**객체 검출 (Object Detection)**


객체 검출이란 입력 영상에서 사용자가 원하는 객체를 찾고 객체의 위치와 바운딩 박스를 구하는 것을 말한다. 객체 검출은 크게 (템플릿이 있을 때의 검출, 템플릿이 없을 때의 검출, 학습 데이터가 있을 때의 검출)로 나뉘게 된다.

1번 같은 경우, (b)라는 템플릿에 해당하는 객체를 찾는 것을 목표로 한다. 따라서 입력 영상 (a)에 템플릿을 Sliding-Window하면서 특정 객체의 위치를 알아낸다. 이후 해당 위치에 바운딩 박스를 그리는 작업을 한다.

2번 같은 경우, 대표적으로 직선 검출을 이용한 차선 검출이 있다. 직선을 검출할 때는 Hough Transform 따위를 이용하면 된다.

3번이 바로 우리가 하려는 CNN 기반 객체 검출이다. 라벨링을 한 사람이면 알겠지만 영상의 객체에 대해 바운딩 박스를 그리는 것과 더불어 해당 객체가 어떤 클래스인지도 남기게 된다.

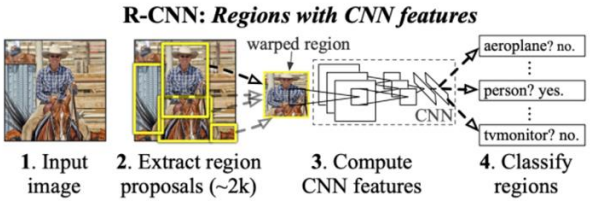
즉 이러한 라벨링 파일, 학습 데이터를 이용하고 학습이 잘 되면 바운딩 박스와 객체 위치 뿐 만 아니라 박스 안 객체의 클래스까지 예측이 가능하다는 사실이다.

지금부터 CNN 방식으로 객체 검출을 어떻게 하는지 알아보겠다.

## RCNN

- R-CNN (Region Convolution Neural Network)

1. Selective search를 이용해서 proposal box들을 찾는다.
2. box들을 CNN에 넣어서 feature를 파악한다.
3. Feature들을 SVM을 통해 분류한다.
4. 분류된 box와 ground truth를 비교하여 가중치를 업데이트한다.
5. 분류가 된 box들의 점수를 보고 특정 점수 이상의 값을 가진 box만 표시한다.
6. Box regressor을 학습해서 box의 위치를 정확히 물체 위로 조정한다.






가장 기본적인 네트워크는 R-CNN이다. R-CNN은 크게 6가지 단계로 이루어지는데 각 단계에 대해 차근차근 알아 보겠다.

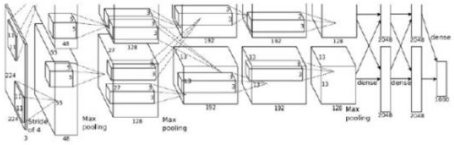
## RCNN

- R-CNN (Region Convolution Neural Network)

1. Selective search를 이용해서 proposal box들을 찾는다.

Segmentation을 한 다음 object가 있을만한 모든 영역에 대해 proposal box를 찾는다.
이렇게 제안된 proposal box 중 2000개만 사용한다.

2. box들을 CNN에 넣어서 feature를 파악한다.



영상에서 box에 해당하는 부분을 모두 자르고 잘라낸 영상들을 AlexNet에 넣는다. (resize to 227*227*3)



먼저 Selective Search라는 작업을 통해 proposal box를 찾는다. selective search란 입력 영상을 segmentation하고 segmentation한 결과를 기준으로 객체가 있을만한 영역을 찾는 것이다. segmentation은 영상을 픽셀 단위로 분류하는 것을 말한다. 같은 분류에 해당하는 픽셀들은 같은 색을 지니게 된다. 처음 segementation을 하면 위 그림보다 훨씬 더 많이 분류된 형태가 된다. 하지만 이렇게 하면 proposal box가 대책없이 많아지기 때문에 작은 영역을 큰 영역으로 통합하는 과정도 필요하다. 이 과정을 거치고 나온 박스 중 2000개를 사용하겠다는 의미이다.

(작은 영역을 큰 영역으로 통합)

2000개의 Proposal Box들은 AlexNet을 통해 각각의 특징이 파악된다. (Image -> Feature Vector)

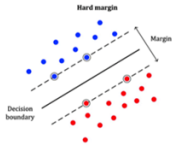
여기서 CNN이 쓰이는데, CNN을 사용하는 이유는 영상에 대해 올바른 Feature를 파악하기 위함이다. 일반 DNN과는 달리 형상 데이터가 유지되고 Receptive Field를 알고 있는 점을 올바른 Feature파악에 도움이 된다.

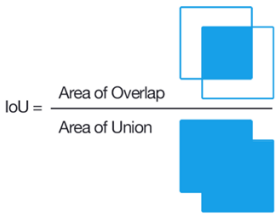
AlexNet에 입력으로 주기 위해 (227x227x3)으로 변환하는 과정이 필요하다.

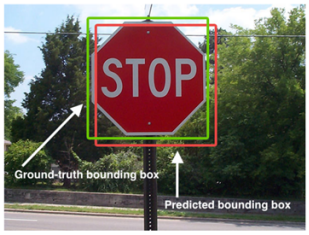
RCNN

- R-CNN (Region Convolution Neural Network)

3. Feature들을 SVM을 통해 분류한다.




2000개의 box를 사용하였으므로 Feature도 2000개가 생성된다.
이 Feature들을 SVM으로 분류한다. (class별로 SVM 분류)

4. 분류된 값(box)과 ground truth를 비교하여 가중치를 업데이트한다.

5. 분류가 된 box들의 점수를 보고 특정 점수 이상의 값을 가진 box만 표시한다.

**(Ground-truth:** 학습하고자 하는 데이터의 원본 혹은 실제 값)

가톨릭대학교 6

2000개의 Feature가 생성되면 이 Feature들은 SVM의 입력으로 들어가서 이진 분류를 한다. 이진 분류는 (객체를 포함하는가, 아닌가)로 나뉘며 class별로 SVM이 존재한다.

하지만 처음부터 잘 분류하지는 않을 것이다. 따라서 분류된 값(proposal box feature)과 ground

truth(정답 박스)를 비교해서 가중치를 업데이트하는 과정이 필요하다.

이 때 비교는 IoU(Interaction over Union)을 이용한다. 표지판 사진을 예시로 들 수 있다. 초록 색 영역이 사람이 그은 정답 박스이고 빨간 색 영역이 컴퓨터가 제안한 박스라고 하자. 이 두 박스의 합집합을 분모에 넣고 교집합을 분자에 넣는다. 이 분수의 값이 클수록 많이 두 박스는 많이 겹치는 것을 의미할 것이고 학습이 잘 됐다는 것을 의미한다. (올바르게 박스를 잘 제안함) 반대로 분수의 값이 작을수록 학습이 잘 안 됐다는 것을 의미한다.

