1. YOLO v1 소개

- 2016년 컴퓨터 비전 분야 최고 학회인 CVPR(컴퓨터 비전 및 패턴인식)에 발표
- Object Detection
- 주요 특징

1. You Only Look Once

- 이미지를 쪼개지 않고 전체 이미지를 한번 통과 시킴
- 이미지 전체를 딱 한번만 봄

2. Unified

- 통합된 모델 제안
- Region Proposal과 Feature Extraction단계를 통합한 1 stage 방법

3. Real Time

- 1 stage 동작으로 속도 매우 빨라짐
- 이에 real time object detection(실시간 객체 탐지)이 가능해졌음
- https://www.youtube.com/watch?v=K9a6mGNmhbc

2. YOLO v1 제안방법

1. YOLO Detection System

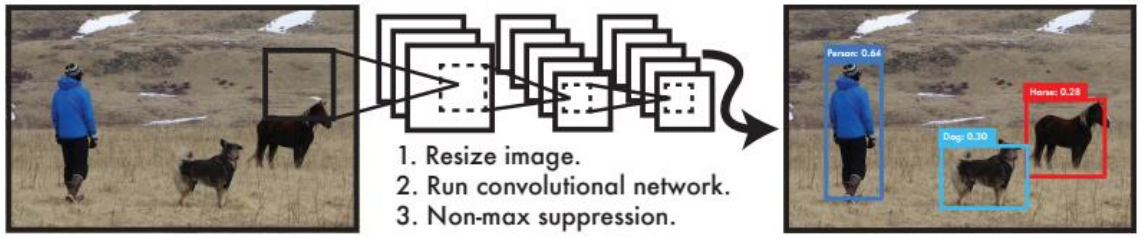


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

1. 448*448로 이미지 resize
2. 단 1개의 CNN 네트워크에 통과
3. Non-Maximum Suppression을 통해 최종 bounding box의 Location과 Class 결정

2. YOLO v1 제안방법

1. YOLO Detection System

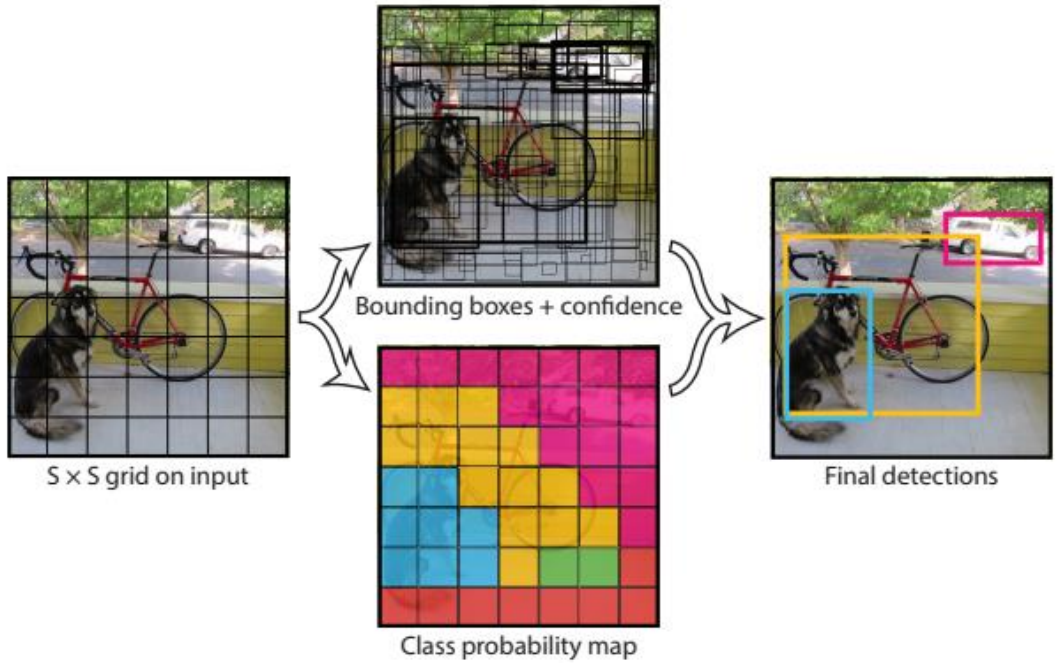


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

- ✓ Confidence: 해당 박스안에 object가 있을 확률
- ✓ C개의 조건부 확률: 해당 박스안에 object가 있을 때, $Class_t$ 일 확률
- ✓ IOU_{pred}^{truth}: 실제 Bounding box와 예측 Bounding box가 얼마나 겹치는지

1. 이미지를 S*S 그리드 셀로 나눈다
2. 각각의 그리드에서 물체가 있을 만한 영역을 B개의 Bounding Box로 예측한다
3. 박스의 Confidence를 계산한다.

