YOLO 알고리즘
(You Only Look Once)

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

김영수

출처

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

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detec-

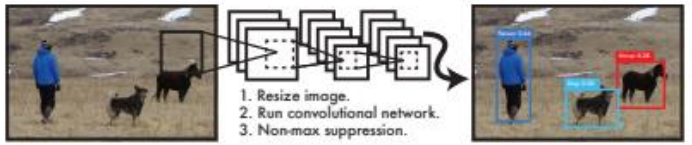


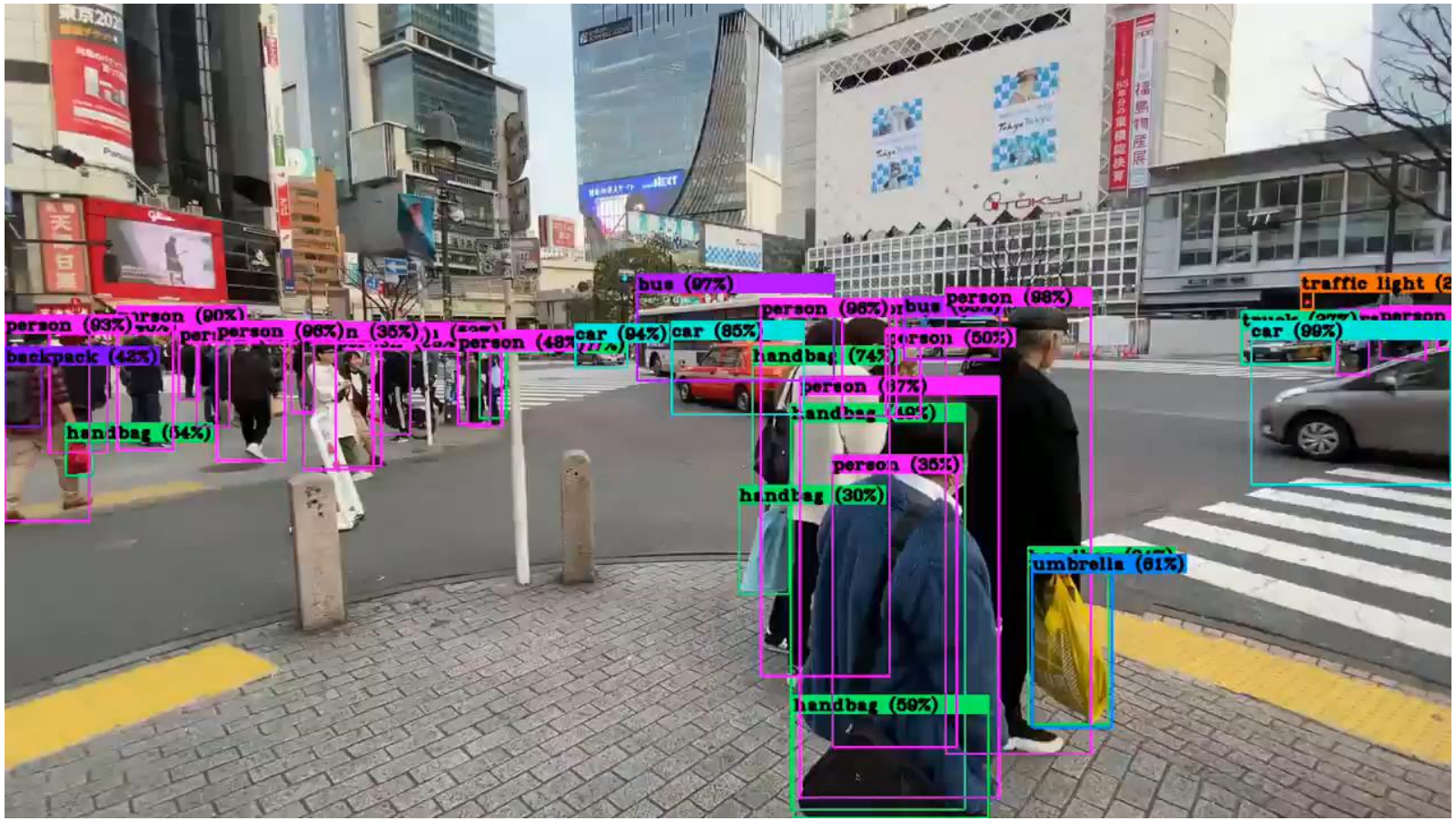
Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

수업목표

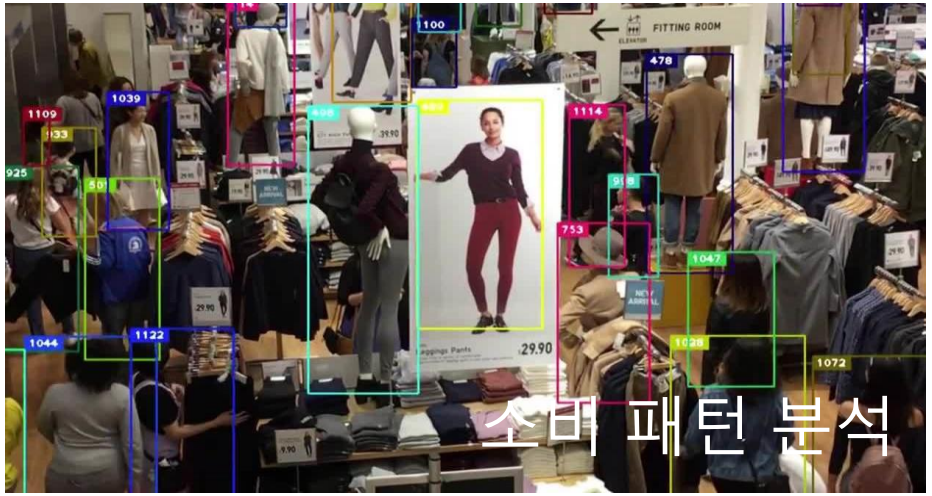
“YOLO 알고리즘의 처리과정 이해”

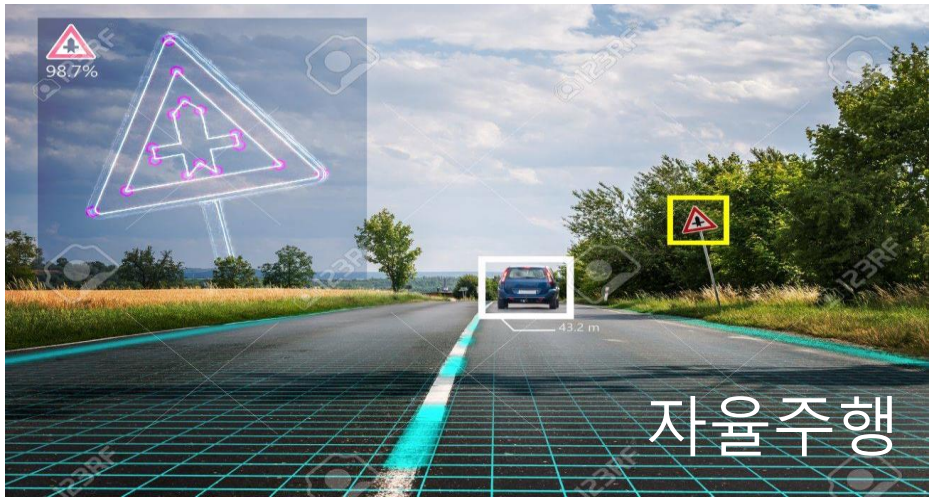


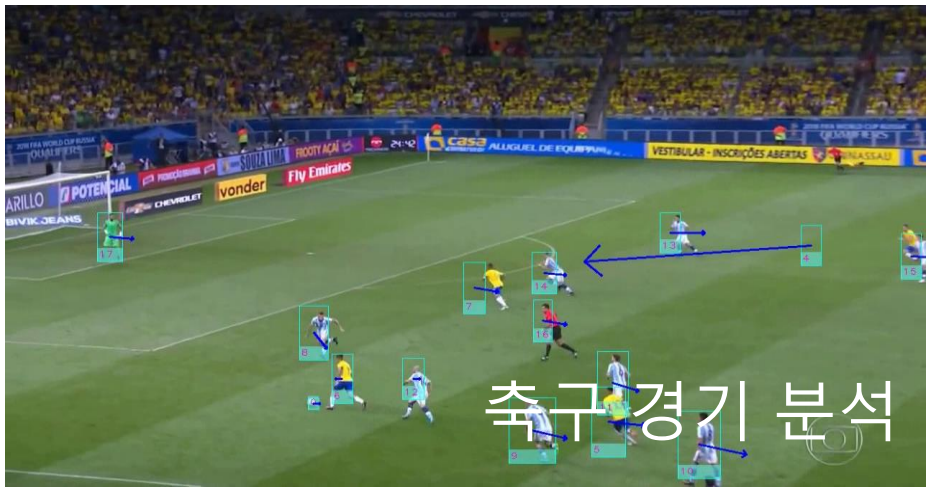
YOLO 알고리즘이 중요한 이유

- **속도**
  - 물체를 실시간으로 예측하여 감지 속도 향상
- **높은 정확도**
  - 최소한의 배경 오류로 정확한 결과를 제공
- **학습 기능**
  - YOLO는 객체의 표현을 학습하고 이를 객체 탐지에 적용할 수 있는 뛰어난 학습 기능을 보유

YOLO의 활용(예)

