YOLO 논문 리뷰

YOLO 특징

1. 빠르다.
2. 이미지 전체를 이용하여 예측한다.
3. generalizable representations을 학습한다.

Real-time object detection

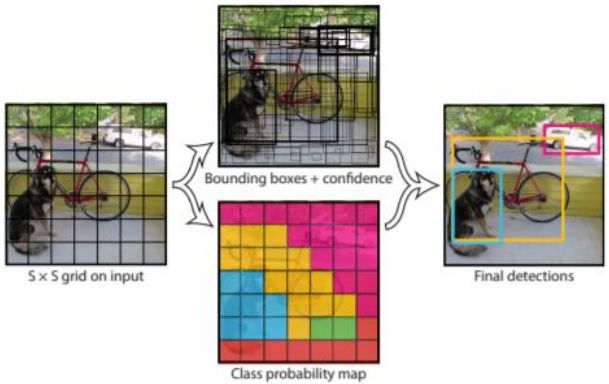
- 실생활에 적용되기 위해 real-time이 필요함
- base 45 fps
- fast 144 fps
- R-CNN은 느림

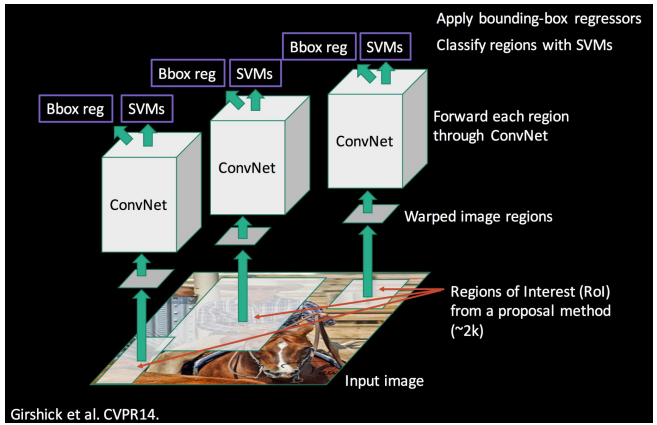
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	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.

Unified Detection

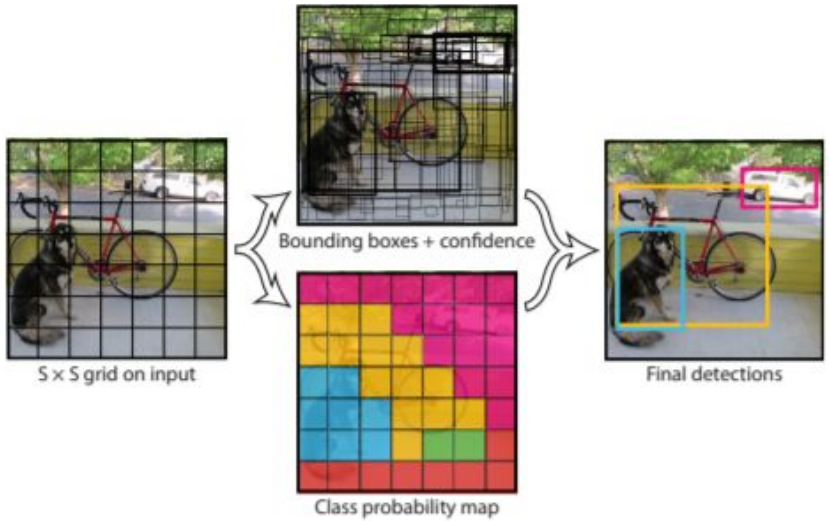
- 1 stage detection
- regression
- bounding box와 class probability를 동시에 예측
- YOLO vs R-CNN
  - yolo 역시 각 grid 에서 potential bounding bow를 제안하는 형식이지만 공간적 제약을 통해 중복된 detection을 줄임





Unified Detection

1. 입력 이미지를 S x S gird 로 나눈다.
2. 각 grid cell 마다 B개의 bounding box와 box별 confidence score를 예측한다.
3. 각 grid cell 마다 conditional class probability를 예측한다.



