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

Abstract

Object Detection 대한 새로운 접근 방식인 YOLO: 1-stage detector로 특징 분류와 classification을 regression문제로 frame화 함, 매우 빠름

1. Introduction

기존 detection 모델은 classfier를 재정의하여 detector로 사용

classification이란 하나의 이미지를 보고 그것이 개인지 고양이인지 판단하는 것

하지만 object detection은 하나의 이미지 내에서 개는 어디에 위치해 있고, 고양이는 어디에 위치해 있는지 판단하는 것 → 따라서 object detection은 classification 뿐만 아니라 위치 정보도 판단해야함 (DPM, R-CNN)

DPM은 이미지 전체를 sliding window로 object detection을 진행, R-CNN은 이미지 안에서 bounding box를 생성하기 위해 region proposal을 하고, 제안된 bounding box에 classifier를 적용하여 classification을 진행, 그 다음 bounding box를 조정하고, 중복된 검출을 제거하고, 객체에 따라 box의 점수를 재산정하기 위해 post-processing을 진행 → 이런 복잡함 때문에 R-CNN은 느리고 각 절차를 독립적으로 훈련시켜야 해서 optimization에 어려움

그래서, object detection을 single regression problem으로 재정의: 이미지의 픽셀로부터 bounding box의 위치(coordinates), 클래스 확률(class probabilities)을 구하기까지의 일련을 절차를 하나의 회귀 문제로 재정의

이미지를 한 번만 보면 객체를 검출할 수 있다 하여 이름이 YOLO(you only look once)

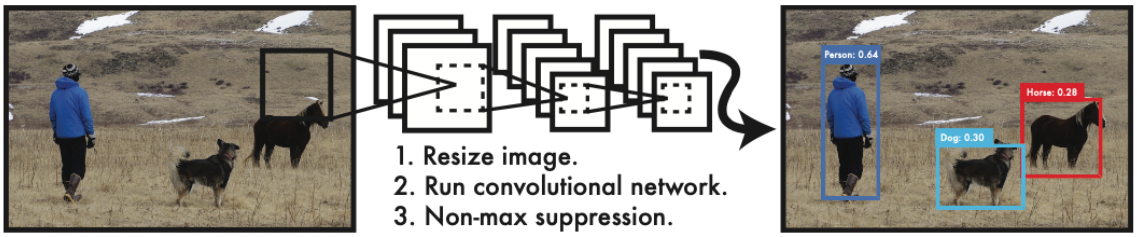


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

→하나의 convolutional network가 여러 bounding box와 그 bounding box의 클래스 확률을 동시에 계산, YOLO는 이미지 전체를 학습하여 곧바로 detection performance을 최적화

YOLO의 이런 통합된 모델은 기존의 객체 검출 모델에 비해 여러 가지 장점이 존재

1. YOLO는 굉장히 빠름: 왜냐하면 YOLO는 기존의 복잡한 object detection process를 single regression problem 바꾸었기 때문
2. YOLO는 예측을 할 때 이미지 전체를 사용: sliding window나 region proposal 방식과 달리, YOLO는 훈련과 테스트 단계에서 이미지 전체를 보고, 그래서 클래스의 모양에 대한 정보뿐만 아니라 주변 정보까지 학습하여 처리 → 아무 물체가 없는 background에 반점이나 노이즈가 있으면 그것을 물체로 인식하는 background error가 Fast R-CNN에 비해 적음
3. YOLO는 물체의 일반적인 부분을 학습: 다른 모델에 비해 YOLO는 훈련 단계에서 보지 못한 새로운 이미지에 대해 더 robust함

하지만, YOLO는 빠르게 객체를 검출할 수 있다는 장점은 있지만 정확성이 다소 떨어짐

2. Unified Detection

YOLO는 input images를 S x S grid로 나눔: 만약 어떤 객체의 중심이 특정 grid cell 안에 위치한다면, 그 그리드 셀이 해당 객체를 검출해야 함

각각의 grid cell은 B개의 bounding box와 그 bounding box에 대한 confidence score를 예측

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

IOU(intersection over union)는 실제 bounding box와 예측 bounding box의 합집합 면적 대비 교집합 면적의 비율: IOU = (실제 bounding box와 예측 bounding box의 교집합) / (실제 bounding box와 예측 bounding box의 합집합)

