YOU ONLY LOOK ONCE: UNIFIED REAL-TIME OBJECT DETECTION

20200961 최소연

Contents

1. Introduction

◦ Deformable parts models(DPM)

- Sliding window 방식으로 object detection을 하는 모델

◦ R-CNN

- 이미지 안에서 bounding box를 생성하기 위해 region proposal이라는 방법을 사용
- 속도가 느리고, 각 절차를 독립적으로 훈련시켜야 하기 때문에 최적화가 힘듬

1. Introduction

◦ Yolo

- object detection을 하나의 회귀 문제(single regression problem)로 정의
- 장점 :매우 빠름

  이미지 전체를 사용 -> background error가 적음

  일반화된 특징을 학습 -> 정확도가 더 높다
- 단점 : 최근 SOTA objection detection 모델에 비해 정확도가 조금 떨어짐

2. Unified Detection

◦ Yolo

1. 입력 이미지를 S x S grid로 나눔
2. 어떤 객체의 중심이 특정 grid cell 안에 위치한다면 그리드 셀이 해당 객체를 검출

◦ 각 grid cell은 B개의 Bounding Box와 그 Bounding box에 대한 confidence score 예측

◦ confidence score

- bounding box가 객체를 포함한다는 것을 얼마나 믿을만한지
- 예측한 bounding box 가 얼마나 정확한지
- $\Pr(Object) * IOU_{pred}^{truth}$
- Grid cell에 물체가 X : Pr(Obejct)=0
- Grid cell에 물체가 O : Pr(Obejct)=1

- IOU = (실제 bounding box와 예측 bounding box의 교집합) / (실제 bounding box와 예측 bounding box의 합집합

2. Unified Detection

◦ Bounding Box

- (x, y) 좌표 : bouning box 중심의 grid cell 내의 상대 위치. 0~1 사이의 값
- (w, h) : bounding box의 상대 너비와 상대 높이. 0~1 사이의 값
- Confidence : confience score

◦ 각 grid cell은 **conditional class probabilities(C)**를 예측

◦ conditional class probabilities(C)

- 그리드 셀 안에 객체가 있다는 조건 하에 그 객체가 어떤 class 인지에 대한 조건부 확률
- 하나의 그리드 셀에는 오직 하나의 클래스(class)에 대한 확률 값만 구함

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

2. Unified Detection

◦ class-specific confidence score

- These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

$$
\begin{aligned}
&class\ specific\ confidence\ score \\
&= \Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} \\
&= \Pr(Class_i) * IOU_{pred}^{truth}
\end{aligned}
$$

2. Unified Detection

S=7

B=2

C=20

S x S x (B*5 + C) = (7 * 7 * 30)

