Review: YOLO: Real-Time Object Detection

YOLO

- Object Detection
  - Image 내 object를 탐지해내는 방법
  - R-CNN과 같은 방법론들이 존재
    - 느리다, 최적화가 어렵다.

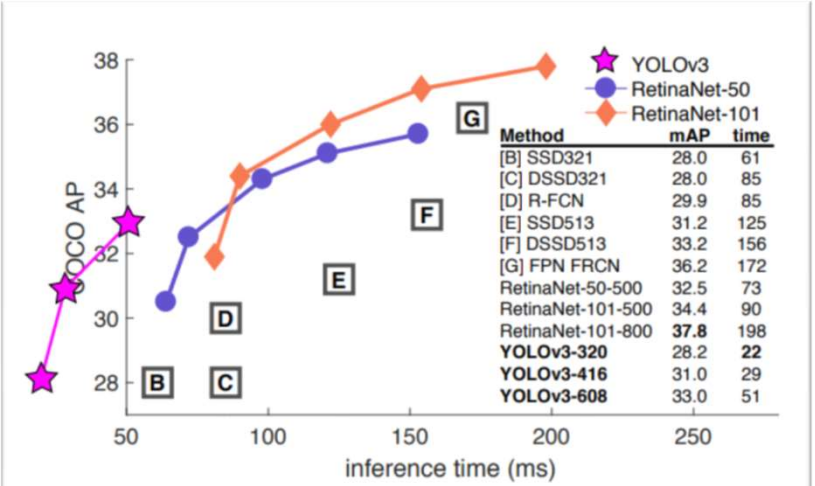
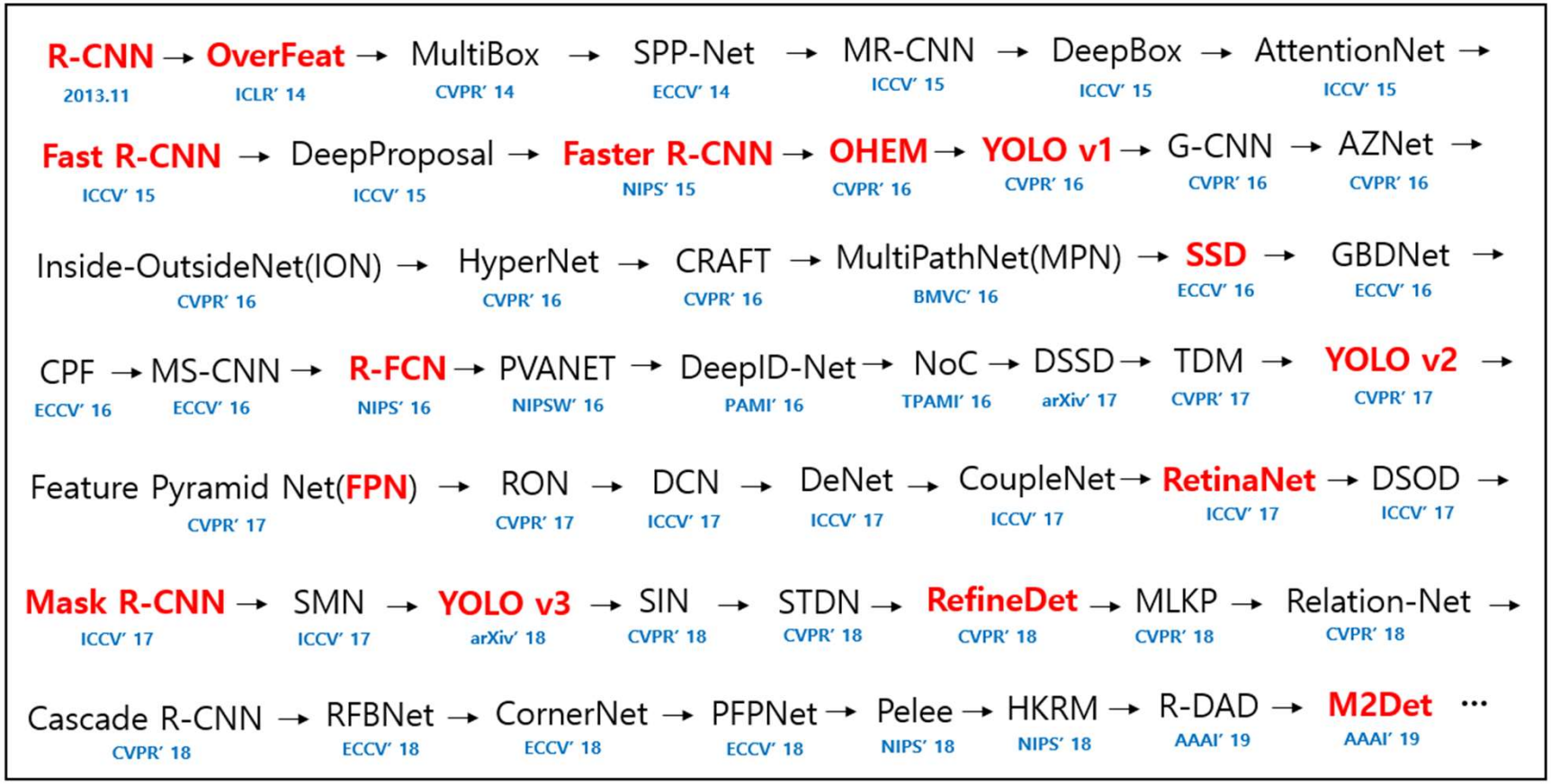


