

YOLO

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

Object Detection

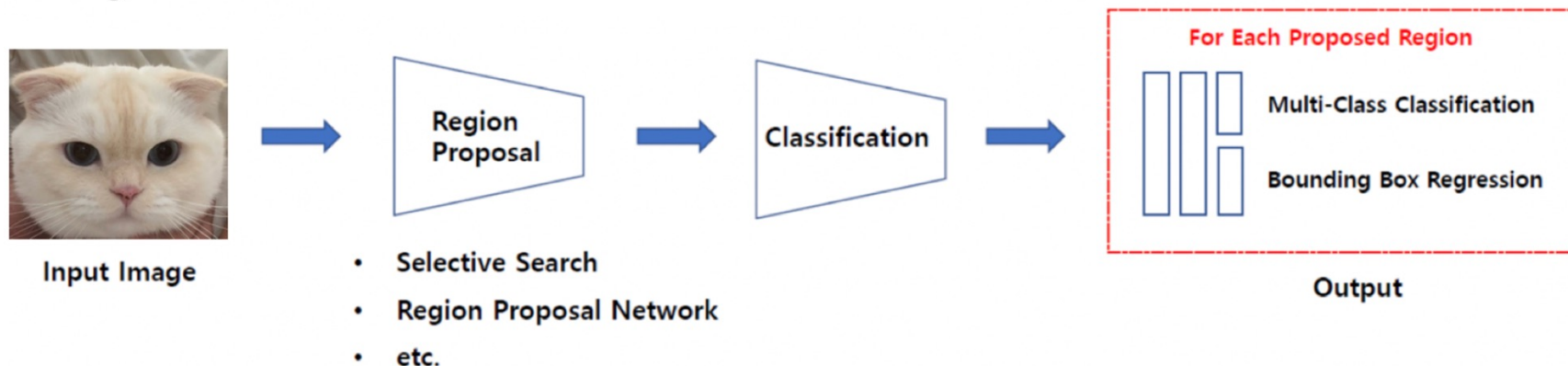
Object Detection : 물체가 있는 영역의 위치정보를 Bounding Box로 찾고 Bounding Box내에 존재하는 사물의 라벨(Label)을 분류하는 문제



Classification + Localization

2-Stage vs 1-Stage

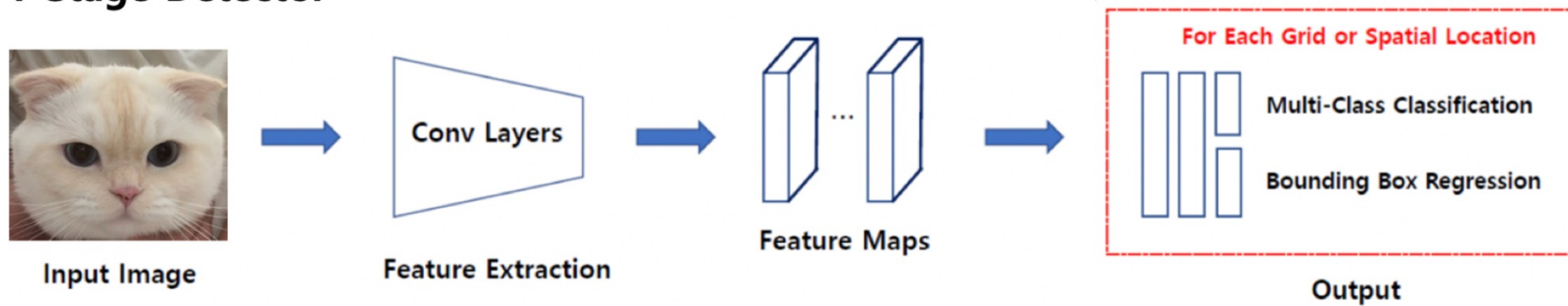
2-Stage Detector



Ex) **R-CNN 계열** (R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, Mask R-CNN ...)

1. 정확하다
2. 느리고 최적화가 어렵다

1-Stage Detector




Ex) **YOLO 계열** (YOLO v1, v2, v3) , **SSD 계열** (SSD, DSSD, DSOD, RetinaNet, RefineDet ...)

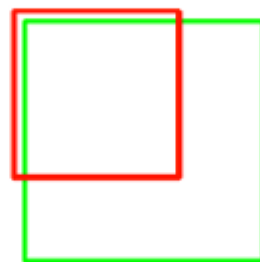
빠르지만 상대적으로 정확도(mAP)가 떨어진다.

Intersection over Union(IoU)

- IoU는 두 가지 물체의 위치(Bounding Box)가 얼마나 일치하는지를 0~1 사이의 값으로 나타내는 지표
- 2개의 Bounding Box가 일치할 수록 1에 가까운 값이 되고, 일치하지 않을수록 0에 가까운 값이 나온다.

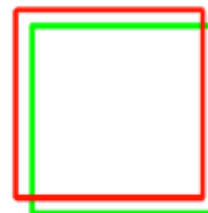
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


IoU: 0.4034



Poor

IoU: 0.7330



Good

IoU: 0.9264



Excellent

Precision, Recall

- **Precision(정밀도/정확도)** : 전체 검색된 결과들 중 제대로 검색된 결과물의 비율

→ 올바르게 검출한 수 / 전체 검출한 수

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ Detections}$$

- **Recall(재현율/검출율)** : 검색 되어야할 항목들 중 실제 검색된 결과물들의 비율

→ 올바르게 검출한 수 / 실제 물체의 수

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ Ground\ truths}$$

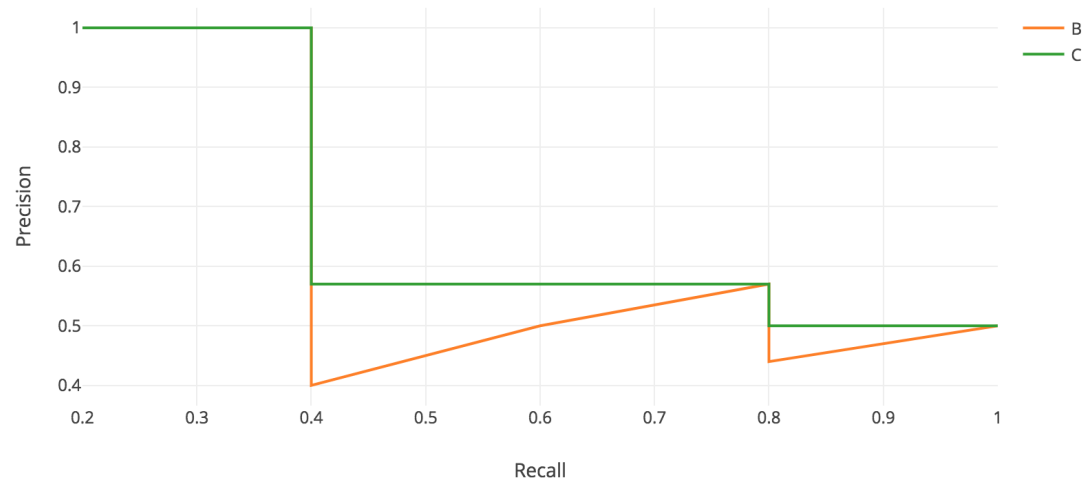
e.g.) 이미지 내에 강아지가 20마리 존재, 이때 모델이 10마리의 강아지를 검출하고, 5마리는 정확히 맞추었다고 하면 $Precision = 5 / 10 = 0.5$, $Recall = 5 / 20 = 0.25$

- **Positive 판단의 기준** : 일정 IoU 임계치 값을 넘기면 맞춘 것으로 간주

Precision, Recall

- **Average Precision(AP)** : Precision-Recall curve의 아래쪽 면적 크기
- **Mean Average Precision(mAP)** : Class들의 Average Precision의 평균

Rank	Correct	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0



사람처럼 어디에 무엇이 있는지 한 번에 판단하자!

