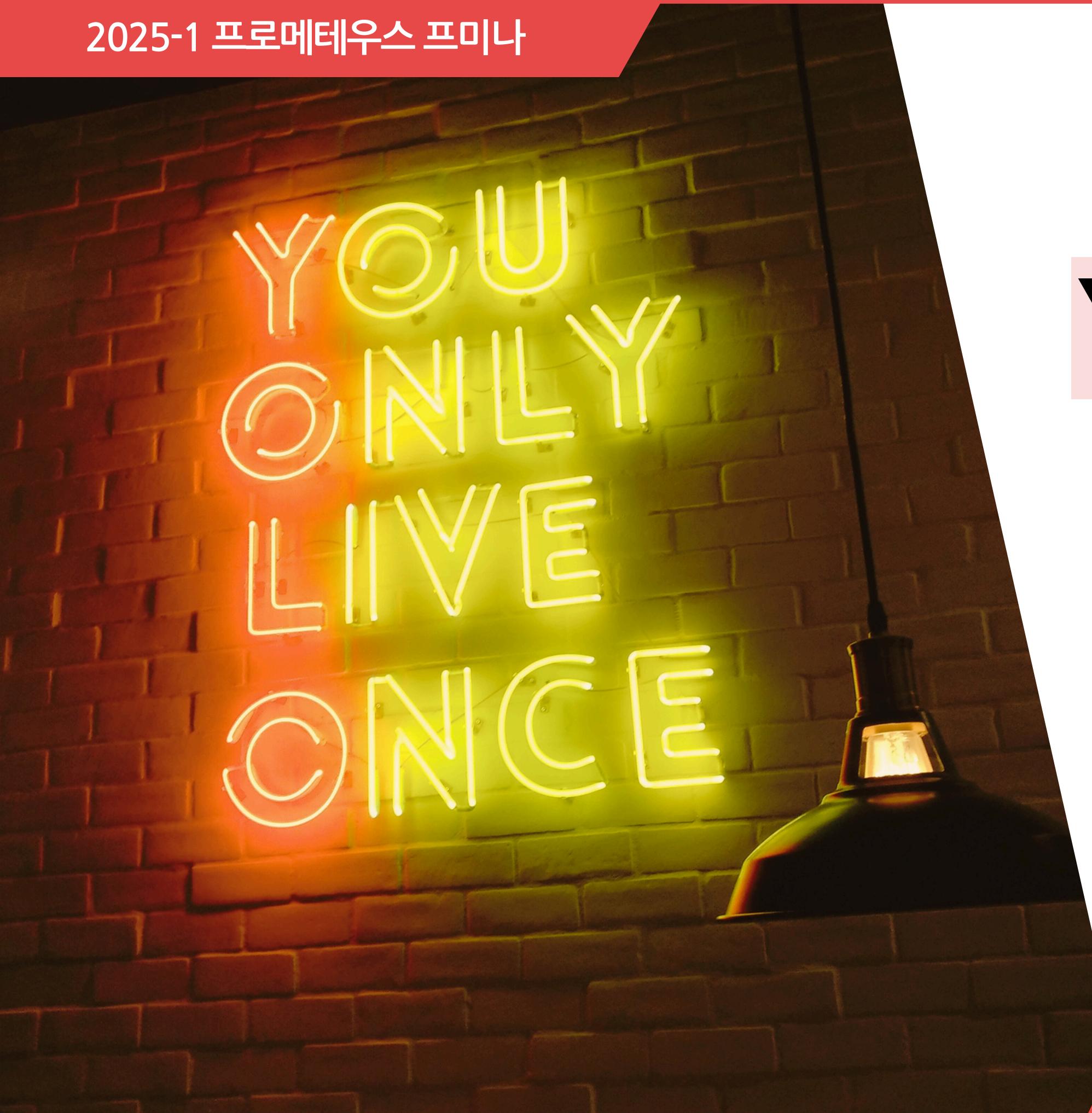


2025-1 프로메테우스 프미나



YOLO 논문 리뷰

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

Presented by 딥러닝 기초 스터디
(강지영, 박찬영, 한예원)



Contents

01 배경 지식 - CNN과 Object Detection 그리고 YOLO (강지영)

02 논문 전반부 - Unified Detection : Design & Training (한예원)

03 논문 후반부 - Comparison to Other Detection Systems & Experiments (박찬영)