그리드 셀에 아무 객체가 없다면 Pr(Obejct)=0이고, confidence score도 0 , 그리드 셀에 어떤 객체가 확실히 있다고 예측했을 때, 즉 Pr(Object)=1일 때가 가장 이상적

→ 따라서 confidence score가 IOU와 같다면 가장 이상적인 score

각각의 bounding box는 5개의 예측치로 구성(x, y, w, h, confidence): (x, y) 좌표 쌍은 bounding box 중심의 그리드 셀(grid cell) 내 상대 위치(0~1 사이의 값) / (w, h) 쌍은 bounding box의 상대 너비와 상대 높이(0~1 사이의 값) / confidence=confidence score(Pr*IOU)

$$\Pr(\text{Class}_i|\text{Object})$$

각각의 grid cell은 C(conditional class)를 예측

→ 그리드 셀 안에 객체가 있다는 조건 하에 그 객체가 어떤 class인지에 대한 조건부 확률: 그리드 셀에 몇 개의 bounding box가 있는지와는 무관하게 하나의 그리드 셀에는 오직 하나의 class에 대한 probability만 구함

테스트 단계에서는 conditional class probability(C)와 개별 bounding box의 confidence score를 곱해주는데, 이를 각 bounding box에 대한 class-specific confidence score라고 함

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

→ class-specific confidence score는 conditional class probability와 confidence score를 곱한 값

이 score는 bounding box에 특정 class 객체가 나타날 확률(=Pr(Class_i))과 예측된 bounding box가 그 클래스(class) 객체에 얼마나 잘 들어맞는지(=IOU_pred^truth)를 나타냄