SVM 결과에서 객체가 있다고 분류한 박스가 IoU가 0.2나 0.3이면 옳다고 볼 수 없기 때문에 학습을 반복해서 올바른 결정 경계를 갖도록 하는 것이다.

이렇게 학습이 완료되면 분류된 box들의 점수(IoU)를 보고 특정 점수 이상의 값을 가진 box만 남긴다.

RCNN

- R-CNN (Region Convolution Neural Network)

6. Box regressor을 학습해서 box의 위치를 정확히 물체 위로 조정한다.

$P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$

proposal box (중심점x,y,너비,높이)

$G = (G_x, G_y, G_w, G_h).$

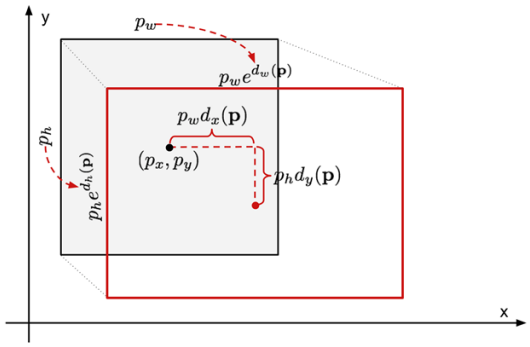
ground truth box (중심점x,y,너비,높이)

$d_x(P),\ d_y(P),\ d_w(P),\ \text{and}\ d_h(P).$

필요한 이동량 예측치를 구하는 함수 <얻고자 하는 함수>

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

p를 이동시키는 함수의 식(이미지 크기와 비례해서 조정)






특정 점수 이상인 box 는 올바른 클래스 정보를 지니고 있을 것이다. 하지만 Selective Search 로 제안된 박스이기 때문에 정답 박스와는 차이가 있을 수 밖에 없다.

따라서 Box Regressor 라는 것을 학습해서 box 의 위치를 정확히 물체 위로 조정하는 작업이 필요하다.

Regressor 즉 회귀 문제를 풀어서 박스를 조정할 것이라는 의미인데 어떻게 이것이 가능한지 설명을 하겠다.

P 는 proposal box 의 정보를 나타낸다. 즉 중심점 x,y 와 box 의 가로,세로를 나타낸다.

G 는 groud truth box 의 정보를 나타낸다. 마찬가지로 중심점 x,y 와 box 의 가로,세로를 나타낸다.

우리가 원하는 것은 P 가 G 가 되도록, 혹은 G 와 거의 유사하게 만들려고 하는 것이다.

이것이 가능하려면 D 라는 함수를 알아야 된다. D 는 P 가 G 가 되기 위해 필요한 이동량을 예측하는 함수라고 이해하면 된다.

예를 들어 Px 를 D 함수에 넣으면 Gx 까지 이동하는데 필요한 이동량(Dx(p))을 예측한다. 이동량만 제대로 알면 Px+Dx(p)를 통해 Gx 의 근사치를 구할 수 있기 때문이다.

참고로 그림 설명에서는 Pw*Dx(p)를 한 것을 더하는 것으로 보여주는데 이것은 이미지 크기와 비례하게 조정을 할 필요가 있기 때문이라고 이해하면 된다.

따라서 우리는 이동량 예측치 함수 D 를 학습하면 된다.

P.s. 다른 설명에서는 D 에 대한 설명을 'P 를 G 로 이동시키는 매핑 함수'라고 하는데, 매핑하는데 중요한 역할을 하는 함수라고 해서 매핑 함수라고 말하는 것 같다.

## RCNN

- R-CNN (Region Convolution Neural Network)

6. Box regressor을 학습해서 box의 위치를 정확히 물체 위로 조정한다.

$P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$

proposal box (중심점x,y,너비,높이)

$G = (G_x, G_y, G_w, G_h).$

ground truth box (중심점x,y,너비,높이)

$d_x(P),\ d_y(P),\ d_w(P),\ \text{and}\ d_h(P).$

필요한 이동량 예측치를 구하는 함수 <얻고자 하는 함수>

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

p를 이동시키는 함수의 식(이미지 크기와 비례해서 조정)

$d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$

d함수: 학습 가능한 weight * pool5 layer에서 얻어낸 Feature Vector

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t^i_\star - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2$$

weight를 얻기 위한 loss function <선형 회귀 학습>

ti (실제 필요한 이동량) – di (이동량 예측치) 를 기준으로 모델 업데이트

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

P를 G로 이동시키기 위해 필요한 이동량

가톨릭대학교 8

다행히 우리는 P 와 G 정보를 알고 있다. 즉 제안한 박스와 정답 박스 간의 차이(이동량)를 알 수 있는 것이다. 그 이동량은 t 식을 보면 된다.

앞 설명과 연결을 짓자면 Pwdx(P)+Px 를 해서 G 에 가까운 근사치(G^x)가 나온다면, Pwtx+Px 를 하면 Gx 가 나온다는 의미이다.

D함수는 학습 가능한 Weight와 CNN 풀링 레이어에서 얻어낸 Feature을 곱으로 이루어진다.

이 때 Weight는 Loss Function(SSE+L2 norm)을 최저로 만드는 w를 말한다.

SSE를 보면 Ti(실제 이동량)-Di(예측 이동량)을 기준으로 모델을 업데이트하는 것을 확인할 수 있다.

정리하면 [ P->(D함수 모델)->D(p) ]라는 것인데, D(p)와 T의 차이를 기준으로 모델을 업데이트하면서 D모델을 발전시키겠다는 의미이다. D모델이 발전하면 D(p)는 T와 유사한 값을 지니게 된다.

결국 실수 P가 입력으로 주어지면 T와 유사한 값 D(p)를 반환하는 실수 예측 문제, Regression문제에 불과하다는 의미이다.

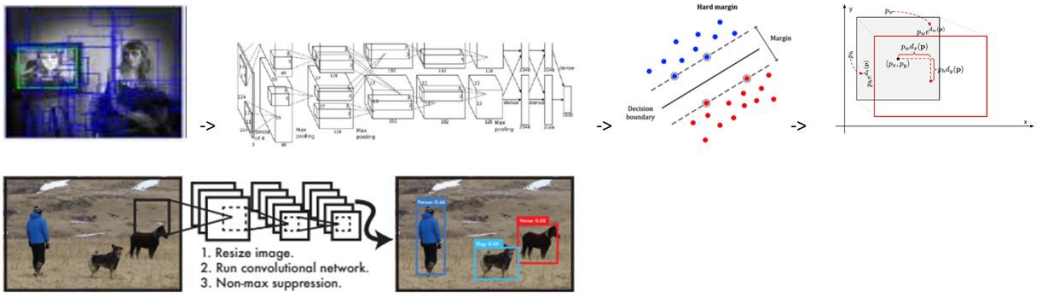
학습이 잘 돼서 T와 유사한 값 D(p)를 반환하면 p를 이동시키는 수식을 통해 Box를 옮기면 된다.