YOLO 논문 리뷰



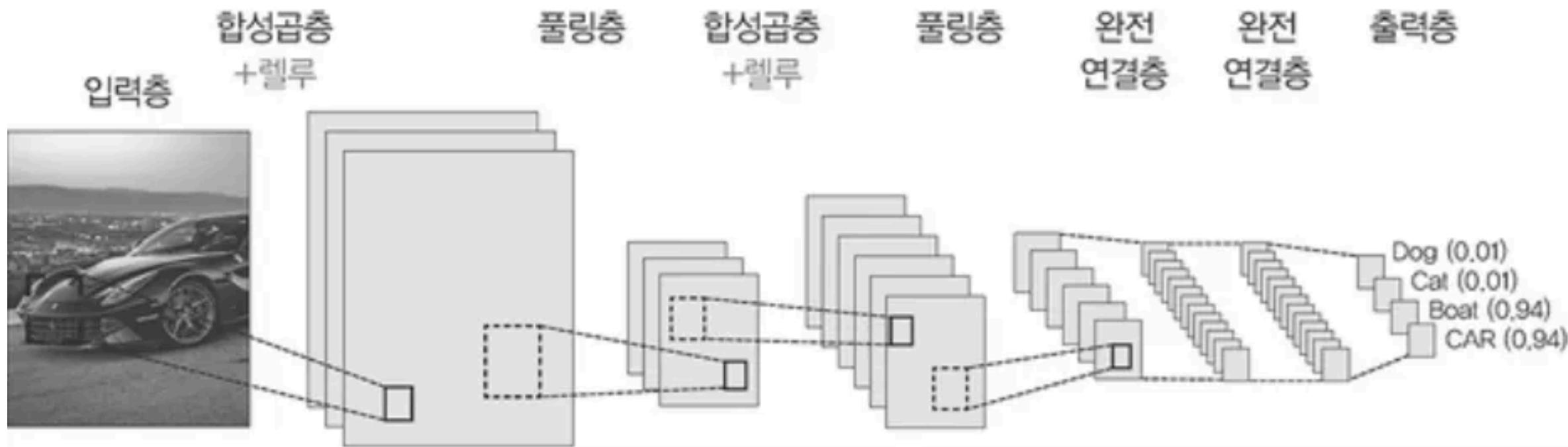
#1

CNN과 Object Detection 그리고 YOLO

Presented by 강지영



CNN



- **합성곱층**
 - 입력 데이터에서 특성 추출
 - 스트라이드 단위로 커널을 이동시켜 이미지의 모든 영역을 훑어 특성을 추출 → 특성맵
- **풀링층**
 - 특성 맵의 차원을 다운 샘플링해 주요 특성 벡터 추출
- **완전연결층**
 - 추출된 특징을 일렬로 나열해 분류 작업을 수행할 수 있는 형태로 변환