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

4. 각각의 그리드마다 C개의 클래스의 조건부 확률을 구한다.

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

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

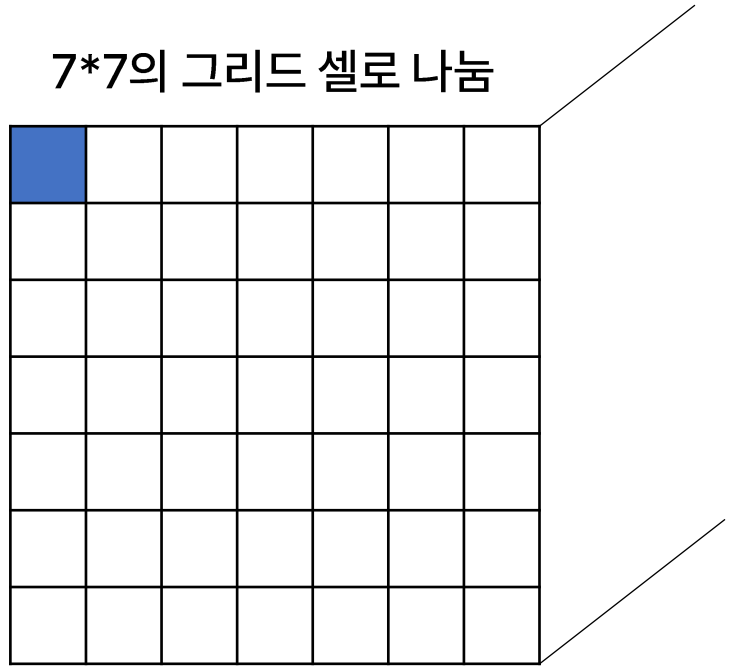
2. YOLO v1 제안방법

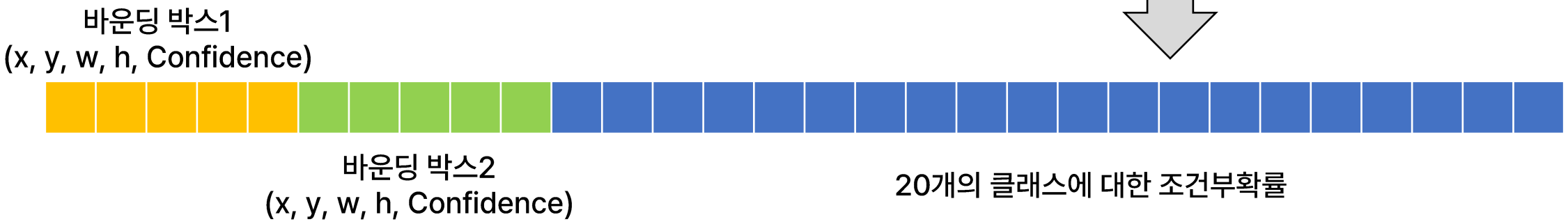
1. YOLO Detection System

PASCAL VOC detection dataset에 아래와 같이 사용

- 그리드 수(S) = 7
- 바운딩 박스의 개수(B) = 2
- Label(C) = 20

➔ **최종 예측 결과는 7*7*30 텐서**





2. YOLO v1 제안방법

2. Network Design

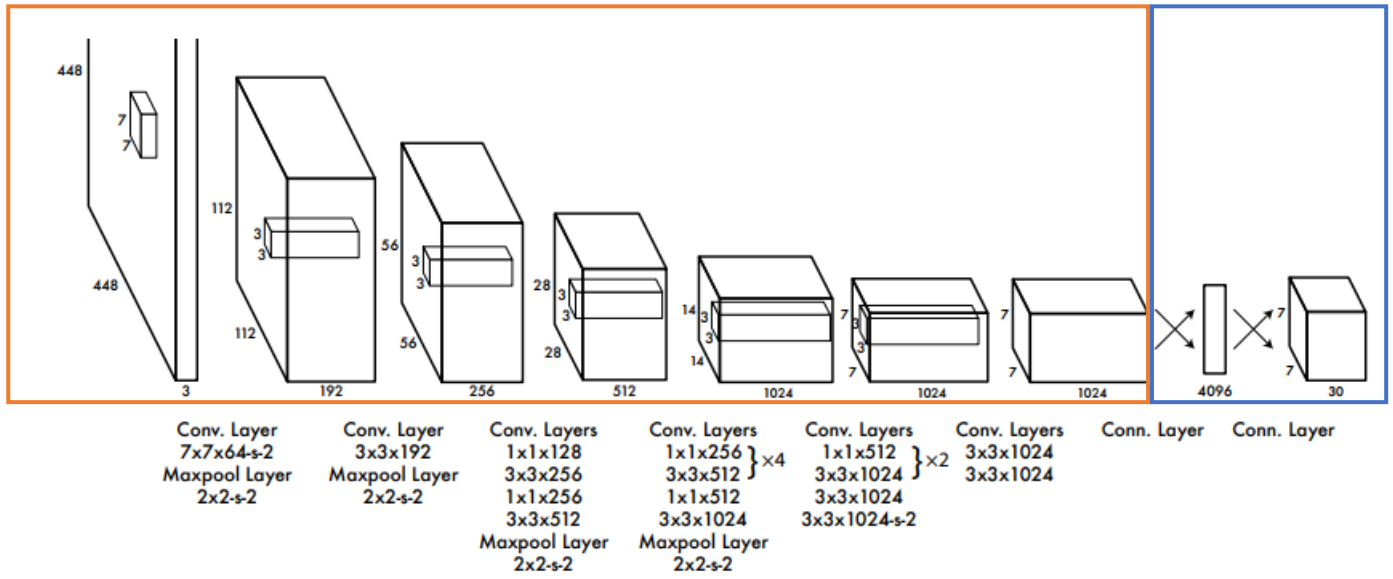


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

24개의 Convolution Layer과 2개의 Fully-Connected Layer

