Object Detection

POSTECH MIP Lab.

TA: Seunghun Baek, Joonhyuk Park, Soojin Hwang

Goal

- Understand object detection
  - Basic concept
  - R-CNN
  - Evaluation metric
- Exercise1: Take a look Pascal VOC dataset
- Exercise2: Implement IoU, NMS

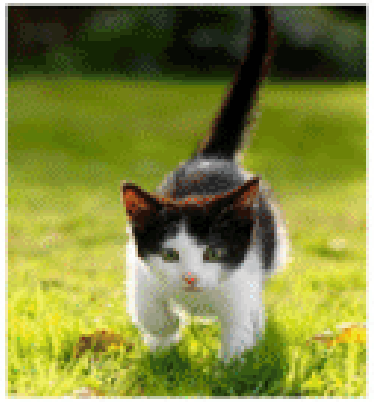
Detection Task

Semantic Segmentation

CAT GRASS TREE

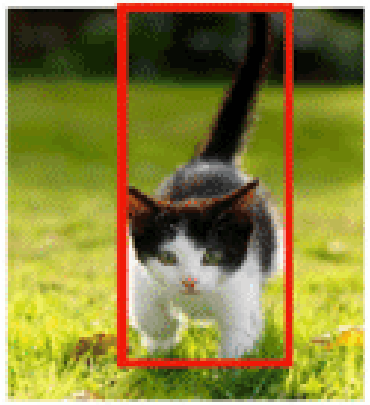
No object
Just pixels

Classification



CAT

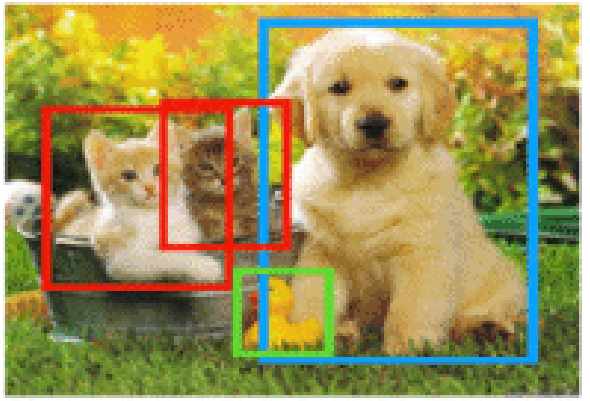
Classification + localization



CAT

Single object

Object detection



CAT DOG DUCK

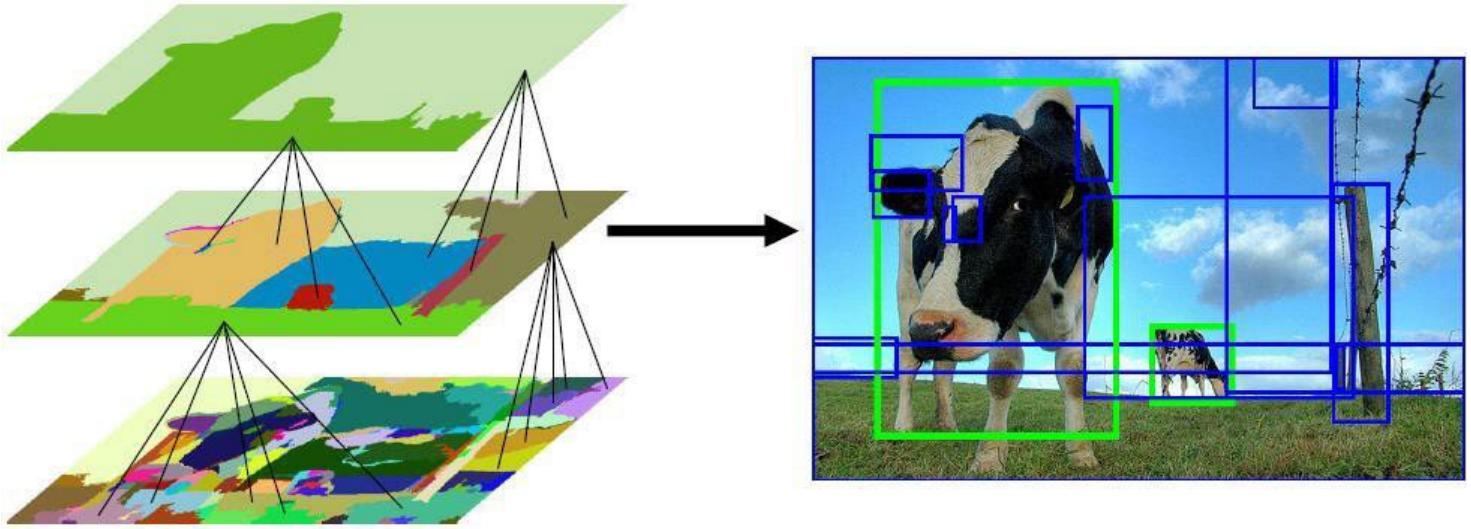
Instance segmentation

CAT CAT DOG DUCK

Multiple objects

Object Detection

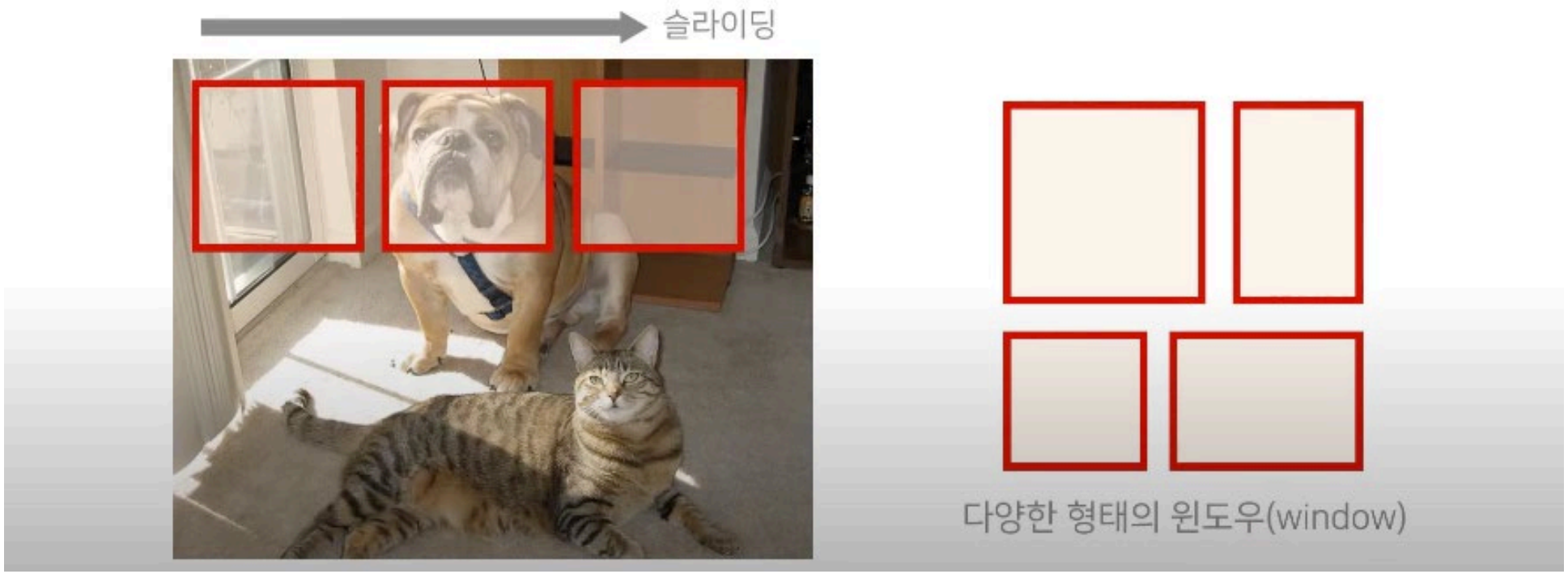
- Detection = (multi) Localization + Classification
- Localization
  - Sliding window
  - Region proposal
  - Branch-and-bound search
  - Other techniques



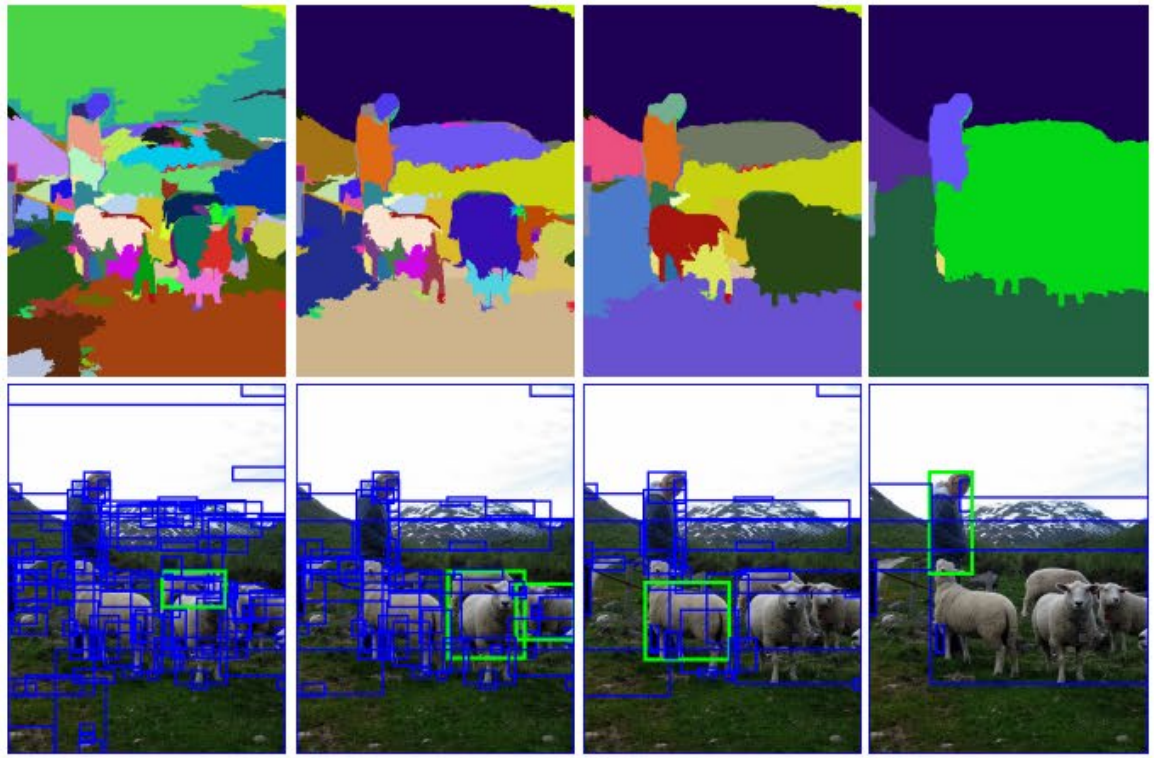
- Classification
  - Classifying the localized object candidate regions

