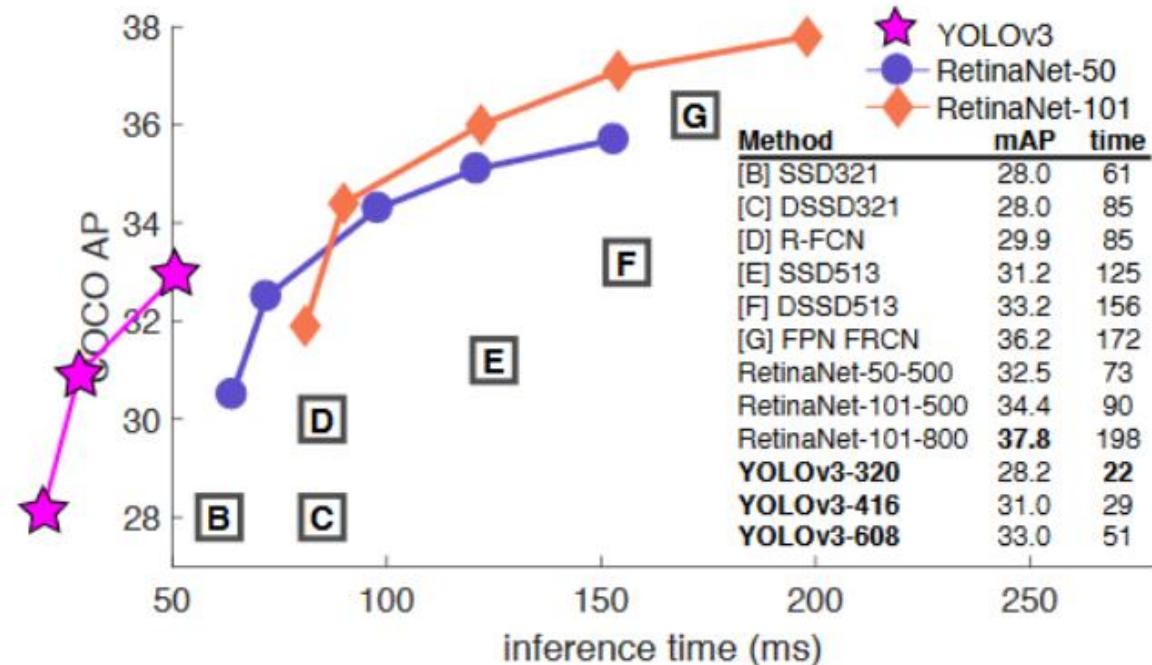
You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev.



Yolo v3 성능

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320×320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP₅₀ in 51 ms on a Titan X, compared to 57.5 AP₅₀ in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at <https://pjreddie.com/yolo/>.



