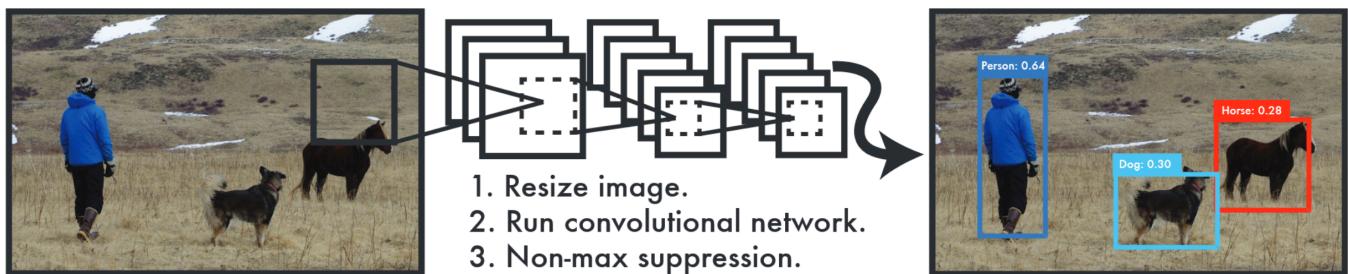


YOLO (2016)

서론

이전에 Object Detection 해결을 위한 접근이 분류였다면, 이를 회귀 문제로 재정의하여 해결하고자 한다. 단일 신경망이 전체 이미지에서 바운딩 박스와 클래스 확률을 직접 예측한다. 전체 detection pipeline이 단일 네트워크 이므로, **end-to-end learning**^[1]이 가능하다. 즉 **Unified Detection**을 하게 되는 것, Unified Detection은 object detection에 필요한 기능인 특징 추출, bounding box 생성, classification 생성을 한개의 model로 수행할 수 있다는 의미.



간단히 3단계를 거치게 된다.

1. 입력 이미지 448×448 로 resize
2. 단일 convolution network 통과
3. model의 confidence의 따른 결과 threshold

작동 방식

1. 이미지 분할

전체 이미지를 $S \times S$ 그리드로 나눈다. 만약 어떤 객체의 중심점이 특정 그리드 셀(grid cell) 안에 포함된다면, 그리드 셀이 해당 객체를 탐지할 책임을 진다. 즉, <특정 객체>의 중심에 위치한 그리드 셀을 제외한 다른 셀들은 신경쓸 필요 없이 오직 그 셀만 <특정 객체>를 예측하는 임무를 맡게 된다.

2. 그리드 셀의 예측

각 그리드 셀은 두 가지를 예측한다.

- B 개의 bounding box 와 confidence score

bounding box 와 confidence score 는 다음과 같이 5개의 값을 갖는다.

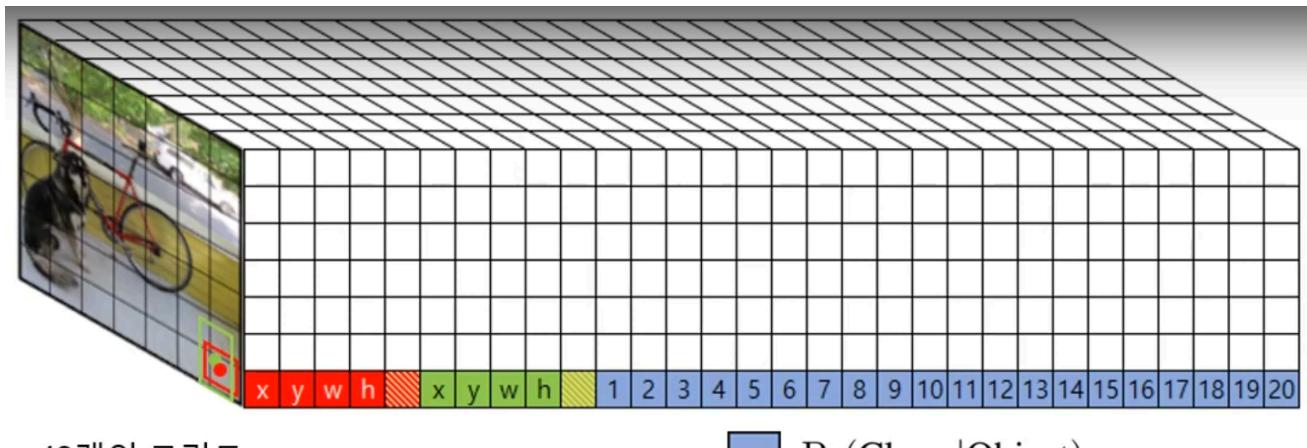
$(x, y, w, h, \text{confidence})$

- (x, y) : box의 중심 좌표 (그리드 셀 내에서의 상대적 위치)