Sliding Window

- 이미지에서 다양한 형태의 윈도우(window)를 슬라이딩(sliding)하며 물체가 존재하는지 확인합니다.
  - 너무 많은 영역에 대하여 확인해야 한다는 단점이 있습니다.



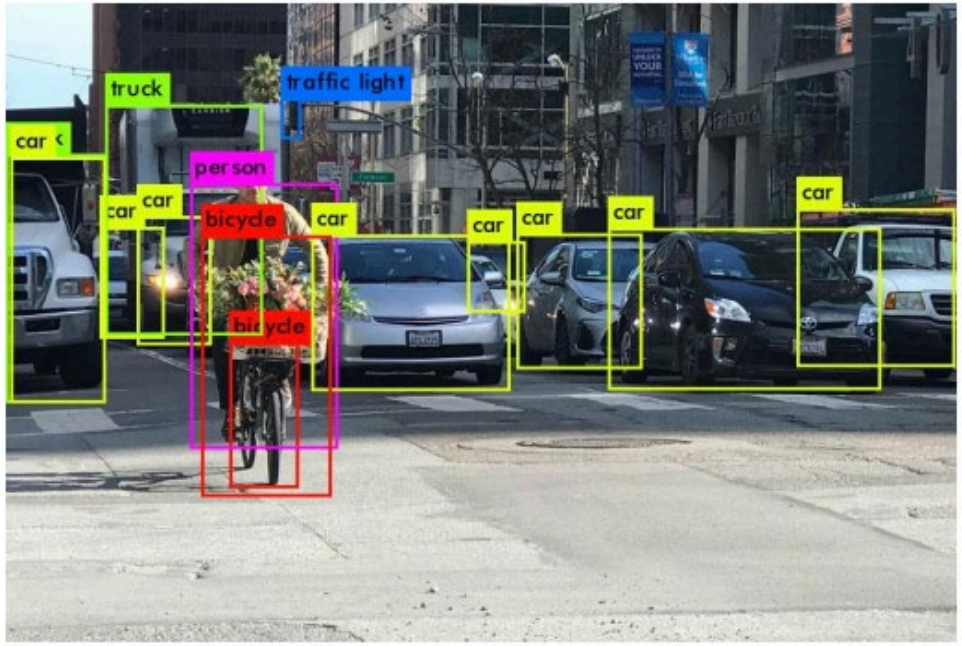
Region Proposal : Selective Search



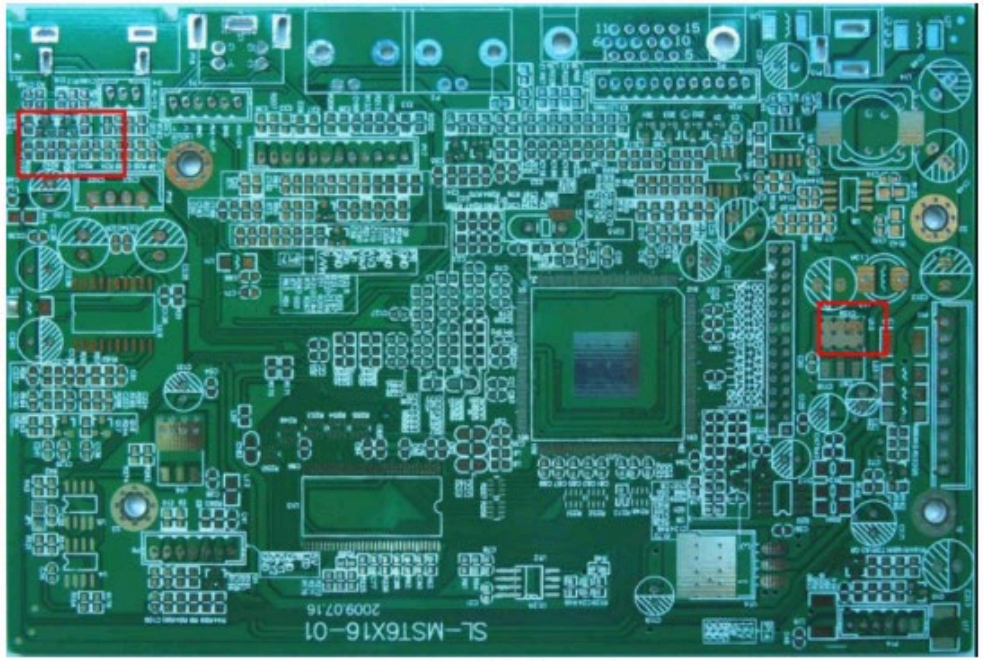
(a)

(b)

Why object detection?



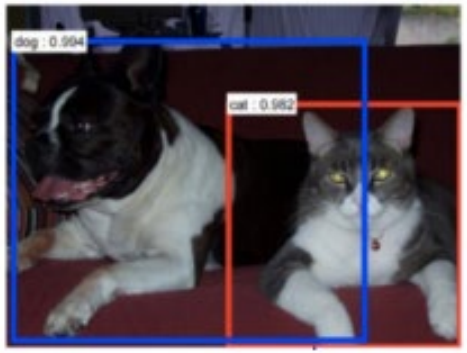
Autonumous driving

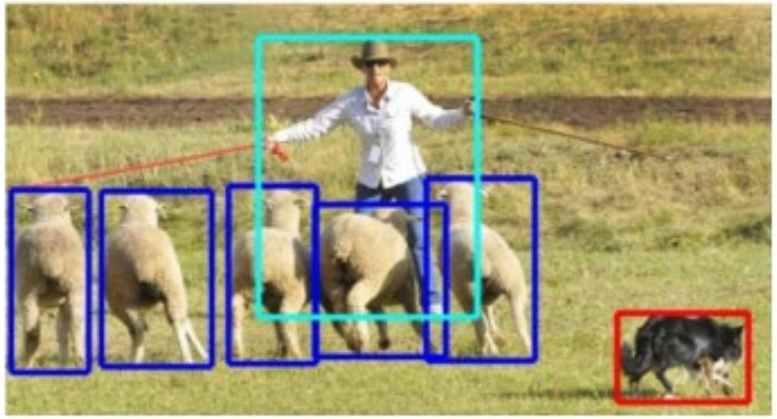


IC board with defects detection

Popular Datasets

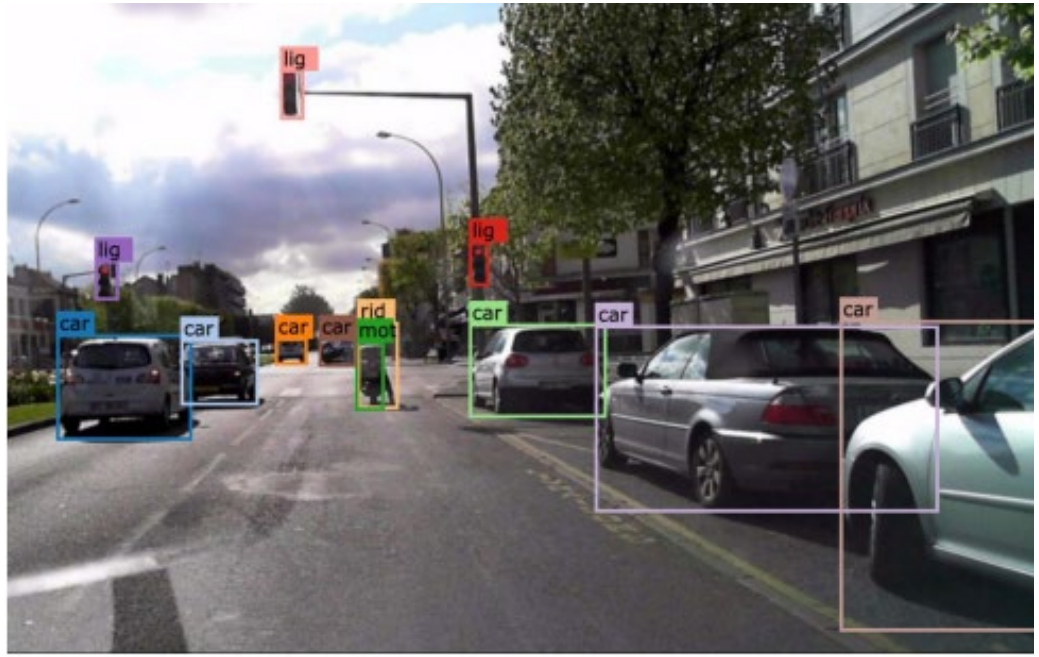
- Training data consists of image, bounding box coordinates with class labels
- PASCAL VOC
  - Medium-scale object detection dataset
  - 20 classes
  - Train (5K), Validation (2K)
- COCO (Common Object In Context)
  - A large-scale object detection dataset
  - 80 categories
  - Train (81K), validation (41K)