- Regression Problem
- Single Network
- Fast
- Less false positive on background
- General representations + New Domain

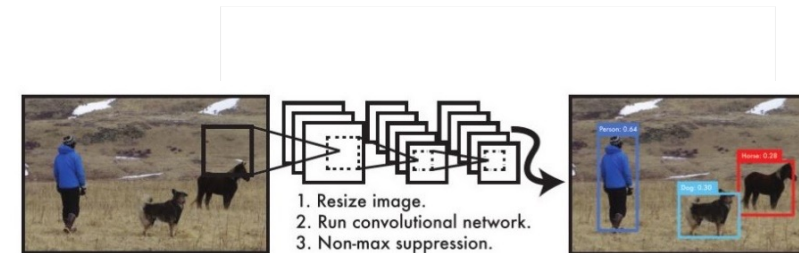


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

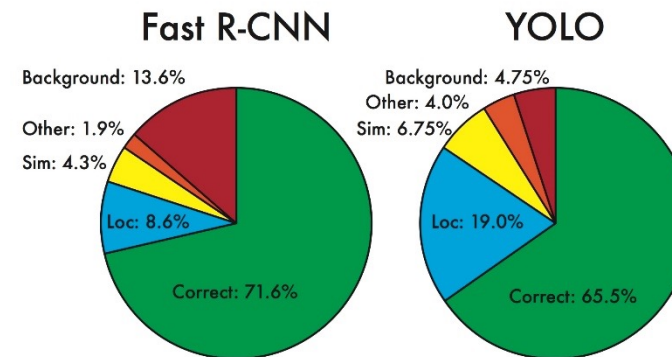
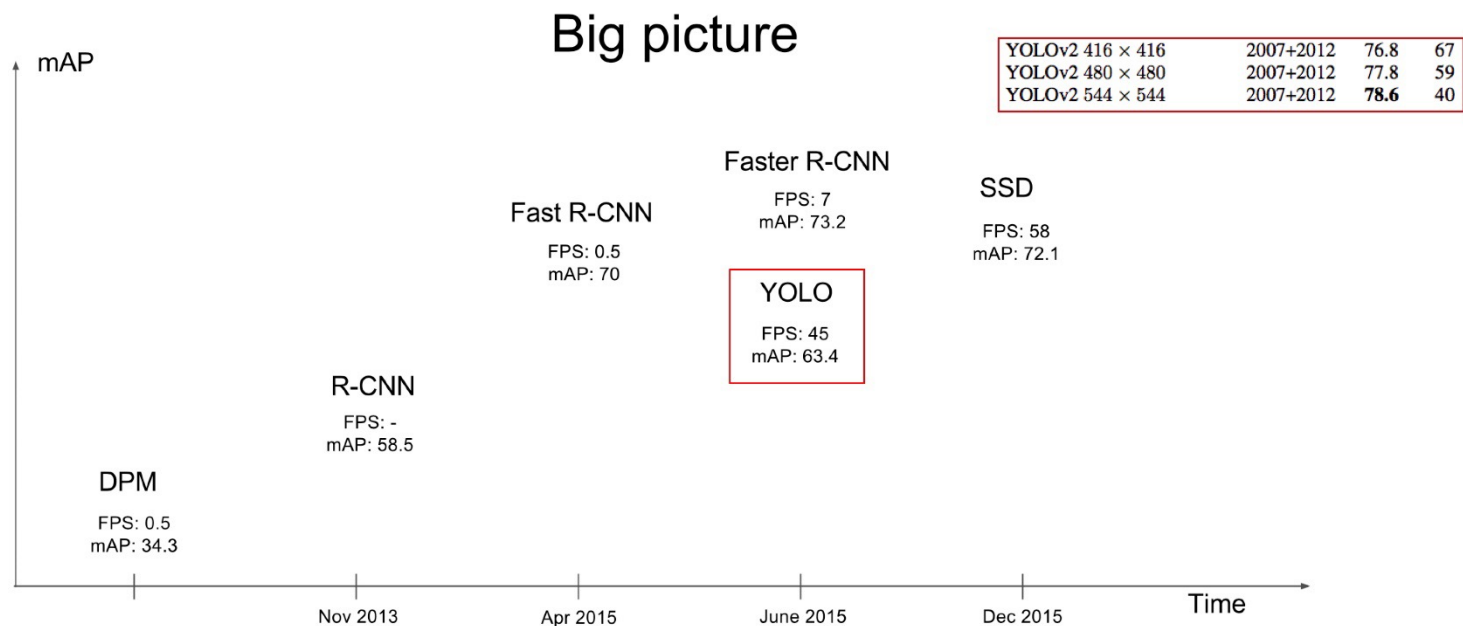


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories ($N = \#$ objects in that category).

