

The background features a close-up of a woman's face, looking slightly to the right. Overlaid on this are various digital and technological elements: a grid of binary code (0s and 1s) in a light blue color, and a collage of small, semi-transparent images on the right side. These images include a globe, a city at night, a person working on a laptop, a rocket launch, a satellite, and various abstract digital patterns. The overall color palette is dominated by blues and greys, with some warmer tones from the collage images.

# 딥러닝

## (Deep Learning)

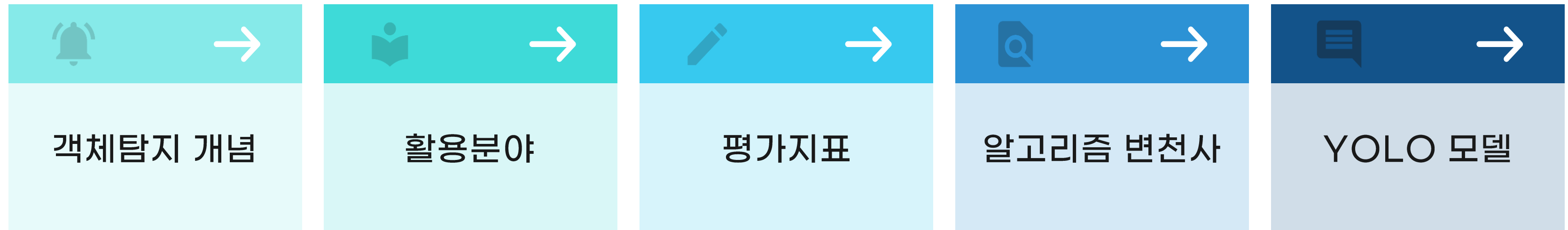
최성우



# 학습목표

- 객체 탐지의 개념을 이해할 수 있다.
- 객체 탐지 모델을 사용할 수 있다.







# 객체 탐지 (Object Detection)

| 이미지 내에 있는 객체를 탐지해보자!