Speed + Accuracy

Yolo Version

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

V1

빠른 Detection 시간
그러나 낮은 정확도

V2

수행시간과 성능 모두 개선

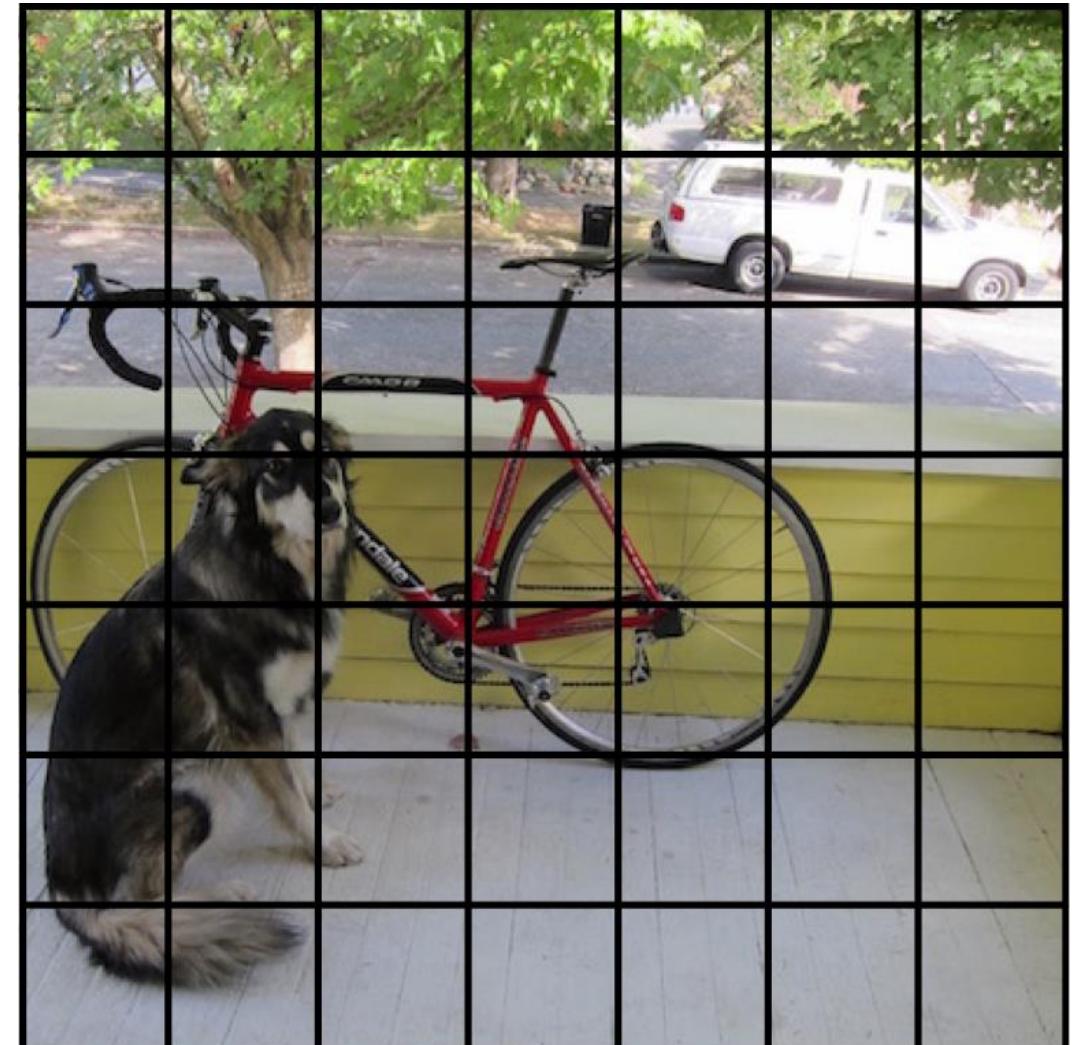
V3

수행시간은 조금 느려졌으나
성능 대폭 개선

Yolo Version – Yolo v1

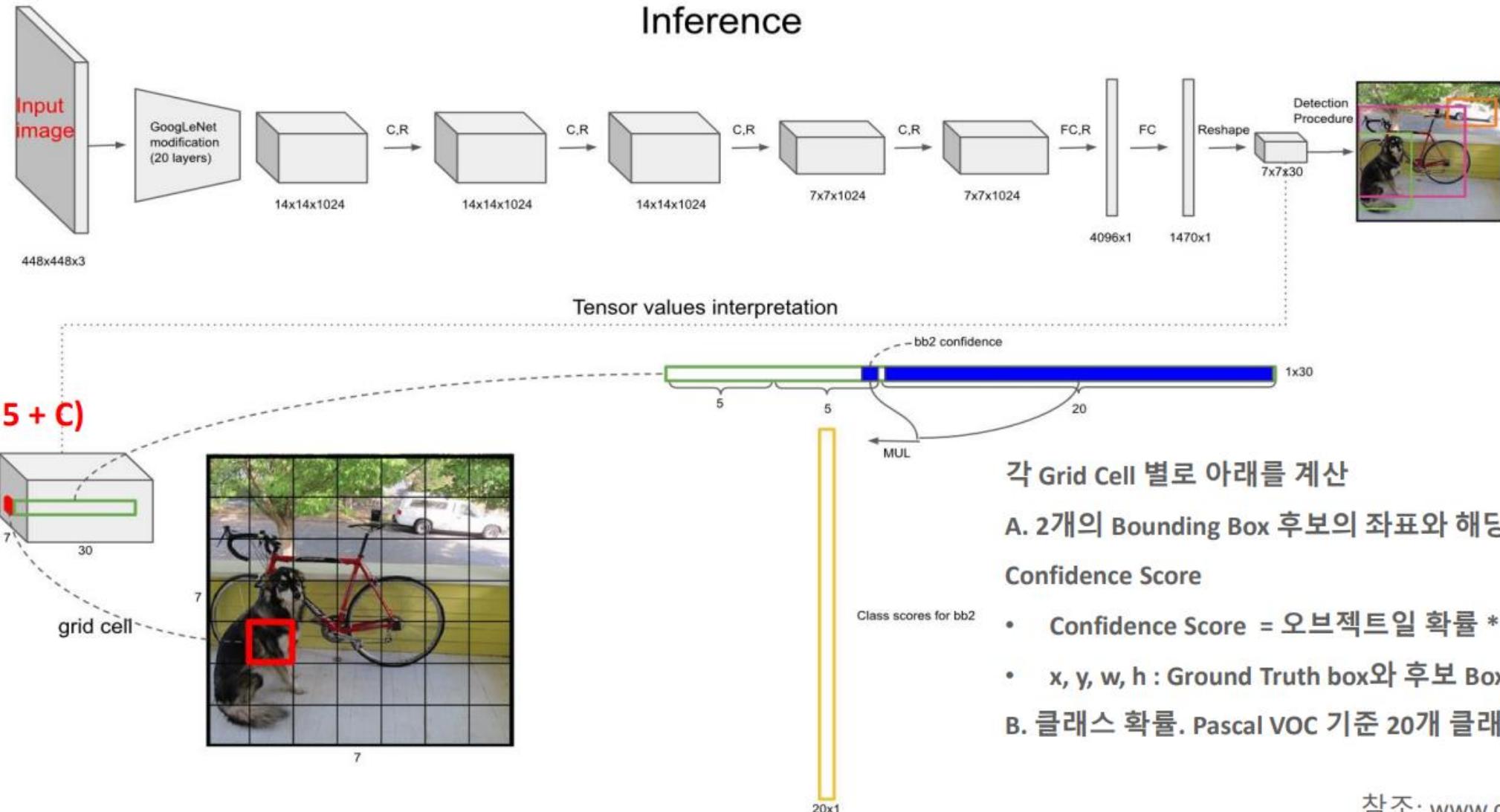
- Yolo V1 은 입력 이미지를 $S \times S$ Grid로 나누고 각 Grid의 Cell이 하나의 Object에 대한 Detection 수행
- 각 Grid Cell 이 2개의 Bounding Box 후보를 기반으로 Object의 Bounding Box 를 예측