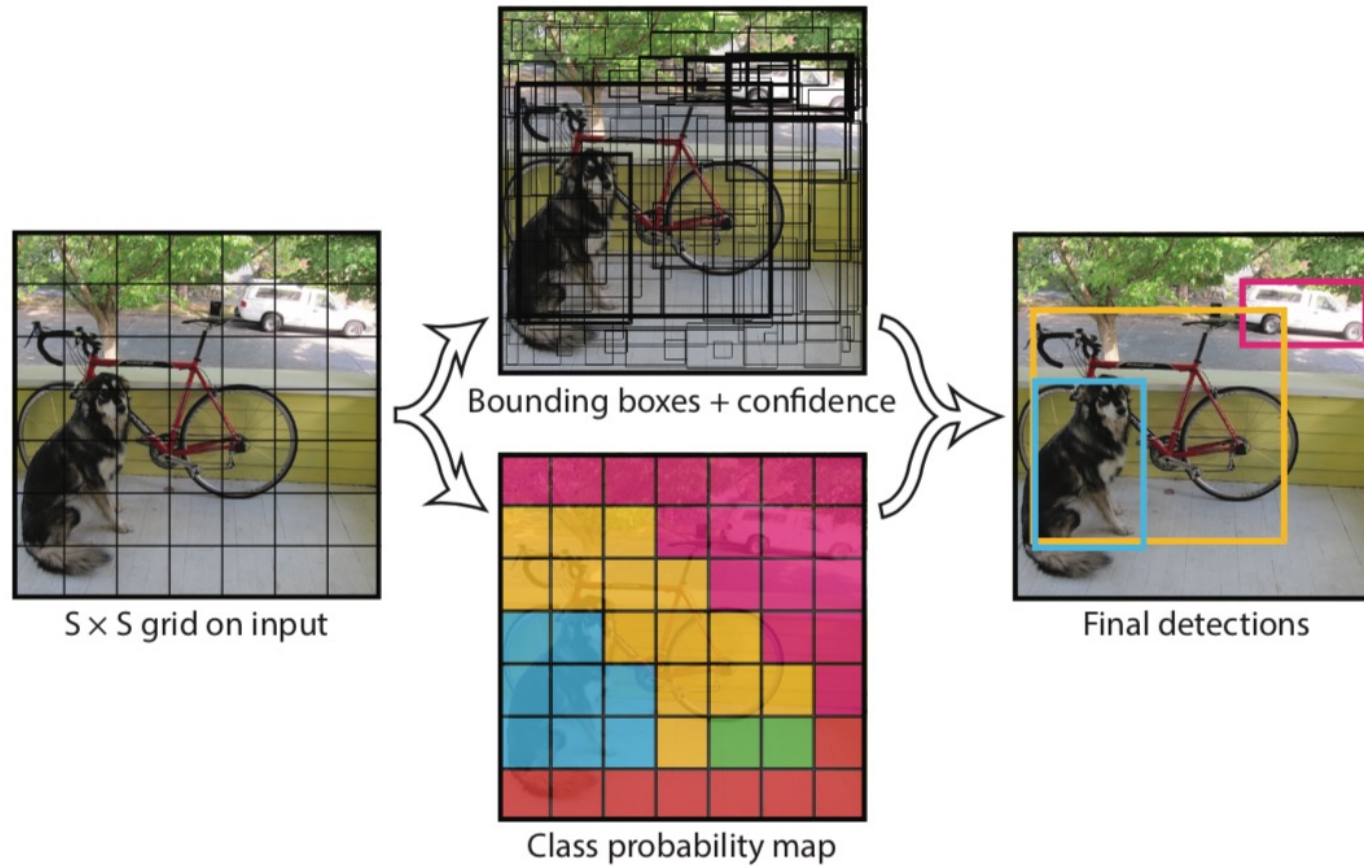
Evaluation on VOC2007



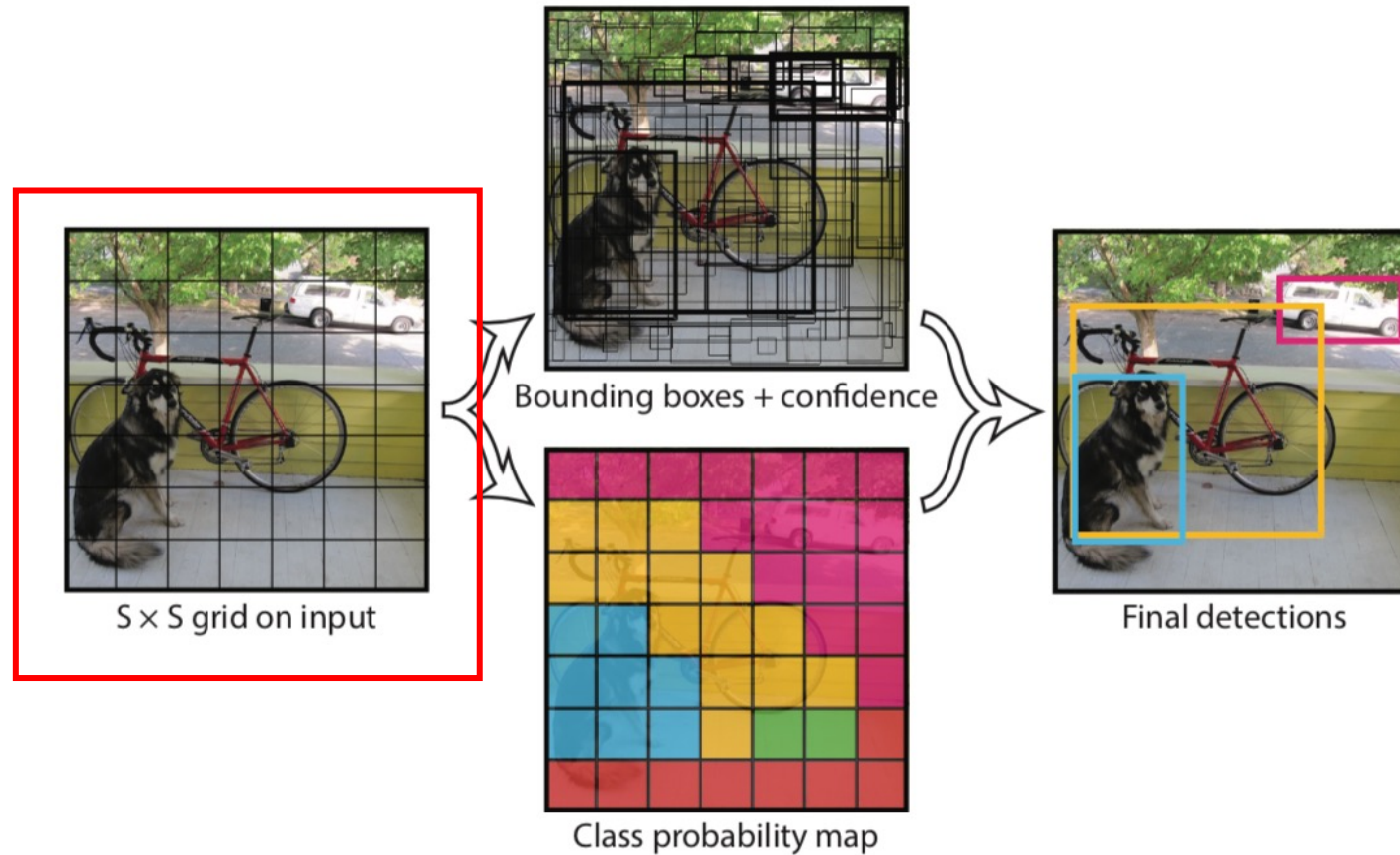
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7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

Slide courtesy of DeepSystem.io “[YOLO: You only look once Review](#)”

Unified Detection



Unified Detection



1. 입력 이미지를 $S \times S$ 개의 grid cell 나눔

(논문, $S = 7$)

Unified Detection

2. 각각의 Grid cell은 **B개의 Bounding box**와 그 Bounding box에 대한 **Confidence score**를 예측

