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

1. Introduction

이전의 detection은 객체를 감지하기 위해 이미지에서 region proposal 방법을 사용하여 bounding box를 생성한 다음 이 box에서 분류를 실행한다. 이후 fine tuning과 중복된 box들을 제거하는 과정을 실행하지만 각 구성요소들을 별도로 훈련시켜야 하기 때문에 속도가 느리고 최적화에 어려움을 겪는다.

이 논문에서는 object detection을 이미지 픽셀에서 bounding box coordinates와 클래스 확률을 도출해내는 단일 회귀 문제로 재구성한다. 이를 통해 전체 이미지를 한 번만 사용하여 object detection을 수행한다.

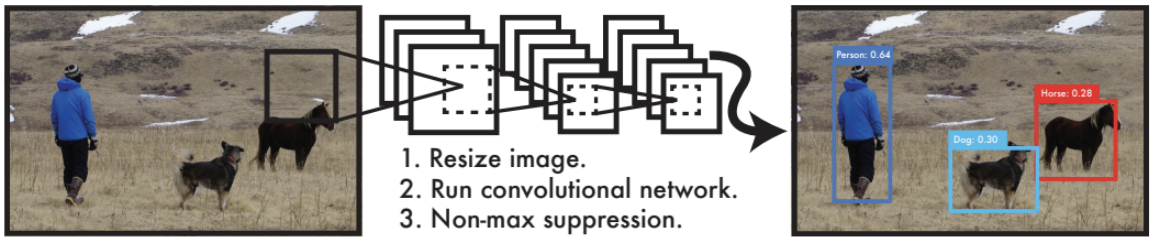


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Figure 1을 보면, 단일 convolution network는 동시에 여러 개의 bounding box와 해당 box에 대한 class의 확률을 예측한다.

다시 말하면, YOLO는 전체 이미지를 학습하고 object detection을 최적화한다. 이러한 과정을 통한 이점은 크게 세 가지로 나눌 수 있다.

1. YOLO는 detection을 회귀 문제로 간주하기 때문에 전체 파이프라인이 복잡하지 않다. 따라서 굉장히 빠른 속도를 가진다.

2. YOLO는 object를 예측할 때 이미지에 대해 전체적으로 추론한다. sliding window나 RPN과는 다르게 전체 이미지를 보기 때문에 클래스에 대한 contextual 정보를 인코딩한다.

YOLO는 이미지의 background를 object로 판단하는 오류가 이전 방법에 비해 절반 이상 낮다.

3. YOLO는 object의 일반화 가능한 representation을 학습한다. YOLO는 일반화가 용이하기 때문에 새로운 도메인에 강력하다.

하지만 이러한 이점에도 불구하고 최신 detection 방법들에 비해 정확도가 떨어진다고 한다. 이미지에서 object를 빠르게 인식할 수 있지만 특히 나눠진 그리드 셀보다 작은 object의 경우 정확한 위치를 찾는데 어려움을 겪는다.

2. Unified Detection

YOLO에서는 object detection의 개별 구성요소를 단일 신경망으로 통합한다. 이 네트워크는 전체 이미지와 모든 object에 대해 global하게 추론한다. 따라서 end-to-end training과 real-time을 높은 정밀도를 유지하면서 가능하게 한다.