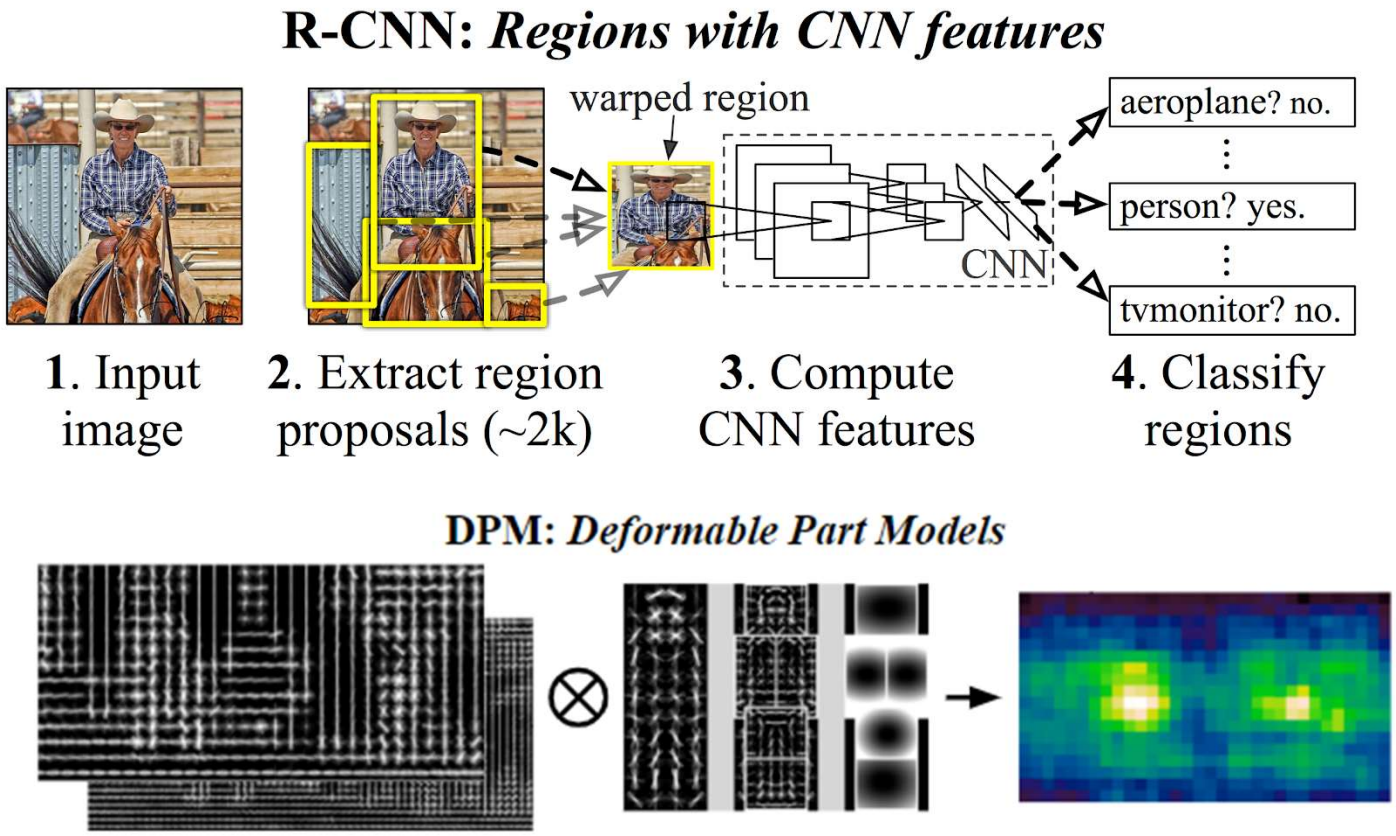
Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

Object Recognition…



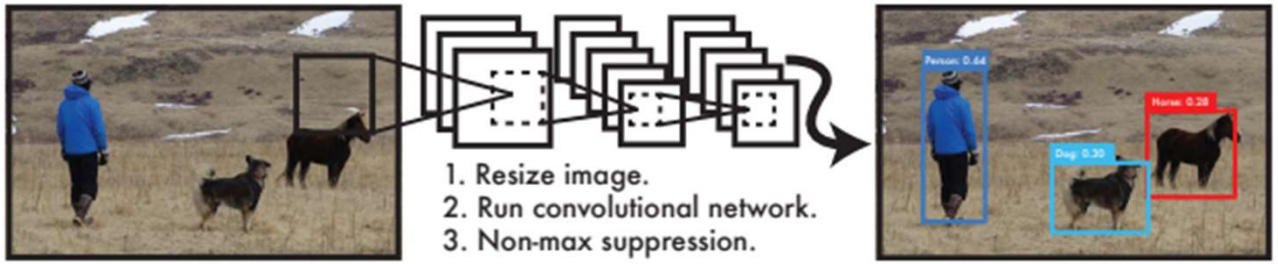
R-CNN

- Sliding window, DPM, R-CNN all train region-based classifier to perform detection



Why YOLO?

- R-CNN은 복잡한 처리과정으로 인간의 시각 시스템을 모방하기에는 아직..
  - R-CNN의 활용처를 생각해보면, 느리면 쓸 수 없다..
- YOLO(You Only Look Once)는,
  - 이미지 내 bounding box와 class probability를 single regression problem으로 간주
  - 이미지에 대해 한 번의 연산(모든 픽셀에 대해)을 통해 object의 종류와 위치를 추측



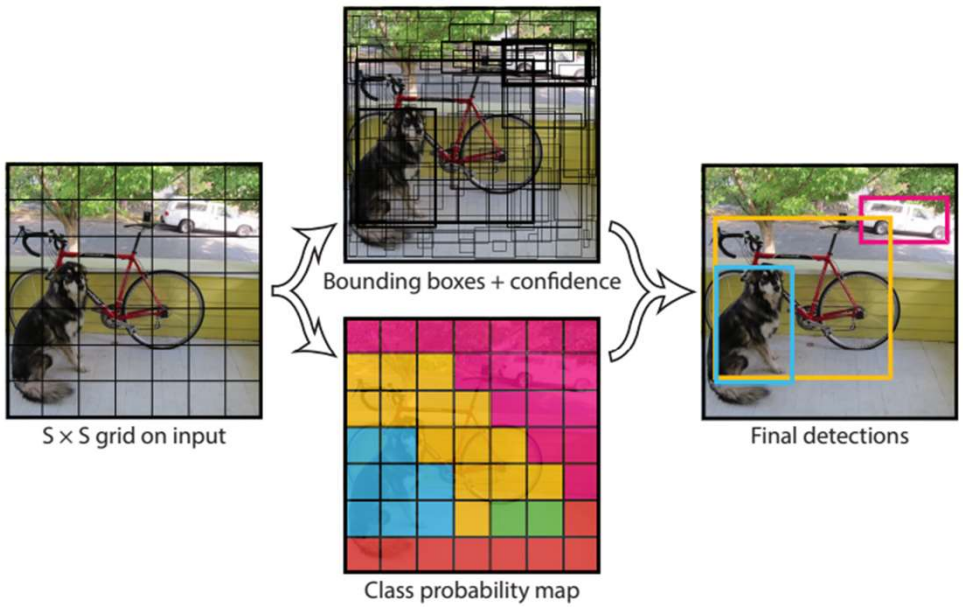
  - 1) resize the input image 448x448
  - 2) run a single convolution network on the image
  - 3) thresholds the resulting detections by the model's confidence

Pros / Cons of YOLO

- Pros
  - 간단한 처리과정으로 속도가 상대적으로 매우 빠르다.
  - 기존의 다른 real-time detection system과 비교할 때, 2배 높은 mAP (mean Average Precision)
  - Image 전체를 한번에 바라보는 방식으로 class에 대한 맥락적 이해도가 높다.
    - 낮은 background error (False-Positive)를 보인다.
  - Object에 대한 좀 더 일반화 된 특징을 학습한다.
    - Natural Image로 학습하고, Art Work에 테스트 했을 때, 다른 시스템보다 높은 성능을 보임

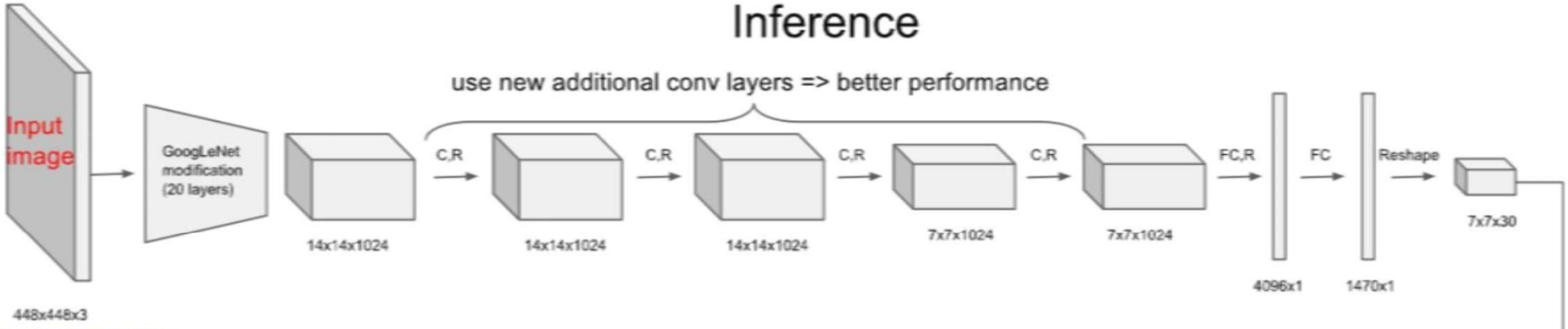
- Cons
  - 상대적으로 낮은 정확도 (특히, 작은 object에 대해 - YOLO v1)