Popular Datasets

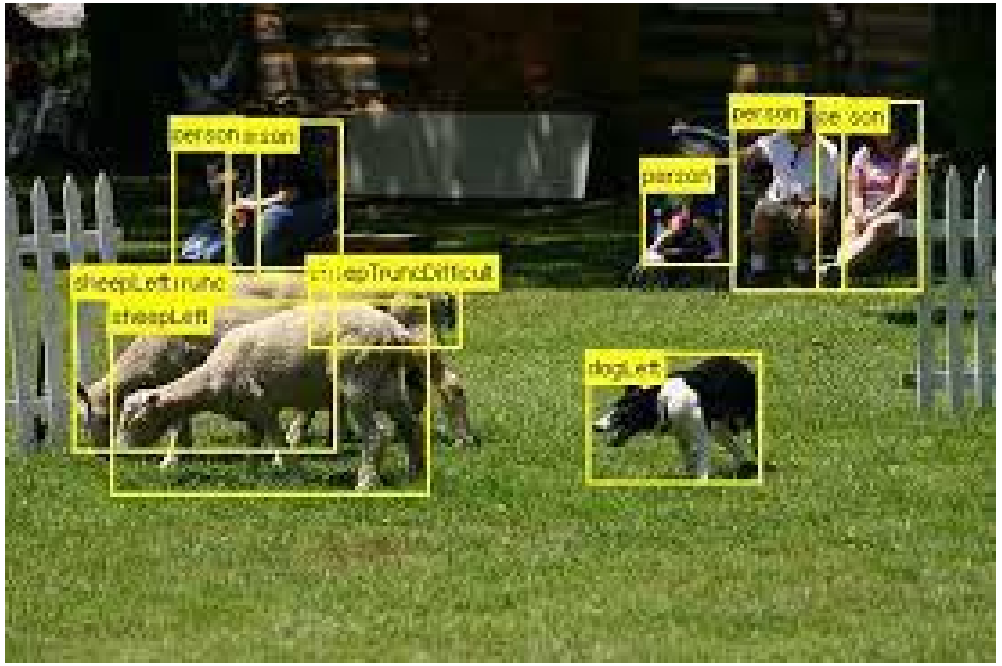
- Training data consists of image, bounding box coordinates with class labels
- Berkley DeepDrive
  - Urban street scenes (50 cities)
  - 30 classes
  - 100K videos



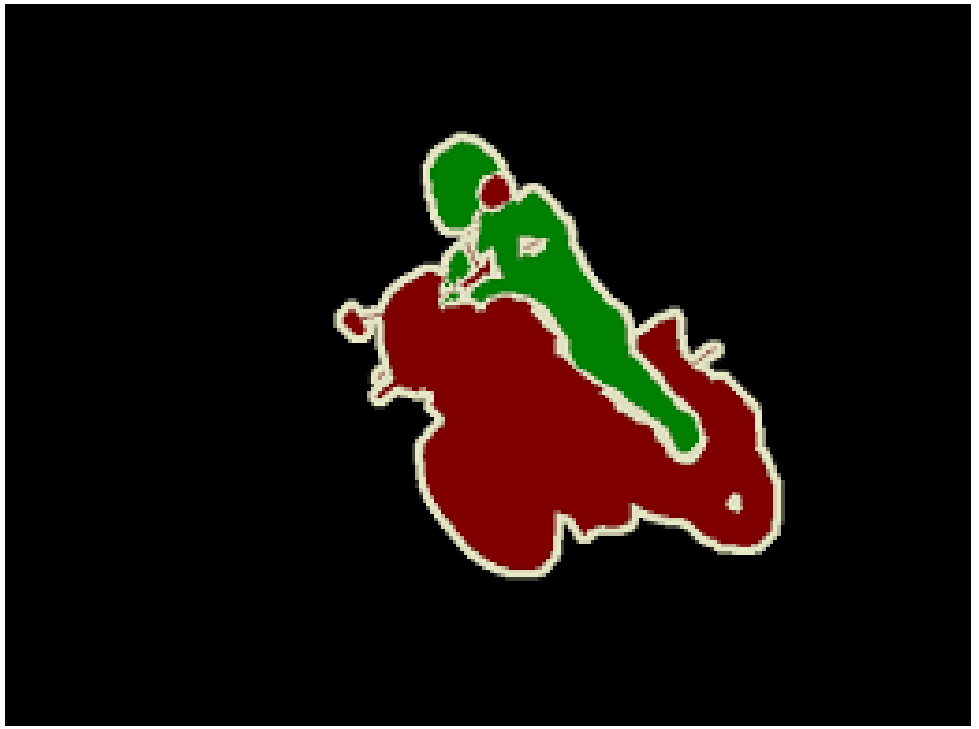
PASCAL VOC

- This dataset has been widely used as **a benchmark for object detection, semantic segmentation, and classification tasks.**
- The PASCAL Visual Object Classes (VOC) 2007, 2012 dataset contains **20 object categories** including vehicles, household, animals, and other: aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person.
- Each image in this dataset has **pixel-level segmentation annotations, bounding box annotations, and object class annotations.**
- 9963 images with 24640 objects 50% for train/val 50% for test

PASCAL VOC



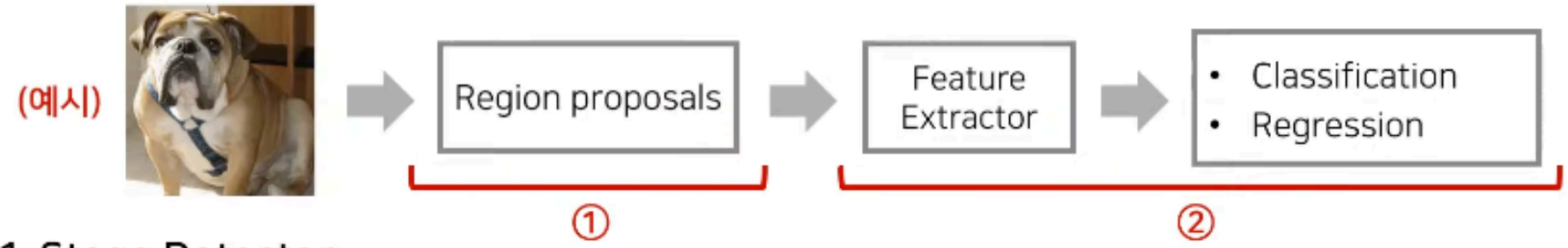
Object Detection Annotation



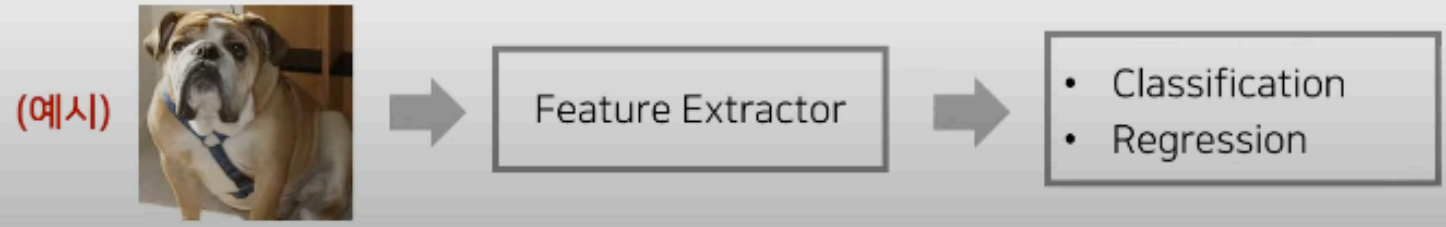
Segmentation Annotation

2-stage 방식과 1-stage 방식 비교

- 2-Stage Detector
  - 물체의 ① 위치를 찾는 문제(localization)와 ② 분류(classification) 문제를 **순차적으로** 해결합니다.

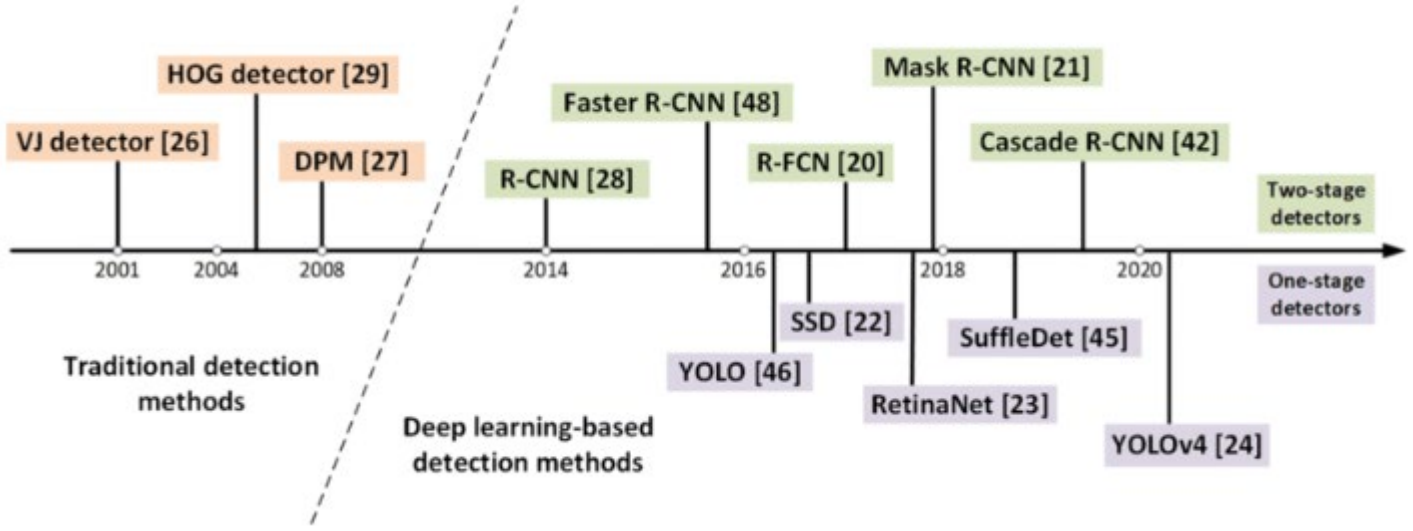


- 1-Stage Detector
  - 물체의 위치를 찾는 문제(localization)와 분류(classification) 문제를 **한 번에 해결**합니다.