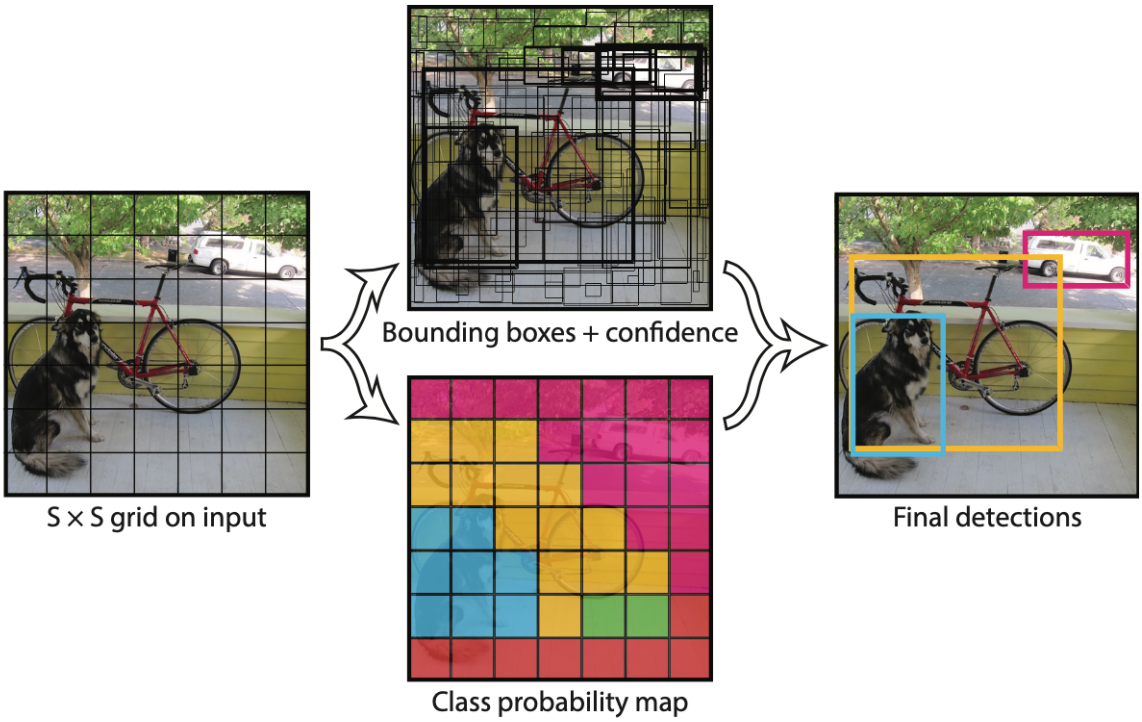


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

2.1. Network Design

YOLO의 신경망 구조는 image classification에 사용되는 GoogLeNet에서 따옴: 총 24개의 convolutional layers과 2개의 fully connected layers으로 구성되어 있고, GoogLeNet의 인셉션 구조 대신 YOLO는 1 x 1 reduction layer과 3 x 3 convolution layer 결합을 사용

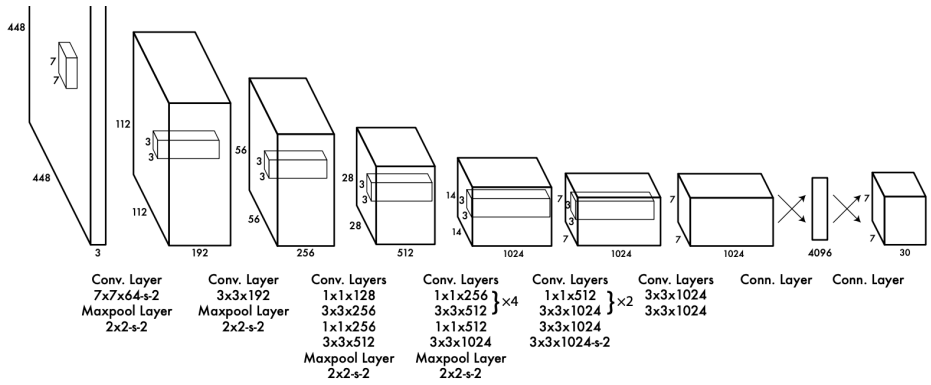


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

→ 최종 output은 7 x 7 x 30의 prediction tensors

2.2. Training

1000개의 class를 갖는 ImageNet 데이터 셋으로 YOLO의 convolution layer을 pretrain: pretrain을 위해서 24개의 convolution layer 중 첫 20개의 convolution layer만 사용했고, 이어서 average-pooling layer과 fully connectied layer을 연결 → ImageNet 2012 validation 데이터 셋에서 88%의 정확도 기록

ImageNet은 classification를 위한 데이터 셋이어서, object detection 모델로 바꾸어야 함 → pretrain된 20개의 convolution layer 뒤에 4개의 convolution layer 및 2개의 fully connected layer를 추가해 성능을 향상시켰고 입력 이미지의 해상도를 224 x 224에서 448 x 448로 증가시킴

이 신경망의 최종 output은 class probabilities과 bounding box 위치정보(coordinates)

YOLO 신경망의 마지막 계층에는 linear activation function를 적용했고, 나머지 모든 계층에는 leaky ReLU를 적용

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

YOLO의 loss는 SSE(sum-squared error)를 기반으로 해서, 최종 output의 SSE를 optimize 해야함

하지만, SSE를 최적화하는 것이 YOLO의 최종 목적인 mAP(평균 정확도)를 높이는 것과 완벽하게 일치하지 않음: bounding box의 위치를 얼마나 잘 예측했는지에 대한 loss인

localization loss와 class를 얼마나 잘 예측했는지에 대한 loss인 classification loss가 있는데, 두 loss의 가중치를 동일하게 두고 학습시키는 것은 좋은 학습이 아님

또 다른 문제가 있는데, 배경 영역이 전경 영역보다 더 크기 때문에 그리드 셀에 객체가 없다면 confidence score=0이고, 대부분의 그리드 셀의 confidence score=0이 되게 학습할 수밖에 없고 이는 모델 불균형을 초래

이를 개선하기 위해, 객체가 존재하는 bounding box 좌표(coordinate)에 대한 loss의 가중치를 증가시키고, 객체가 존재하지 않는 bounding box의 confidence loss에 대한 가중치는 감소: 이를 위해 두 개의 파라미터를 사용했는데, λ_coord와 λ_noobj이며, λ_coord=5, λ_noobj=0.5로 가중치 셋팅