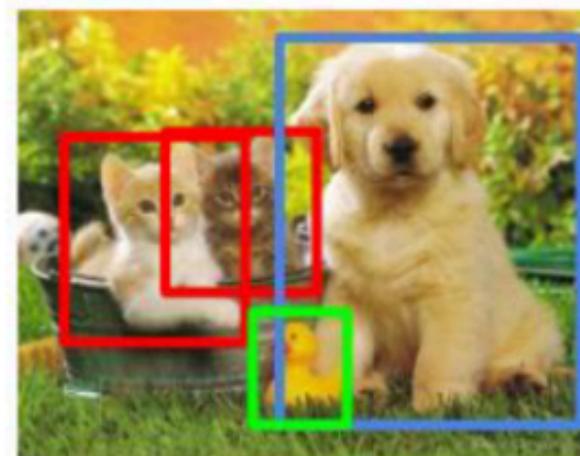
Object Detection

Classification



CAT

Object Detection



CAT, DOG, DUCK

Instance
Segmentation



CAT, DOG, DUCK

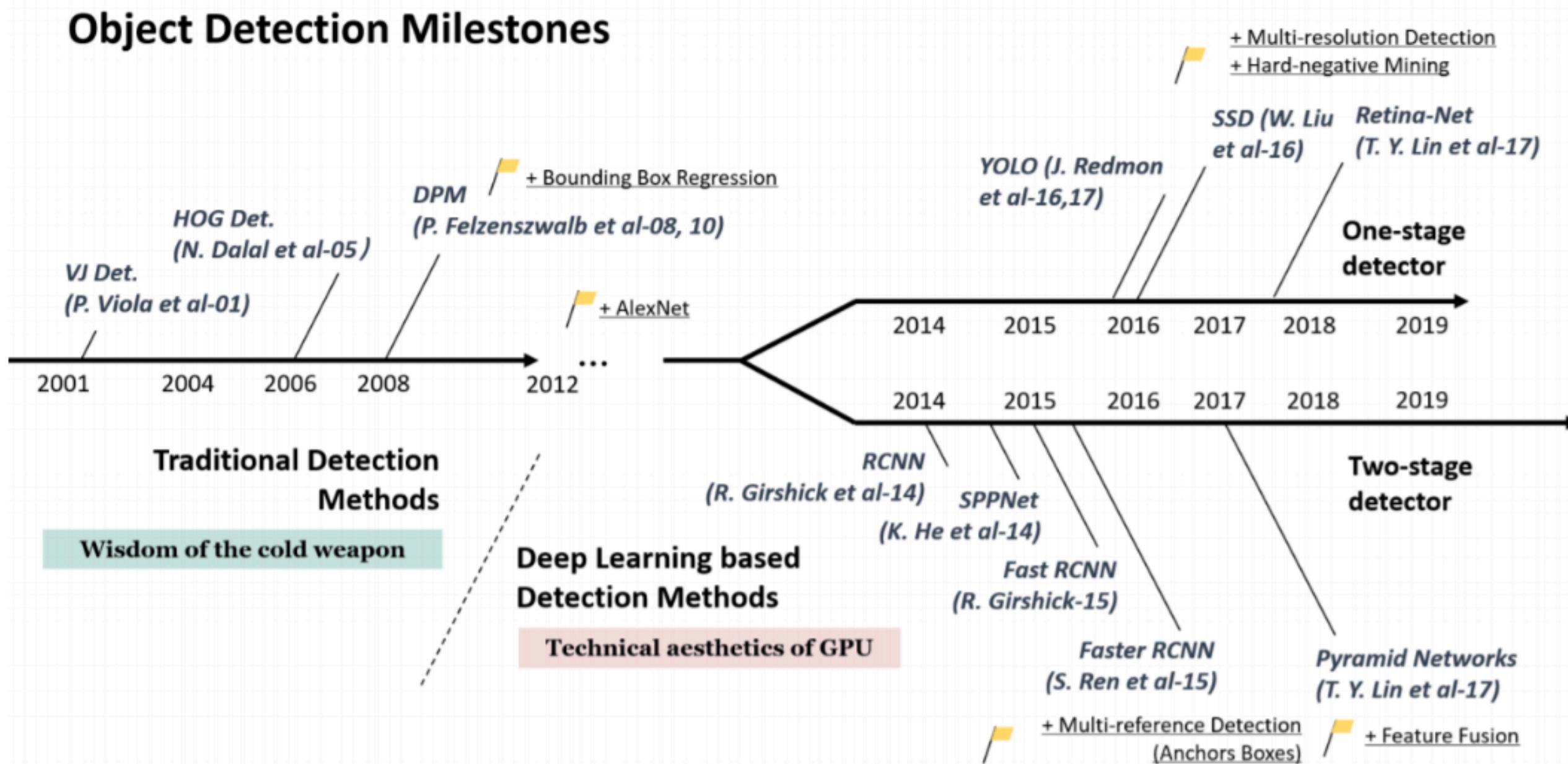
Object Detection 객체인식

: 이미지나 영상 내에 있는 여러 객체에 대해 분류하고 위치를 검출하는 컴퓨터 비전 기술

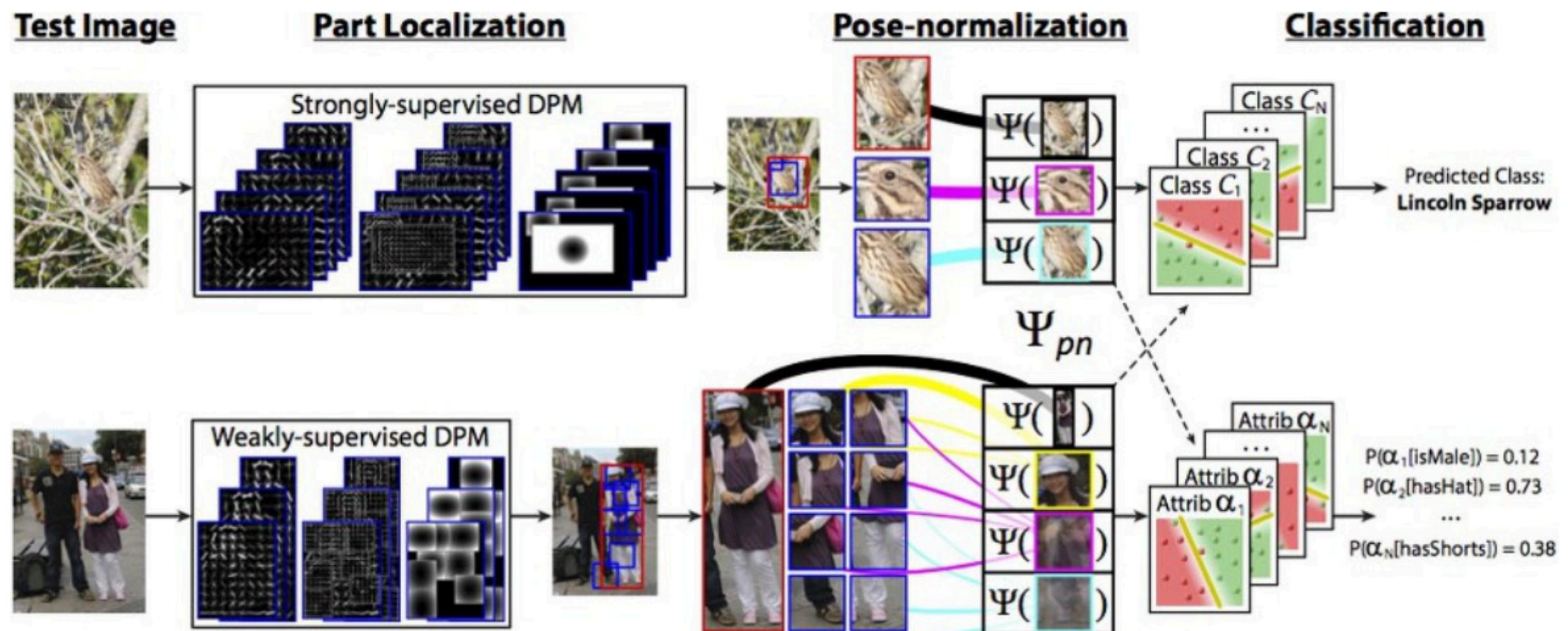
객체 인식 = 여러 가지 객체에 대한 분류 + 객체의 위치 정보를 파악하는 위치 검출



DPM → R-CNN → YOLO



DPM (Deformable Parts Model)



객체를 탐지하기 위한 여러 단계

1. Part Localization
 - a. 이미지 전체를 훑으며 슬라이딩 윈도우 방식으로 객체의 part 탐지
2. Pose_normalization
 - a. 비슷한 파트끼리 정렬(전처리 단계)
3. Classification

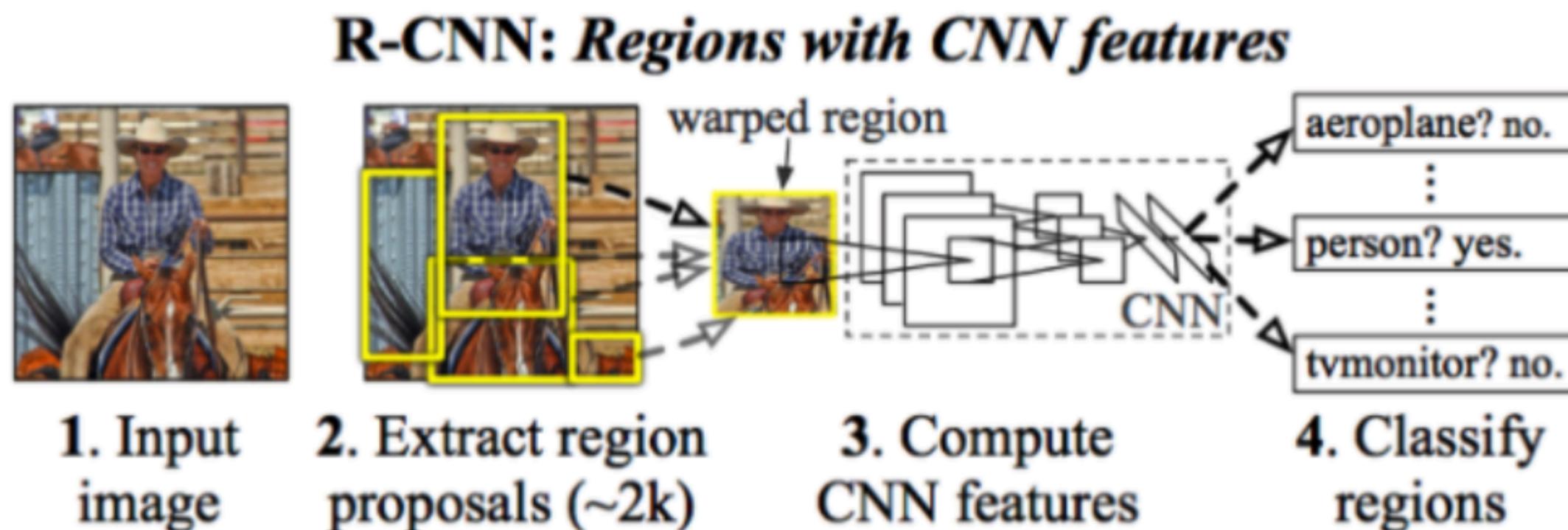


R-CNN

(Region-based CNN)

