Object Detection:

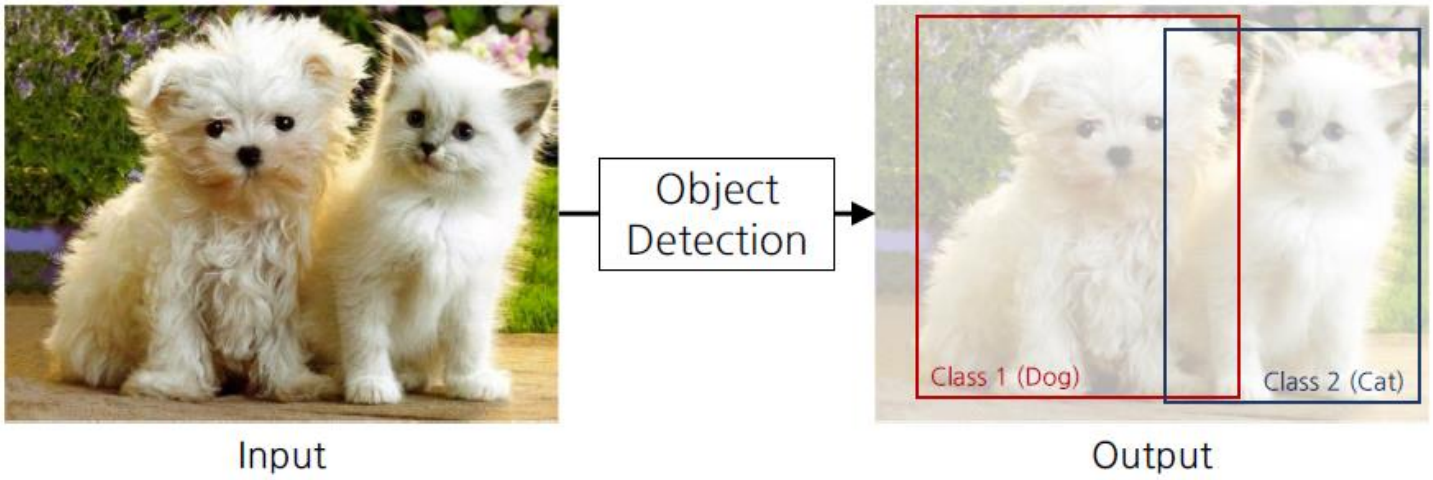
'1-Stage Detector' 소개

(R-CNN, Fast R-CNN, Faster R-CNN을 중심으로)

자료출처: Fastcampus 등

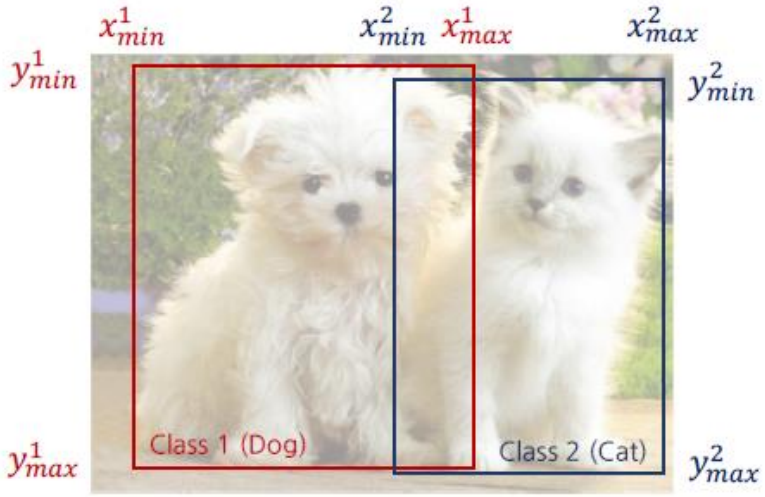
Object Detection이란?

- 이미지에서 물체의 위치(bounding box)와 종류(class index)를 알아내는 문제
  - Object detection = Regression + Classification



Input & Output

- Input
  - Image (H×W×C)



물체에 외접하는 직사각형

i 번째 class

- Outputs
  - Bounding boxes (N_{obj}×4)
  - Class indices (N_{obj}×1 or N)

bounding_boxes = [

$[x^1_{min}, y^1_{min}, x^1_{max}, y^1_{max}]$,

$[x^2_{min}, y^2_{min}, x^2_{max}, y^2_{max}]$

]

→ (N_{obj}, 4) dim

class_indices = [

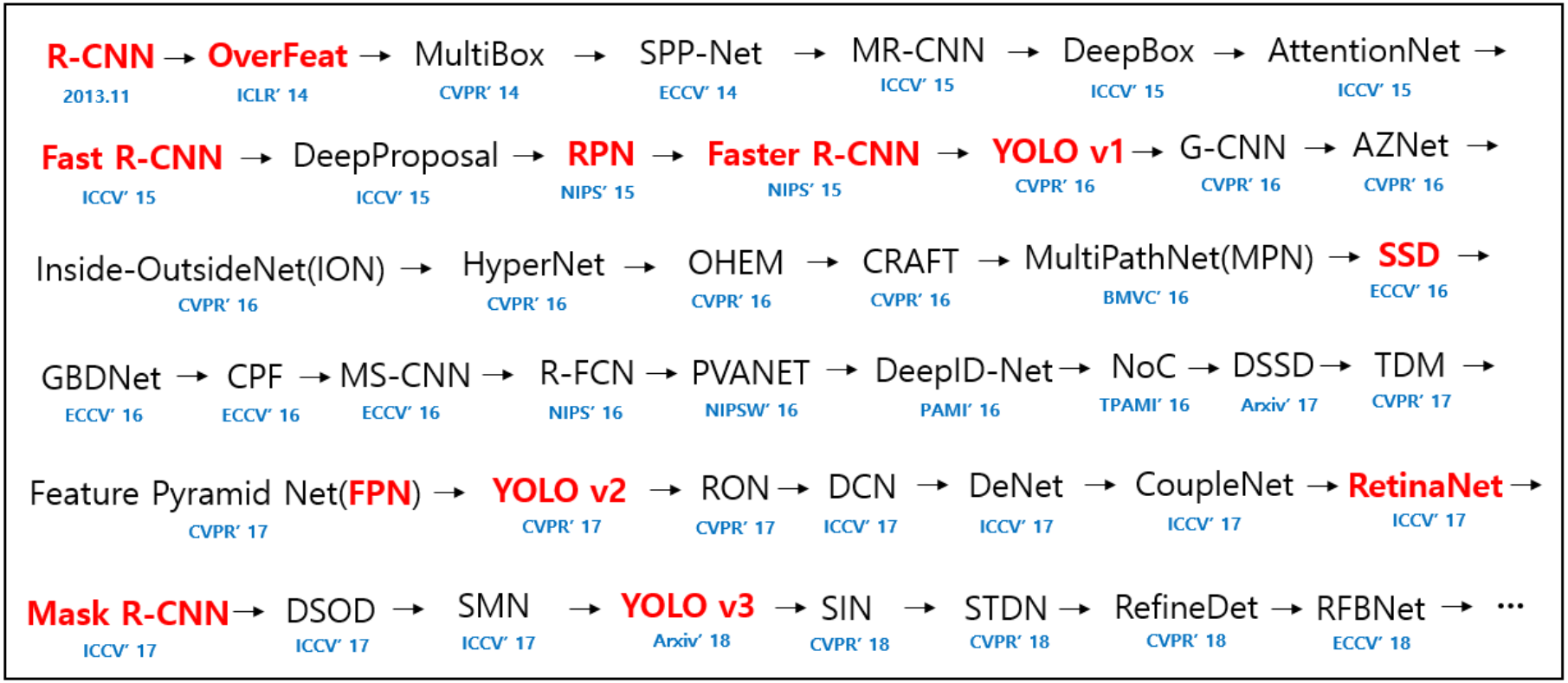
[1],

[2]

]