ex. $(0.5, 0.5)$ 라면 해당 그리드 셀의 중심 좌표에 해당

- (w, h) : box의 크기 (전체 이미지에 대한 상대적 크기)
ex. (0.5, 0.5)라면 box의 크기가 전체 이미지 해상도의 절반, (224, 224)였다면 (112, 112) 크기)
- confidence: bounding box 예측이 얼마나 정확한지에 대한 점수
 $\text{Pr}(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$ 로 정의. 즉 '이 박스 안에 객체가 있을 확률'과 '예측된 박스가 실제 객체가 실제 객체 박스와 얼마나 겹치는지(IOU)'를 모두 반영한 값. 객체가 없다면 0에 가깝다.
- C 개의 conditional class probability
각 셀의 bounding box 의 수 B 와 관계없이, 그 그리드 셀당 C 개의 conditional class probability ($=\text{Pr}(\text{Class}_i|\text{Object})$, 객체가 있다는 가정하에 i 번째 class일 확률)을 만든다.
중요한 점은 B 개의 bounding box 를 예측하더라도, 클래스 확률은 한 세트만 예측한다.
ex. $C = 20$ 이고 (0.1, 0.9)라면 1번째 클래스는 10%, 2번째 클래스는 90%의 확률이라는 뜻

3. 최종 예측



각 그리드 셀들에 대한 예측 값들은 하나의 Tensor로 통합된다.

$S = 7, B = 2, C = 20$ 일 때 최종 출력 Tensor 크기는 $7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$ 가 된다.
이는 각 그리드($S \times S$)별로 B 개의 bounding box 와 confidence score 를 예측한 값($B \times 5$)과 conditional class probability C 를 합했다는 뜻이다.