R-CNN 계열: Two-stage detector

객체를 탐지하기 위한 여러 단계



1. Extract region proposals
 - a. 바운딩박스 후보군 약 2000개를 추출
 - b. 동일한 input size로 만들기 위해 잘라냄
2. Compute CNN features
 - a. input size의 동일한 크기 이미지를 CNN 모델에 적용
3. Classify regions
 - a. classification

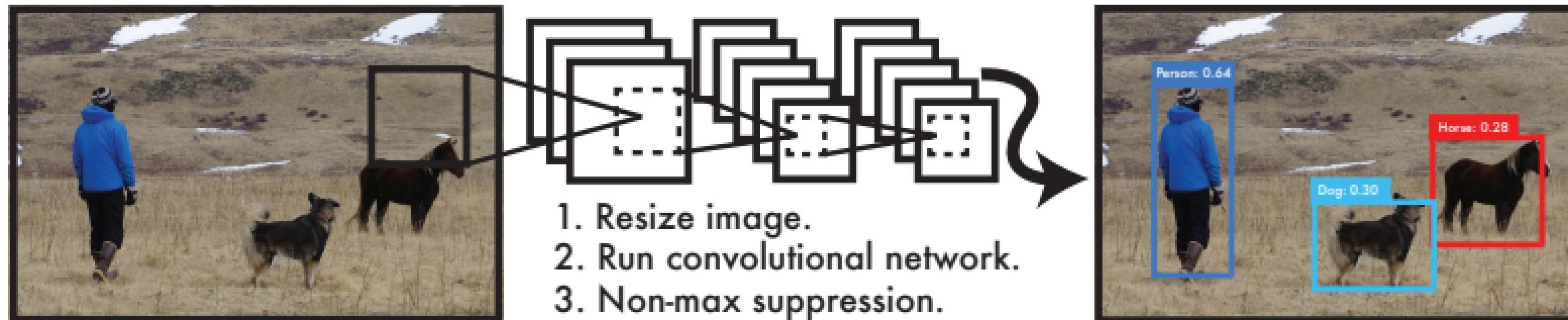


YOLO

(You Only Look Once)

YOLO: One-stage detector

하나의 통합된 신경망이 전체 이미지를 한 번에 보고 바운딩 박스와 클래스 정보를 동시에 예측
→ 성능 우수, 구조 단순, 빠름



1. 입력 이미지 448×448 resize
2. 하나의 CNN 실행
3. 신뢰도가 일정 수준 이상인 탐지만 필터링





#2

Unified Detection : Design & Training

Presented by 한예원



Unified Detection

YOLO는 이미지 전체의 특징으로 bounding box 예측!

1. input images를 $S \times S$ grid로 나눔
2. object의 중심이 grid cell 안에 위치하면 해당 object 검출
3. bounding box의 confidence score 예측

$$\Pr(\text{Object}) * IOU_{pred}^{truth}$$

*IOU = 교집합/합집합

*최종 예측 tensor의 dimension = $S \times S \times (B \times 5 + C)$



Unified Detection

bounding box는 5 predictions으로 구성

(x, y) : grid cell에서 bounding box 중심의 상대 위치 (0,1)

w: bounding box의 상대 width

h: bounding box의 상대 height

confidence: confidence score



Unified Detection

grid cell은 C conditional class probabilities를 예측한다.

$$C(\text{conditional class probabilities}) = \Pr(\text{Class}_i | \text{Object})$$

*bounding box 갯수에 상관없이 클래스는 하나만 예측

*테스트 단계에서는 confidence score을 곱해 다음 식을 사용

class specific confidence score

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



Network Design

24 convolution layer + 2 fully-connected layer → prediction tensor(7*7*30)

