

# YOLO\_v3

---

# LOW-CODE

교육

수도권 ICT 이노베이션 스퀘어

성명

류재영

# CONTENTS



01 YOLOv3 Architecture

02 DarkNet-53

03 FPN

04 Loss Function

05 DataSet

06 Anchor Box

07 Training and Evaluation

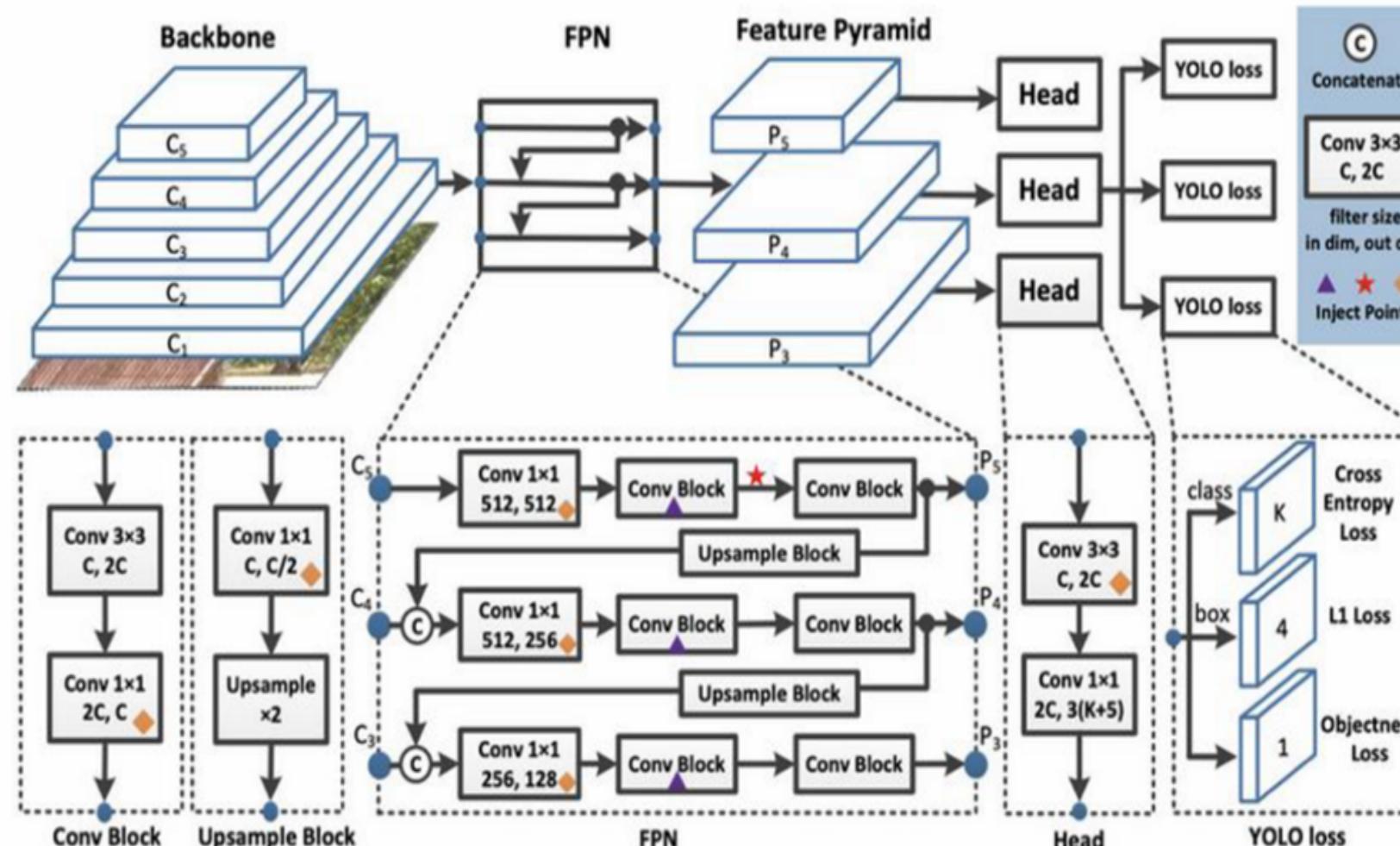
08 NMS and Result



01

# YOLOv3 Architecture

# YOLOv3 Architecture



- **Backbone**

주로 특징 추출을 담당하는 기본 신경망 구조로,  
입력 이미지에서 유용한 특징 맵을 생성