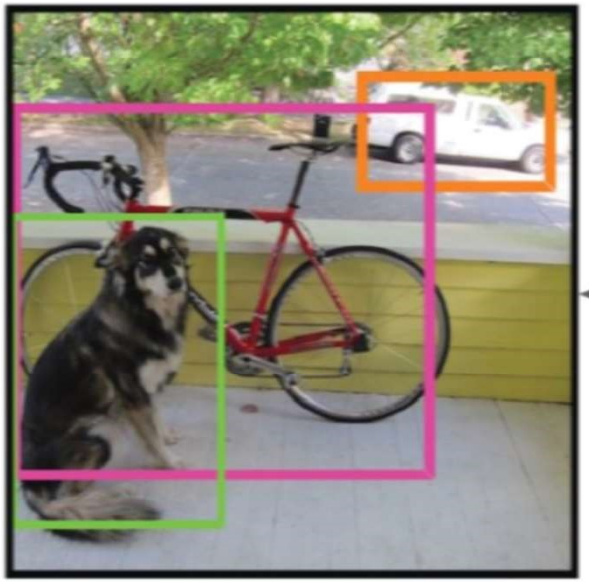
YOLO: Overview



- Input Image를 $S \times S$ 개의 grid로 나눈다.
- 각각의 grid는
  - B개의 bounding box
  - Bounding box에 대한 confidence score를 갖는다.
  - ex) Grid에 object가 존재하지 않으면 score = 0
  - C개의 conditional class probability를 갖는다.
- 각각의 bounding box는
  - x, y, w, h, confidence 로 구성
  - (x,y): bounding box의 중심점, grid cell에 대한 상대 값
  - (w,h): 전체 이미지의 weight, height에 대한 상대 값
  - ex) x 가 grid cell의 왼쪽에 있다면, x = 0
  - ex) y 가 grid cell의 오른쪽에 있다면, y = 0.5
  - ex) bbox의 width가 이미지 with의 절반이라면 w = 0.5

Inference



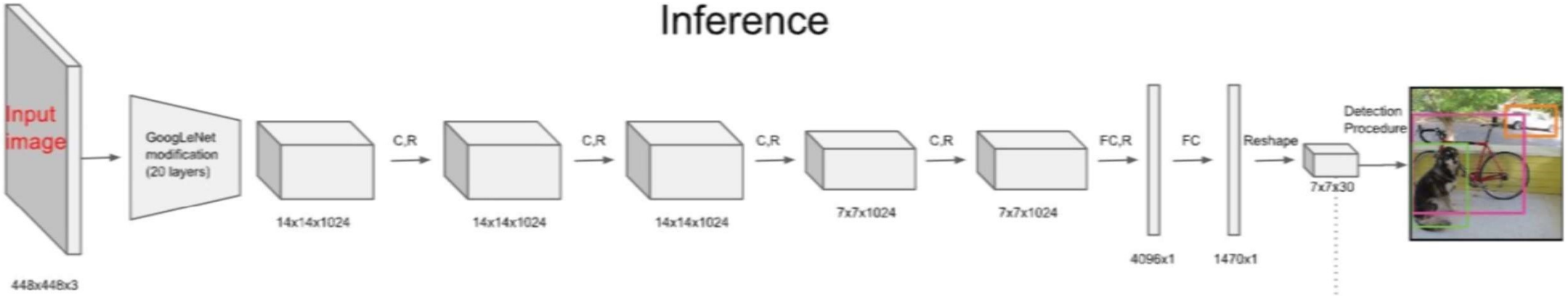


Detection Procedure

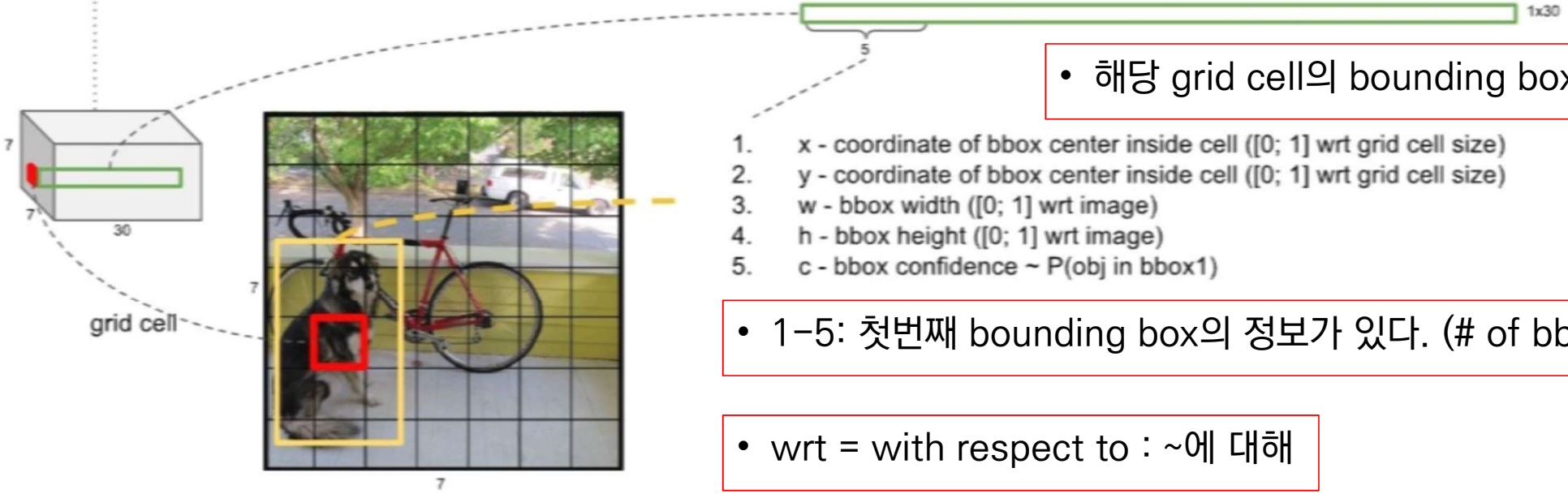
- Input Image는 448 * 448 * 3
- Output Image는 찾아낸 object가 표시