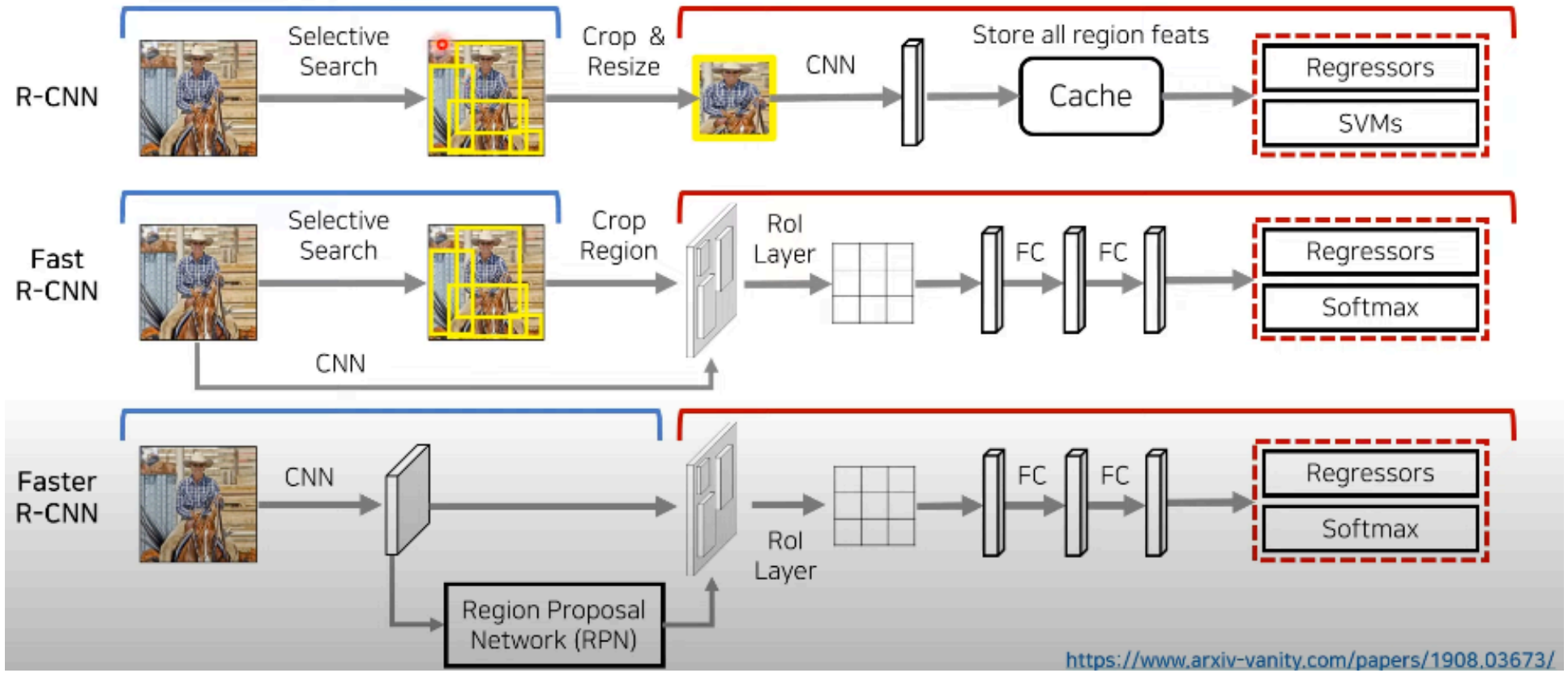
출처 : https://www.youtube.com/watch?v=jqNCdjOB15s

Object detection



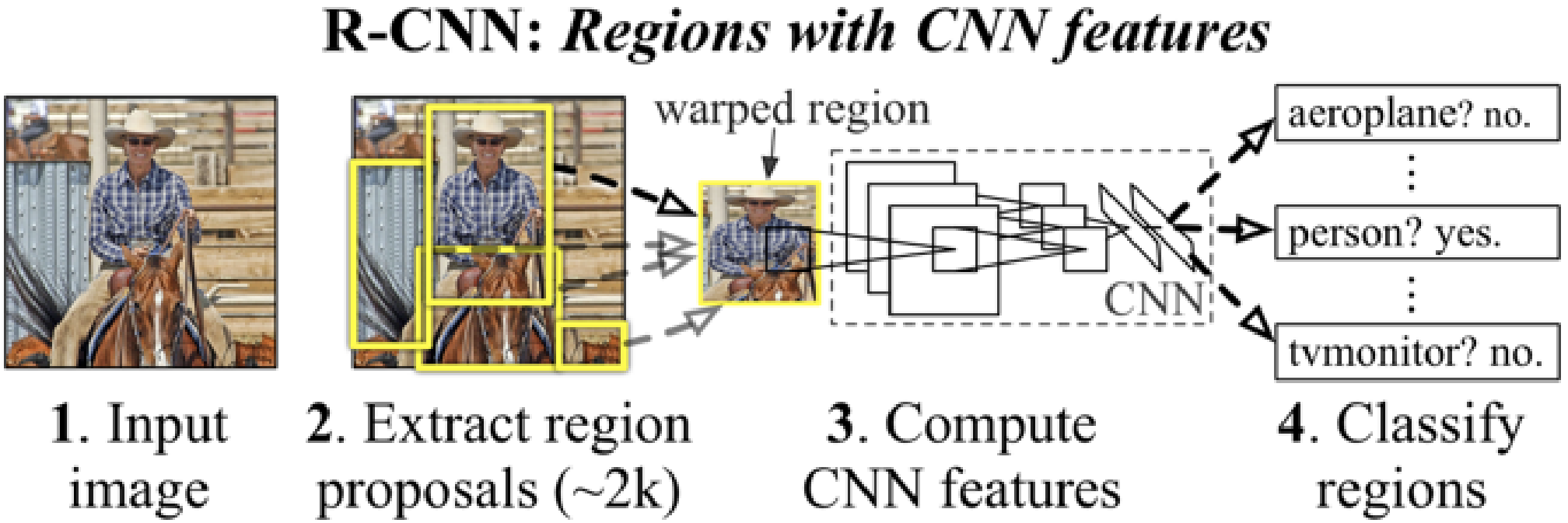
출처 : Road object detection: a comparative study of deep learning-based algorithms

