

YOLO

You Only Look Once: Unified, Real-Time Object Detection 2023.03.31

Abstract

우리는 Object detection의 새로운 접근으로 YOLO를 소개합니다. 이전 연구에서 classifier를 사용해왔지만, 우리는 object detection을 bounding box를 분류하고 class 확률을 관련짓는 회귀 문제로 초점을 맞췄다. 한개의 신경망은 BB(Bounding Box)와 class확률을 한 번에 평가에서 전체 이미지를 통해 예측한다. 전체적인 detection 파이프라인이 단독 네트워크로 이루어져 있어 end-to-end로 수행이 가능하다.

우리의 통합된 구조는 매우 빠릅니다. 우리의 기본 모델은 초당 45프레임의 속도를 자랑합니다. 더 작은 Fast YOLO 모델의 경우 다른 실시간 검출기에 비해 2배 높은 mAP 스코어와 초당 155프레임의 속도를 달성합니다. YOLO는 다른 최첨단 모델과 비교하여 localization 오류를 더 많이 만들어냅니다. 그러나 배경에 대한 false positive는 적습니다. 따라서 YOLO는 객체를 잘 감지하고 구분합니다.

Introduction

사람들은 이미지를 전체적으로 보며 물체가 어디에 있는지, 어떠한 행동을 하고 있는지 즉각적으로 파악합니다. 또한, 빠르고 정교하며 이는 운전과 같은 복잡한 일에 대해 적은 의식적인 생각을 통해 수행합니다. 특별한 센서 없이 자율주행 object detection에 대한 빠르고 정확한 알고리즘은 사람들에게 실시간적으로 정보를 전달해주며, 범용적이고 반응적인 로봇의 잠재성을 확장시켜줍니다.

현재 detection 시스템은 classifier를 사용하고 있습니다. 객체를 감지하기 위해서, 객체에 대한 classifier를 사용하며 다양한 위치와 크기에 대해 평가합니다. DPM과 같은 model은 전체 이미지를 간격을 둘며 classifier가 작동하는 sliding window 방식을 사용합니다.

더 최근의 연구에서 R-CNN은 이미지에서 BB를 생성하기 위해 region-proposal을 사용합니다. 그리고 이러한 boxes에 대하여 classifier를 작동시킵니다. classifier 이후, 이전의 작업은 bounding box를 수정하기 위해 사용되어지며, 중복으로 탐지된 것을 제거하고, score를 재산정합니다. 이러한 복잡한 pipeline은 각각의 요소가 개별적으로 학습되어져야 하기 때문에 학습하기 어렵습니다.

우리는 object detection을 image pixel로 부터 BB의 영역과 class probability를 나타내는 single regression problem으로 초점을 다시 두었습니다. YOLO를 사용한다면 한 번에 객체가 어디에 있는지 어떤 객체인지 알 수 있습니다. YOLO는 상당히 간단하며 단독적인 convolution network가 여러개의 BB와 class probability를 예측합니다. 이러한 통합적인 모델은 object detection에서의 이전 모델을 넘어 몇몇의 이점을 가져다 줍니다.

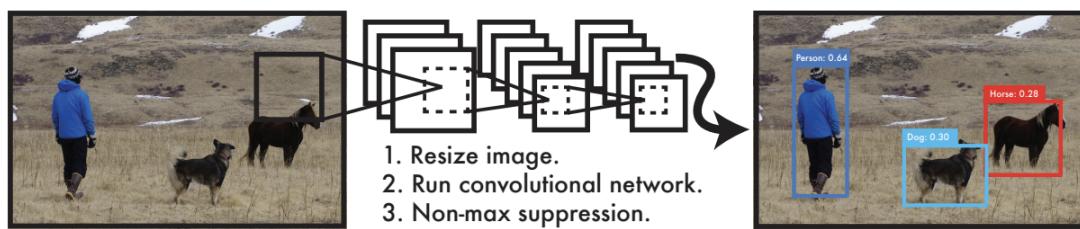


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

먼저, YOLO는 너무나도 빠릅니다. detection을 회귀 문제로 해결하기 때문에 더 이상의 복잡한 구조가 필요 없습니다. 우리는 테스트로 새로운 이미지를 예측하기 위해 YOLO를 사용해본 결과 45fps를 기록하였고, 더욱 간단한 구조에서는 150fps를 기록했습니다. 이는 스트리밍 영상에 대해서 25ms의 지연안에서 실시간으로 detection을 진행할 수 있음을 의미합니다. 게다가 YOLO는 다른 실시간 탐지 모델보다 2배 이상의 mAP를 기록합니다.