Unified Detection

- bounding box = (x, y, w, h)
  - x, y : 해당 grid cell 안에서의 object center의 좌표 -> 0~1 사이의 값으로 normalize
  - w, h : 전체 이미지에 대한 크기의 비율 -> 0~1 사이의 값으로 normalize

- confidence score
  - Pr(Object) * IOU
  - 모델이 bounding box 안에 object가 있다고 얼마만큼 확신하는지
    * bounding box가 얼마만큼 적절하게 fit 되어 있는지
  - no object : 0
  - exist object : IOU

Unified Detection

- conditional class probability
  - Pr(Class_i | Object)
  - grid cell이 object를 포함하고 있다고 할 때 그 object가 어떤 class인지 (조건부 확률)
  - grid cell 단위로 계산

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

yolo output 형태 : S * S * (B * 5 + C)

Network

- base : 24 conv + 2 fc
- fast : 9 conv (fewer filters) + 2 fc

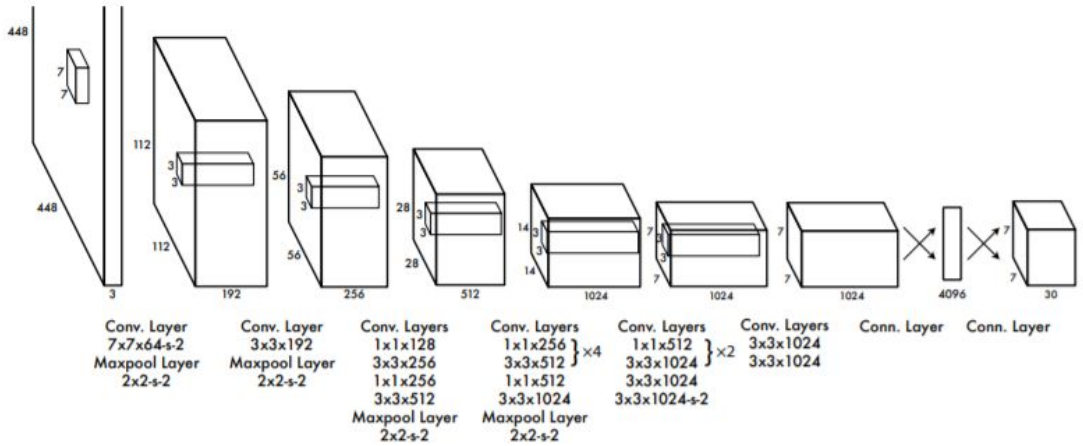


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Training Detail

1. pre-trained convolution layer
   a. ImageNet 을 이용하여 첫 20개 conv layer pre-train
   b. 학습된 20개 layer 뒤에 4개의 conv layer와 2개의 fc layer를 붙임
2. 입력 해상도 224*224 -> 448*448
   a. fine-grained 정보를 이용하기 위해서 해상도를 올림
3. bounding box의 x, y, w, h를 0~1 값으로 정규화
   a. box의 크기가 클수록 error의 영향을 줄이기 위해서 w, h는 log scale 사용
4. 마지막을 제외한 모든 layer에서 leaky relu 사용
5. Sum-squared error를 loss function으로 사용
   a. average precision(AP)을 최대화 하는 목적과 완벽하게 일치하지는 않음
   b. bounding box 위치에 관한 error에 더 큰 가중치를 둠
   c. grid cell에 object가 없는 경우가 대부분이기 때문에 그 경우에 대한 가중치를 작게 둠

Training Detail

6. grid cell 안의 bounding box 중 가장 ground truth와 IOU가 높은 것을 이용하여 loss를 계산 (“responsible”)

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

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

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

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

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

Limitation

- grid를 나누고 각 grid에서 하나의 class를 가지는 방식이므로 작은 object가 군집해 있는 경우에 성능이 떨어진다.
- 데이터로 부터 bounding box를 학습하기 때문에 새로운 형태의 box에 robust 하지 못하다.
- down sampling을 사용하는 구조이므로 비교적 coarse한 feature를 사용
- box의 크기에 상관없이 error가 동일한 크기로 적용된다. -> 박스의 위치에 대한 error가 error의 대부분을 차지한다.

Experiments

- artwork 데이터에 대해서도 YOLO는 안정적으로 좋은 성능을 보여줌
  - general한 특징을 학습한다.
- 배경에서 object를 예측하는 실수가 적다
  - 전체 이미지를 통해 학습하기 때문에 전체적인 맥락을 잘 잡아낸다고 볼 수 있다.
- R-CNN + YOLO의 성능이 아주 좋다
  - R-CNN이 배경에서 object를 예측하는 실수를 yolo가 잡아줄 수 있다.
  - But, 처리 속도는 R-CNN에 의해 결정된다. (느림)
- YOLO는 single class detector보다 general purpose detector에 알맞다

끝