→ (N_{obj}, 1) dim

Paper List - since ‘2014



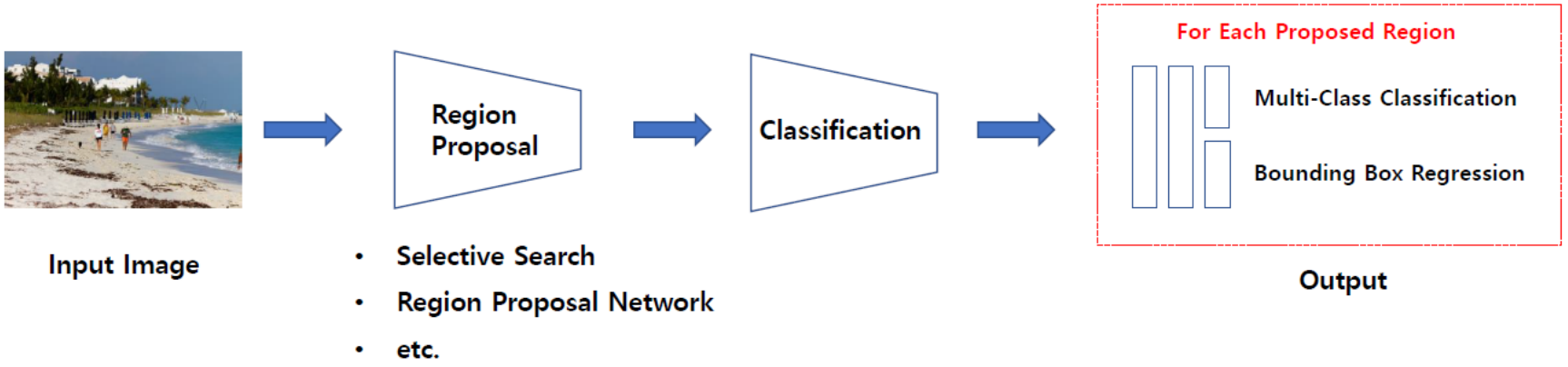
2-stage: R-CNN, SPP-NET, Fast R-CNN, Faster R-CNN

1-stage: Yolo v1,2,3, SSD, RetinaNet

source: https://github.com/hoya012/deep_learning_object_detection

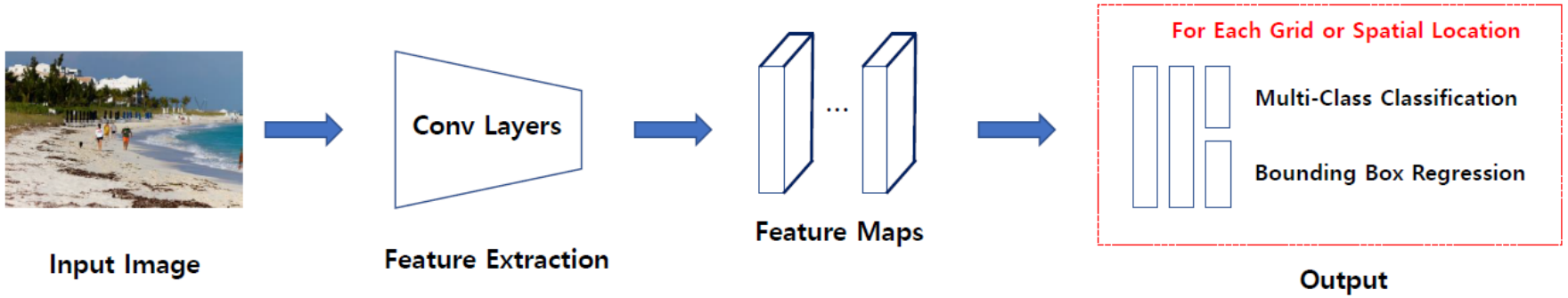
2-stage vs. 1-stage

- 2-stage approaches
  - 대표적으로 R-CNN 계열의 접근 방식
  - 탐색 영역을 찾는 과정(Region Proposal), 해당 영역을 분류(Classification)하는 과정을 **순차적으로 수행**
  - 느리지만 정확한 방법 → 높은 정확도를 요구하는 task에 사용 (ex, challenge, high-end PC environment)

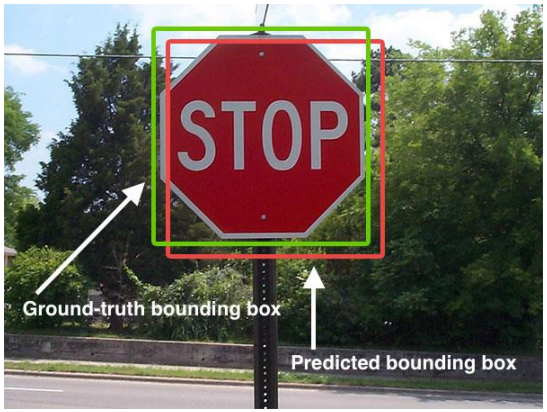


2-stage vs. 1-stage

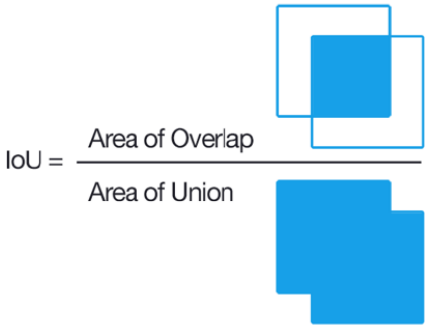
- 1-stage approaches
  - 대표적으로 SSD, YOLO 계열의 접근 방식
  - 탐색 영역을 찾는 과정(Region Proposal) + 해당 영역을 분류(Classification)하는 과정 **동시에 수행**
  - 2-stage에 비해 부정확지만 빠른 방법 → 빠른 연산속도를 요구하는 task에 사용 (ex, embedded device)



Background: Metric#1 - IoU



IoU(Intersection Over Union)

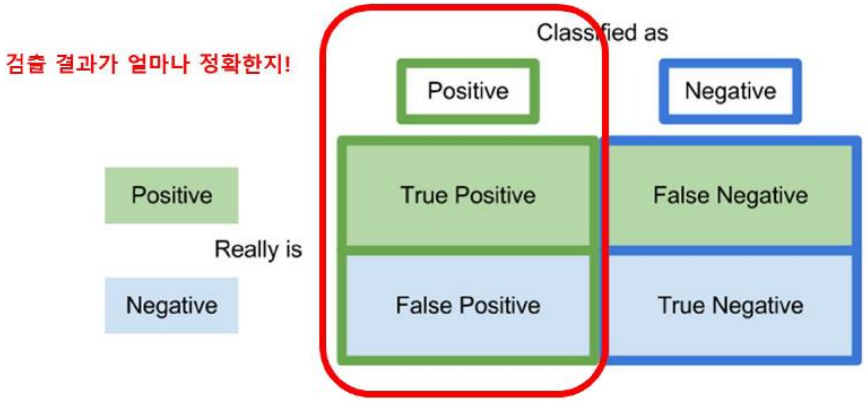


[IoU threshold]

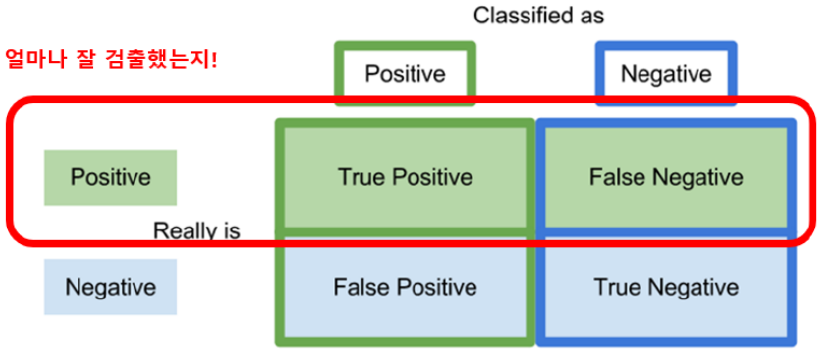
- **VOC: 0.5**
- **ImageNet: min (0.5, (w*h/((w+10)*(h+10)))**
- **MS COCO: mAP@[.5 : .05 : .95]**
  average mAP over different IoU thresholds, from 0.5 to 0.95, step 0.05 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95)