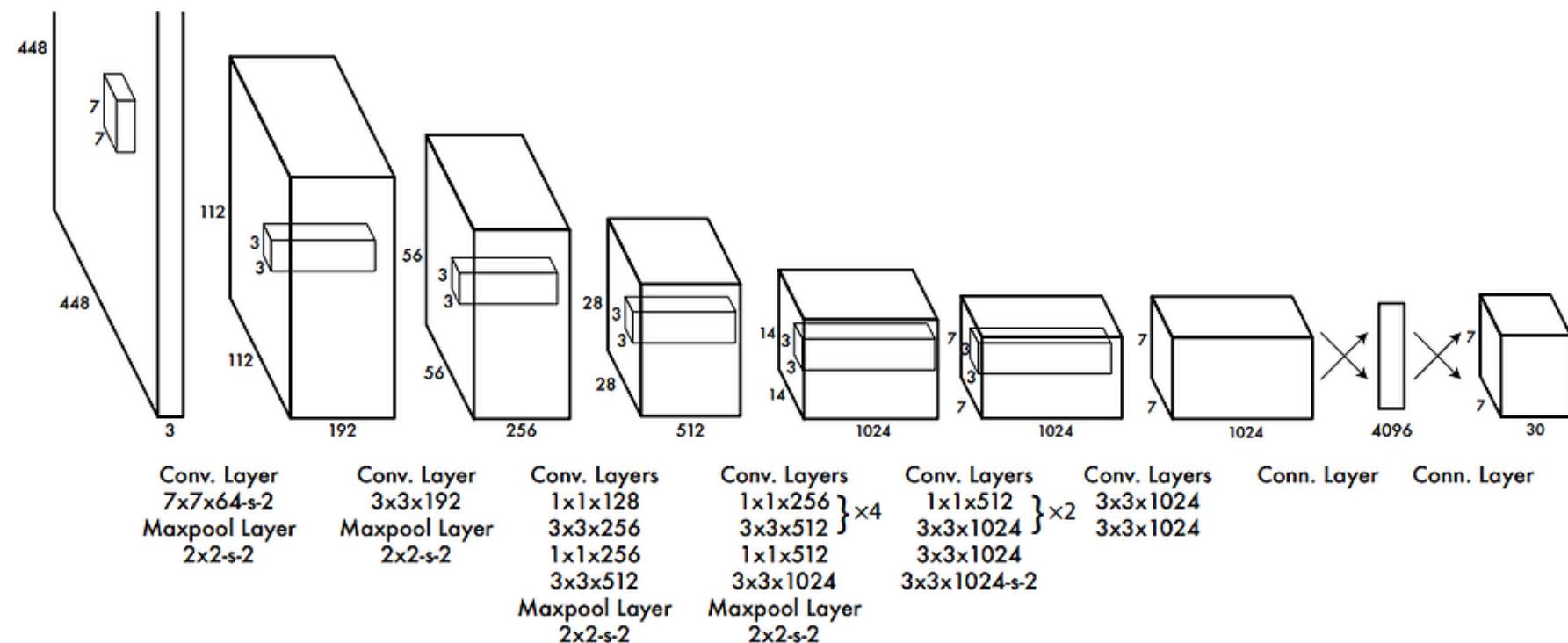


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

*Fast YOLO : 9 convolution + filter



Training

*사전 훈련은 ImageNet 1000-class competition data로, 20 convolution layers만
> 88% accuracy

*나머지 4개 convolution layers, 2개 fully connected layer를 추가해 분류 모델을 탐지 모델로 만들어줌
(가중치 임의로 초기화, input 해상도 224*224에서 448*448로 높여줌)

*최종 output은 class probabilities, bounding box의 coordinates(w, h, x, y) → 0,1 사이로 정규화

*마지막 layer에서는 linear activation function, 나머지 계층에는 leaky ReLU 적용

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$



Training

최적화

*loss는 SSE 기반: localization loss, classification loss → 이 두 가중치를 동일하게 취급하여 최적화

다만 이 경우 문제 발생

1. 이 최적화가 정확도를 높이는 좋은 방법은 아님
2. 이미지 내 대부분 grid cell엔 object가 없음 → 대부분 confidence score가 0점 → 모델 불균형
3. bounding box 사이즈에 상관없이 동일한 가중치: 실제로는 작은 bounding box가 위치변화에 더 민감



Training

2번 문제 개선

: coordinate에 대한 loss 가중치 증가, object 없는 bounding box의 confidence loss 가중치 감소

> 파라미터 $\lambda_{coord}=5$, $\lambda_{noobj}=0.5$

3번 문제 개선

: bounding box의 w, h에 루트 취해줌



Training

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

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

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

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

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

(1) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, x와 y의 loss를 계산.

(2) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, w와 h의 loss를 계산

(3) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, confidence score의 loss를 계산. ($C_i = 1$)

(4) Object가 존재하지 않는 그리드 셀 i의 bounding box predictor j에 대해, confidence score의 loss를 계산. ($C_i = 0$)

(5) Object가 존재하는 그리드 셀 i에 대해, conditional class probability의 loss를 계산. ($p_i(c)=1$ if class c is correct, otherwise: $p_i(c)=0$)



Training

overfitting 방지

- dropout layer : rate .5
- data augmentation : 원본 이미지 20%까지 random scaling, traslation





#3

Comparison to Other Detection Systems & Experiments

Presented by 박찬영



Comparison to Other Detection Systems (논문 3장)

🤔 기존 모델은 Object Detection을 어떻게 수행했을까?

- > **DPM (Deformable Parts Model)** : 슬라이딩 윈도우 기반
(정적 특성 추출 → 분류 → 박스 예측)
- > **R-CNN** : Selective Search 기반 (영역 제안 → 특성 추출 → 점수화/후처리 → 중복 제거)
- R-CNN은 속도가 느림(약 40초), YOLO는 더 적은 Bounding Box 제안(98개 vs 2000개)
- > **Fast/Faster R-CNN** : 속도 개선, 그러나 여전히 실시간 처리는 어려운 속도



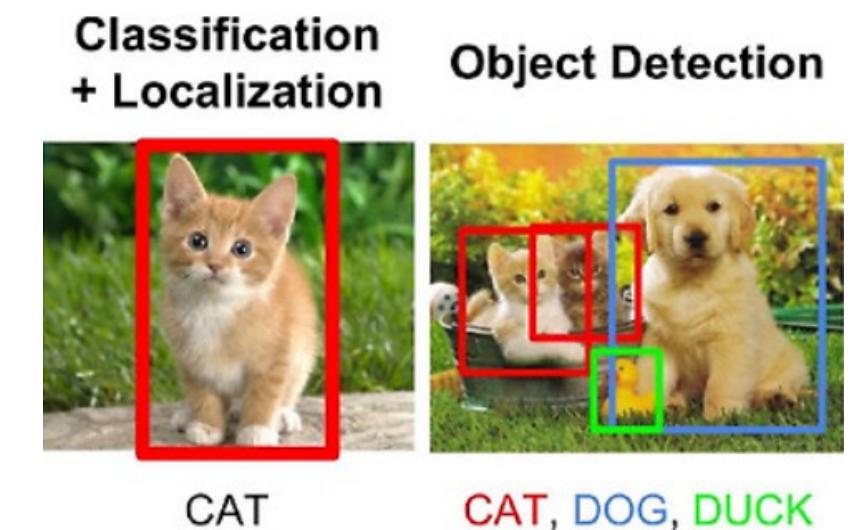
📌 YOLO는 Object Detection의 모든 과정을
하나의 Neural Network로 통합하여 **더 빠르다!**



Comparison to Other Detection Systems (논문 3장)

🤔 기존 모델로 Object Detection을 충분히 수행할 수 있지 않나?

- > **Deep Multibox** : CNN 기반 영역 제안, 그러나 일반적인 객체 탐지는 불가 (단일 클래스 예측 최적화)
- > **OverFeat** : 슬라이딩 윈도우 기반, 지역 정보만 활용(Localization 적합)
- > **MultiGrasp** : Grid Approach를 사용하여 단일 객체의 그리드 영역 예측



📌 YOLO는 전역(Global) 정보를 활용하여 일반적인(General) 다중 클래스 객체를 탐지할 수 있다!



