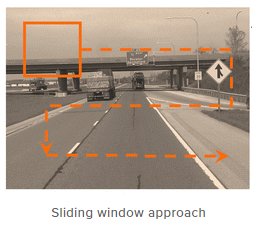
19/10/18 최초작성

참고사이트 주소는 맨 아래 참고.

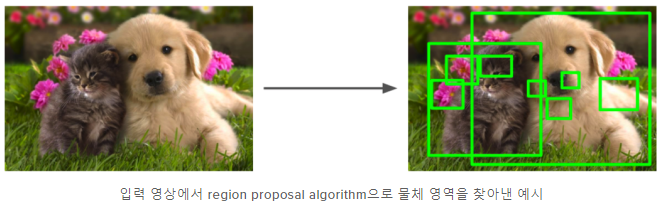
Object detecting(객체 검출)

* Sliding window - 가장 간단한 Object detection 방법



image가 있다면 사각형 틀을 이동시키며 모든 영역에 대해 검출하는 방법

* 연산이 많아 비효율적.
* Region Proposal – 물체가 있을법한 영역을 찾아주는 방법



여러 알고리즘들이 있으며 selective search, Edge boxes 등 존재.

YOLO (You Only Look Once)

* 객체 검출을 위해 고안된 DNN으로서, 테두리상자 조정(Bounding Box Coordinate)과 분류(Classification)를 동일 신경만 구조를 통해 동시에 실행하는 통합인식을 구현한 것이 특징.
* 기존 R-CNN 계열은 속도가 느림 (Region Proposal 된 수가 많음)
* YOLO는 proposal 방식에 있어 그리드(grid) 방식을 채택함.
* grid -> image를 S x S 개의 cell로 나누어 각 cell이 곧 proposal 수가 됨.

모델 정의 및 절차

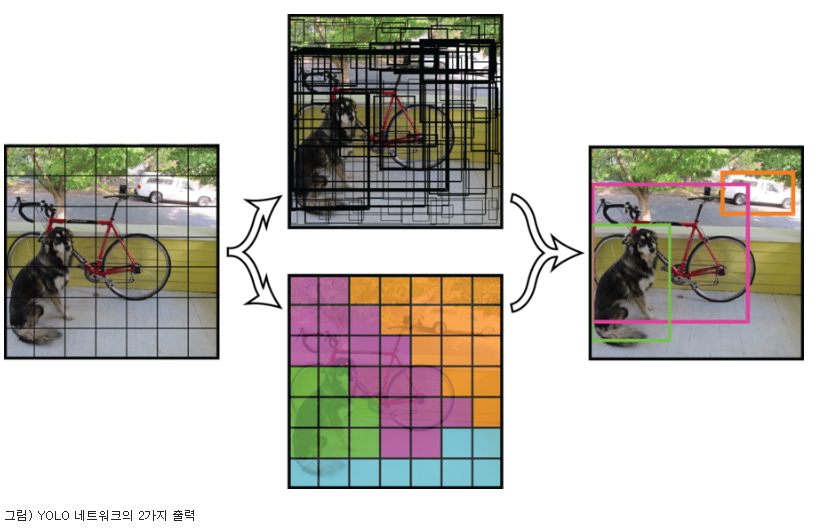
1. 이미지를 S x S 개의 그리드(grid)로 분할.
2. 이미지 전체를 신경망에 넣고 특징 추출을 통해 예측 텐서(grid 별 Bounding box 정보, confidence score, class probability 포함)생성.
   1. Confidence Score : 물체가 있을 확률 \* 예측된 Bounding Box와 실제 Bounding Box의 IOU (IOU: Intersection Over Union, IOU = 교집합 / 합집합)



* 1. Class Probability : object가 각 class 일 확률



1. grid 별 예측 정보를 바탕으로 Bounding box 조정 및 classification 수행

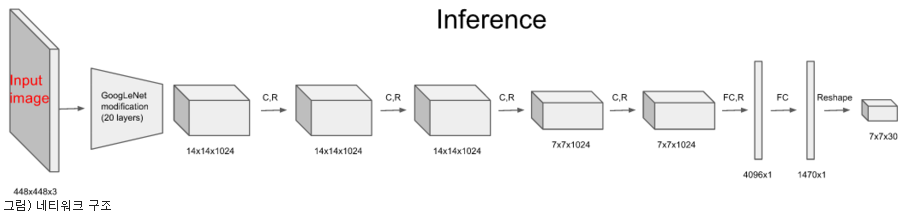


* 왼쪽 이미지가 input image 이고 7 x 7 grid(YOLOv1 case)로 나눈 것이다.
* 중간의 이미지 2개를 합한것이 YOLO network 출력의 결과물이다. 위의 이미지가 각 cell 당 예측한 bounding box이며 object가 있을 확률(confidence score)이 높은 것은 굵게 표현되었다. 각 cell당 예측하는 bounding box의 개수(anchor)는 2개이다.(YOLOv1 case)