Background: Metric#2 - mAP

Precision = TP / (TP + FP)



Recall = TP / (TP + FN)



ex)

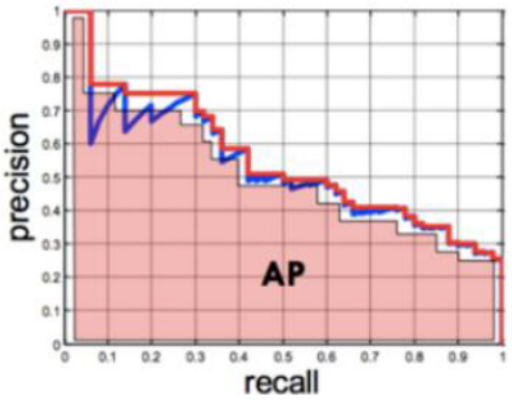
$$Precision = \frac{TP}{\text{total positive results}}$$

$$Recall = \frac{TP}{\text{total cancer cases}}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

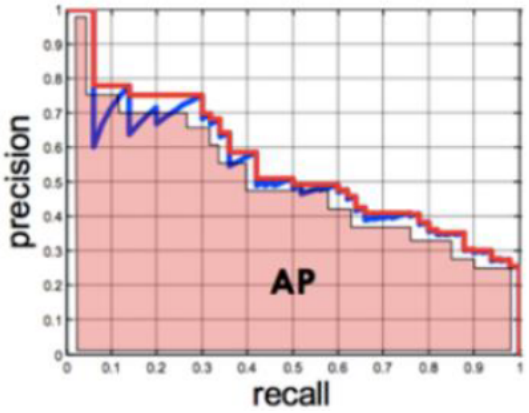
Background: Metric#2 - mAP

AP(Average Precision)



- Recall을 0, 0.1, .., 1로 바꾸었을 때의 Precision 값들의 평균
- 하나의 Class마다 하나의 AP 값을 계산 가능

mAP(Mean Average Precision)

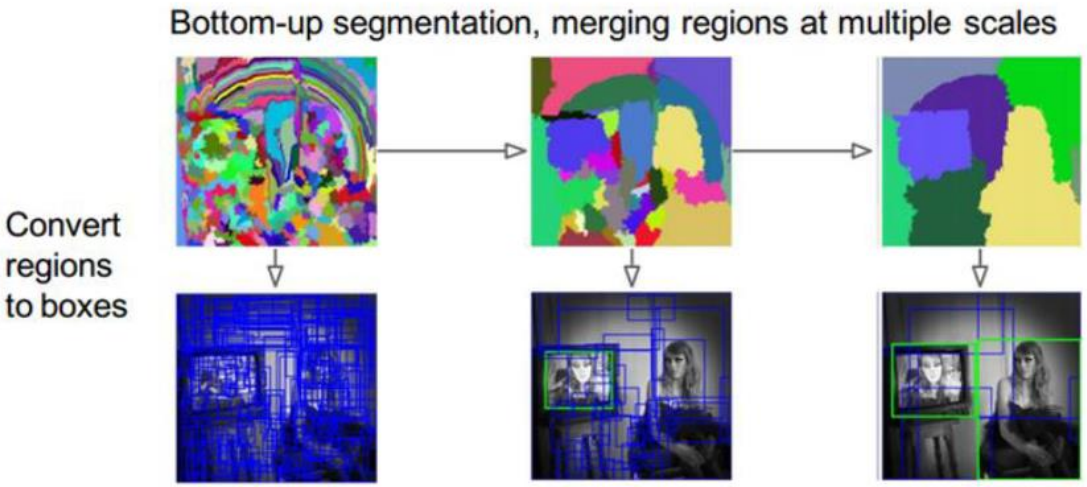


- 각 class마다 구한 AP 값들의 평균을 mAP 지표로 사용
- 대부분 challenge, 논문에서 주로 사용하는 metric

mAP@.75 means the mAP with IoU=0.75.

Background: 전통적인 Detection 알고리즘

- Sliding Window
- **<u>Selective Search</u>: DL 이전 방법 중 가장 우수**
  - 영상의 계층적 구조를 활용하여 영역 탐색 및 그룹화
  - Sliding Window 방식에 비해 빠른 속도
  - 추후 R-CNN, SPP-Net 등에 사용됨



Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$
  $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$
  Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$
  Calculate similarity set S_t between r_t and its neighbours
  $S = S \cup S_t$
  $R = R \cup r_t$

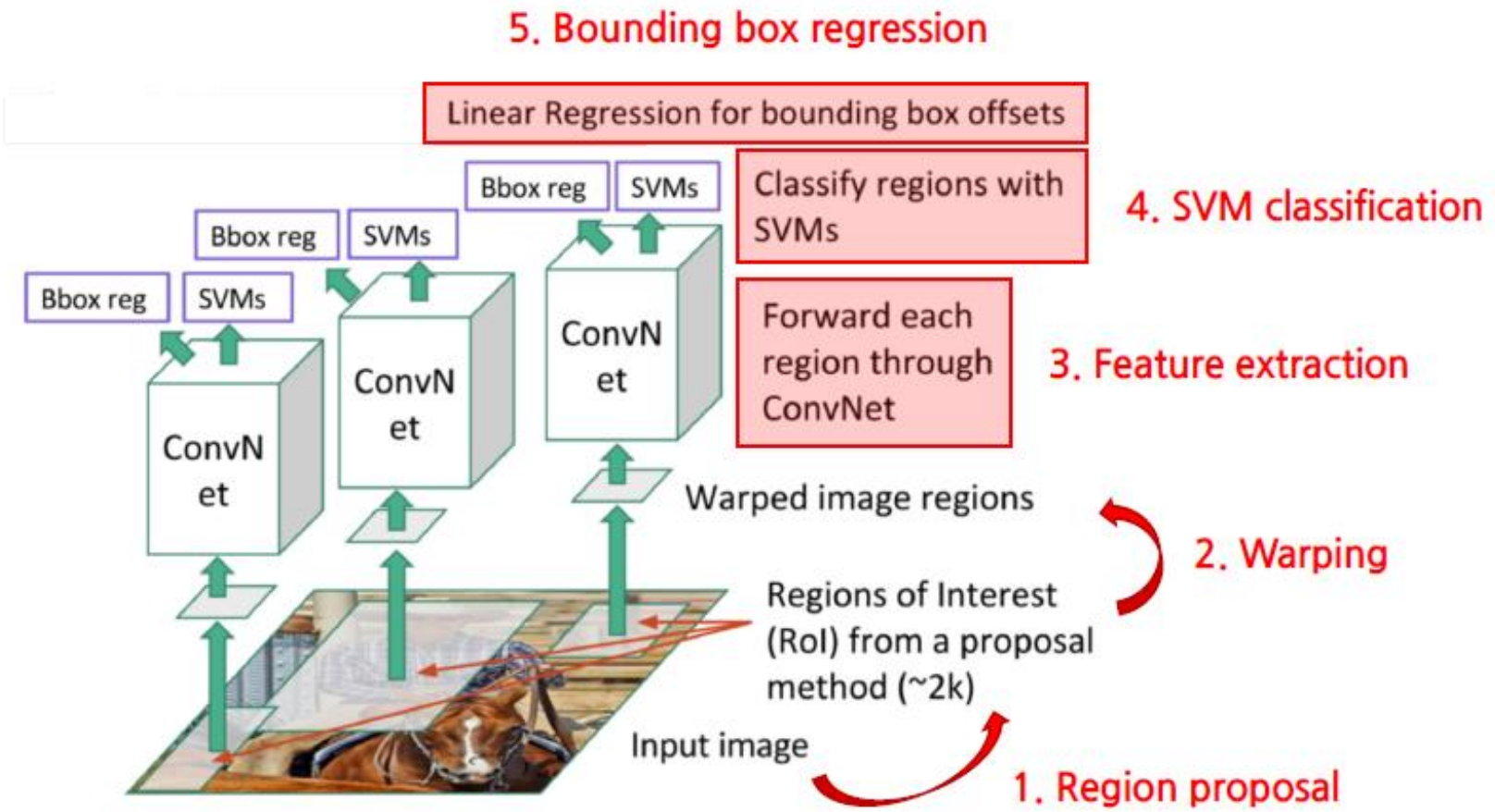
Extract object location boxes L from all regions in R