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$ (p를 이동시키는 수식)

YOLO

- YOLO (You Only Look Once)

1. 딱 한 번만 본다
2. 통합한다 (One-stage-method)
3. 빠르다




가톨릭대학교 9

RCNN을 정리하면 이렇다.

(2000개의 ROI 생성 -> CNN 특징 추출 -> SVM 분류 -> Box Regressor로 박스 조정)

하지만 지금부터 설명할 YOLO는 이 과정이 아래처럼 정리된다.

(이미지 -> CNN을 통한 분류 및 박스 조정)

이 과정을 비교해보면 YOLO는 다음과 같은 특징을 지닌다.

1. 딱 한 번만 본다. (2000개의 영상을 보지 않고 1개의 원본 영상만 본다.)

2. 통합한다. (객체의 위치 및 바운딩 박스 찾기와 클래스 분류를 한 번에 한다.)

3. 빠르다. (1, 2과정을 통해 어떤 객체 검출 프로세스보다 빠르다.)

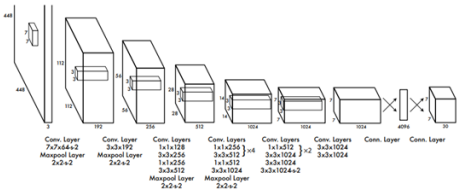
## YOLO

- YOLO (You Only Look Once)

1. 이미지를 SxS 그리드로 분할한다. (논문에서 S=7)
2. 이미지 전체를 CNN에 넣고 특징 추출을 통해 Prediction Tensor (SxSx(B*5+C))생성
3. 바운딩 박스 및 클래스를 조정한다.

T. 인퍼런스 과정에서는 Non Maximum Suppression을 진행하고 결과를 보인다.


S × S grid on input




Final detections



YOLO는 크게 세 단계로 나타낼 수 있다. 인퍼런스 과정에서는 NMS 알고리즘이 추가로 진행된다.

각 단계에 대해 차근차근 알아 보겠다.

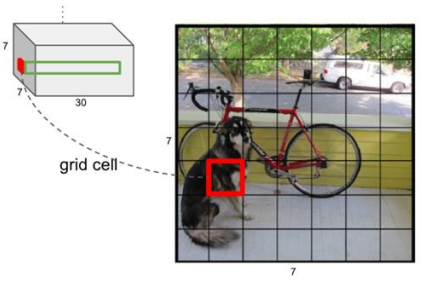
## YOLO

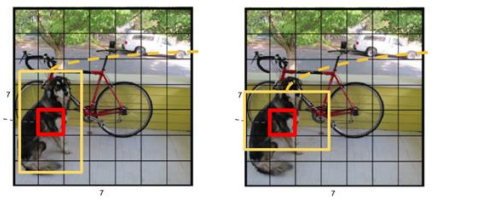
- YOLO (You Only Look Once)

1. 이미지를 SxS 그리드로 분할한다.
2. 이미지 전체를 CNN에 넣고 특징 추출을 통해 Prediction Tensor (SxSx(B*5+C))생성

S: 그리드 숫자, B: 그리드 별로 갖는 바운딩 박스의 개수, 5: 바운딩 박스의 정보(x,y,w,h,c_s), C: 클래스 개수만큼의 조건부 확률

Ex) S:7, B:2, C:20 -> (7x7x(2x5+20)) -> (7x7x30)





그리드 안에 바운딩 박스의 중앙점(x,y)이 위치해있음



YOLO는 영상을 SxS 그리드로 먼저 분할한다.

이후 이미지 전체를 CNN에 넣고 특징 추출을 통해 Prediction Tensor를 생성한다.

Prediction Tensor는 (B*5+C)개만큼의 정보가 셀 개수(S^2)만큼 있다.

이 때 B는 그리드 별로 갖는 바운딩 박스의 개수를 의미한다. 논문에서는 B를 2라고 설정했는데

그림 예시를 보면 두 바운딩 박스의 중앙점(x,y)이 한 그리드 안에 위치해있는 것을 확인할 수 있다.

5는 바운딩 박스의 정보를 나타낸다. 바운딩 박스의 정보는 중점x,y, width, height, confidence score를 의미한다.

c는 클래스 개수만큼의 조건부 확률을 의미한다. 논문에서 클래스 개수는 20으로 선정했다.

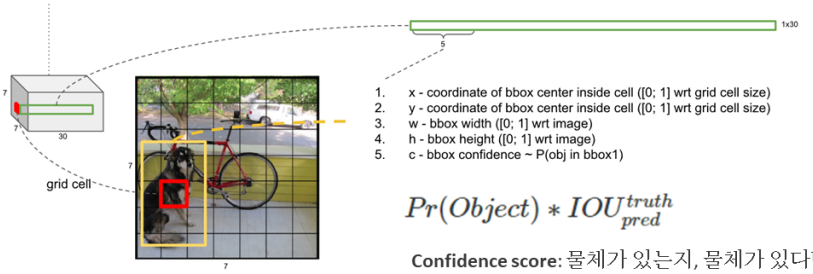
즉 한 그리드 당 2*5+20=30개의 (바운딩박스,클래스) 정보를 지닌다고 보면 된다.

## YOLO

- YOLO (You Only Look Once)

  S: 그리드 숫자, B: 그리드 별로 갖는 바운딩 박스의 개수, 5: 바운딩 박스의 정보(x,y,w,h,c_s), C: 클래스 개수만큼의 조건부 확률

  Ex) S:7, B:2, C:20 -> (7x7x(2x5+20)) -> (7x7x30)




1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

$$Pr(Object) * IOU_{pred}^{truth}$$

**Confidence score**: 물체가 있는지, 물체가 있다면 바운딩 박스에 얼마나 포함되어 있는지의 신뢰도

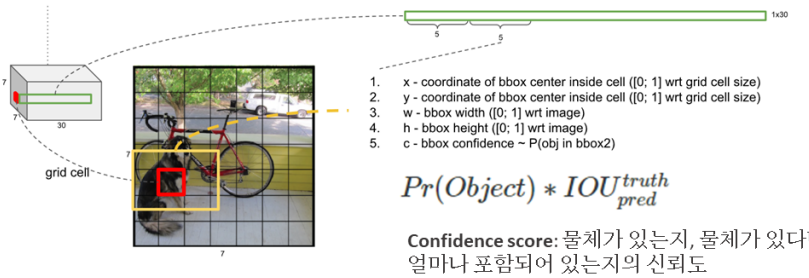


## YOLO

- YOLO (You Only Look Once)