2-stage object detection

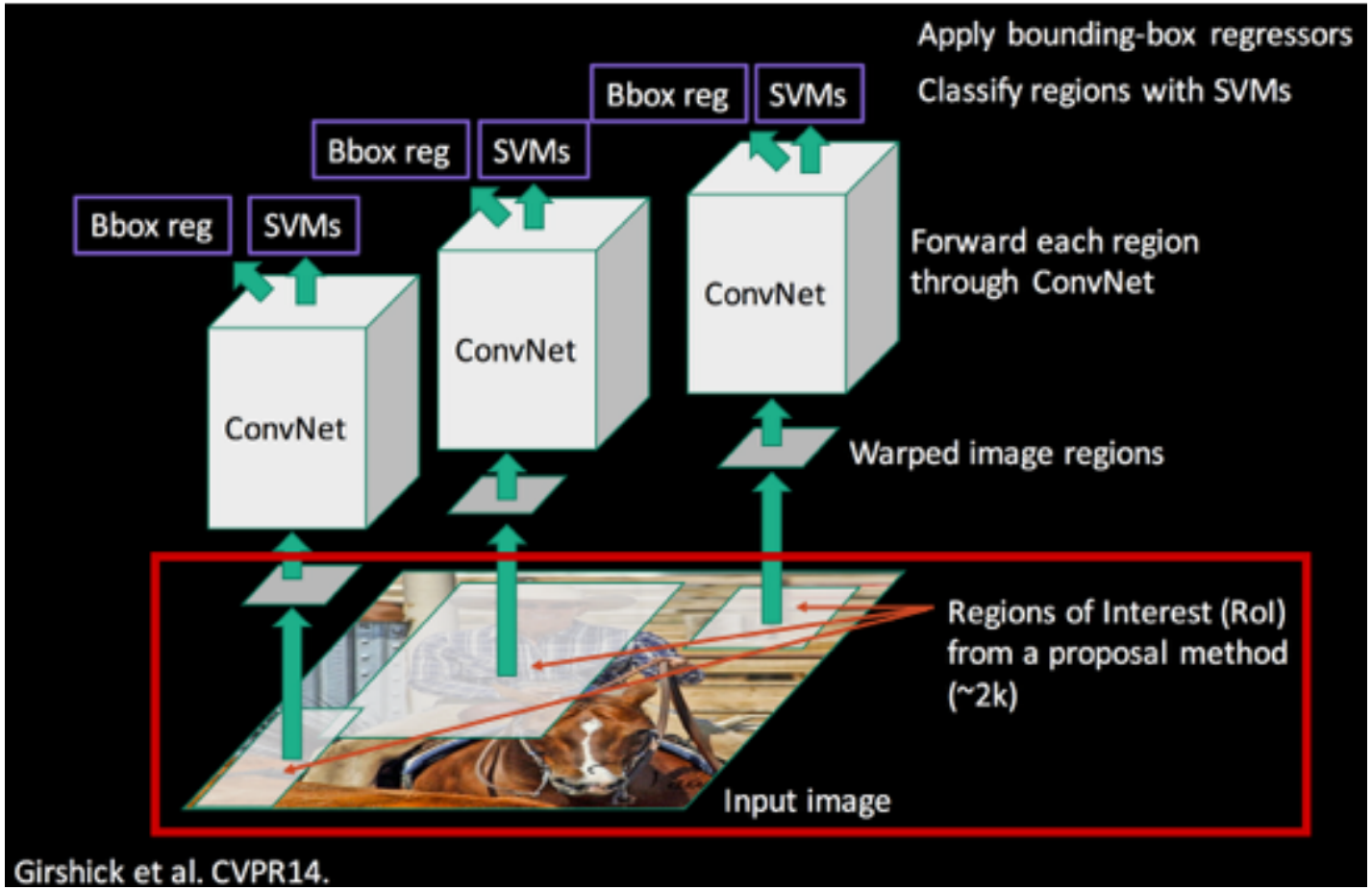


출처 : https://www.youtube.com/watch?v=jqNCdjOB15s

Region Convolutional Neural Network (RCNN)



Region Convolutional Neural Network (RCNN)

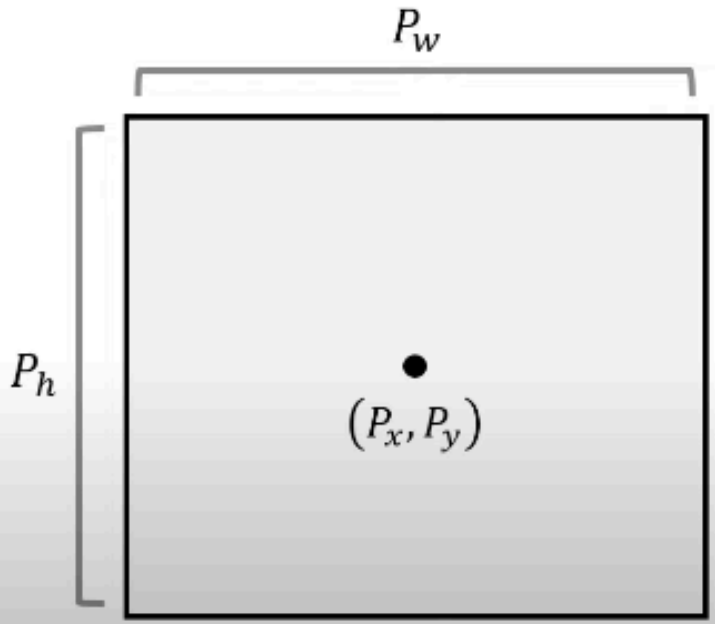


Bounding Box Regression

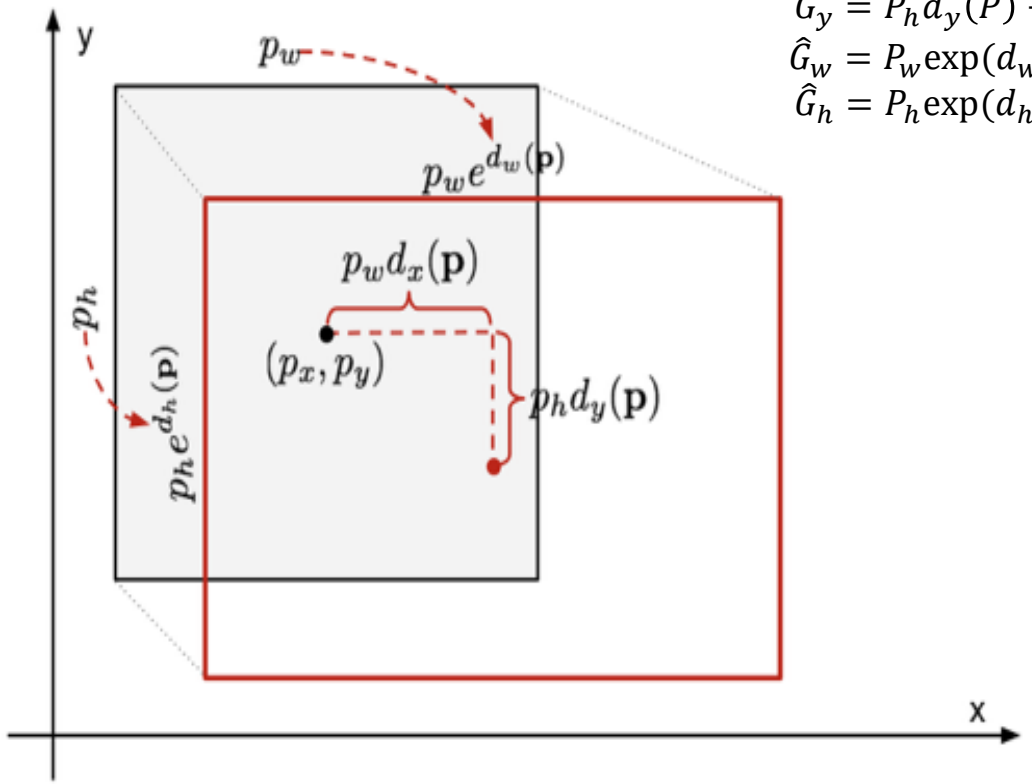
- Not exact the position of bounding boxes by Selective Search
- We need Bounding Box Regression that reduces the difference btw the Predicted Box and the Ground Truth Box
- It is a linear regression model

Bounding Box Regression

- **지역화(localization) 성능을 높이기 위해** bounding-box regression을 사용합니다.
- 학습 데이터셋: $\{(P^i, G^i)\}_{i=1,\ldots,N}$
  - **예측** 위치: $P^i = (P_x^i, P_y^i, P_w^i, P_h^i)$
  - **실제** 위치: $G^i = (G_x^i, G_y^i, G_w^i, G_h^i)$



Bounding Box Regression



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

- 학습할 4개의 파라미터(parameter)
  - $d_x(P), d_y(P), d_w(P), d_h(P)$
  - Linear regression을 진행합니다.

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

Bounding Box Regression

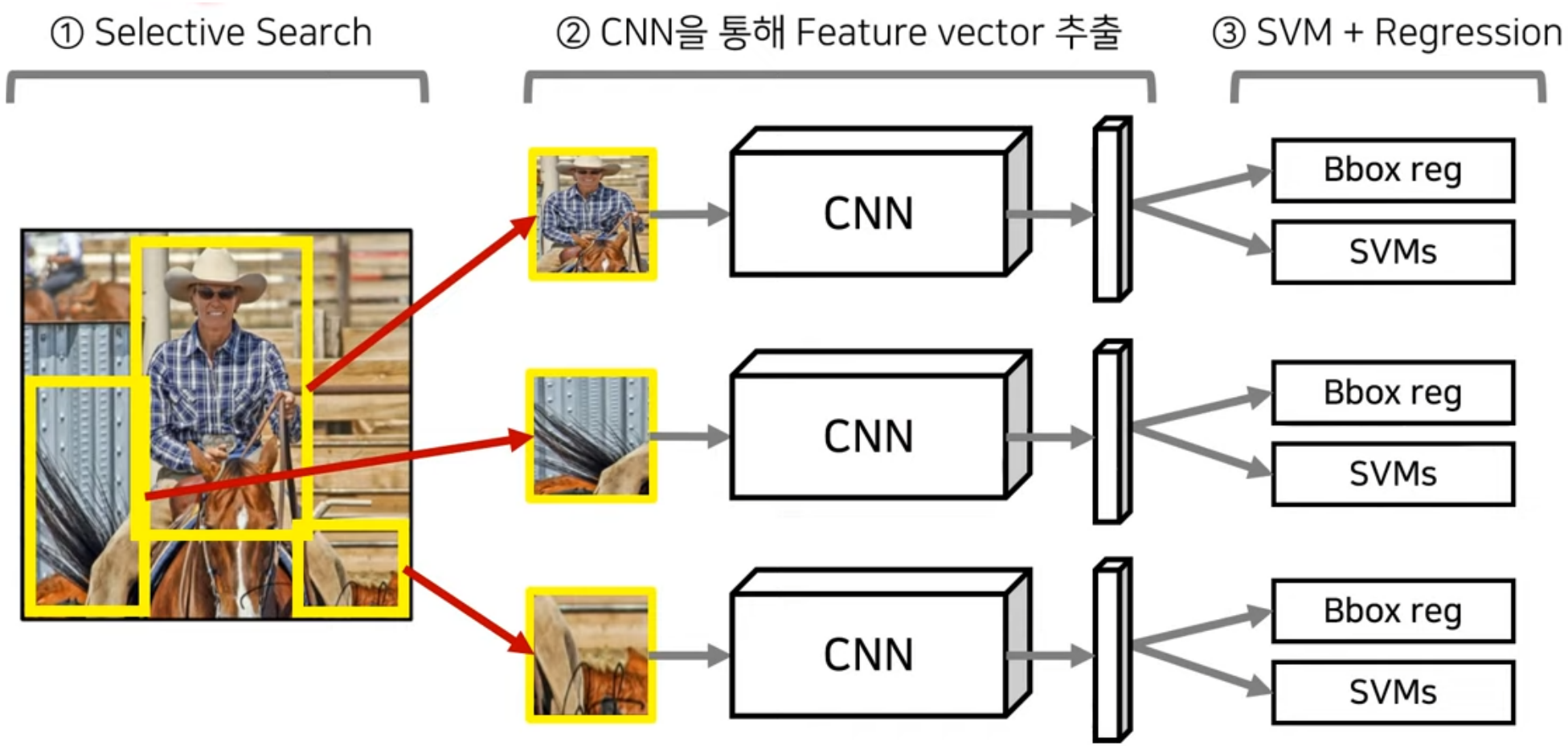
$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P))$$