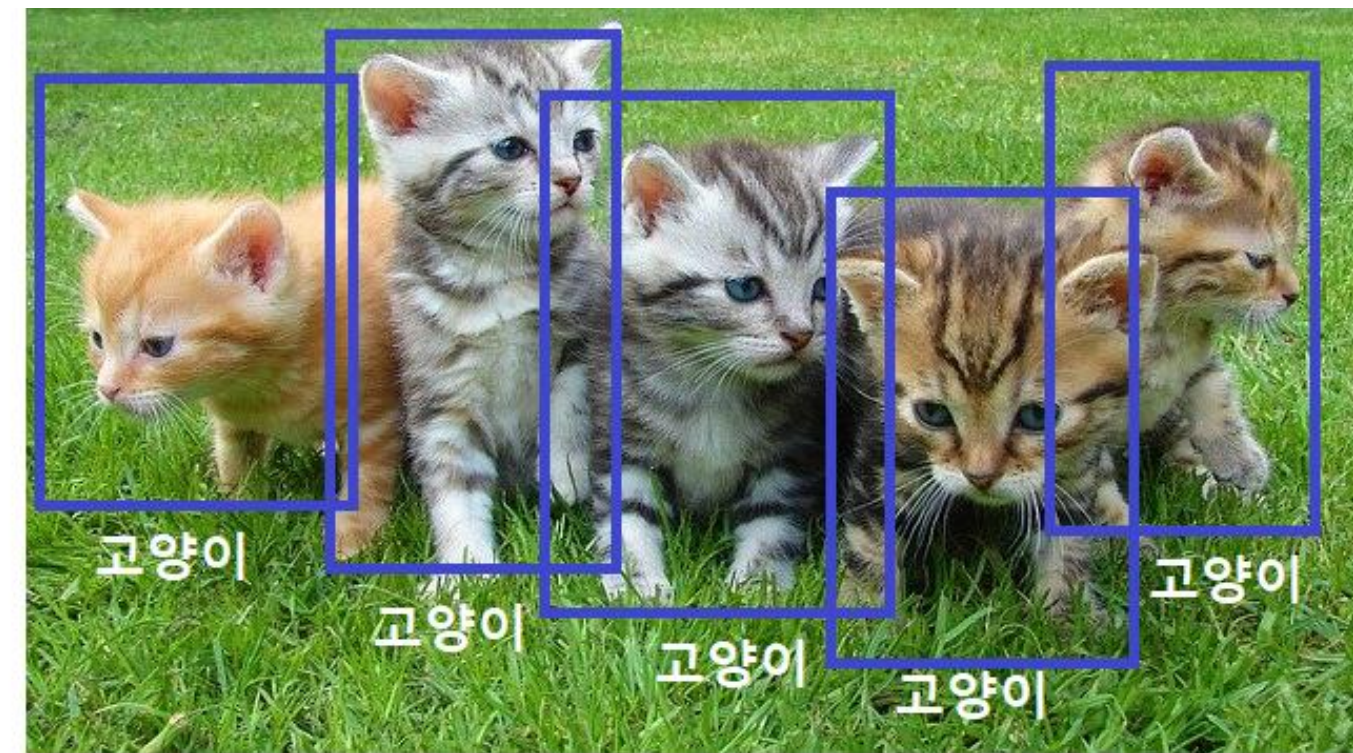


## 객체 탐지란?

- 이미지 내에서 객체(사물, 사람 등 형체가 있는 물체)를 감지해 내는 것



고양이

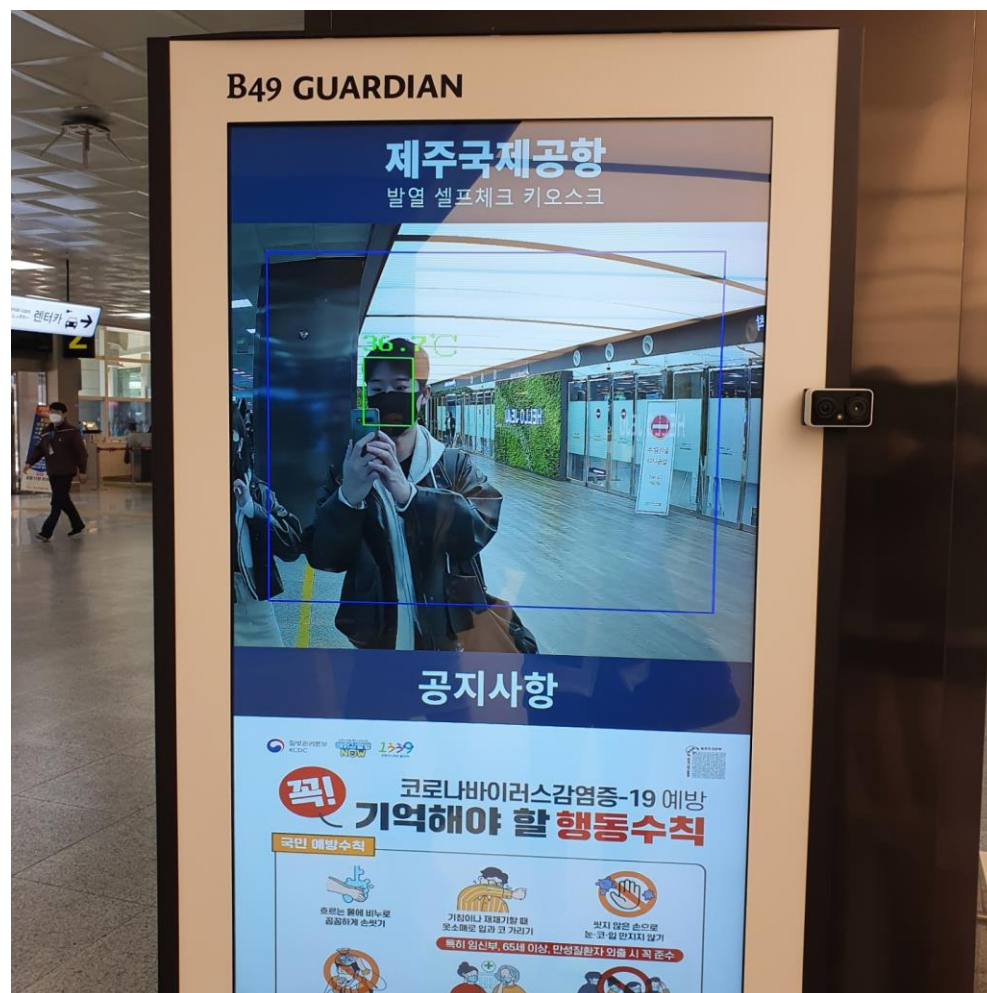


**왼쪽은 이미지 분류(Classification), 오른쪽은 객체 탐지(Object Detection)**

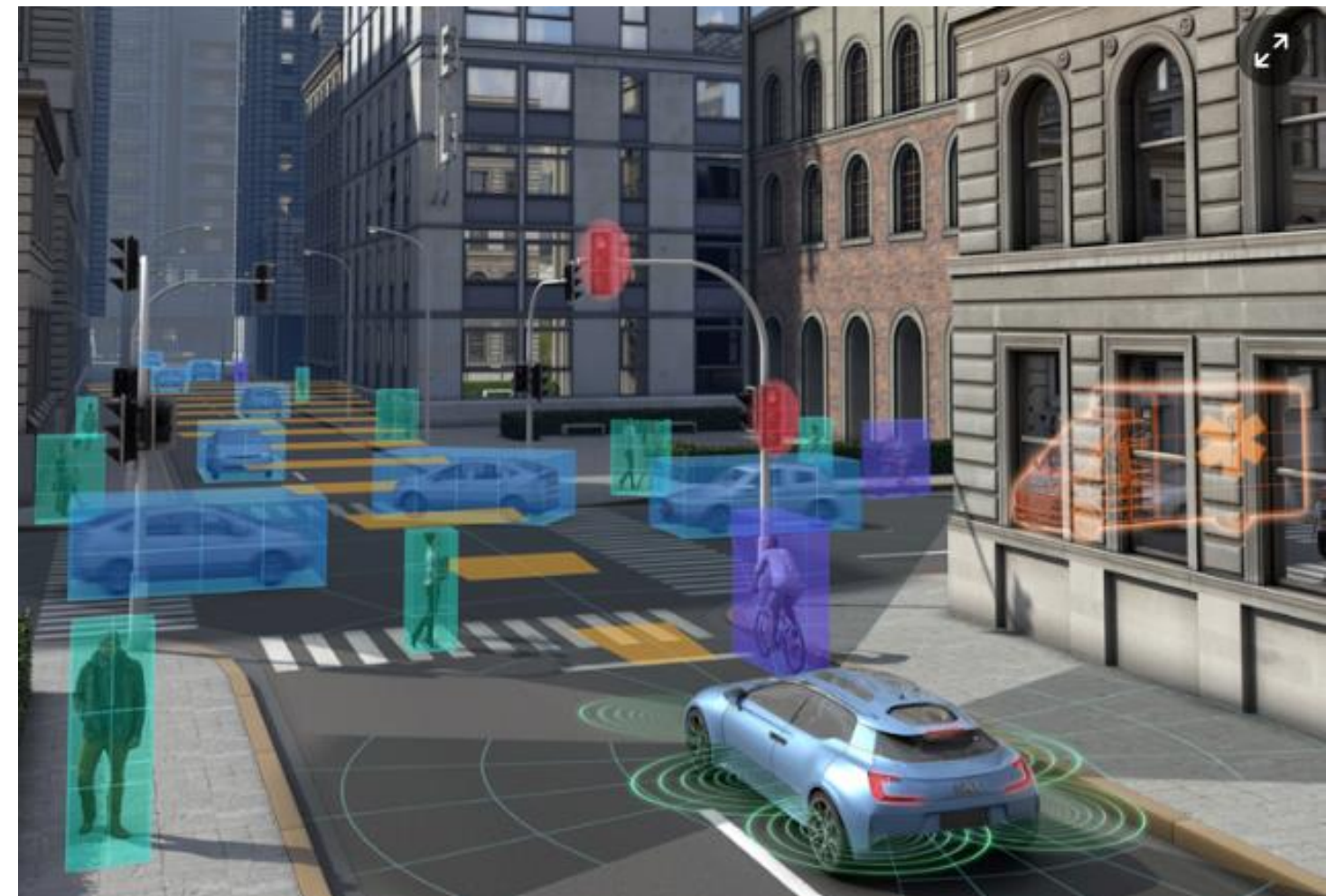


# 객체 탐지(Object Detection)

## 활용 분야



공항 검색대



스마트 시티



# 객체 탐지(Object Detection)

## 활용 분야



CCTV 위급상황 감지

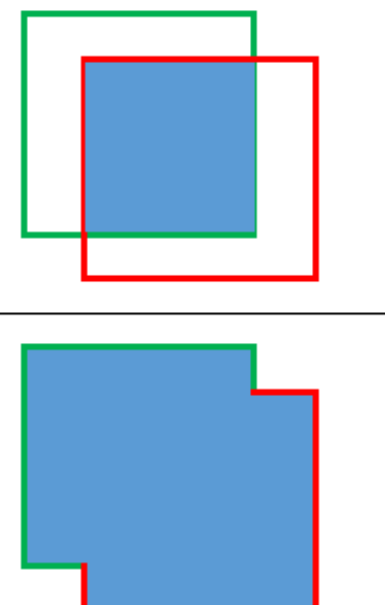


드론을 활용한 교통상황 파악

## 평가 지표

### IoU(Intersection over Union)

- 이미지 내에 있는 하나의 객체를 탐지할 때 사용하는 평가지표
- 실제 객체 면적과 모델이 예측한 면적의 교차영역 / 전체영역
- 범위는 0 ~ 1.0 사이(일반적으로 0.5가 넘으면 맞게 예측했다고 판단)

$$\begin{aligned} B_p &= \text{실제 (Gound Truth)} \\ B_{gt} &= \text{예측 (Prediction)} \\ IOU &= \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \\ &= \frac{\text{실제} \cap \text{예측 중복지역}}{\text{실제} \cup \text{예측 전체 영역}} \end{aligned}$$


waytoliah.com



## 평가 지표

### mAP(mean Average Precision)

- 한 이미지에서 여러 개의 객체(클래스)를 탐지할 때 사용되는 평가지표
- 클래스 별 평균 정밀도(AP)를 계산 후 클래스 전체에 대한 평균(mean)을 구함
- 범위는 0 ~ 1.0 사이, IoU 기준에 따라 mAP는 달라짐(IoU ↑ 라면 mAP ↓)