  S: 그리드 숫자, B: 그리드 별로 갖는 바운딩 박스의 개수, 5: 바운딩 박스의 정보(x,y,w,h,c_s), C: 클래스 개수만큼의 조건부 확률

  Ex) S:7, B:2, C:20 -> (7x7x(2x5+20)) -> (7x7x30)




1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

$$Pr(Object) * IOU_{pred}^{truth}$$

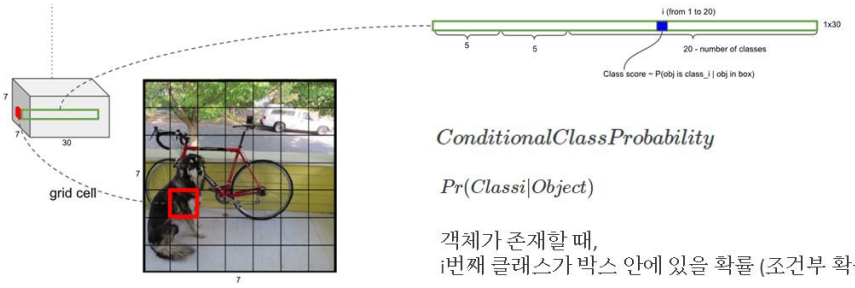
**Confidence score**: 물체가 있는지, 물체가 있다면 바운딩 박스에 얼마나 포함되어 있는지의 신뢰도

## YOLO

- YOLO (You Only Look Once)

  S: 그리드 숫자, B: 그리드 별로 갖는 바운딩 박스의 개수, 5: 바운딩 박스의 정보(x,y,w,h,c_s), C: 클래스 개수만큼의 조건부 확률

  Ex) S:7, B:2, C:20 -> (7x7x(2x5+20)) -> (7x7x30)




$ConditionalClassProbability$

$Pr(Classi|Object)$

객체가 존재할 때,
i번째 클래스가 박스 안에 있을 확률 (조건부 확률)



한 그리드(셀)에 대해 30개의 정보가 무엇인지 나타내는 그림들이다. (B:2, C:20)

처음 5개는 첫 번째 바운딩 박스의 정보이다.

이 때 다섯번째로 Confidence Score가 저장되는데, 이것은 바운딩 박스에 물체가 있는지, 물체가 있다면 바운딩 박스에 얼마나 포함되어 있는지를 나타내는 신뢰도이다.

그 다음 5개는 두 번째 바운딩 박스의 정보이다.

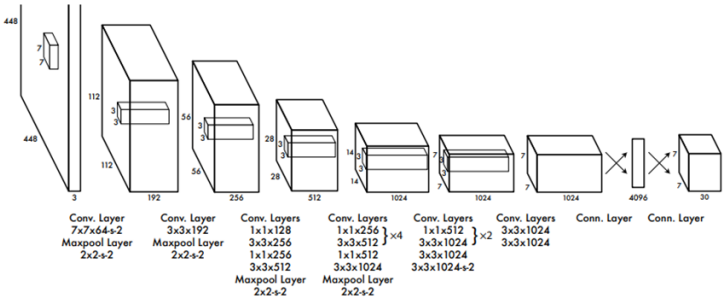
그 다음 20개는 각 클래스의 조건부 확률 정보이다.

Pr(Classi|Object) (i=1~20)가 저장이 되는데 이것의 의미는

객체가 존재할 때 i번 클래스가 박스에 있을 확률을 나타낸다.

## YOLO

- YOLO (You Only Look Once)




GooLeNet for Image Classification 모델을 기반으로 함 (24 Convolutional Layers + 2 Fully Connected Layers)
1x1 Convolution Layer를 번갈아 사용하면서 특징 공간이 줄어드는 것이 특징



이번에는 YOLO에서 사용하는 CNN 구조를 알아 보겠다.

YOLO는 GooLeNet을 기본 네트워크로 사용한다. 즉 학습한 GooLeNet을 파인튜닝해서 사용하겠다는 의미이다. GoogLeNet의 inception을 가져와 1x1 conv를 활용하여 연산량을 줄이려고 하는 것을 볼 수 있다.

사실 GoogLeNet과 완전히 동일하지는 않고 살짝 변형된 네트워크인데, 24개의 Convolution Layer와 2개의 Fully Connected Layer 구조를 지녔다는 점이 특징이다.

## YOLO

- YOLO (You Only Look Once)

loss function:


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

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

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

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

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

$\lambda_{\text{coord}}$ $\lambda_{\text{noobj}}$ : 바운딩 박스의 변수에 대한 값을 더 반영, 객체가 없는 영역에 대한 값을 덜 반영하기 위한 하이퍼 파라미터 (객체가 없는 영역이 훨씬 많기 때문)

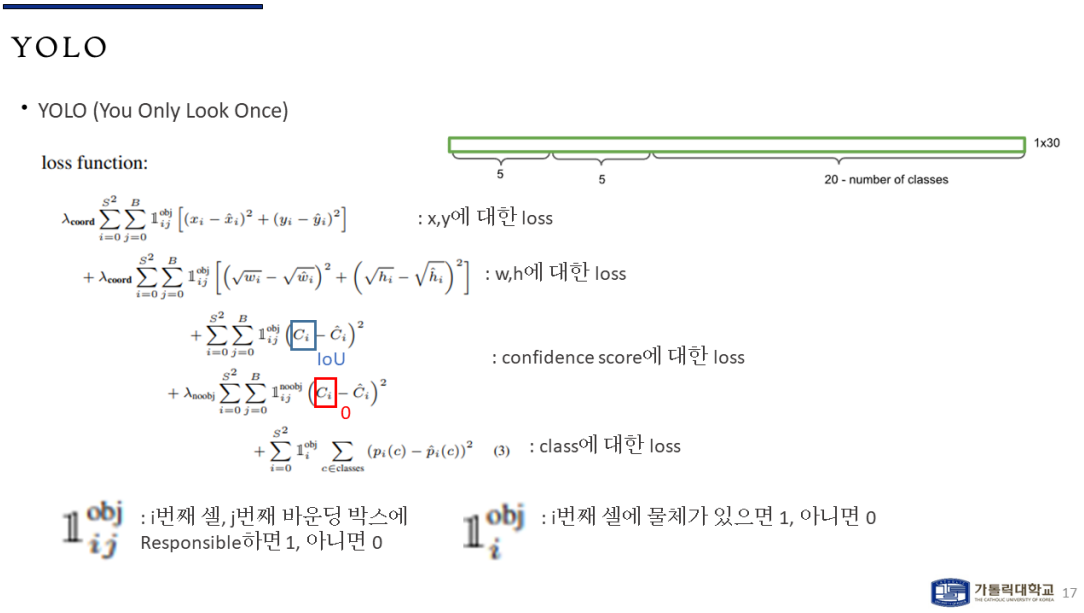
YOLO의 loss function은 5개의 loss가 합쳐진 구조라고 볼 수 있다. 첫 번째는 바운딩 박스의 중점(x,y)을 얼마나 잘 학습했는지 평가하는 loss, 두 번째는 w,h에 대한 loss, 세 번째와 네 번째는 confidence score에 대한 loss인데 객체가 있는 케이스와 객체가 없는 케이스를 나눠서 평가한 것이다. 마지막 class에 대한 loss이다.