2. YOLO v1 제안방법

2. Network Design

네트워크 특징

❖ Pre-trained Network

: 앞부분의 20개의 Conv Layer는 GoogLeNet을 이용하여 ImageNet 1000-class competition dataset을 사전에 학습한 결과를 Fine-tuning한 네트워크
: 4개의 Conv Layer와 2개의 Fully Connected Layer를 추가하여 네트워크 구성

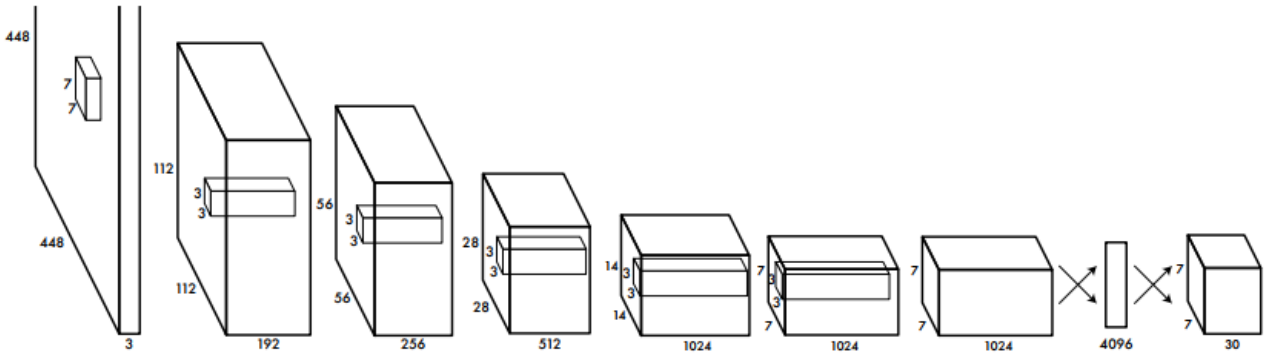
❖ Reduction Layer

: GoogLeNet이 사용한 inception 대신 간단하게 1*1 Reduction layer와 3*3 Conv layer 사용

❖ Activation Function

: 맨 마지막 layer ➔ Linear Activation Function
: 나머지 layer ➔ Leaky ReLU

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



2. YOLO v1 제안방법

2. Network Design

Non-Maximum Suppression(NMS, 비최대 억제)

