5 / 29 (월)

You Only Look Once (YOLO)

Abstarct

- Object detection을 하는데 Classification으로 사용을 하다가 Regression으로 사용함
- Full image 평가 한번으로 Bounding box와 class probabilities를 예측 할 수 있음.
- End-to-end : 한번에 처리가 가능, 많은 pipeline이 필요하지 않게 되었음.
- YOLO는 45 frames per second, Fast YOLO는 155 fps
- 최근의 기술에 비해서 YOLO는 지역적 오류는 많이 만들었지만, 배경을 object로 예측하지는 않음.
- YOLO 는 이전에 쓰이던 DPM, R-CNN의 성능을 능가했음.

Introduction

- 사람의 탐지능력은 좋다고 말로 풀어서 쓰고 있음.
- DPM(Deformable parts models) - Sliding window method : 모든 이미지 부분에 접근해서 분류.
- R-CNN - Region proposal method : 잠재적인 Bounding box를 만들어서 분류, 이후에 후처리로 Bounding box를 다듬는데 복잡한 부분의 박스를 제거하고, 다른 장면들의 물체를 기반으로 다시 점수를 매김.
- 이 복잡한 pipelines들은 최적화하기에 느리고 어렵다. 왜냐면 각각의 구성요소들을 따로따로 훈련을 시켜야 되니까 그럼. <2stage>

- YOLO에서는 물체 탐지하는데에 있어서 single regression problem으로 이미지픽셀의 좌표와 물체의 확률값을 줌. <1stage> (End-to-end)
- YOLO의 첫 번째 장점 - Object detection을 하면서 regression problem으로 바뀌면서 pipeline이 많이 필요없어졌기 때문에 굉장히 빠름.
- YOLO의 두 번째 장점 - 예측을 할 때 이미지를 전체를 본다. Slinding window, Region proposal 기술들과는 달리 전체 이미지를 보기 때문에 배경을 물체로 보는 에러율이 낮

음. (Fast R-CNN은 최고의 물체 탐지 모델이지만 이미지의 전체적인부분을 보는 것이 아니기 때문에 배경을 물체라고 탐지하는 에러율이 YOLO보다 높음.)

- YOLO의 세 번째 장점 - 물체를 일반화 할 수 있는 표현이 가능함.
- YOLO는 최신 기술들에 비해서 물체 탐지의 정확도에서는 뒤떨어지지만, 물체를 빠르게 탐지함에 있어서 좋은 성능을 가짐. 하지만 작은 지역적인 것을 탐지하는데 있어서는 어려움이 있음.

Unified Detection

- YOLO는 각각의 Bounding box를 전체의 이미지로부터 예측하기 위해 특징을 사용함.
- 이미지의 모든 class의 Bounding box를 동시에 예측함. —> 전체 이미지를 가지고 탐지
- YOLO는 높은 Average precision을 유지하면서 실시간 탐지와 End-to-end가 가능하게 만들어짐.