Experiments (논문 4장)

기존 모델 중에는 실시간 처리 가능한 모델이 없었나?

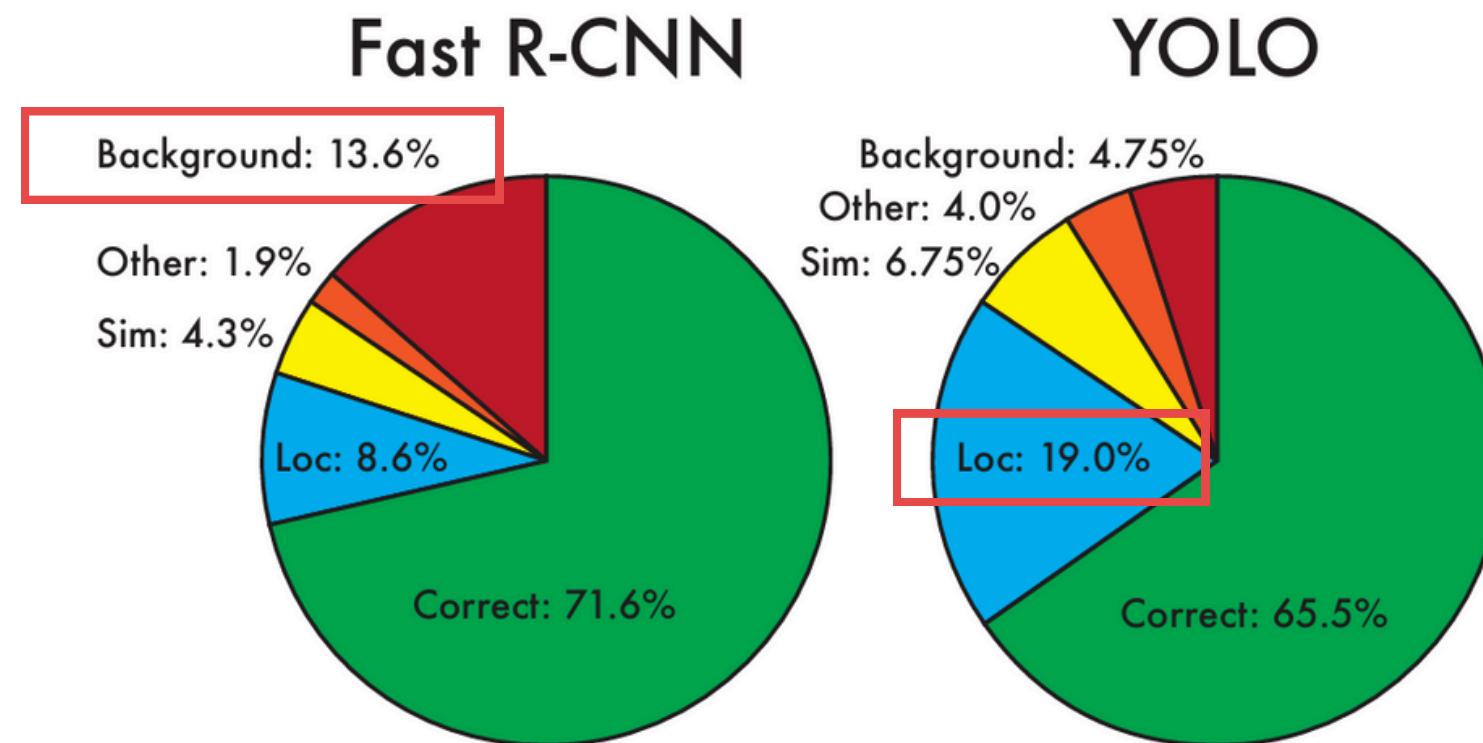
Real-Time Detectors	Train	mAP	FPS	테스트 기준 : PASCAL VOC 2007
100Hz DPM [30]	2007	16.0	100	
30Hz DPM [30]	2007	26.1	30	
Fast YOLO	2007+2012	52.7	155	→ 기존 실시간 모델 대비 2배 높은 정확도, 빠른 속도
YOLO	2007+2012	63.4	45	→ Fast YOLO 대비 mAP 10% 증가
<hr/>				
Less Than Real-Time				
Fastest DPM [37]	2007	30.4	15	
R-CNN Minus R [20]	2007	53.5	6	
Fast R-CNN [14]	2007+2012	70.0	0.5	→ Selective Search 2초
Faster R-CNN VGG-16[27]	2007+2012	73.2	7	→ RPN(Regional Proposal Network) 기반 속도 단축, 그러나 YOLO 대비 6배 느린 속도
Faster R-CNN ZF [27]	2007+2012	62.1	18	→ YOLO 대비 2.5배 느린 속도, 적은 정확도

📌 YOLO는 **높은 정확도와 빠른 속도**를 동시에 보인다!



Experiments (논문 4장)

🤔 YOLO와 Fast R-CNN을 합쳐볼까?



	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

R-CNN이 예측한 Bounding Box
 → YOLO가 유사한 박스를 예측하는지 확인

→ 정확도 증가, 그러나
 YOLO의 빠른 속도를
 이점으로 활용 X

📌 YOLO와 Fast R-CNN을 합쳐 Background Error를 줄일 수 있다!



Experiments (논문 4장)

YOLO와 Fast R-CNN을 합쳐볼까?