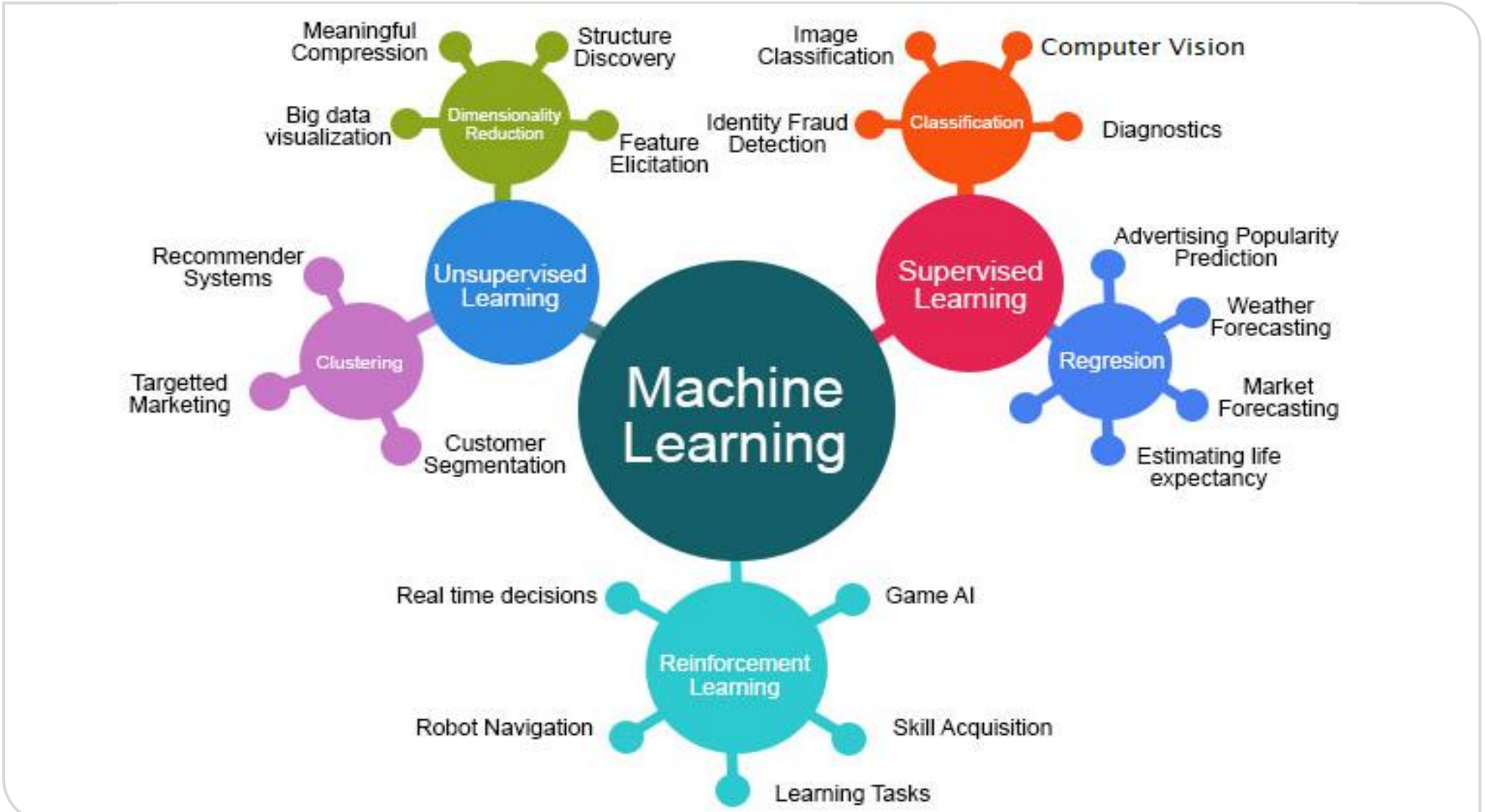






목차

객체 탐지 알고리즘 개요(1/3)



객체 탐지 알고리즘 개요(2/3)

Single Object

Classification

Q. "이 이미지는 무엇인가?"
- 이미지에서 single object 감지

A. "고양이 일 것 같습니다."

$y = \{P_{고양이}, P_{개}, P_{오리}\}$
$= \{ 0.9, \ 0.05, \ 0.05 \}$

CAT

Classification + Localization

Q. "이 이미지는 무엇이 어디쯤에?"
- single object 감지 + 위치 추적

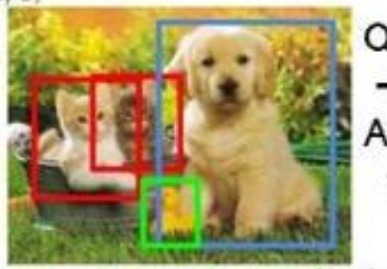
A. "고양이가 중간쯤에 있습니다."

$y = \{P_{고양이}, P_{개}, P_{오리}, N_x, N_y, N_w, N_h\}$
$= \{0.9, \ 0.3, \ 0.1, 0.3, 0, 0.5, 0.9\}$

CAT

Multiple Objects

Object Detection



Q. "어떤 것들이 어디쯤에?"
- 이미지에서 복수 object 위치 감지

A. "고양이는 어디쯤, 오리는 어디쯤…"

$$y = \begin{Bmatrix} P_{고양이} & P_{개} & P_{오리} \\ N_x & N_x & N_x \\ N_y & N_y & N_y \\ N_w & N_w & N_w \\ N_h & N_h & N_h \end{Bmatrix}$$

CAT, DOG, DUCK

Instance Segmentation

Q. "어떤 것들이 어디에?"
- 복수 object 정확한 영역 감지

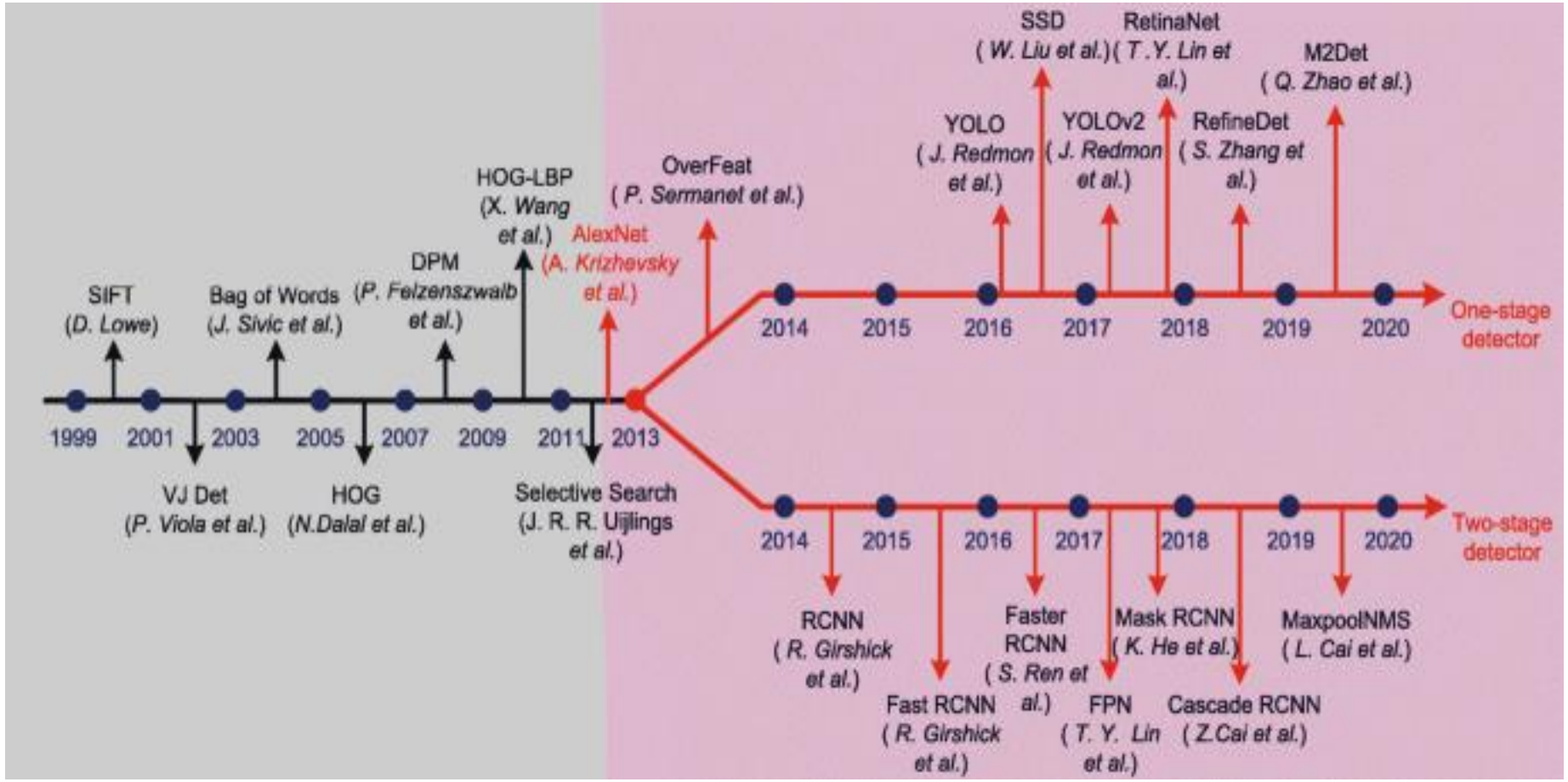
A. "고양이는 어디, 오리는 어디…"

$$y = \begin{Bmatrix} P_{고양이} & P_{개} & P_{오리} \\ N_{x1} & N_{x1} & N_{x1} \\ N_{x\cdots} & N_{x\cdots} & N_{x\cdots} \\ N_{y1} & N_{y1} & N_{y1} \\ N_{y\cdots} & N_{y\cdots} & N_{y\cdots} \end{Bmatrix}$$