https://youtu.be/L0tzmv--CGY

Inference



Tensor values interpretation

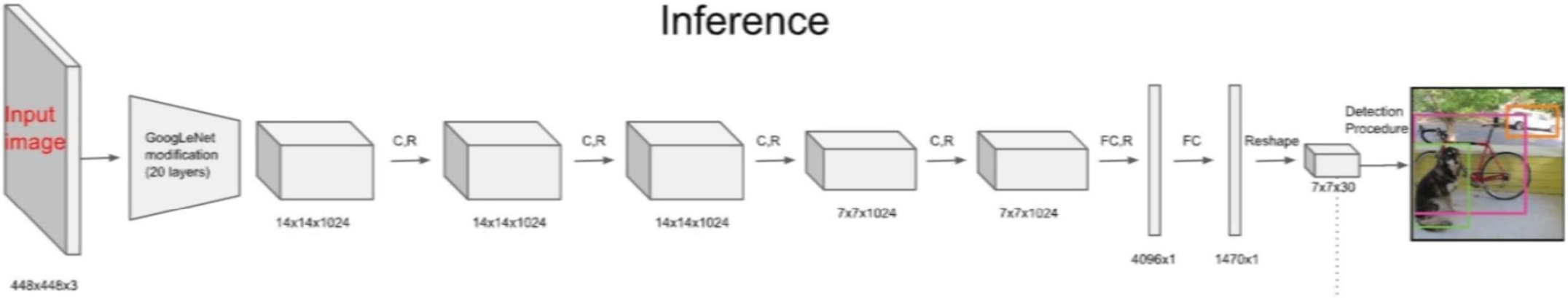


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

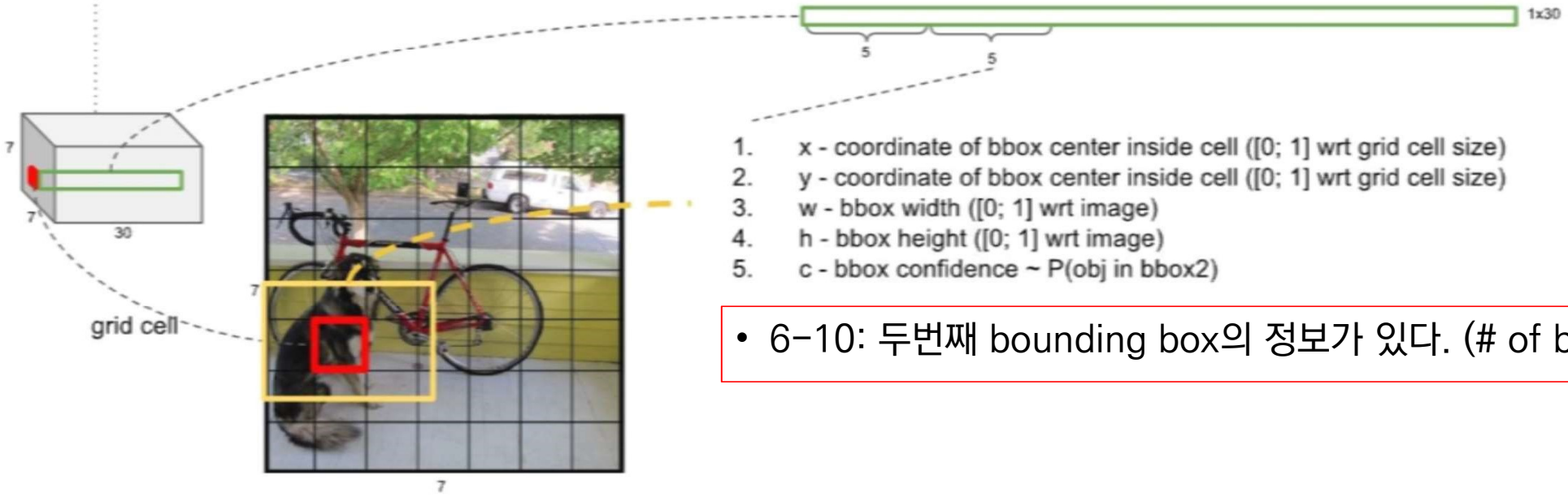
- 해당 grid cell의 bounding box
- 1–5: 첫번째 bounding box의 정보가 있다. (# of bbox = 2)
- wrt = with respect to : ~에 대해