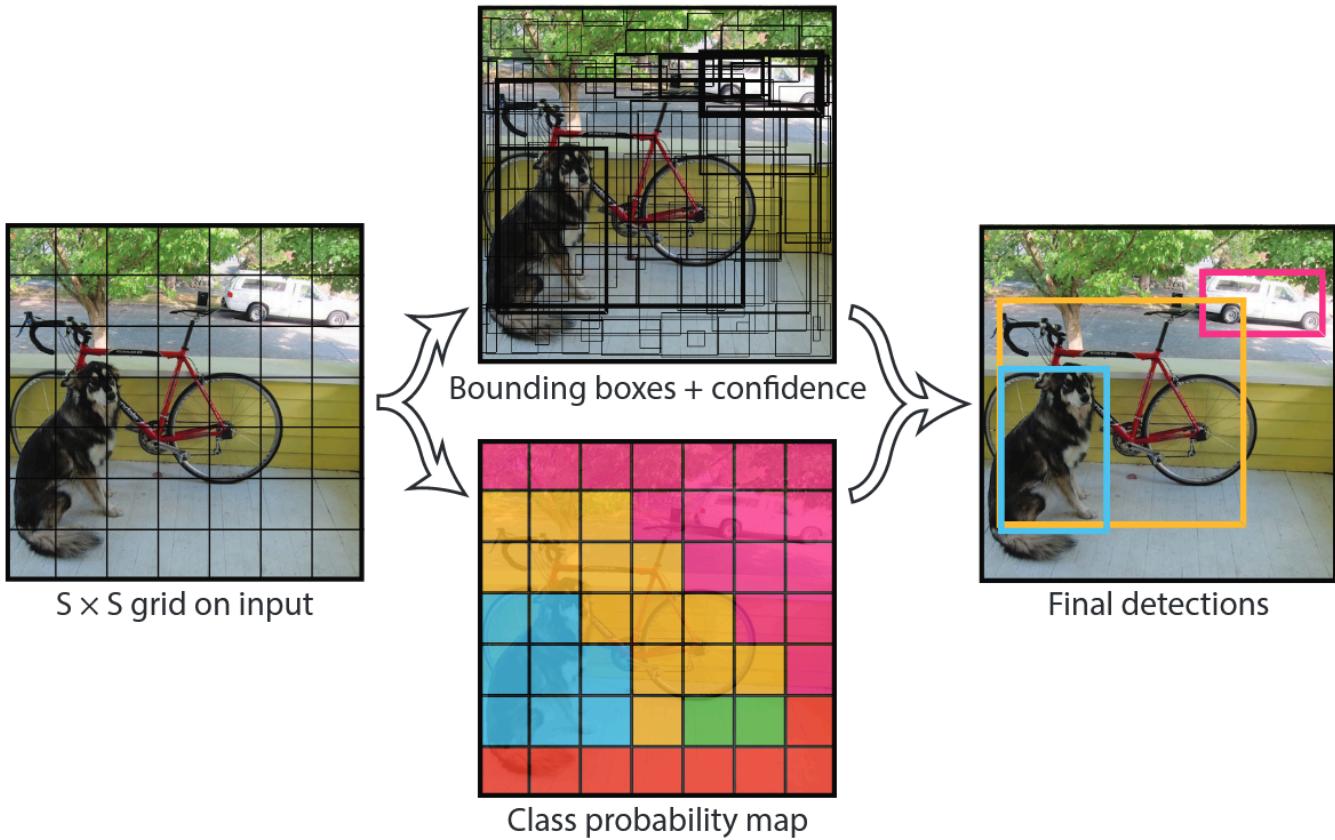
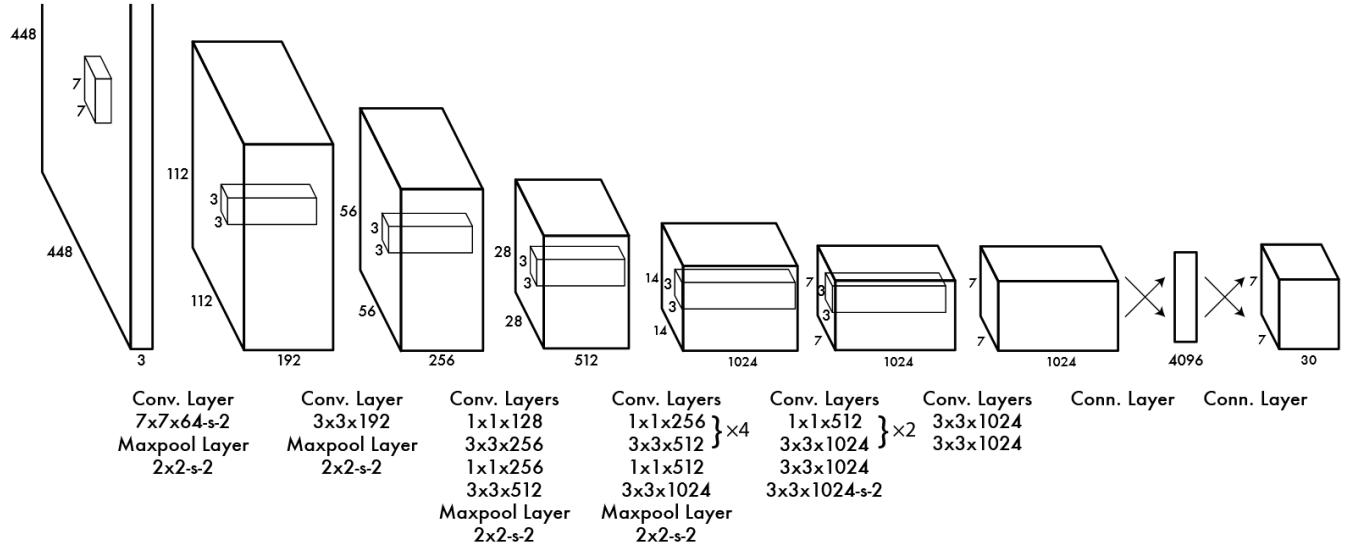


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

모델 구조



24개의 Conv layer와 2개의 FC layer로 구성된다. GoogLeNet에서 영감을 받아 inception module 대신 1×1 reduction layer 이후 3×3 convolutional layer 사용.

(※ **Fast YOLO**의 경우는 24개의 Conv layer 대신 9개 사용, 그 외는 YOLO와 동일)

학습

사전 학습이 끝난 모델을 **Object Detection**용으로 변환하여 본격적인 학습을 시작하게 된다.

사전 학습

20개의 Conv layer 뒤에 Average-Pooling과 FC를 붙여 classification model을 만들어 ImageNet 1000-class dataset을 사전 학습시킨다. (※ 여기서 입력 이미지 해상도는 224×224 이다.)

검증 데이터셋에서 88%의 top-5 정확도를 달성함.

Object Detection 학습

이미지의 보편적인 특징을 추출할 수 있는 pretrained network의 backbone을 활용하여 이를 Object Detection용으로 변환한다.

모델 구조 변경

- pretrained network 뒤에 4개의 Conv layer, 2개의 FC Layer 추가
- Object Detection에서는 정교한 시각 정보를 요구하기 때문에, 입력 이미지 해상도를 2배로 키운다. ($224 \times 224 \Rightarrow 448 \times 448$)

손실 함수 설계

예측값과 실제값의 차이를 계산(sum-squared error)하는 과정에서 몇 가지 문제를 해결하기 위한 장치가 있다.