CAT, DOG, DUCK

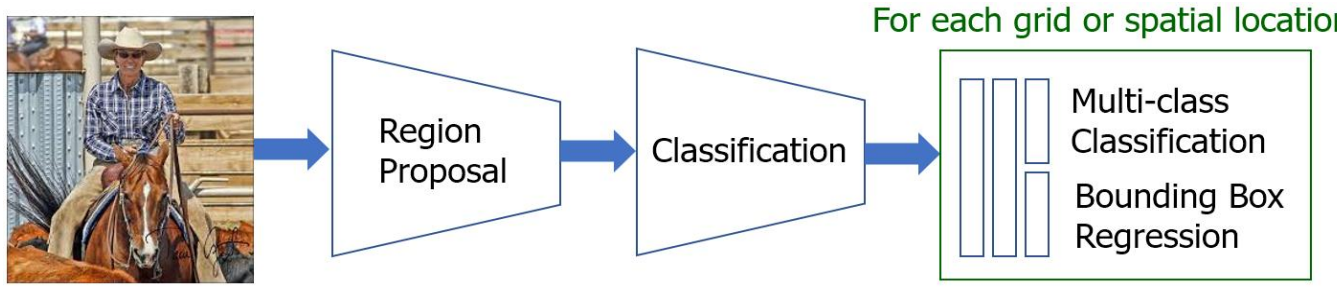
√ 객체 탐지 알고리즘 개요(3/3)

- 객체 탐지 알고리즘의 발전과정

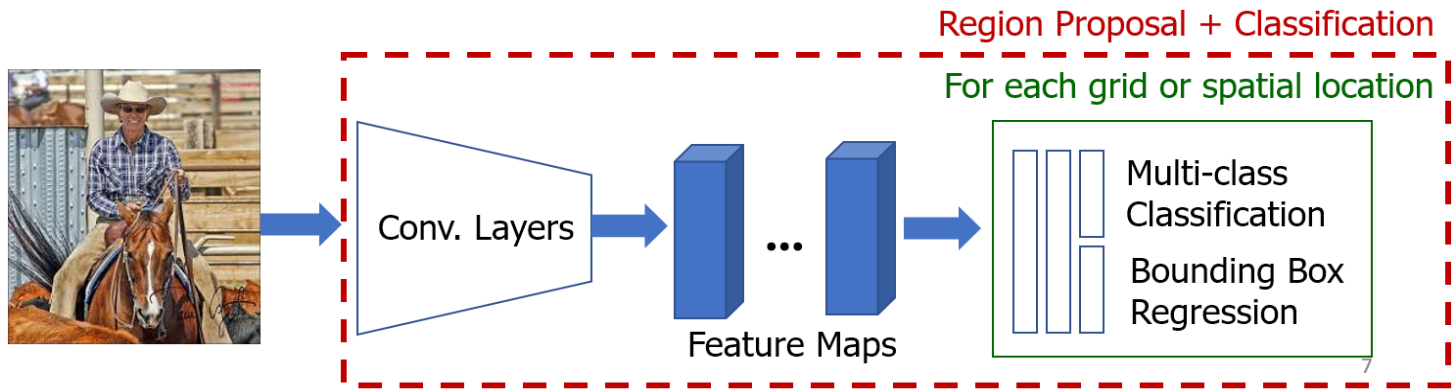


딥러닝 기반의 객체 탐지 방식 비교

- **2-stage 객체 탐지 알고리즘**
  - Localization(후보 obj. 위치 제안) → Classification 순차적으로 수행



- **1-stage 객체 탐지 알고리즘**
  - Localization과 Classification을 동시에 수행



딥러닝 기반의 객체 탐지 방식 비교

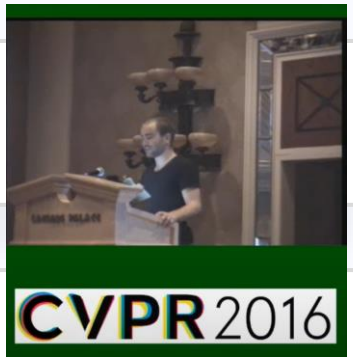
- R-CNN 알고리즘 처리방식 : 2-stage detection

✓속도가 너무 늦다.

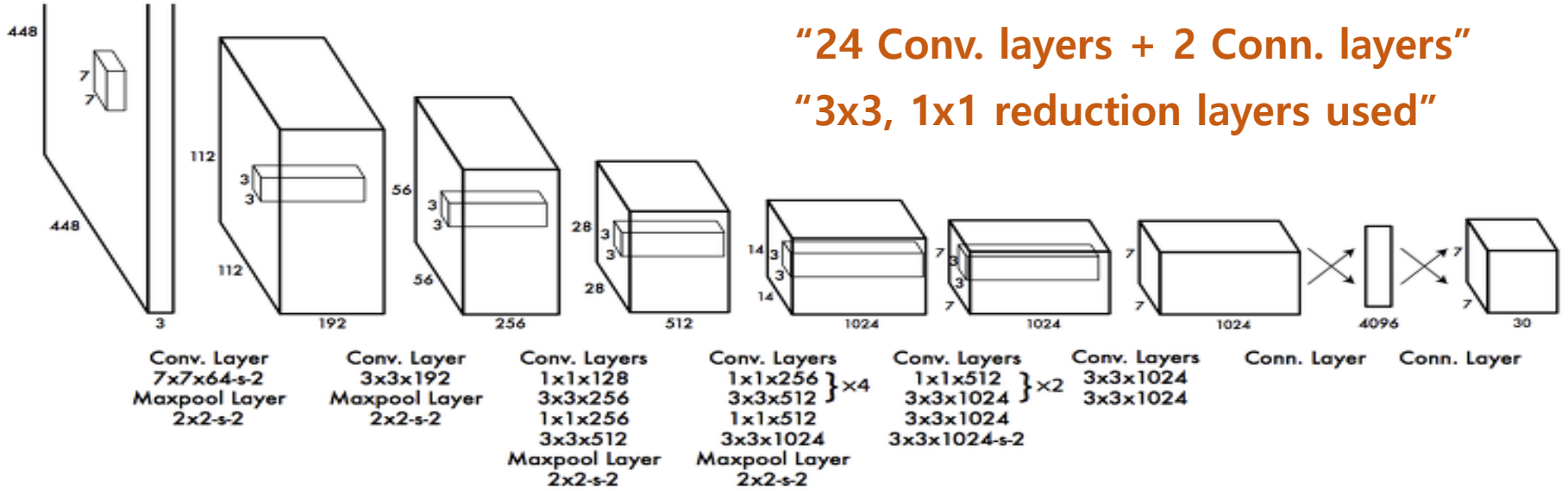
✓부정확한 경계상자

- Region Propos

YOLO 알고리즘 개요



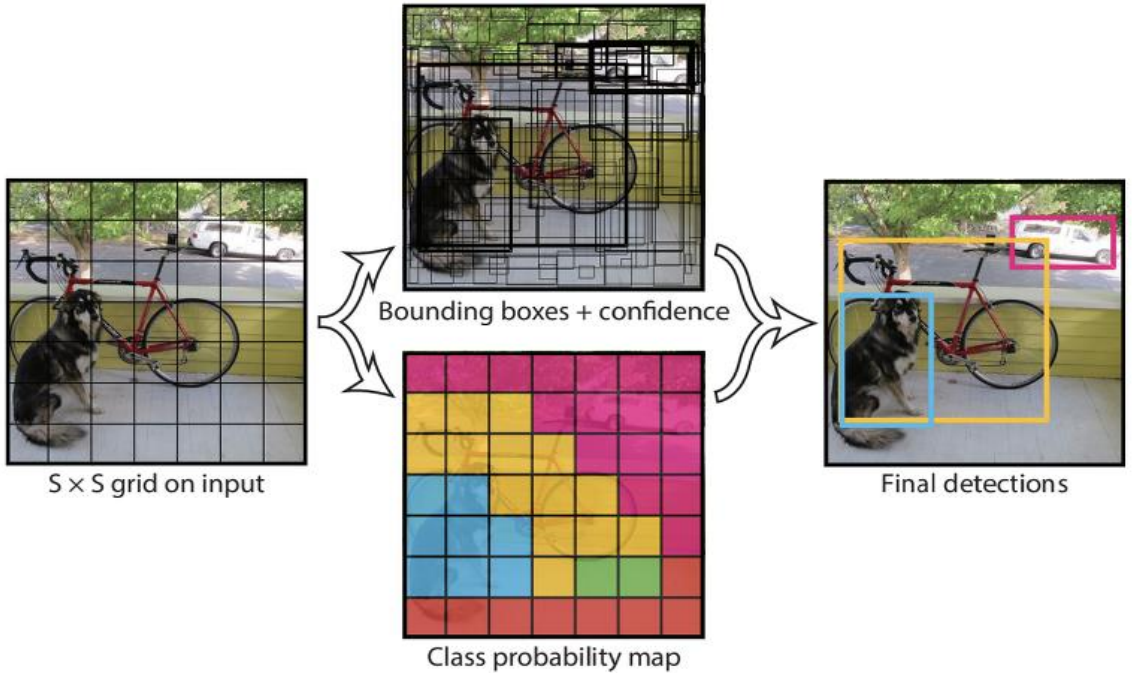
- **CVPR 2016, Joseph Redmon이 제안**
  - 기존 객체 탐지 알고리즘(2-stage detector)의 속도문제 해결
- **통합 탐지 방식(1-stage or unified detection)**
  - Region Proposal과 Classification을 동시(한번)에 처리
- **Backbone 모델로서 GoogLeNet 사용**



"24 Conv. layers + 2 Conn. layers"