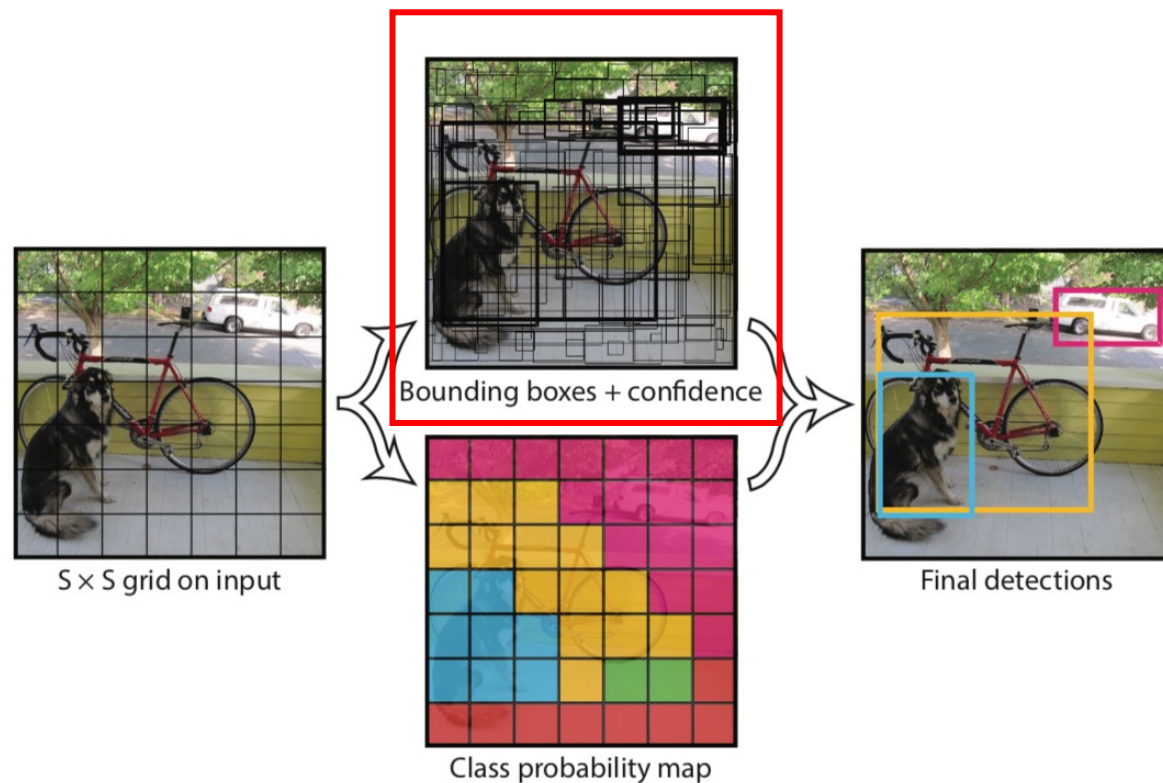
$$\Pr(Object) * IOU_{pred}^{truth}$$

3. 그리드 셀마다 C개의 Conditional class probability 계산 → 가장 높은 확률의 Class 할당

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

(논문, B = 2)

Bounding boxes : box의 중심 x, y좌표, 넓이(W), 높이(h)



Unified Detection

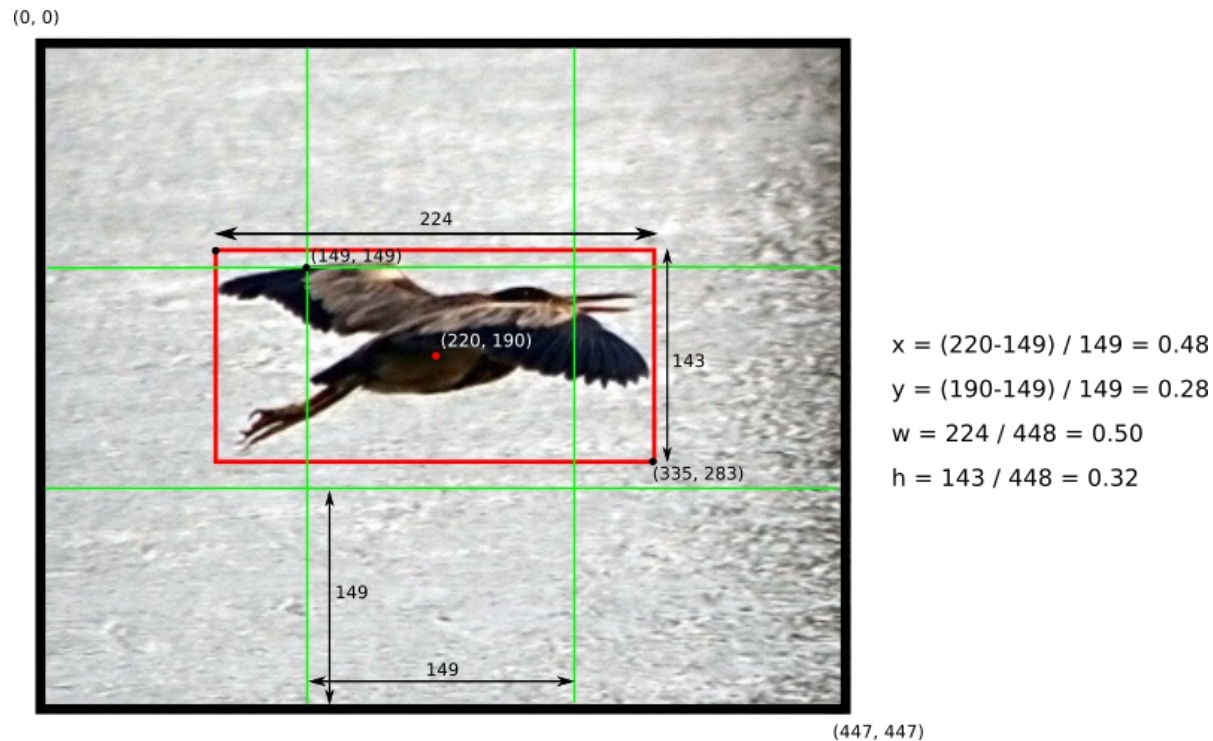
각각의 Bounding box는 **x, y, w, h** 그리고 **confidence**를 지녔다. (5개의 예측값을 가진다.)

(x, y): Bounding box의 중심점, Grid cell 범위에 대한 상대 값 (0 ~ 1사이로 정규화)

e.g) x, y 좌표가 Grid cell의 정가운데 중심점에 존재한다면 $x=0.5, y=0.5$

(w, h): 전체 이미지의 width, height에 대한 상대 값 (0 ~ 1사이로 정규화)