: Object detector가 예측한 Bounding Box 중 정확한 Bounding Box를 선택하도록 하는 기법

1. 모든 Bounding Box에 대하여 threshold 이하의 Confidence score를 가지는 Box는 제거
2. 남은 Bounding Box들을 Confidence score 기준으로 내림차순 정렬
3. 맨 앞에 있는 Bounding Box 하나를 기준으로, 다른 Bounding Box와 IoU값 계산

   ➔ IoU가 threshold 이상인 Bounding Box는 제거

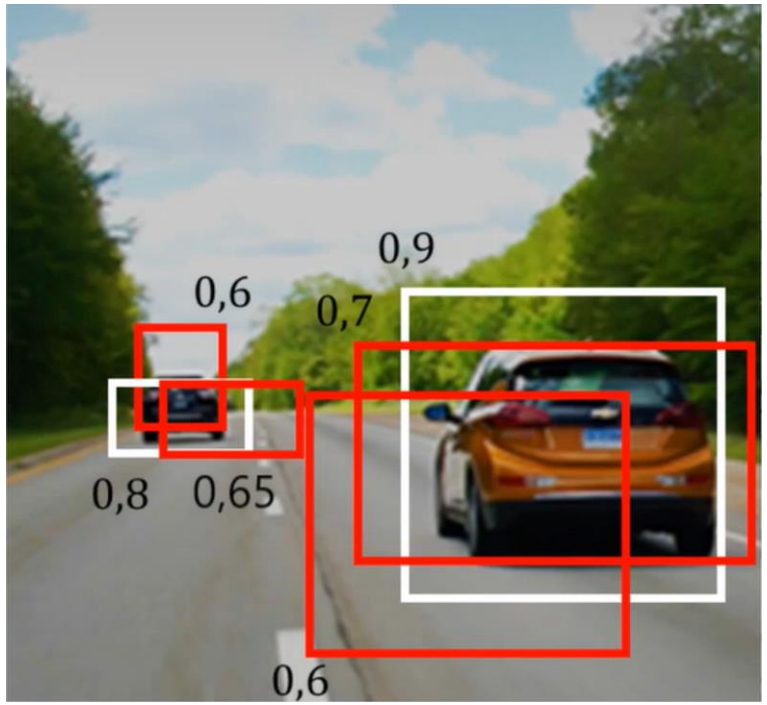
   ➔ 많이 겹칠수록 같은 부분의 같은 물체를 검출하고 있다고 판단하기 때문!

4. 위 과정을 순차적, 반복적으로 시행하여 모든 Bounding Box를 비교하고 제거

7*7*2개의 Bounding Box 중 Non-Maximum Suppression을 통해
최종 Bounding Box의 Location과 Class를 결정

2. YOLO v1 제안방법

2. Network Design



Class(Label) = Car의 예시

1. Confidence threshold = 0.4 , IoU threshold = 0.4라고 가정
2. Bounding Box를 Confidence 기준으로 내림차순 정렬

➔ [0.9, 0.8, 0.7, 0.65, 0.6(왼), 0.6(오)]

3. 가장 높은 0.9를 기준으로 모든 박스와 비교

➔ 0.8, 0.65, 0.6(왼)은 겹치지 않으므로 삭제X

➔ 0.7, 0.6(오)는 IoU threshold 이상이므로 제거

➔ [0.9, 0.8, 0, 0.65, 0.6, 0]