- **FPN**

여러 해상도의 특징 맵을 결합하여 다양한 크기의  
객체를 효과적으로 탐지

- **Head**

추출된 특징 맵을 바탕으로 최종적으로 객체의  
위치와 클래스 등을 예측

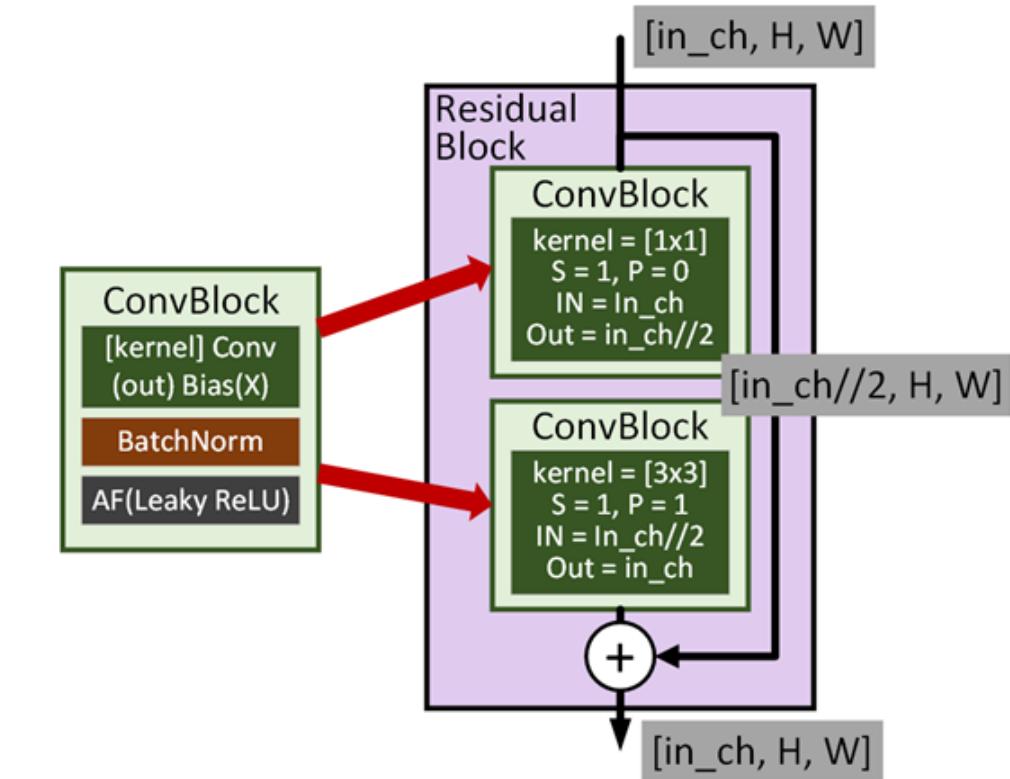
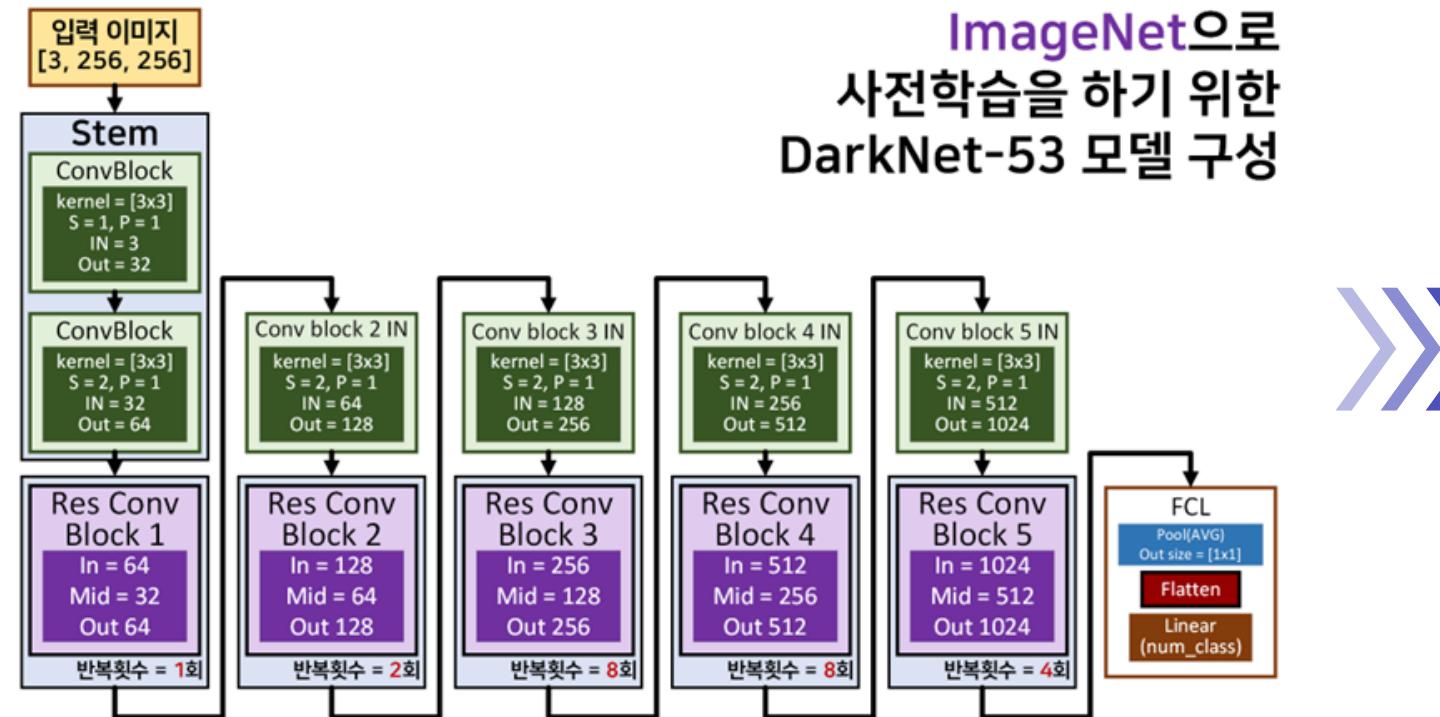
- **YOLO loss**

모델의 예측값과 실제 값 간의 차이를 계산하여,  
모델이 학습할 수 있도록 하는 지표

02

## DarkNet-53

# DarkNet-53 구조



DarkNet-53은 ImageNet 데이터셋으로 사전학습이 수행되었으며, 이때 이미지의 전처리는  $[3 \times 256 \times 256]$ 으로 수행