e.g) Bounding box의 width, height가 전체 이미지의 절반이라면 $w=0.5, h=0.5$



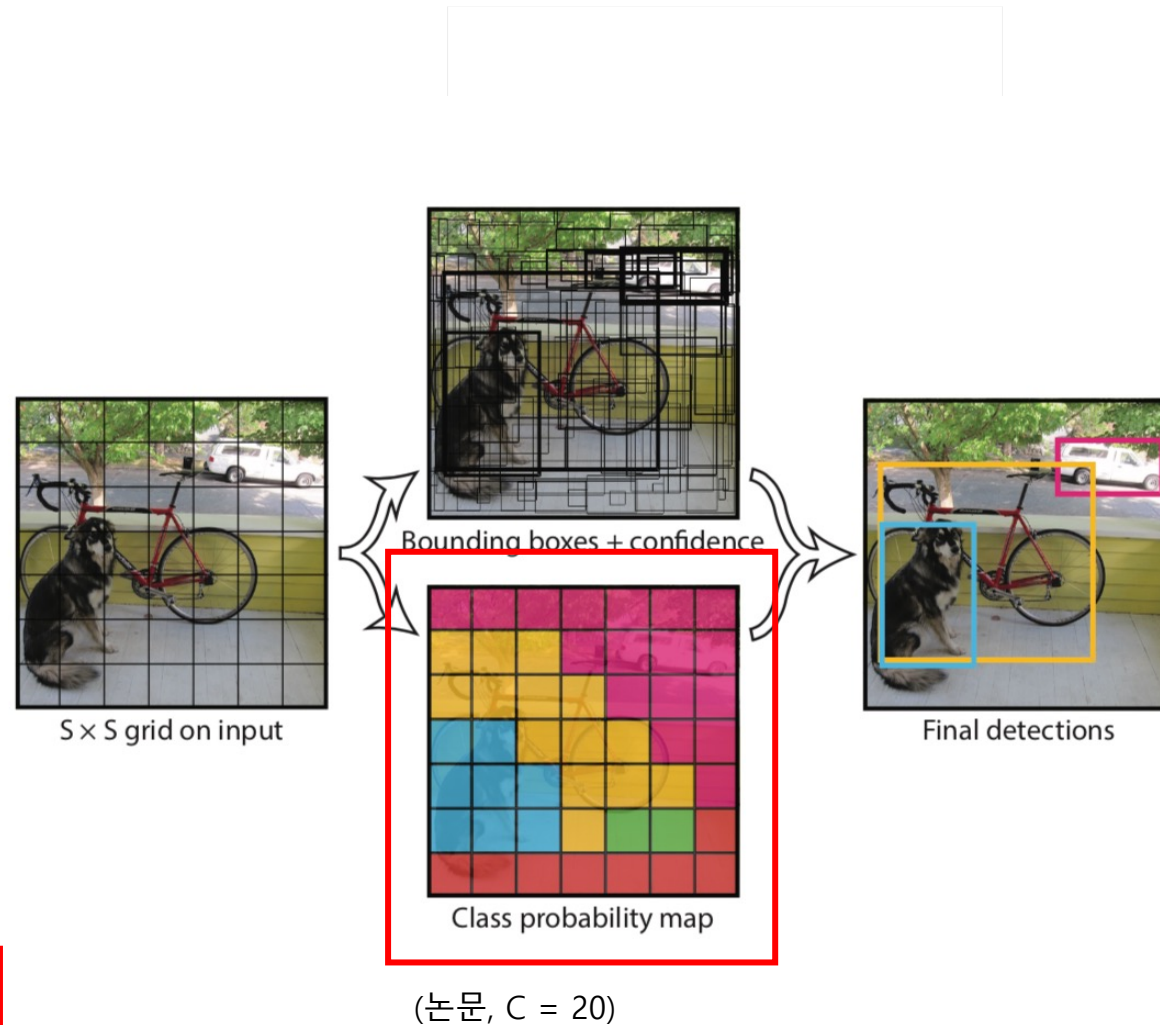
Unified Detection

2. 각각의 Grid cell은 B개의 Bounding box와
그 Bounding box에 대한 Confidence score를 예측

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

3. 그리드 셀마다 **C개의 Conditional class probability**
계산 → 그리드 셀마다 가장 높은 확률의 Class 할당

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



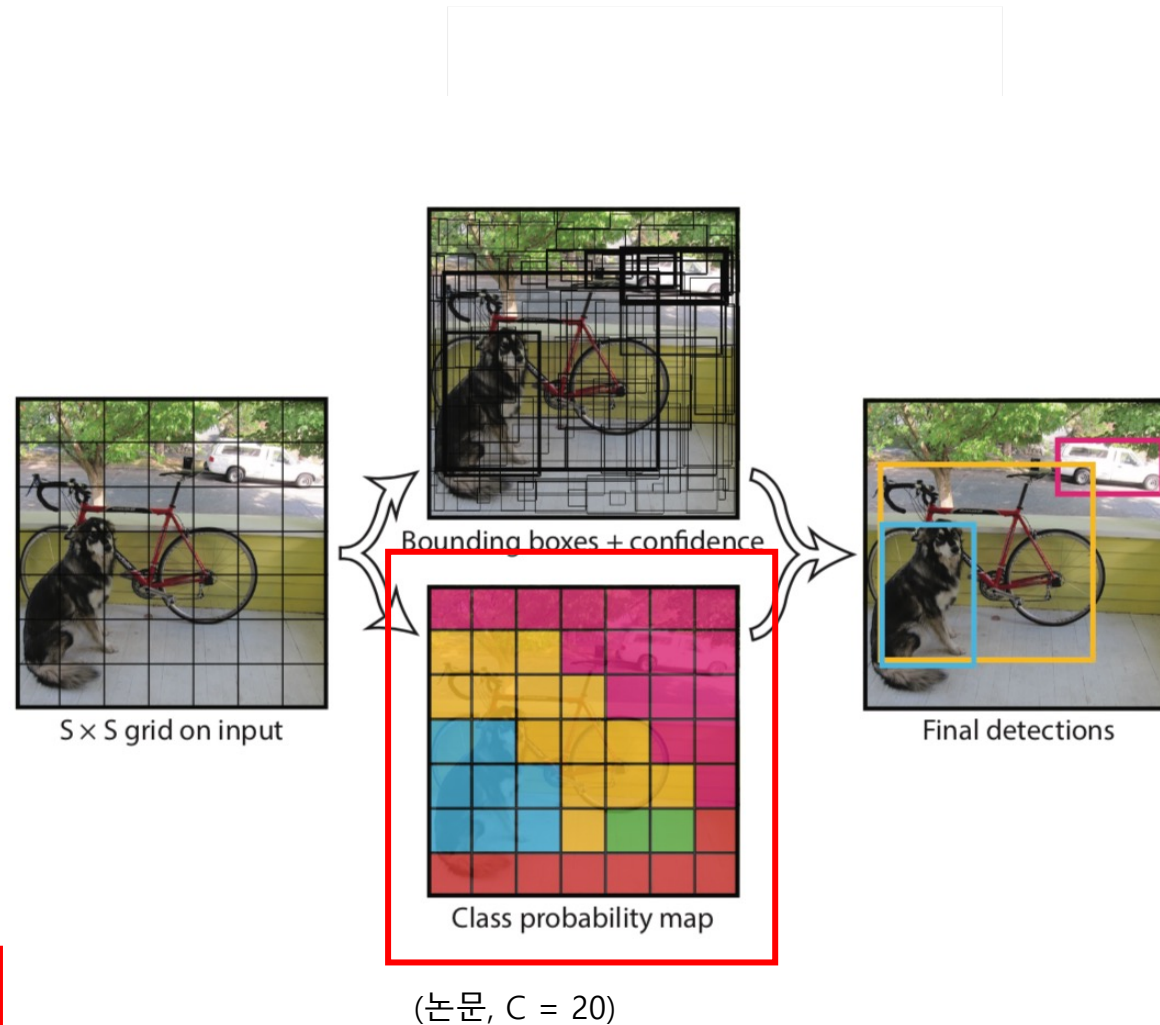
Unified Detection

2. 각각의 Grid cell은 B개의 Bounding box와
그 Bounding box에 대한 Confidence score를 예측

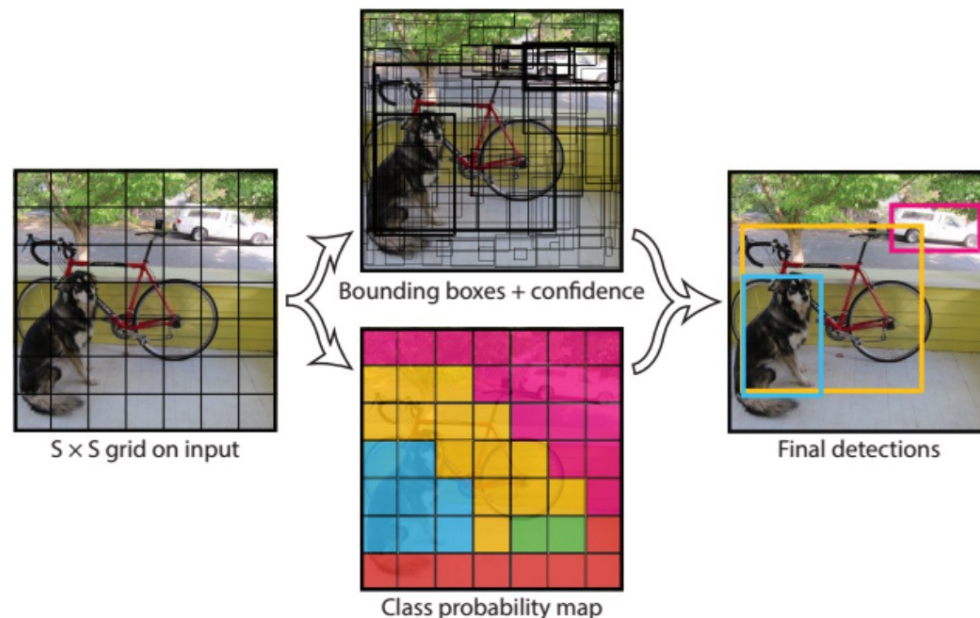
$$\Pr(Object) * IOU_{pred}^{truth}$$

3. 그리드 셀마다 **C개의 Conditional class probability**
계산 → 그리드 셀마다 가장 높은 확률의 Class 할당

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



Unified Detection



Hyperparameters :