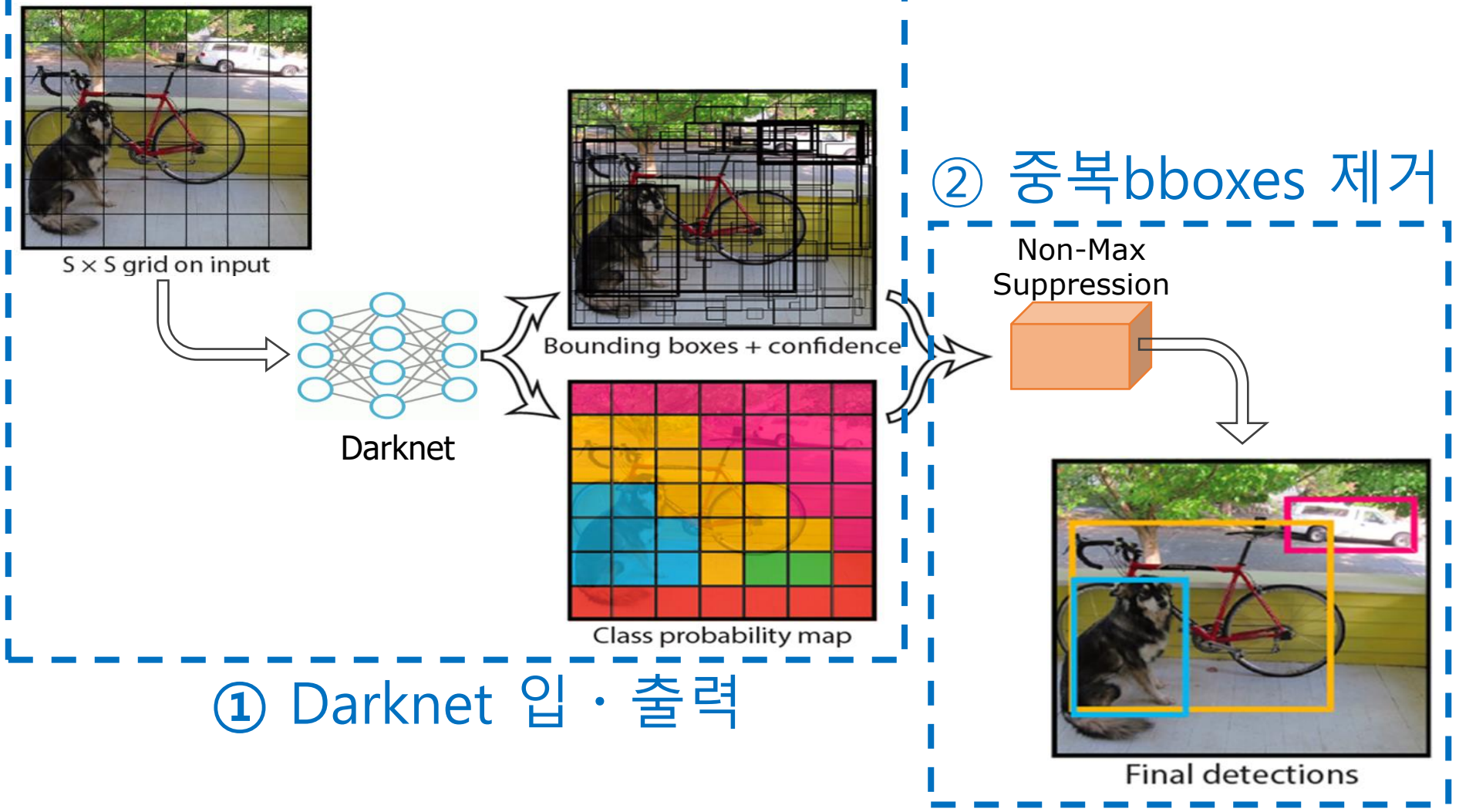
"3x3, 1x1 reduction layers used"

Unified Detection



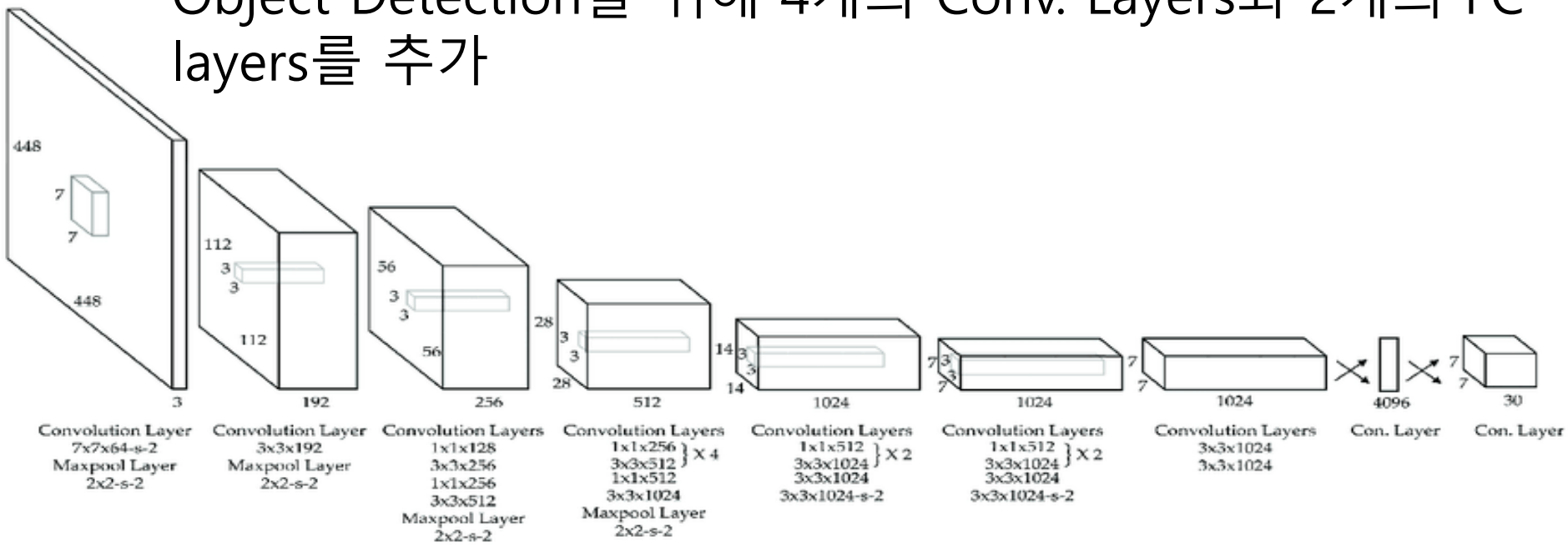
- Region proposal, feature extraction, classification, bbox regression → one-stage detection으로 통합
- 이미지 전체로부터 얻은 Feature map을 활용하여 bbox 예측 & 모든 클래스에 대한 확률 계산
- SxS grid cell → each grid cell, B bbox prediction + confidence & class probabilities → SxSx(Bx5+C)

YOLO 알고리즘 처리절차



Darknet 입·출력

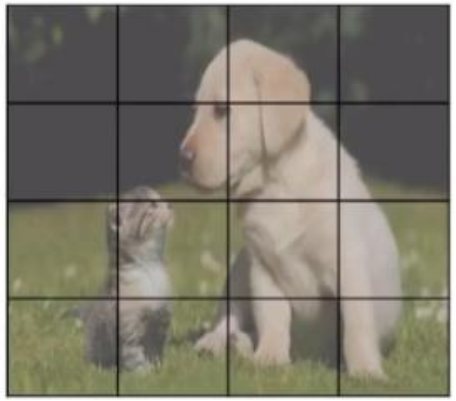
- **Darknet**
  - Backbone : GoogLeNet
  - 24개의 Conv. Layers과 2개의 FC Layers으로 구성
    - 20 Conv. Layers은 ImageNet Pretrained Weights를 활용
    - Object Detection을 위해 4개의 Conv. Layers와 2개의 FC layers를 추가



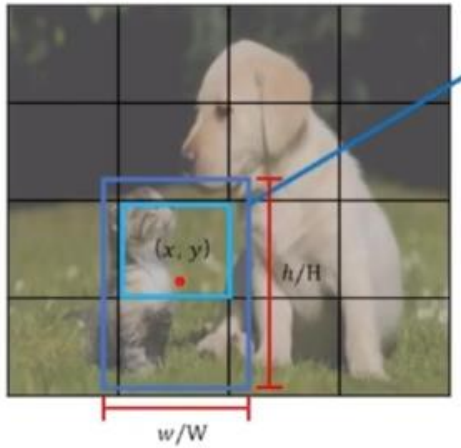
Darknet 입·출력

Unified Detection

예시) S = 4, B = 2, C = 20



Resized image 를
4x4 grid 로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

x
y
w
h
p_c

bbox의 중심좌표의 위치
(grid cell 기준)

Input image W, H 로 normalize

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

cf. $\Pr(Object)$:
물체가 bbox 내에 있으면 1,
없으면 0

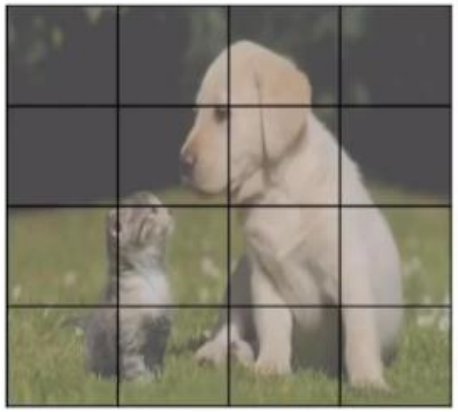
$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

12

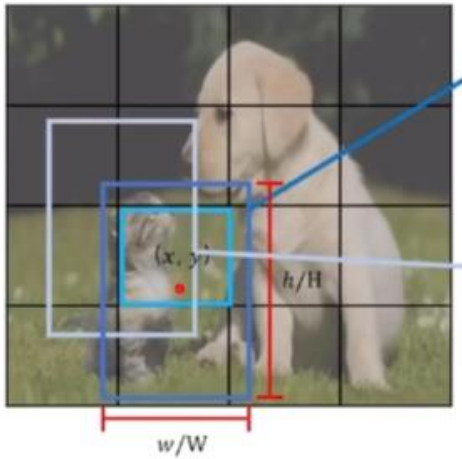
Darknet 입·출력

Unified Detection

예시) S = 4, B = 2, C = 20



Resized image 를
4x4 grid 로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

bbox #2

x
y
w
h
p_c
x
y
w
h
p_c

bbox의 중심좌표의 위치
(grid cell 기준)