deepsystems.io https://youtu.be/L0tzmv--CGY

Inference



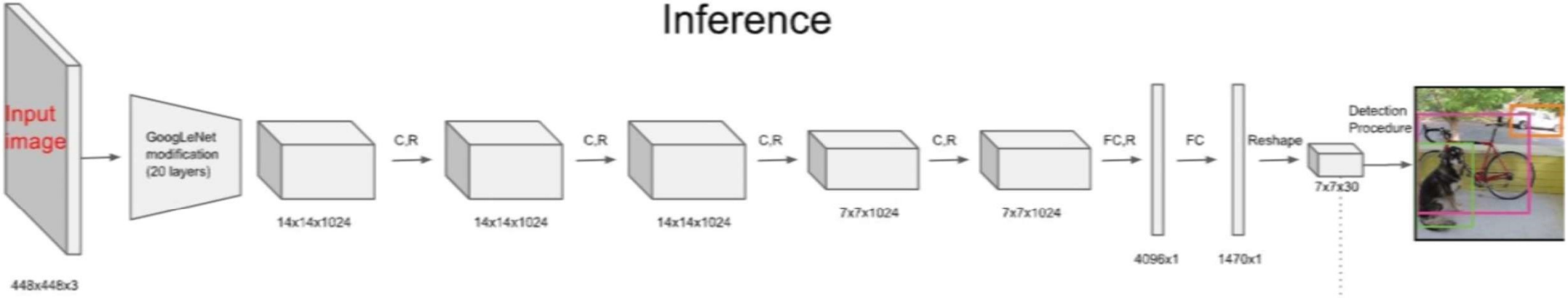
Tensor values interpretation

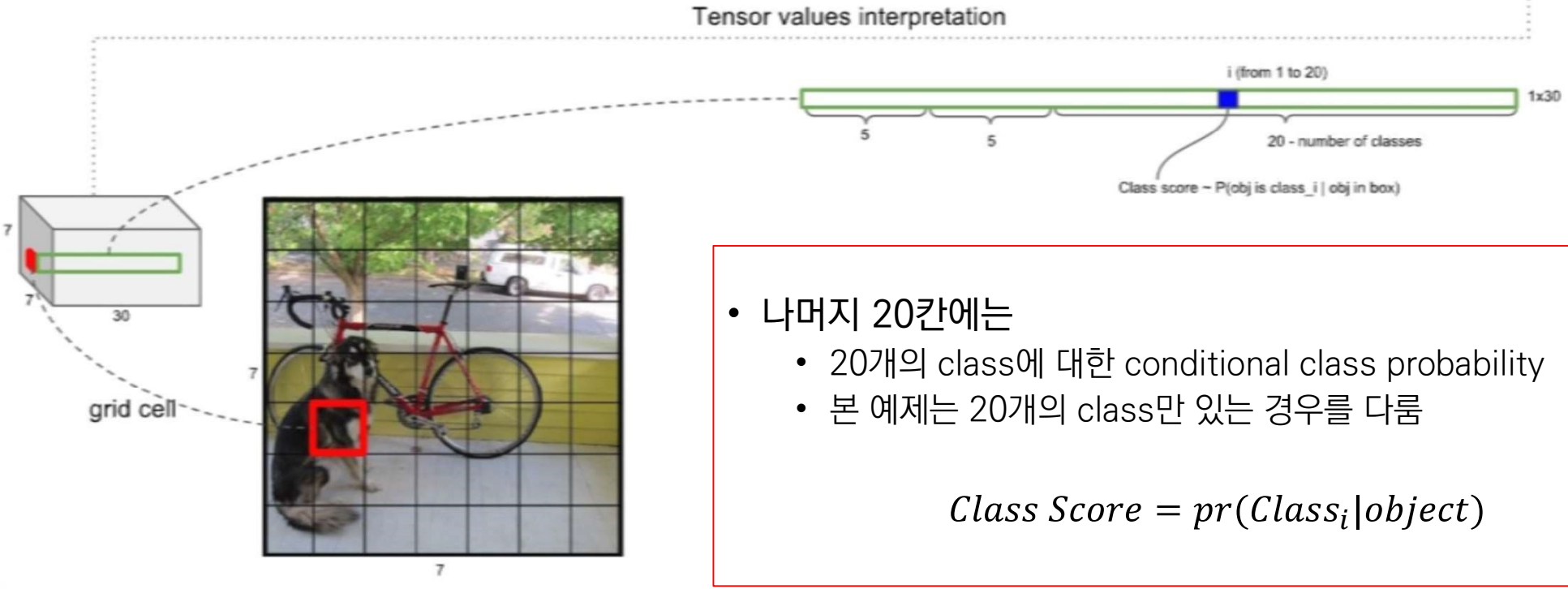


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

- 6-10: 두번째 bounding box의 정보가 있다. (# of bbox = 2)

Inference

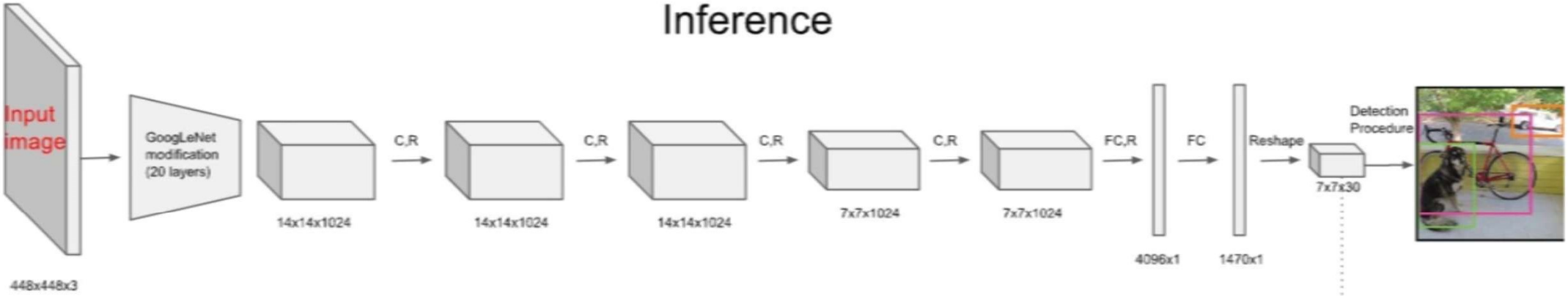


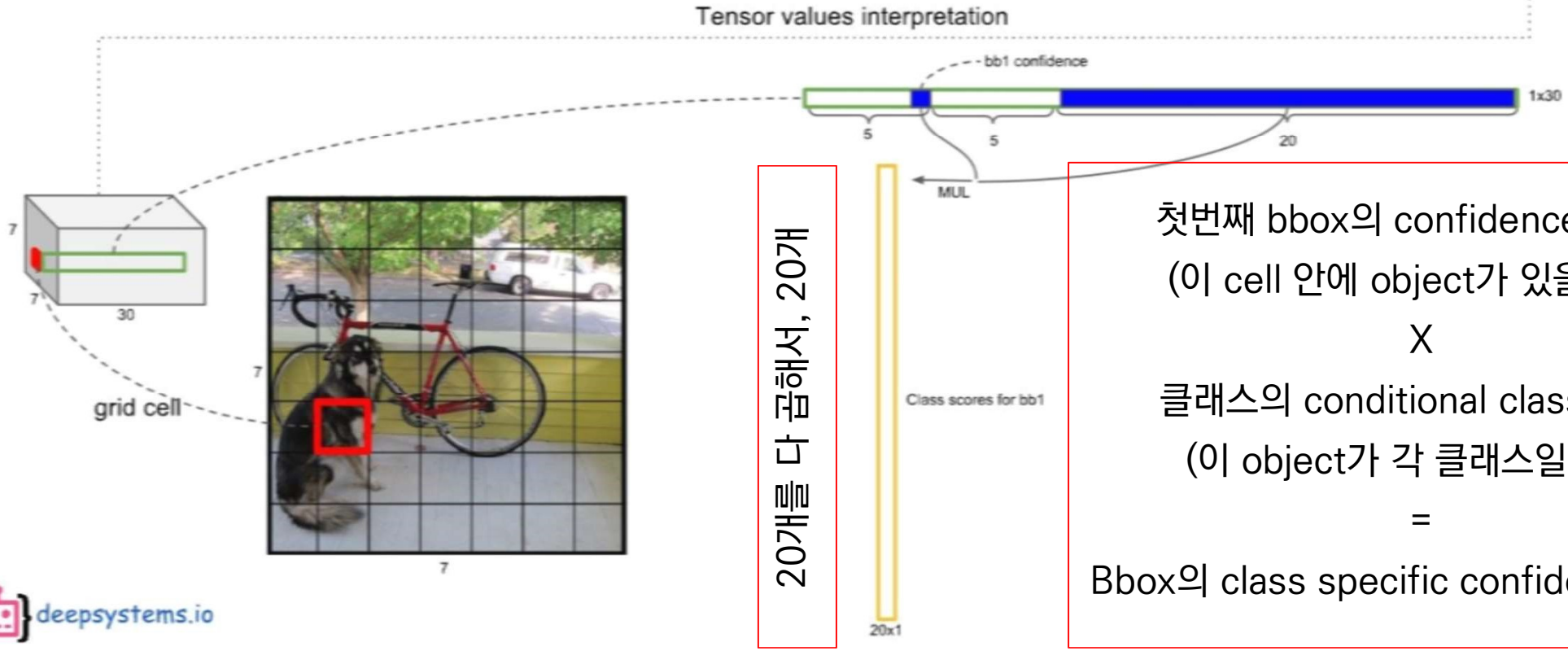


- 나머지 20칸에는
  - 20개의 class에 대한 conditional class probability
  - 본 예제는 20개의 class만 있는 경우를 다룸