448 X 448
Image

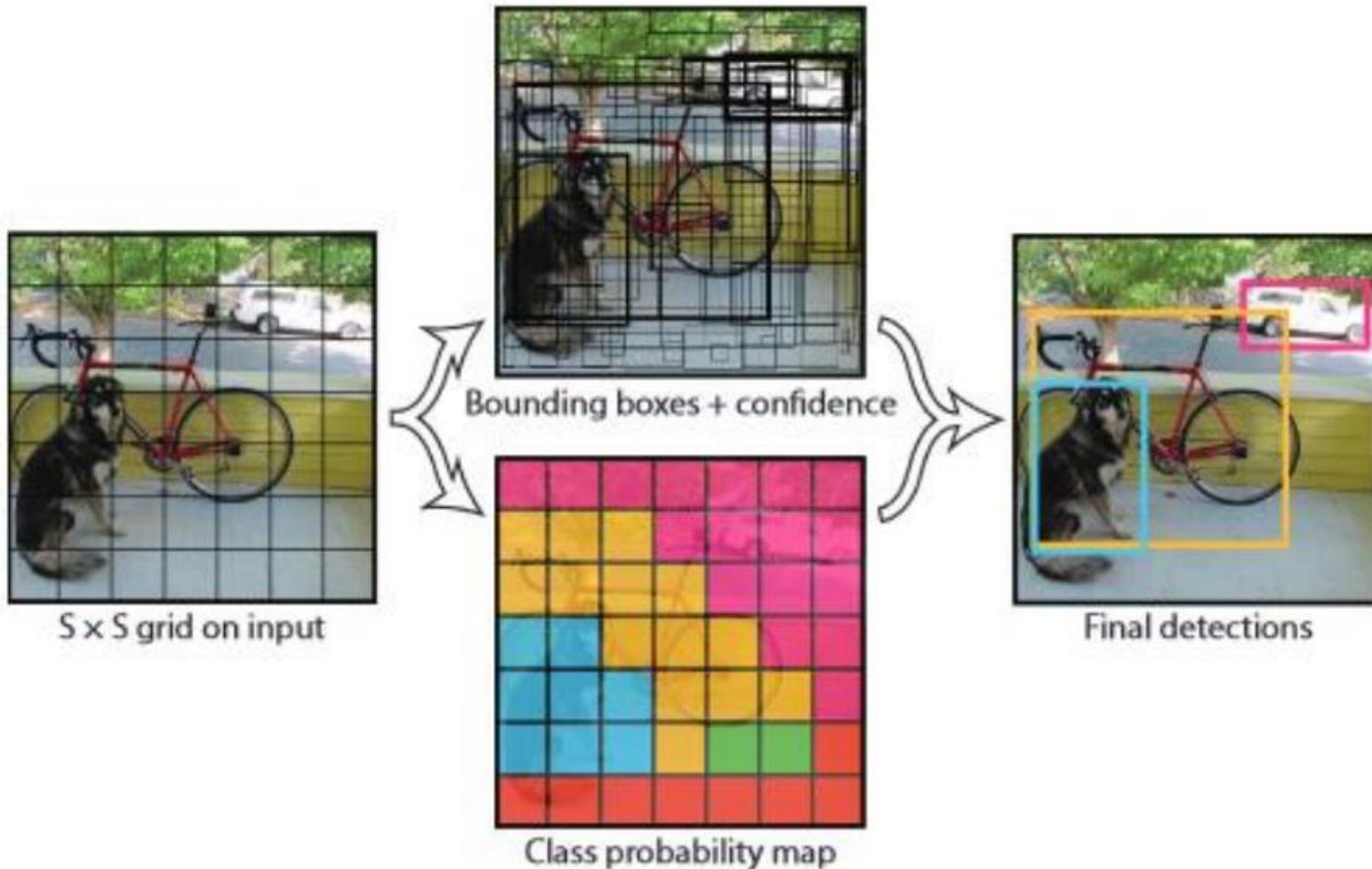


$S \times S$ Grid

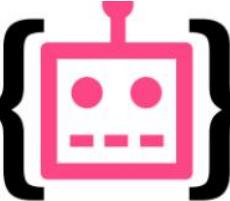
Yolo Version – Yolo v1



Yolo Version – Yolo v1

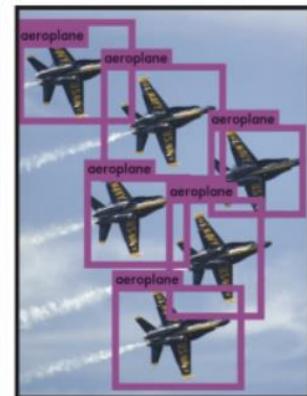
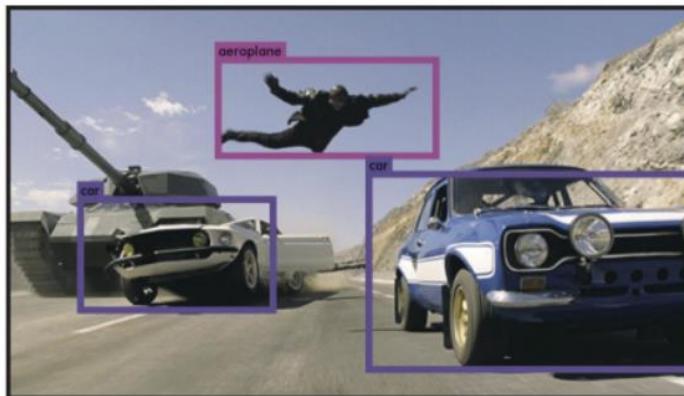
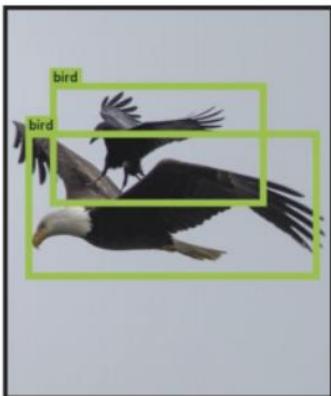


Yolo Version – Yolo v1

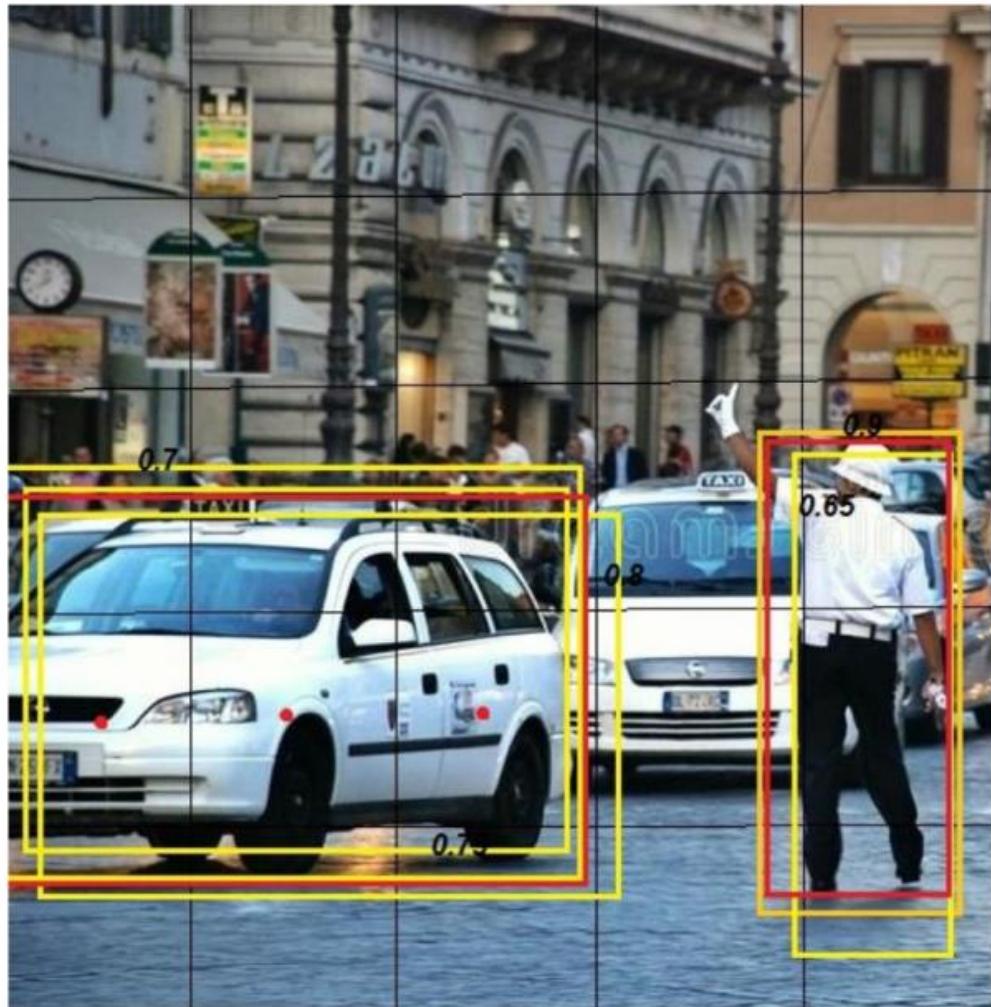
Обзор от  deepsystems.io

YOLO

You Only Look Once: Unified, Real-Time Object Detection
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi



NMS(Non Max Suppression)으로 최종 박스 예측



개별 Class 별 NMS 수행

1. 특정 Confidence 값 이하는 모두 제거
2. 가장 높은 Confidence값을 가진 순으로 Bbox 정렬
3. 가장 높은 Confidence를 가진 Bbox와 IOU와 겹치는 부분이 IOU Threshold 보다 큰 Bbox는 모두 제거
4. 남아 있는 Bbox에 대해 3번 Step을 반복

Object Confidence와 IOU Threshold로 Filtering 조절

Yolo Version – Yolo v1

Detection 시간은 빠르나 Detection 성능이 떨어짐. 특히 작은 Object에 대한 성능이 나쁨

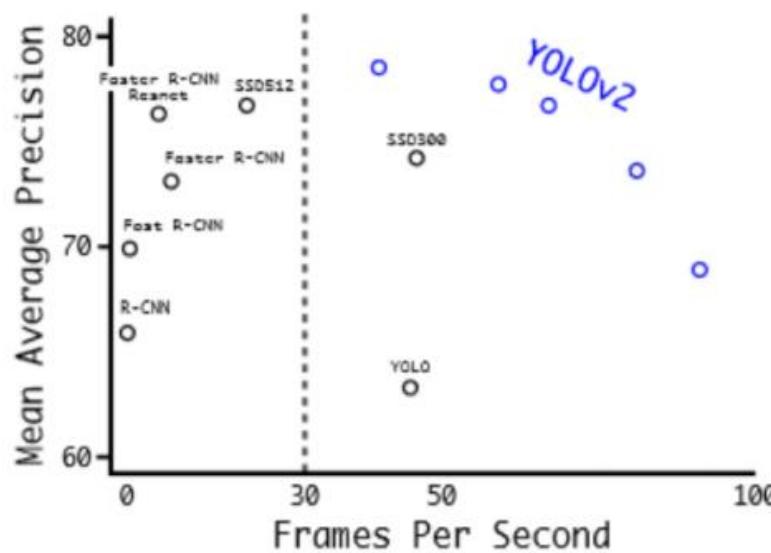
Yolo Version – 버전별 비교

항목	V1	V2	V3
원본 이미지 크기	446 X 446	416 X 416	416 X 416
Feature Extractor	Inception 변형	Darknet 19	Darknet 53
Grid당 Anchor Box 수	2개	5개	Output Feature Map당 3개 서로 다른 크기와 스케일로 총 9개
Anchor box 결정 방법		K-Means Clustering	K-Means Clustering
Output Feature Map 크기 (Depth 제외)		13 x 13	13 x13, 26 X 26, 52X52 3개의 Feature Map 사용
Feature Map Scaling 기법			FPN(Feature Pyramid Network)

Yolo Version – Yolo v2

- V1 에 비해 시간이 빨라짐
- Small object에 대해서 개선 됨

PASCAL VOC 2007 Detection 시간



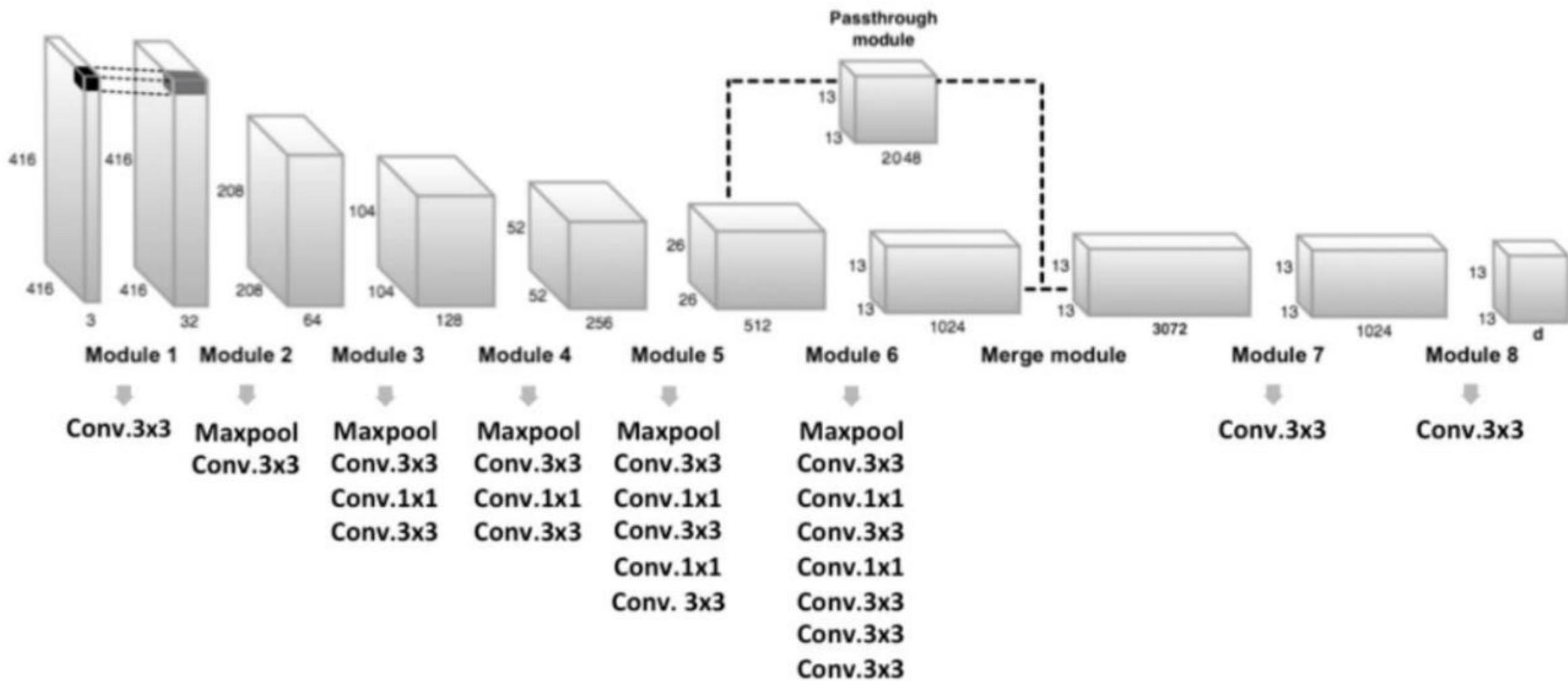
MS-COCO 기준 Detection 성능

		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [5]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN[1]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN[15]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300 [11]	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512 [11]	trainval35k	26.8	46.5	27.8	9.0	28.9	41.9	24.8	37.5	39.8	14.0	43.5	59.0
YOLOv2 [11]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4