R-CNN, Region-based CNN (CVPR 2014)

Key Idea

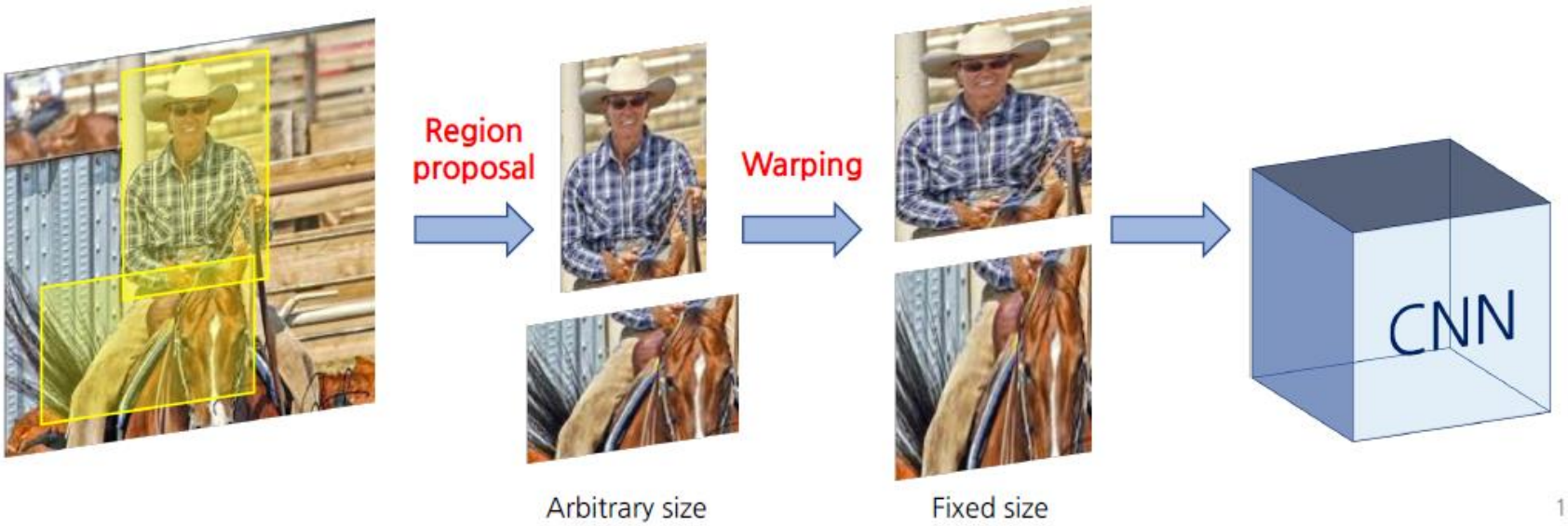
- Region proposal로 찾아낸 영역에 대해 CNN을 적용한다.
  - 본 논문에서는 Selective search(bottom-up region proposal)를 이용함.
    - 다른 region proposal 기법을 이용할 수도 있음.
  - CNN은 classification에 뛰어난 성능을 보임은 이미 잘 알려짐.
    - AlexNet, 2012
- Classification dataset을 이용한 pretraining으로 detection 성능을 높인다.
  - ImageNet classification(~10M) ≫ VOC object detection(~10K)
  - Pretrain on ImageNet → Fine-tune on target detection dataset.

R-CNN: Overview



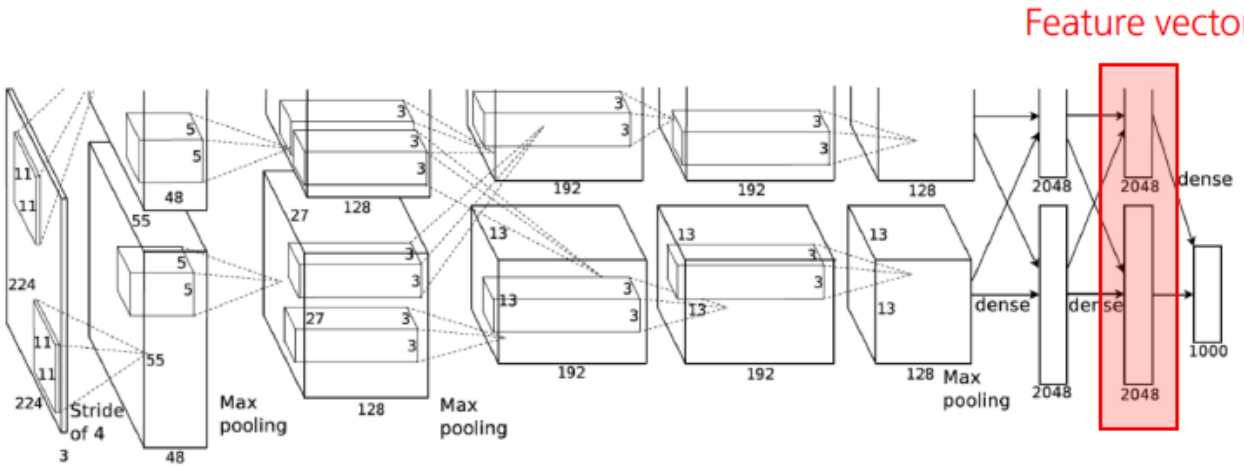
R-CNN: Warping

- CNN의 입력으로 넣기 위해 region proposal을 일정한 크기와 비율로 변형
  - Crop & Resize



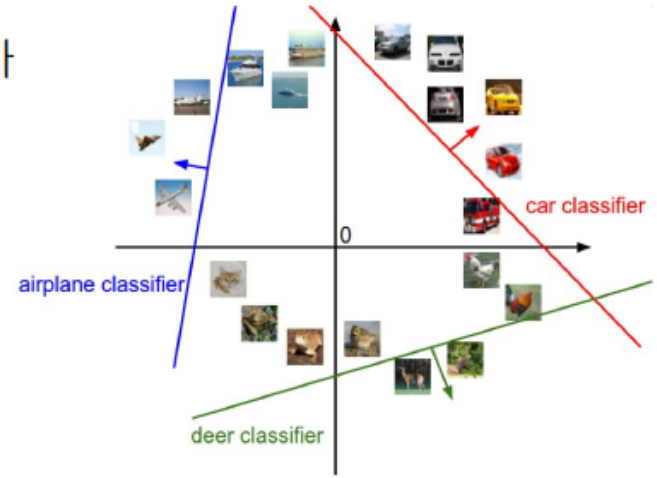
R-CNN: Feature Extraction

- Architecture: AlexNet (VGGNet으로도 실험)
- Classification을 위한 마지막 레이어 직전의 4096 size vector를 feature로 이용



R-CNN: SVM Classification

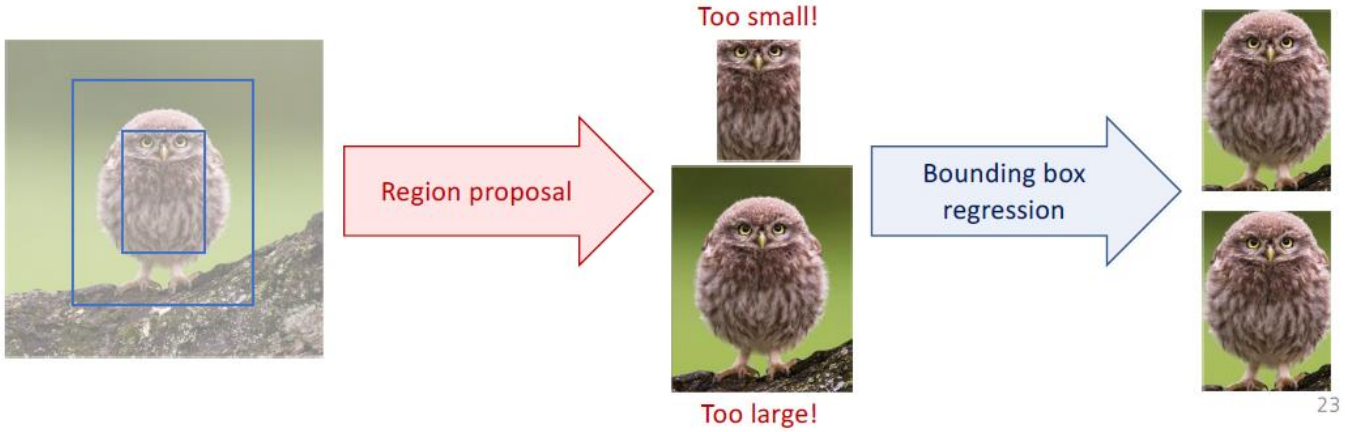
- CNN feature space에서 support vector를 계산
- N+1개의 **one-vs-rest linear SVMs** 이용
  - N개의 클래스에 “background”도 하나의 클래스로 추가
  - N+1 클래스의 multinomial classification을 수행