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 [27]	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 [28]	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 [32]	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

테스트 기준 : PASCAL VOC 2012

→ Fast R-CNN + YOLO의 높은 성능

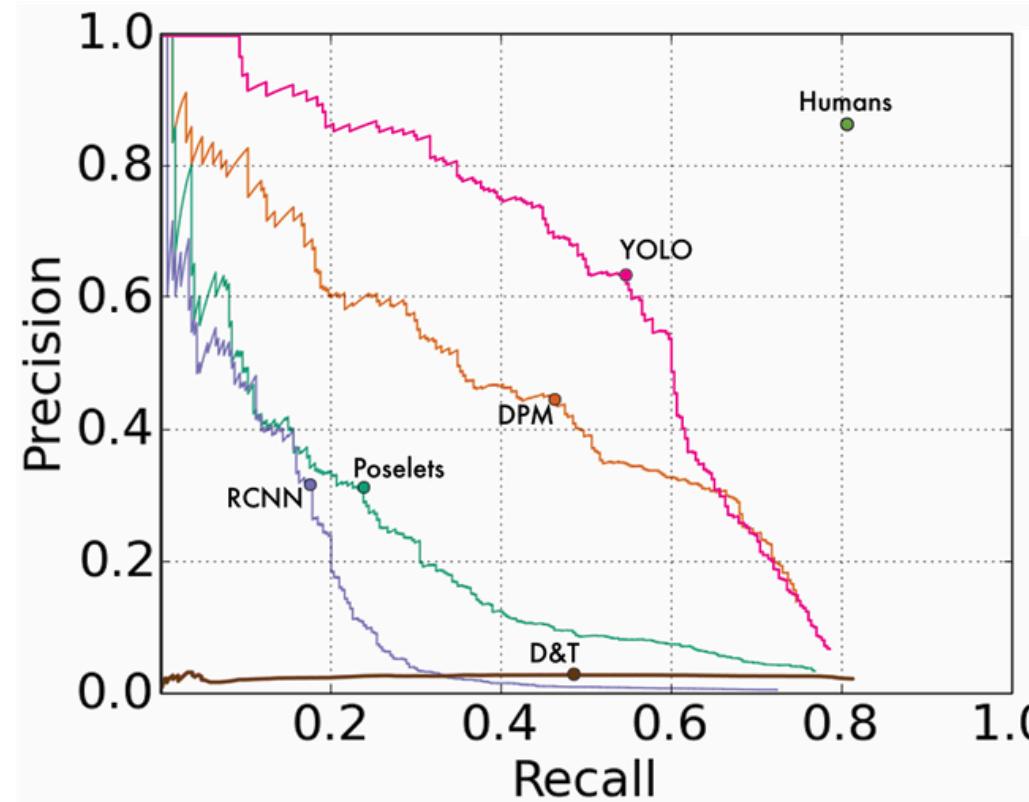
→ 작은 물체 탐지의 어려움

📌 YOLO는 작은 물체 탐지에 어려움을 겪는다!



Experiments (논문 4장)

 YOLO는 일반적인 Object Detection 과제에 적합할까?



테스트 기준 : Picasso Dataset, People-Art Dataset

	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.

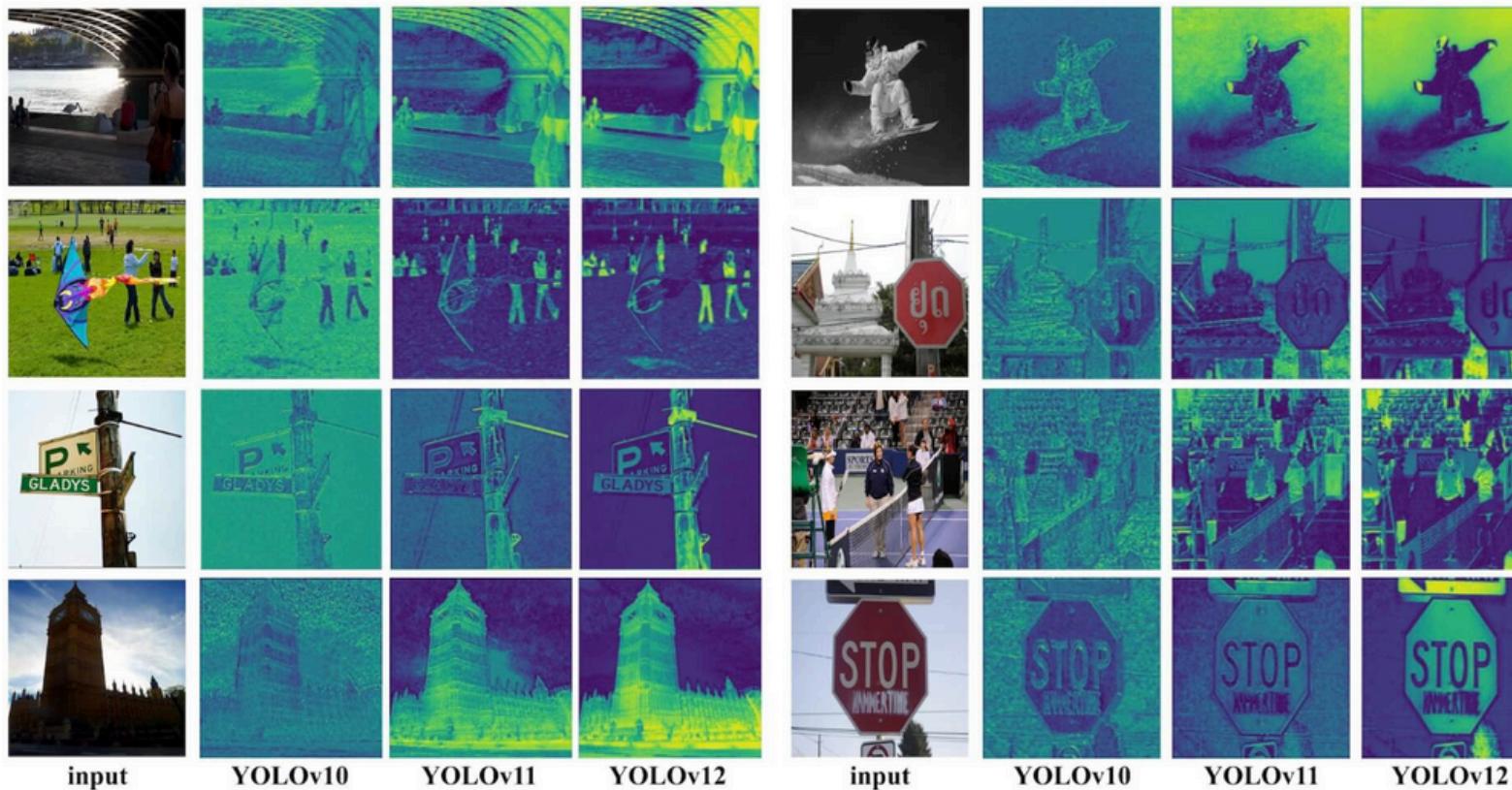
→ R-CNN은 PASCAL VOC에 과적합 (작은 지역만 봄)
→ DPM은 성능 유지 (물체의 모양에 대한 공간적 모델)

📌 YOLO는 이미지 전체 정보를 활용하여 **일반적인 객체 탐지에 적합**하다!