$S : 7$

$B : 2$

$C : 20$ (Class의 수)

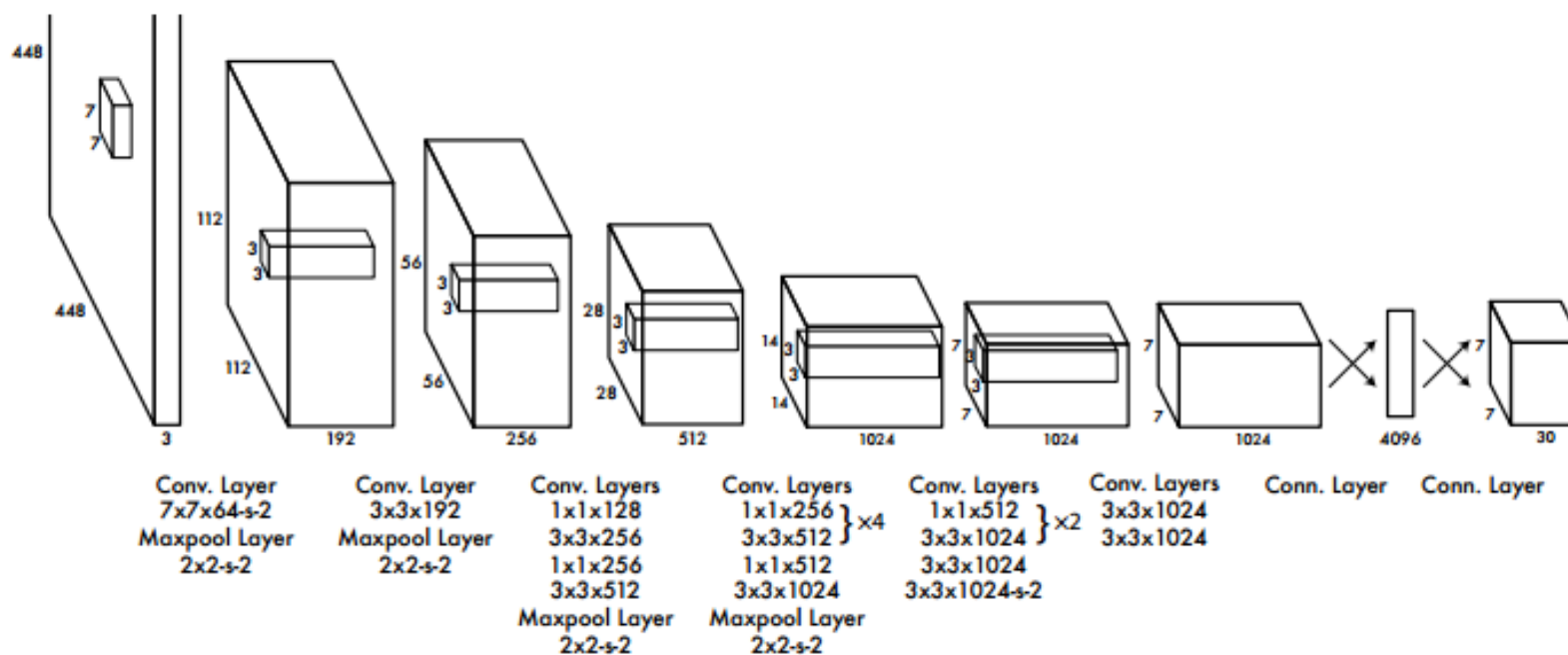
Output :

$7 \times 7 \times 30$

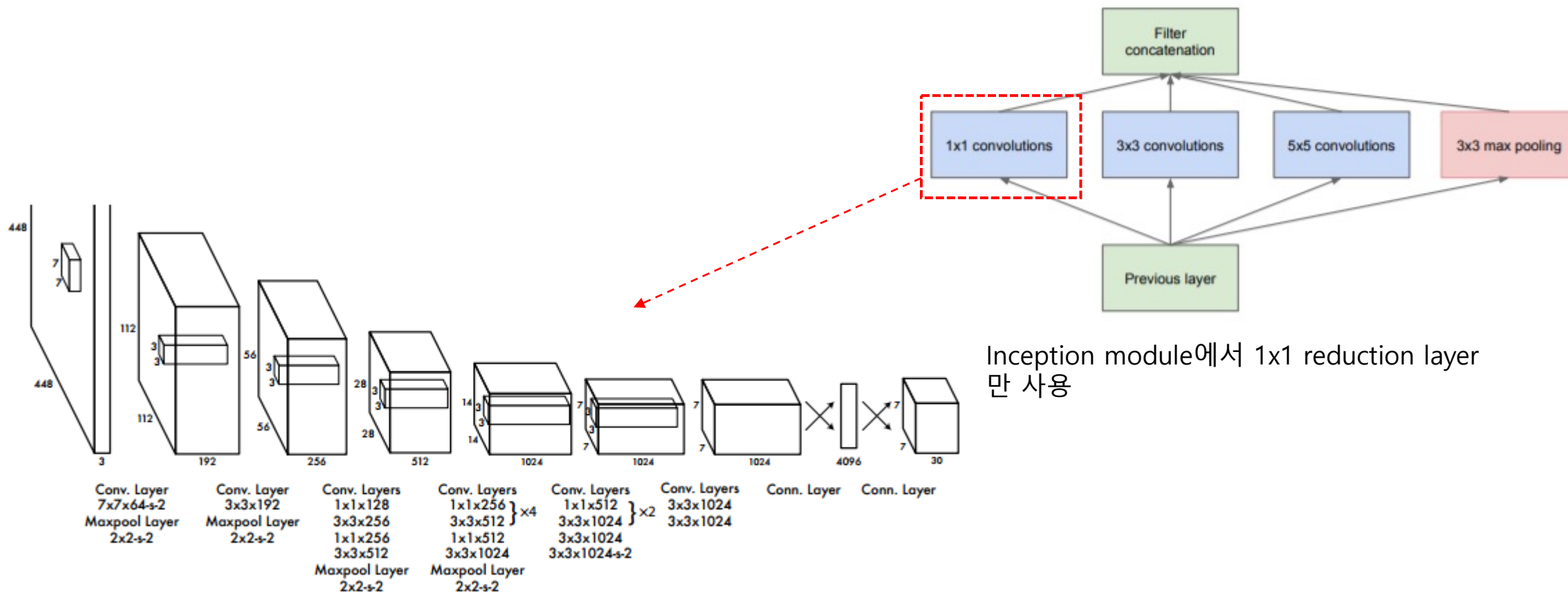
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Network Design