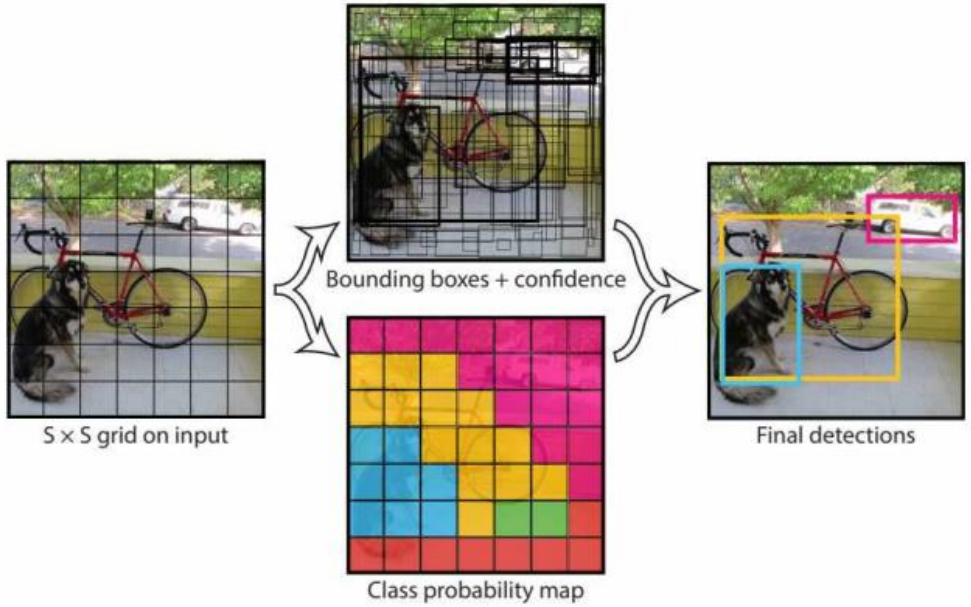


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

2. Unified Detection

2. 1. Network Design

Yolo 모델

- 하나의 CNN 구조
- GoogLeNet for image classification 모델을 기반
- 24개의 convolutional layer + 2개의 fully-connected layer
- convolutional layer : feature 추출, fully-connected layer : class 확률, Bounding Box 좌표 예측

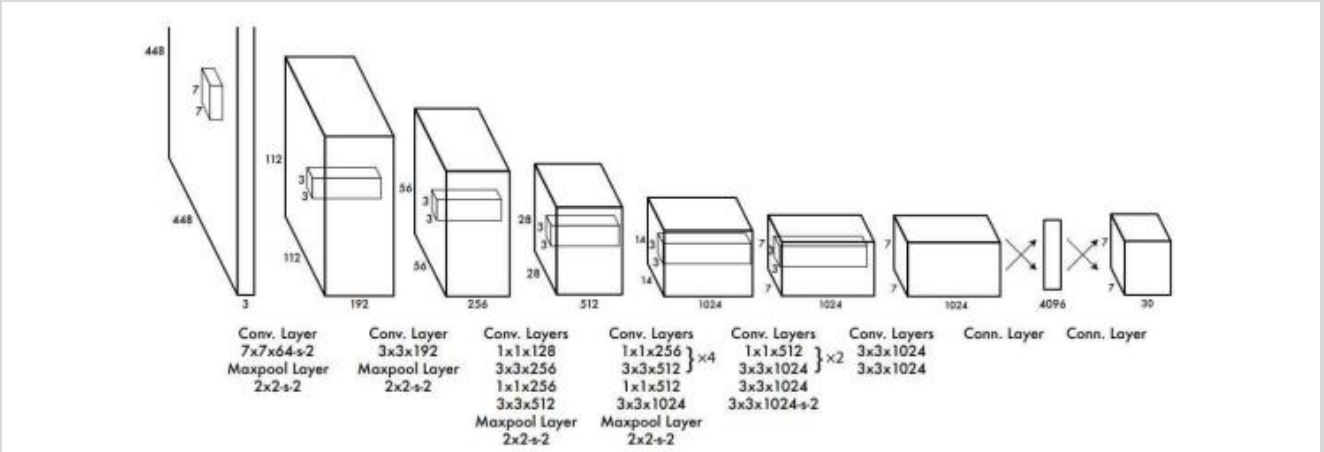


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

2. Unified Detection

2. 2. Training

- 1,000개의 클래스를 갖는 ImageNet data set으로 YOLO의 convolutional layer 을 pretrain
- Pretrain에서는 20개의 convolutional layer만 사용
- 훈련(training)과 추론(inference)을 위해 Darknet 프레임 워크를 사용

- Pretrain 된 분류(classification) model -> object detection model로 변경
- Pretrain 된 20개의 convolutional layer 뒤에 4개의 convolutional layer 및 2개의 fully-connected layer 을 추가하여 성능을 향상
- 입력 이미지의 해상도를 224 x 224에서 448 x 448로 증가

- 신경망의 최종 output = class probabilities, bounding box 위치정보(coordinates)

2. Unified Detection

- YOLO 신경망의 마지막 Layer 에는 선형 활성화 함수(linear activation function)를 적용
- 나머지 모든 계층에는 leaky ReLU 적용

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

2. Unified Detection

◦ Loss

- SSE(sum-squared error) 기반
- 최종 아웃풋의 SSE(sum-squared error)를 최적화(optimize)

- localization loss와 : bounding box의 위치 를 얼마나 잘 예측했는지
- classification loss : 클래스를 얼마나 잘 예측했는지