- 문제 1: 모든 이미지에서 대부분의 그리드 셀은 객체가 포함되지 않는다(배경이 대부분이고 이는 객체를 포함하지 않음). 이로 인해 셀의 confidence score 를 0으로 만들려는 gradient가 객체가 있는 소수의 셀의 gradient보다 훨씬 커서 모델이 불안정해질 수 있다.
해결책: 두 개의 파라미터 λ_{coord} (coordinate)와 λ_{noobj} (no object) 도입. 바운딩 박스 좌표 예측 오차의 중요도를 높이고(논문에서는 $\lambda_{coord} = 5$), 객체가 없는 박스의 confidence 오차의 중요도는 낮춘다(논문에서는 $\lambda_{noobj} = 0.5$.)
- 문제 2: Sum-squared error는 같은 크기의 오차라도 큰 박스에서보다 작은 박스에서 훨씬 치명적이다. (ex. 10x10 박스에서의 2픽셀 오차와, 100x100 박스에서의 2픽셀 오차)
해결책: 박스의 너비와 높이(w, h)를 직접 예측하는 대신, 너비와 높이의 제곱근(\sqrt{w}, \sqrt{h})을 예측하도록 한다. 이렇게 하면 작은 박스에서 발생하는 작은 오차가 loss에 더 크게 반영.
- 문제 3: 하나의 그리드 셀이 여러개의 바운딩 박스를 예측할 때, B 개의 예측기 중에서 어떤 예측기에 책임을 물을지 정해야한다.
해결책: 각 객체에 대해, 해당 객체의 실제 박스(Ground Truth)와 IOU가 가장 높은 예측기 하나에게만 책임(responsible)을 부여한다. 손실 함수는 이 '책임'이 있는 예측기의 좌표 및 신뢰도 오차에 대해서만 패널티 부여. 이를 통해 각 예측기가 특정 크기, 비율(종횡비), 종류의 객체 예측에 전문화 되도록 한다.

변수	설명
λ_{coord}	BBox의 좌표 예측 오차의 중요도 조절 (논문에서는 5)
λ_{noobj}	객체가 없는 BBox의 confidence 오차 중요도 조절 (논문에서는 0.5)
1_i^{obj}	객체가 「셀 i 」에 있는지 여부 (0 or 1)
1_{ij}^{obj}	「셀 i 」의 「 j 번째 BBox 예측기」가 해당 예측에 대한 책임이 있는지 여부 (0 or 1) (1_{ij}^{noobj} 는 이에 반대)