두번째로 YOLO는 예측을 할 때 이미지 전체로부터 이유를 찾습니다. sliding window나 region proposal기반의 모델과는 달리 학습과정과 테스트과정에서 전체 이미지를 학습하여 클래스와 모양에 대해 전체적인 문맥정보를 도출합니다. 상위의 detection model인 Fast R-CNN은 이미지 전체를 학습하지 않기 때문에 간혹 배경을 물체로 판단합니다. YOLO는 이에 비해 이와 같은 오류를 덜 범합니다.

세번째로 YOLO는 객체에 대해 잘 학습합니다. 자연스러운 이미지에 대해서 학습하고 나서 미술 작품에 대해 테스할 때, YOLO는 다른 모델에 비해 더 뛰어난 수행을 합니다. 새로운 도메인이나 예상치 못한 데이터가 들어올 경우 더 일반화를 잘 시키기 때문입니다.

YOLO는 여전히 정확도 면에서는 sota model 뒤에 놓여져 있습니다. 반면에 빠르게 물체를 확인하며 작은 물체라도 정밀하게 잡아냅니다. 우리는 실험을 통해 이러한 tradeoff관계를 실험합니다.

Unified Detection

Object detection의 각 요소를 하나의 네트워크로 통합합니다. 이 네트워크는 각각의 바운딩 박스를 예측하기 위해 전체 이미지를 사용합니다. 또한 모든 BB는 모든 클래스에 대해서 확률을 구합니다. YOLO는 end-to-end 학습을 하며 준수한 정확도를 유지함과 동시에 실시간적인 스피드를 냅니다.

먼저 이미지를 $S \times S$ grid로 나눕니다. 객체의 중심이 grid cell안에 있는 경우, 그 grid cell은 객체를 탐지해야 합니다. 각각의 grid cell은 B 바운딩 박스를 예측하고 box에 대해 confidence score를 계산합니다. confidence score는 물체가 box안에 존재할 확률과 box의 정확성에 대해 나타냅니다. 우리는 이를 $Pr(\text{Object}) * IOU(\text{truth}, \text{pred})$ 로 정의합니다. cell에 객체가 없을 경우, confidence score는 0이 됩니다. 우리는 confidence score가 IOU와 동등하기를 원한다.

각각의 BB는 5개의 예측 x, y, w, h 와 confidence로 이루어진다. (x, y) 는 grid cell에서의 상대적인 객체의 중심 위치를 나타낸다. 또한 (w, h) 는 BB의 너비와 높이를 뜻한다. 최종적으로 confidence prediction은 예측 BB와 ground truth BB사이의 IOU값을 나타낸다.

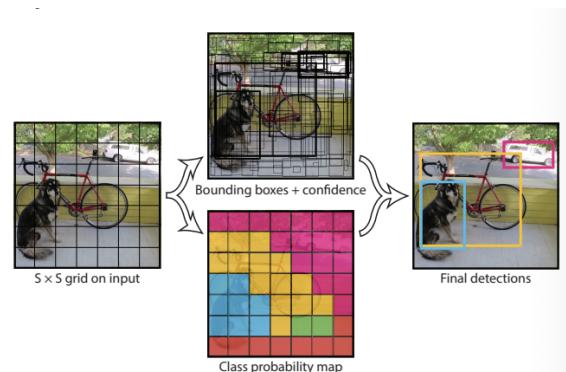


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

각각의 BB는 모든 class의 확률을 $Pr(\text{Class}_i | \text{Object})$ 로 예측한다. 이 확률은 grid cell이 객체를 포함하는 조건에서 구성되어진다.
 $Pr(\text{Class}_i | \text{Object}) * Pr(\text{Object}) * IOU(\text{truth}, \text{pred}) = Pr(\text{Class}_i) * IOU(\text{truth}, \text{pred})$

- 최종적으로 모든 grid cell은 A개의 Bounding Box를 가지며 N개의 class를 예측한다고 할 때, 각각의 grid cell은 $A * 5(\text{confidence}, x, y, w, h) + N(\text{Class EA})$ 를 예측해야 하며 결국 $7 * 7$ (grid cell EA) * ($A * 5(\text{confidence}, x, y, w, h) + N(\text{Class EA})$)의 tensor를 가지게 된다.