Input image W, H 로 normalize

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

cf. $\Pr(Object)$:
물체가 bbox 내에 있으면 1,
없으면 0

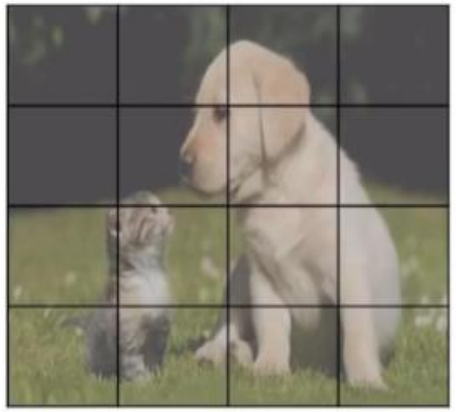
$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

13

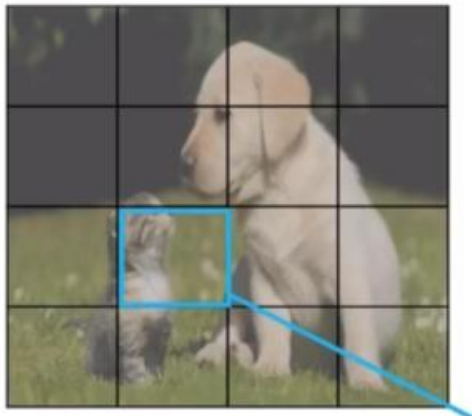
Darknet 입·출력

Unified Detection

예시) S = 4, B = 2, C = 20



Resized image 를
4x4 grid 로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

bbox #2

x
y
w
h
p_c
x
y
w
h
p_c
c_1
c_2
c_3
⋮
c_{20}

bbox의 중심좌표의 위치
(grid cell 기준)

Input image W, H 로 normalize

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

cf. $\Pr(Object)$:
물체가 bbox 내에 있으면 1,
없으면 0

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

$\Pr(Class_i|Object)$

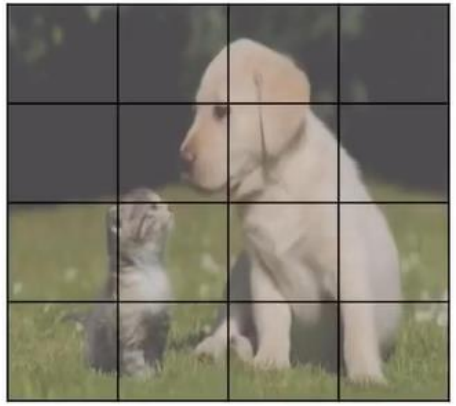
: 물체가 bbox 내에 있을 때,
Grid cell에 있는 object가 i번째
class 에 속할 확률

14

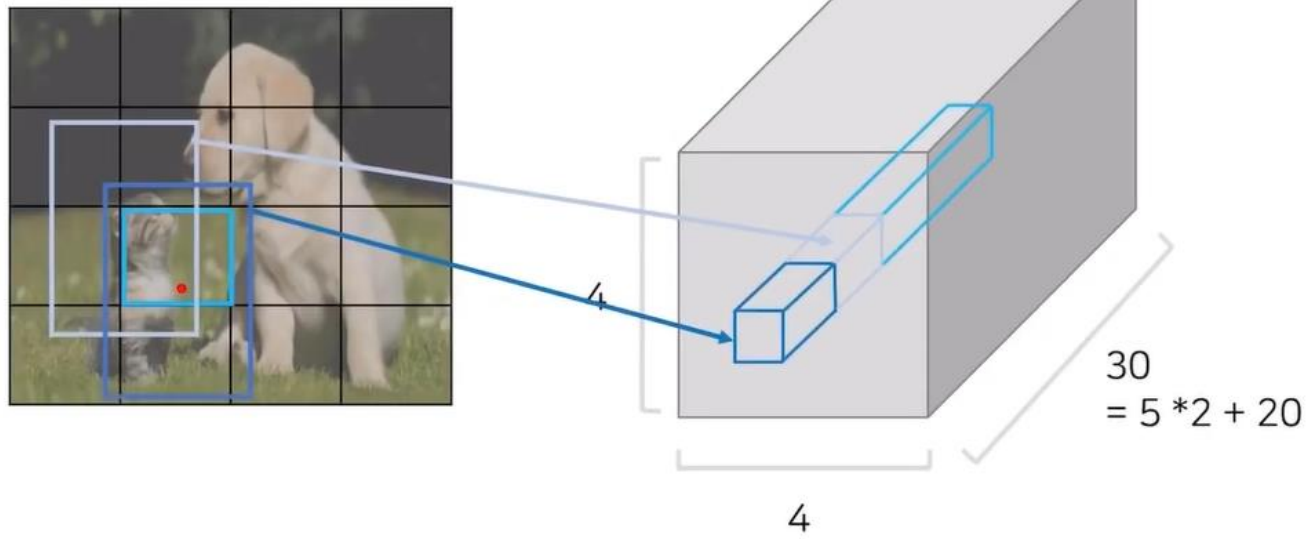
Darknet 입·출력

Unified Detection

예시) S = 4, B = 2, C = 20



Resized image 를
4x4 grid 로 분할



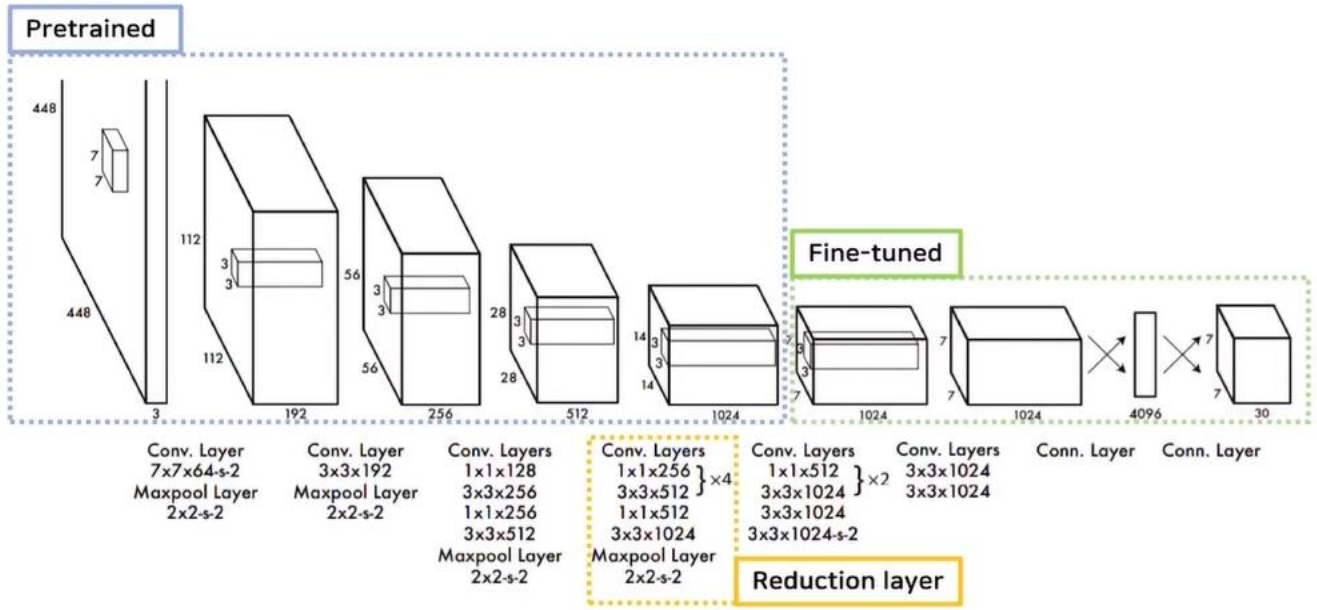
Grid cell 마다 bbox 2개씩 예측

Output tensor

Darknet 입·출력

Network Design – GoogLeNet

1x1 reduction layer:
https://zzsza.github.io/data/2018/05/14/cs231n-cnn/



- 24 conv layer + 2 fc layer / Fast Yolo: 9 conv layer + 2 fc layer
  - 20 conv layer : pretrained with 1000-class ImageNet (input image: 224 x 224)
  - 4 conv layer + 2 fc layer: fine-tuned with PASCAL VOC (input image: 448 x 448)
- 중간에 1 x 1 reduction layer 로 연산량 감소 (filter 개수를 input dim 보다 작게 하였을 때)

17

Darknet 입·출력

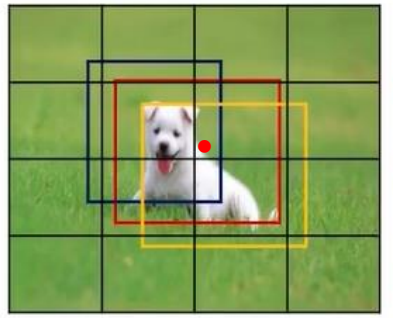
Training Stage

- 특정 object 에 responsible 한 cell i 는 GT box의 중심이 위치하는 cell 로 할당
- Yolo는 여러 bbox를 예측하지만, 학습단계에서는 IOU_{pred}^{truth} 가장 높은 bbox 1개만 사용

  $\Rightarrow \mathbb{1}_{ij}^{obj}$ 로 cell i 에서 responsible 한 j 번째 bbox 를 표시하여 loss function 에 반영

Input Image

Conv layers



Groundtruth
예측한 yellow bbox
예측한 blue bbox

-GT box 의 중심이 cell 6 에 위치 ← •이 있는 cell 번호

- 노란색 bbox 만 학습에 사용하고, $\mathbb{1}_{6,yellow}^{obj}$ = 1 로 표시

IoU를 기준으로

$$IOU_{yellow\ bbox}^{groundtruth} > IOU_{blue\ bbox}^{groundtruth}$$

Darknet 입·출력

Training Stage – Loss Function

- Mean Squared Error

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

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

- $\mathbb{1}_{ij}^{obj}$: if jth bbox predictor in cell i is responsible for prediction
- $\mathbb{1}_{i}^{obj}$: object appears in cell i
- $\lambda_{coord}, = 5$: bbox coordinates loss 반영 ↑
- $\lambda_{noobj} = 0.5$: no object 의 class probability loss 반영 ↓