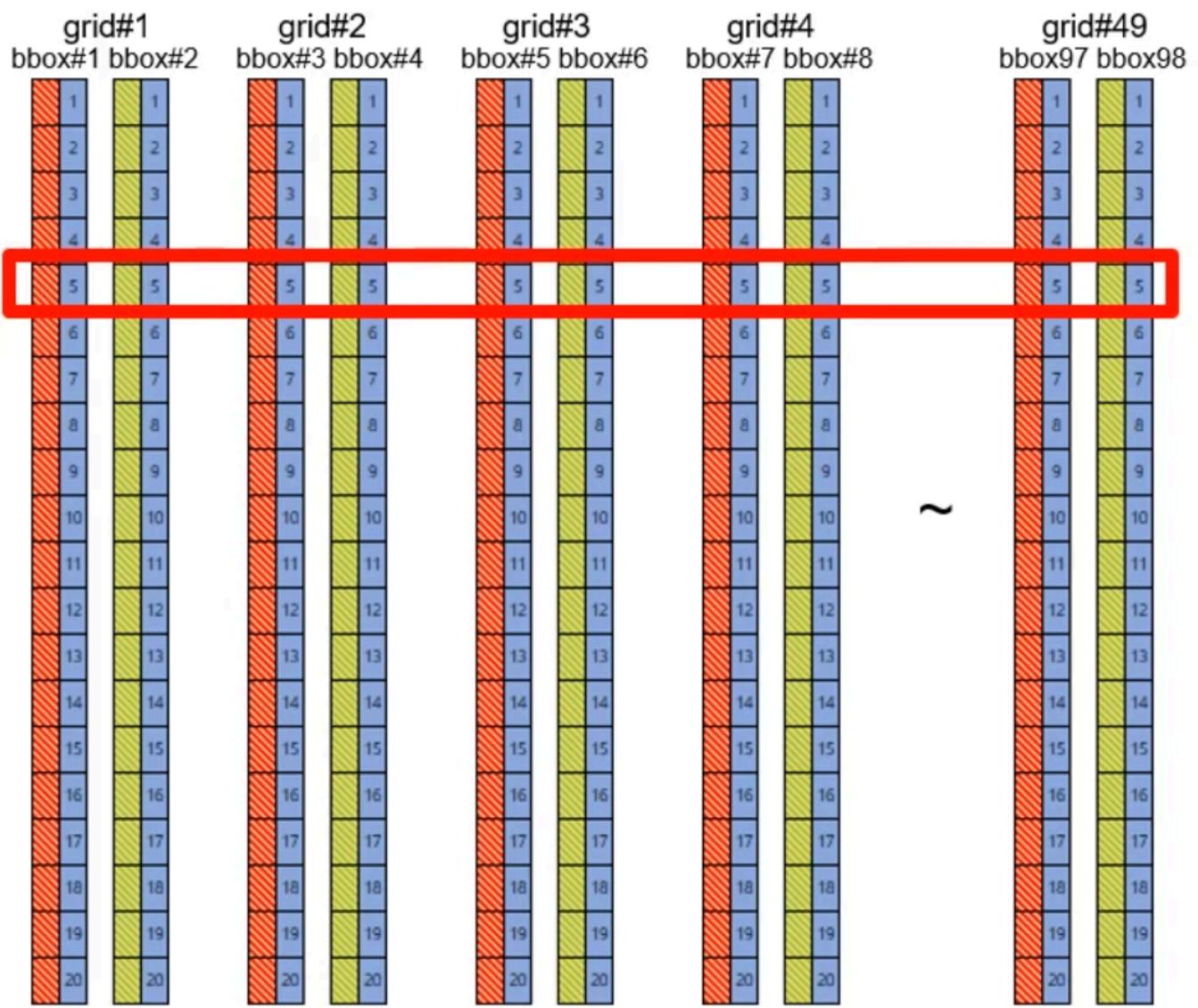
loss function

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

테스트

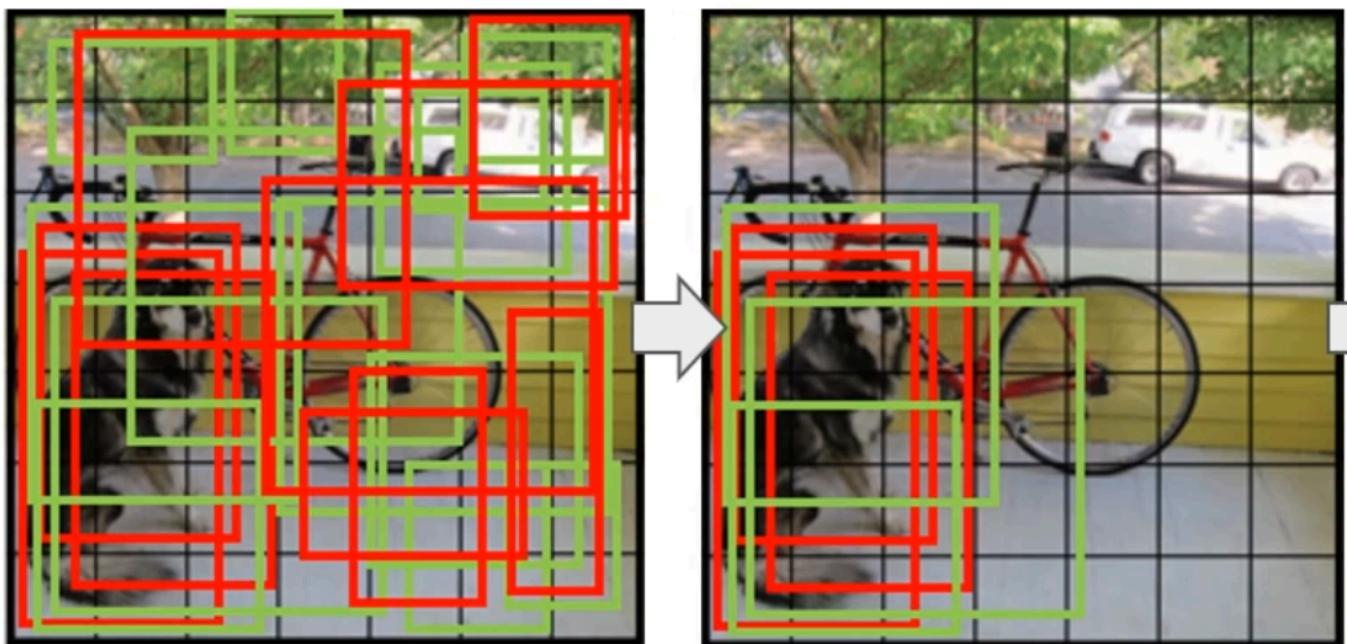
테스트 시 개별 box에 대한 conditional class probability 와 confidence score 를 곱하여 class-specific confidence score 를 얻을 수 있다.