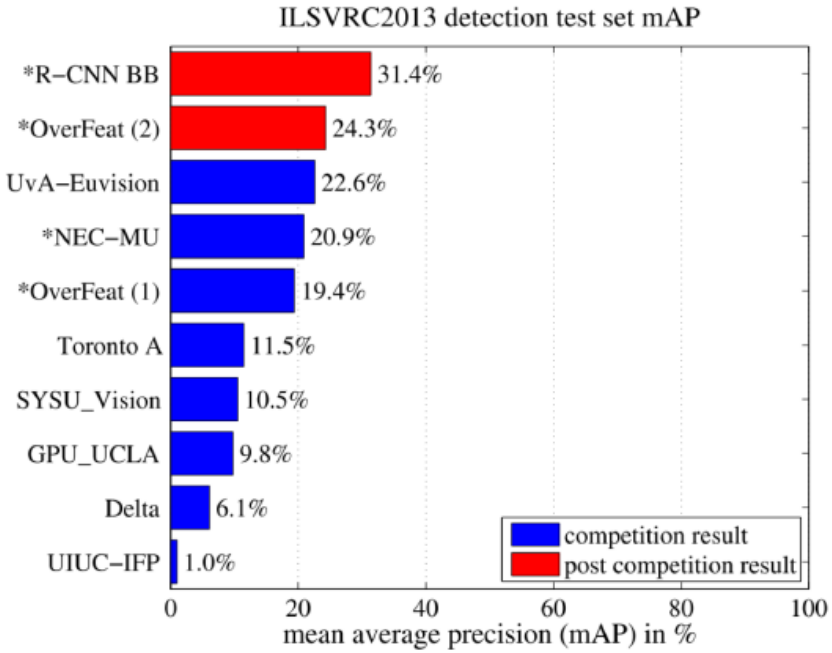
R-CNN: Bounding Box Regression

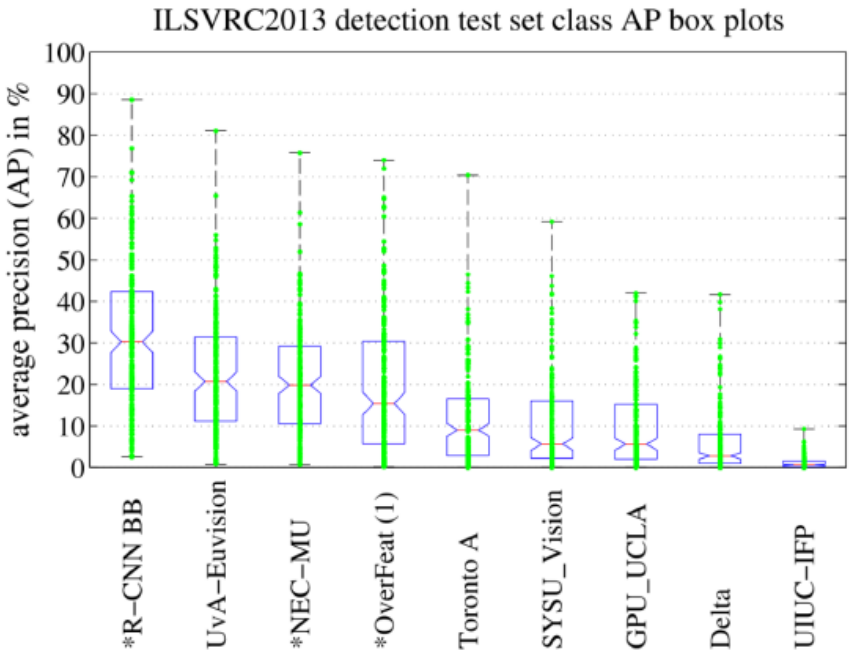
- Selective search로 얻은 region proposal의 정확도를 높이기 위해 다시 한 번 위치 및 크기를 조정하는 작업
- N개의 **linear regressors** 이용
  - Class-specific regression: 각 클래스마다 독립적으로 예측함.
  - Goal: Proposed box → Ground-truth box
    (P_x, P_y, P_w, P_h) (G_x, G_y, G_w, G_h)