4. 0.8를 기준으로 모든 박스와 비교

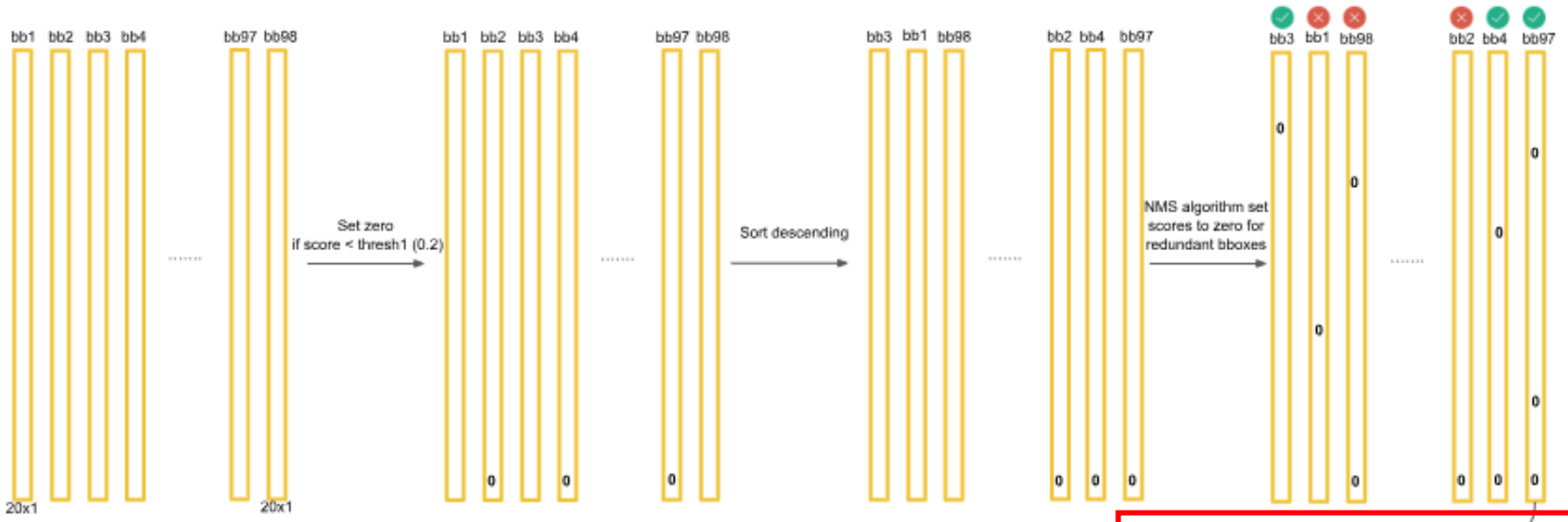
➔ [0.9, 0.8, 0, 0, 0, 0]

➔ 결과적으로 그림의 하얀 박스가 NMS 후 남게 됨!

출처: https://wikidocs.net/142645

2. YOLO v1 제안방법

2. Network Design



Class =max_index(scores for bb)

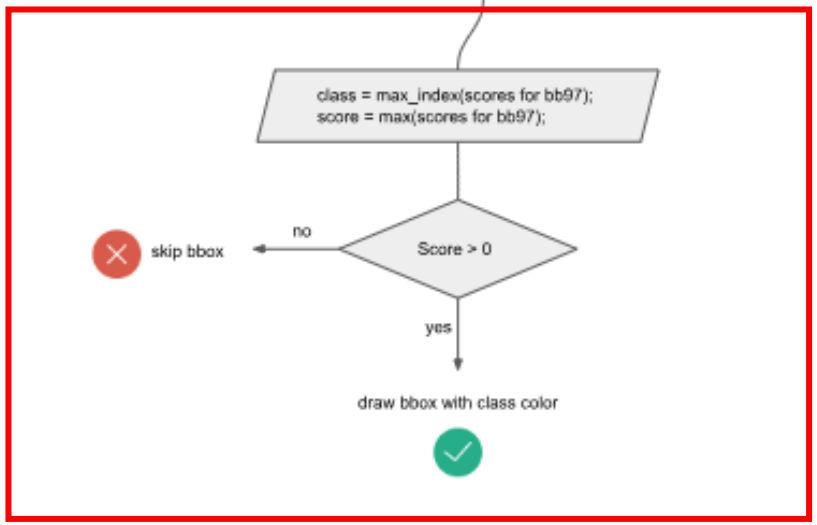
Score = max(scores for bb)

If Score > 0:

해당 박스의 Class와 Score 확정

else:

해당 박스는 skip



출처:https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.g137784ab86_4_2829

2. YOLO v1 제안방법

3. Loss Function

- ✓ λ_{coord}: 객체를 포함하는 cell에 가중치를 주는 파라미터 (본 논문에서는 5로 설정)
- ✓ 1_{ij}^{obj} : i 번째 cell의 j 번째 Bounding box가 객체를 예측하도록 할당 받았을 때 1 , 그렇지 않을 경우 0 ➔ B개의 Bounding box 중 1개만 학습에 사용!
- ✓ λ_{noobj} : 객체를 포함하지 않는 cell에 가중치를 주는 파라미터 (본 논문에서는 0.5로 설정)

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

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

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

Classification Loss

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

2. YOLO v1 제안방법

3. Loss Function

SSE(Sum or Squared Error, 오차제곱합)