Approximate $d_i(P)$ by $\boldsymbol{w}_i^T \phi_5(P)$

$$d_x = (G_x - P_x)/P_w$$
$$d_y = (G_y - P_y)/P_h$$
$$d_w = \log(G_w/P_w)$$
$$d_h = \log(G_h/P_h)$$

$$\boldsymbol{w}_i^* = \underset{\boldsymbol{w}_i}{\operatorname{argmin}} \sum_{k=1}^{N} \left(d_i^k - \boldsymbol{w}_i^\top \phi_5(P^k) \right)^2 + \lambda \|\boldsymbol{w}_i\|^2$$

Region Convolutional Neural Network(R-CNN)



출처 : https://www.youtube.com/watch?v=jqNCdjOB15s

Region Convolutional Neural Network (RCNN)

- Slow processing time
  - It needs to iterate forward propagation of input image patch over all proposals (~ 20 00 forward propagations in practice)
- Separate optimization of model components
  - Feature: CNN
  - Classifier: SVM
  - Region proposal: Selective Search Window
  - Post-processing: Bounding box regression

→ It is not desirable to the find optimal combination of all components

Problem Setting – Classification

- Image classification predicts which class among predefined categories appears in an image
- Given image X, predict single class label Y

Image classification
(what? *I don't care where*)

✓ Loss: Cross entropy loss

$$L(\boldsymbol{y}, i) = -\log\left(\frac{\exp(y_i)}{\sum_{j=1}^{C}\exp(y_j)}\right)$$

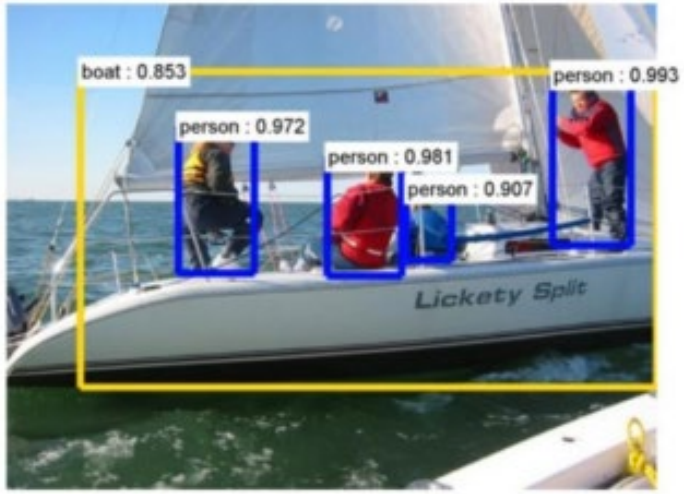
$$= -y_i + \log\left(\sum_{j=1}^{C}\exp(y_j)\right)$$

✓ Evaluation: Mean accuracy over classes

e.g. CIFAR-10, MNIST, CUB

Problem Setting – Object Detection

- Predict box form to define where predefined objects are
- Given image X, predict class labels {Y} and bounding box coordinates {(Px,Py,Ph,Pw)}



Object detection
(what + where?)

✓ Loss : Classification Loss + Regression Loss

✓ Evaluation: mAP (mean Average Precision)

Evaluation - TP, FN, FP, TN

예측 결과 (Predict Result)

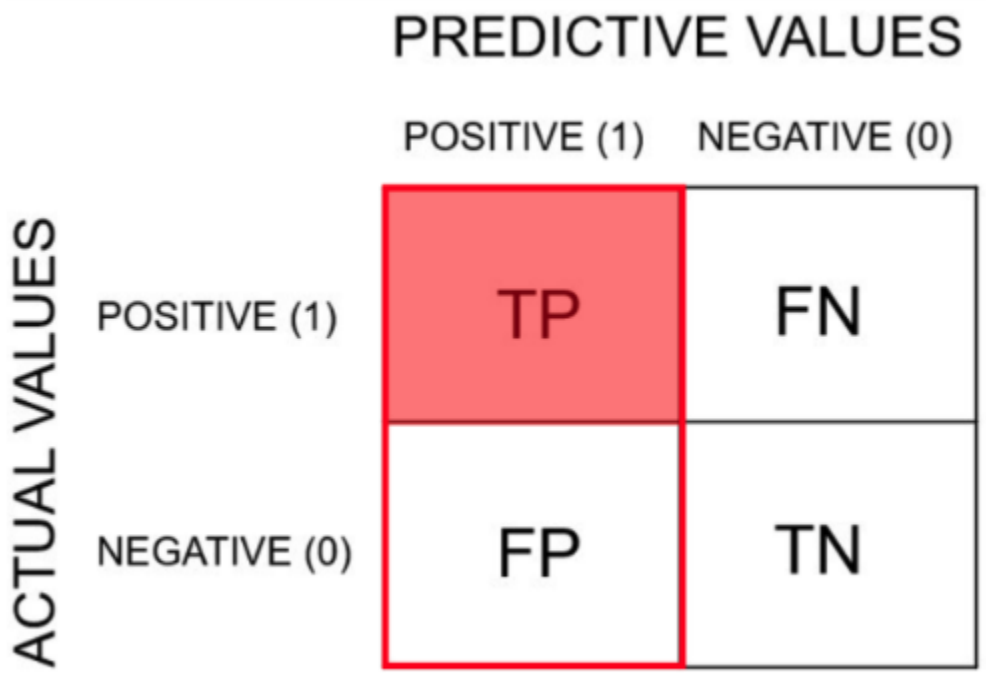
실제 정답 (Ground Truth)	Positive	Negative
Positive	TP (True Positive) 있다고 올바르게 판단	FN (False Negative) 없다고 잘못 판단
Negative	FP (False Positive) 있다고 잘못 판단	TN (True Negative) 없다고 올바르게 판단

- Precision (정확도)
  - 올바르게 탐지한 물체의 수(TP) / 모델이 탐지한 물체의 개수(TP + FP)
- Recall (재현율)
  - 올바르게 탐지한 물체의 수(TP) / 실제 정답 물체의 수(TP + FN)

출처 : https://www.youtube.com/watch?v=jqNCdjOB15s

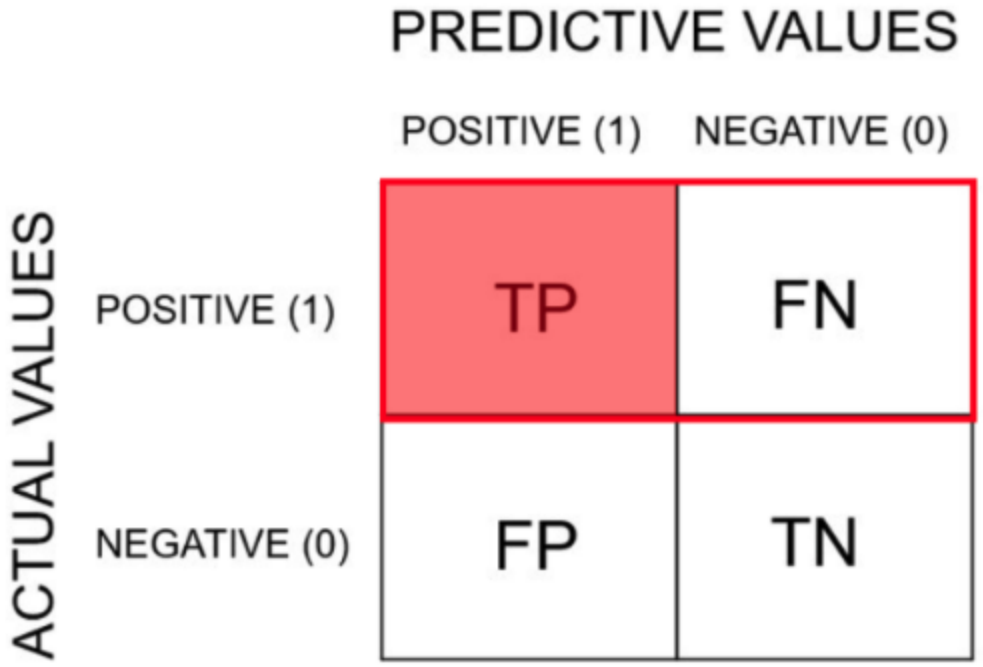
Evaluation - Precision

- 정확도
- Positive로 예측한 것들 중에 실제 Positive인 것들의 비율
- **(True positive / predicted positive)**



Evaluation – Recall

- 재현율
- 실제 Positive 중에서 positive로 제대로 예측한 비율
- **(True positive / Actual positive)**



Evaluation - Precision and Recall

- 예시 1) 강아지가 20마리 존재할 때 모델이 10마리의 강아지를 검출하여 5마리는 정확히 맞힌 경우
  - **정확도(Precision)** = 5 / 10 = 50%, **재현율(Recall)** = 5 / 20 = 25%
- 예시 2) 강아지가 10마리 존재할 때 모델이 20마리의 강아지를 검출하여 7마리는 정확히 맞힌 경우
  - **정확도(Precision)** = 7 / 20 = 35%, **재현율(Recall)** = 7 / 10 = 70%

- Precision (정확도)
  - 올바르게 탐지한 물체의 수(TP) / 모델이 탐지한 물체의 개수(TP + FP)
- Recall (재현율)
  - 올바르게 탐지한 물체의 수(TP) / 실제 정답 물체의 수(TP + FN)

- 만약 모든 영역에 대하여 전부 사물이 존재한다고 판단을 해버리면 어떤 일이 벌어질까요?
  - 재현율은 높아지지만, 정확도는 떨어지게 됩니다.
- 만약 매우 확실할 때만(confidence가 높을 때만) 사물이 존재한다고 판단하면 어떤 일이 벌어질까요?
  - 정확도는 높아지지만, 재현율은 떨어지게 됩니다.