$$Class\ Score = pr(Class_i|object)$$

Inference



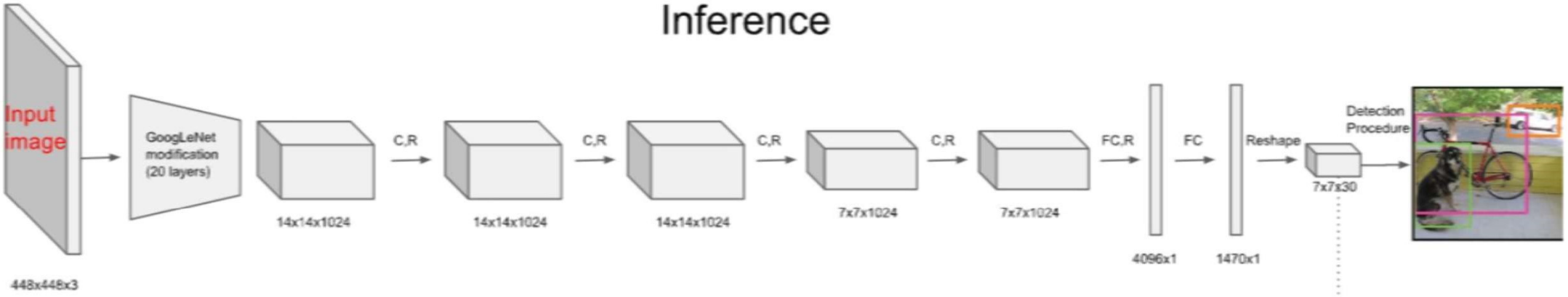


20개를 다 곱해서, 20개

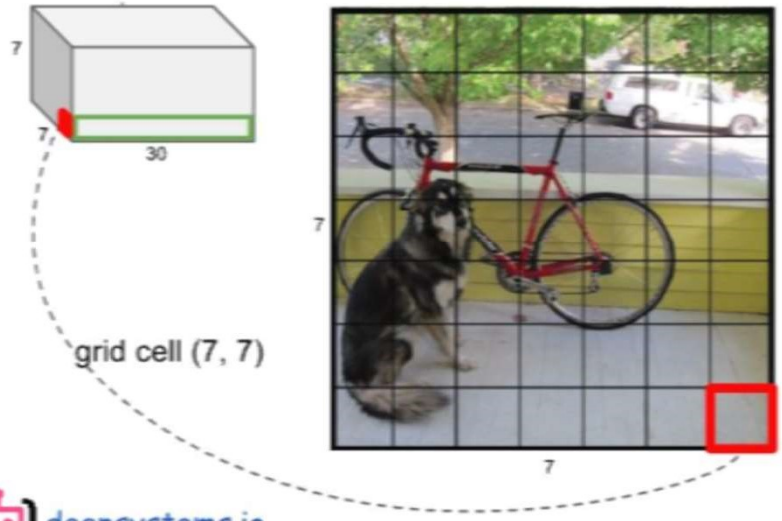
첫번째 bbox의 confidence score
(이 cell 안에 object가 있을 확률)
X
클래스의 conditional class prob.
(이 object가 각 클래스일 확률)
=
Bbox의 class specific confidence score

deepsystems.io

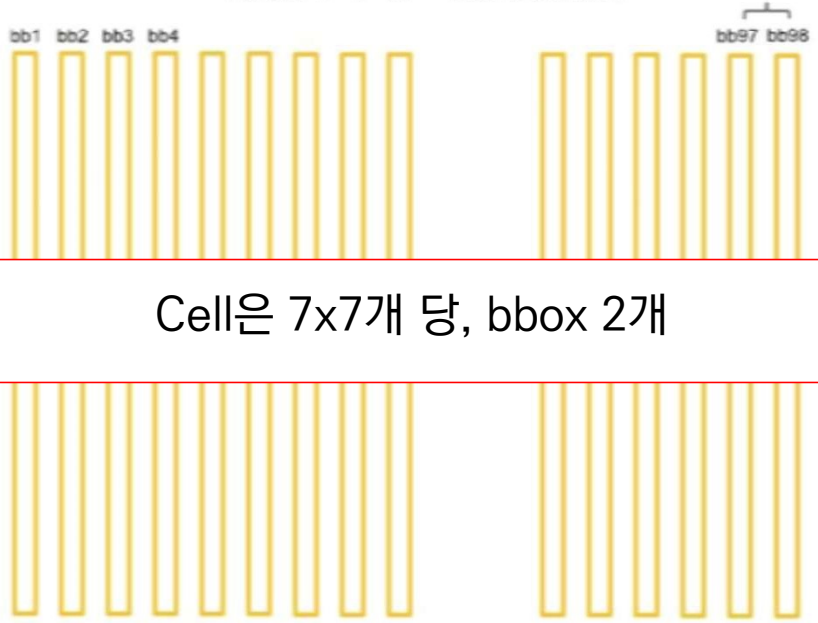
Inference



Tensor values interpretation



Total 7*7*2 = 98 bboxes



Cell은 7x7개 당, bbox 2개

deepsystems.io

Detection Procedure

https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.g137784ab86_4_2333

Network Design

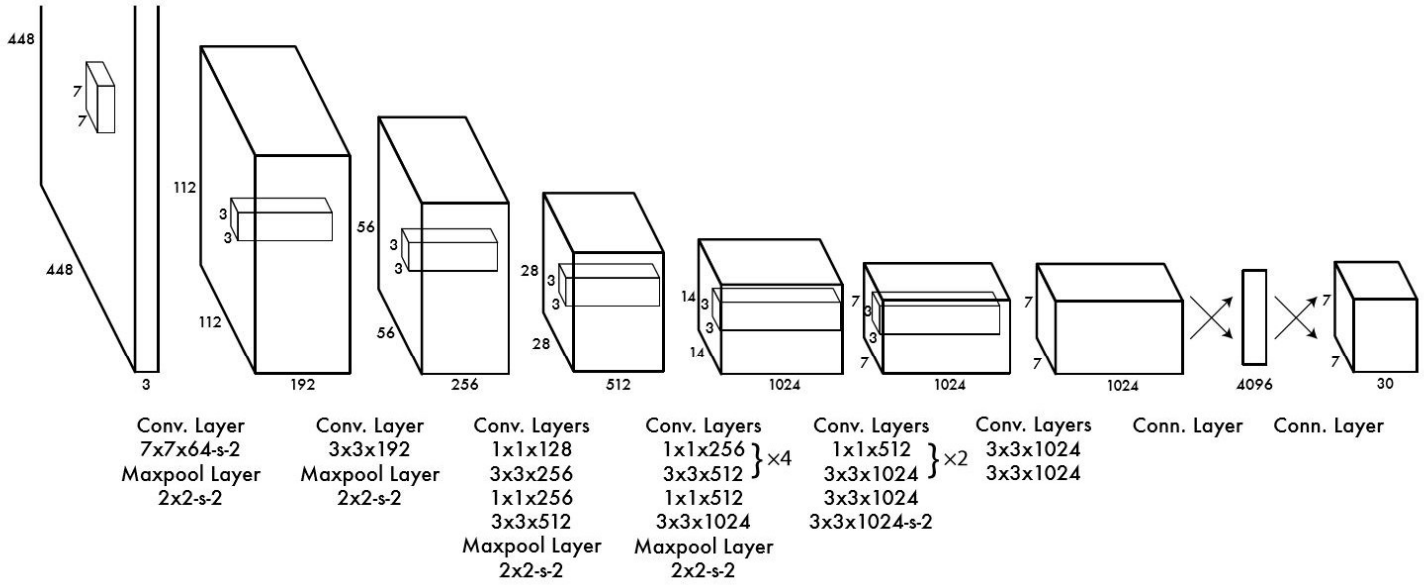


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

- Google LeNet for image classification 기반, 24 Convolution Layer, 2 FC Layer
- Fast YOLO는 9개의 Convolution Layer