Real-Time Detection In The Wild & Conclusion (논문 5~6장)

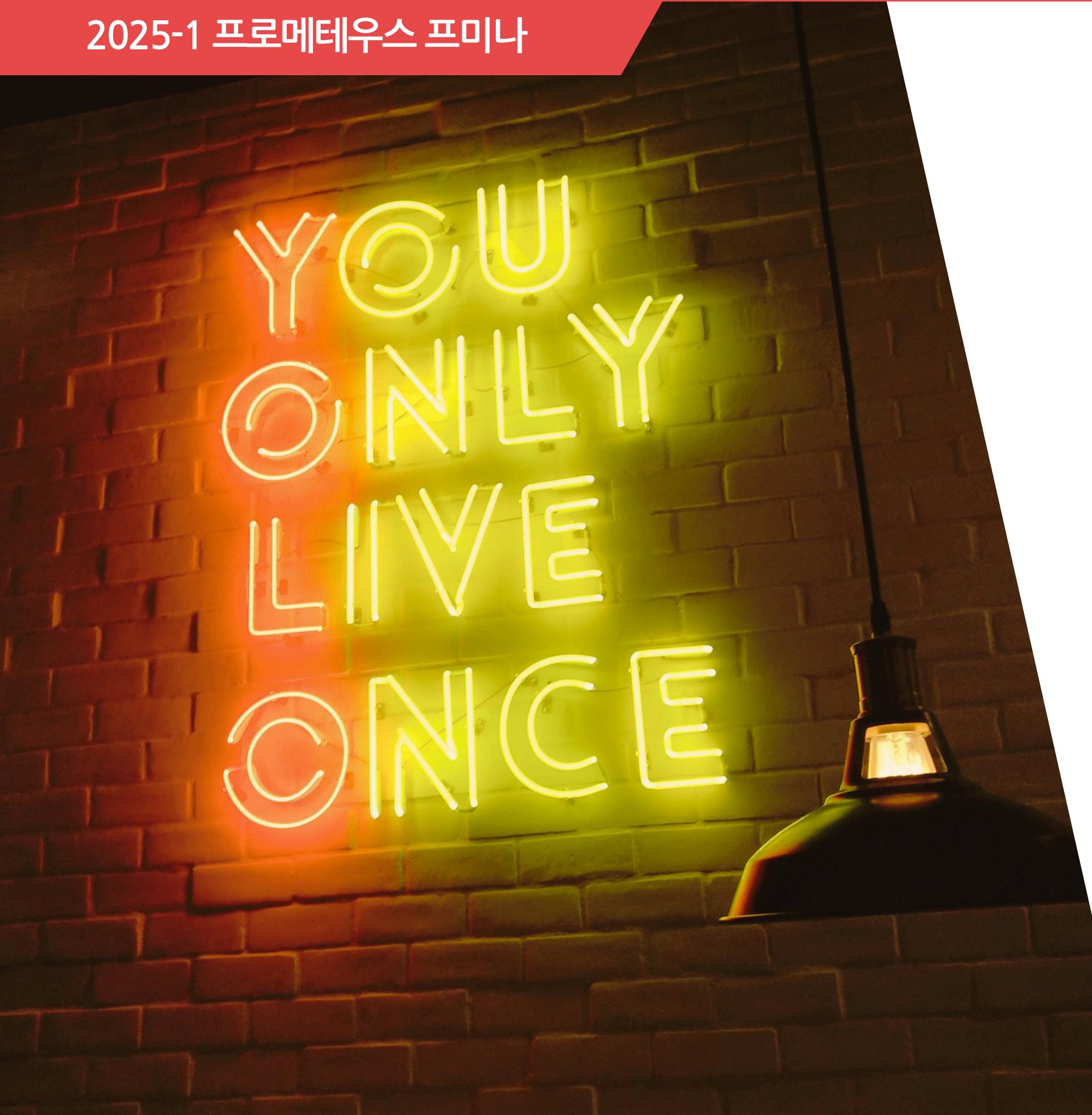
- ✓ YOLO는 Object Detection을 위한 **Unified Model**
- ✓ YOLO는 **높은 정확도와 빠른 속도**를 갖기에 **실시간 처리**가 가능 (Tracking System 사용)
- ✓ YOLO는 **일반적인 객체 탐지**에도 적합



YOLO v2 (2017) : Global Average Pooling, Anchor Box 도입
YOLO v3 (2018) : Feature Pyramid Network → **작은 객체 탐지 개선**
YOLO v4 (2020) : SPP + PANet, Data Augmentation, Quantization
YOLO v5 (2020) : Pytorch 변환, Mosaic Augmentation
YOLOX (2021) : SPP, Anchor-free (중심점만 사용), SimOTA
YOLO v6 (2022) : Rep-PAN, PTQ와 QAT → 양자화 체계 최적화
YOLO v7 (2022) : E-ELAN, Soft Label 생성 → 파라미터 효율성 개선
YOLO v8 (2023) : SPP + PANet + SAM → **경량화된 모델**
YOLO v9 (2024) : PGI, Gradient Path Planning → 정보 병목 문제 해결
YOLO v10 (2024) : Dual Label Assignments (NMS 제거), Rank-guided Block Design
YOLO v11 (2024) : C3k2 Block + SPPF + C2PSA → **다양한 태스크 지원**
YOLO v12 (2025) : Area Attention, R-ELAN, 주의 집중 아키텍처 최적화 (플래시어텐션)

참고 : <https://c0mputermaster.tistory.com/30>





감사합니다 :)

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