- 모든 grid cell 에서 예측한 B개의 bbox 의 좌표와 GT box 좌표

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

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

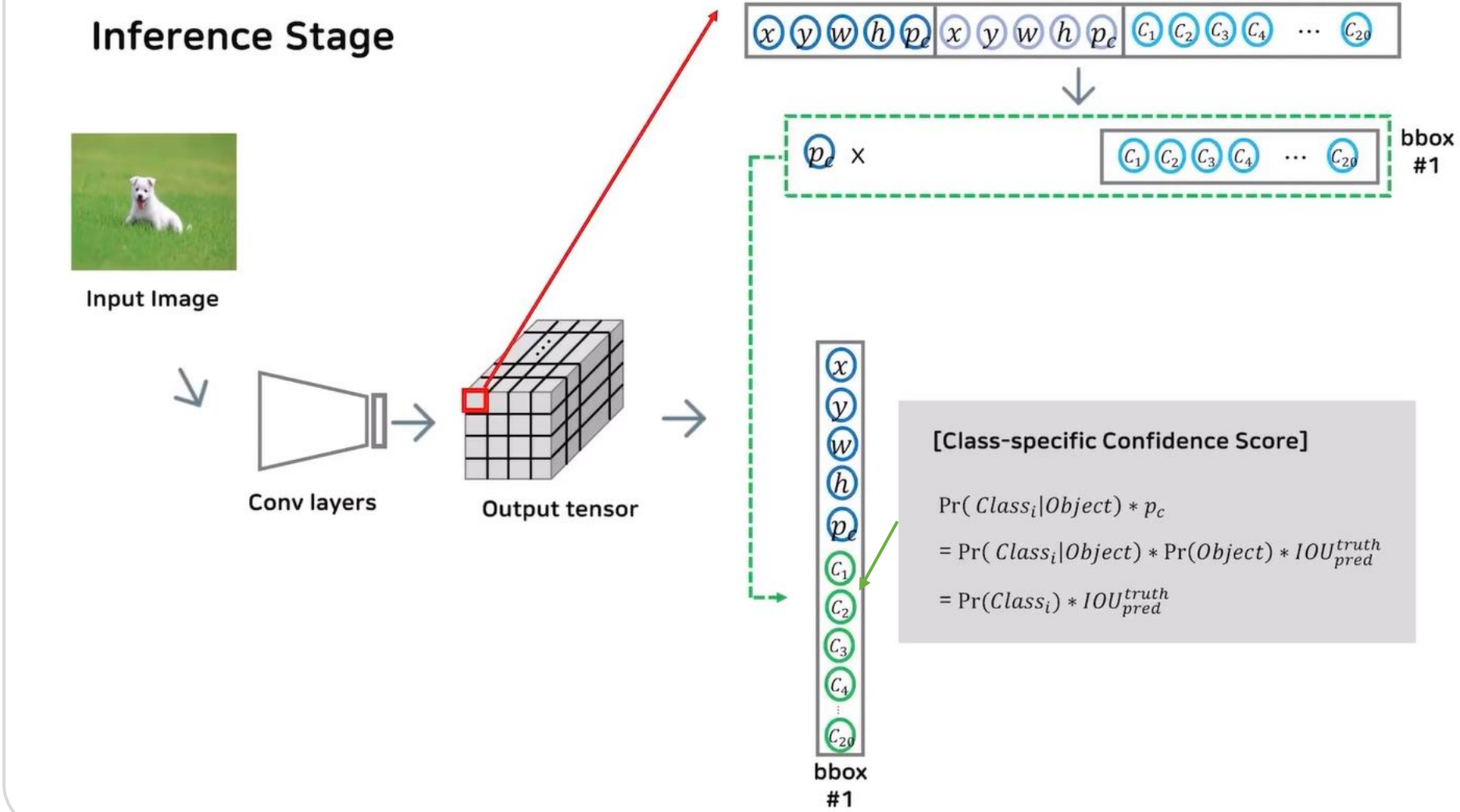
- 모든 grid cell 에서 예측한 B개의 $\Pr(Class_i|Object)$ 와 GT 값

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

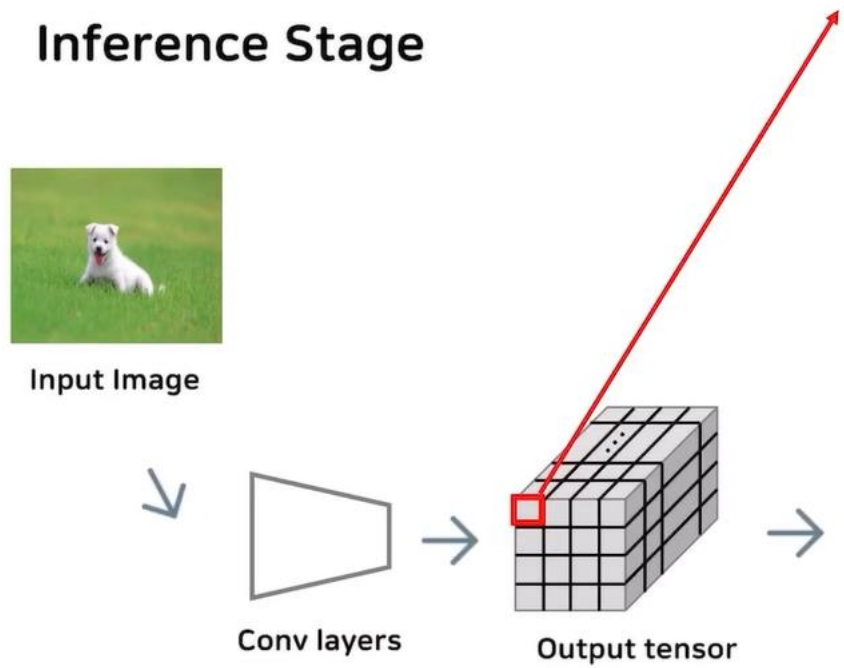
- 모든 grid cell의 $\Pr(Object) * IOU_{pred}^{truth}$ 예측값과 GT box 값

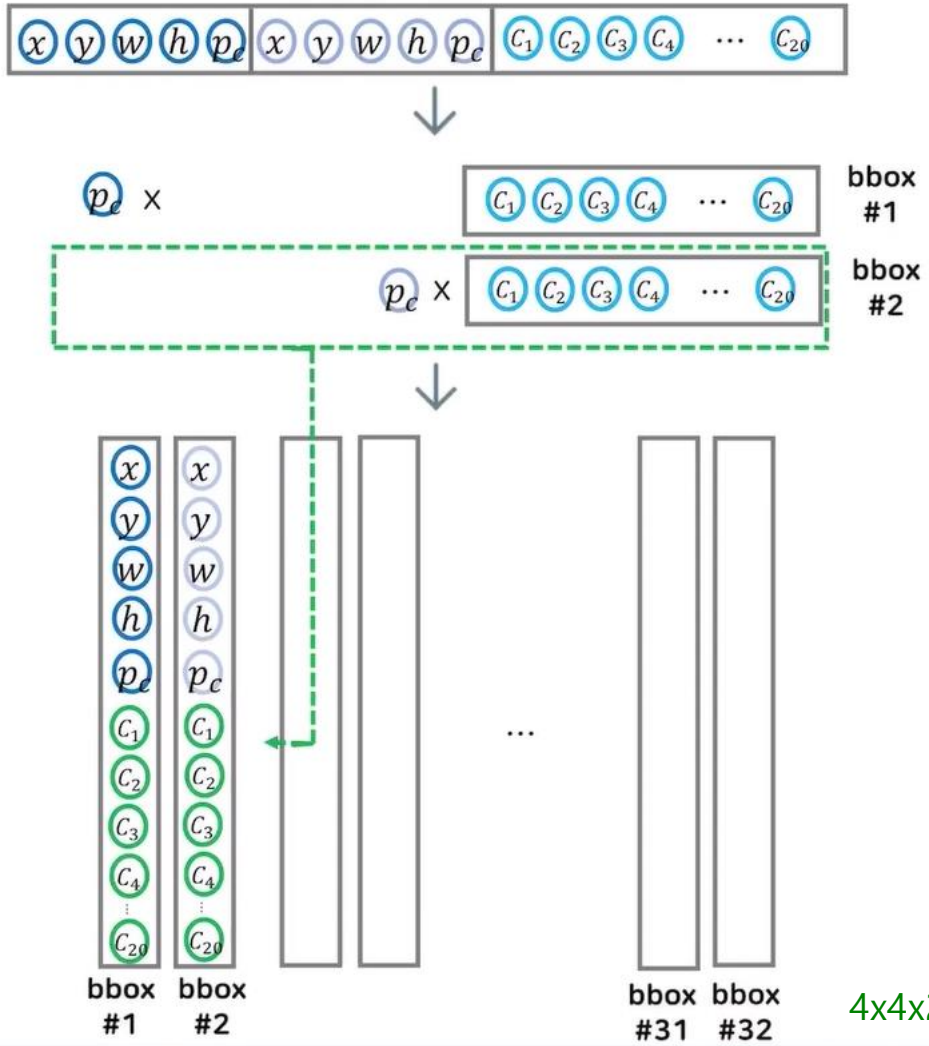
⇒ grid cell 에 object 존재하는 경우의 오차 & predictor box 로 선정된 경우의 오차만 학습

Darknet 입·출력


Inference Stage
Input Image
Conv layers
Output tensor
bbox #1
bbox #1
[Class-specific Confidence Score]
Pr(Class_i|Object) * p_c
= Pr(Class_i|Object) * Pr(Object) * IOU_pred^truth
= Pr(Class_i) * IOU_pred^truth