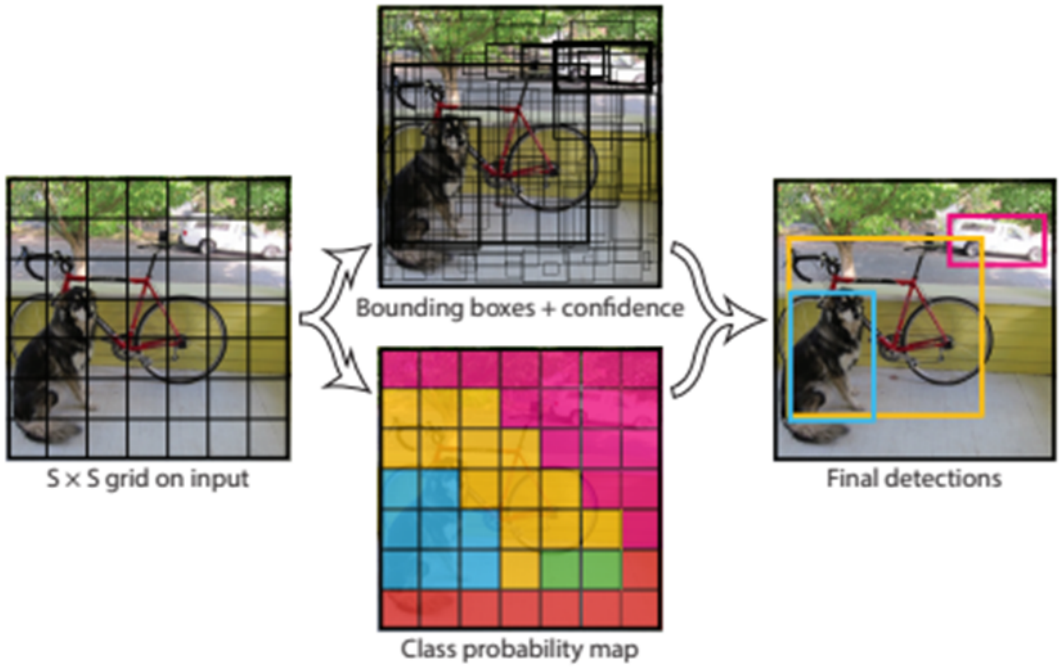
YOLO에 대한 방법을 이제 설명할거임

- S x S grid로 나눔 —> Grid cell은 물체를 탐지하기 위함.
- 각각의 Grid cell은 Bounding box와 Confidence score를 예측함.
- 이 Confidence score는 모델에 반영이 되는데, Bounding box에 물체가 포함되어 있고, Bounding box가 예측하는 정확성에 대해서도 알아봄.
- Confidence score = P_r(Object) * IOU^truth_pred
- 만약에 Bounding box안에 물체가 존재하지 않으면 Confidence score는 0임.

- Bounding box는 x, y, w, h, confidence score로 이루어짐.
- x, y는 Grid cell의 경계와 연관된 box의 중심 좌표임.
- w, h는 이미지의 width와 height임.
- Confidence score는 예측 박스와 훈련 데이터셋에 대한 실제 정답 사이의 IOU 예측을 나타냄.
- 각 Grid cell은 C : Conditional class probabilities도 예측함. Pr(Class_i | Object).
- 이 확률은 물체에 포함되는 Grid cell에 따라서 결정됨.

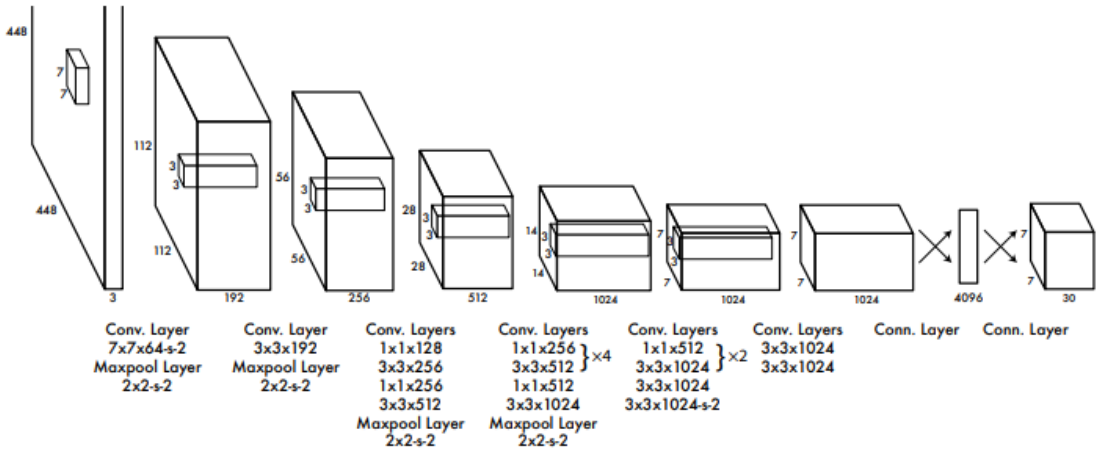
- $\Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth}$

  Class probabilities | Confidence score | Prediction



- S : Grid cell, B : Bounding box, C : Class probabilities임.
- 예측은 S x S x (B * 5 + C) tensor 값으로 나오게 됨.
- PASCAL VOC에서는 S = 7, B = 2, C = 20 —> 7 x 7 x 30 tensor로 예측을 하게 함.

Network Design



GoogLeNet의 inception module을 사용하는 것 대신에 1 x 1, 3 x 3 convolution을 사용함

Training

- YOLO는 Sum Squared Error를 사용함. 이유 - Average Precision을 높이는데는 적합하지 않지만 최적화하는데 쉬워서 사용함.
- Localization error, classification error 동일한 weight를 부여
- 모델의 불균형을 야기 할 수 있음. —> 이거를 해결하기 위한 방법이 아래 나옴

- Bounding box로 인한 loss를 증가시키고, 물체가 포함되지 않은 Bounding box에 대한 confidence 예측으로부터 loss를 감소시킴.
- lambda_coord, lambda_noobj —> lambda_coord = 5, lambda_noobj = .5
- Sum-squared error도 큰 box와 작은 box에 동등한 weights 값을 가짐.
- 큰 box에서의 오류는 작은 box에서의 오류보다 덜 중요함.
- 위의 문제를 해결하기 위해서 width와 height에 제곱근을 곱함.

Loss function

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

Limitations of YOLO

- 새 떼나, 작은 새와 같은 작은 물체들을 나타낼 때 어려움이 있음.
- 큰 box안에 있는 작은 error는 일반적으로 IOU에 큰 영향이 있지 않은 반면, 작은 box에 있는 작은 error는 IOU에 큰 영향을 끼침.