출처 : https://www.youtube.com/watch?v=jqNCdjOB15s

Evaluation – Precision and Recall

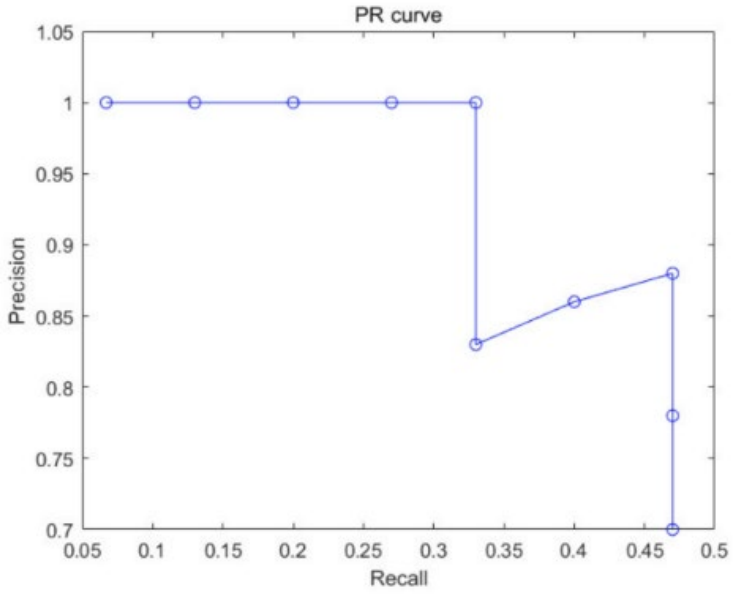
- Recall 과 Precision은 일반적으로 반비례의 경향성.
- Precision과 Recall 모두 중요한 평가 지표인데 둘 다 고려할 수 있는 방법이 없을까? → Average Precision

Evaluation - Average Precision (Precision-Recall AUC)

- Let's consider 15 objects in an image and 10 objects are predicted

Detections	confidences	TP or FP
A	57%	TP
B	78%	TP
C	43%	FP
D	85%	TP
E	91%	TP
F	13%	FP
G	45%	TP
H	68%	FP
I	95%	TP
J	81%	TP

Detections	confidences	TP or FP	누적 TP	누적 FP	Precision	Recall
I	95%	TP	1	0	1/1=1	1/15=0.067
E	91%	TP	2	0	2/2=1	2/15=0.13
D	85%	TP	3	0	3/3=1	3/15=0.2
J	81%	TP	4	0	4/4=1	4/15=0.27
B	78%	TP	5	0	5/5=1	5/15=0.33
H	68%	FP	5	1	5/6=0.83	5/15=0.33
A	57%	TP	6	1	6/7=0.86	6/15=0.4
G	45%	TP	7	1	7/8=0.88	7/15=0.47
C	43%	FP	7	2	7/9=0.78	7/15=0.47
F	13%	FP	7	3	7/10=0.7	7/15=0.47



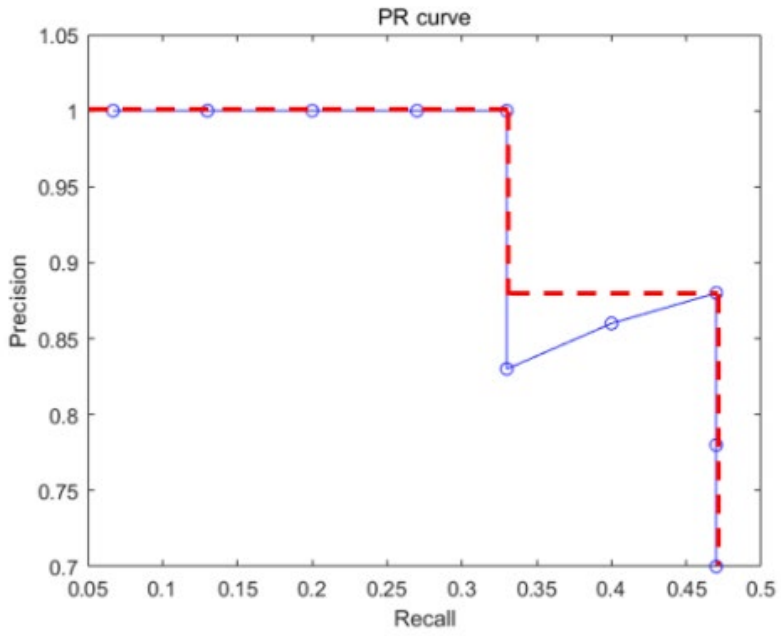
출처 : https://lapina.tistory.com/98

Evaluation – AP and mAP

- Evaluation

✓ Average precision

$$AveragePrecision = \frac{\sum_{r \in Recall([0,1])} Precision(t_r)}{|Recall([0,1])|}$$



✓ Mean Average precision

$$meanAveragePrecision = \frac{\sum_{c \in C} AveragePrecision_c}{|C|}$$

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
SS	2000	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	**65.7**	80.4	64.2
RPN	300	12	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
RPN	300	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
RPN	300	COCO+07++12	**75.9**	**87.4**	**83.6**	**76.8**	**62.9**	**59.6**	**81.9**	**82.0**	**91.3**	**54.9**	**82.6**	**59.0**	**89.0**	**85.5**	**84.7**	**84.1**	**52.2**	**78.9**	65.5	**85.4**	**70.2**

IoU (Intersection over Union)

- IoU란 **두 바운딩 박스가 겹치는 비율**을 의미합니다.
  - **성능 평가 예시**: mAP@0.5는 정답과 예측의 IoU가 50% 이상일 때 정답으로 판정하겠다는 의미입니다.
  - **NMS 계산 예시**: 같은 클래스(class)끼리 IoU가 50% 이상일 때 낮은 confidence의 box를 제거합니다.

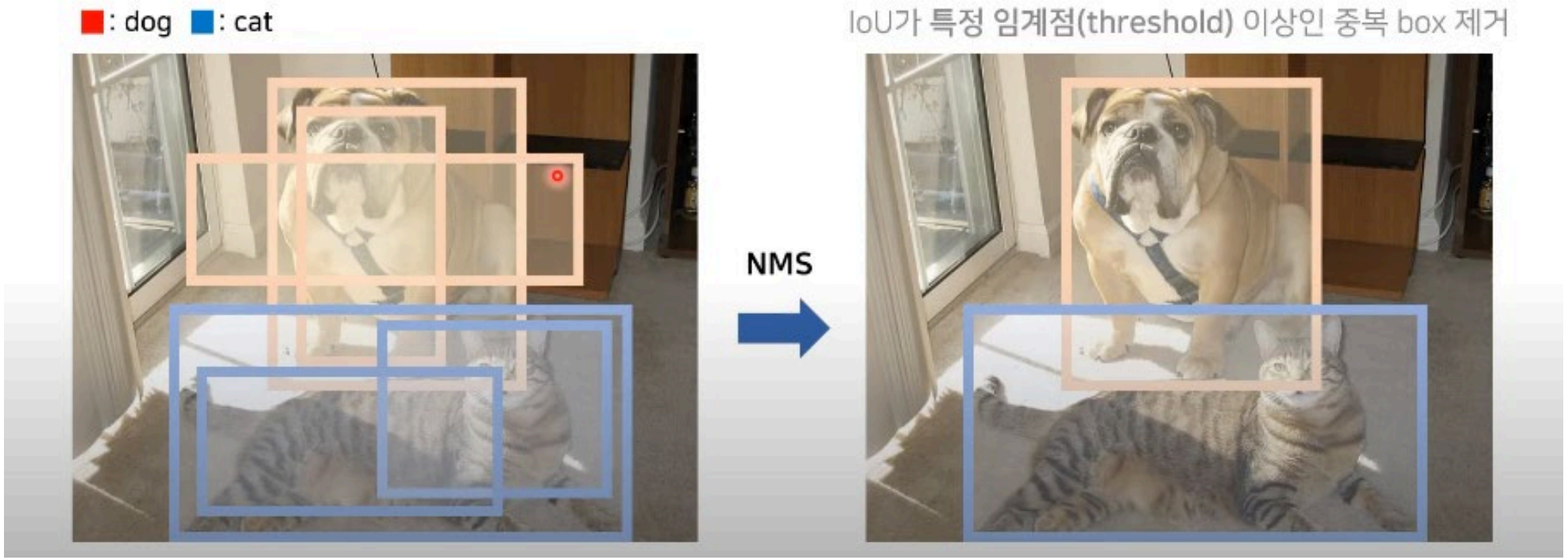
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

(교집합)

(합집합)

