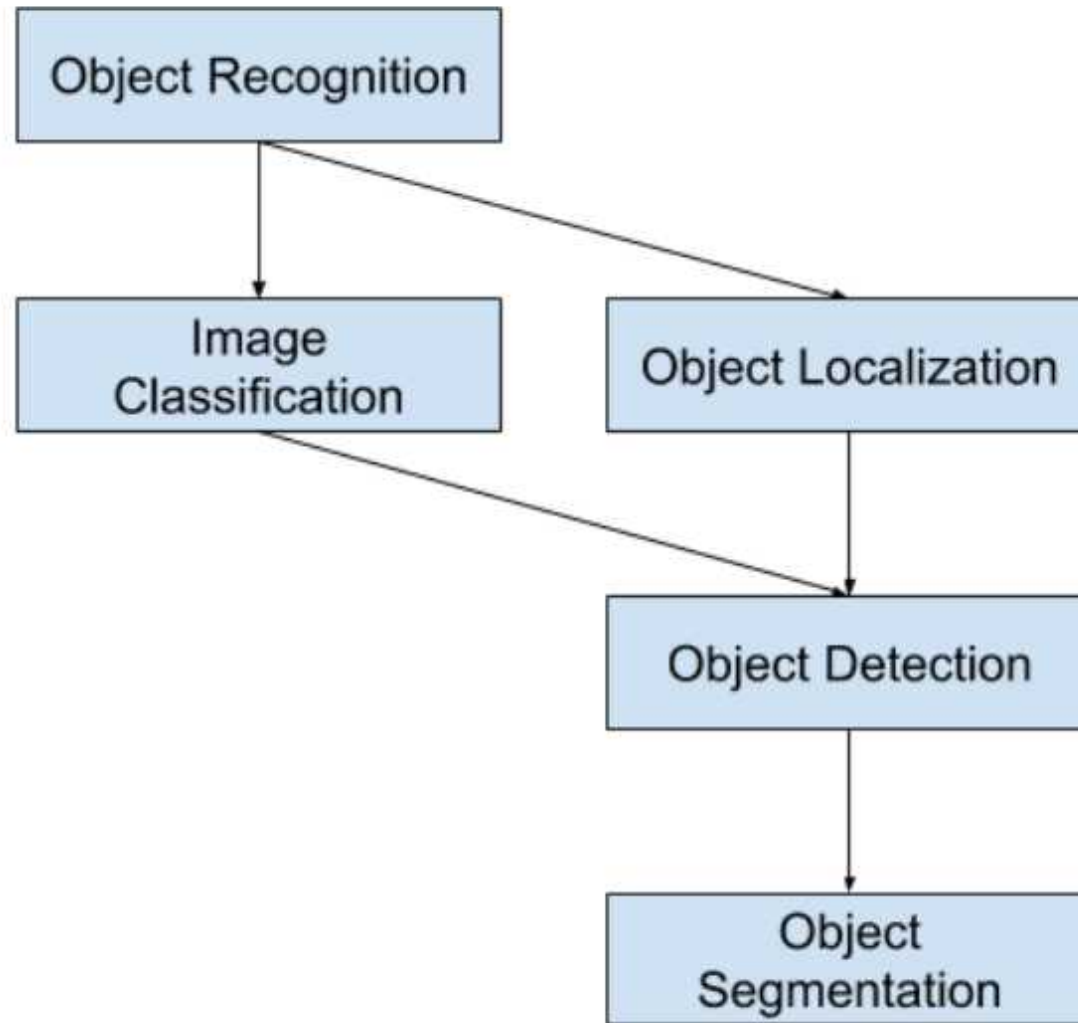


# Object Recognition

# Recognition



# Recognition

**Classification**



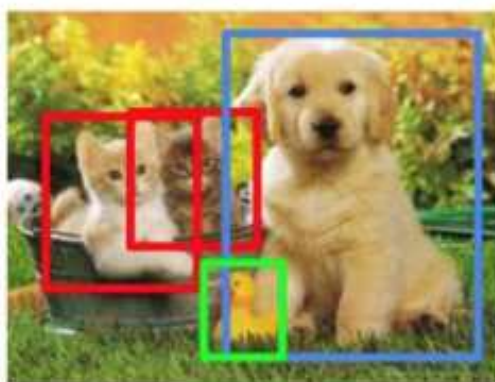
CAT

**Classification  
+ Localization**



CAT

**Object Detection**



CAT, DOG, DUCK

**Instance  
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

## Classification



**CAT**

No spatial extent

## Semantic Segmentation



**GRASS, CAT,**  
**TREE, SKY**

No objects, just pixels

## Object Detection



**DOG, DOG, CAT**

## Instance Segmentation



**DOG, DOG, CAT**

Multiple Object

[This image is CC0 public domain](#)

# Localization/Detection/Segmentation

Object(s)의 위치를 찾아내는 것



ELLEN



ELLEN



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING

하나의 Object

여러 개의 Object들



# Localization/Detection/Segmentation



ELLEN



ELLEN



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING

- Localization: 단 하나의 Object 위치를 Bounding box로 지정하여 찾음
- Object Detection: 여러 개의 Object들에 대한 위치를 Bonding box로 지정하여 찾음.
- Segmentation: Detection보다 더 발전된 형태로 Pixel 레벨 Detection 수행

# Localization과 Detection



ELLEN



ELLEN



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING



ELLEN, JULIA, PETER, JENNIFER, BRADLEY,  
BRAD, MERYL, KEVIN, LUPITA, CHANNING

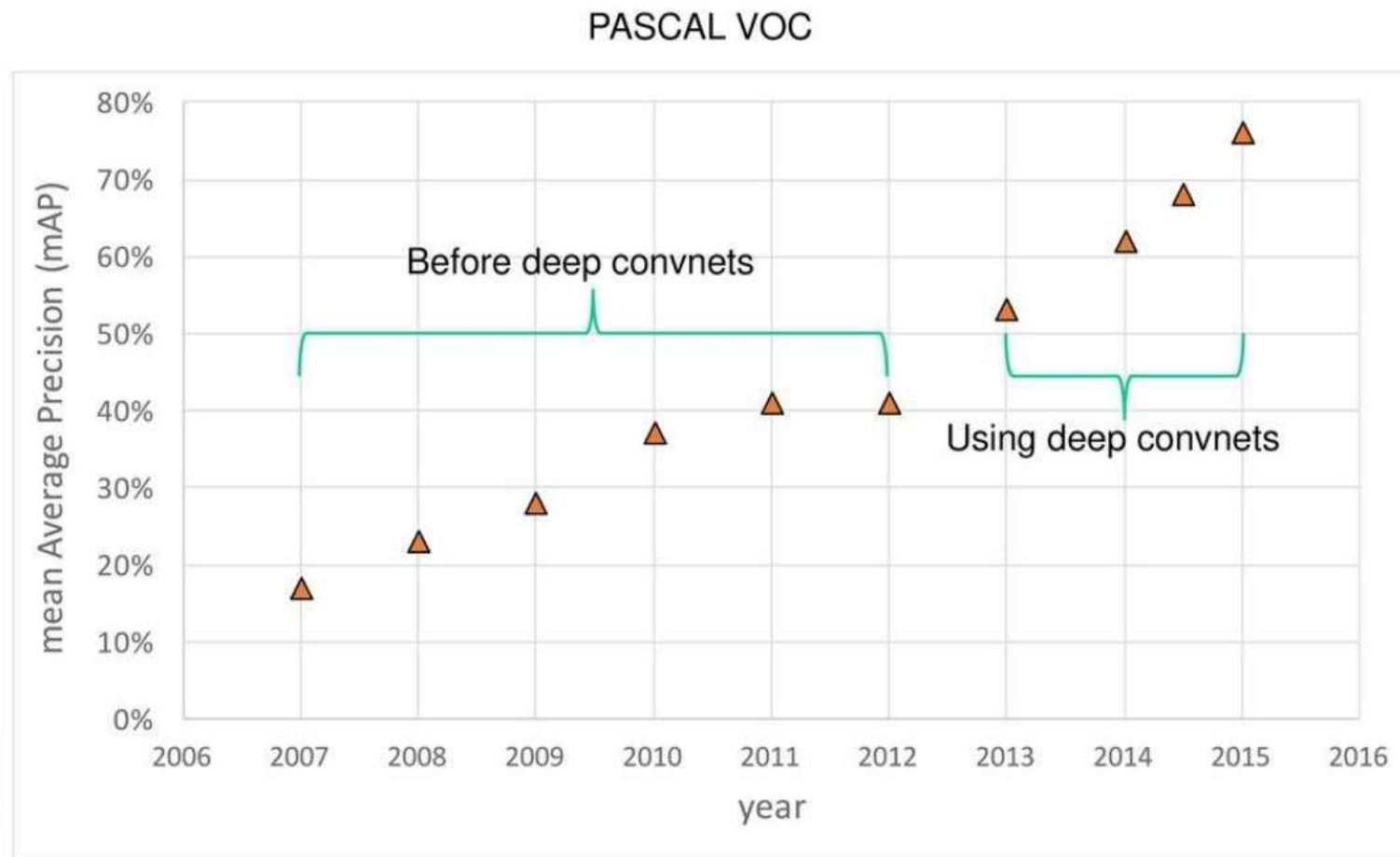
- Localization/Detection은 해당 Object의 위치를 Bounding box로 찾고, Bounding Box내의 오브젝트를 판별합니다.
- Localization/Detection은 Bounding box regression(box의 좌표값을 예측)과 Classification 두개의 문제가 합쳐져 있습니다.
- Localization에 비해 Detection은 두개 이상의 Object를 이미지의 임의 위치에서 찾아야 하므로 상대적으로 Localization 보다 여러가지 어려운 문제에 봉착하게 됩니다.