x, y : box 중심점의 좌표
h, w : box의 너비와 높이



R-CNN: Performance – ILSVRC 2013



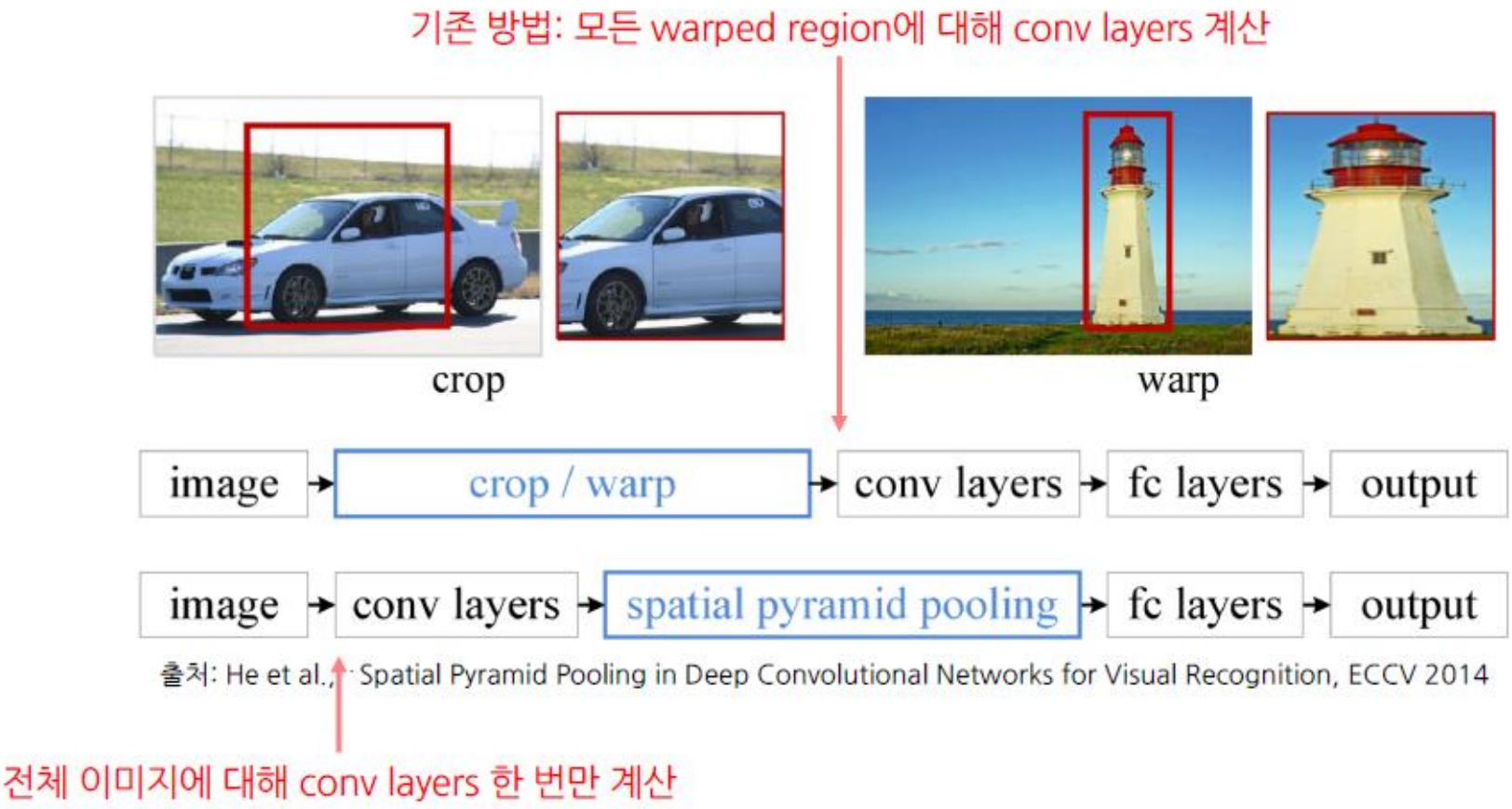


R-CNN: 단점

- Too slow Training/Inference
  - Training: 84h
  - Inference: 13s/image on a GPU, 53s/image on a CPU
  - 모든 region proposal에 대해 CNN feature extraction 수행하는 것이 가장 큰 문제
- Separated training process
  - CNN 학습 → feature를 모두 저장한 상태로 SVM 학습, bounding box regression 학습
  - SVM classification과 bounding box regression이 post-hoc 방식으로 진행
    - CNN feature extractor의 parameter는 고정된 상태로 학습되지 않음.

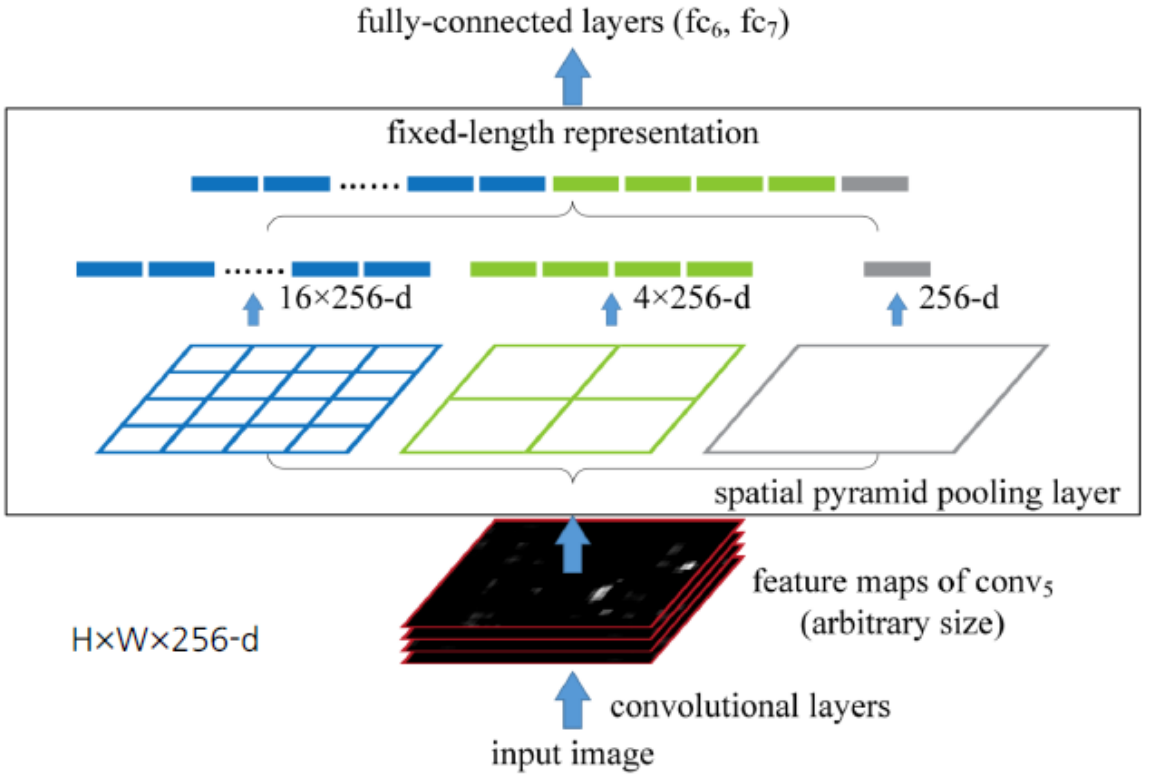
SPP-Net (ECCV 2014)