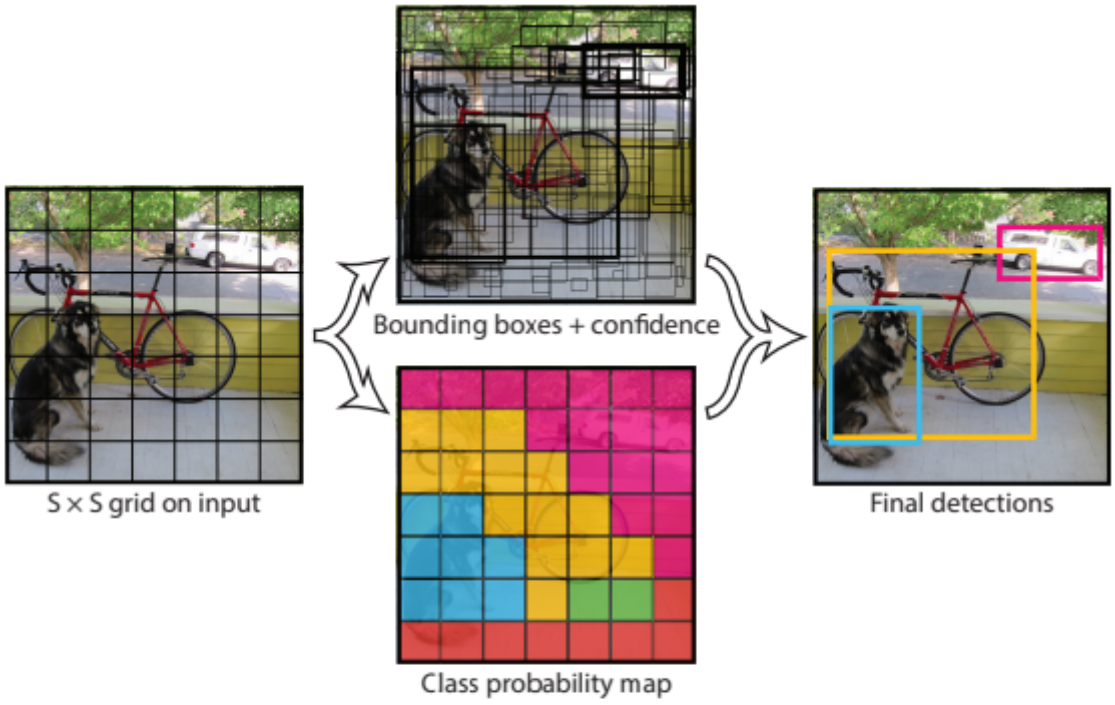


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

PASCAL VOC에서 YOLO를 평가하기 위해 S=7, B=2를 사용한다. PASCAL VOC에는 20개의 레이블이 지정된 클래스가 있으므로 C=20이다. 최종 예측은 7∗7∗30(box 2개의 좌표+신뢰점수+20개 클래스) tensor로 표현된다.

먼저, 입력 이미지를 S x S 그리드로 나눈다. 각 그리드 셀은 그리드 중심에 object가 있다면 그 object를 탐지하며 bounding box B와 해당 box에 대한 confidence score(신뢰 점수)를 예측한다. Confidence score는 모델이 box에 object를 포함하고 있을 확률을 0에서 1 사이 확률로 나타낸다.

또한, 각 그리드 셀은 조건부확률 C인 Pr(Class|Object)를 예측한다. box 수와 상관없이 그리드 셀당 하나의 클래스 확률 집합을 예측한다.

따라서, 이후에 test 시 conditional class probability과 각 box의 confidence prediction을 곱하여 각 box에 대한 특정 클래스별 confidence score를 산출한다.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

즉, box에 나타나는 클래스의 확률과 예측된 box가 object에 얼마나 잘 맞는지 모두 인코딩한다.

2.1 Network Design

YOLO 모델은 CNN으로 구현하고 PASCAL VOC detection dataset에서 평가했다. 네트워크의 conv layer는 이미지에서 feature를 추출하고 fc layer에서는 확률값과 좌표를 예측한다.

네트워크 아키텍처는 GoogLeNet을 기반으로 활용했고, 24개의 conv layer와 2개의 fc layer가 존재한다.

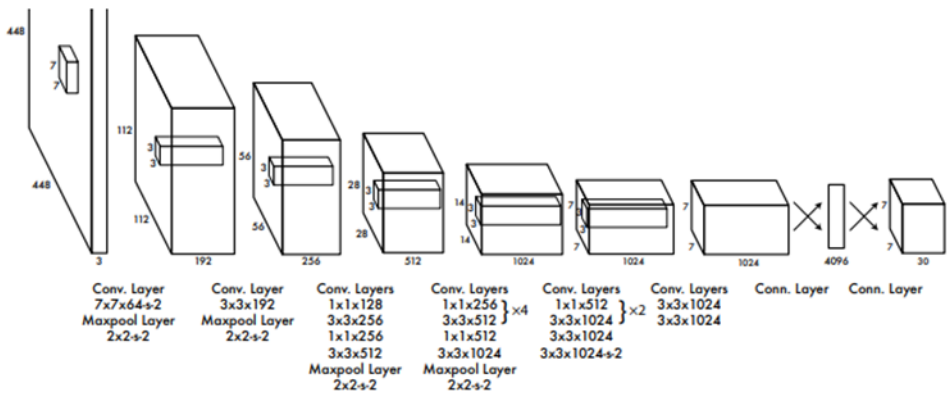


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

2.2 Training

먼저, ImageNet 1000-class dataset에서 conv layer를 pretrain 시켜준다. 여기서는 처음 20개의 conv layer와 average pooling layer, fc layer를 사용한다.