# Object Detection



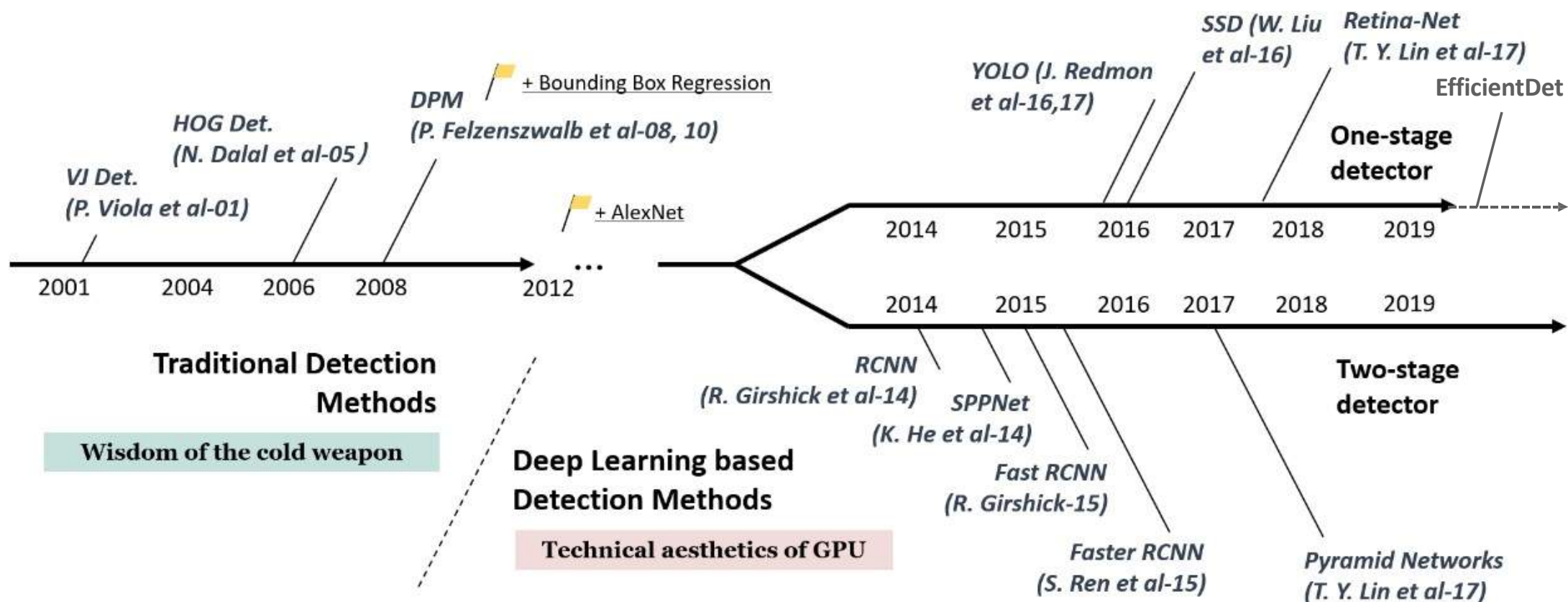
- 인간의 경우 (아직 완전히 밝혀지지 않은 모종의 매커니즘에 의해) '선택적 주의 집중(selective attention)' 및 '문맥(context)'에 기반한 '종합적 이해' 등의 과정을 거치며, 이 작업을 직관적으로 빠른 속도로 정확하게 수행
- 반면, 기계는 '선택적 주의 집중' 능력이 없기 때문에 픽셀의 값을 빠짐없이 하나하나 다 살펴봐야 함
- 이 과정에서 속도가 느려질 수밖에 없으며, 이렇게 읽어 들인 픽셀로부터 어떻게 '문맥' 정보를 추출하고, 또 이들을 어떻게 '종합하고 이해' 하는 것이 최적인지도 알지 못하므로 그 성능 또한 인간에 한참 뒤떨어질 수밖에 없음

# Object Detection – Deep learning 도입 후 크게 성능향상



# Object Detection History

<https://arxiv.org/pdf/1905.05055.pdf>



# Object Detection의 주요 구성 요소

영역 추정

Region Proposal

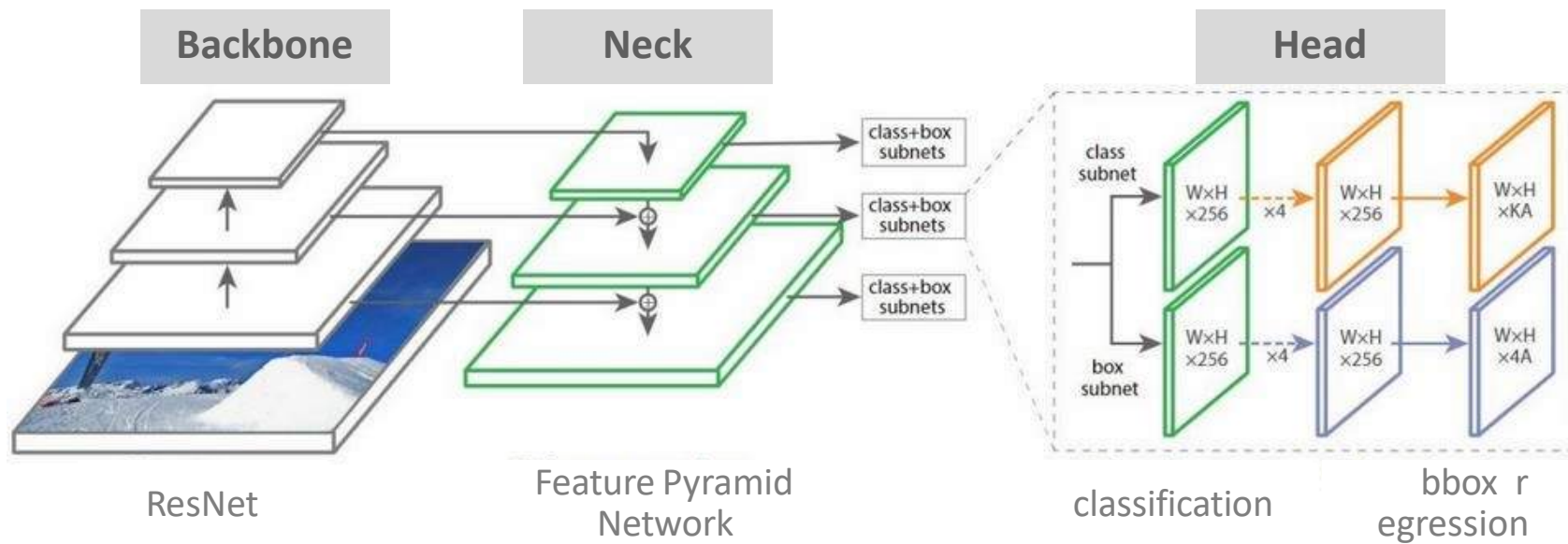
Detection을 위한 Deep  
Learning 네트워크 구성

Feature Extraction  
&  
FPN  
& Network  
Prediction

Detection을 구성하는  
기타 요소

IOU,  
NMS,  
mAP,  
Anchor box

# 일반적인 Object Detection 모델



# Object Detection의 난제

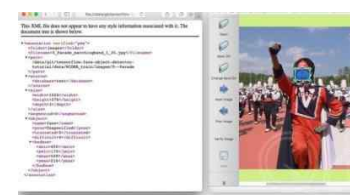
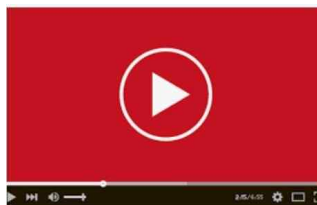
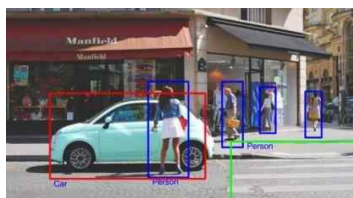
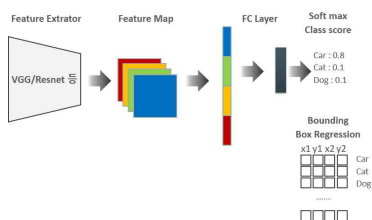
Classification + Regression을 동시에

다양한 크기와 유형의 오브젝트가 섞여 있음

중요한 Detect 시간

명확하지 않은 이미지

데이터 세트의 부족



이미지에서 여러 개의 물체를 classification함과 동시에 위치를 찾아야 함

크기가 서로 다르고, 생김새가 다양한 오브젝트가 섞여 있는 이미지에서 이들을 Detect해야 함.