Yolo Version – Yolo v2

- Batch Normalization
- High Resolution Classifier : 네트워크의 Classifier 단을 보다 높은 resolution(448x448)로 fine tuning
- 13 x 13 feature map 기반에서 개별 Grid cell 별 5개의 Anchor box에서 Object Detection
- Darknet-19 Classification 모델 채택
- 서로 다른 크기의 image들로 네트워크 학습

Yolo Version – Yolo v2



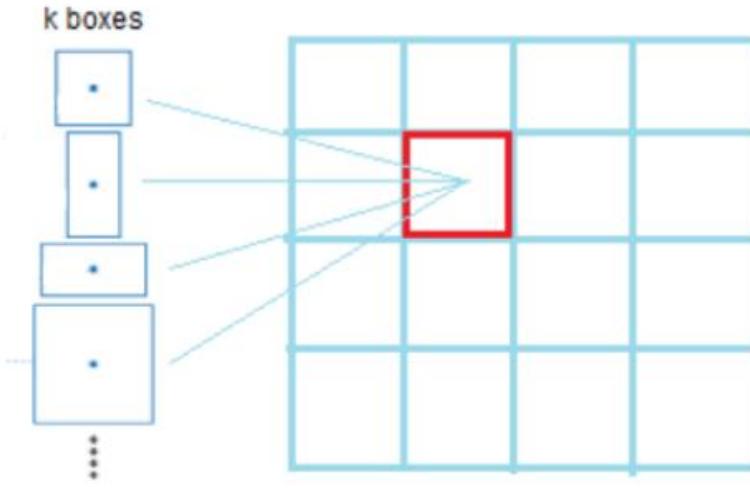
Yolo Version – Yolo v2

Anchor Box로 1Cell에서 여러 개 Object Detection

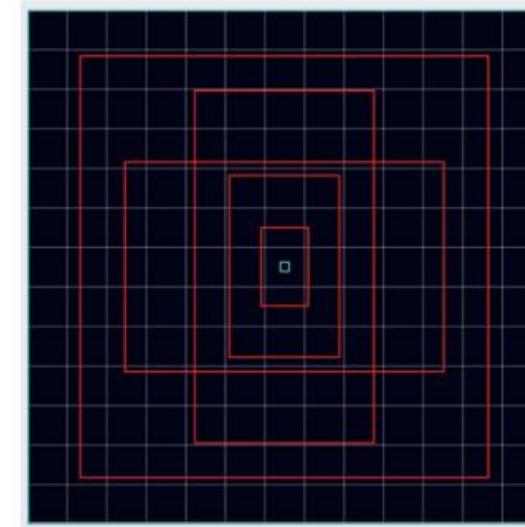
SSD와 마찬가지로 1개의 Cell에서 여러 개의 Anchor를 통해 개별 Cell에서 여러 개 Object Detection 가능

K-Means Clustering을 통해 데이터 세트의 이미지 크기와 Shape Ratio 따른 5개의 군집화 분류를 하여 Anchor Box를 계산