다음으로 detection을 수행하기 위해 모델을 변환한다. pretrain 된 모델에 4개의 conv layer와 2개의 fc layer를 추가한다. 또한 detection에는 fine-grained visual information(세밀한 시각정보)이 필요하기 때문에 네트워크의 해상도를 input size를 224 x 224에서 448 x 448로 높여준다.

마지막 layer에서는 클래스 확률과 bounding box 좌표를 모두 예측한다. 최종 layer에서는 linear activation function을 사용하고 다른 모든 layer에서는 leaky ReLU activation을 사용한다.

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \qquad (2)$$

Leaky Rectified Linear Activation

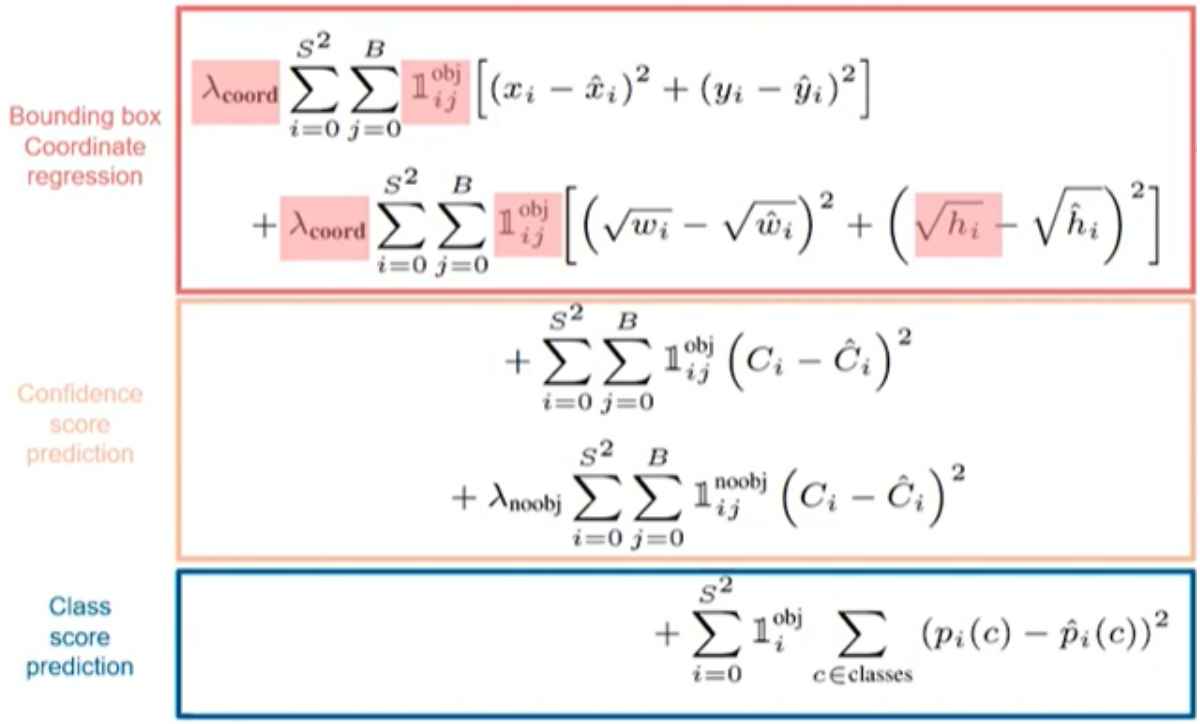
이후, 모델의 출력으로부터 제곱합 오차(Sum-Squared Error)를 최적화한다. Average precision 높여주지는 않지만 최적화하기 쉽기 때문에 제곱합 오차를 사용하였다고 한다. 하지만 이미지에서 object가 아닌 그리드 셀들이 더욱 많고 이 셀들은 confidence score을 0으로 가지기 때문에 object가 있는 그리드 셀의 gradient에도 영향을 미칠 수 있다.