Network Design

YOLO는 convolution network에서는 fc layer가 최종 확률을 예측하는 동안 초기 conv layer가 이미지에서 특징을 추출한다. 우리는 GoogLeNet에서 영감을 받았다. YOLO는 24개의 conv layer와 2개의 fc layer로 이루어진다. GoogLeNet의 인셉션 모듈을 사용하는 대신 1×1 reduction layer와 3×3 conv layer를 사용한다.

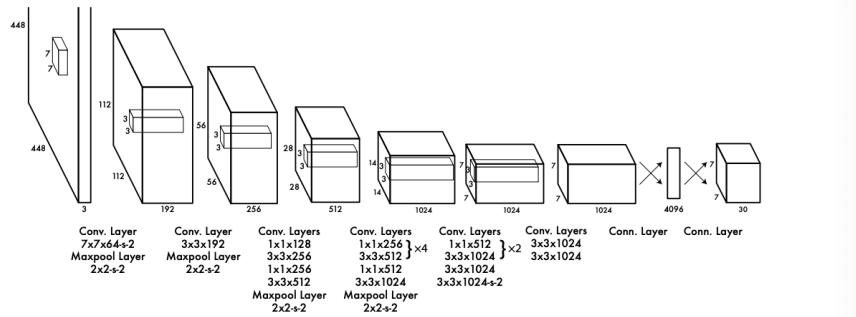
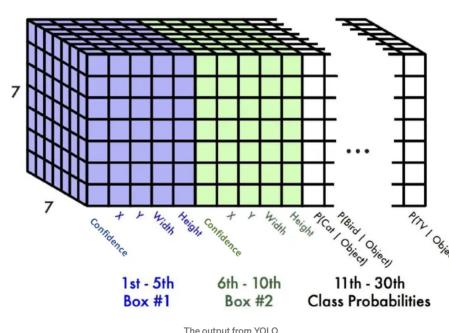


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Fast YOLO의 경우 24개의 conv layer 대신 9개의 conv layer를 사용한다. 최종적인 final output의 경우 $7 \times 7 \times 30$ tensor로 이루어진다. → Gridcell의 개수 * (각 Grid cell 내 Bounding Box의 개수 * 5 + 예측하는 class의 개수)



Training

우리는 convolution layers를 ImageNet을 이용해 pretrain을 시켰다. pretrain을 위해서 초기 20개의 conv layer와 averaging pooling layer와 fc layer를 사용했다. 1주일간 학습을 진행하였고 ImageNet2012 validation set에서 88%의 성능을 냈다. 이후 랜덤 가중치 초기화를 진행한 4개의 conv layer와 connected layer를 더해주었다. 그리고 detection은 시각 정보에 의해 영향을 많이 받기 때문에 input의 해상도를 224×224 에서 448×448 로 늘려주었다. 마지막 레이어에서 class의 확률과 BB의 정보를 예측하며 BB의 정보는 정규화를 통해 x, y, w, h 는 $0 \sim 1$ 사이에서의 값을 갖는다. 그리고 final layer 뒤에서의 활성화 함수는 leaky Relu를 사용한다.

SSE 에러를 통해 학습을 진행한다. SSE 에러는 학습을 빠르게 진행시키지만 mAP에 대해서는 정확히 작동하지 않는다. 이는 로컬레이제이션과 분류 오류를 동등하게 해서 학습을 진행시키지만 완벽하지 않을 수 있습니다. 모든 grid cell은 객체를 포함하지 않는데 이는 종종 cell의 객체가 없는 것으로 학습 되어져 모델의 불안정성을 이끌며 조기종료를 유발한다. 이를 해결하기 위해서 BB의 위치 예측값에 대한 loss를 증가시키고 confidence prediction에 대한 loss값을 파라미터 $\lambda_{coord} = 5$, $\lambda_{noobj} = 0.5$ 로 하여 조정한다.

SSE 에러는 또한 큰 BB는 작은 BB는 동등하게 오차값을 갖는다. SSE는 큰 BB에서 작은 오차는 작은 BB에서의 같은 오차보다 작은 오차를 반영한다. 이에 따라 너비와 높이에 대해 제곱근을 취해 부분적으로 문제를 해결합니다.