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

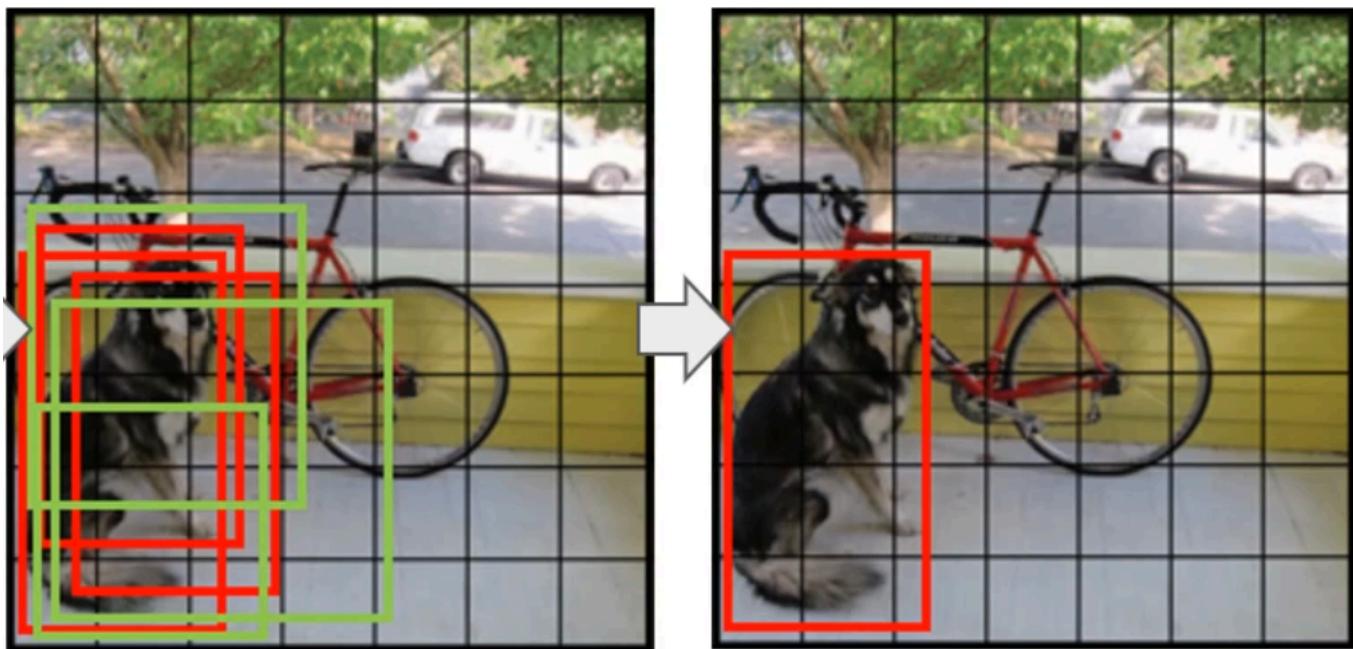


그리드가 $7 \times 7 = 49$, 각 그리드 별로 2개의 예측기가 만든 BBox가 있을 것이므로 총 98개의 BBox가 있다. 각 BBox에 대한 confidence score 와 conditional class probability 가 있으므로 이 둘을 곱한 위와 같은 결과, 즉 BBox에 대한 class-specific confidence score 들이 나올 것이다.

이때 class-specific confidence score 가 특정 수치 이하이면 제거



이후 남은 BBox에서 서로 IOU가 특정 수치이상이면 class-specific confidence score 가 제일 높은 것만 남기고 나머지는 제거한다. 이를 **NMS(non-maximal suppression)**라고 부른다.



한계점

- 각 그리드 셀은 두 개의 박스만 예측하고, 오직 하나의 클래스만 가질 수 있으므로 강한 공간적 제약 (spatial constraint)이 있다.
가까이 붙어 있는 여러 개의 작은 객체를 탐지하는 데 어려움이 있다. (ex. 새 떼)

2. 알고리즘이 아닌 데이터에서 바운딩 박스를 예측하도록 학습하므로 새롭거나 비정상적인 종횡비, 형태를 가진 객체에 대한 일반화 성능이 떨어진다.
 3. 부정확한 위치 예측(Localization)
입력 이미지에서 downsampling layers를 거치기 때문에, 비교적 거친 특징(coarse features)을 사용하여 바운딩 박스를 예측한다. 이로 인해 타 모델에 비해 위치 정확도가 떨어지는 경향이 있다.
 4. 작은 박스와 큰 박스의 오차를 동일하게 취급하는 문제
큰 박스에서의 작은 오차는 비교적 괜찮지만(benign), 작은 박스에서의 작은 오차 IOU 값에 훨씬 큰 영향을 미친다. 비록 이를 너비와 높이의 제곱근을 예측하도록 하였지만, 완벽한 해결책이 되지는 않는다.
-

성능

실시간 탐지 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

Fast YOLO: mAP(정확도) 52.7%, 속도 155FPS로 당시 다른 Real Time 탐지기보다 2배이상 정확한 성능이다.