Loss는 많이 쓰이는 SSE(오차 제곱합)를 채택해서 얼마나 적게 틀렸는지를 평가하게 된다. 참고로 w(너비)와 h(높이)는 root가 씌워 졌는데 이것은 큰 box의 오차가 작은 box의 오차보다 큰 가중치를 받지 않게 하기 위함이다. (root없으면 큰 box의 오차가 훨씬 커서 큰 box기준으로 학습이 더 많이 됨)

**주의 깊게 바라볼 점**은 **Lamda coord와 Lambda noobj**이다. 이것은 사용자가 직접 설정하는 하이퍼 파라미터인데, 바운딩 박스의 변수 값을 더 반영하고 객체가 없는 바운딩 박스의 변수 값은 덜 반영하는 것을 권장하고 있다. 따라서 논문에서 전자는 5, 후자는 0.5로 지정한다.

이렇게 하는 이유는 객체가 없는 바운딩 박스가 훨씬 많기 때문에 객체가 없는 바운딩 박스에 대해서만 많이 학습할 가능성이 커지기 때문이다.

즉 소수에 해당하는 객체가 있는 바운딩 박스는 잘 학습이 되지 않아서 객체 검출 성능이 현저히 낮게 된다. 이러한 데이터 불균형 문제를 하이퍼 파라미터로 해결할 필요성이 있다.




**두 번째로 주의 깊게 바라볼 점**은 **1 함수들**이다. 1함수들은 (1 obj ij)와 (1 obj i)로 나눠지게 된다. i는 셀에 대한 인덱스, j는 바운딩 박스에 대한 인덱스에 해당한다.

두 함수로 나누는 이유는 바운딩 박스를 고려할 필요가 없는 loss도 존재하기 때문이다. 바로 class에 대한 loss이다. 이 loss는 각 셀(그리드)에 대한 Pr(c)에만 관심이 있다.

1 함수는 기본적으로 바운딩 박스에 객체가 있으면 1, 없으면 0을 주면서 객체가 있는 경우에만 학습이 되도록 설정하는 역할을 한다.

물론 (1 noobj ij)같은 경우 객체가 없으면 1, 있으면 0이 된다.

다만 여기서 responsible하다는 의미를 알아야 한다. 여기서 responsible하다는 것은 i번째 셀이 객체가 있는 것과 더불어 j번째 바운딩 박스가 가장 IoU가 높았을 때 responsible하다고 표현한다. 즉 B가 2이기 때문에 두 바운딩 박스 중 한 박스만 responsible하다는 의미이다. 결론적으로 학습을 할 때는 한 박스만을 이용해서 학습을 하겠다고 생각하면 된다.

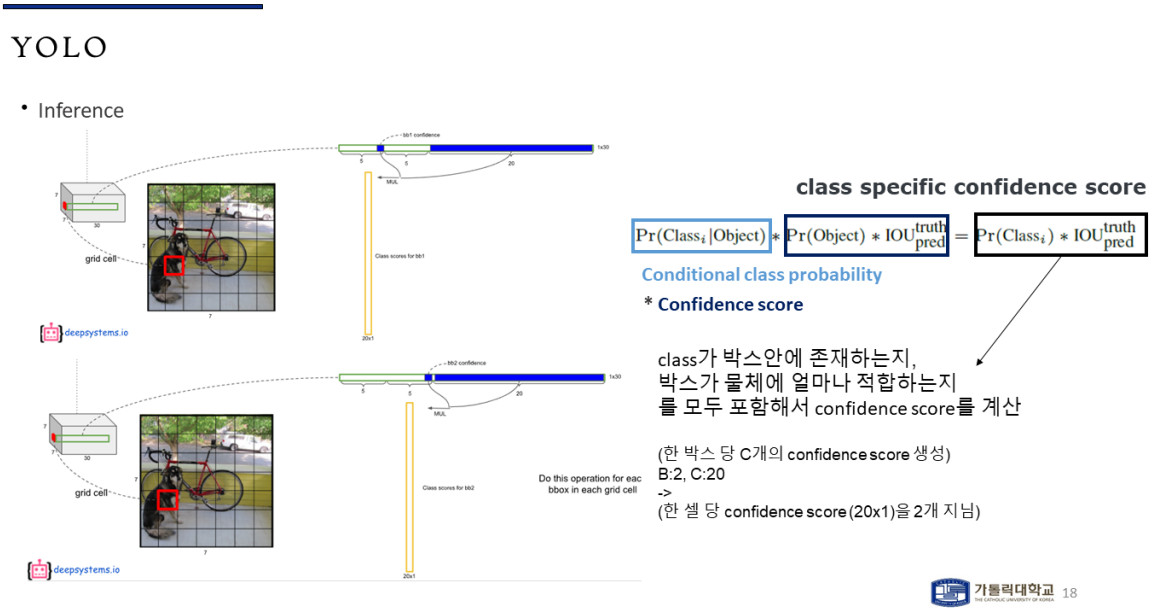
**세 번째로 주의 깊게 바라볼 점**은 **confidence score의 정답**이다.

객체가 있을 때의 confidence score는 IoU랑 같아야 할 것이다.

객체가 없을 때의 confidence score는 0이랑 같아야 할 것이다.

왜냐하면 confidence score= Pr(Object)*IoU인데 학습이 잘 됐다면 객체가 있을 때의 Pr(Object)는 1이 돼야 하고 객체가 없을 때의 Pr(Object)는 0이 돼야 하기 때문이다.

따라서 Pr(Object)가 1혹은 0이 되도록 학습이 진행되어야만 한다.




이번에는 YOLO의 인퍼런스 과정에 대해 살펴보겠다.

인퍼런스를 할 때는 새로운 confidence score가 제안된다.