- ✓ λ_{coord}: 객체를 포함하는 cell에 가중치를 주는 파라미터 (본 논문에서는 5로 설정)
- ✓ [illegible] box에 [illegible] 받았을 때 1 , 그렇지 않을 경우 0 ➔ B개의 Bounding box 중 1개만 학습에 사용!
- ✓ λ_{noobj} : 객체를 포함하지 않는 cell에 가중치를 주는 파라미터 (본 논문에서는 0.5로 설정)

loss function:

Localization

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

Classification Loss

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

$$SSE(Sum\ of\ Squared\ Error) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

예측 결과와 실제 값의 차이를 제곱하여 총 합을 구한 값

3. 실험 결과

1. 다른 real time 방법들과 성능 비교

- ✓ mAP(mean Average Precision): 얼마나 잘 탐지했는지
- ✓ FPS(Frames Per Second): 얼마나 빨리 탐지 했는지

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

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

3. 실험 결과

2. VOC2007 에러 분석

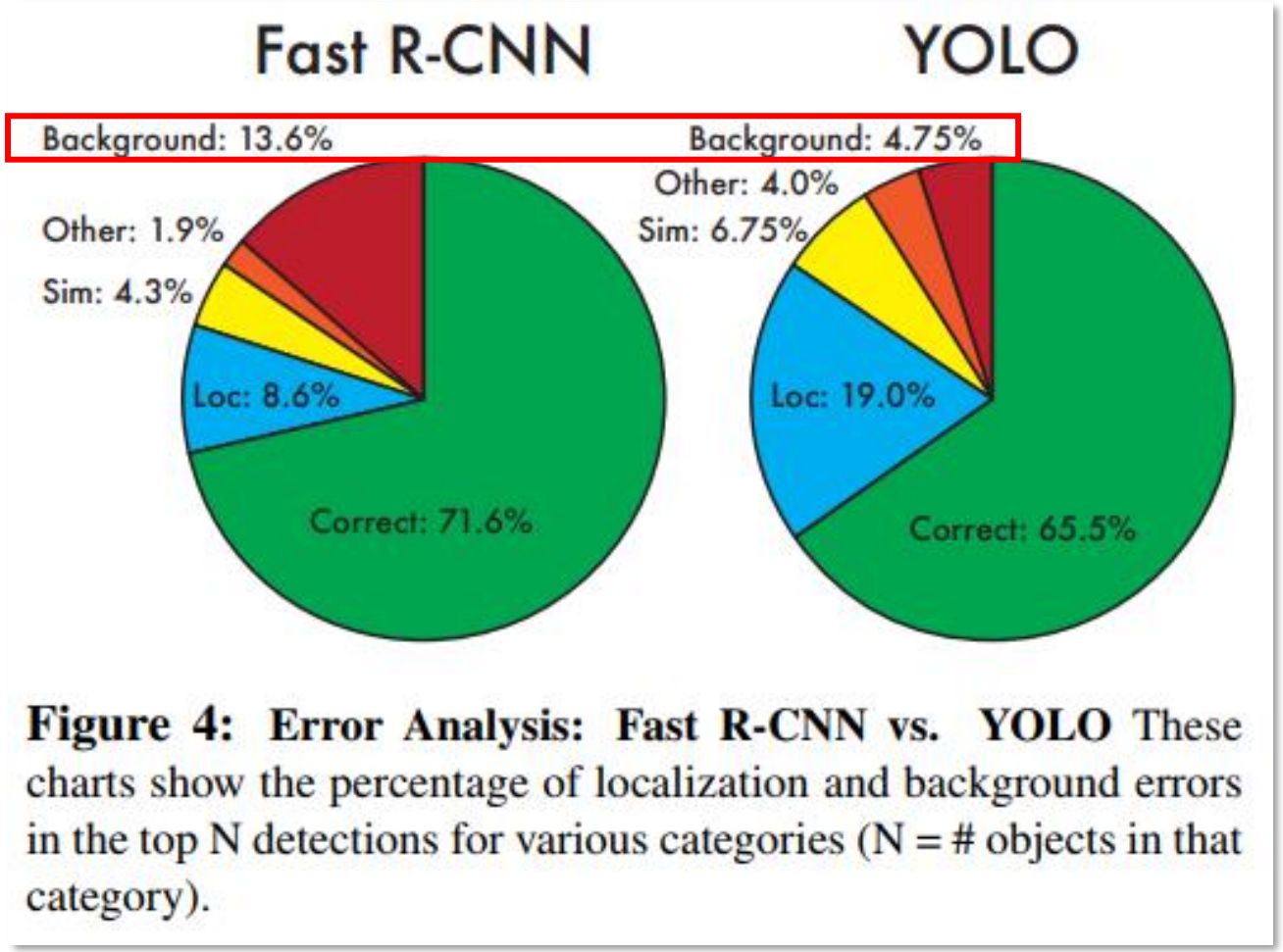


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

3. 실험 결과

3. Fast R-CNN과 YOLO v1의 결합

	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**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