ResNet과 유사한 residual connections를 사용해 더 깊은 네트워크에서 발생할 수 있는 기울기 소실 문제를 해결

# Backbone 성능 비교

DarkNet-53이 속도와 연산량 측면에서 ResNet-152보다 1.5배 정도 좋은 성능을 보이며 정확도도 비슷하고 속도는 2배정도 빠름

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

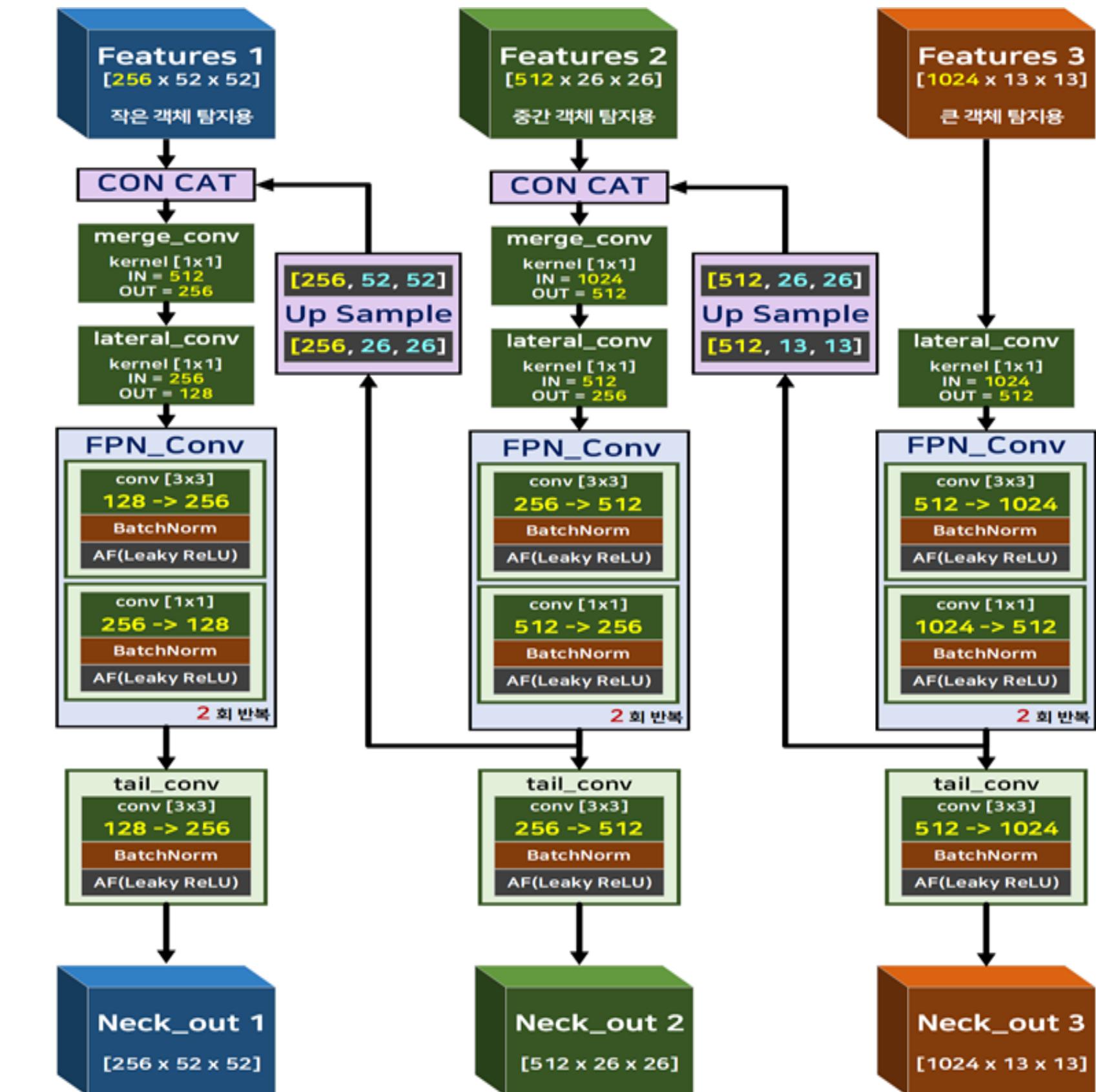
\*BFLOP/s는 1초당 가능한 연산량을 의미

# 03

# FPN

# FPN 이란?

YOLOv3는 FPN 구조를 사용하여 서로 다른 해상도의 특징 맵을 결합합니다. 이는 하위 레이어에서 추출한 저해상도 특징 맵과 상위 레이어에서 추출한 고해상도 특징 맵을 통합하는 과정입니다. 이렇게 결합된 특징은 다양한 크기의 객체를 더 잘 인식할 수 있게 해줍니다.





# 04

## Loss Function

# Loss Function 의 종류

## Localization Loss

Pred B\_Box와  
GT B\_Box가  
얼마나 일치하는지?

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

## Confidence Loss

B\_Box의 마지막  
인자값인 CS  
[Confidence Score]  
으로 계산하는 손실값

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

## Classification loss

인식한 객체가  
라벨 객체 종류랑  
같은지? (Softmax)