Key Idea



SPP-Net: Spatial Pyramid Pooling

임의 크기의 feature map에서
고정 크기의 representation을 추출



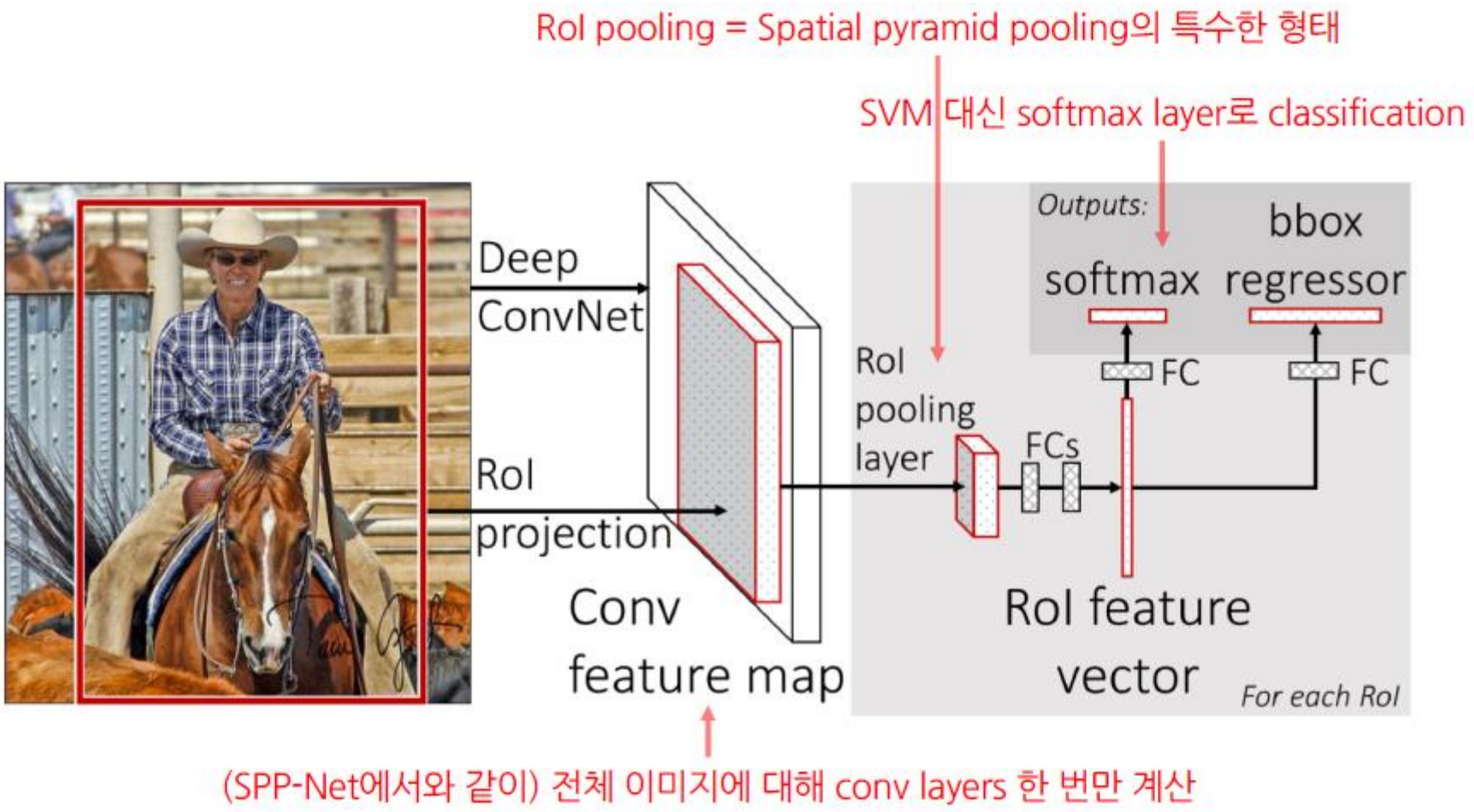
Fast R-CNN(ICCV 2015): R-CNN의 속도 개선

Key Idea

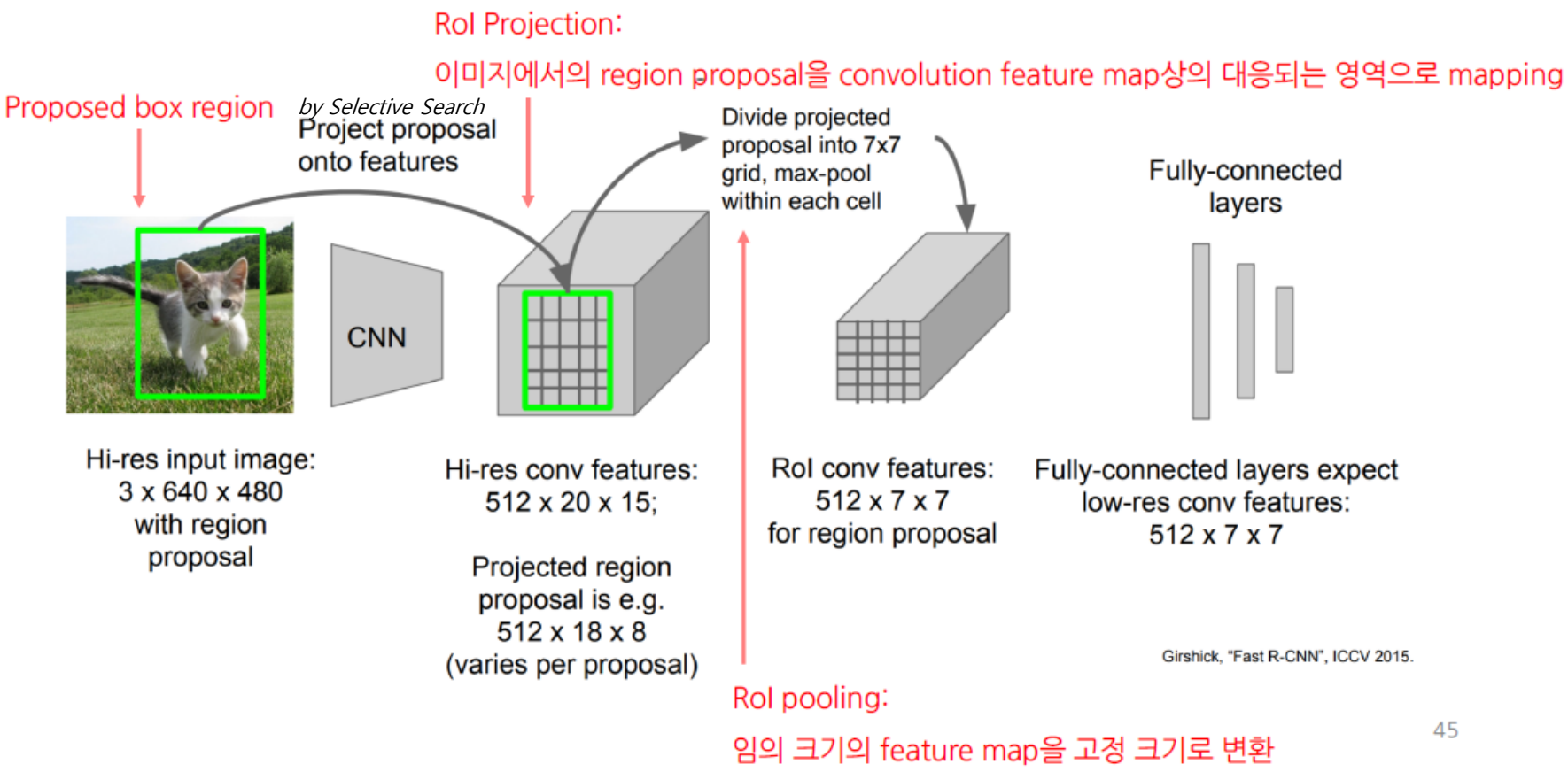
- R-CNN & SPP-Net
  - 학습이 1. softmax classification(pretraining), 2. SVM classification, 3. bounding box regression의 3단계로 진행됨.
- Multi-task loss 구성
  - Pretraining 이후 2와 3의 과정이 동시에 진행되게 함.
- Hierarchical sampling & CNN sharing
  - Network의 모든 부분에 대해 in-memory batch update 가능

End-to-end learning
(Post-hoc X)