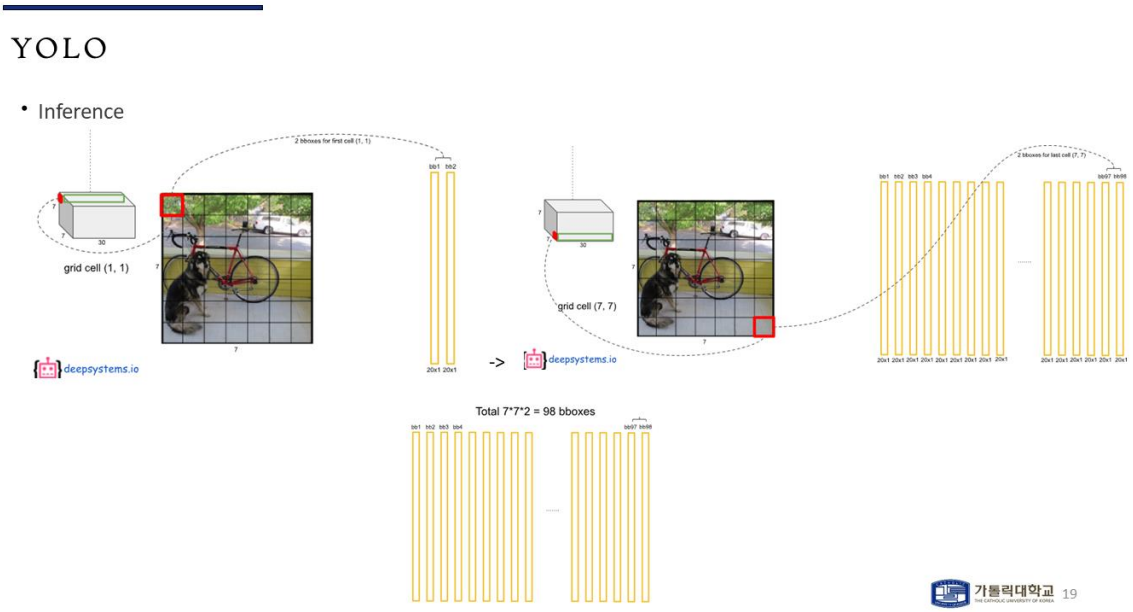
바로 기존의 클래스 조건부확률과 confidence score를 곱한 것인데 이 의미는

**(class가 박스안에 존재하는지, 박스가 물체에 얼마나 적합한지를 모두 포함한 점수)**라고 생각하

면 된다.

한 셀에 대해서 confidence score는 b개(바운딩 박스 개수)만큼 있고 클래스 조건부 확률은 c(개(클래스 개수)있으니 이 둘을 곱하면 한 셀에 대해서 (c*1)*b개의 confidence score를 가진다고 보면 된다.

논문에서 c=20, b=2니까 한 셀당 confidence score(20*1)을 2개 가진다고 생각하면 된다.



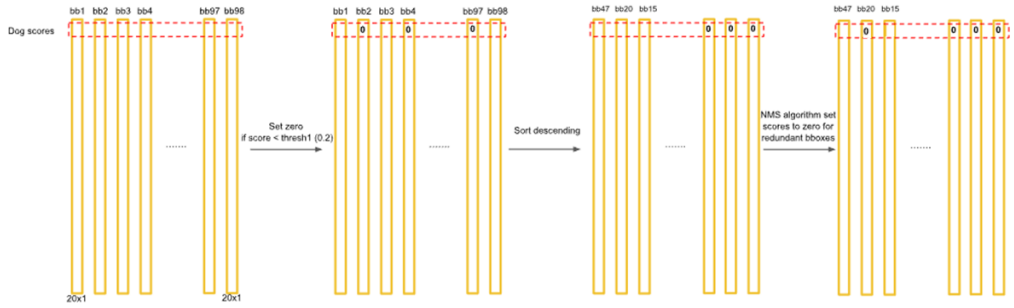

즉 49개의 셀에 대해서 confidence score를 모두 계산하면

confidence score(20*1)을 98개 가진다고 생각하면 된다.

다른 말로 98개의 bbox에 대한 confidence score(20*1)를 가진다는 의미이다.

## YOLO

- Inference




1. Threshold를 넘지 않는 score -> 0 처리
2. 내림차순 정렬
3. NMS를 통해 0 Score 추가 -> 논문에서는 98개의 박스 중 2개만 남게 됨 (Dog Box)

   Non Maximum Suppression: IoU를 기반으로 일정 수준 겹쳐있는 박스들을 모두 제거



confidence score을 구하는 것은 필요 없는 박스를 제거하는 것에 의의가 있다.

같은 행은 같은 class를 나타내게 되는데 예를 들어 첫 행은 dog에 대한 confidence scores이다.

각각의 행에 대한 점수들로 필요 없는 박스를 지워 나가는 것이다.

첫 행인 Dog Scores에 대해 살펴 보겠다.

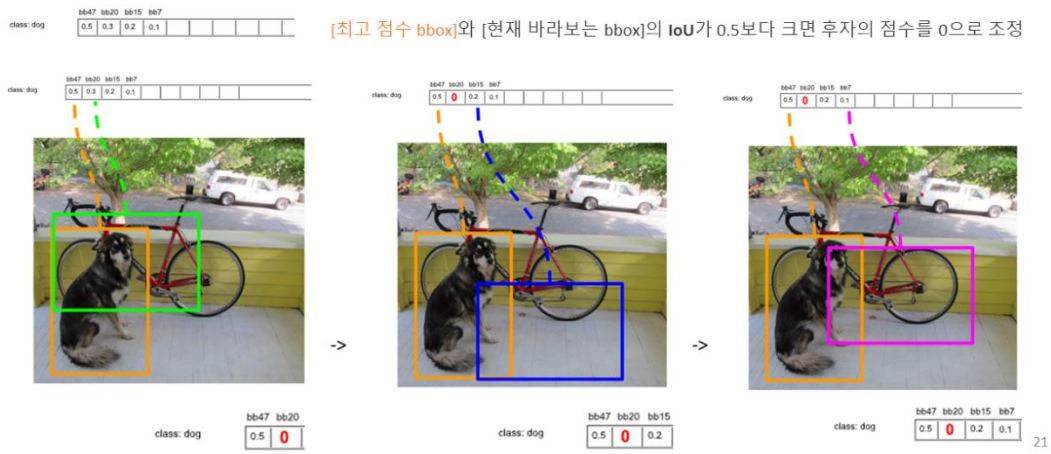
첫 번째로 일정 수준의 임계치를 넘지 않는 박스에 대해서는 score를 0처리한다. score를 0처리한다는 것은 해당 dog 관련 box에서 제거하겠다는 의미이다.

두 번째로 내림차순을 통해 점수들을 확인한다.