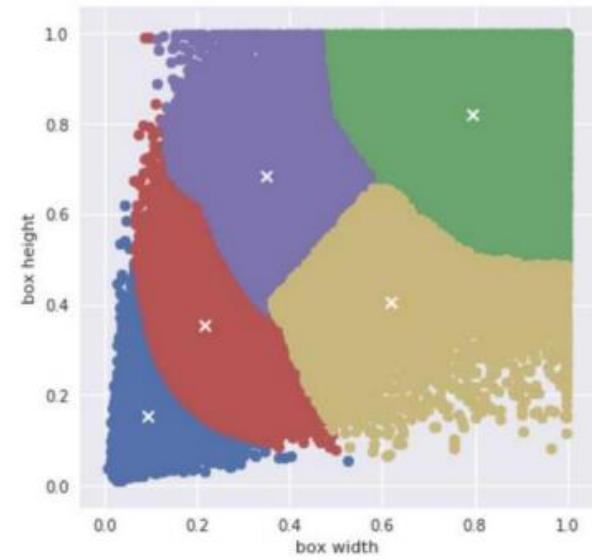
Convolutional With Anchor Boxes



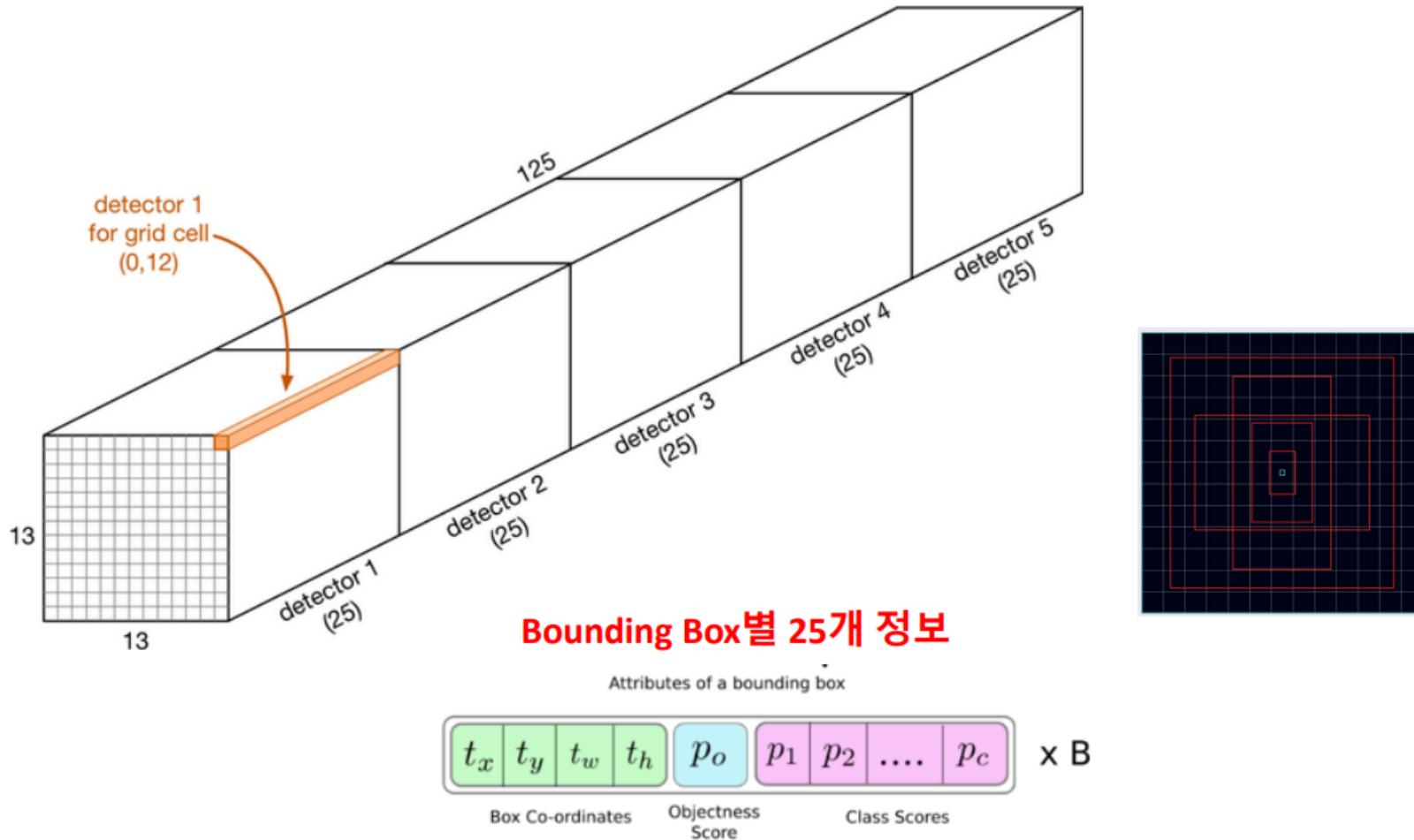
5개 Anchor Box



K-Means Clustering



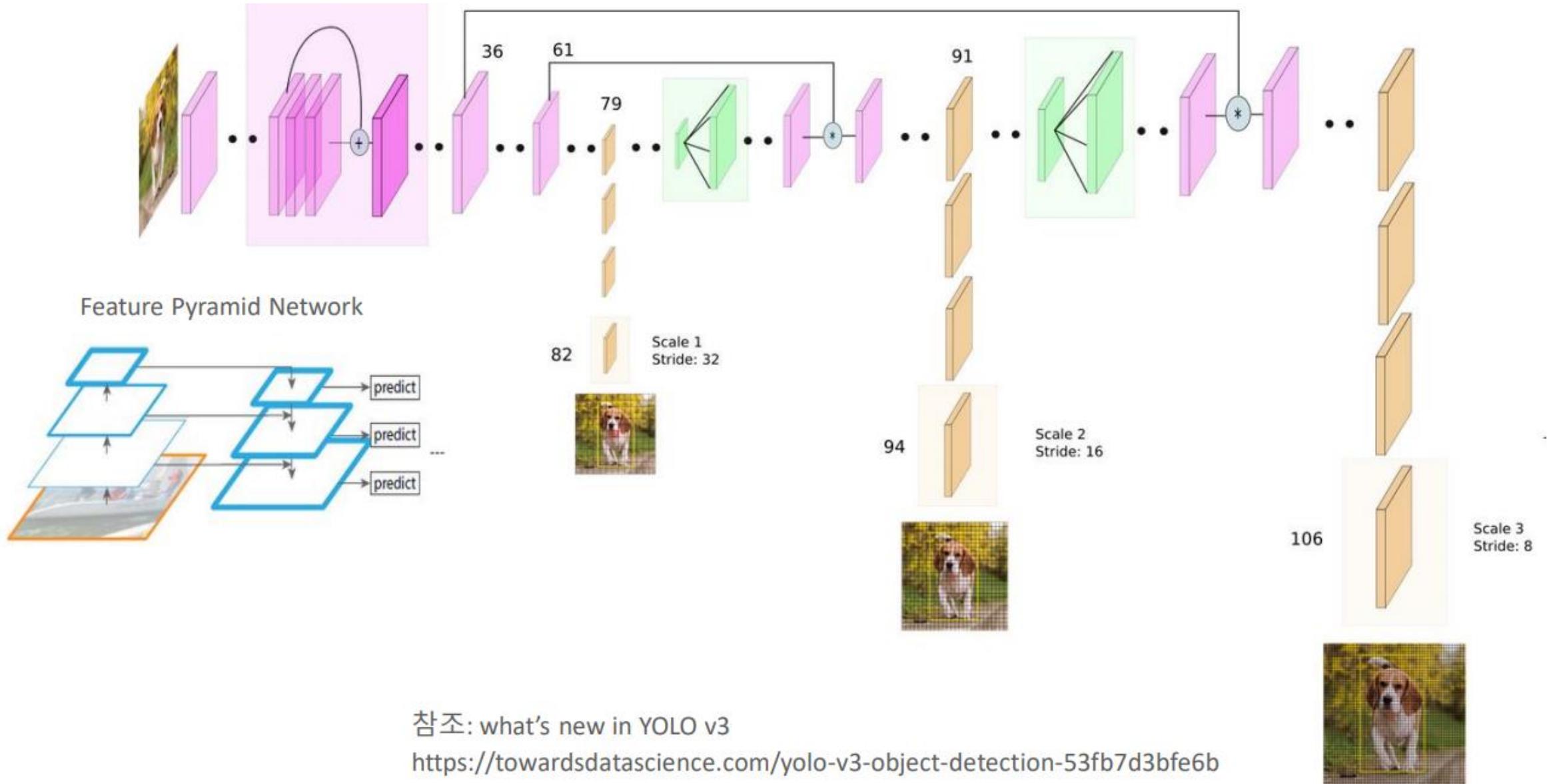
Yolo Version – Yolo v2 Feature Map



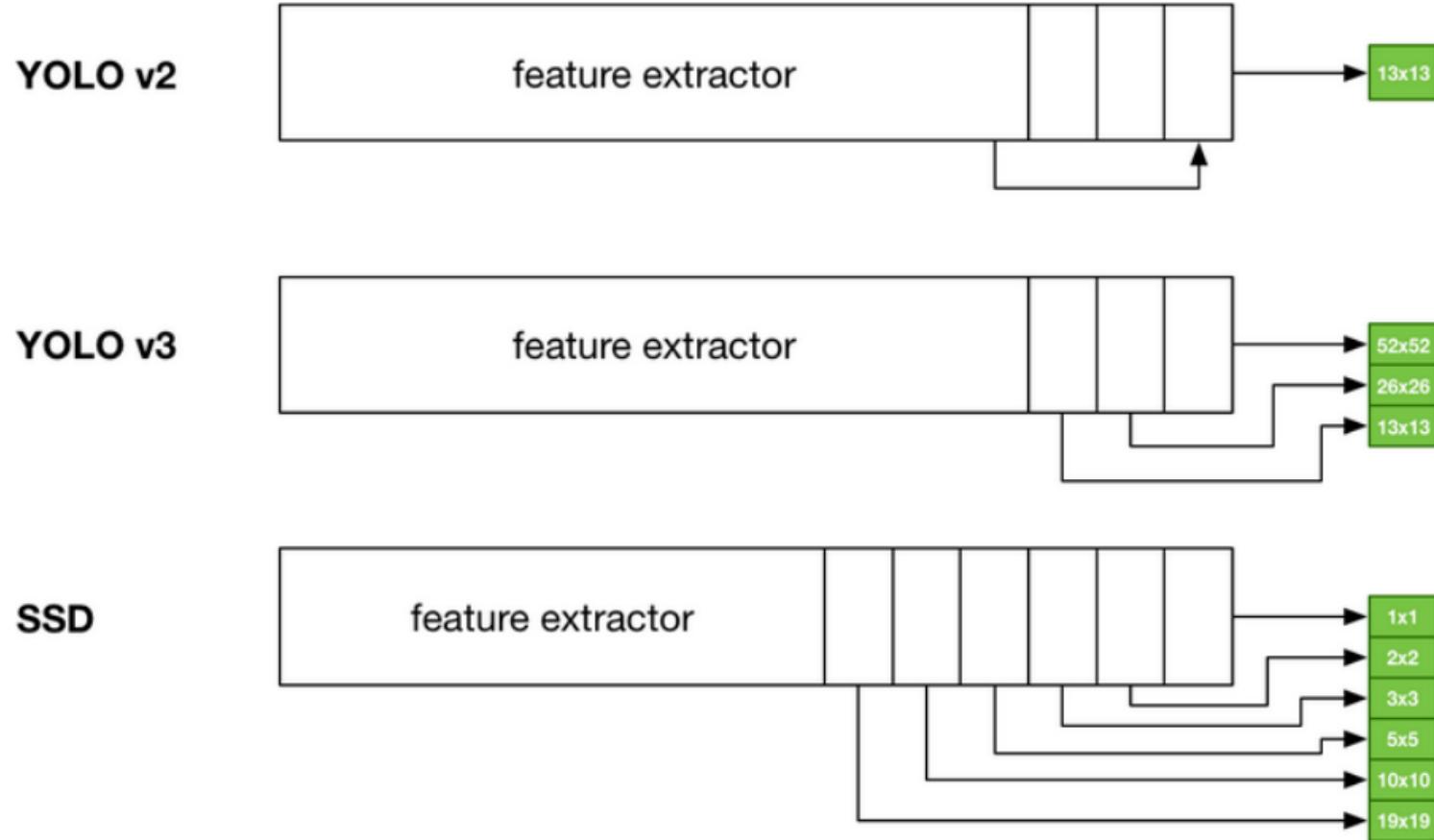
Yolo Version – Yolo v3