Darknet 입·출력



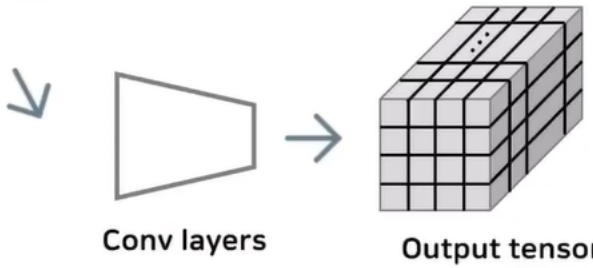


- Single network 로 detection 가능함
- PASCAL VOC dataset 기준, 이미지 1개당 98 개 bbox 생성 & 각 클래스에 대한 예측값 (grid cell: 7x7)
- object 당 bbox 개수가 많으므로 NMS 적용필요

√ 중복 Boxes 제거

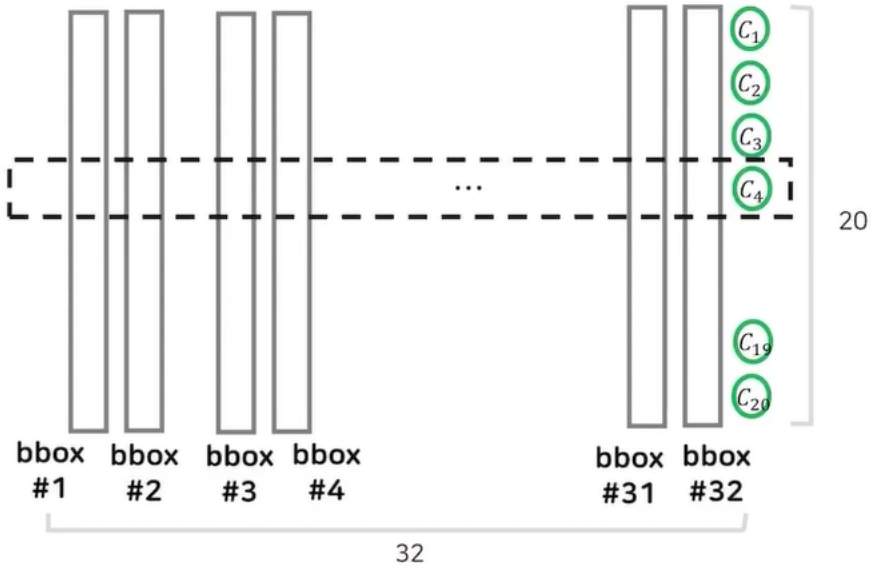
Inference Stage

Input Image



Non-Maximum Suppression

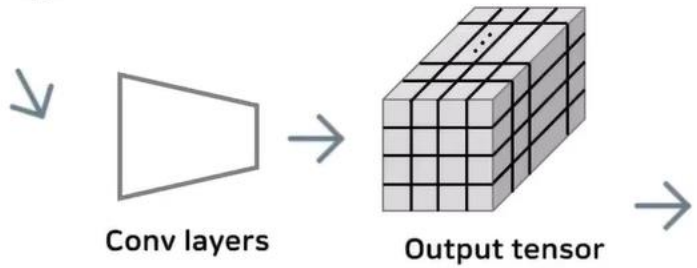
- 각 object에 대해 예측한 여러 bbox 중에서 가장 예측력 좋은 bbox만 남기기 위함



중복 Boxes 제거

Inference Stage

Input Image



Non-Maximum Suppression

- 각 object에 대해 예측한 여러 bbox 중에서 가장 예측력 좋은 bbox만 남기기 위함

- pseudo code

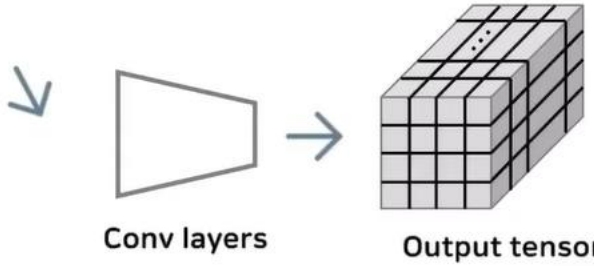
```
for i in range(len(class)):
  if bbox['p_c'] < θ:
    bbox_list.remove(bbox)

  while (not processed bbox exists):
    selected_bbox = bbox with the highest p_c
    bbox_list.remove(other boxes which has high IOU with selected bbox)
```

√ 중복 Boxes 제거

Inference Stage

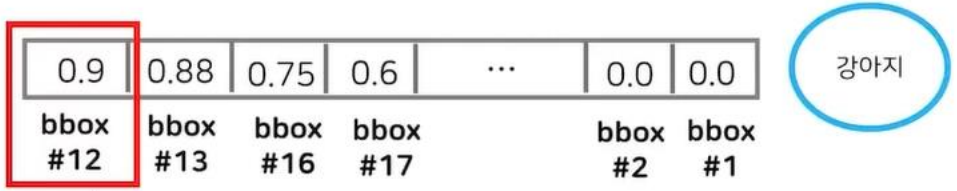
1) Object 가 1개인 경우

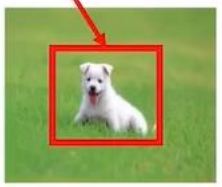
Input Image



Non-Maximum Suppression

예시)





- class: 강아지
- confidence: 0.9
- bbox: #12

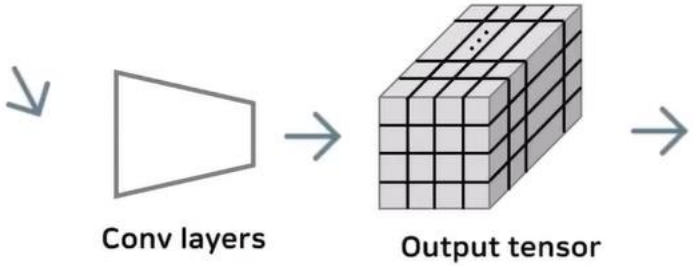
- Class 강아지를 가장 잘 예측하는 bbox는 #12 로 결정
- 나머지 bbox는 bbox#12 와의 IOU 가 높아서 NMS에 의해 모두 제거됨.

√ 중복 Boxes 제거

Inference Stage

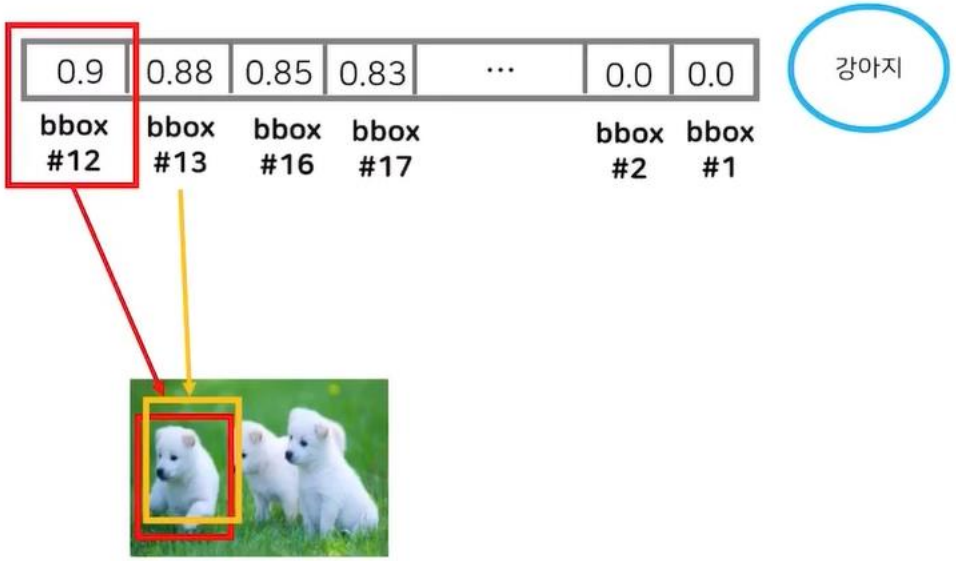
2) 같은 class 속하는Object 가 2개인 경우

Input Image



Non-Maximum Suppression

예시)



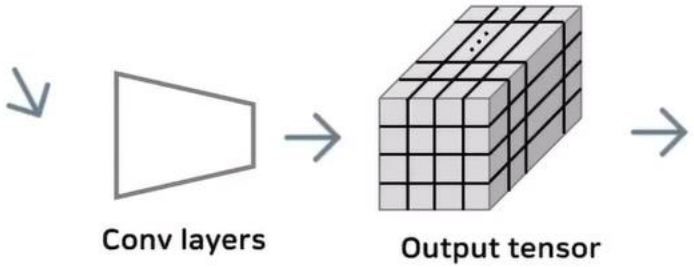
- Bbox#12 와 bbox#13의 IOU가 높으므로 NMS에 의해 제거됨.