3. 실험 결과

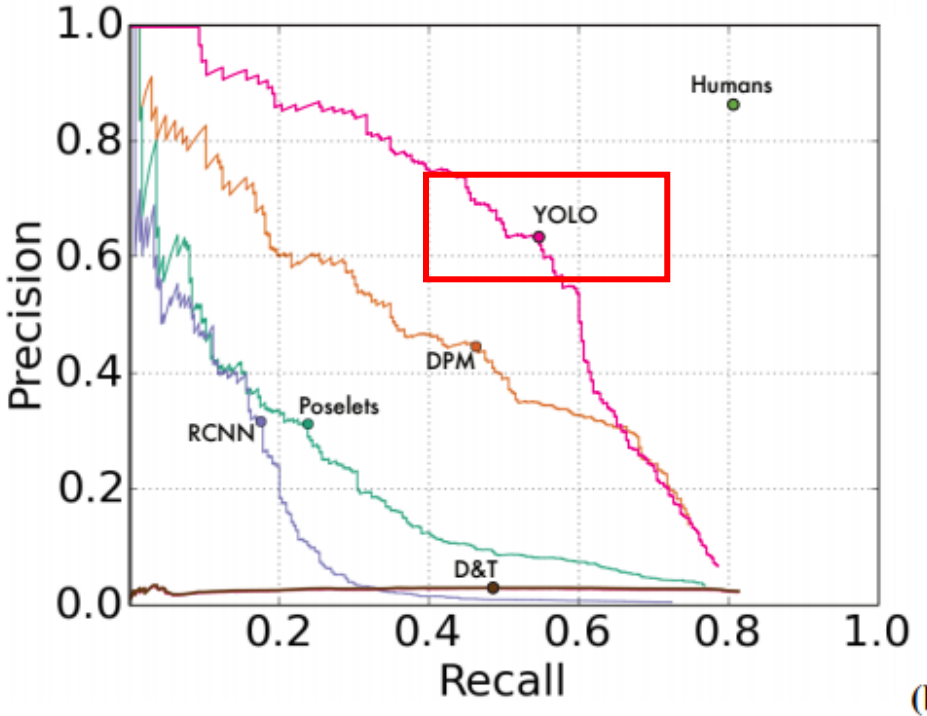
4. VOC2012 성능

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 [28]	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 [29]	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 [33]	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

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

3. 실험 결과

5. 일반화 성능



(a) Picasso Dataset precision-recall curves.

	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.

Figure 5: Generalization results on Picasso and People-Art datasets.