이러한 문제를 해결하기 위해 λ_coord (= 5)와 λ_noobj (= 0.5)의 파라미터를 추가하여 bounding box 좌표 예측으로 인한 loss를 높이고, object가 포함되지 않은 box에 대한 loss를 줄인다.

- λ_coord : coordinates에 대한 loss와 다른 loss들과의 균형을 위한 파라미터
- λ_noobj : object가 있는 box와 없는 box 간 균형을 위한 파라미터

또한, Sum-squared error를 사용하면 bounding box가 큰 객체와 작은 객체에 동일한 가중치를 줄 때도 문제가 생길 수 있다. 작은 object bounding box는 조금만 어긋나도 결과에 큰 영향을 주지만 큰 object bounding box의 경우에는 그렇지 않다. 이를 해결하기 위해서 YOLO에서는 width와 height에 square root를 씌워준다.

다음의 multi-part loss function을 최적화한다.



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

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

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

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

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

1^obj_i : 셀 i에 object가 나타나는 경우 (가장 높은 IoU)

1^obj_ij : 셀 i에 j번째 object가 나타나는 경우

1. x, y 좌표를 truth와 prediction을 제곱하여 error를 계산한다.

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

2. Bounding box의 w, h에 대한 error를 루트를 씌워 계산한다.

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

3. Object를 포함한 bounding box에 대한 confidence error를 계산한다. (Ci = 1)

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

4. Object를 포함하지 않은 bounding box에 대한 confidence error를 계산한다. (Ci = 0)

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

5. p(c) 클래스 확률에 대한 error를 계산한다. 여기서는 object를 포함한 bounding box에 대해서만 계산한다. (pi(c) = 1, if class c is correct, otherwise: 0)

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

논문에서는 아래와 같이 모델을 학습시켰다.

- epoch = 135, batch size = 64, momentum = 0.9, decay = 0.0005
- learning rate scheculing: 첫 epoch에서 0.001로 시작해서 75 epoch까지 0.01으로 학습시킨다. 이후 30 epochs 동안 0.001로 학습하고, 마지막 30 epochs 동안 0.0001으로 학습시킨다.
- overfitting을 막기 위해 dropout(rate = 0.5)과 data augmentation을 활용했다.

2.3 Inference

PASCAL VOC에서 네트워크는 이미지 한 장 당 98개(7x7x2)의 bounding box와 각 box에 대한 클래스 확률을 예측한다.

하지만 YOLO에서의 그리드 디자인은 한 가지 단점이 존재한다. 하나의 object를 여러 그리드 셀이 동시에 검출하는 경우가 있기 때문에 그 object에 대한 bounding box가 여러개 생성될 수 있다. 이 다중 검출(multiple detections) 문제를 해결하기 위해 각 클래스 별로 Non-maximal suppression(NMS)를 사용한다.

2.4 Limitaiton of YOLO

YOLO에서는 각 그리드 셀이 하나의 클래스만 가질 수 있기 때문에 공간적 제약이 존재한다. 따라서 새 떼와 같이 무리를 지어 나타나는 여러 작은 객체가 몰려있는 경우에는 잘 검출하지 못한다.

또한, training 과정에서 학습하지 못한 종횡비(aspect ratio)를 만나면 테스트 단계에서 어려움을 겪는다.

마지막으로, loss function을 훈련하는 동안에 큰 bounding box와 작은 bounding box의 loss에 동일한 가중치를 둔다. 큰 bounding box보다 작은 bounding box는 위치 변화에 따른 IoU변화가 심하기 때문에 localization에 대한 문제가 발생한다.

3. Comparison to other Detection Systems

YOLO 모델을 다른 detection 모델들과의 유사점과 차이점을 비교한다.

Deformable parts models.