- YOLO의 네트워크 구조는 **GoogLeNet**을 기반으로 한다.
- 24개의 Convolutional Layer + 2개의 Fully Connected Layer
(Fast YOLO에서는 9개의 Convolutional Layer 사용)



Network Design



Inception module에서 1x1 reduction layer
만 사용

GoogLeNet 모델 기반으로 생성

Network Design - Training

- 20개의 Conv layer는 1000-class ImageNet으로 pretrain
- 해상도를 위해 input size 변경 (224 -> 448)
- 마지막 layer를 제외한 모든 layer는 Leaky ReLU 사용

Feature Extractor

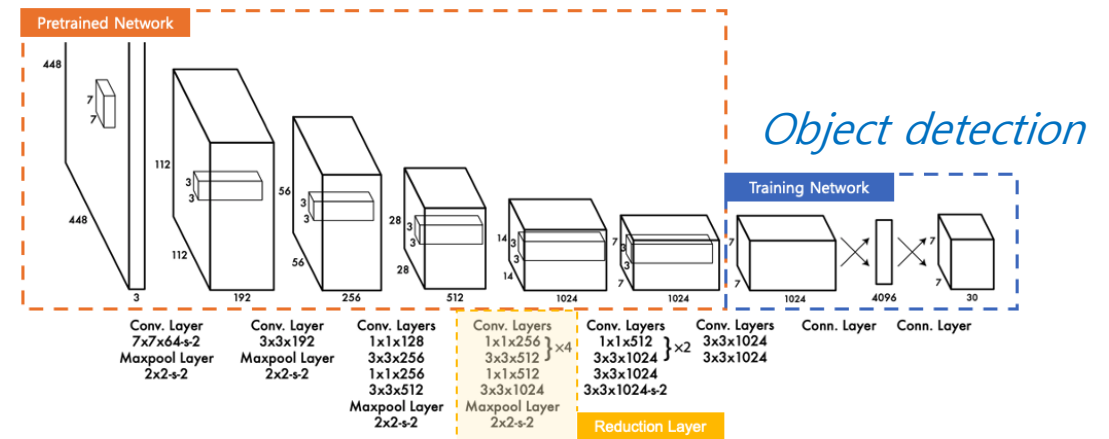
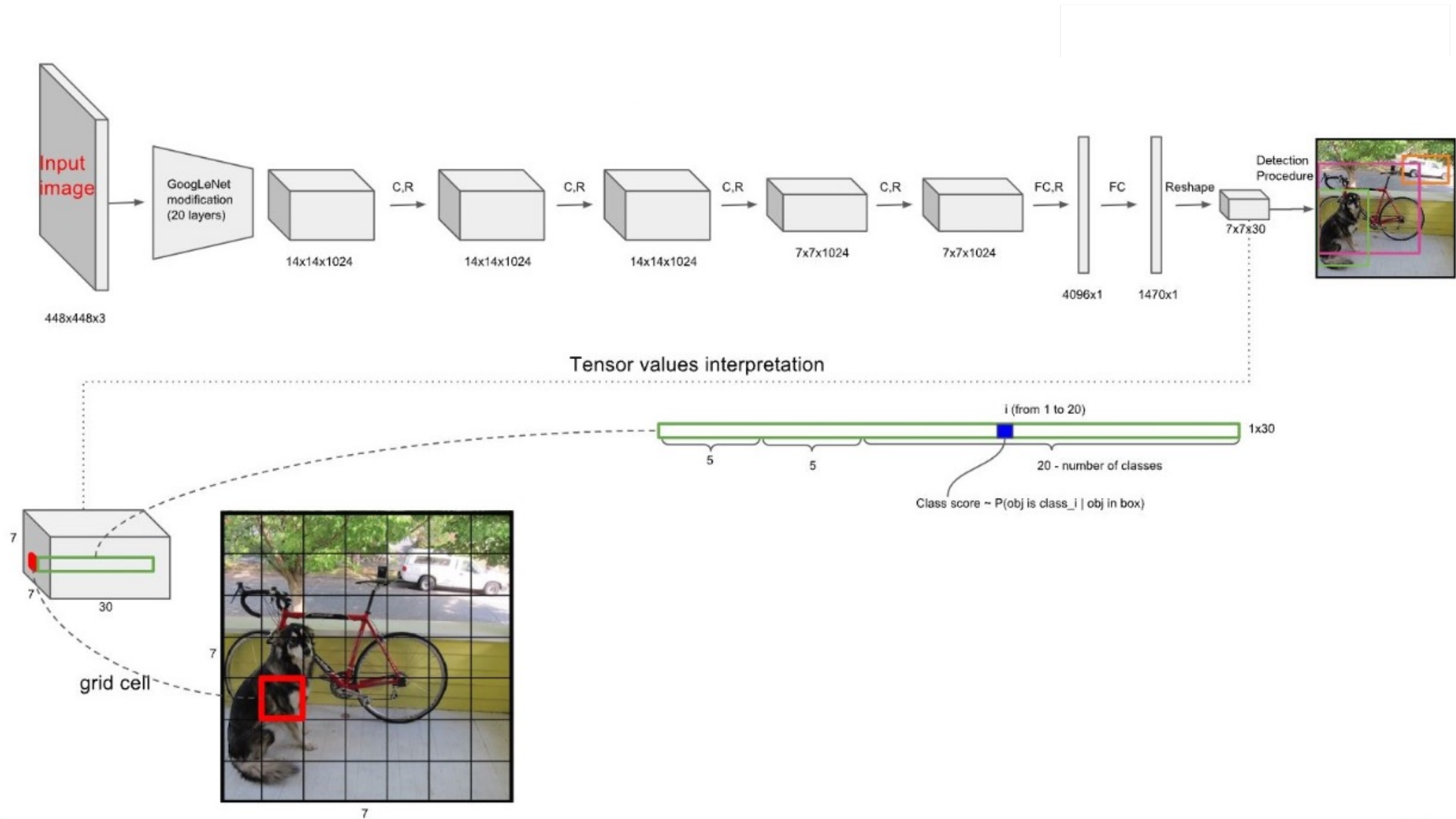
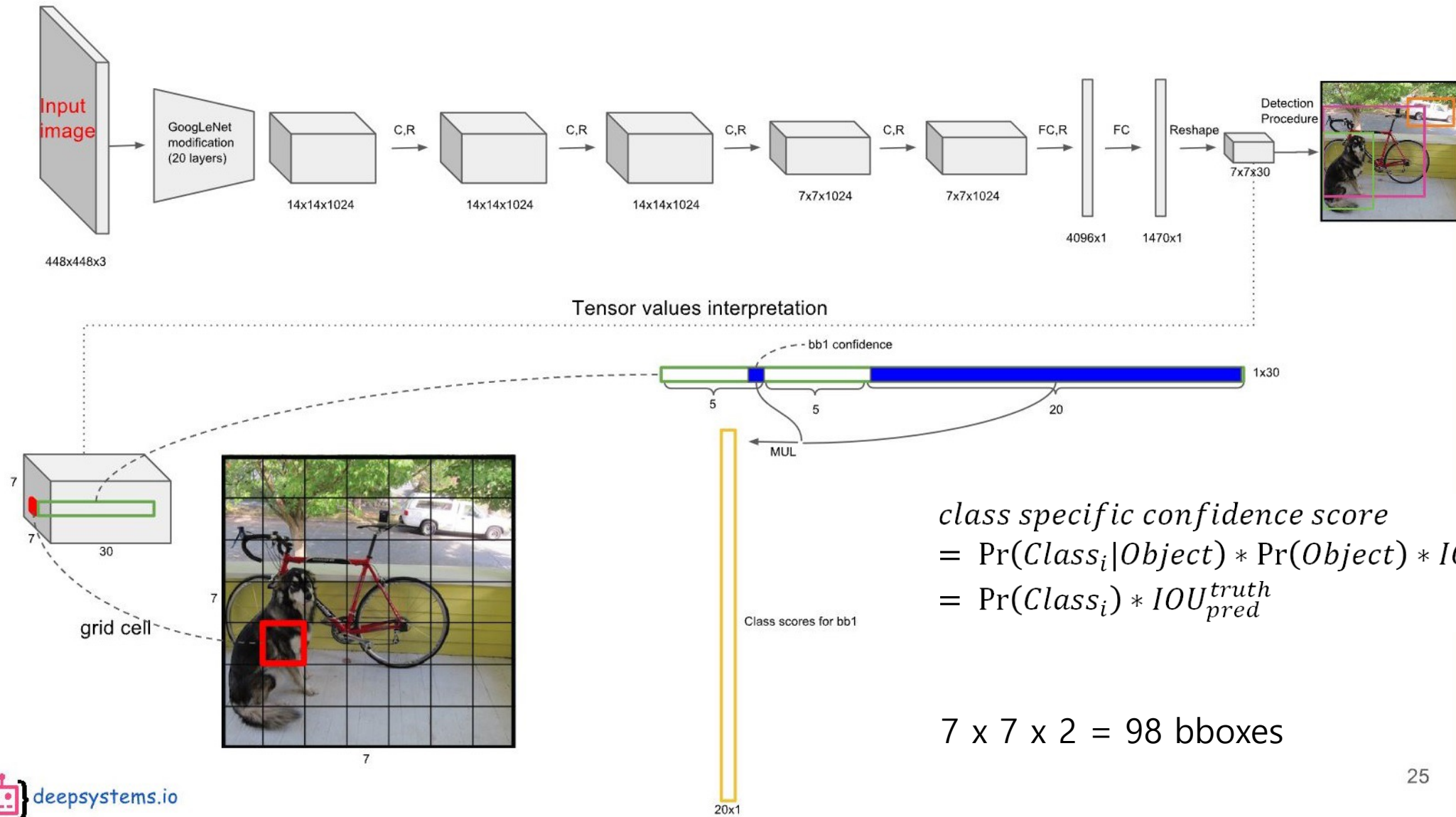