3. 실험 결과

5. 일반화 성능

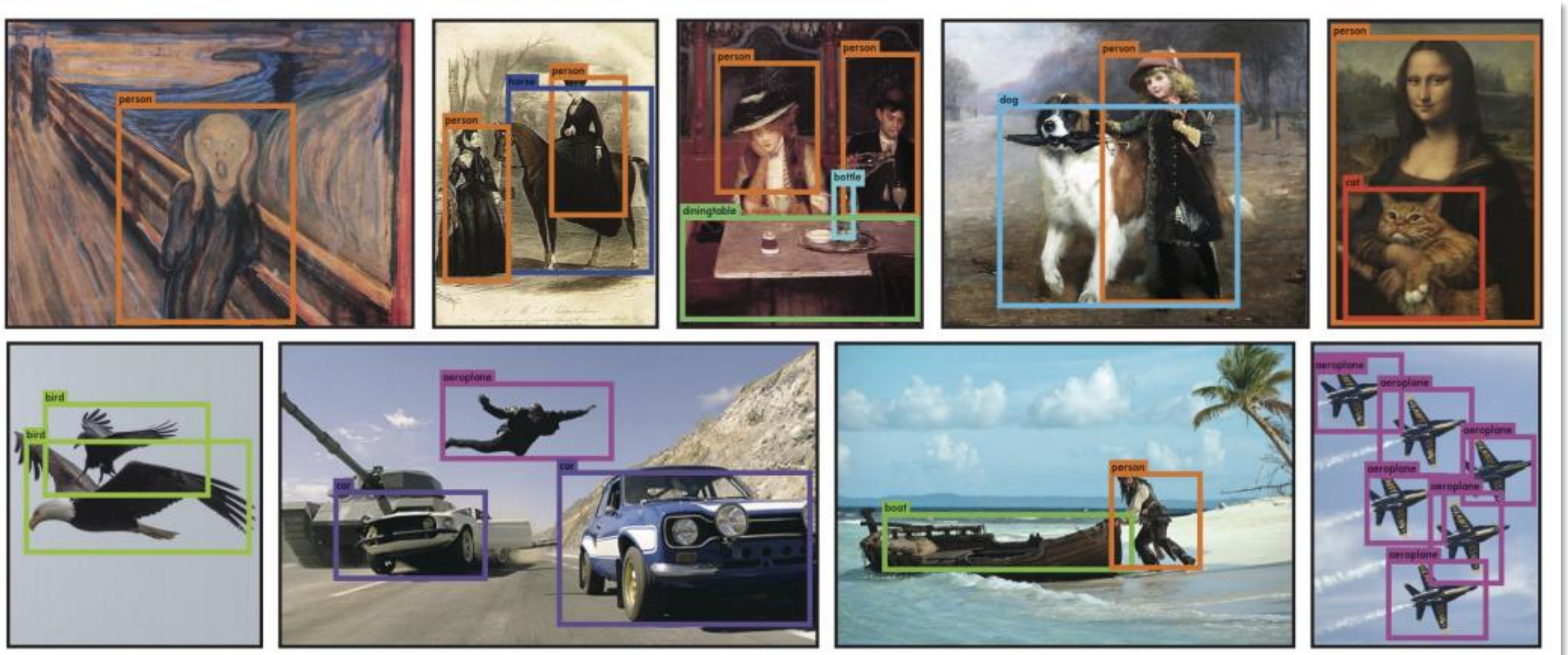


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

4. 결론

장점

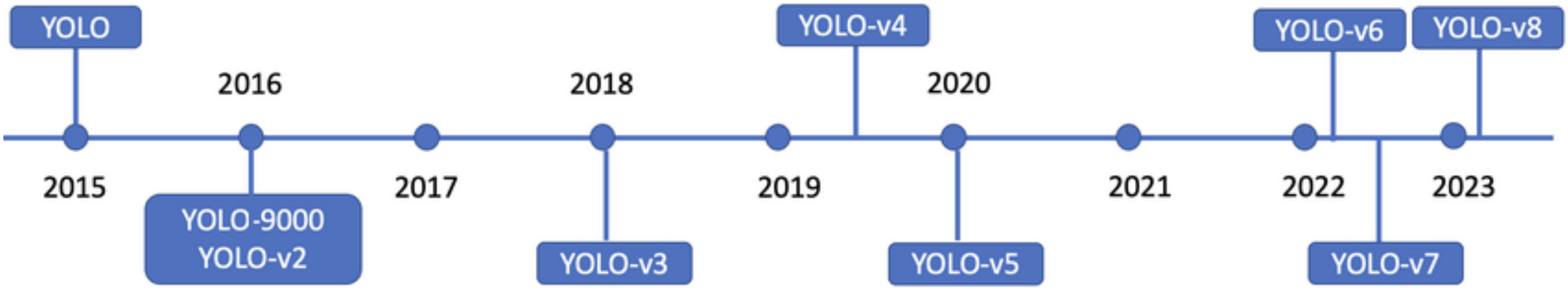
- 매우 빠르다
  ➔ 1 stage로 동작하기 때문
- 전체 문맥을 보고 판단한다
  ➔ Region Proposal 방식과 달리 이미지 전체를 기준으로 판단하므로 문맥 고려 가능
- 일반화 능력이 뛰어나다
  ➔ 사진 데이터 학습을 Artwork 데이터에 테스트 했을 때에도 좋은 성능을 확인할 수 있었음

한계

- 비교적 성능이 떨어짐
  ➔ 속도는 매우 빨라졌지만, 성능은 기존 2 stage 방법들에 비해 떨어짐
- 모여있거나 겹쳐있는 물체를 검출하는 성능이 떨어짐
  ➔ 하나의 grid 당 2개의 박스만 검출하도록 설계되었기 때문

4. 결론

YOLO Timeline



출처: https://www.researchgate.net/figure/Timeline-of-You-Only-Look-Once-YOLO-variants_fig1_369379818