Deformable parts models(DPM)은 sliding window 방식을 통해 detection을 수행한다. DPM은 정적 feature을 추출하고 영역을 분류하며 점수가 높은 영역에 대한 bounding box를 예측하는데 분리된 파이프라인을 사용한다. 이에 반해 YOLO는 이 모든 부분을 단일 컨볼루션 신경망으로 대체한다. 네트워크는 feature 추출, bounding box 예측, NMS 및 contextual reasoning을 동시에 수행한다.

R-CNN.

R-CNN과 그 변형은 이미지에서 object를 찾기 위해 region proposal을 사용한다. Selective search는 잠재적 bounding box를 생성하고, CNN 추출 기능을 생성하고, SVM은 상자에 점수를 매기고, 선형 모델은 bounding box를 조정하며, NMS 기능은 중복된 탐지를 제거한다. 이 복잡한 파이프라인의 각 단계는 독립적으로 fine-tuning해야 하며 매우 느려서 시험 시간에 이미지당 40초 이상이 걸린다.YOLO는 각 그리드 셀이 bounding box를 예측하고 그 box에 대해 점수를 계산한다는 부분에서 R-CNN과 유사하다. 하지만 YOLO는 각 그리드 셀의 공간적 제약 때문에 하나의 object가 여러 번 검출되는 경우가 R-CNN에 비해 적다. YOLO는 또한 selective search의 약 2000개에 비해 이미지당 98개로 훨씬 적은 수의 bounding box를 제안한다. 마지막으로, 우리 시스템은 이러한 개별 구성 요소를 공동으로 최적화된 단일 모델로 결합한다.

4. Experiments

먼저 PASCAL VOC 2007에서 다른 모델과 YOLO를 비교한다. YOLO와 R-CNN 계열의 차이를 이해하기 위해 R-CNN의 최고 성능 버전 중 하나인 YOLO와 Fast R-CNN으로 VOC 2007의 오류를 살펴본다. 또한 VOC 2012 결과를 제시하고 mAP를 현재의 최첨단 방법과 비교한다. 마지막으로, 우리는 YOLO가 두 개의 예술품 데이터 세트의 다른 검출기보다 새로운 도메인으로 더 잘 일반화된다는 것을 보여준다.

4.1 Comparison to Other Real-Time Systems

Fast YOLO는 PASCAL에서 가장 빠른 물체 감지 방법이다. 우리가 아는 바로는 현존하는 물체 감지기 중 가장 빠르다. 52.7%의 mAP로 실시간 탐지에 대한 이전 작업보

다 두 배 이상 정확하다. R-CNN Minus R은 selective search을 static bounding box proposal로 대체한다.

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.

FPS : Frames per Second

4.2 VOC 2007 Error Analysis

YOLO와 Fast R-CNN 사이의 차이를 추가로 조사하기 위해 VOC 2007에 대한 결과의 상세 분석을 살펴본다. 여기서 Diagnosing Error in Object Detectors 논문에 소개된 error estimation 방법론을 사용했다. 다음과 같은 기준으로 객체 검출이 정확한지, 틀렸다면 어떤 error type인지를 구분했다.

- Correct: correct class and IOU > .5
- Localization: correct class, .1 < IOU < .5
- Similar: class is similar, IOU > .1
- Other: class is wrong, IOU > .1
- Background: IOU < .1 for any object