√ 중복 Boxes 제거

Inference Stage

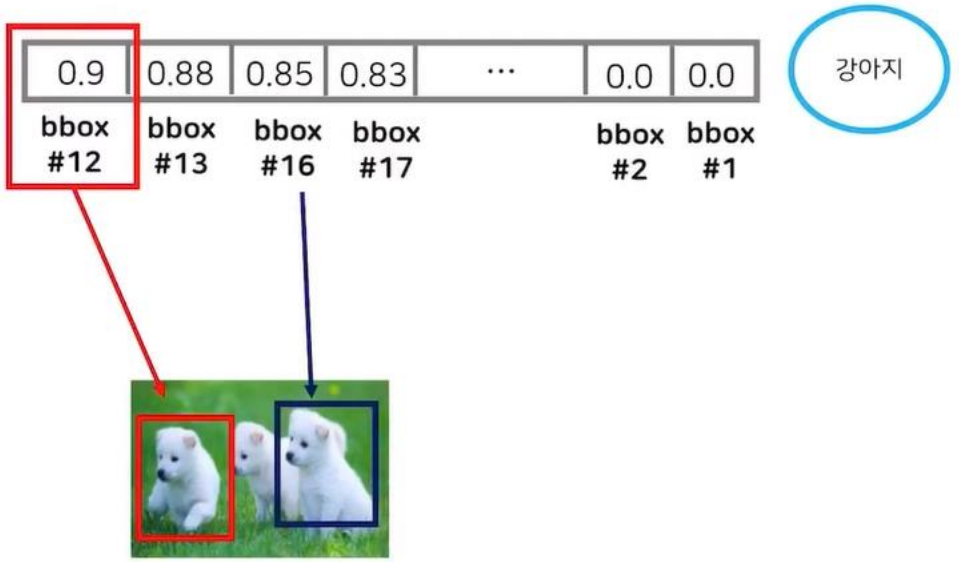
2) 같은 class 속하는Object 가 2개인 경우

Input Image



Non-Maximum Suppression

예시)



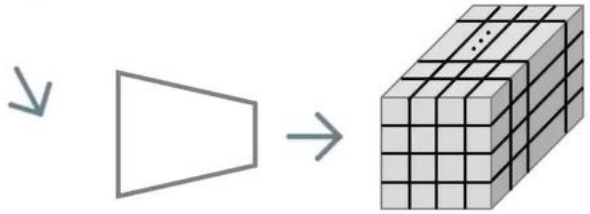
- Bbox#12 와 bbox#16의 IOU가 낮으므로 NMS에 의해 제거되지 않음.

√ 중복 Boxes 제거

Inference Stage

3) 다른 class 속하는 Object

Input Image

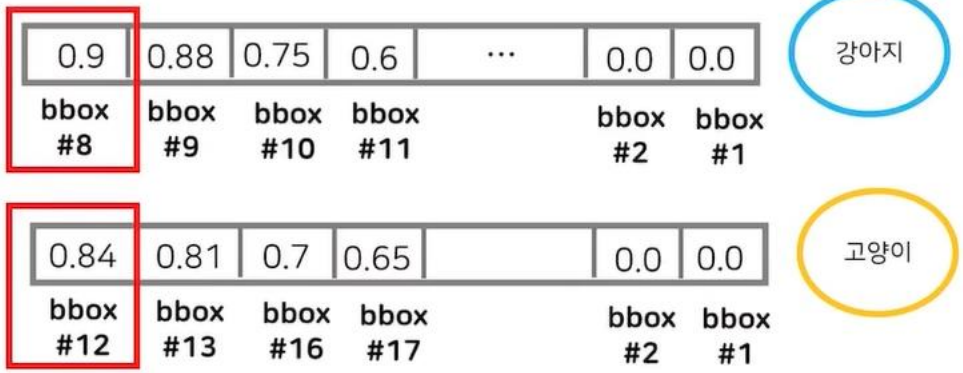


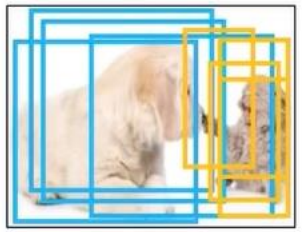
Conv layers

Output tensor

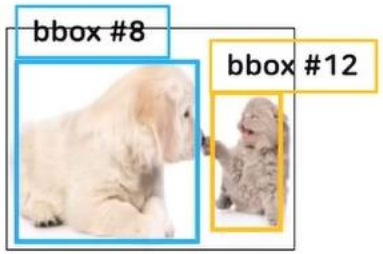
Non-Maximum Suppression

예시)





NMS 적용 전



NMS 적용 후

29

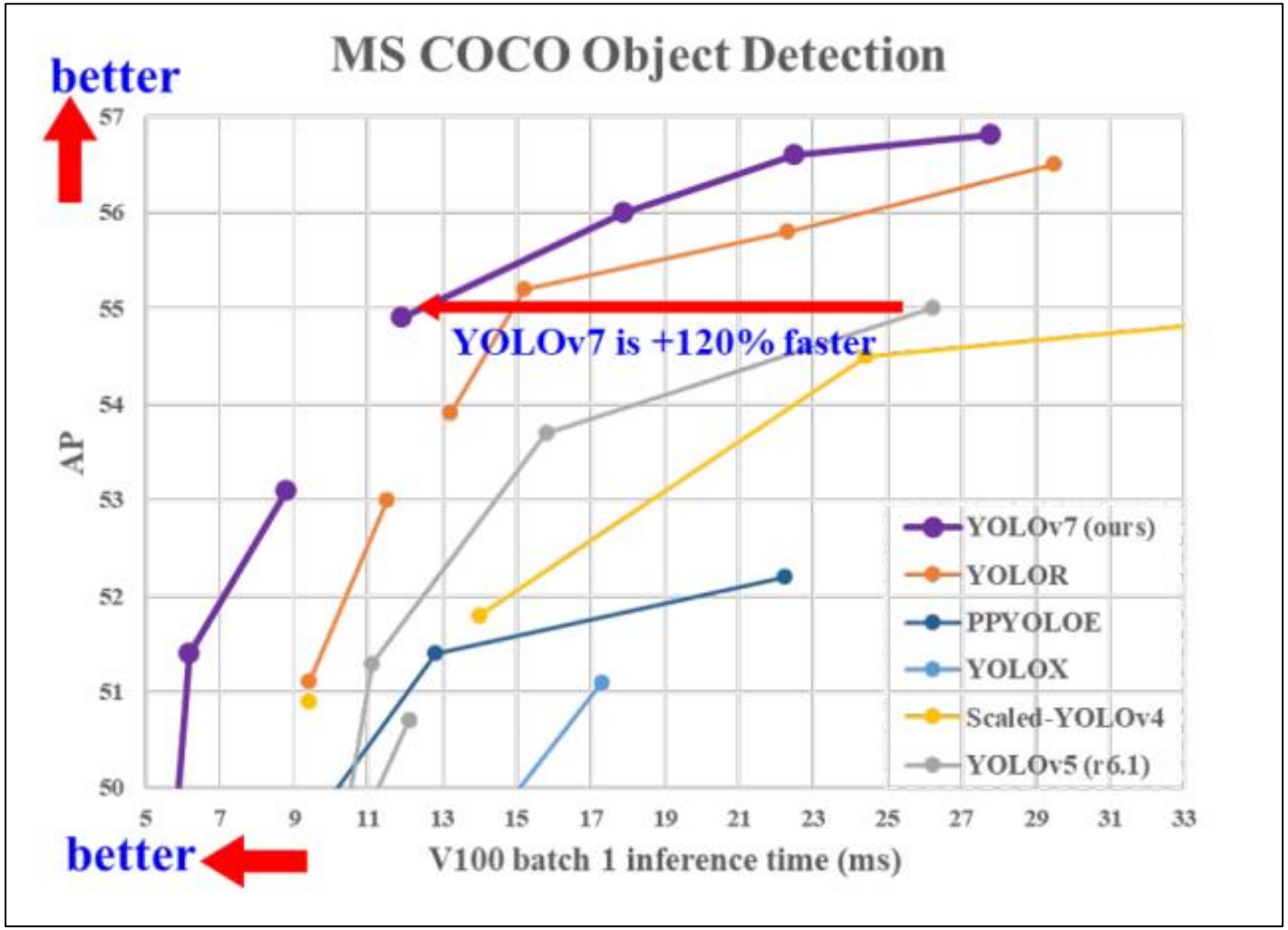
YOLO의 성능

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

- YOLO는 준수한 성능에 실시간 탐지가능한 속도 보장
- 이후 YOLO는 높은 정확도와 속도를 보장하는 실시간 객체 탐지 알고리즘으로 발전

YOLO를 활용한 F-16 전투기 탐지

YOLO 버전별 차이점 비교



YOLO 버전별 차이점 비교

- **YOLO v1 (2016)**: 실시간 객체 검출을 위한 딥러닝 기반의 네트워크
- **YOLO v2 (2017)**: v1에서 성능 개선 및 속도 향상
- **YOLO v3 (2018)**: 네트워크 구조와 학습 방법을 개선하여 Object Detection 정확도와 속도 개선
- **YOLO v4 (2020. 04)**: SPP와 AN 기술을 적용하여 Object Detection 정확도와 속도 개선
- **YOLO v5 (2020. 06)**: 전작보다 정확도 10% 이상 향상, 모델 크기 축소
- **YOLO v6 (2022. 07)**: 훈련 과정의 최적화, Trainable bag-of-freebies 제안
- **YOLO v7 (2022. 09)**: 알고리즘의 효율성 향상, 시스템 탑재를 위한 Quantization과 Distillation 방식 도입
- **YOLO v8 (2023. 01)**: 새로운 저장소를 출시하여 객체 감지, 인스턴스 세분화 및 이미지 분류 모델 Train을 위한 통합 프레임 워크로 구축

Main contribution

1.Object detection을 regression problem으로 관점 전환

2.Unified Architecture: 하나의 신경망으로 classification & localization 예측

3.DPM, RCNN 모델보다 속도 개선

4.여러 도메인에서 Object detection 가능

YOLO의 한계

1.각 Grid Cell마다 8개의 Bounding box만 추측해야 한다는 공간적 제약성이 있어 가까이 붙어있는 물체를 판별하기 어려움

2.물체가 작으면 bbox 간의 IoU 값이 차이가 작아서 근소한 차이로 클래스가 결정되기 때문에 탐지 성능이 낮음

3.부정확한 위치 판별(Localization)

4.데이터로부터 Bounding box를 훈련시키기 때문에, 학습 데이터에 없는 물체는 검출이 어려움

질의 및 응답

출처

[1] Redmon, Joseph, et al. “You only look once: Unified, real-time object detection.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] YOLO CVPR 2016 발표자료, https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_57

[3] https://www.youtube.com/watch?v=O78V3kwBRBk