Week1_예습과제_김도희

You Only Look Once

: Unified, Real-Time Object Detection

Goal

How to frame object detection as a **regression problem** to **spatially separated bounding boxes** and associated **class probabilities** in **one evaluation**.

⇒ **단일 신경망**을 사용해 이미지에서 객체를 한 번에 탐지하고 분류할 수 있도록 설계된 모델

1. Introduction

- R-CNN

  > 💡 Use region proposal methods to **first generate potnetial bounding boxes** in an image and then run **a classifier** on these proposed boxes. After classification, **post-processing** is used to refine the bounding boxes, eliminate deplicate detections, and rescore the boxes based on other objects in the scene.

  Region proposal ⇒ classification ⇒ box regression의 단계를 거쳐 inference time이 길다.

- Benefits of YOLO
  1. fast
  2. global predictions
  3. generalizable representations of object

2. Unified Detection

end-to-end training이 가능하고 꽤 높은 정확도를 가지며 빠르게 처리가 가능

S X S 개의 그리드

각각의 그리드들은 B개의 바운딩 박스를 갖고, 이 바운딩 박스마다 신뢰도가 존재한다.

신뢰도 점수는 바운딩박스가 객체를 가지고 있을 확률과 얼마나 정확하게 예측하는지를 반영한다.

$Pr(Classi|Object)$ X $Pr(Object)$ X $IoU_{truthpred}$ = $Pr(Classi)$ X $IoU_{truthpred}$

Tensor의 구조: S X S X 30 (B=2)

(1-5: 1번째 바운딩 박스에 대한 정보, 6-10: 2번째 바운딩 박스에 대한 정보, 11-30: 카테고리 예측 확률 값)

2.1 Network Design

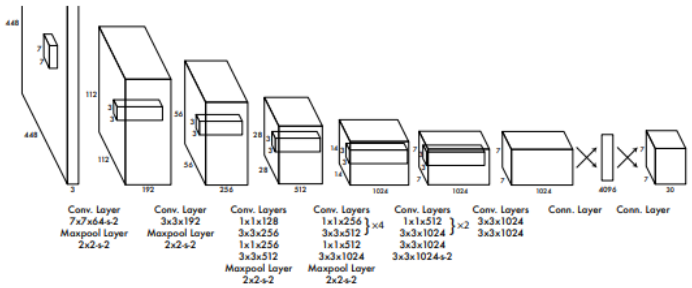

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Convolution layers(이미지 특징 추출) + Fully connected layers(클래스 확률과 바운딩 박스의 좌표를 추론)

2.2 Training

- loss function

$$\text{Total Loss} = \text{Coordinate Loss} + \text{Objectness Loss} + \text{Classification Loss}$$

  - Coordinate Loss

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

    1_{ij}^{obj}: i번 셀이 j번 바운딩 박스를 가지고 실제 객체를 포함하는 경우 1, 그렇지 않은 경우 0.

    x_i, y_i, w_i, h_i: 실제 바운딩 박스의 좌표 및 크기.

    $\hat{x_i}, \hat{y}_i, \hat{w}_i, \hat{h_i}$: 예측된 바운딩 박스의 좌표 및 크기.

    λ_{coord}: Coordinate Loss에 대한 가중치. YOLO는 좌표 손실에 큰 비중을 두므로, 이 가중치는 보통 다른 손실에 비해 크게 설정.

    sqrt를 해주는 이유: 사이즈 차이에 대한 페널티를 균일하게 만들기 위해

  - Objectioness Loss

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

    C_i: 실제 신뢰도(객체가 존재할 확률).

    $\hat{C}_i$: 예측된 신뢰도(객체가 존재할 확률, IoU).

λ_{noobj}: 객체가 없는 영역에서의 신뢰도 손실에 대한 가중치. 이 값은 보통 0.5로 설정되어 **객체가 없는 영역에 대한 페널티**를 줄이도록 함.

- Classification Loss

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

$p_i(c)$: 실제 클래스 확률.

$\hat{p}_i(c)$: 예측된 클래스 확률.