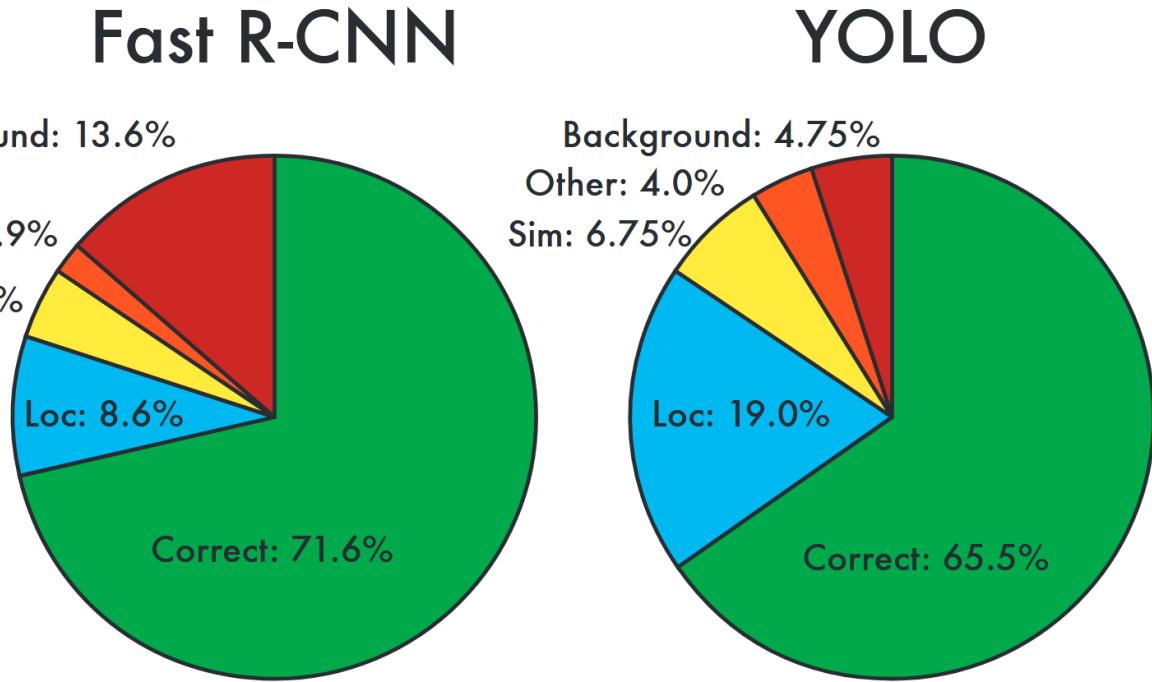
YOLO(표준모델): mAP 63.4%로 더 높고, 속도는 45 FPS 달성. 이와 같은 속도는 실시간 기준 (30FPS)를 뛰어 넘는 속도이다.

당시 가장 성능이 좋았던 모델 중 하나인 Faster R-CNN (VGG-16)은 mAP가 73.2%로 더 높지만, 속도는 7 FPS에 불과해 실시간 탐지는 불가능하였다.

⇒ 정확도(mAP)를 약간 희생하는 대신, 다른 모델들이 따라올 수 없는 압도적인 처리 속도(FPS)를 확보하였다.

오류 유형 분석

YOLO는 다른 모델과 실수하는 유형이 다르다. 아래 그래프는 YOLO와 Fast R-CNN의 오류를 비교한 그래프이다.



약점: Localization

YOLO의 가장 큰 약점은 **객체의 위치를 정확하게 잡는 능력(Localization)**이다. 전체 오류 중 19.0%가 위치 오류일 정도로, 다른 모든 오류를 합친 것보다 비중이 크다.