Detect 시간이 중요한 실시간 영상 기반에서 Detect해야 하는 요구사항 증대

오브젝트 이미지가 명확하지 않은 경우가 많음. 또한 전체 이미지에서 Detect할 오브젝트가 차지하는 비중이 높지 않음.(배경이 대부분을 차지하는 경우가 많음)

훈련 가능한 데이터 세트가 부족(MS Coco dataset 80개, Google Open Image 500개)하며 annotation을 만들어야 하므로 훈련 데이터 세트를 생성하기가 상대적으로 어려움

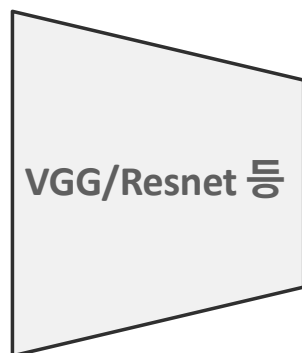


# Object Localization 개요

원본 이미지



Feature Extrator



VGG-16



Feature Map



FC Layer



Soft max  
Class score

Car : 0.8  
Cat : 0.1  
Dog : 0.1

# Object Localization 개요

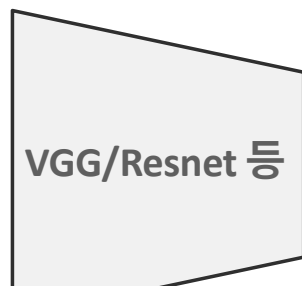
원본 이미지



Annotation 파일

```
<annotation>
  <folder>VOC2012</folder>
  <filename>2007_000032.jpg</filename>
  <source>
    <database>The VOC2007 Database</database>
    <annotation>PASCAL VOC2007</annotation>
    <image>flickr</image>
  </source>
  <size>
    <width>500</width>
    <height>281</height>
    <depth>3</depth>
  </size>
  <segmented>1</segmented>
  <object>
    <name>aeroplane</name>
    <pose>Frontal</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>104</xmin>
      <ymin>78</ymin>
      <xmax>375</xmax>
      <ymax>183</ymax>
    </bndbox>
  </object>
</annotation>
```

Feature Extrator



VGG-16



Feature Map



FC Layer



Soft max  
Class score

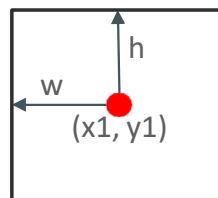


Car : 0.8  
Cat : 0.1  
Dog : 0.1

(x1, y1)

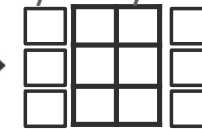


(x2, y2)



Bounding Box Regression x

1 y1 x2 y2



Car  
Cat  
Dog

.....

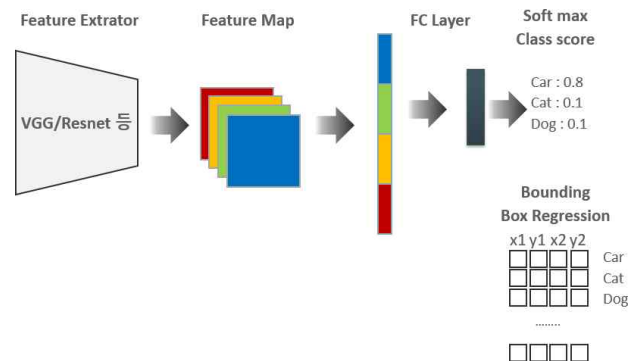


# Object Localization – Bounding box 학습

원본 이미지의 Bounding box



학습된 Object의 Bounding Box



$$\begin{aligned}x1 &= w10*f1 + w20*f2 + w30*f3 + \dots\dots\dots \\y1 &= w11*f1 + w21*f2 + w31*f3 + \dots\dots\dots \\x2 &= w12*f1 + w22*f2 + w33*f3 + \dots\dots\dots \\y2 &= w13*f1 + w23*f2 + w33*f3 + \dots\dots\dots\end{aligned}$$

가중치 update

가중치 update

$$\begin{aligned}x1 &= w10*f1 + w20*f2 + w30*f3 + \dots\dots\dots \\y1 &= w11*f1 + w21*f2 + w31*f3 + \dots\dots\dots \\x2 &= w12*f1 + w22*f2 + w33*f3 + \dots\dots\dots \\y2 &= w13*f1 + w23*f2 + w33*f3 + \dots\dots\dots\end{aligned}$$

# 여러 이미지와 Bounding box 좌표로 학습



# Object Localization 예측 결과



Class	Confidence Score	X1	Y1	X2	Y2
Car	0.9	50	60	220	150
Dog	0.1	210	10	240	30
Cat	0.1	160	100	180	120

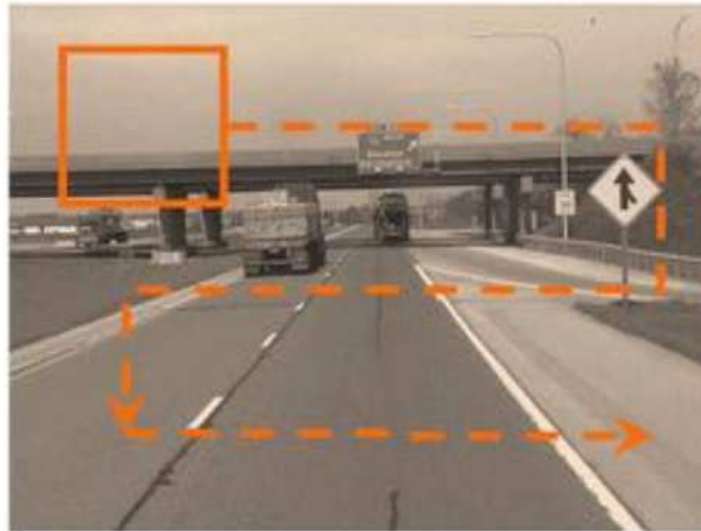
# Object Detection – 두개 이상의 Object를 검출

이미지의 어느 위치에서 Object를 찾아야 하는가?





# Naïve Approach



- 물체가 존재할 수 있는 모든 크기의 영역(different scale & ratio)에 대해 sliding window 방식으로 이미지를 모두 탐색하면서 classification을 수행
- 탐색해야 하는 영역의 수가 너무 많기 때문에 classifier가 충분히 빠르지 않다면 연산 시간이 오래 걸리는 비효율적인 방법

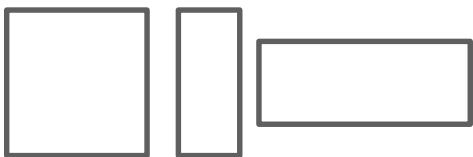
# Naïve(Transitional) Approach

- Haar Features
- Cascading Classifiers
- the Viola-Jones algorithm

# Sliding Window 방식

Window를 왼쪽 상단에서 부터 오른쪽 하단으로 이동시키면서 Object를 Detection하는 방식

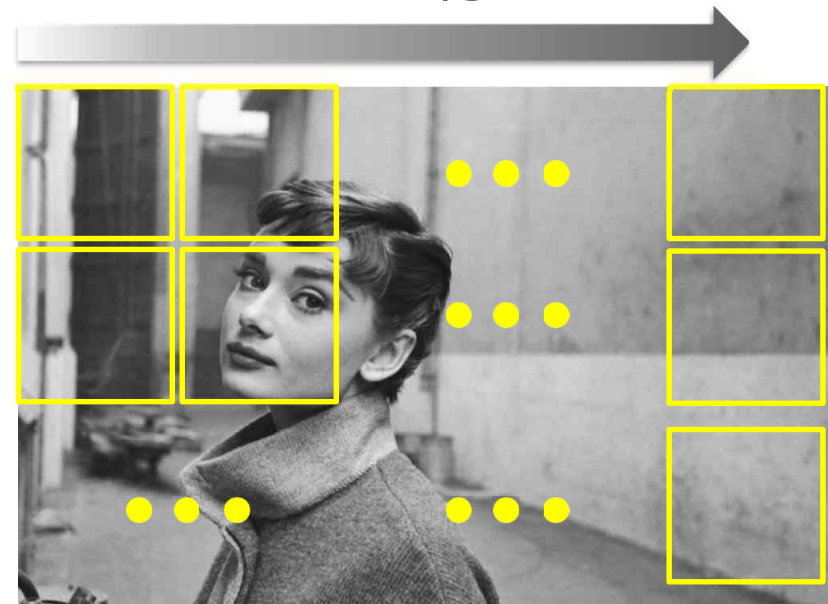
다양한 형태의 Window를 각각 sliding 시키는 방식



Window Scale은 고정하고 scale을 변경한 여러 이미지를 사용하는 방식

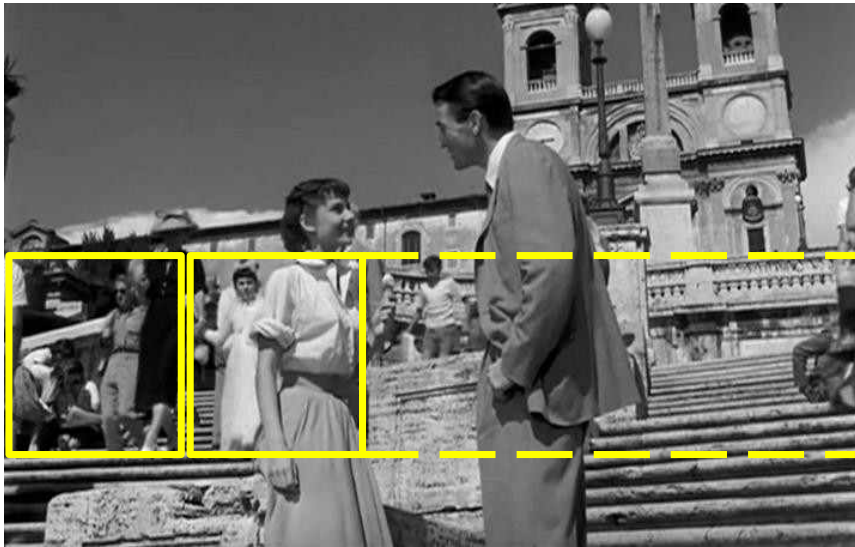


Window 이동



- 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

# 이미지 Scale 조정에 따른 여러 크기의 Object Detection



# CNN based Approach (Region Proposal)

윈도우의 일부(하위 집합)만을 활용해 객체 탐지를 효율적으로 할 수 있는 방법

## [1] Two-Stage Methods

- 객체를 포함할 가능성이 높은 영역을 선택적 탐색(Selective Search)같은 컴퓨터 비전 기술을 활용하거나 딥러닝 기반의 영역 제안 네트워크(RPN; Region Proposal Network)를 통해 선택. 후보군의 윈도우 세트를 취합하면 회귀 모델과 분류 모델의 수를 공식화해 객체 탐지
- 높은 정확도를 제공하지만 Single-Stage Methods보다 처리 속도가 느림
- Faster R-CNN, R\_FCN, FPN-FRCN