세 번째로 NMS 알고리즘을 통해 겹쳐있는 박스들을 제거한다. 이 때도 물론 Dog 관련 box에서 제외하겠다는 의미이다.

YOLO

- Non Maximum Suppression




NMS는 겹치는 박스를 제거하기 위한 알고리즘이다.

위 사진은 Dog Class를 기준으로 박스를 제거하는 예시이다.

박스들은 점수를 기준으로 내림차순되어 정렬돼 있다.

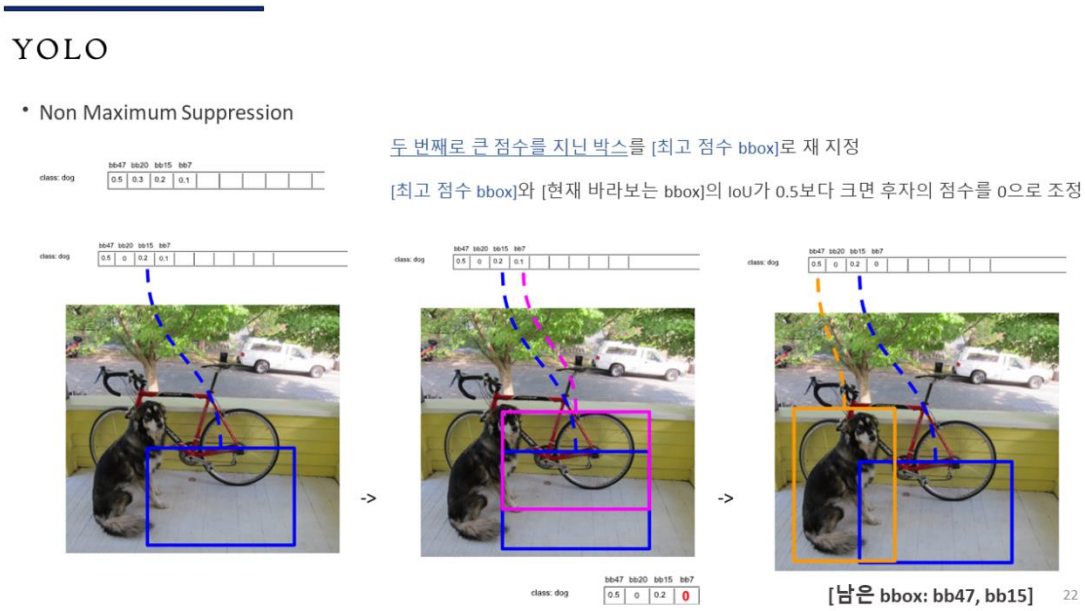
Dog Class에 대해 가장 높은 점수를 지닌 박스는 주황색 박스이다. 이 박스를 [최고 점수 bbox]로 지정한다. 그리고 이후의 박스들을 차례차례 [현재 바라보는 bbox]로 지정하고 두 bbox간 IoU가 0.5면 [현재 바라보는 bbox]를 삭제하는 알고리즘이 된다.

첫 번째 비교는 IoU가 0.5이상이라 Score가 0이 되었다.

두 번째 비교와 세 번째 비교는 IoU가 0.5미만이라 Score는 유지된다.

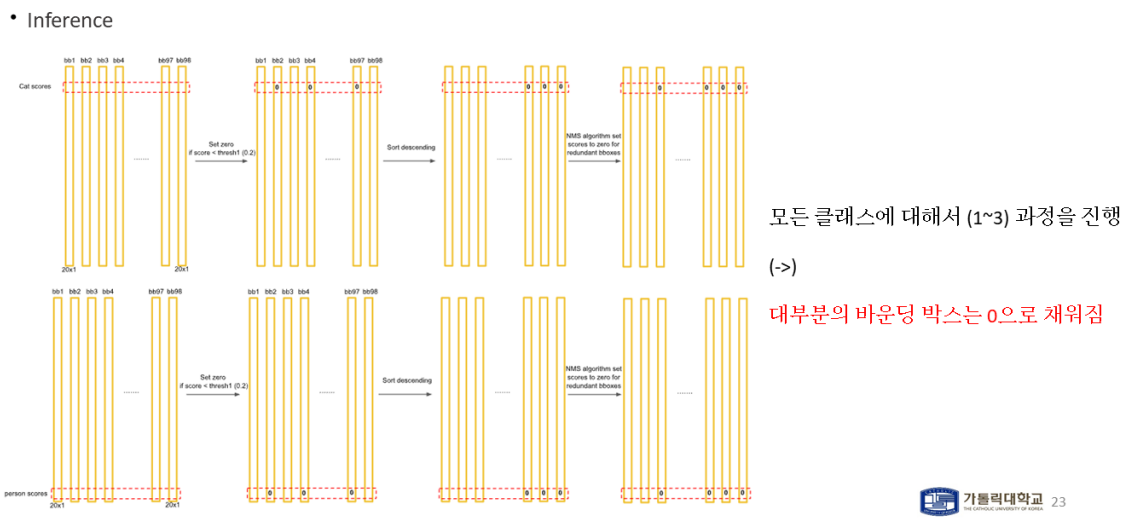
다섯번째 박스부터는 score가 처음부터 0이므로 비교가 무의미하다.

(아예 개와 관련없는 박스)




모든 박스에 대해 조사가 끝나면 두 번째로 큰 점수를 지닌 박스를 [최고 점수 bbox]로 지정한다. 그리고 똑같은 알고리즘을 진행한다.

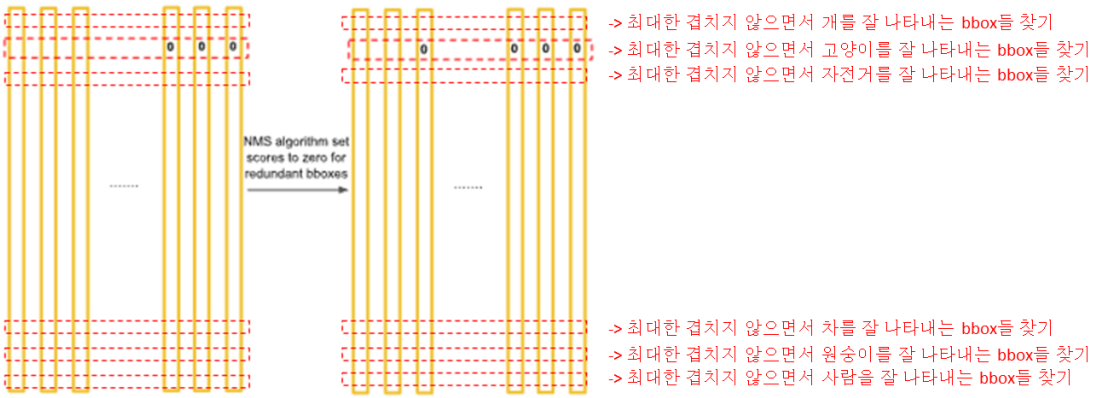
Dog Class에 대해 NMS가 끝나고 남은 박스는 두 개였다고 한다. (주황, 파랑)




이 과정을 모든 클래스에 대해서 진행한다. 그럼 대부분의 바운딩 박스는 0으로 채워지게 된다. 각 행마다 Threshold 및 NMS처리가 되면서 매우 적은 양의 bbox들만 0보다 큰 score를 가지게 될 것이다. 그리고 클래스가 20개라고는 하나 영상에서 20 종류의 객체가 나오지 않았으므로 어떤 행들은 모두 0이 될 것이다.