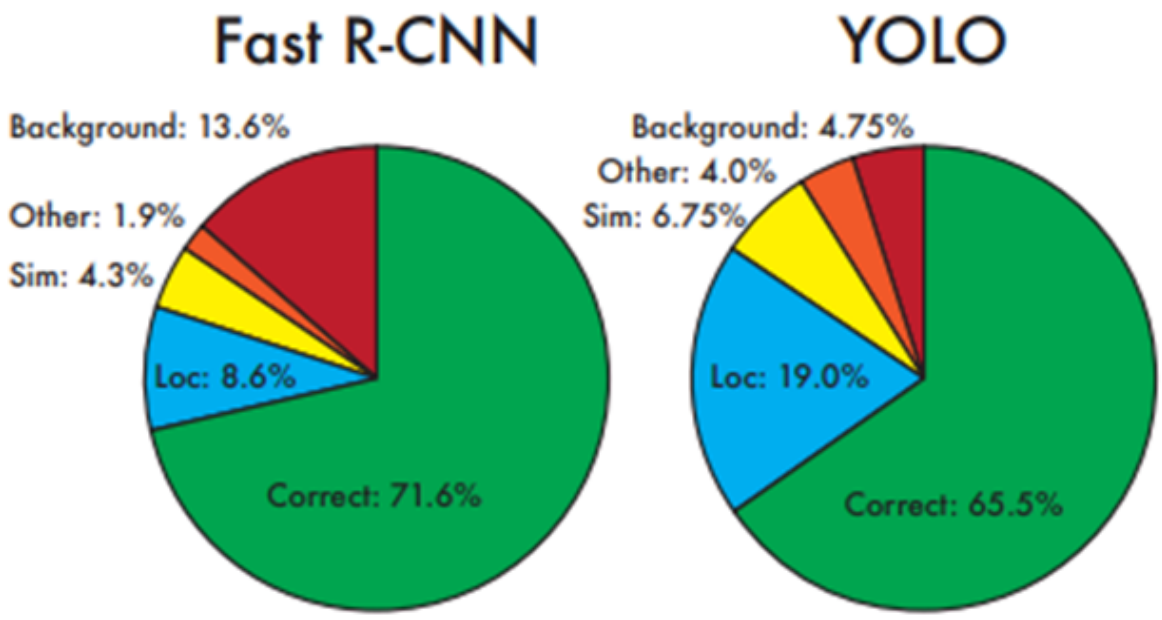


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Figure 4는 20개 클래스 전체에 걸쳐 평균화된 각 오류 유형의 분석을 보여준다.결과를 보면 YOLO는 object의 위치를 올바르게 지정하는 데 어려움을 겪는다. Localization error는 다른 모든 소스를 합친 것보다 더 많은 YOLO error를 차지한다. Fast R-CNN은 localization error를 훨씬 적게 발생시키지만 background error는 훨씬 더 많이 발생한다.

4.3 Combining Fast R-CNN and YOLO

YOLO는 Fast R-CNN에 비해 background error가 훨씬 적다. 따라서 Fast R-CNN에 YOLO를 결합하여 background error를 줄인다면 굉장히 높은 성능을 낼 수 있을 것이다.

R-CNN이 예측하는 모든 boudning box에 대해 YOLO도 유사하게 예측하는지를 체크한다. 만약 R-CNN이 예측한 bounding box와 YOLO가 예측한 bounding box가 유사하다면 두 bounding box가 겹치는 부분을 bounding box로 잡는다.

	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.

이러한 앙상블은 각 모델을 개별적으로 실행한 다음 결과를 결합하기 때문에 YOLO의 속도로부터 이익을 얻지 못한다. 그러나 YOLO는 매우 빠르기 때문에 Fast R-CNN에 비해 계산 시간이 크게 추가되지 않는다.

4.4 VOC 2012 Results

VOC 2012 테스트 세트에서 YOLO는 57.9%의 mAP 점수를 보여준다. 이는 VGG-16을 사용한 원래의 R-CNN에 가까운 최신 기술보다 낮은 수치이다. 우리의 시스템은 다른 모델들에 비해 작은 물체들로 인해 어려움을 겪는다. 'bottle', 'sheep', 'tv/monitor'에 대한 YOLO 점수는 다른 모델보다 8-10% 낮다. 그러나 'cat', 'train' 에서는 YOLO는 더 높은 성능을 달성한다.

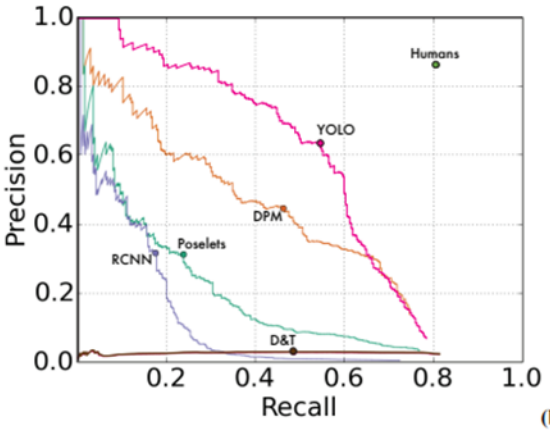
VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	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.

Fast R-CNN과 YOLO 결합된 모델은 가장 성능이 뛰어난 detection 방법 중 하나이다.