YOLO는 grid cell마다 여러 BB를 예측합니다. 학습하여 각 box 예측기는 하나의 가장 높은 성능을 나타내는 BB를 찾아내기 위해 노력하며 이는 전문화가 이루어질 수 있습니다. 최종적인 로스 산정 방식은 다음과 같습니다.

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

Inference

학습을 위해서 우리는 이미지마다 98개의 BB를 찾아냈고 BB마다 class의 확률을 예측했습니다. grid 디자인은 BB예측에서 공간적 다양성을 강화시킵니다. 각각의 grid cell에서 객체를 명확히 잡아냅니다. 그러나 큰 물체나 경계 근처에 여러 셀에 있는 물체는 여러 셀에 의해 더욱 쉽게 포착됩니다. 그리고 NMS는 여러개 중복 detection에 사용되어지면 추가했을 경우 2~3% 정도의 추가 mAP성능을 얻을 수 있었습니다.

Limitations of YOLO

YOLO는 grid cell에서 두개의 BB와 하나의 class만 예측할 수 있는 제약이 있습니다. 이는 근처에 있는 다중 물체, 예로 작은 새같은 경우에 대하여 제약이 생깁니다.

그리고 bounding box를 통해 예측을 하기 때문에, 새롭거나 보지 못한 종횡비와 구성에 제약이 생깁니다.

마지막으로 우리는 큰 상자와 작은 상자가 동일한 오차를 가질 경우 작은 상자의 오류의 경우 더 큰 loss값을 갖게 됩니다.

Comparison to Other Detection Systems

Experiments

먼저 다른 real-time detection model과 비교합니다. YOLO와 R-CNN의 차이를 이해해시키기 위해 background false positive에서 더 월등하다는 것에 대해 보여줍니다. 최종적으로 우리는 두개의 다른 데이터 셋 상에서 YOLO가 새로운 도메인에서 다른 모델보다 뛰어나다는 것을 보여줍니다.

Comparision to Other Real-Time Systems

Real-time detection에 대한 많은 연구가 이루어졌지만, 아직까지는 DPM model만이 30FPS 이상을 달성하고 있습니다.

우리는 mAP성능과 속도에 대한 trade-off관계에 대해 살펴봅니다. fast YOLO가 가장 빨랐으며 52.7% mAP를 기록합니다. 이는 다른 real-time detection에 비해 두배 이상 정확합니다. YOLO는 63.4% mAP를 기록합니다. 우리는 YOLO를 vgg-16을 통해서 학습해 본 결과 성능이 약간 올라가지만 속도가 느려지는 것을 확인했습니다. R-CNN의 selective search를 static bounding box proposals로 교체한 결과 속도는 빨라졌으나 성능에서 좋지 못했습니다. 이와 같이 mAP성능과 속도에 대한 trade-off관계를 증명할 수 있었습니다.

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
<hr/>			
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

VOC 2007 Error Analysis

YOLO와 Fast R-CNN에서의 차이점을 더 살펴보기 위해 예측 결과에 대해서 살펴봅니다.

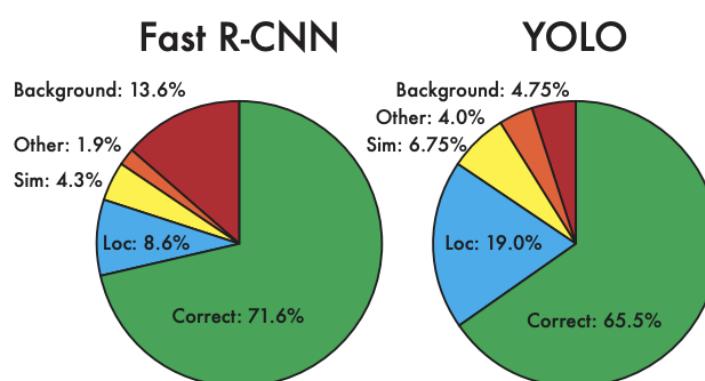


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories ($N = \#$ objects in that category).

- Correct : $\text{IOU} > 0.5$
- Localization : correct class, $0.1 < \text{IOU} < 0.5$
- Similar : class is similar, $\text{IOU} > 0.1$
- Other : class is wrong, $\text{IOU} > 0.1$
- Background : $\text{IOU} < 0.1$ for any object

이 원형 그래프를 살펴 보면 Fast R-CNN은 Background error가 많이 생기는 반면 YOLO는 Localization에서 오류가 많이 발생하고 있습니다.