◦ 이미지 내에 대부분의 grid cell에는 object가 없음 -> confidence score=0 -> 대부분의 grid cell의 confidence score=0이 되도록 학습 -> model의 불균형

◦ 객체가 존재하는 bounding box 의 confidence loss에 대한 가중치 = 증가

◦ 객체가 존재하지 않는 bounding box의 confidence loss에 대한 가중치 = 감소

2. Unified Detection

- SSE는 큰 bounding box와 작은 boudning box에 대해 모두 동일한 가중치로 loss를 계산
- 작은 bounding box : 위치 변화에 민감
- 작은 bounding box : 위치 변화에 민감 X
- bounding box의 너비(widht)와 높이(hegith)에 square root를 취해서 SSE 문제 해결

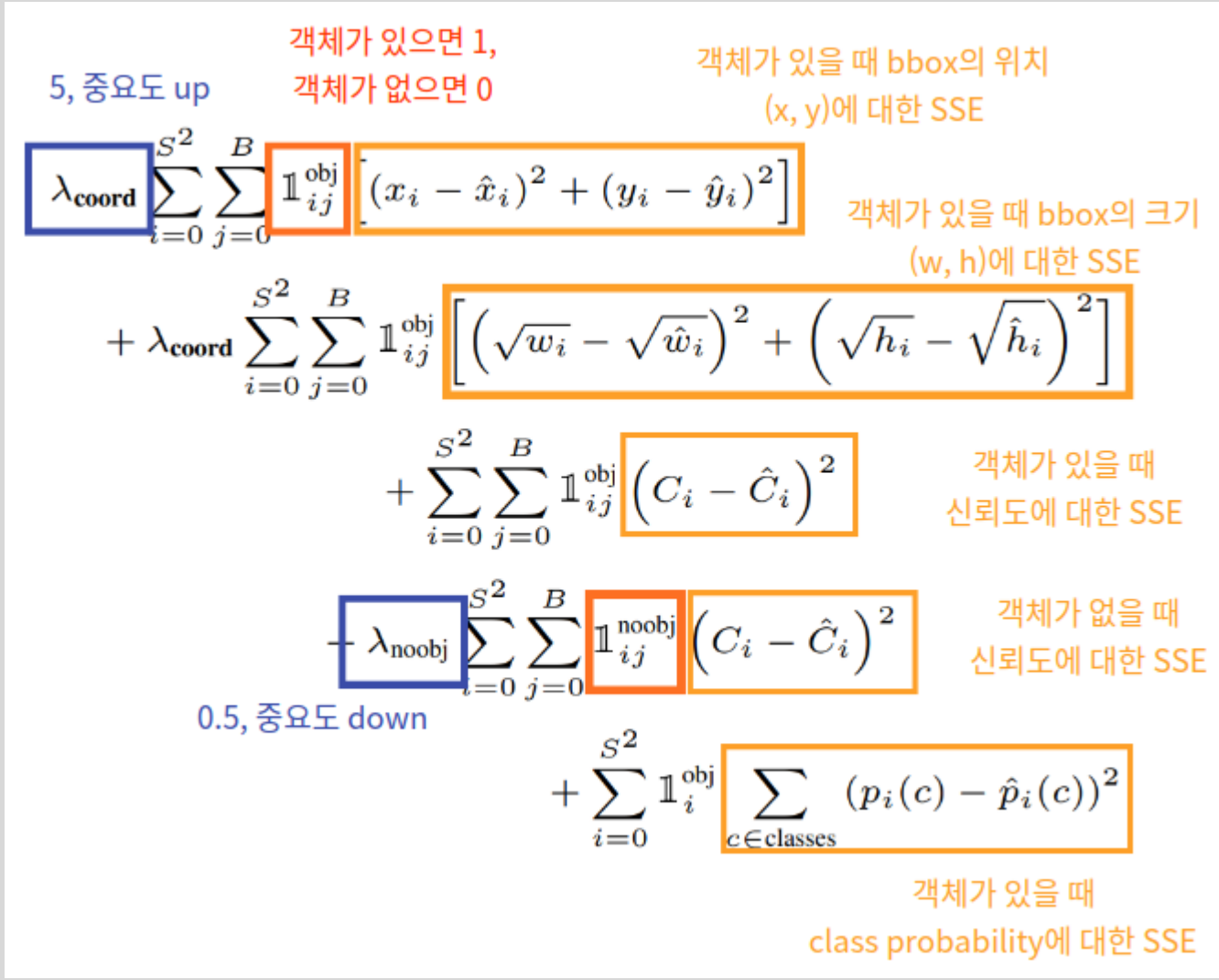
- 하나의 grid cell 당 여러 개의 bounding box를 예측
- 훈련 단계 에서 하나의 bounding box predictor가 하나의 객체에 대한 책임
- 예측된 여러 bounding box 중 실제 object를 감싸는 ground-truth boudning box와의 IOU가 가장 큰 것을 선택