### ※ AP(Average Precision)

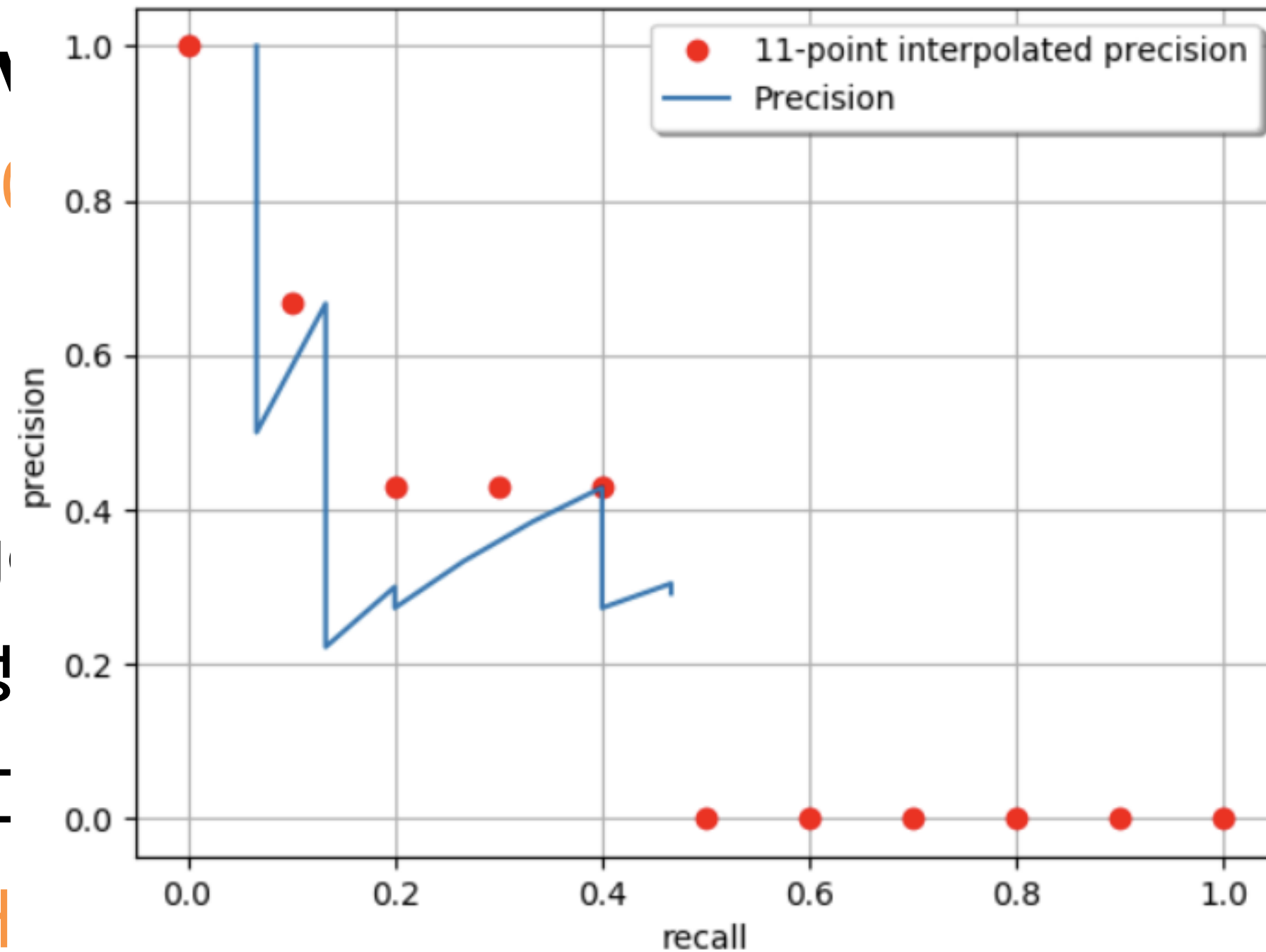
- 일반적으로 정밀도(Precision)와 재현율(Recall)은 반비례 관계
- 두 값을 모두 고려하여 모델의 성능을 판단하는 것이 더 합리적
- 재현율의 11개 지점에 대해 정밀도를 각각 구해서 이를 평균낸 것이 AP



## 평가 지표

### mAP(mean Average Precision)

- 한 이미지에서 (
- 클래스 별 평균
- 범위는 0 ~ 1.0
- ※ AP(Average Precision)
- 일반적으로 정
- 두 값을 모두 :
- 재현율의 11개



≡ 평가지표

평균(mean)을 구함

(↑ 라면 mAP ↓)