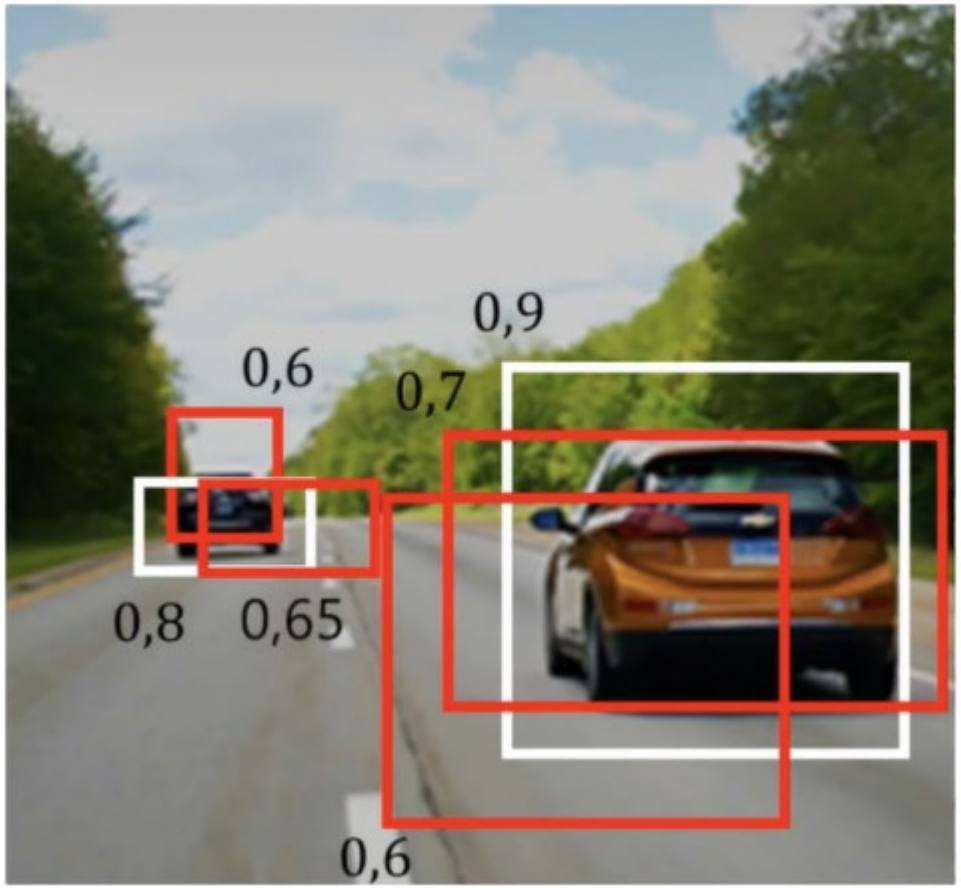
NMS(Non Maximum Suppression)

- 객체 검출(object detection)에서는 **하나의 인스턴스(instance)**에 하나의 bounding box가 적용되어야 합니다.
  - 따라서 여러 개의 bounding box가 겹쳐 있는 경우에 하나로 합치는 방법이 필요합니다.



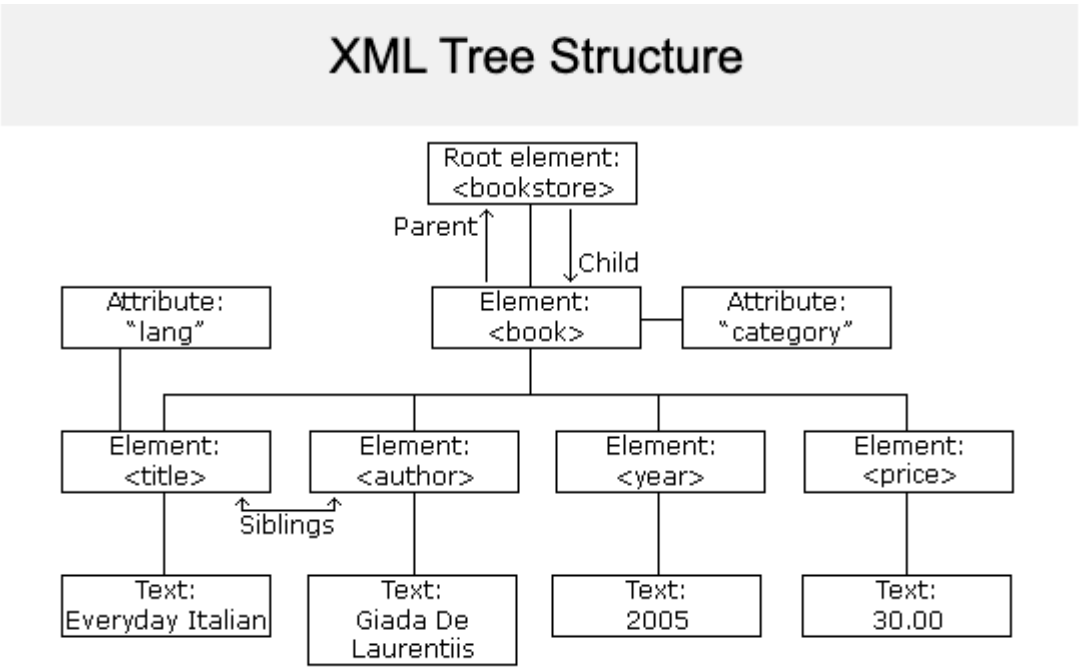
NMS(Non Maximum Suppression)

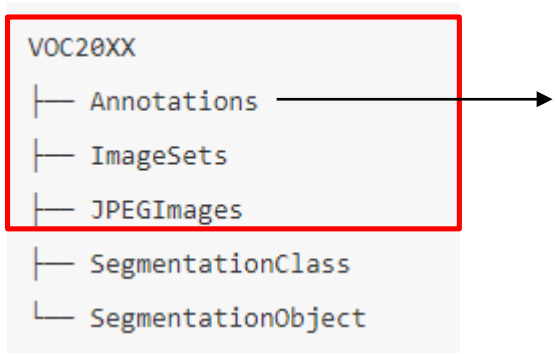
1. Remove the bounding boxes below a specific confidence score
2. Sort the remaining bounding boxes in ascending order
3. From the box with the highest confidence score, obtain the IoU with the remaining boxes and remove the boxes larger than a certain threshold



PASCAL VOC Dataset 실습

Appendix: XML data structure






```
<annotation>
  <folder>VOC2007</folder>
  <filename>000001.jpg</filename>
  <source>
    <database>The VOC2007 Database</database>
    <annotation>PASCAL VOC2007</annotation>
    <image>flickr</image>
    <flickrid>341012865</flickrid>
  </source>
  <owner>
    <flickrid>Fried Camels</flickrid>
    <name>Jinky the Fruit Bat</name>
  </owner>
  <size>
    <width>353</width>
    <height>500</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>dog</name>
    <pose>Left</pose>
    <truncated>1</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>48</xmin>
      <ymin>240</ymin>
      <xmax>195</xmax>
      <ymax>371</ymax>
    </bndbox>
  </object>
</annotation>
```

IoU, NMS 실습

Implement IoU

```python
def bbox_iou(box1, box2):
    """
    Returns the IoU of two bounding boxes
    box 1 : (1, 4) shaped pytorch tensors - sinlge GT bounding box
    box 2 : (N, 4) shaped pytorch tensors - multiple predictions from network
    """
    b1_x1, b1_y1, b1_x2, b1_y2 = box1[:,0], box1[:,1], box1[:,2], box1[:,3]
    b2_x1, b2_y1, b2_x2, b2_y2 = box2[:,0], box2[:,1], box2[:,2], box2[:,3]

    ### Intersection rectangle coordinate
    ### Return (1,4) tensor
    inter_rect_x1 = torch."""your code here"""
    inter_rect_y1 = torch."""your code here"""
    inter_rect_x2 = torch."""your code here"""
    inter_rect_y2 = torch."""your code here"""

    ### Practice
    ### Clamps all elements in input into the range [min, max]
    inter_area = torch.clamp("""your code here""", min=0.)\
            * torch.clamp("""your code here""", min=0.)


    ### Calculate IoU
    area_1 = """your code here"""
    area_2 = """your code here"""
    iou = """your code here"""

    return iou
```

Implement NMS

```python
def nms(boxes, scores, threshold):
    """
    boxes: (N, 4), each row-> (x1, y1, x2, y2), x2 > x1, y2 > y1
    scores: (N,), each value in [0, 1]
    threshold: iou threshold
    """
    x1 = boxes[:, 0]
    y1 = boxes[:, 1]
    x2 = boxes[:, 2]
    y2 = boxes[:, 3]

    areas = (x2 - x1) * (y2 - y1)
    order = scores.argsort() ### Sort in ascending order and save the indices

    keep = []

    while len(order) > 0:
        ### The index with the highest score.
        idx = """your code here"""
        ### Keep it
        keep.append(boxes[idx])
        ### remove the index from the candidate list.
        order = """your code here"""

        if len(order) == 0:
            break ### Get out of while loop

        ### Order bounding boxes according to the order.
        xx1 = x1[order]
        xx2 = x2[order]
        yy1 = y1[order]
        yy2 = y2[order]

        ### Intersection coordinate
        ix1 = torch."""your code here"""
        iy1 = torch."""your code here"""
        ix2 = torch."""your code here"""
        iy2 = torch."""your code here"""

        ### Width and height of the intesection, calculate intersection
        w = torch.clamp("""your code here""", min=0.)
        h = torch.clamp("""your code here""", min=0.)

        inter = """your code here"""
        union = """your code here"""
        iou = """your code here"""

        ### Keep the boxes with iou less than threshold
        order = """your code here"""

    return keep
```