또 다른 문제로, SSE는 큰 bounding box와 작은 boudning box에 대해 모두 동일한 가중치로 loss를 계산하지만 작은 bounding box가 큰 bounding box보다 작은 위치 변화에 더 민감함 → 이를 개선하기 위해 bounding box의 width와 height에 square root를 취함: 너비와 높이가 커짐에 따라 그 증가율이 감소해 loss에 대한 가중치를 감소시키는 효과

YOLO는 하나의 grid cell에 여러 개의 bounding box를 예측하는데, training단계에서 예측된 여러 bounding box 중 실제 객체를 감싸는 ground-truth bounding box와의 IOU가 가장 큰 것 하나의 bounding box를 선택

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

loss function

1_i^obj: grid cell i 안에 객체가 존재하는지 여부(존재하면 1, 존재하지 않으면 0)

1_ij^obj: grid cell i의 j번째 bounding box predictor가 사용되는지 여부

(1) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, x와 y의 loss를 계산

(2) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, w와 h의 loss를 계산

(3) Object가 존재하는 그리드 셀 i의 bounding box predictor j에 대해, confidence score의 loss를 계산(Ci = 1)

(4) Object가 존재하지 않는 그리드 셀 i의 bounding box predictor j에 대해, confidence score의 loss를 계산. (Ci = 0)

(5) Object가 존재하는 그리드 셀 i에 대해, conditional class probability의 loss를 계산. (p_i(c)=1 if class c is correct, otherwise: p_i(c)=0)

λ_coord: coordinates(x, y, w, h)에 대한 loss와 다른 loss들과의 균형을 위한 balancing parameter

λ_noobj: 객체가 있는 box와 없는 box 간에 균형을 위한 balancing parameter

VOC 2007, 2012 훈련 및 검증 데이터 셋을 활용하여 135 epochs로 YOLO 모델을 훈련: batch size=64, momentum=0.9, decay=0.0005로 설정하고, 초반에는 learning rate를 0.001에서 0.01로 천천히 상승시키고 이후 75 epoch 동안에는 0.01, 30 epoch 동안에는 0.001, 그리고 마지막 30 epoch 동안은 0.0001로 learning rate를 설정, overfitting을 막기 위해 dropout=0.5를 적용하고 data augmentation 진행

2.3. Inference

파스칼 VOC 데이터 셋에 대해서 YOLO는 한 이미지 당 98개의 bounding box를 예측해주고, 그 bounding box마다 class probabilities을 구함

하지만 YOLO의 grid design에서, 하나의 객체를 여러 그리드 셀이 동시에 검출하는 경우 bounding box가 여러 개 생길 수 있음(multiple detections): non-maximal suppression을 통해 개선

2.4. Limitations of YOLO

1. YOLO는 하나의 그리드 셀마다 두 개의 bounding box를 예측 → spatial constraints: 새 떼와 같이 작은 물체가 몰려 있는 경우 공간적 제약 때문에 객체 검출이 제한적임
2. YOLO 모델은 데이터로부터 bounding box를 예측하는 것을 학습하기 때문에 훈련 단계에서 학습하지 못했던 새로운 aspect ratio를 마주하면 힘듬
3. YOLO 모델은 큰 bounding box와 작은 bounding box의 loss가 동일한 가중치 → incorrect localization: 큰 bounding box에 비해 작은 bounding box가 위치 변화에 따른 IOU 변화가 더 심함

3. Comparison to Other Detection Systems

Deformable parts models(DPM): sliding window 방식을 사용, 서로 분리된 파이프라인으로 구성되어 있어 독립적인 파이프라인이 각각 feature extraction, region classification, bounding box prediction을 수행

R-CNN: region proposal 방식을 사용, selective search라는 방식으로 여러 bounding box를 생성하고, convolution neural network로 feature를 추출하고, SVM으로 bounding box에 대한 점수를 측정후 linear model로 bounding box를 조정하고, non-max suppression로 중복된 검출을 제거

Deep MultiBox: R-CNN과 달리 Selective search를 사용하는 대신 region을 예측하기 위해 convolution neural network를 훈련, single object detection만 가능