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

# Loss 수식 v1과 v3 비교

## Yolo v1 Loss

### Localization Loss

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

### Objectness Loss

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

### Classification Loss

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



## Yolo v3 Loss

### Localization Loss

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

### Objectness Loss

$$\text{BCE} = - \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

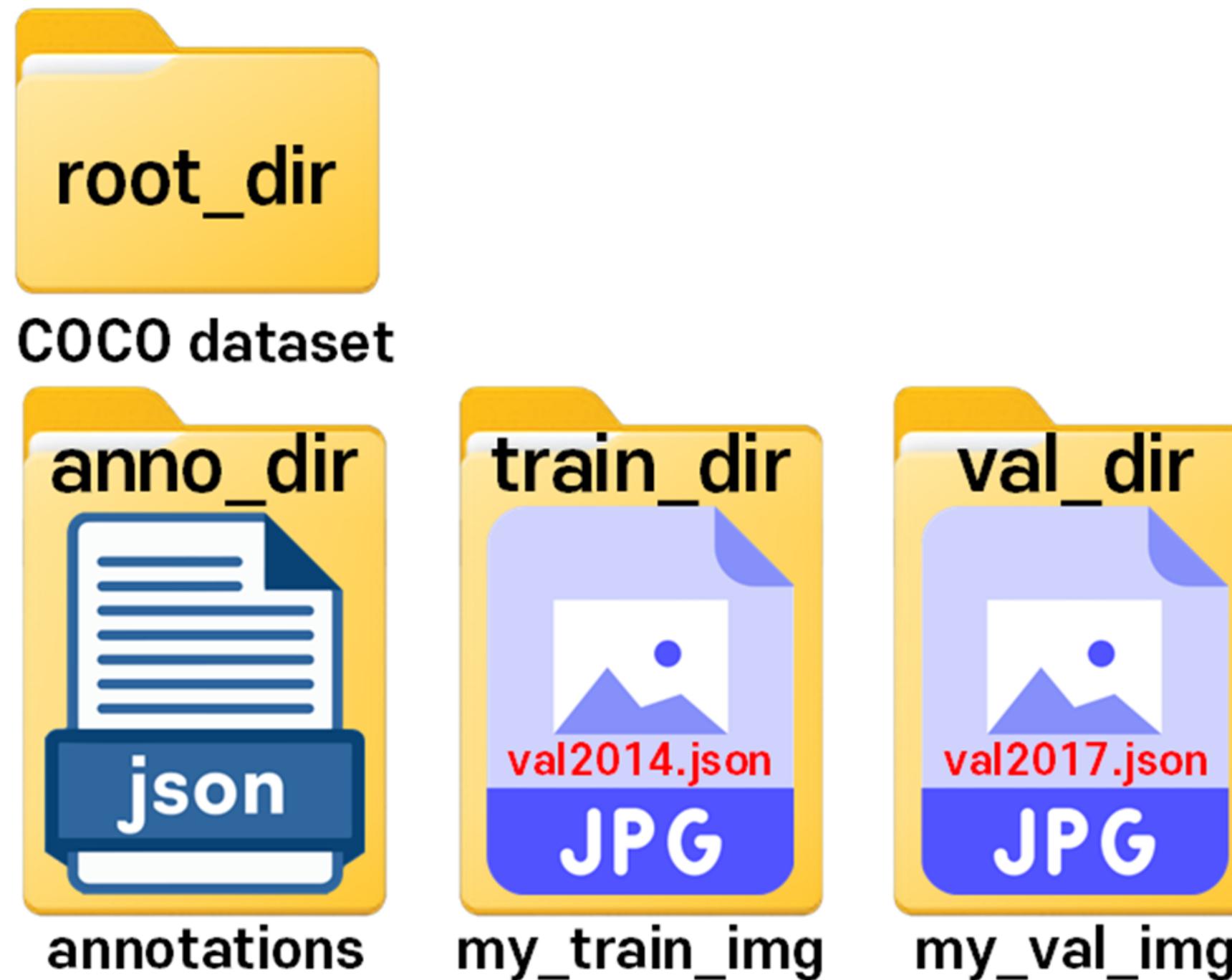
### Classification Loss

$$\text{BCE} = - \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

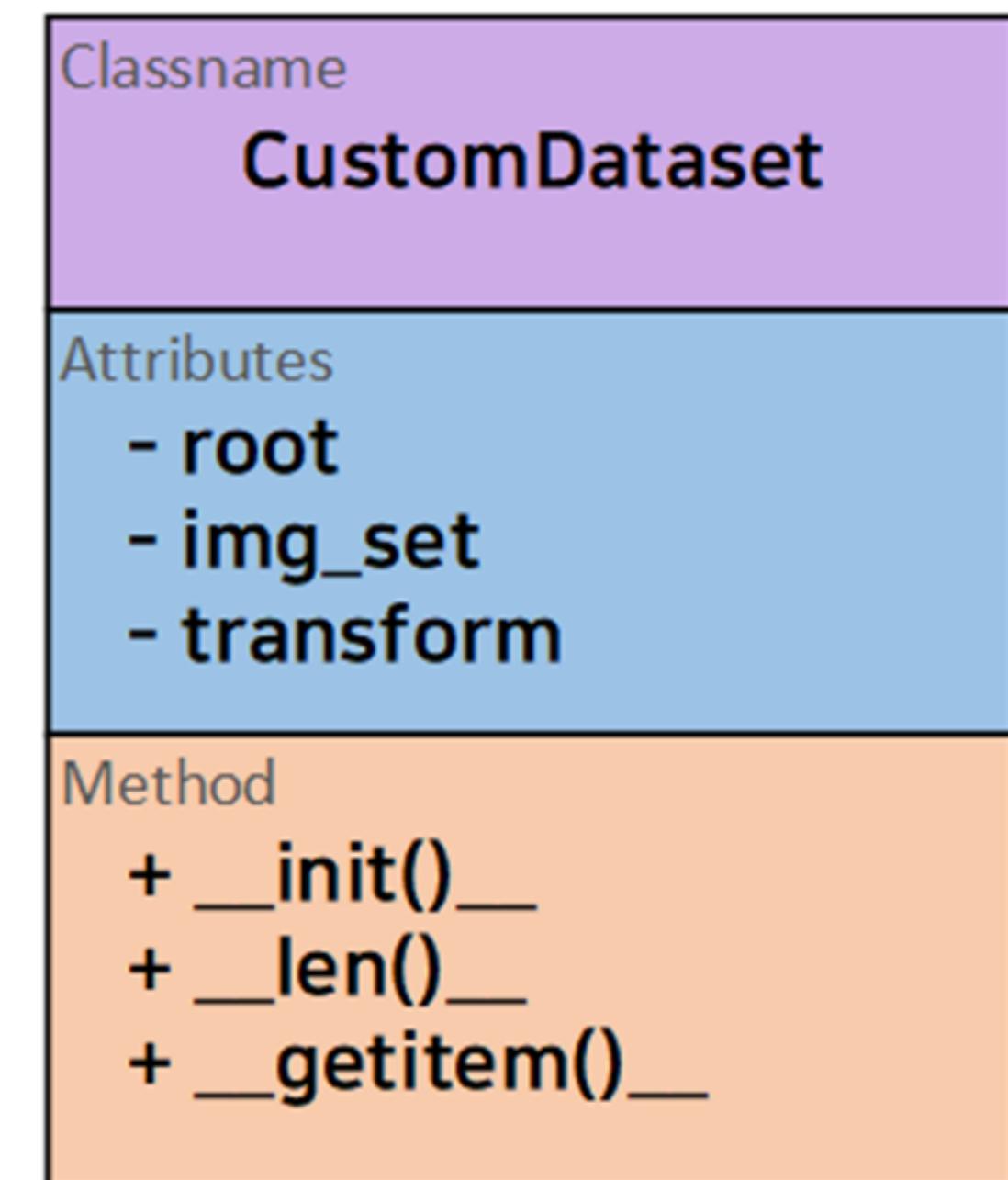
05

## coco DataSet

## Data Path



## Data Class

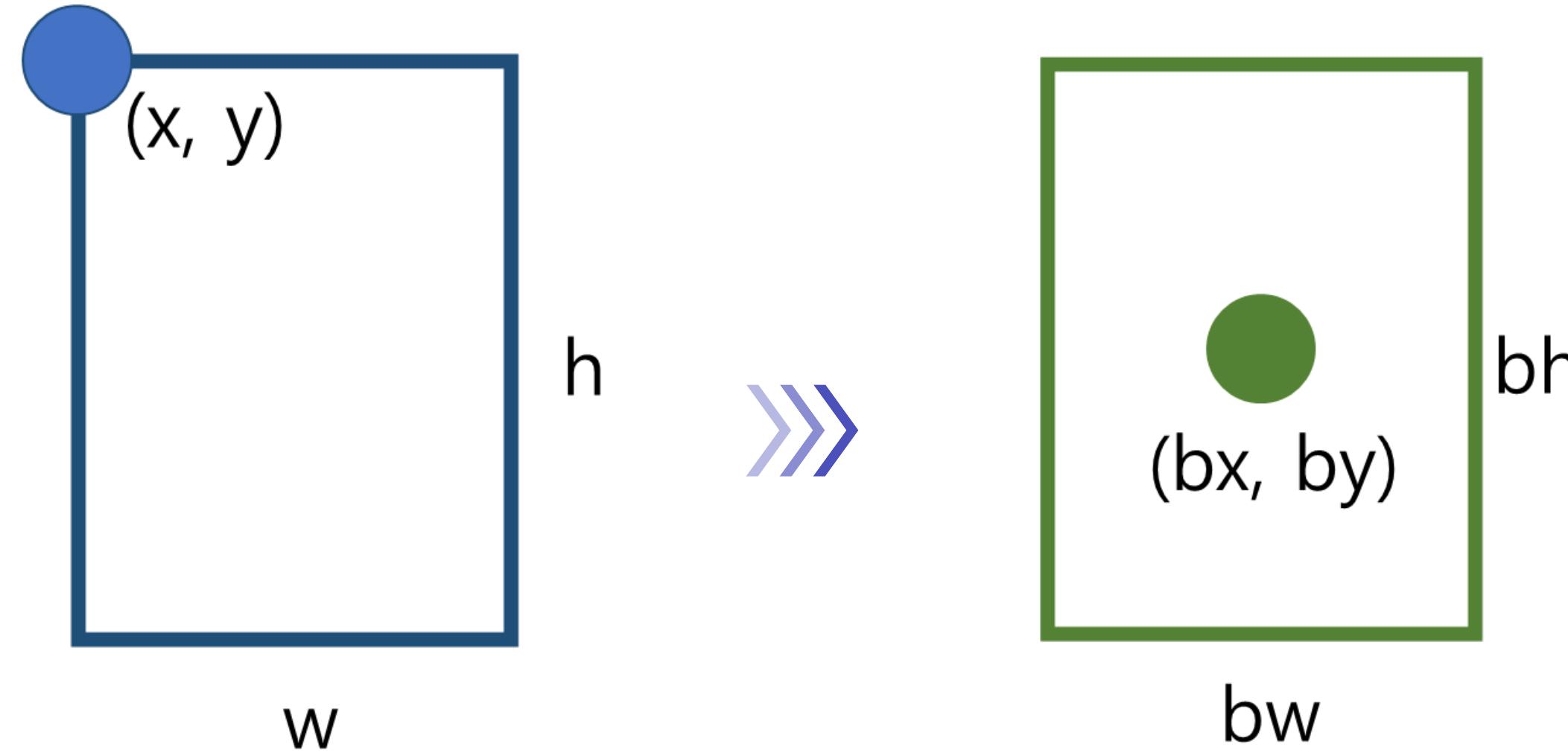


# Data 클래스 필터링

COCO dataset category_id 구성(붉은색 : 사용하지 않는 ID)									
1	person	2	bicycle	3	car	4	motorcycle	5	airplane
6	bus	7	train	8	truck	9	boat	10	traffic light
11	fire hydrant	12	street sign	13	stop sign	14	parking meter	15	bench