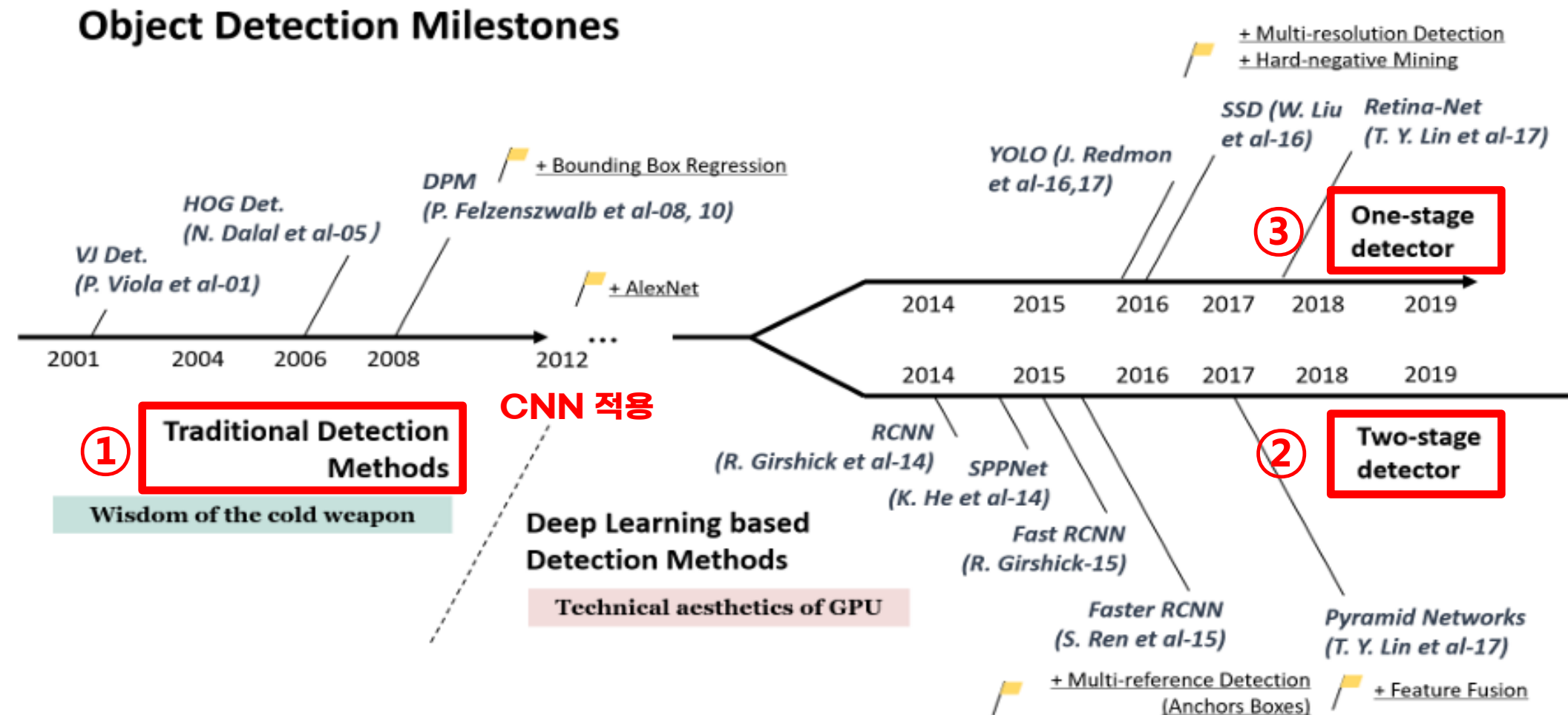
이러 관계

리적

코낸 것이 AP



## 객체 탐지 알고리즘의 변천사



### 발전 순서

1. Traditional Detection Methods
2. Two-stage detector
3. One-stage detector



## 객체 탐지 알고리즘의 변천사

### 1. Traditional Detection Methods



#### 슬라이딩 윈도우(Sliding Window)

- 고정된 크기의 Window로 이미지의 좌상단부터 우하단으로 일일이 객체를 검출해 나가는 방식

#### 문제점

- 객체가 없는 영역도 무조건 Sliding해야 하며  
이미지의 Scale을 조절해서 스캔하며 검출하는  
방식이므로 수행시간이 긴 것에 비해 검출 성능 ↓



## 객체 탐지 알고리즘의 변천사

### 2. Two-stage detector

- 영역추정(Region Proposal)과 탐지(Detection) 두 단계를 따로 수행
- Sliding Window의 비효율성으로 인해 R-CNN 알고리즘에서는 ‘객체가 있을 법한 2000개의 영역’을 찾고 ‘그 영역에 대해서만 객체를 탐지’하는 두 단계를 제안
- R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN 등의 모델이 있음



## 객체 탐지 알고리즘의 변천사

### 영역추정(Region Proposal)의 문제점

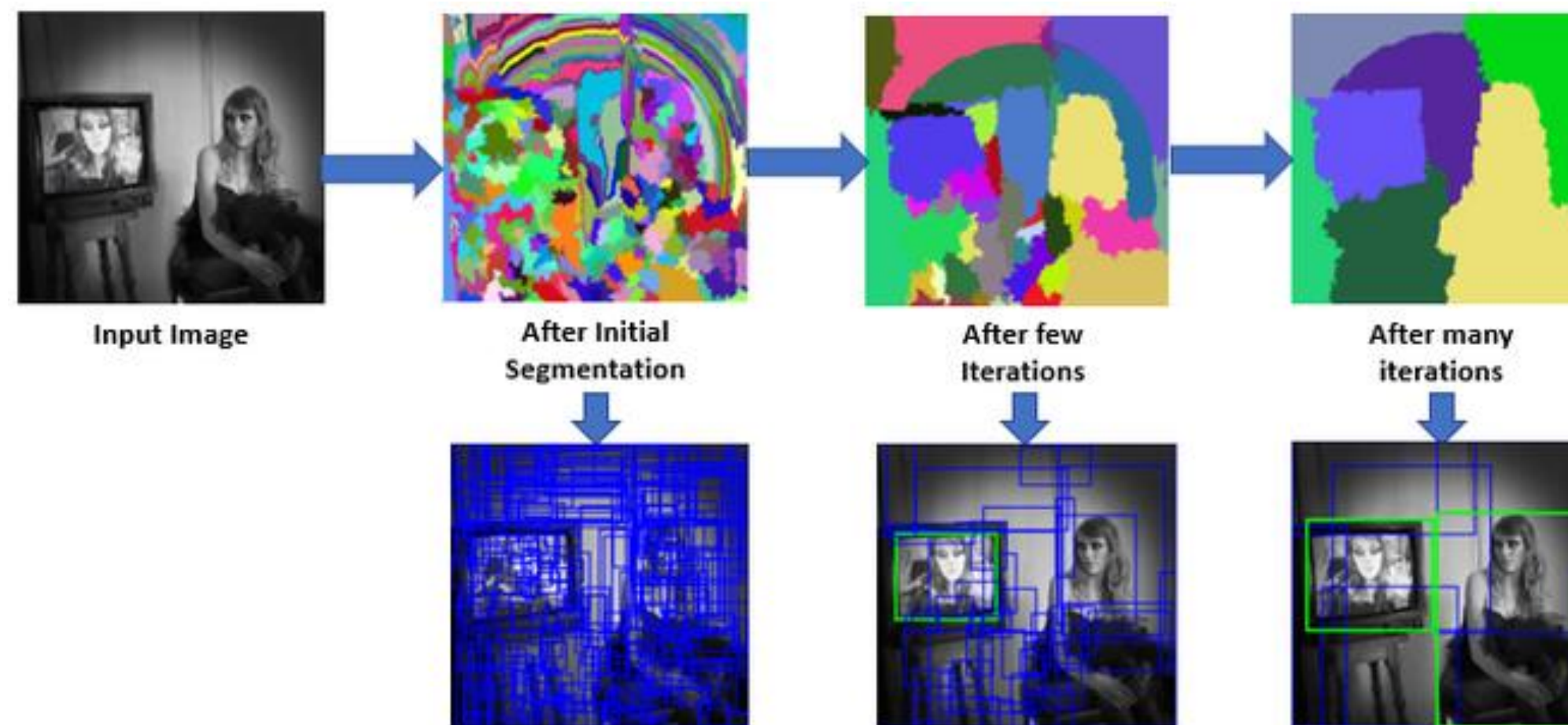
- 객체들이 각기 다른 크기와 형태를 가지고 있다면 후보 영역을 찾는 정확도 ↓
- 영역추정의 정확도를 향상시키기 위해 미리 이미지에서 **객체 영역을 분할**해 둬