- Feature Pyramid Network 유사한 기법을 적용하여 3개의 Feature Map Output에서 각각 3개의 서로 다른 크기와 scale을 가진 anchor box로 Detection
- Classification 단계 보다 높은 Classification을 가지는 Darknet-53
- Multi Labels 예측: Softmax가 아닌 Sigmoid 기반의 logistic classifier로 개별 Object의 Multi labels 예측

Yolo Version – Yolo v3

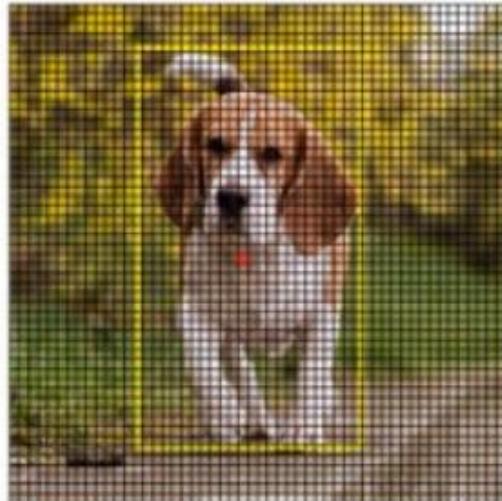
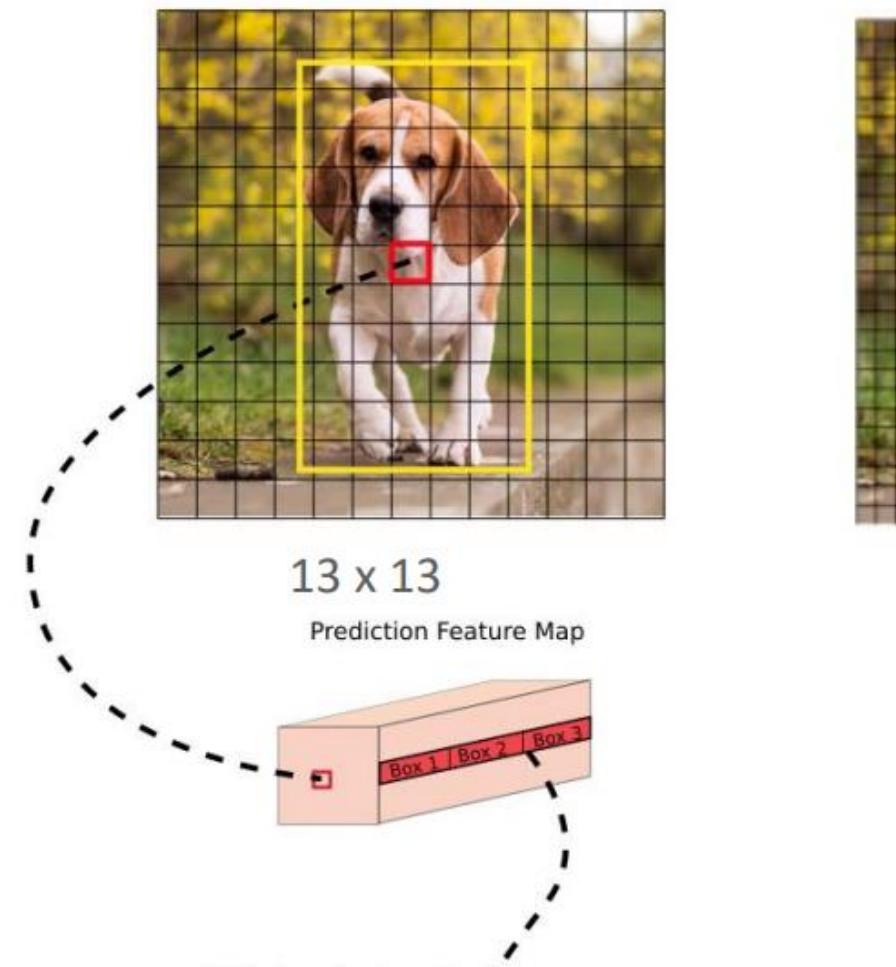