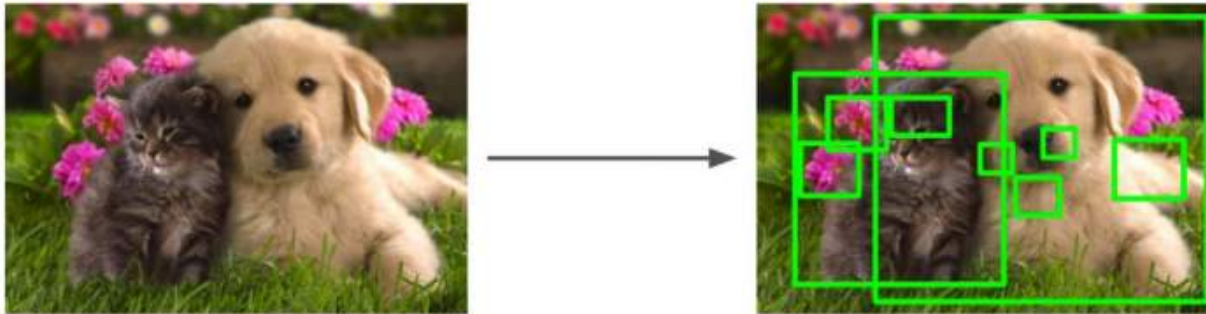
# CNN based Approach (Region Proposal)

## [2] Single-Stage Methods

- 정해진 위치와 정해진 크기의 객체만 찾는 방법. 이 위치와 크기들은 대부분의 시나리오에 적용할 수 있도록 전략적으로 선택
  - 보통 원본 이미지를 고정된 사이즈 그리드 영역으로 나누고 각 영역에 대해 형태와 크기가 미리 결정된 객체의 고정 개수를 예측
- Two-Stage Methods 보다는 정확도가 떨어지지만 빠른 처리가 가능하기 때문에 보통 실시간 탐지에 활용
- YOLO, SSD, RetinaNet



# Region Proposals



- Sliding Window 방식의 비효율성을 개선하기 위해서, 입력 영상에서 '물체가 있을 법한' 영역을 빠른 속도로 찾아내는 알고리즘
- Region proposal을 활용하면, sliding window 방식에 비해 search space가 확연하게 줄어들기 때문에 훨씬 빠른 속도로 Object Detection을 수행

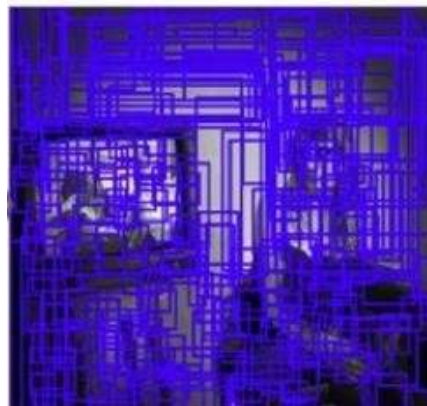
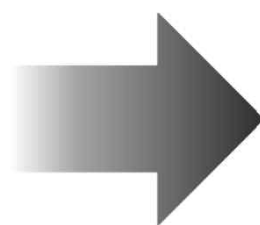
# Region Proposal(영역 추정) 방식

“Object가 있을 만한 후보 영역을 찾자”



원본 이미지

Select



후보 Bounding Box 선택

최종 후보 도출



최종 Object Detection

# Selective Search

- 전체 영역에서 4가지 유사도 색상, 텍스쳐, 사이즈, fill(shape) 기반으로 그것들을 둘러싸는 다양한 크기의 window를 찾음
- python seletive-search  
참고: <https://pypi.org/project/selective-search/>
  - mode: single, fast, quality
  - Similarity m`asure :CTSF (color, texture, size, fill)
  - Starting Regions(k)
  - Number of combination 1,8,80
- hierachial groupingalgorithm
  - 유사도 높은 것끼리 서로 합쳐서 크기 점점 확장해서 물체가 있을 만한 영역 잡음

# Selective Search



원본이미지



mode='single'



mode='fast'



# Selective Search – Region Proposal의 대표 방법

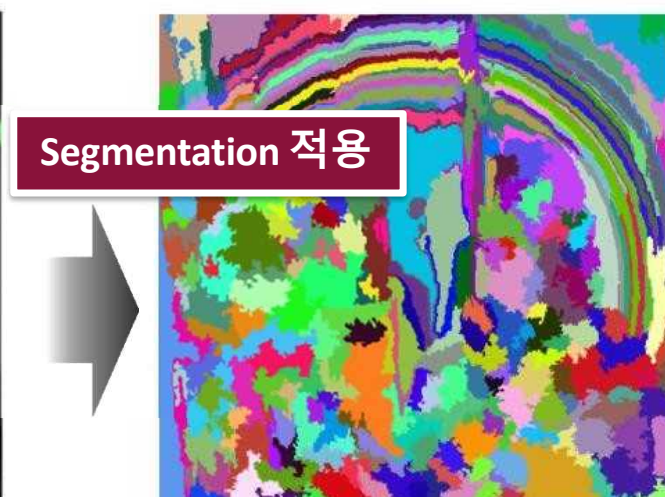
빠른 Detection과 높은 Recall 예측 성능을 동시에 만족하는 알고리즘

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

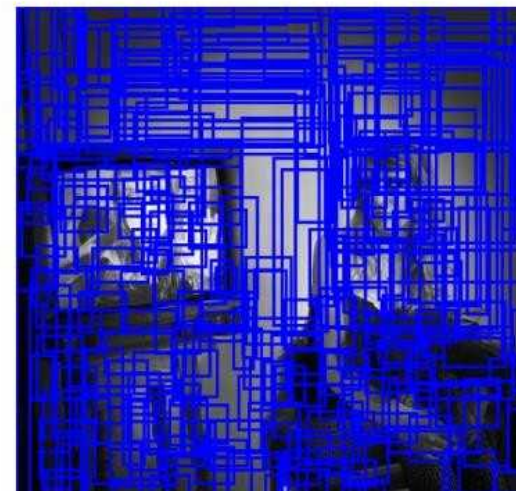
Selective Search는 최초에는 Pixel Intensity기반한 graph-based segment 기법에 따라 Over Segmentation을 수행  
(각각의 object들이 1개의 개별 영역에 담길 수 있도록 많은 초기 영역을 생성 by Felzenszwalb and Huttenlocher 2004)



원본 이미지



최초 Segmentation  
(Over Segmentation)



후보 Objects

# Selective Search의 수행 프로세스

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 반복 하면서 Region Proposal을 수행

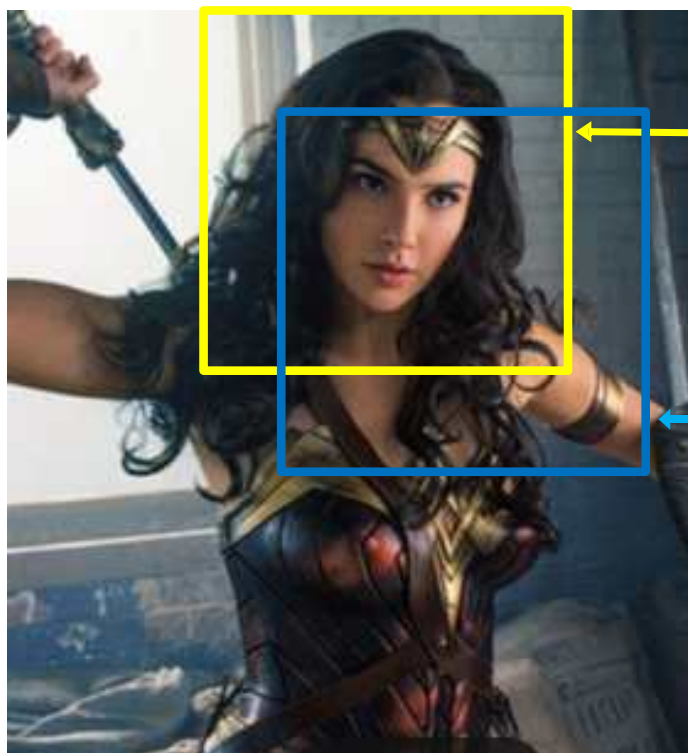




# Object Detection 성능 평가 Metric - IoU

## IoU: Intersection over Union

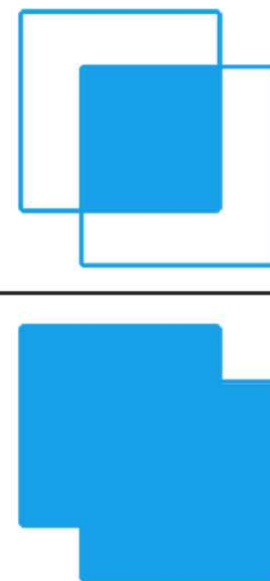
모델이 예측한 결과와 실측(Ground Truth) Box가 얼마나 정확하게 겹치는가를 나타내는 지표