## 객체 탐지 알고리즘의 변천사

### 선택적 검색(Selective Search)



- 1) 처음에는 분할된 모든 부분들을 Bounding box로 만들어 리스트에 추가
- 2) 색상, 무늬, 크기, 형태에 따라 유사도가 비슷한 부분들을 그룹핑(Bbox 개수 감소)
- 3) 1, 2 단계 지속 반복



## 객체 탐지 알고리즘의 변천사

### 3. One-stage detector

- Two-stage detector는 Selective search 방식으로 인해 과거 대비 높은 정확도로 객체 탐지가 가능했지만, 여전히 낮은 속도로 실시간 적용은 어려웠음
- One-stage detector는 영역추정과 객체탐지를 통합해 한 번에 수행
- 가장 큰 장점은 탐지 속도의 획기적인 향상으로 실시간 탐지가 가능



# Yolo (You Only Look Once)

| 실시간 객체 탐지 알고리즘의 선봉장!





## Yolo(You Only Look Once)

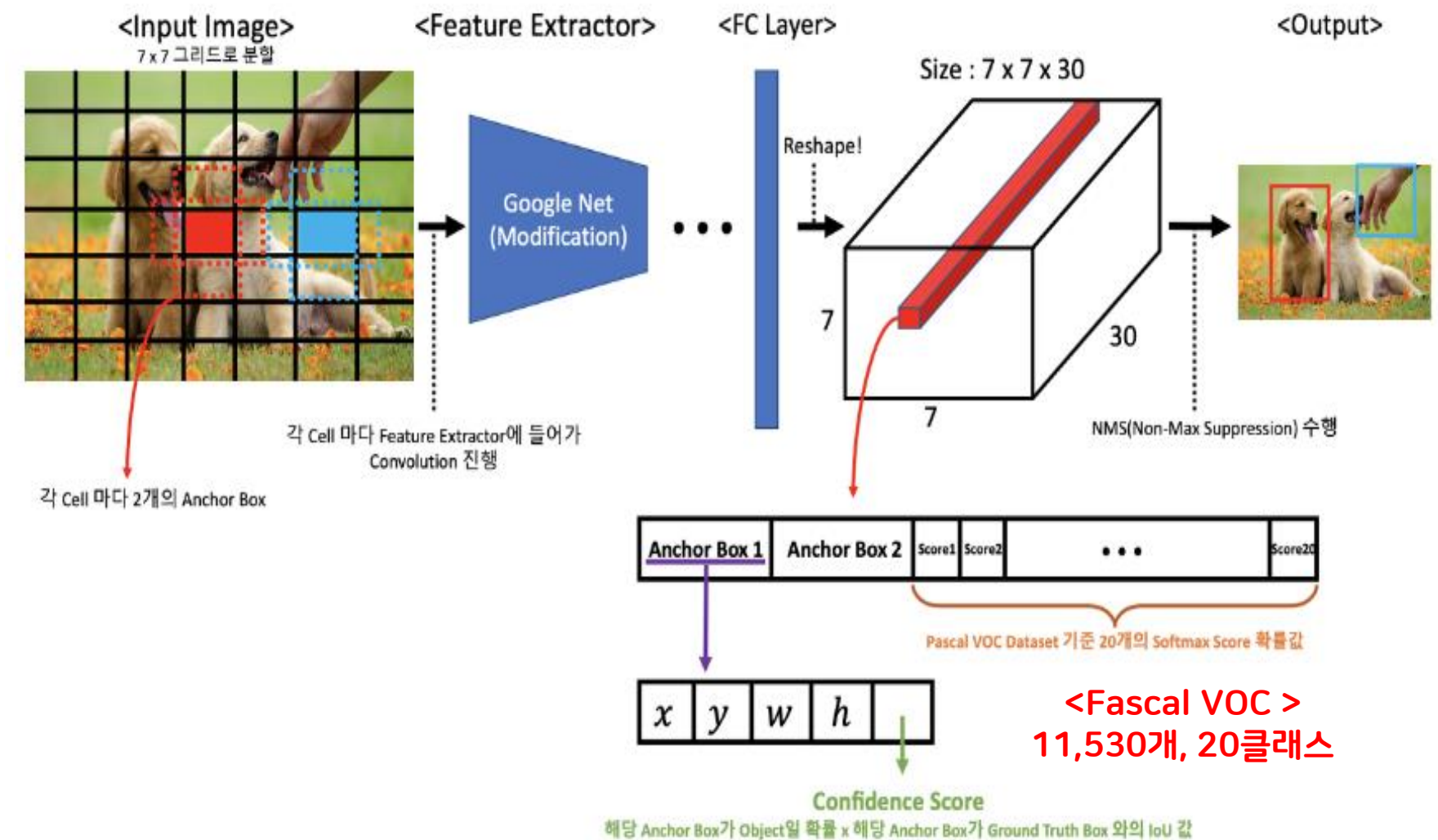
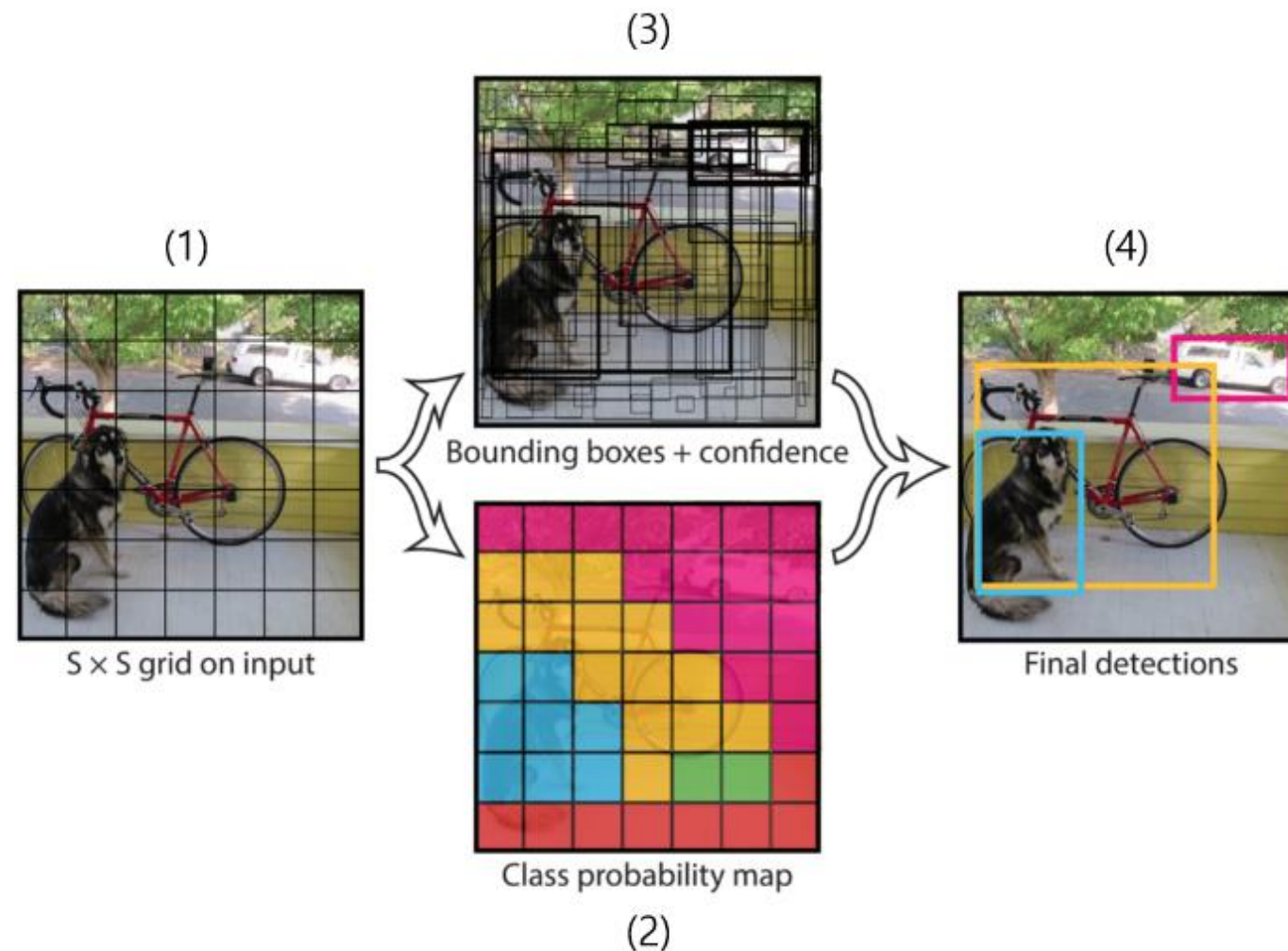
- One-stage detector 방식의 가장 잘 알려진 실시간 객체 탐지 알고리즘
- 2016년 version 1 부터 2023년 version 8까지 오픈소스로 출시됨
- Yolo v1(GoogleNet 적용)은 Two-stage detector의 Faster R-CNN (vgg16 적용)보다 6배 빠른 속도로 논문에 기재됨