YOLO: Loss function

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

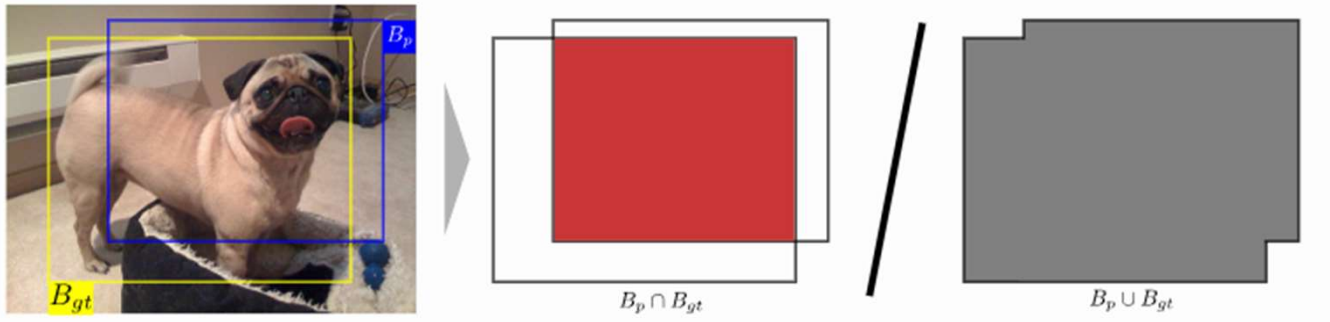
YOLO: Loss function

- YOLO는 각 grid cell 마다 다수의 bounding box를 예측하지만,
  - 탐지된 객체를 잘 포함하는 box하나를 선택해야 한다!

- 이를 위해, ground truth와 IOU를 계산해서 가장 높은 IOU를 가진 하나를 선택한다.

- IOU (intersection over union)
  - 예측 바운딩 박스 B_p와 Ground Truth 바운딩 박스 B_{gt}에 대하여, 서로가 얼마나 겹치는지를 평가

$$IOU(B_p, B_{gt}) = \frac{area\ of\ B_p \cap B_{gt}}{area\ of\ B_p \cup B_{gt}}$$



YOLO: Classification loss

- 우선 박스도 박스지만, 제대로 된 객체를 검출해 냈는지 역시 중요하다.

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

  - Where,
    - $\mathbf{1}_i^{obj} = 1$, if object appears in cell i, otherwise 0.
    - $\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

- Example
  - 어떤 셀 i에서 객체가 탐지되었는데, i 셀의 정답은 개일 확률은 1이고, 고양이일 확률은 0이다.
  - 이 경우, 고양이가 0.3, 개가 0.7로 예측 되었다면, classification loss는?

  - Threshold… 개념이 필요할까?

YOLO: Localization Loss

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

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

- Where,
  - $\mathbf{1}_i^{obj} = 1$, if object appears in cell i, otherwise 0.
  - λ_{coord} increases the weight for the loss in the boundary box coordinates.

- Example
  - 1. 객체가 없다고 검출된 경우, 이 Loss는 얼마인가?
  - 2. 셀 1개에 대해 bounding box가 2개가 검출되었다. 이때, S^2와 B는 각각 얼마인가?
  - 3. 객체가 검출되었다고 가정하고 $[x, y, w, h, confidence] = [0.3, 0.4, 0.1, 0.2, 1.0]$ 로 검출된 bbox의 label은 $[0.32, 0.41, 0.08, 0.19, 1.0]$이다. 이때 Localization Loss는 얼마인가?

YOLO: Confidence Loss

- 객체가 탐지된 경우 confidence loss function은 아래와 같으며,

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

  - Where, $\hat{C}_i$ is the box confidence score of the box j in cell i.

- 객체가 탐지되지 않은 경우 confidence loss function은 아래와 같다.

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

  - 1_{ij}^{noobj} is the complement of 1_{ij}^{obj}
  - λ_{noobj} weight down the loss when detecting background.

- λ_{noobj}는 보통의 경우에 0.5 정도를 준다고 한다.. 왜일까?
  - 그림에는 배경이 많다는 사실을 기억해보자 .. 클래스 불균형..

YOLO: Loss function

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

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

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

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

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

한번 다시 봅시다.
Loss Function은 절대 어렵지 않습니다.

YOLO Training

- ImageNet 1000-class dataset으로 convolution layer → pre-training
- Pre-training 이후, 4 Convolution Layers와 2 FC layer를 추가
- Bounding Box의 width와 height는 이미지의 width와 height로 normalize (0~1)
- Bounding Box의 x와 y는 특정 grid cell 위치의 offset 값을 사용 (0~1)
- λ_{coord}: 5, λ_{noobj} : 0.5
- Batch size : 64
- Momentum : 0.9 with decay rate 0.0005
- Learning rate w/ schedule : 0.01 (~75 epoch), 0.001 (105 epoch), 0.0001 (135 epoch)
- Dropout Rate : 0.5
- Data Augmentation : Random scaling and translation of up to 20% the original image size
- Activation Function : Leaky Rectified Linear Activation

$$\phi(x) = \begin{cases} x, & if\ x > 0 \\ 0.1x, & otherwise \end{cases}$$

YOLO: Result!

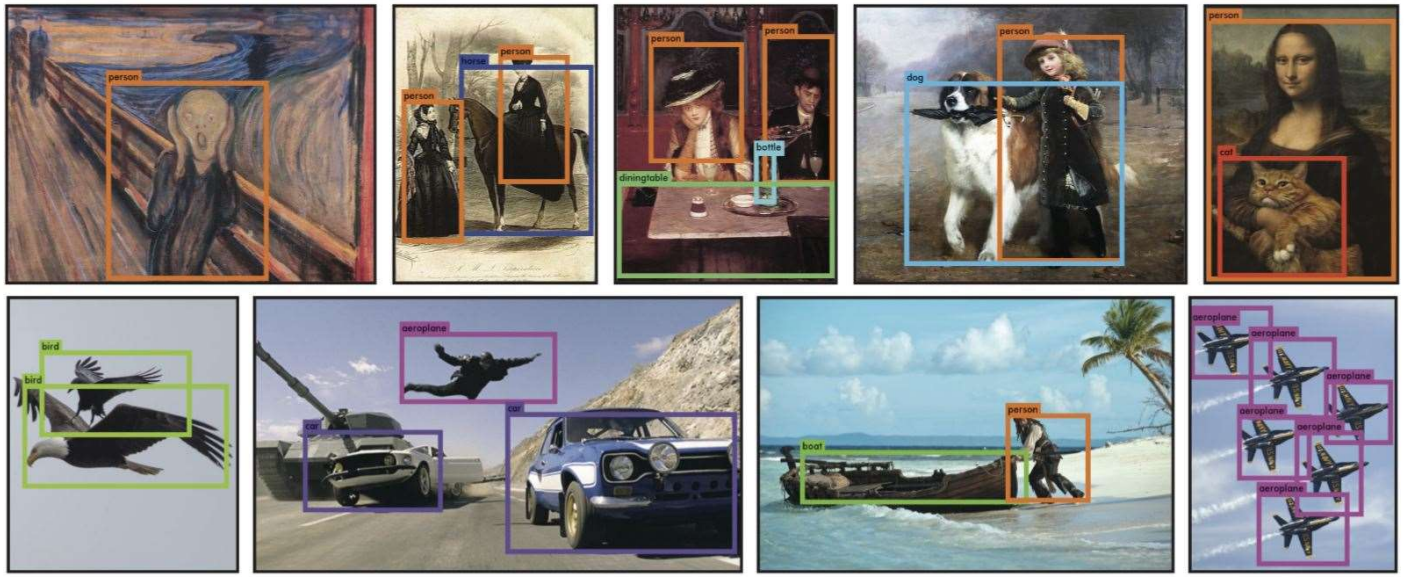
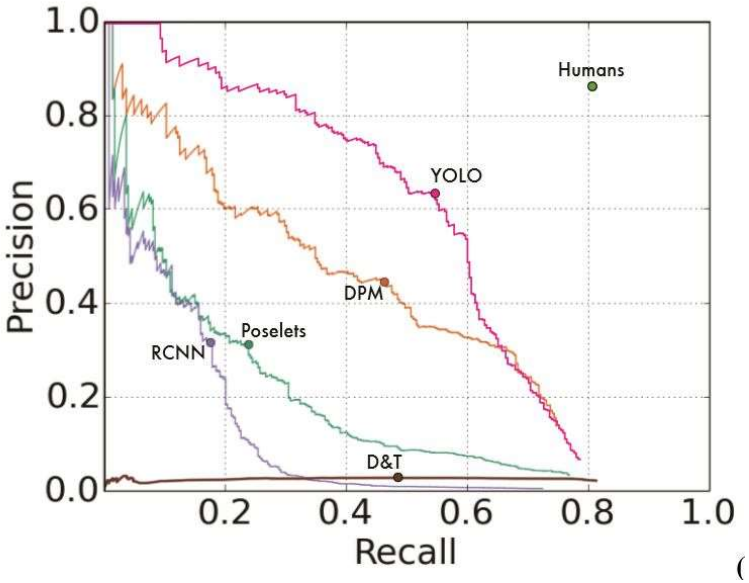