실측(Ground Truth)  
Bounding box

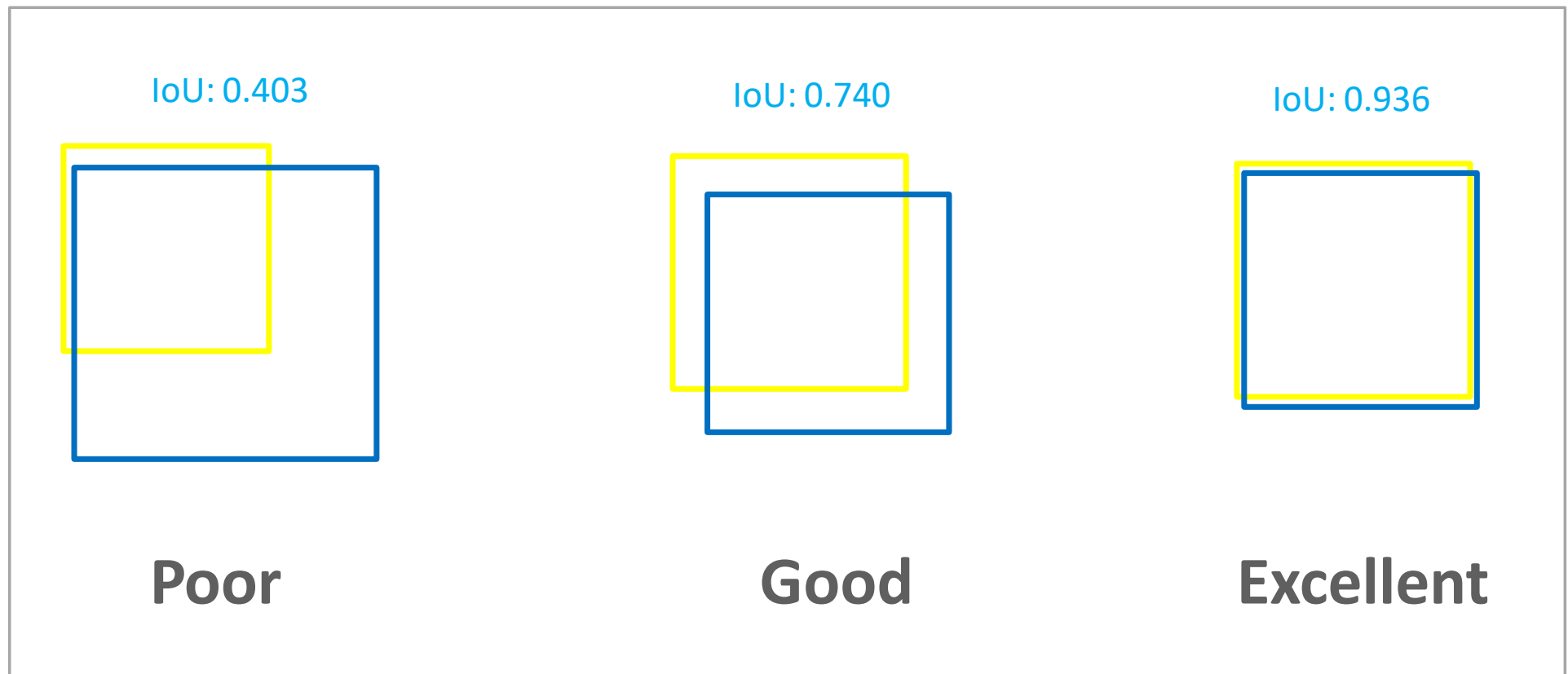
예측(Predicted)  
Bounding box

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

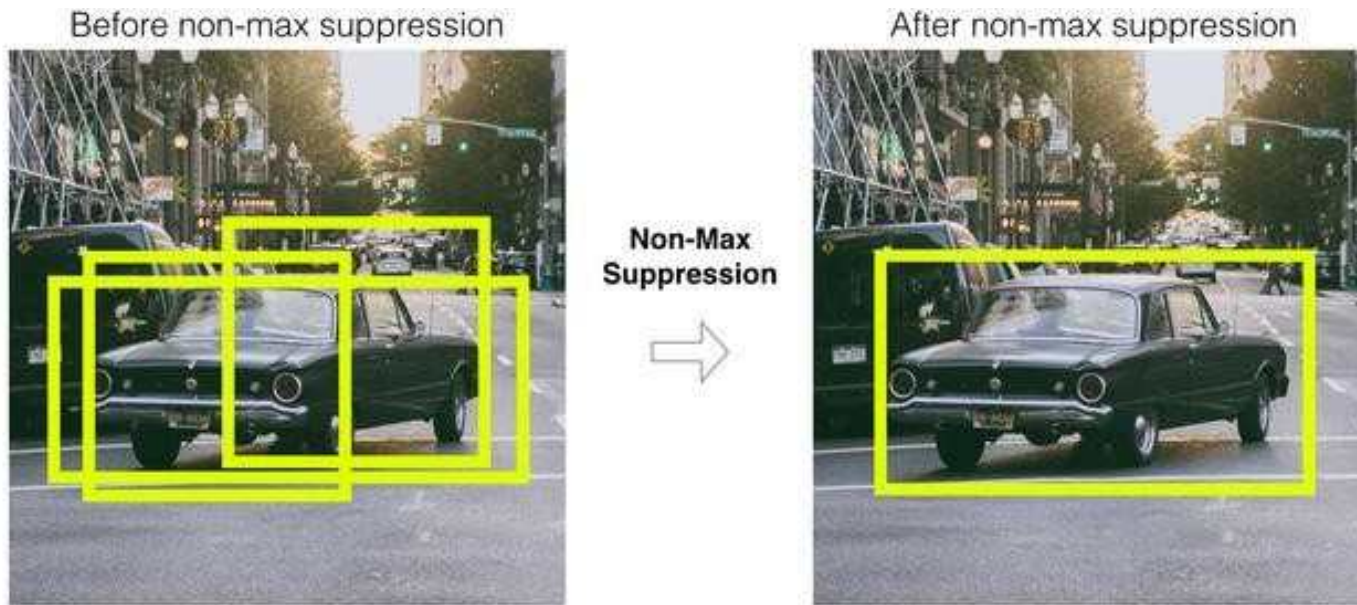


IoU는 개별 Box가 서로 겹치는 영역/ 전체 Box의 합집합 영역

## IoU에 따른 Detection 성능



# NMS(Non Max Suppression)

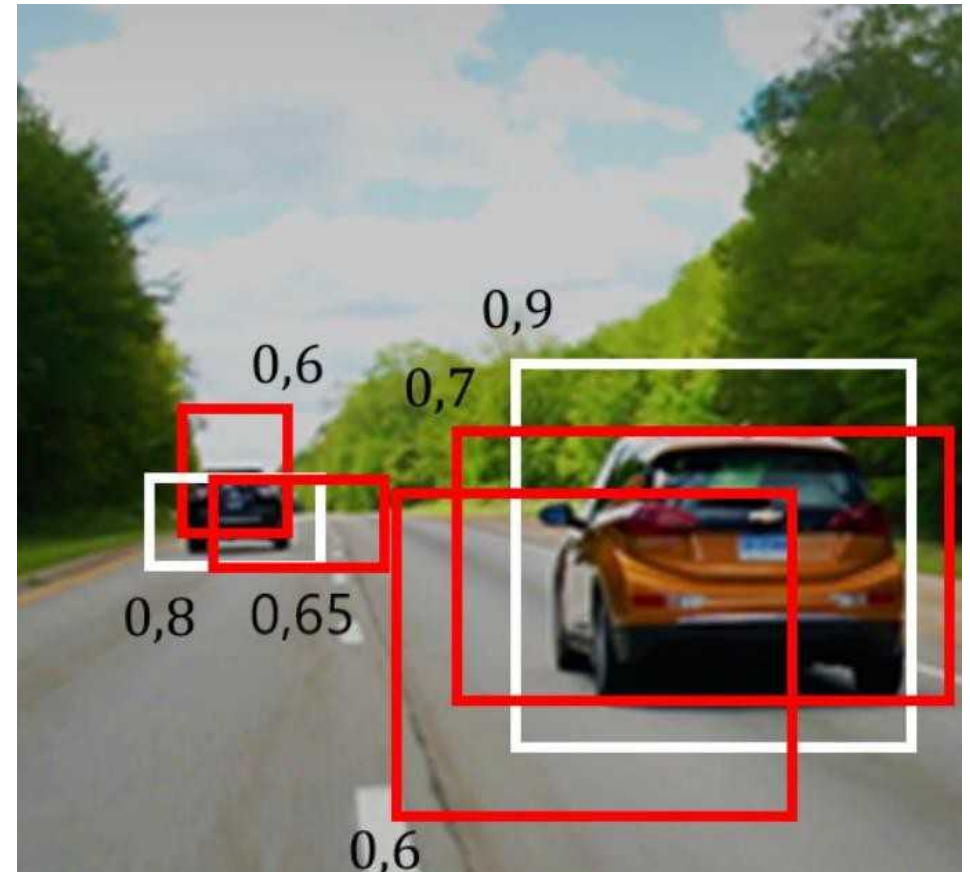


- Object Detection 알고리즘은 Object 가 있을 만한 위치에 많은 Detection을 수행하는 경향이 강함.
- NMS는 Detected 된 Object의 Bounding box중에 비슷한 위치에 있는 box를 제거하고 가장 적합한 box를 선택하는 기법