# 실시간 객체 탐지(Real time Object Detection)

## Yolo - v1

- 입력 이미지(448 x 448 x 3)를 7 x 7 Grid 영역으로 나눔
- 각 Grid cell당 2개의 랜덤한 Bounding Box를 생성(총 98개)





## Yolo - v1

- 각각의 Bbox는  $x, y, w, h$  와 Confidence Score로 구성됨
- $x, y$ 는 Bbox의 중심점,  $w, h$  는 너비, 높이
  - ex)  $x$ 가 cell의 가장 왼쪽에 있다면 0이고  $y$ 가 cell의 중간에 있다면 0.5
  - ex) Bbox의  $w$ 가 이미지 전체 너비의 절반이라면  $w$ 는 0.5
- Confidence Score는 Bbox가 객체를 포함한다는 예측 확신 정도의 지표
  - ※  $P_r$  : Grid cell 내에 물체가 존재할 확률 (존재하면 1, 아니면 0)

$$\text{Confidence Score} : P_r(\text{Object}) * IOU_{pred}^{truth}$$

- Confidence Score가 0.5 이하인 Bbox는 모두 삭제 (기준은 사용자 지정 가능)
- 값이 가장 높은 Bbox만 남기고 나머지는 삭제하여 한 객체당 하나의 Bbox만 남김

## Yolo - v2, v3

### v2

- Bbox의 개수를 늘리고 GoogelNet 대신 Darknet-19 모델을 사용하여 v1에 비해 **mAP 향상**
- Multi-Scaling기법을 사용하여 v1의 문제점인 **작은 객체에 대한 인식률 향상**