아래의 이미지는 해당 cell에 어떤 class(개, 자전거 등)가 있을 지에 대한 정보(class probability)이다. class는 20개이다. (YOLOv1 실험 논문 case)

* 우측 이미지는 network의 결과물을 통해 생성한 것이다. (NMS)

YOLO 네트워크 구조



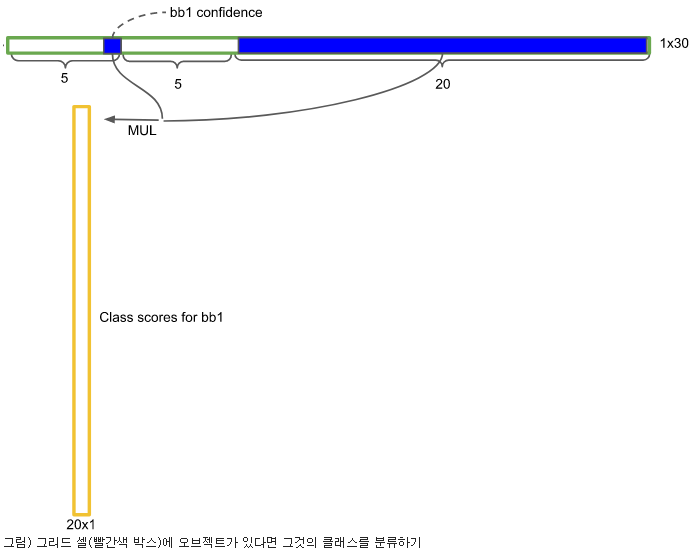
* YOLOv1 Network는 448x448x3 input image를 받아 24회의 convolution layer, 2회의 fully connected layer를 거쳐 reshape를 통해 7 x 7 x 30의 출력(예측 텐서)을 갖는다.
  + shape -> S x S x (5 x B + C), S x S : grid, B : Bounding Box 수, C : Class 개수
* 예측텐서의 값을 바탕으로 값을 조정해서 학습 -> 비용함수를 사용. (뒤에 내용 나옴)
* 각 cell에 30개 채널에 정보는 2개의 Bounding Box에 대한 정보와 각 class에 대한 정보가 있다.
  + 각 cell의 정보 – 1. Bbox에 object가 있을 확률(confidence)

2~3. Bbox의 중심점 좌표(해당 cell의 중심에 대해 0~1값을 가짐)

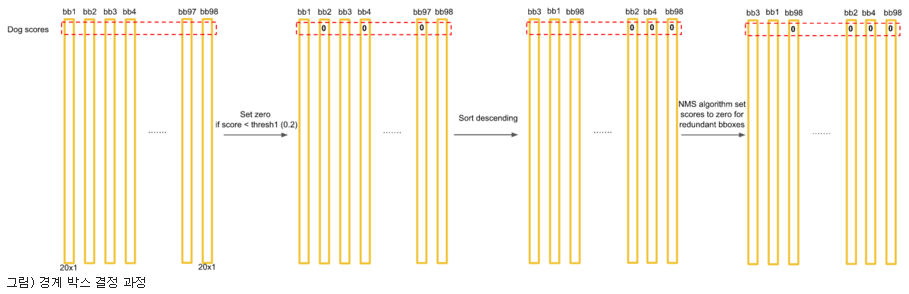
4~5. Bbox의 가로 세로값(전체 image 크기에 대해 0~1값을 가짐)

11~30. 각 class의 probability

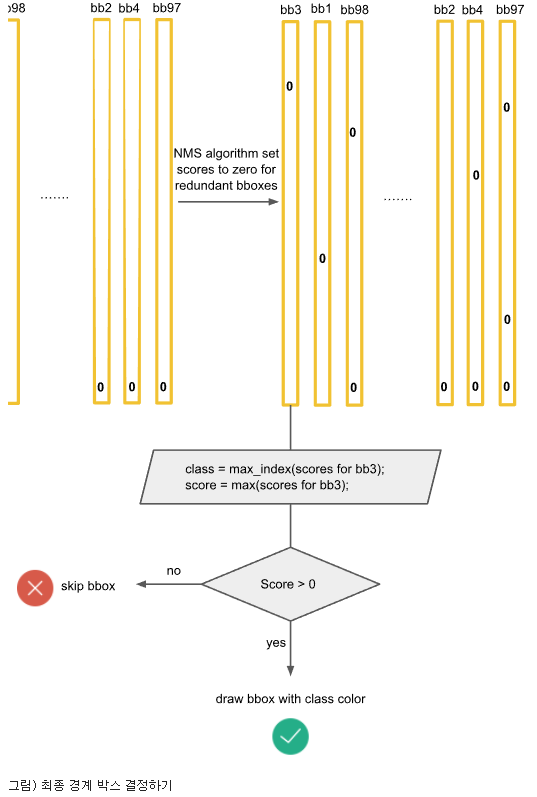
* 학습을 통해 적절한 Bounding box들을 찾았다고 하면



* 각 grid cell의 채널에는 2개의 예측된 Bbox가 있고 그 Bbox에 object가 존재할 confidence와 각 class의 probability를 곱하면 2x7x7= 98개의 Bbox에 대한 class별 확률이 나오게 된다.