16	bird	17	cat	18	dog	19	horse	20	sheep
21	cow	22	elephant	23	bear	24	zebra	25	giraffe
26	hat	27	backpack	28	umbrella	29	shoe	30	eye glasses
31	handbag	32	tie	33	suitcase	34	frisbee	35	skis
36	snowboard	37	sports ball	38	kite	39	baseball bat	40	baseball glove
41	skateboard	42	surfboard	43	tennis racket	44	bottle	45	plate
46	wine glass	47	cup	48	fork	49	knife	50	spoon
51	bowl	52	banana	53	apple	54	sandwich	55	orange
56	broccoli	57	carrot	58	hot dog	59	pizza	60	donut
61	cake	62	chair	63	couch	64	potted plant	65	bed
66	mirror	67	dining table	68	window	69	desk	70	toilet
71	door	72	tv	73	laptop	74	mouse	75	remote
76	keyboard	77	cell phone	78	microwave	79	oven	80	toaster
81	sink	82	refrigerator	83	blender	84	book	85	clock
86	vase	87	scissors	88	teddy bear	89	hair drier	90	toothbrush

논문에서 COCO 데이터는 80개의 클래스라고 했지만 실제로 90개이다  
 이중 10개는 없는 내용이므로 클래스 필터링과 라벨 매핑이 필요

## BBOX 좌표 변환



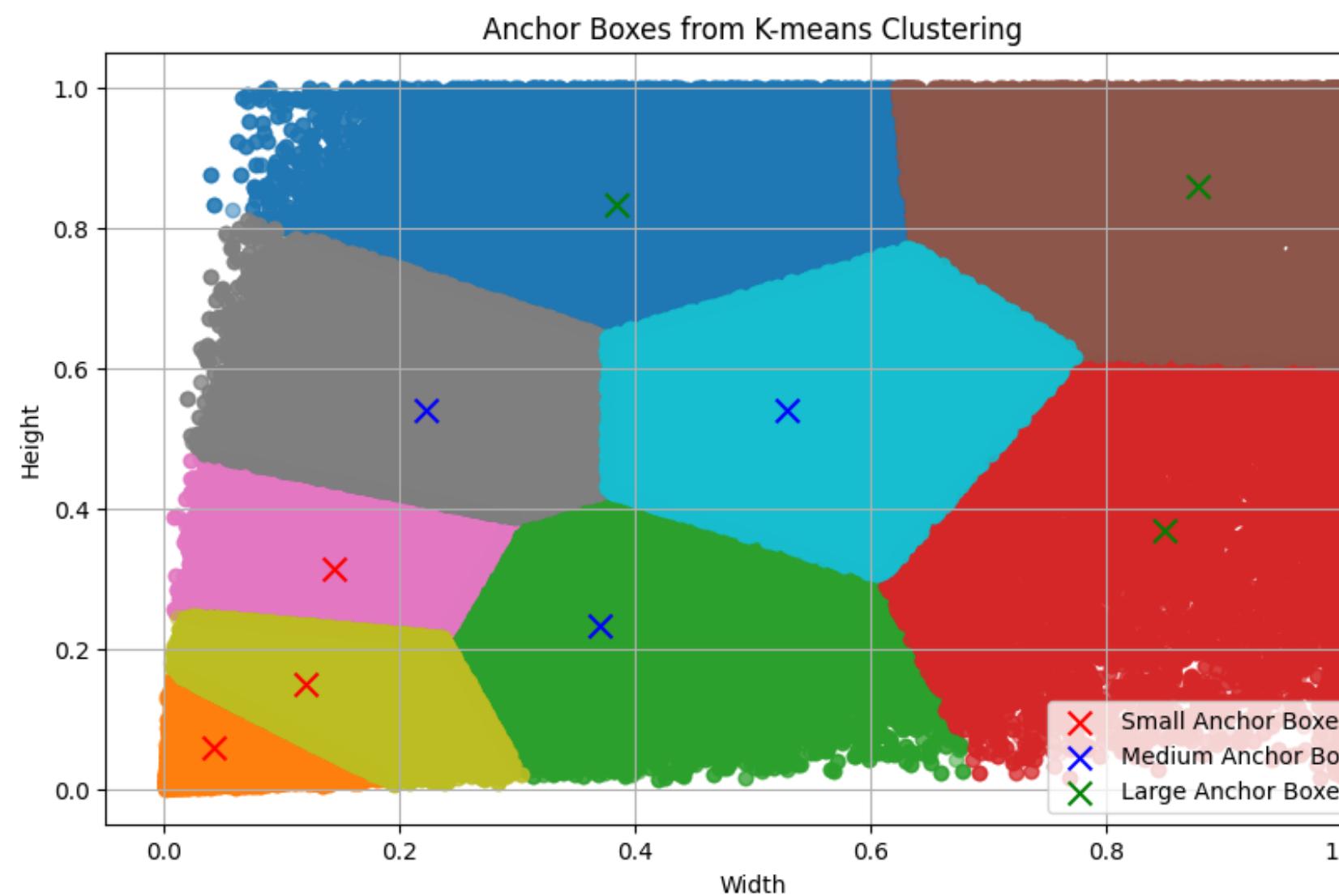
COCO 데이터셋 Annotation  
BBOX 좌표 구성  
[ $x, y, w, h$ ]