# NMS 수행 로직

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

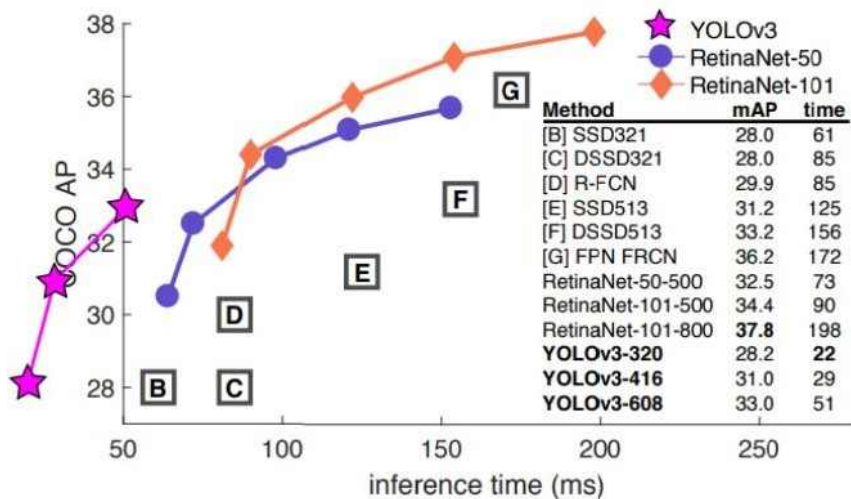
Confidence score가 **높을 수록**,  
IOU Threshold가 **낮을 수록** 많은 Box가 제거됨.



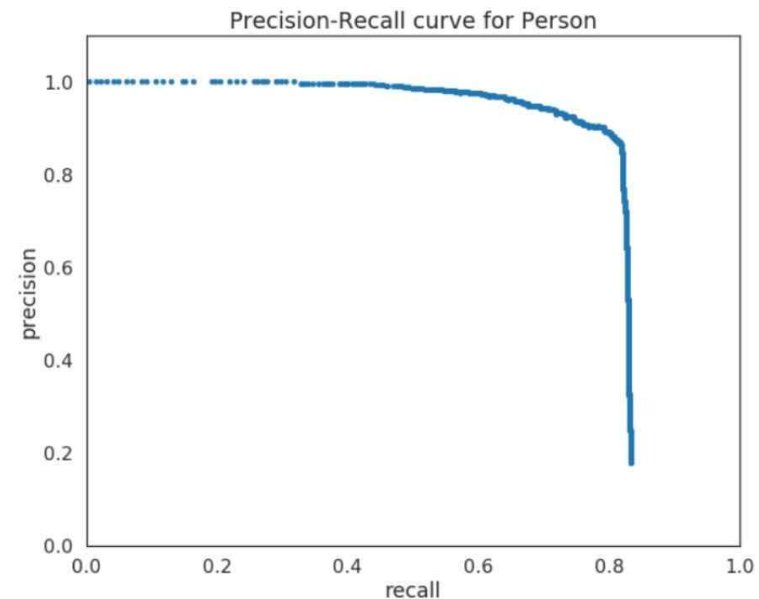
# Object Detection 성능 평가 Metric - mAP

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

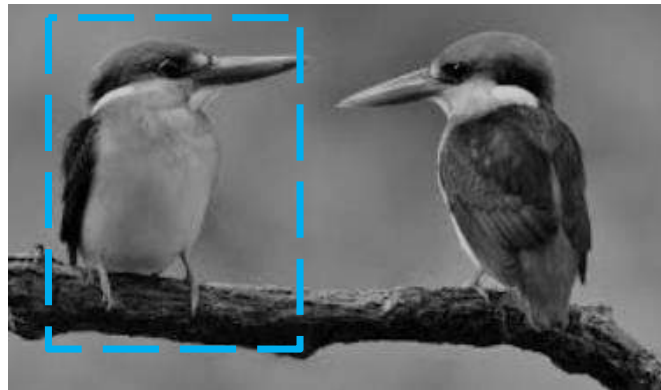


# 정밀도(Precision)과 재현율(Recall)

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%

# IOU 값에 따라 Detection 예측 성공 결정

Object Detection에서 개별 Object에 대한 검출(Detection) 예측이 성공하였는지의 여부를 IOU로 결정.  
일반적으로 PASCAL VOC Challenge 에서 사용된 기준을 적용하여 IOU가 0.5 이상이면 예측 성공으로 인정.  
(그러나 COCO challenge에서는 여러 개의 IOU 기준을 변경해 가면서 예측 성공을 적용)


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} > 0.5$$

예측 성공  
(True Positive)



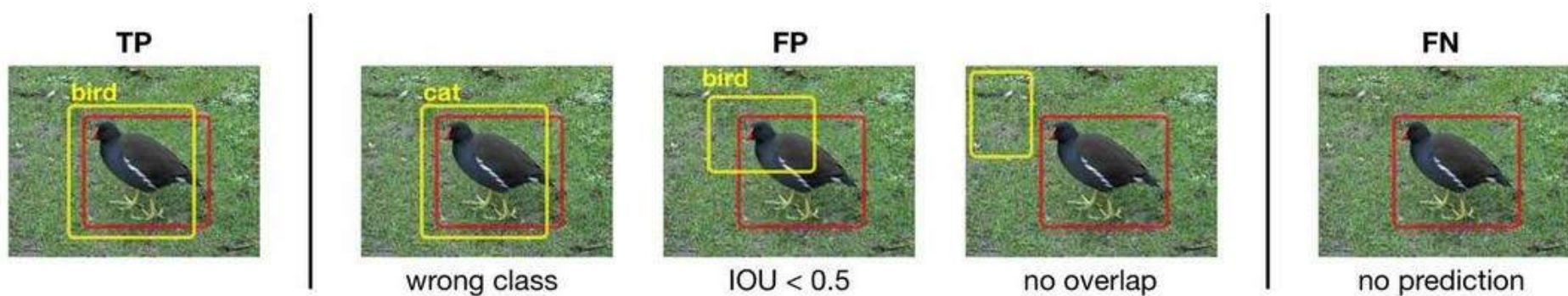
# 오차 행렬(Confusion Matrix)

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)
실제 클래스 (Actual Class)	Negative(0)	Negative Negative (True Negative) TN	Negative Positive (False Positive) FP
	Positive(1 )	Positive Negative (False Negative) FN	Positive Positive (True Positive) TP

# Object Detection에서 TP, FP, FN에 따른 정밀도와 재현율

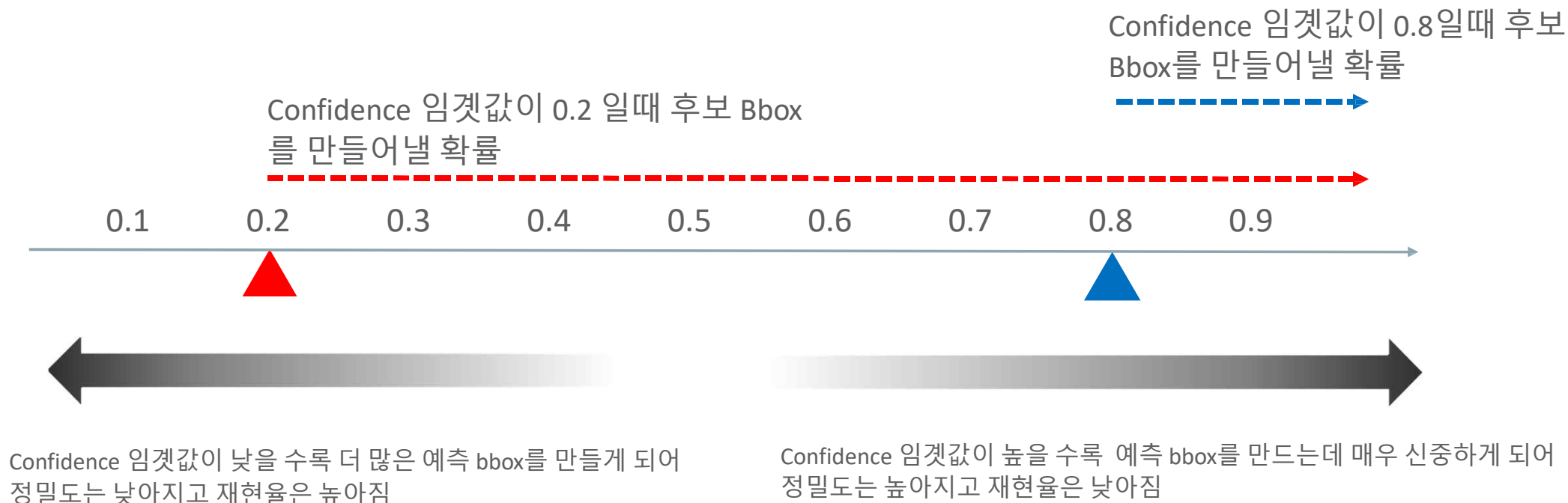
Object Detection에서 TP, FP, FN



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

		예측 클래스(Predicted Class)	
		Negative(0)	Positive (1)
실제 클래스 (Actual Class)	Negative(0)	TN (True Negative)	FP (False Positive)
	Positive(1)	FN (False Negative)	TP (True Positive)

# Confidence 임계값에 따른 정밀도-재현율 변화



Confidence 임계값에 따라 정밀도와 재현율의 값이 변화됨.