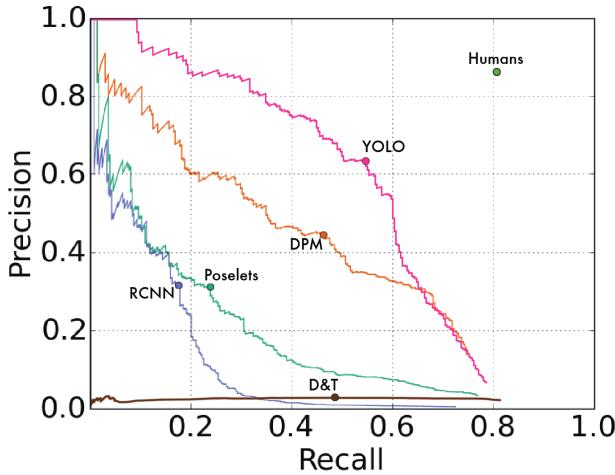
강점: Background

반면, 객체가 없는 배경을 객체로 잘못 탐지하는 오류는 4.75%로 매우 낮다. 이는 13.6%를 기록한 Fast R-CNN보다 3배 가까이 낮은 수치이다.

⇒ 물체의 위치를 정확히 특정하는 데에는 약하지만, YOLO는 이미지 전체의 맥락을 이해하여 배경과 객체를 헷갈리는 실수는 훨씬 적게한다.

일반화 성능

YOLO는 학습 데이터와 다른 새로운 유형의 데이터에서도 일반화 성능이 뛰어나다. 아래 그래프는 일반 사진으로 학습한 뒤 예술 작품 속 객체를 탐지한 결과이다.



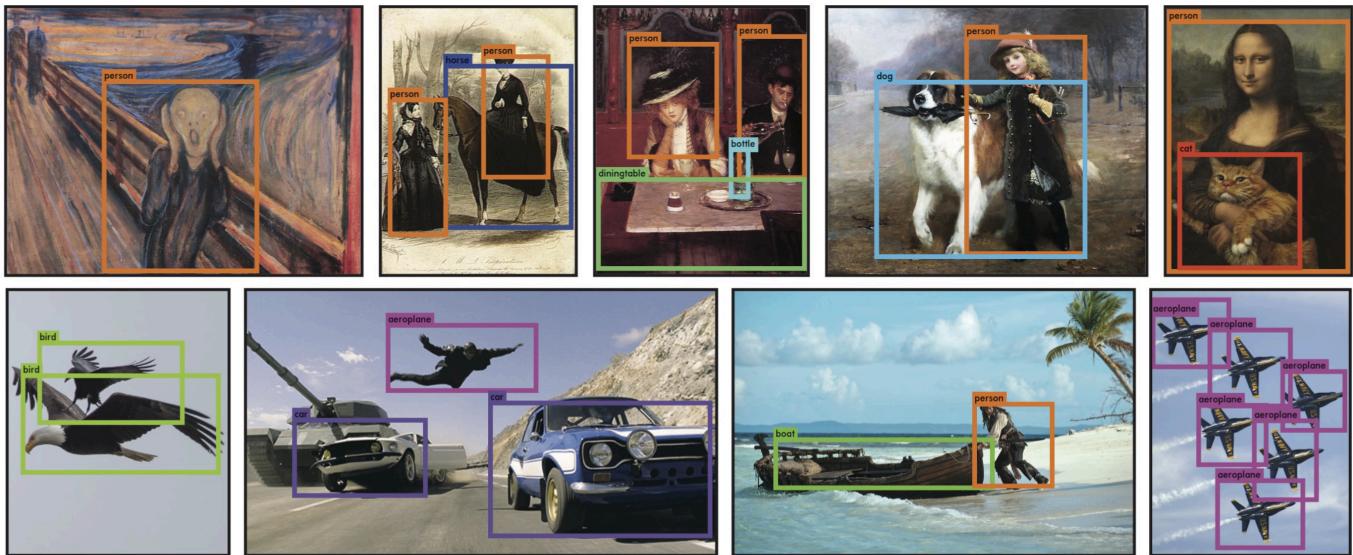
(a) Picasso Dataset precision-recall curves.

	VOC 2007 AP	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 일반 사진(VOC 2007)에서 54.2 AP를 보였지만, 예술 작품(Picasso Dataset)에서는 10.4 AP로 성능이 급격히 하락하였다.

반면, YOLO는 일반 사진에서 59.2 AP, 예술 작품에서 53.3 AP를 기록하며 다른 모델에 비해 성능 하락 폭이 훨씬 적었다.



⇒ YOLO는 객체의 일반적인 형태와 관계를 학습하기 때문에, 데이터 Domain(dataset)이 바뀌어도 다른 모델들보다 안정적인 성능을 유지한다.

-
1. 입력부터 최종 출력까지 중간 단계 없이 처리하는 접근 방식. ↵