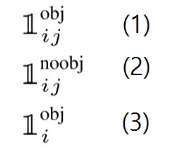
* 첫 번째 class 가 dog class 이고 그 경우를 보면 우선, score 가 threshold(0.2) 이하인 것은 해당 class는 절대 없다고 판단하고 0으로 만들어 준다.
* 그리고 높은 값 순으로 정렬(sort)한다.
* 위 과정까지 하면 한 object에 대해 여러 경계박스가 나타날 수 있다. 중복을 제거하기 위해 NMS(Non-Maximal Suppression, 비-최대값 억제) 알고리즘 사용하여 중복을 제거한다.
* NMS 알고리즘은 Bbox가 일정치(threshold, 0.5) 이상 겹쳐있을 때 최대값을 갖는 하나만 남기는 것.
* NMS 에서는 0.5 이상 겹치지 않으면 또 다른 object가 있을 수 있다고 판단하고 삭제하지 않는다.
* 모든 class에 대해 위 과정을 수행한다.



* 각 Bbox에 대해 class 및 score를 정의한다. class는 가장 높은 score의 index이며 score는 가장 높은 score값이 된다. 그리고 score값이 0이라면 Bbox를 그리지 않는다.
* 그리고 결과값(score)값이 threshold값 이하(0.5) 이면 지워짐.
* 위 과정을 모든 Bbox에 대해 수행한다.
* 최종적으로 object가 있을거다 라고 추측한 Bbox만이 그려진다.

YOLO 모델 Training

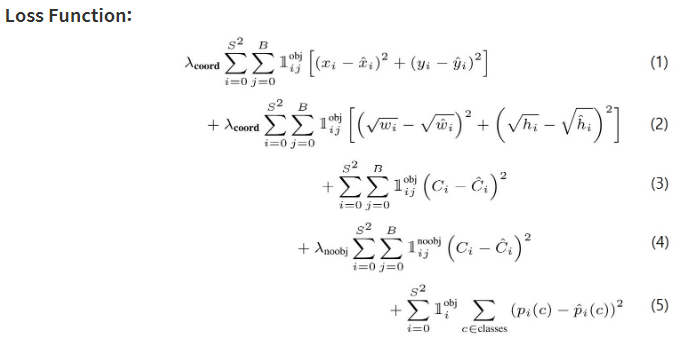
* 전제조건
  + Grid cell의 여러 Bounding box(여기선 2개) 중 ground-truth box(실제 정답)와 IOU가 가장 높은 Bounding box를 predictor(예측기)로 설정한다.
  + 위 기준에 따라 아래 기호 사용.



1. Object가 있는 grid cell i의 predictor Bounding box j

(여러 Bounding box 중 실제 정답과 IOU가 가장 큰 Bounding box)

1. Object가 없는 grid cell i의 Bounding box j
2. Object가 있는 grid cell i



Multi Loss(cost) = Coordinate Loss(1 + 2) + Confidence-Score Loss(3) + No-Object Penalties(4) + Classification Loss(5)

1. Object가 있는 predictor Bounding box에 대한 x 와 y loss 계산 (Bounding box 중심점)
2. Object가 있는 predictor Bounding box에 대한 w 와 h loss 계산 (Bounding box 크기)
3. Object가 있는 predictor Bounding box에 대한 confidence score의 loss 계산 (C\_i = 1)
4. Object가 없는 predictor Bounding box에 대한 confidence score의 loss 계산 (C\_i = 0)
5. Object가 있는 grid cell에 대해 class probability의 loss 계산(해당 class = 1 아니면 0)

 : coordinate(Bounding box의 x y w h)에 대한 loss 값과 다른 loss 균형을 위한 값(5)

 : object가 있는 box와 없는 box간 loss 균형을 위한 값(0.5)

한계점

* 각 gird cell이 하나의 class만 예측할 수 있으므로 작은 object 여러 개가 붙어있으면 예측하지 못함
* Bounding box 형태가 training data를 통해 학습되므로 새로운 형태의 bounding box는 예측하지 못함
* 몇 단계의 layer를 거쳐 나온 feature map을 대상으로 bounding box를 예측하므로 위치가 다소 부정확해지는 경우가 있음.

참고 사이트

[1] <http://blog.naver.com/PostView.nhn?blogId=sogangori&logNo=220993971883>

[2] <https://curt-park.github.io/2017-03-26/yolo/>

[3] <https://pjreddie.com/publications/>