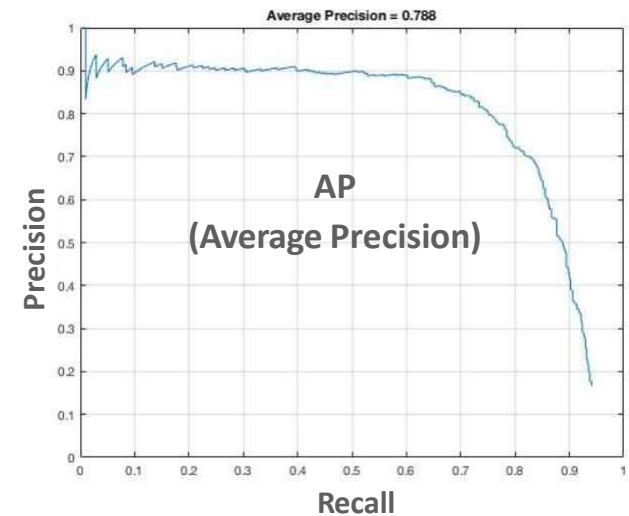
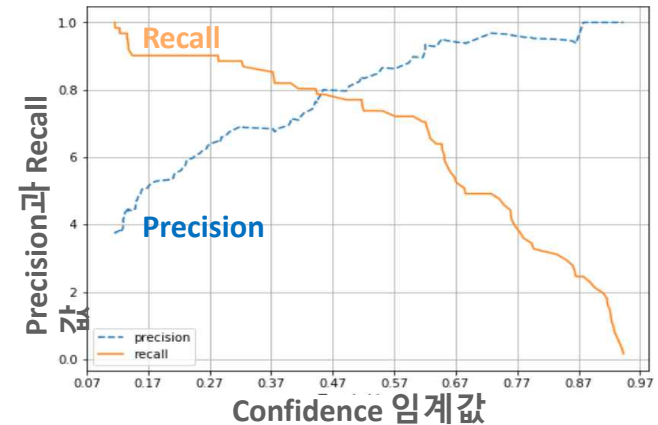
# 정밀도/재현율 트레이드오프

## 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 Precision과 Recall의 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

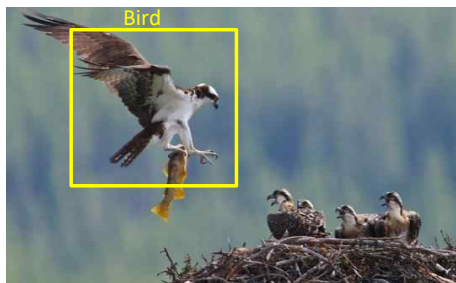
Confidence : 0.7

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

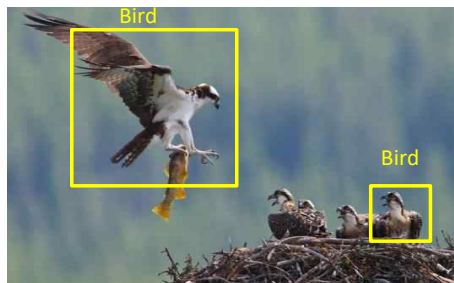
Confidence: 0.6

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

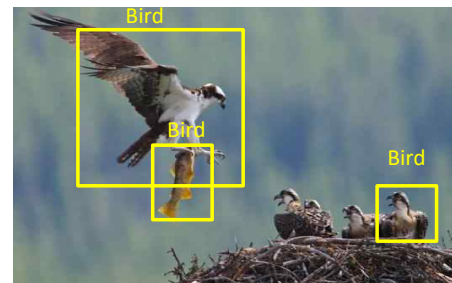
Precision = 1/1 , Recall = 1/5



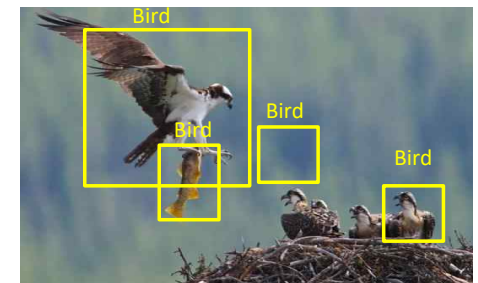
Precision = 2/2 , Recall = 2/5



Precision = 2/3 , Recall = 2/5



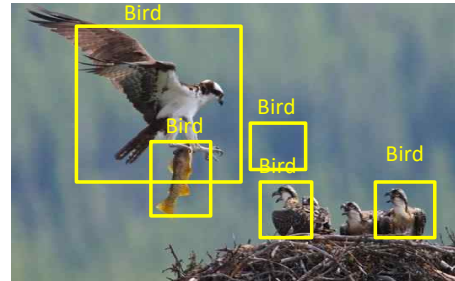
Precision = 2/4 , Recall = 2/5



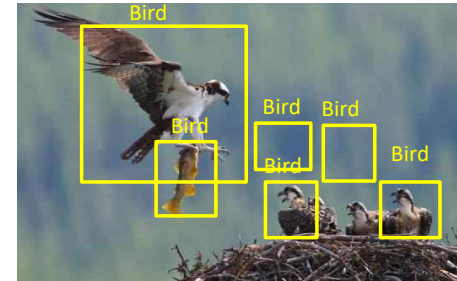
# Confidence에 따른 Precision과 Recall의 변화

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

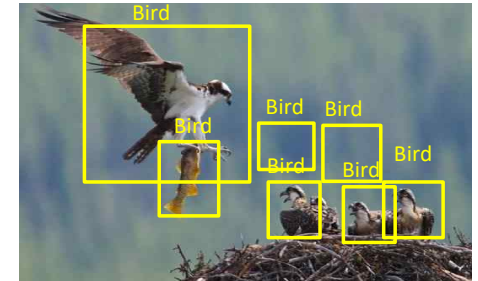
Confidence: 0.5 Precision = 3/5, Recall = 3/5



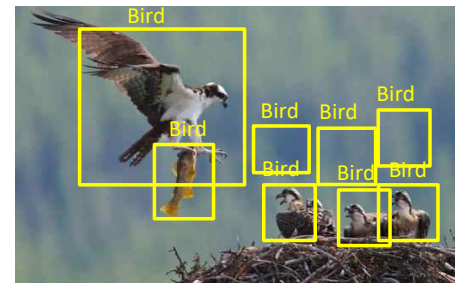
Confidence: 0.4 Precision = 3/6, Recall = 3/5



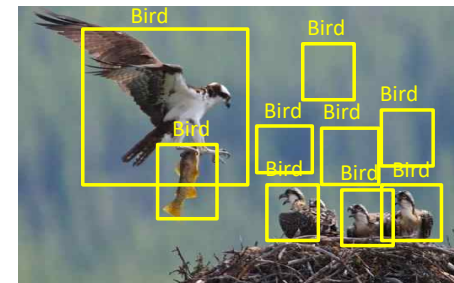
Confidence: 0.3 Precision = 4/7, Recall = 4/5



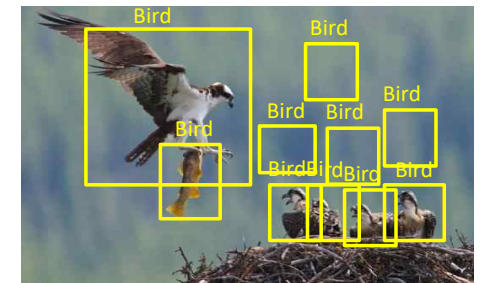
Confidence: 0.2 Precision = 4/8, Recall = 4/5