율로 모델 입력  
BBOX 좌표 구성  
[ $bx, by, bw, bh$ ]

06

## Anchor Box

# K-means Clustering 그레프



- **클러스터의 수**

데이터를 9개의 클러스터로 분할하여 3개의 그룹으로  
3개씩 묶음

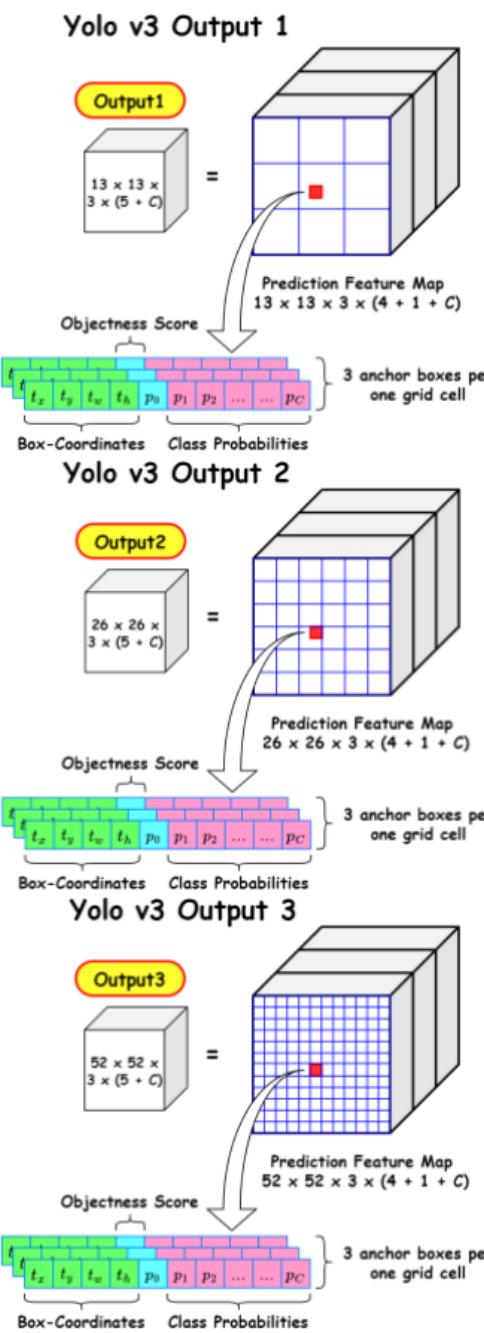
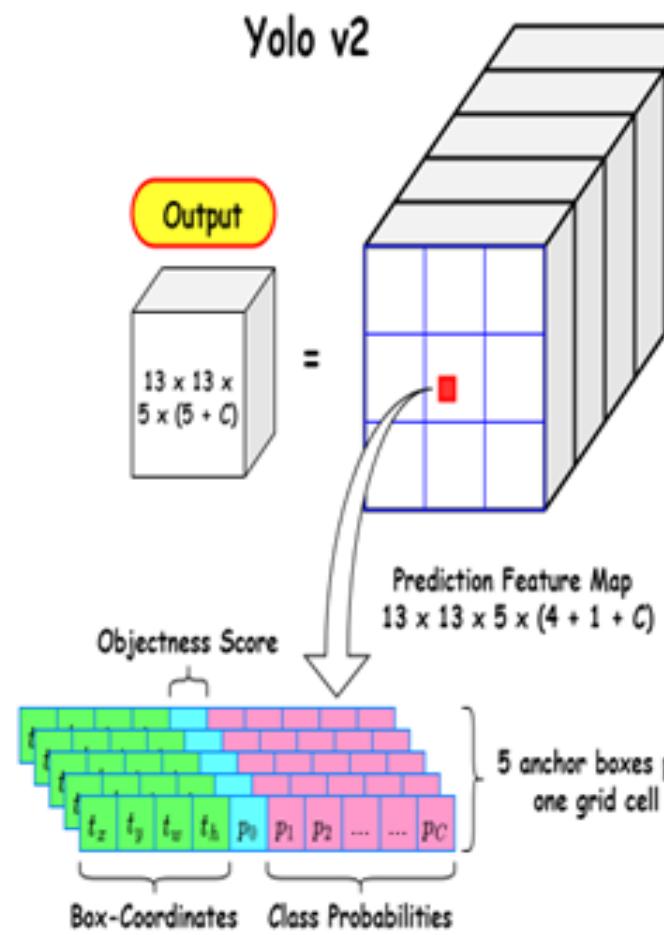
- **클러스터 중심**

각 클러스터의 중심값은 앵커박스들의 크기

## Value

```
array([[[0.0430459 , 0.05774691],  
       [0.12087719, 0.1486785 ],  
       [0.14489405, 0.31324601]],  
  
      [[0.37007827, 0.23355614],  
       [0.22235841, 0.5389649 ],  
       [0.52990178, 0.53865267]],  
  
      [[0.84938177, 0.36934118],  
       [0.38464032, 0.83243351],  
       [0.87748422, 0.85986112]]])
```