### v3

- Darknet-53 모델로 변경하고 내부 구조를 조정하여 **FPS를 2배 이상 향상**
- 특성맵의 크기를 조절하여 크기가 **큰 객체의 검출 성능 향상**
- 다수의 객체 예측시 softmax 대신 클래스 별 **sigmoid를 활용**하여 검출  
(하나의 Bbox안에 복수의 객체가 존재하는 경우 softmax성능 ↓)



## Yolo - v4 ~ v6

### v4

- CSPDarknet53, SPP, PAN, BoF, Bos 등의 기법을 통해 v3에 비해 mAP, FPS를 각각 10%, 12%씩 향상

### v5 ~ v6

- 논문 없이 깃 허브로 코드만 공유, Darknet 대신 Pytorch로 구현
- v4에 비해 낮은 용량, 빠른 속도(높은 FPS), 비슷한 성능(mAP)
- 검출되는 객체의 크기 별 전용 버전인 s, m, l, x로 버전 세분화
- Pascal VOC 데이터 대신 COCO 데이터 셋(20만개, 클래스 80개)으로 훈련

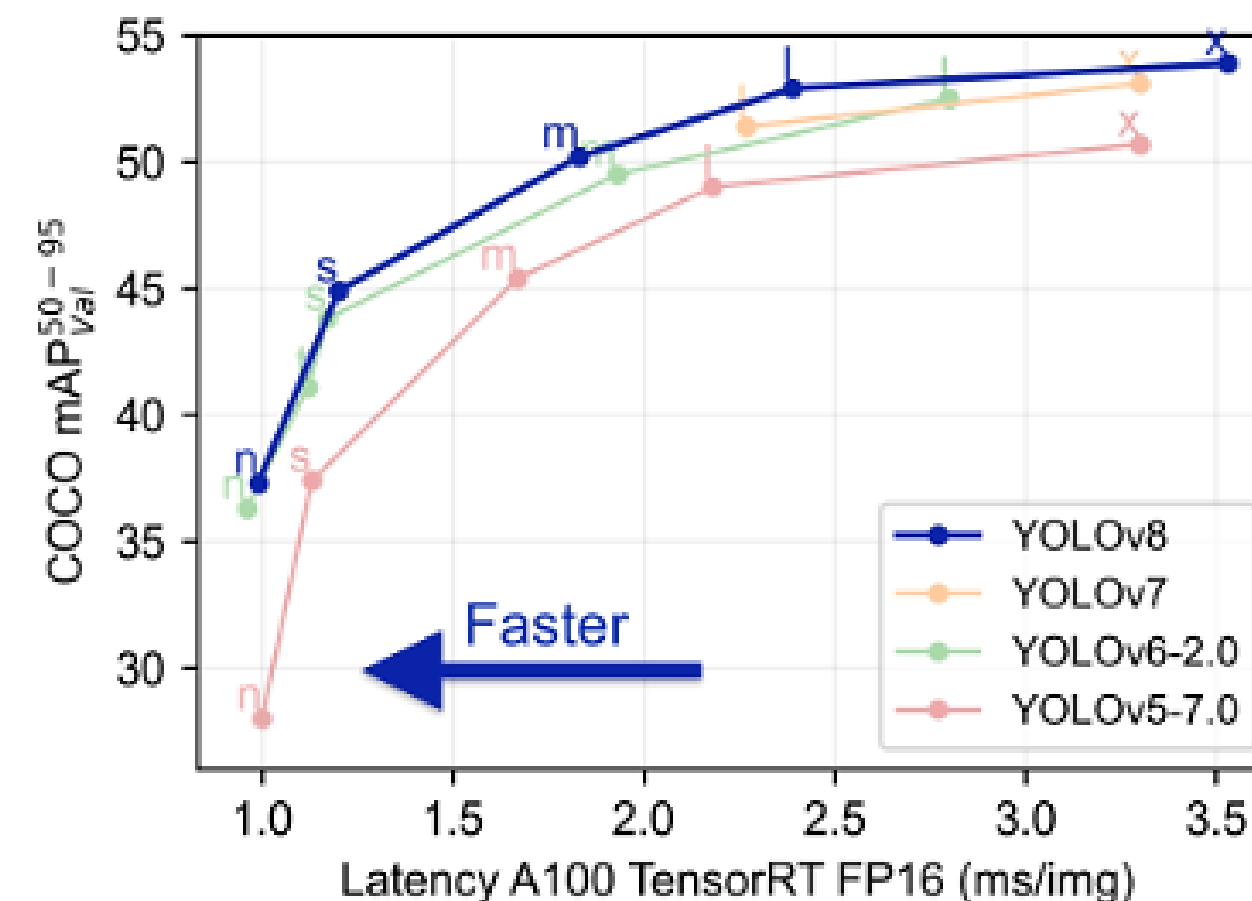
## Yolo - v7, v8

### v7

- COCO 데이터셋 기준 AP 56.8%로 7까지의 Yolo 버전 중 **가장 성능이 높음**
- 인간의 포즈를 추정할 수 있는 **포즈추정 모델** 포함(Yolo에서 첫 등장)

### v8

- 내부 구조를 변경하여 이전버전에 비해 **평균적으로 mAP가 높음**
- 객체 탐지, 인스턴스 세분화, 이미지 분류를 위한 통합 프레임워크로 구축





## 권총 객체 탐지 Yolo v5 실습





# 다음시간에

복습 없이 전부를 이해하려는 것은 정신병 초기 증상이다