Combining Fast R-CNN and YOLO

YOLO는 Fast R-CNN보다 background 오류가 적게 발생합니다. YOLO를 Fast R-CNN에서 background detection으로 사용한다면 더 좋은 성능을 얻을 수 있습니다. 모든 BB에 대해 Fast R-CNN과 YOLO와 비슷하게 생성하는지 확인합니다. YOLO에 의거해 예측을 하며 두 개의 박스를 결합합니다.

	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

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

두개를 합칠 경우 상당한 성능을 갖게 되지만 이럴 경우, 이전의 YOLO의 스피드는 더 이상 유지하기 어렵습니다.

Generalizability

R-CNN의 경우 detection을 위해서 좋은 proposal을 필요로 합니다. 하지만 art data의 경우 실제 현실 데이터와 비교하면 RGM pixel값이 다른 경우가 많아, proposal을 잘 잡아내지 못하고 성능이 급격하게 떨어지는 것을 볼 수 있습니다. 반면, DPM과 YOLO의 경우 객체의 크기와 모양에 대해서 학습을 하기에 성능이 많이 떨어지지 않는 모습을 보여줍니다. 다만 DPM은 애초에 성능이 낮은 모습을 보이고 있습니다.

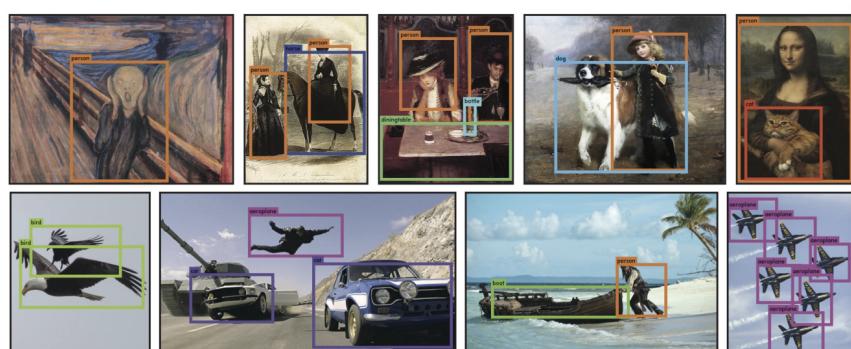
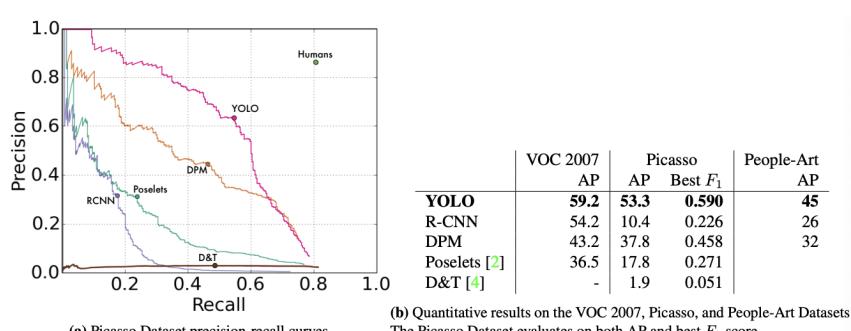


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

Real-Time Detection In The Wild

YOLO는 computer vision 활용을 다룰 수 있는 빠르고, 정확한 object detector입니다. 우리는 YOLO를 webcam에 연결했고, 다양한 실시간 성수행을 확인했습니다. 그 결과, 매력적인 시스템이 탄생했습니다. 트랙킹, 디텍팅, 체인징 등에 대해 잘 수행하는 모습을 보였습니다.

Conclusions

이 논문에서는 object detection의 통합된 모델인 YOLO를 소개했습니다. 이 모델은 간단한 구조이며 전체 이미지를 직접적으로 학습합니다. classifier의 접근 방식과는 다르게 YOLO는 회귀 문제로 접근합니다.

Fast YOLO의 경우 가장 빠른 object detector를 기록하였고 YOLO의 경우 real time object detection 분야에서 sota model을 기록했습니다. YOLO의 경우 새로운 도메인에도 일반화가 가능하며 빠르고 강건한 object detection 어플리케이션에 활용 가능합니다.

Reference

<https://arxiv.org/pdf/1506.02640.pdf>

<https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89>