# Anchor Box v2과 v3 비교



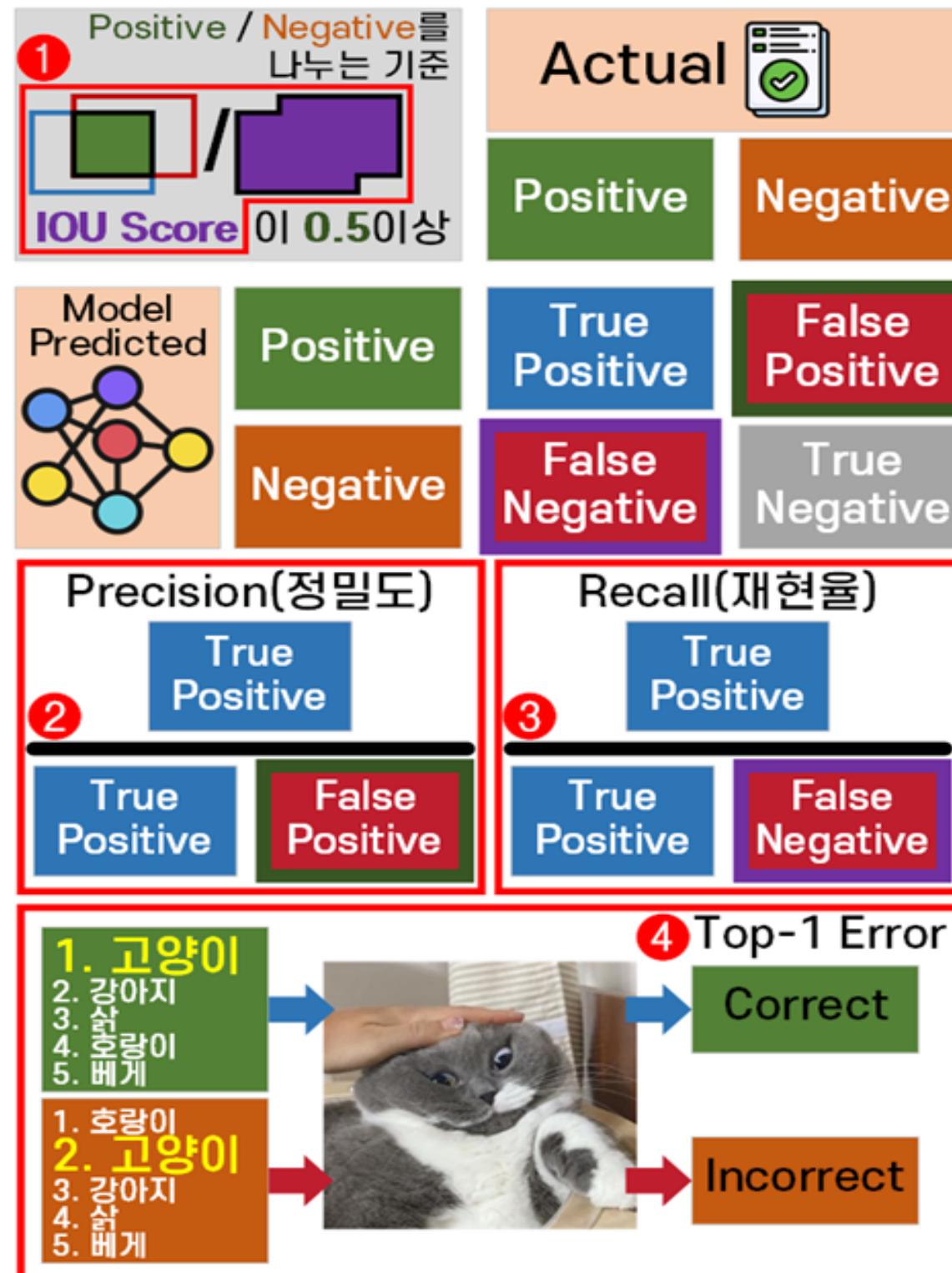
v2에서는 K-Means Clustering의 결과를 통해  
5개의 앵커 박스를 사용

v3에서는 K-Means Clustering의 결과를 통해  
각 그리드 셀(13,26,52)에서 3개의 앵커 박스를  
생성하여 총 9개의 앵커 박스를 사용