- Hyperparameter

  `BATCH_SIZE` = 64

  `MOMENTUM` = 0.9

  `DECAY` = 0.0005

- Considerations

2.3 Inference

하나의 객체를 여러 그리드 셀이 동시에 검출하는 문제 발생 (주로 객체가 여러 그리드에 걸쳐 존재하는 경우)

⇒ Non-maximal suppression 을 통해 해결

- Non-Max Suppression(NMS) Algorithm
  1. 모델이 출력한 객체 감지 결과는 여러 개의 바운딩 박스와 그 상자에 대한 신뢰도 점수로 이루어져 있습니다.
  2. 모든 경계 상자를 신뢰도 점수에 따라 내림차순으로 정렬
  3. 신뢰도 점수가 가장 높은 경계 상자를 선택하고, 이를 최종 결과에 포함
  4. IoU값을 계산하고, IoU 값이 미리 설정한 임계값을 초과하는 상자는 중복된 것으로 간주되어 제거
  5. 반복

  한계 : 겹쳐진 상자들 중 실제로 다른 객체일 수 있는 경우도 같이 제거될 수 있다. ⇒ Soft-NMS

2.4 Limitations

각 그리드마다 하나의 클래스를 가지는 B개의 바운딩 박스를 만들 수 밖에 없다. 따라서 작은 객체가 많이 존재하는 경우, 이를 탐지할 수 없는 문제가 발생한다.

데이터로부터 바운딩 박스를 예측한다는 점에서 새로운 객체에 일반화하여 적용하기가 어렵다.

4. Experiments

4.1 Comparison to Other Real-Time Systems

- **Fast YOLO**
  - 52.7% mAP
  - PASCAL VOC기준 가장 빠른 Object Detection 알고리즘
- **YOLO**
  - 63.4% mAP
  - VGG-16을 feature extractor로 사용한 경우에는 성능이 증가하나, 속도 감소
- **Fastest DPM**
  - 30.4% mAP

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.

R-CNN계열은 mAP는 높으나, 속도 측면에서 느리다.

4.2 VOC 2007 Error Analysis

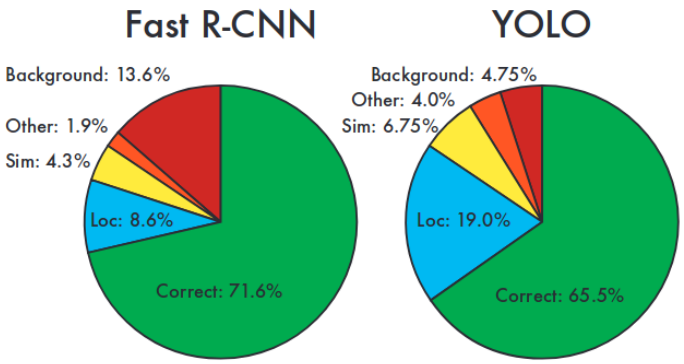

Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Correct: correct class and IoU>0.5

Localization: correct class, 0.1<IoU<0.5

Similar: class is similar, IoU>0.1

Other: class is wrong, IoU>0.1

Background: IoU<0.1 for any object

⇒ YOLO는 Localization error가 심하다.

4.3 Combining Fast R-CNN and YOLO

	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.

5. Real-Time Detection In The Wild

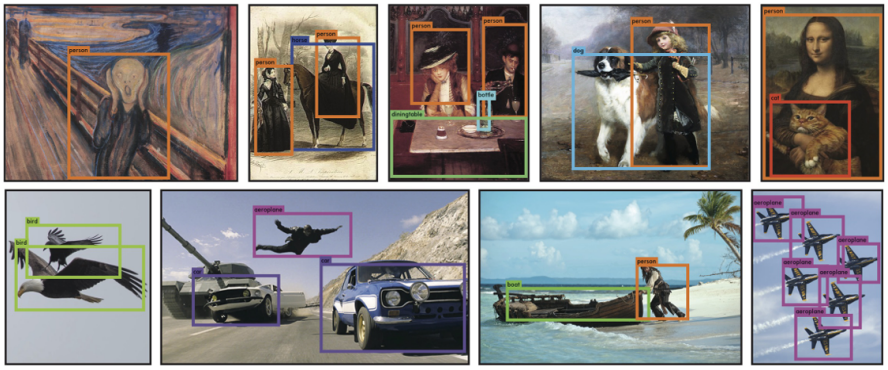

Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.