Yolo Version – Yolo v3



참조: one stage object detector
<https://machinethink.net/blog/object-detection/>

Yolo Version – Yolo v3 Feature Map



t_x	t_y	t_w	t_h	p_o	p_1	p_2	p_c
Box Co-ordinates	Objectness Score				Class Scores			

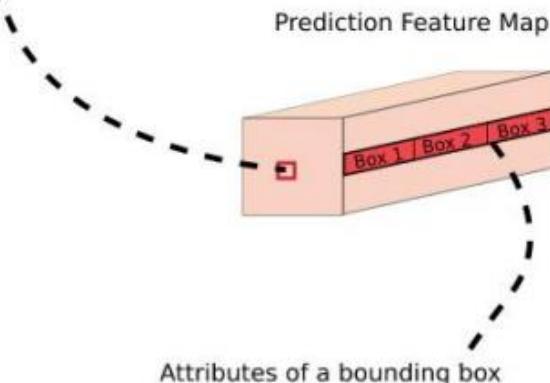
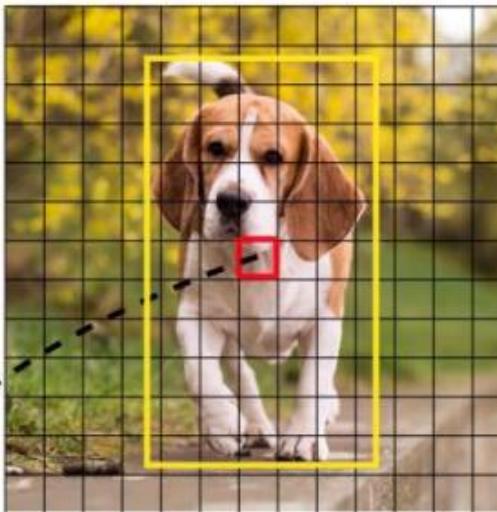
$\times B$

<https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>

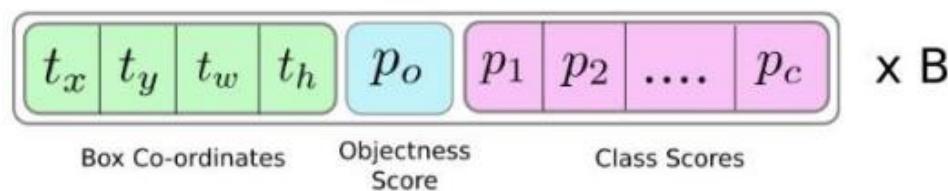
Yolo Version – Darknet53

```
<annotation>
  <folder>VOC2007</folder>
  <filename>003585.jpg</filename>
  <size>
    <width>333</width>
    <height>500</height>
    <depth>3</depth>
  </size>
  <object>
    <name>person</name>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>138</xmin>
      <ymin>183</ymin>
      <xmax>259</xmax>
      <ymax>411</ymax>
    </bndbox>
  </object>
  <object>
    <name>motorbike</name>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>89</xmin>
      <ymin>244</ymin>
      <xmax>291</xmax>
      <ymax>425</ymax>
    </bndbox>
  </object>
  ...

```



Attributes of a bounding box



multi-scale training,
Data augmentation

Yolo Version – Yolo v3 : Training

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

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

Yolo Version – Yolo v3 : Location Prediction

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

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

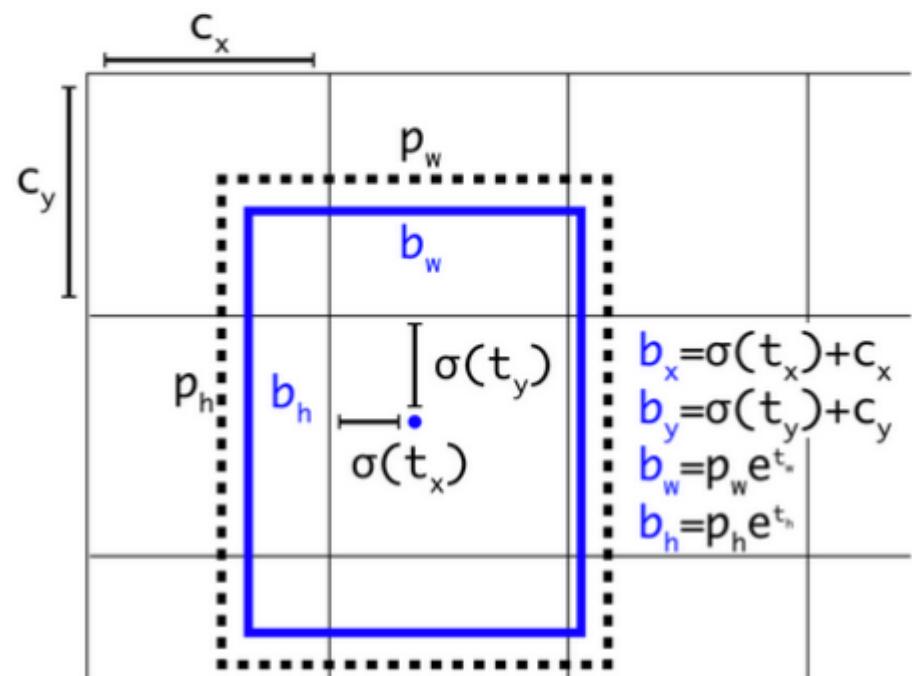
$$b_h = p_h e^{t_h}$$

$$\Pr(\text{object}) \cdot \text{IoU}(b, \text{object}) = \sigma(t_o)$$

(pw,ph): anchor box size

(tx,ty,tw,th) : 모델 예측 offset 값

(bx,by), (bw,bh): 예측 Bounding box 중심 좌표와 Size



Center 좌표가 Cell 중심을 너무 벗어나지 못하도록
0 ~ 1 사이의 시그모이드 값으로 조절

Yolo Version – Yolo v3 : Multi Label Prediction

여러 개의 독립적인 Logistic Classifier 사용