OverFeat: sliding window detecdtion을 효율적으로 수행하지만 disjoint system

MultiGrasp: 하나의 객체를 포함하는 이미지에 대해 하나의 region만 예측

4. Experiments

YOLO를 real-time detection system과 비교 (Fast R-CNN은 이 논문이 나온 시점을 기준으로 성능이 가장 좋은 R-CNN 계열의 모델

4. 1. Comparison to Other Real-Time Systems

Fast YOLO는 PASCAL 데이터 셋 기준으로 가장 빠른 객체 검출 모델(mAP 52.7%)

또한 VGG-16을 사용하여 YOLO를 훈련→ 일반 YOLO에 비해 mAP가 더 높지만 속도는 느렸음

Faster R-CNN 모델은 bounding box proposal을 위해 selective search를 사용하지 않고 신경망을 사용했고, 기존의 R-CNN 계열보다는 속도가 빨라졌지만 여전히 YOLO에 비해서 느림

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.

FPS가 30이라는 것은 1초에 30 프레임의 영상을 처리한다는 뜻

4. 2. VOC 2007 Error Analysis

VOC 2007 데이터 셋에 대해 YOLO와 Fast R-CNN의 성능을 비교(객체 검출이 정확한지, 틀렸다면 어떤 error type인지를 구분)

Correct : correct class and IOU > 0.5

Localization : correct class, 0.1 < IOU < 0.5

Similar : smilar class, IOU > 0.1

Other : wrong class, IOU > 0.1

Background : any object IOU < 0.1

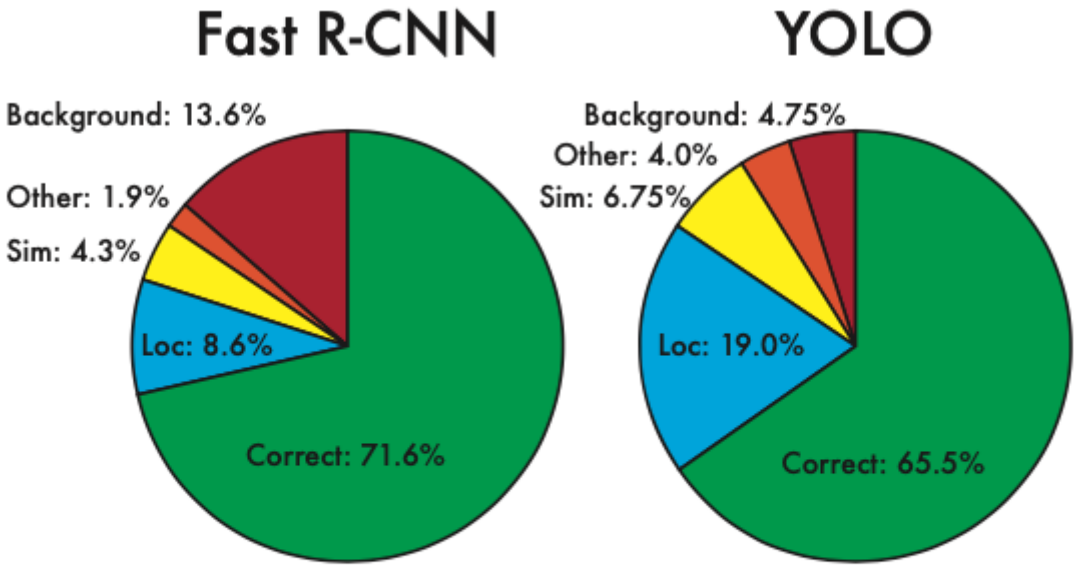


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

YOLO는 localization error가 상대적으로 크고, Fast R-CNN은 YOLO에 비해 localization error가 작음

backgound error는 배경에 아무 물체가 없는데 물체가 있다고 판단하는 false positive error이고, Fast R-CNN은 YOLO에 비해 background error가 3배정도 큼

4. 3. Combining Fast R-CNN and YOLO

YOLO는 Fast R-CNN에 비해 background error가 훨씬 적음→ Fast R-CNN에 YOLO를 결합하여 background error를 줄인다면 굉장히 높은 성능을 낼 수 있을 것: 두 bounding box가 겹치는 부분을 bounding box로 잡으면 됨

	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.