Confidence: 0.1 Precision = 4/9, Recall = 4/5

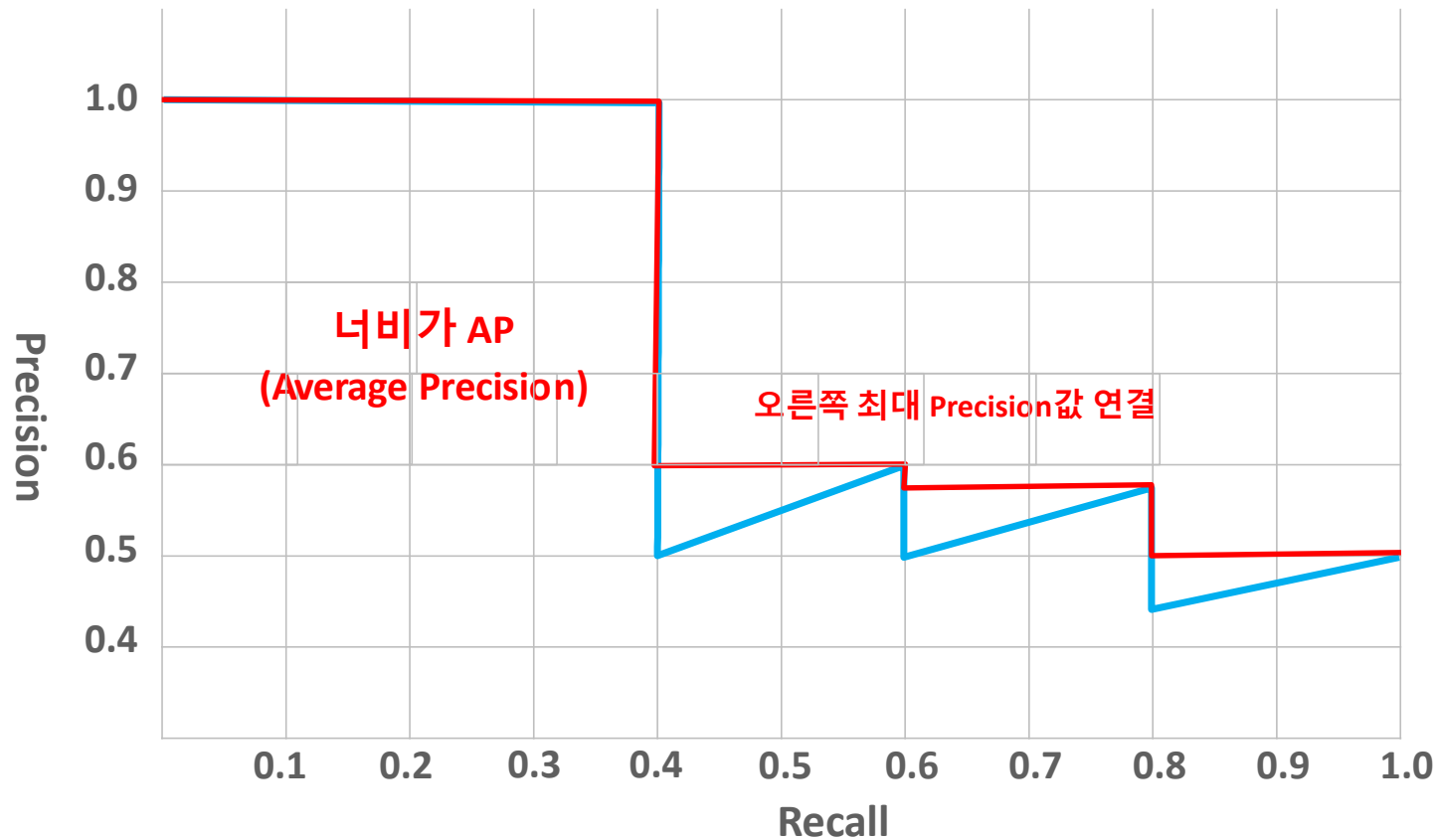


Confidence: 0.05 Precision = 5/10, Recall = 5/5



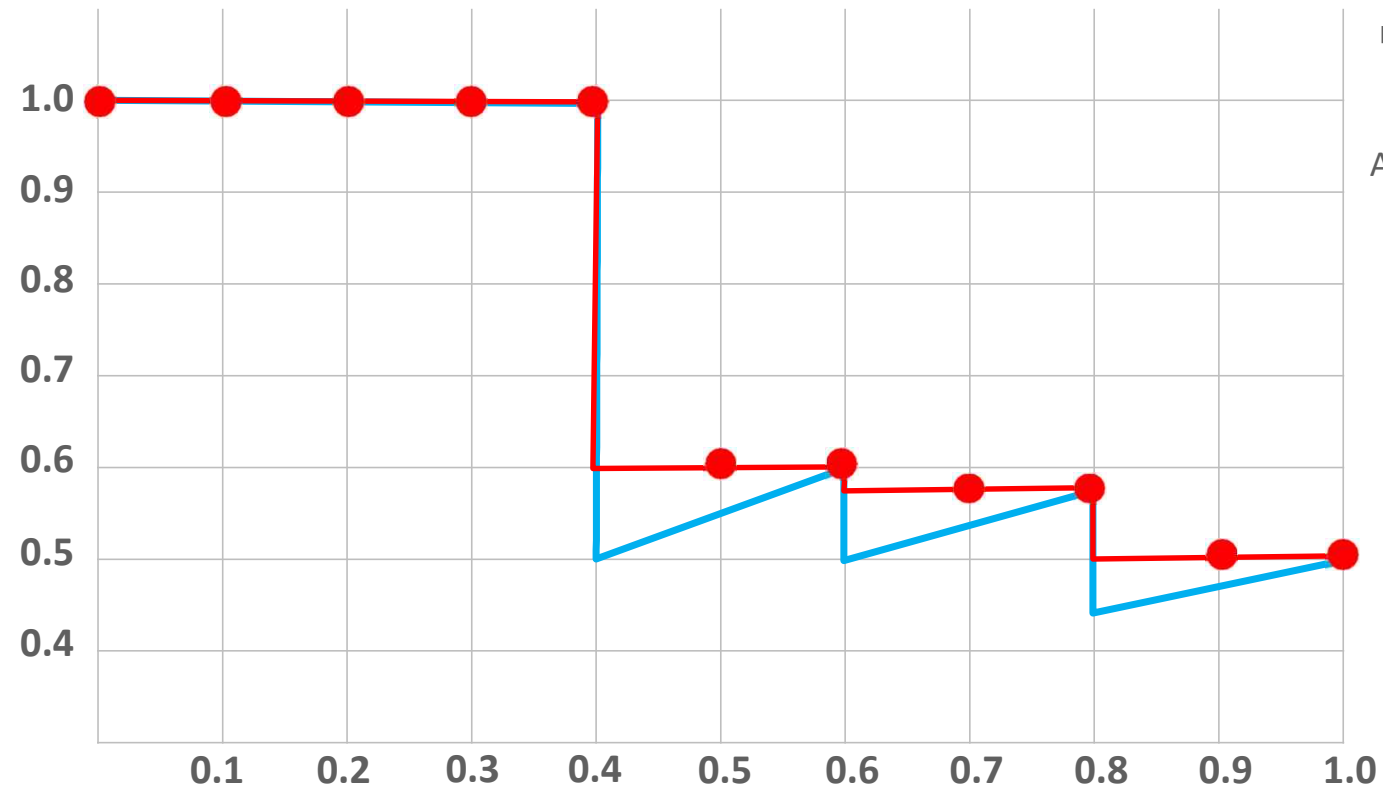


# AP(Average Precision) 계산하기



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

# AP(Average Precision) 계산하기



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

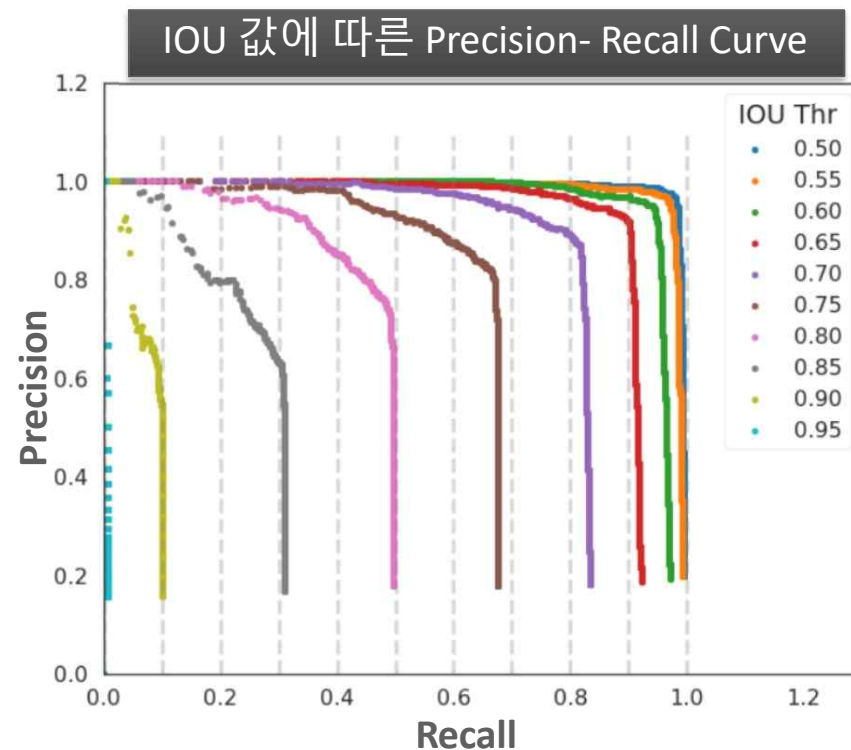
$$\begin{aligned} AP &= \frac{1}{11} \times (mP(r=0) + mP(r=0.1) + \dots + mP(r=1)) \\ &= \frac{1}{11} \times (1.0 + 1.0 + 1.0 + 1.0 + 1.0 + \\ &\quad 0.6 + 0.6 + 0.57 + 0.57 + 0.5 + 0.5) \\ &= \frac{1}{11} \times (5 \times 1.0 + 0.6 \times 2 + 0.57 \times 2 + 0.5 \times 2) \\ &= 0.758 \end{aligned}$$

## mAP(mean Average Precision)

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

# COCO Challenge에서의 mAP

- 예측 성공을 위한 IOU를 0.5 이상으로 고정한 PASCAL VOC와 달리 COCO Challenge는 IOU를 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



# 데이터 세트와 알고리즘 별 mAP 수치 예시

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	<b>78.6</b>	40

VOC 2007 YOLO V2

	backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
<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	<b>52.1</b>
<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	<b>40.8</b>	<b>61.1</b>	<b>44.1</b>	<b>24.1</b>	<b>44.2</b>	51.2

COCO YOLO V2

	backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
<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	<b>52.1</b>
<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	<b>40.8</b>	<b>61.1</b>	<b>44.1</b>	<b>24.1</b>	<b>44.2</b>	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

COCO YOLO V3