## YOLO

- Inference



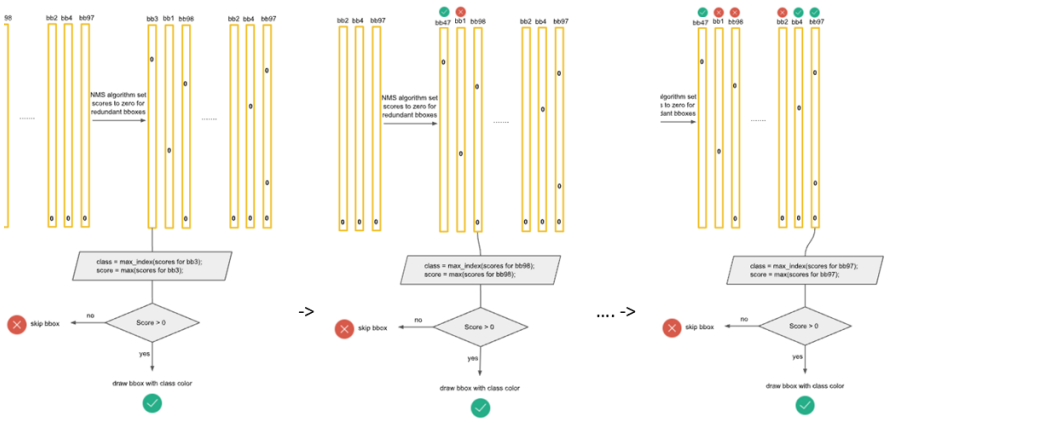



전체에 대해서 조사한다는 것은 (최대한 겹치지 않으면서 클래스를 잘 나타내는 bbox)를 찾는다는 의미와 같다. 근데 만약 한 bbox가 (최대한 겹치지 않으면서 개를 잘 나타내는 bbox)임과 동시에 (최대한 겹치지 않으면서 고양이를 잘 나타내는 bbox)로 선정되면 무슨 일이 일어날까? 해당 bbox는 (개,고양이)로 분류되는 것일까? 이런 문제를 막기 위해 (최종 결정) 작업이 추가된다.

## YOLO

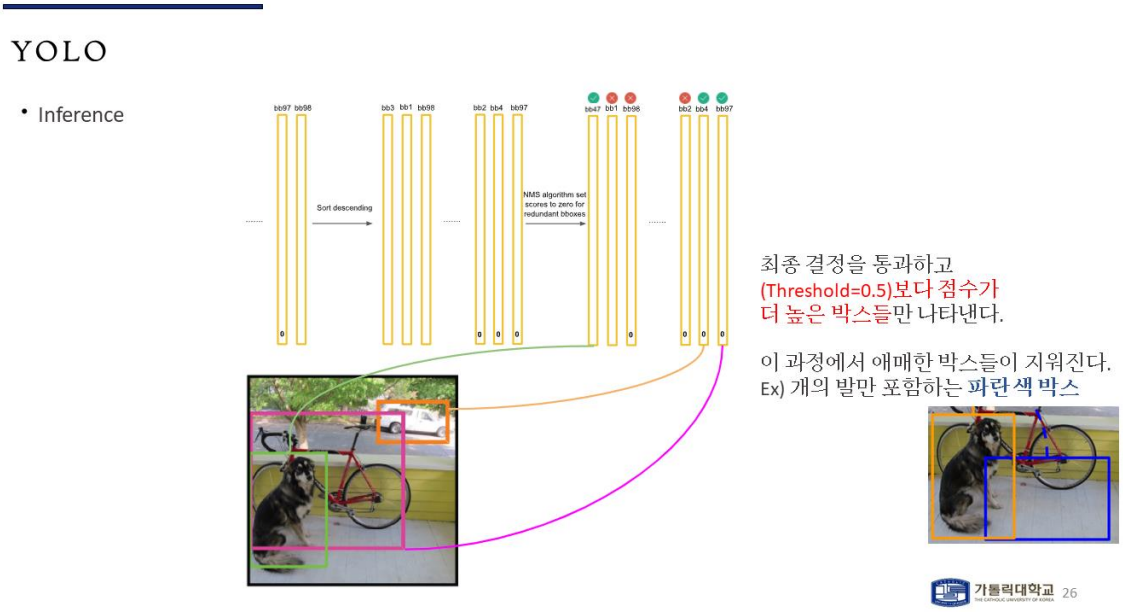
- Inference

박스 중 가장 Score가 큰 값이 0보다 크면 check, 아니면 pass






최종 결정 작업은 각 bbox마다 가장 큰 score를 기준으로 해당 score가 0보다 크면 check, 작으면 pass하겠다는 것이다. 즉 각 bbox마다 가장 확률이 높은 한 클래스를 기준으로 분류하겠다는 의미이다.

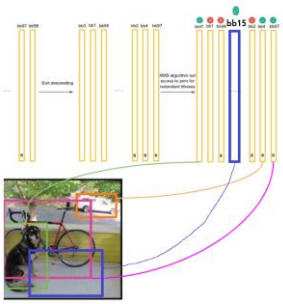



이 최종 결정을 통과하고 (Threshold=0.5)보다 점수가 더 높은 박스들만 나타내면 인퍼런스가 끝난다.

사실 이 Threshold는 사용자가 최종적으로 정하기 나름이다.

다만 Threshold를 너무 낮게 잡으면 애매한 박스들도 보일 수 있다.

예시이기는 하나, NMS를 설명하는 과정에서 Dog Class와 관련된 bbox들을 제거할 때 파란 색 박스(bb15)는 제거되지 않았다. 만약 bb98과 bb2사이에 위 bb15가 최종 결정되었고, 이 때의 클래스가 dog였다고 가정해보자.



컴퓨터 입장에서는 어느 정도 개를 나타내기도 하면서 겹치지 않았기 때문에 제거하지 않은 것이다. 하지만 인간이 보기에는 애매한 박스임이 분명하다.

이런 애매한 박스들을 마지막 Thresholding을 통해 제거하는 것이다.