물론 Fast R-CNN과 YOLO를 결합한 모델은 YOLO에 비해 느리지만, YOLO가 빨라서 Fast R-CNN을 사용하는 것보다는 Fast R-CNN과 YOLO를 결합한 모델을 사용하는 것이 나음

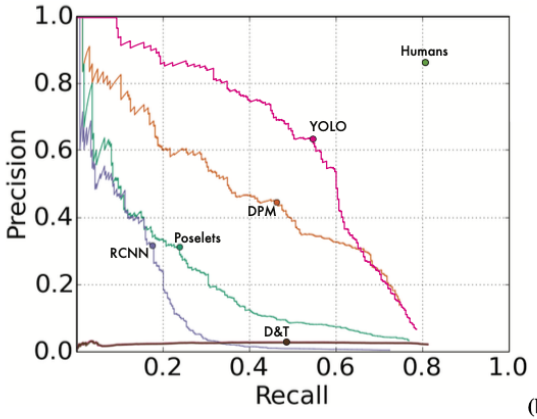
4. 4. VOC 2012 Results

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. 5. Generalizability: Person Detection in Artwork

train 데이터 셋과 다른 분포를 지닌 test 데이터 셋(train 데이터 셋에서 보지 못한 새로운 데이터 셋)을 활용하여 테스트: 피카소 데이터 셋과 일반 예술 작품을 사용



(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.

→ PASCAL VOC 2007에서 학습한 YOLO, R-CNN, DPM 등의 성능을 서로 비교: YOLO는 VOC 2007에서도 가장 높은 정확도를 보였고, 예술 작품에 대해서도 정확도가 크게 떨어지지 않음

5. Real-Time Detection In The Wild

YOLO를 웹캠과 연결하여 실시간(real-time) 객체 검출을 수행

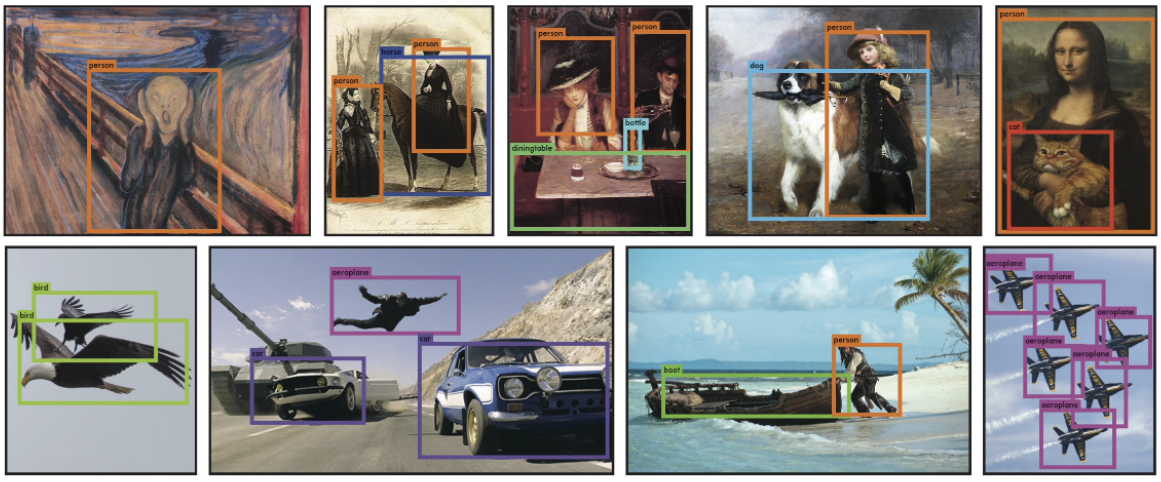


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

6. Conclusion

Object detection을 위한 unified model인 YOLO: YOLO는 detection performance에 해당하는 loss function에 대해 훈련되고 전체 모델은 공동으로 훈련됨

Fast YOLO는 가장 빠른 general-purpose object detection

YOLO는 새로운 도메인으로 잘 일반화되어 빠르고 robust한 object detection에 의존하는 애플리케이션에 이상적