2. Unified Detection

- loss function

- Λ_coord = 5
- Λ_noobj = 0.5

- coord : coordinate prediction 좌표예측
- noobj : no object



2. Unified Detection

- Batch size = 64
- Momentum = 0.9
- Decay = 0.005
- Learning rate : 0.001 -> 0.01 -> 0.001 -> 0.0001 로 변화시키면서 학습
- Dropout과 Data Augmentation으로 overfitting을 방지

2. Unified Detection

- 2. 3. Limitations of YOLO

- 하나의 grid cell마다 오직 하나의 객체만 검출 -> spatial constraints(공간적 제약) 야기
- spatial constraints : 하나의 그리드 셀에 두 개 이상의 객체가 붙어있다면 이를 잘 검출하지 못하는 문제

- incorrect localizations : 큰 bounding box와 작은 bounding box의 loss에 대해 동일한 가중치를 둠 -> 크기가 작은 bounding box는 위치가 조금만 달라져도 성능에 큰 영향

3. Comparison to Other Detection Systems

◦ Deformable parts models(DPM)

- 슬라이딩 윈도(sliding window) 방식
- 하나로 연결된 파이프라인이 아니라 서로 분리된 파이프라 인으로 구성
- 독립적인 파이프라인이 각각 feature extraction, region classification, bounding box 예측 (bounding box prediction) 등을 수행

  -> Yolo는 분리된 파이프라인을 하나의 Convolutional Neural Network으로 대체

◦ R-CNN

- region proposal 방식을 사용
- selective search라는 방식으로 여러 bounding box를 생성
- 컨볼루션 신경망으로 feature를 추출하고, SVM으로 bounding box에 대한 점수를 측정
- 선형 모델(linear model)로 bounding box를 조정하고, non-max suppression로 중복된 검출을 제거

4. Experiments

4. 1 Comparison to Other Real-Time Systems

- Fast R-CNN, Faster R-CNN VGG-16 :

  mAP는 가장 높으나 FPS가 낮음

- Fast Yolo : 가장 빠른 객체 검출 모델
- Yolo : mAP도 비교적 높고, FPS도 높음

- FPS : Frame Per Second. 초당 frame 수

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.

4. Experiments

4. 2. VOC 2007 Error Analysis

- Correct : class가 정확하며 IOU > 0.5 인 경우
- Localization : class가 정확하고, 0.1 < IOU < 0.5 인 경우
- Similar : class가 유사하고 IOU > 0.1 인 경우
- Other : class는 틀렸으나, IOU > 0.1 인 경우
- Background : 어떤 Object라도 IOU < 0.1 인 경우