4.5 Generalizability: Person Detection in Artwork

Object detection을 위한 Academic 데이터 세트는 동일한 분포에서 훈련 및 테스트 데이터를 도출한다. 우리는 YOLO를 피카소 데이터 세트 및 예술 작품에서 사람 탐지를 테스트하기 위한 두 데이터 세트인 People-Art Dataset에서 다른 모델들과 비교한다.



(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.

Figure 5는 YOLO와 다른 방법 간의 비교 성능을 보여준다. 피카소 모델은 VOC 2012에 대해 훈련되고, People-Art 모델은 VOC 2010에 대해 훈련된다.YOLO는 VOC 2007에서 성능이 우수하고, 예술품에 적용했을 때 다른 방식보다 성능이 저하되지 않는다. YOLO는 객체의 크기와 모양, 객체와 객체가 일반적으로 나타나는 위치 간의 관계를 모델링한다. 예술품과 자연 이미지는 픽셀 수준에서 매우 다르지만 물체의 크기와 모양 면에서 유사하므로 YOLO는 여전히 양호한 성능을 보인다.

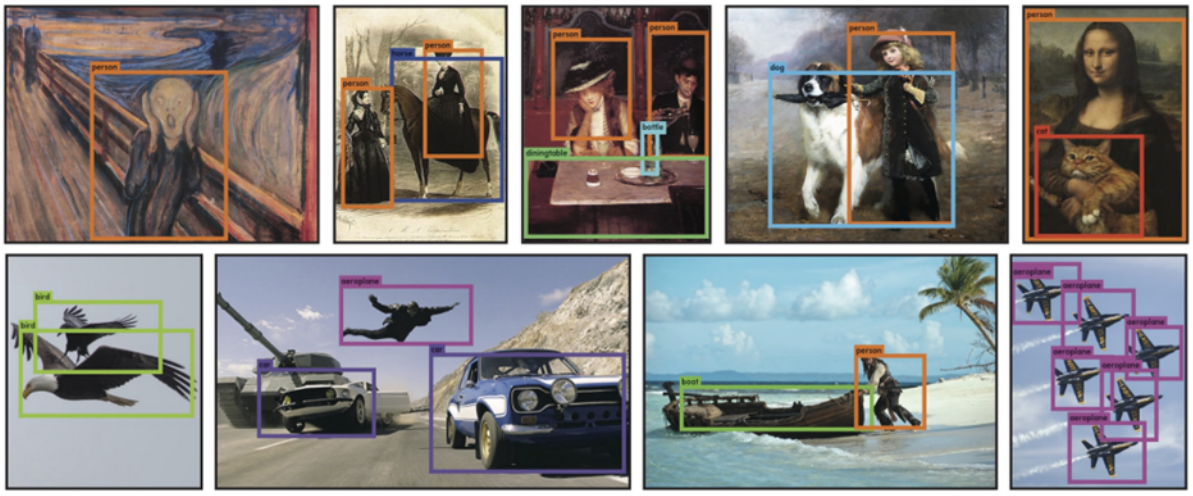


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

5. Real-Time Detection in the Wild

YOLO는 빠르고 정확한 object detector로써 컴퓨터 비전 에플리케이션에 이상적이다. YOLO를 웹캠에 연결하고 카메라에서 이미지를 가져오고 탐지를 표시하는 시간을 포함하여 실시간 성능을 유지하는지 확인할 수 있다.

YOLO는 이미지를 개별적으로 처리하지만 웹캠에 부착하면 추적 시스템처럼 작동해 물체가 움직이고 모양이 변하는 것을 감지한다.

6. Conclusion

YOLO는 구성이 간단하며 전체 이미지에서 직접 학습할 수 있다. Classifier 기반 접근법과 달리 YOLO는 detection performance에 직접 해당하는 loss function에 대해 훈련되며 전체 모델이 공동으로 훈련된다.Fast YOLO는 가장 빠른 object detector이며 YOLO는 real-time object detection에서 최첨단 기술을 구현한다. 또한 YOLO는 새로운 도메인으로 잘 일반화되므로 빠르고 강력한 object detection에 의존하는 애플리케이션에 이상적이다.