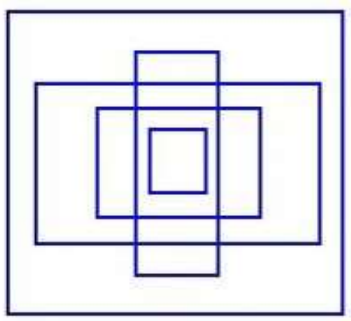
Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.



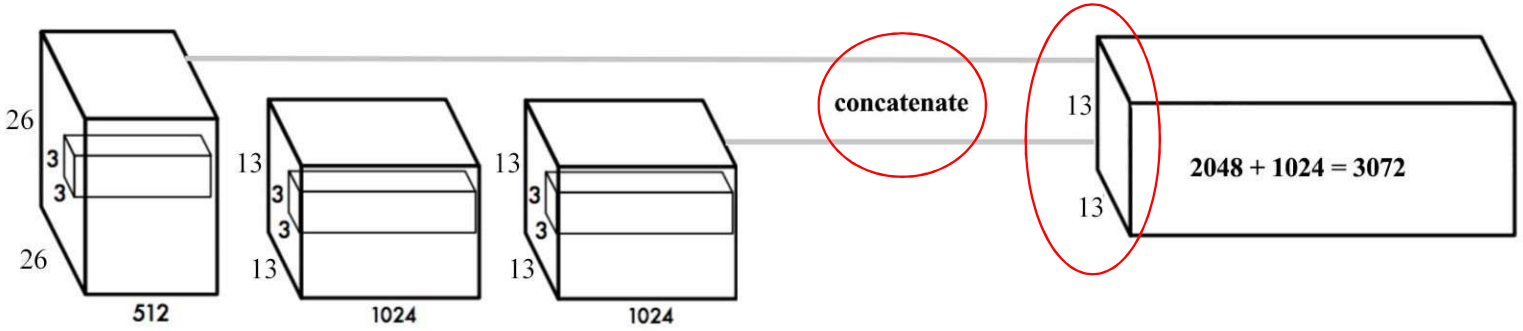
Picasso Dataset.. Curve

YOLOv1 to YOLOv2

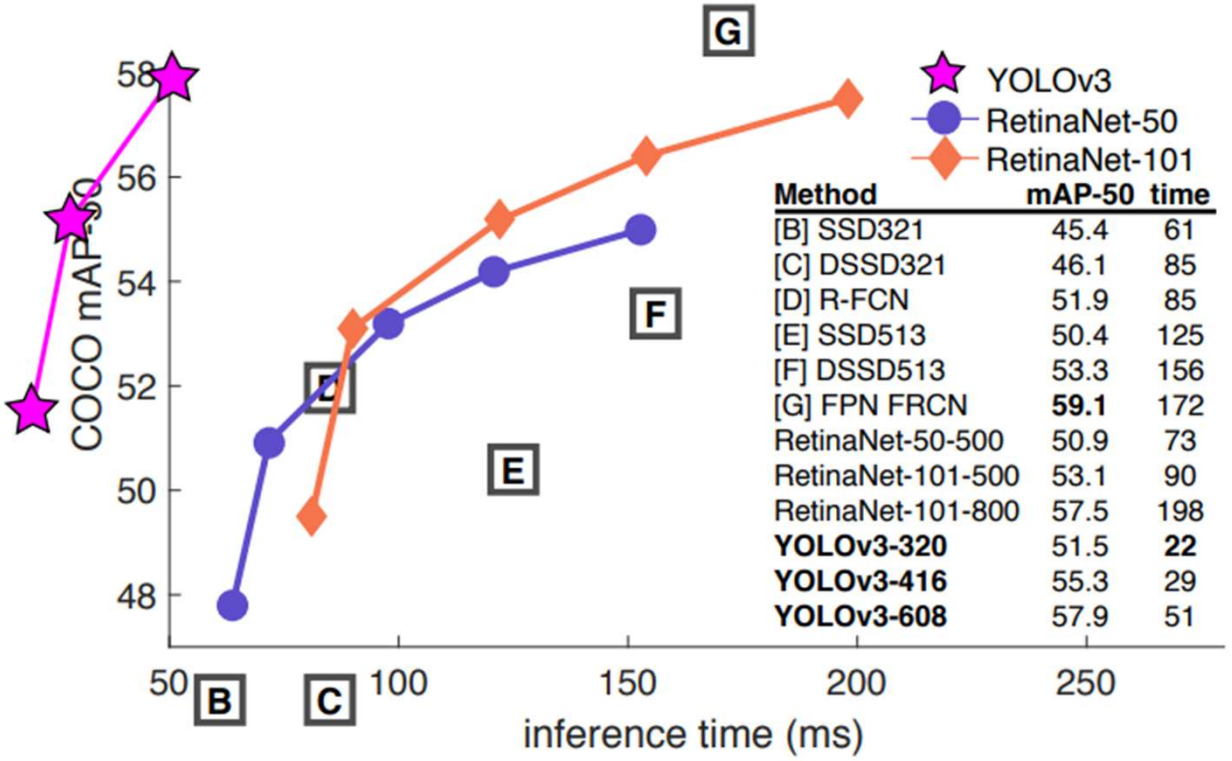
- ADD Batch Normalization Layer
- Convolution with Anchors
  - 사전에 정의된 크기의 box 설정
  - 초기학습에 대한 안정화
  - Object의 크기를 처음부터 맞추기는 어렵다
  - CCTV영상에서 차량 검출이 목적인 경우
    - CCTV내 차량 크기 → Anchor 크기
  - mAP감소, recall 증가 효과



- Remove Fully Connected Layers
- Modify Network Structure



YOLOv2 to YOLOv3



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

- Fine Tuning
- https://pjreddie.com/media/files/papers/YOLOv3.pdf