- Yolo : localization error 가 상대적으로 큼
- Fast R-CNN : background error가 상대적으로 큼

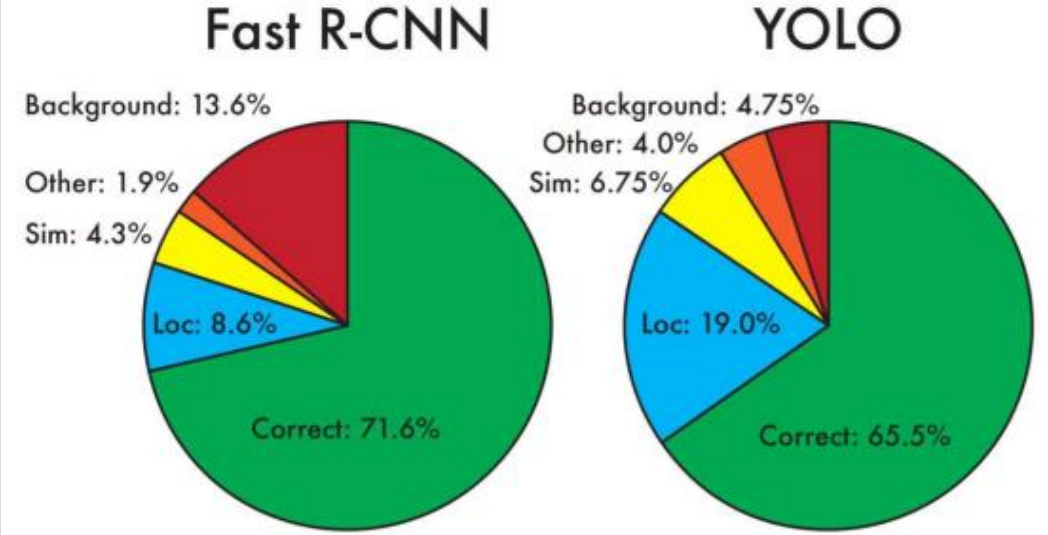


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

4. Experiments

4. 3 Combining Fast R-CNN and YOLO

- Fast R-CNN + YOLO를 -> background error 감소
- R-CNN이 예측한 Bounding Box와 Yolo가 예측한 Bounding Box가 겹치는 부분을 bounding Box로 잡음
- Fast R-CNN : 71.8% mAP
- Fast R-CNN + YOLO : 75% mAP

	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.

4. Experiments

4. 4. VOC 2012 Results

Yolo : 57.9% mAP

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.

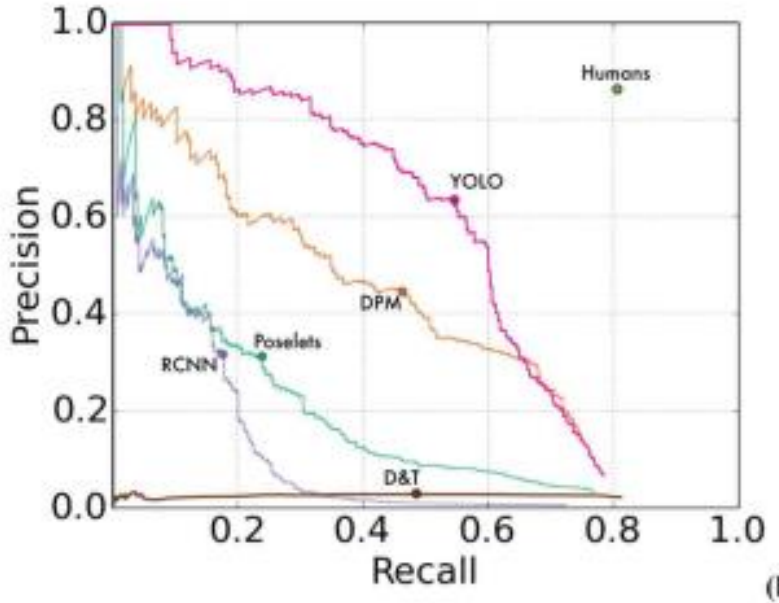
4. Experiments

4. 5. Generalizability: Person Detection in Artwork

R-CNN : VOC 2007에서는 높은 정확도, 예술 작품에 대해서는 낮은 정확도

DPM : 예술 작품에서 정확도가 크게 떨어지지 않지만, VOC에서도 정확도가 높지 않음

Yolo : VOC 2007에서도 높은 정확도, 예술 작품에서도 정확도 떨어지지 않음



(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.

5. Conclusion

5. Conclusion

- 전체 이미지를 한 번에 train 및 detection
- YOLO는 훈련 단계에서 보지 못한 새로운 이미지에 대해서도 강건하다.
- YOLO는 단순하면서도 빠르고 정확하다
- Fast R-CNN에 비해 background error가 작다
- Fast YOLO는 가장 빠른 object detector이다.