07

# Training & Evaluation

# Evaluation 종류



- IoU Score

경계 상자의 교집합과 합집합 비율

- Precision

양성 예측 중 실제 양성 비율

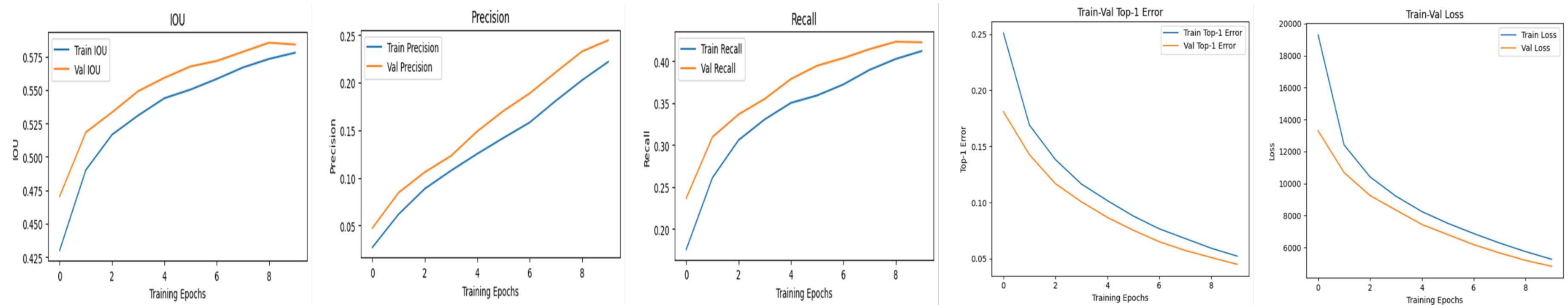
- Recall

실제 양성 중 올바른 양성 예측 비율

- TOP-1 Error

가장 높은 예측이 정답이 아닐 확률

# Evaluation 그래프



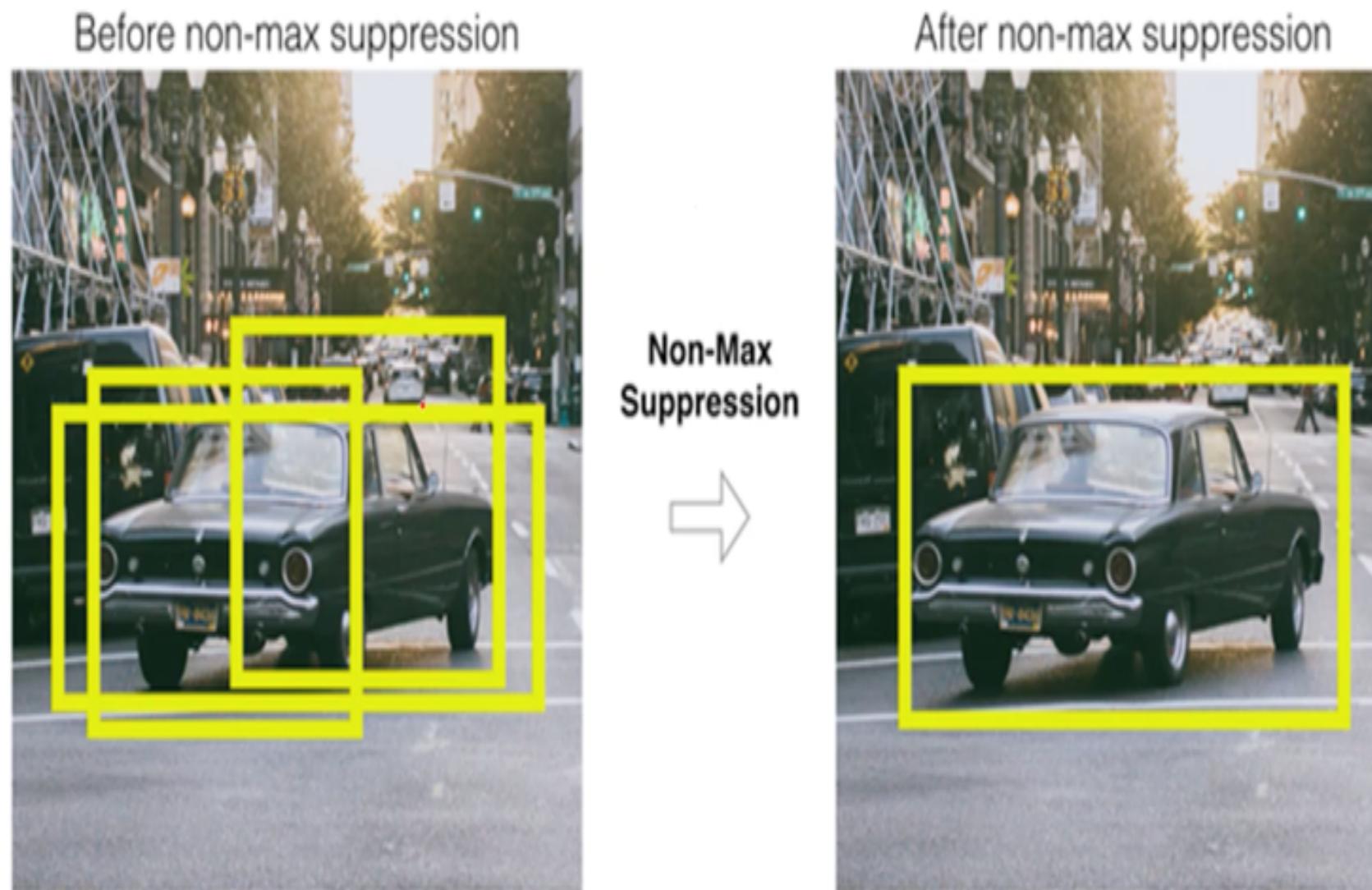
\*자원 부족으로 인해 팀원의 자료를 인용함



08

## NMS & Result

# NMS 기본 원리



- **예측 결과 정렬**

모델이 예측한 경계 상자를 신뢰도 값에 따라 내림차순으로 정렬

- **최고 점수 상자 선택**

가장 높은 점수를 가진 경계 상자를 선택

- **겹치는 상자 제거**

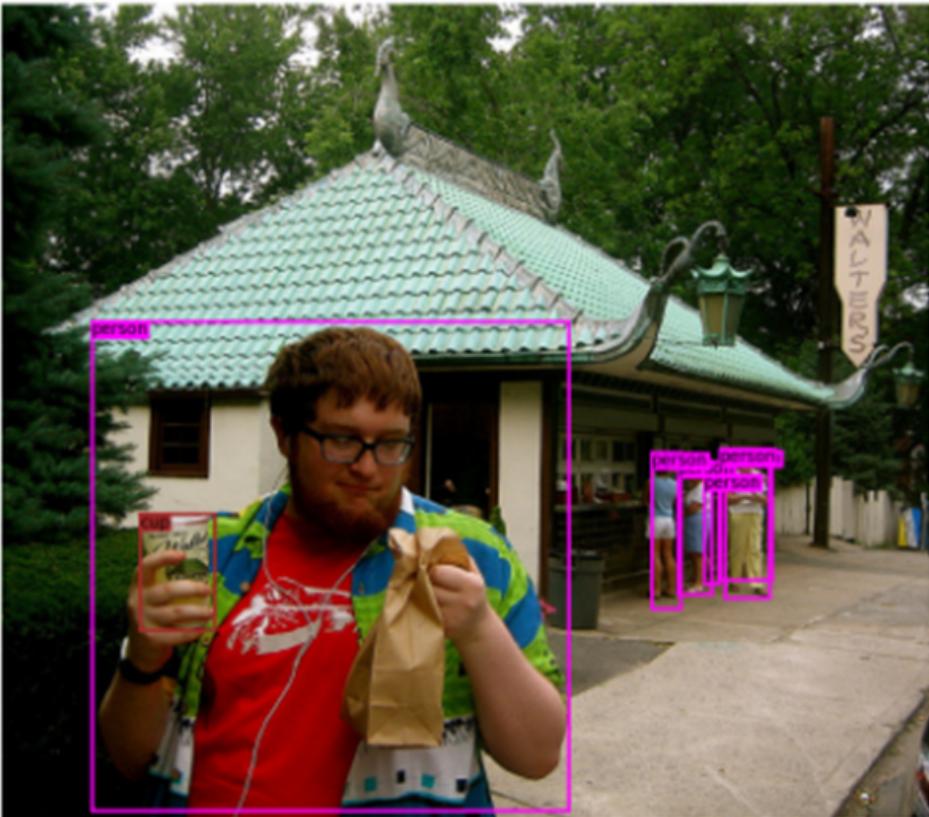
선택한 경계 상자와 나머지 상자 간의 IoU를 계산하여, 임계값을 초과하는 상자는 제거

- **반복**

다음 높은 점수의 경계 상자를 선택하고, 이 과정을 반복, 더 이상 선택할 경계 상자가 없거나 점수가 낮을 때까지 진행

# Result 이미지 확인

## Ground Truth



## 모델 추론



\*자원 부족으로 인해 팀원의 자료를 인용함

체 탐지가 원활하지 않아, 훈련량을 늘릴 수 있도록 자원 확보가 필요  
또한, 모델 구조와 손실 함수 등에서 코드 오류가 없는지 다시 한번 꼼꼼히 점검해야 할 필요가 있음

**THANK  
YOU**