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

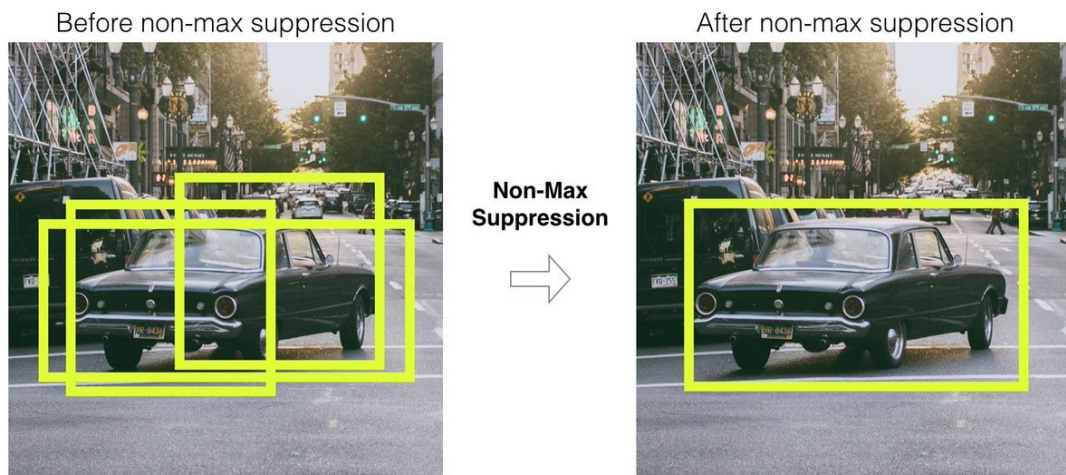
Network Design - Output



Network Design - Output



NMS(Non-Maximum Suppression)



Class별 NMS 수행

1. 특정 Confidence 임계값 이하의 Bbox는 모두 제거
2. 이 중 가장 높은 Confidence값을 가진 Bbox와 다른 모든 Bbox를 비교해서 IOU값을 계산
3. 지정한 IOU Threshold(임계값)보다 큰 Bbox는 모두 제거
4. 위 과정을 반복

Loss Function

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

Loss Function

Multi-Loss Function

Localization loss

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

Confidence loss

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

Classification loss

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

Loss Function

- SSE(Sum-Squared Error) 기반의 Multi-loss function
- 높은 IOU 수치의 Bounding box만 최적화

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

Loss Function

object가 존재하는 grid cell i의 predictor bounding box j
object가 존재하면 1, 존재하지 않으면 0

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

object가 존재하는 grid cell i의 predictor bounding box j
에 대한 bounding box 중심의 제곱오차

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

object가 존재하는 grid cell i의
predictor bounding box j에 대한
bounding box 크기의 오차

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

object가 존재하는 grid cell i의 predictor
bounding box j에 대한 confidence score 오차

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

object가 존재하지 않는 grid cell i의
bounding box j
object가 존재하면 0, 존재하지 않으면 1

Object가 존재하는 grid cell i

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

Object가 존재하는 grid cell i에 대해
conditional class probability 오차

Loss Function

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

λ_{coord} : x, y, w, h loss의 균형을 위한 parameter (default : 5)

λ_{noobj} : confidence loss의 균형을 위한 parameter (default : 0.5)

$\mathbb{1}_{ij}^{\text{obj}}$: Object 존재하는 그리드 셀의 BBox

$\mathbb{1}_{ij}^{\text{noobj}}$: Object 존재하지 않는 그리드 셀의 BBox

$\mathbb{1}_i^{\text{obj}}$: Object 존재하는 그리드 셀

Bounding box의 크기가 클 때보다 작을 때 민감하게 작용하도록 제곱근을 씌워 줌

Limitation

- 각 그리드 셀 마다 여러 개의 Bounding Box를 예측하지만 정작 모든 Bounding Box가 1개의 Class 밖에 가질 수 없기 때문에 여러 개의 객체가 존재하거나 근접한 작은 물체에 대해 잘 탐지하지 못함
- Bounding Box 형태가 주어진 데이터를 통해서만 학습되므로, 학습 데이터에 존재하지 않는 새로운/독특한 형태의 Bounding Box의 경우 정확히 예측하는데 어려움이 있음
- Bounding box를 예측하는데 있어, localization이 다소 부정확하다. 큰 bounding box에서의 작은 오류는 일반적으로 괜찮지만, 작은 Bounding box의 작은 오류는 IOU에 큰 영향을 미치기 때문이다