Fast R-CNN: Overview

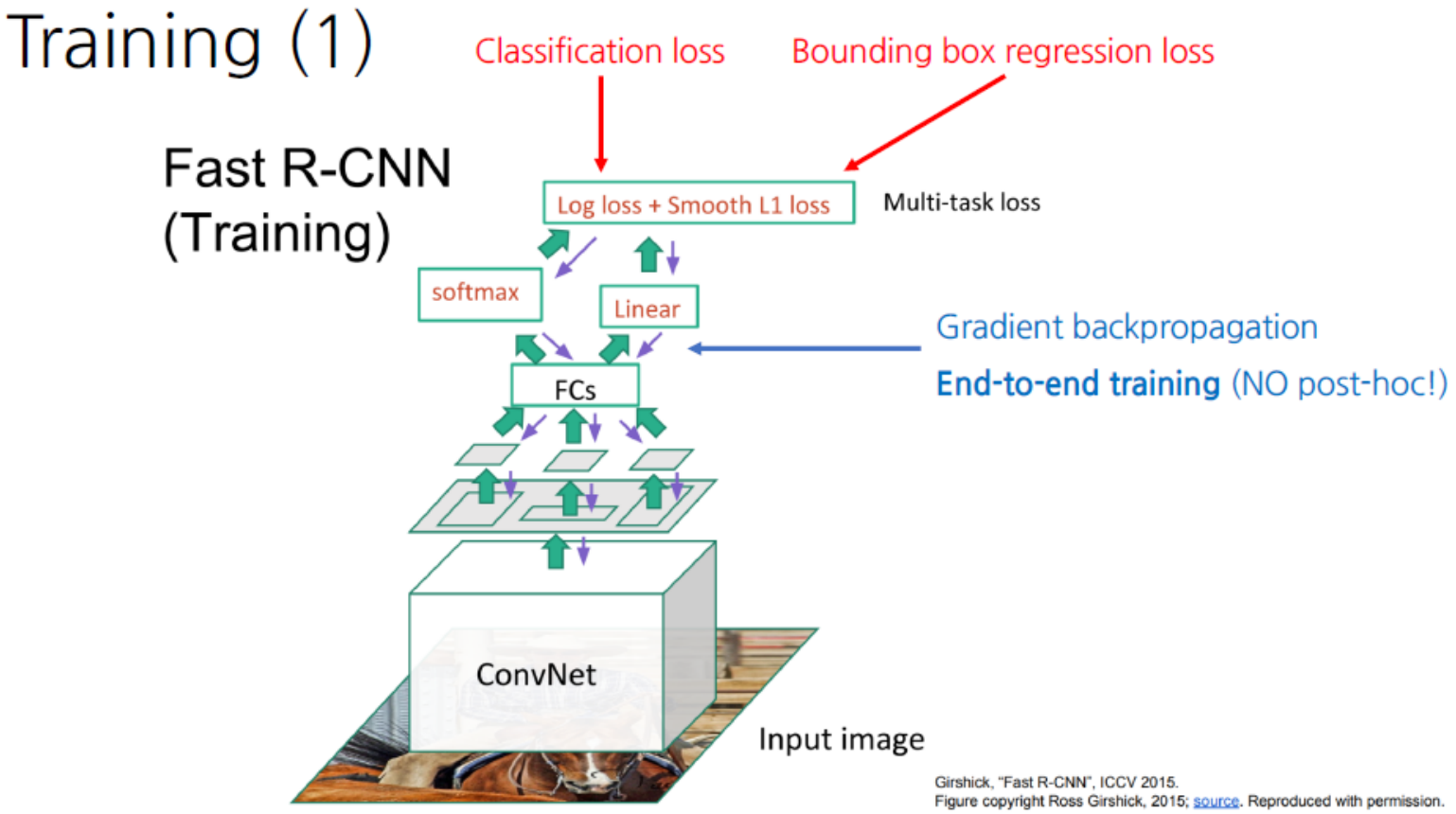


Fast R-CNN: RoI Pooling

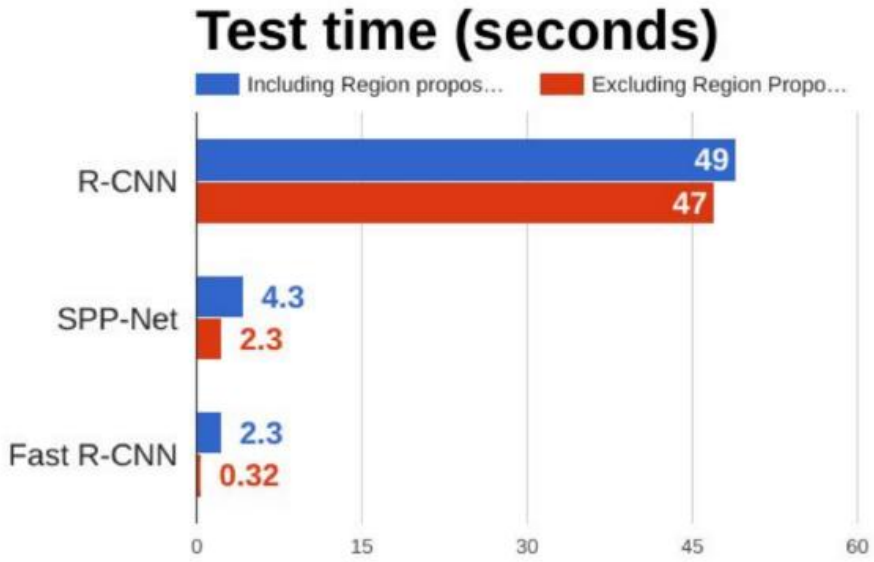


45

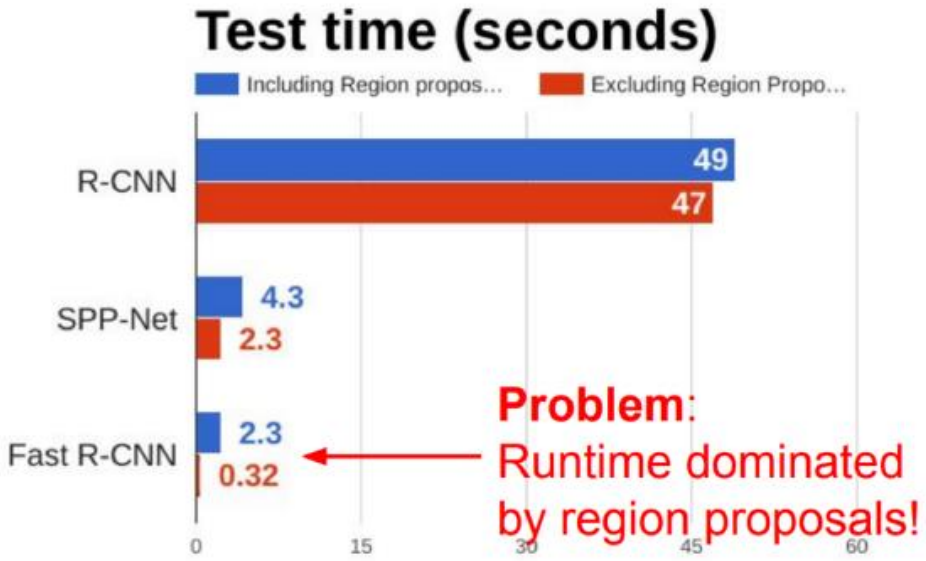
Fast R-CNN: Training



Fast R-CNN: Performance



Fast R-CNN: need a real-time system!



Faster R-CNN

Key Idea

- Selective search 대신 CNN을 이용해서 region proposal을 수행하자!

  → Region proposal network (RPN)

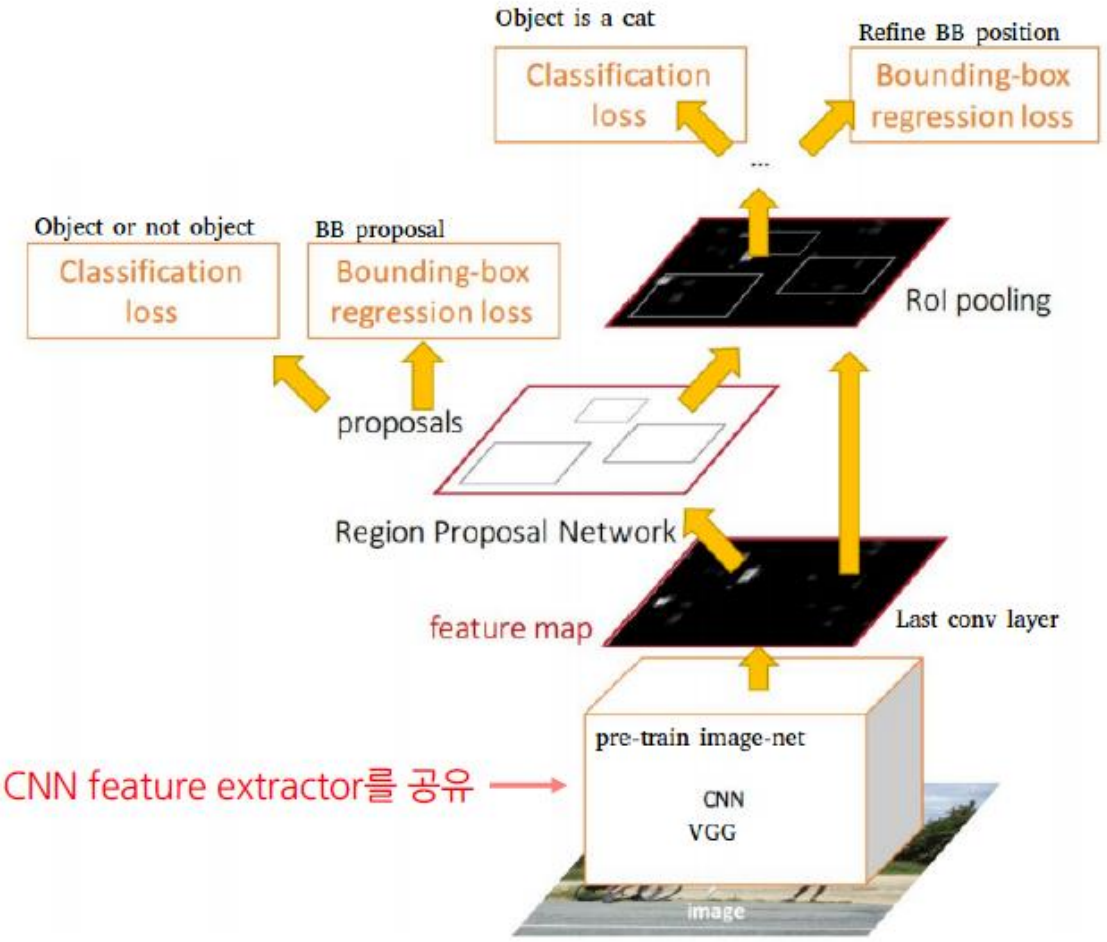
  - 수행 속도 및 성능이 모두 향상됨.

- Faster R-CNN = RPN + Fast R-CNN

  - Region proposal 후에는 기존의 Fast R-CNN을 그대로 이용한다.
  - 단, RPN과 Fast R-CNN은 **CNN feature extractor를 공유**한다.

Faster R-CNN: Overview

Overview



RPN과 classifier/regressor가 CNN feature extractor를 공유 →

Faster R-CNN: RPN(1/3)

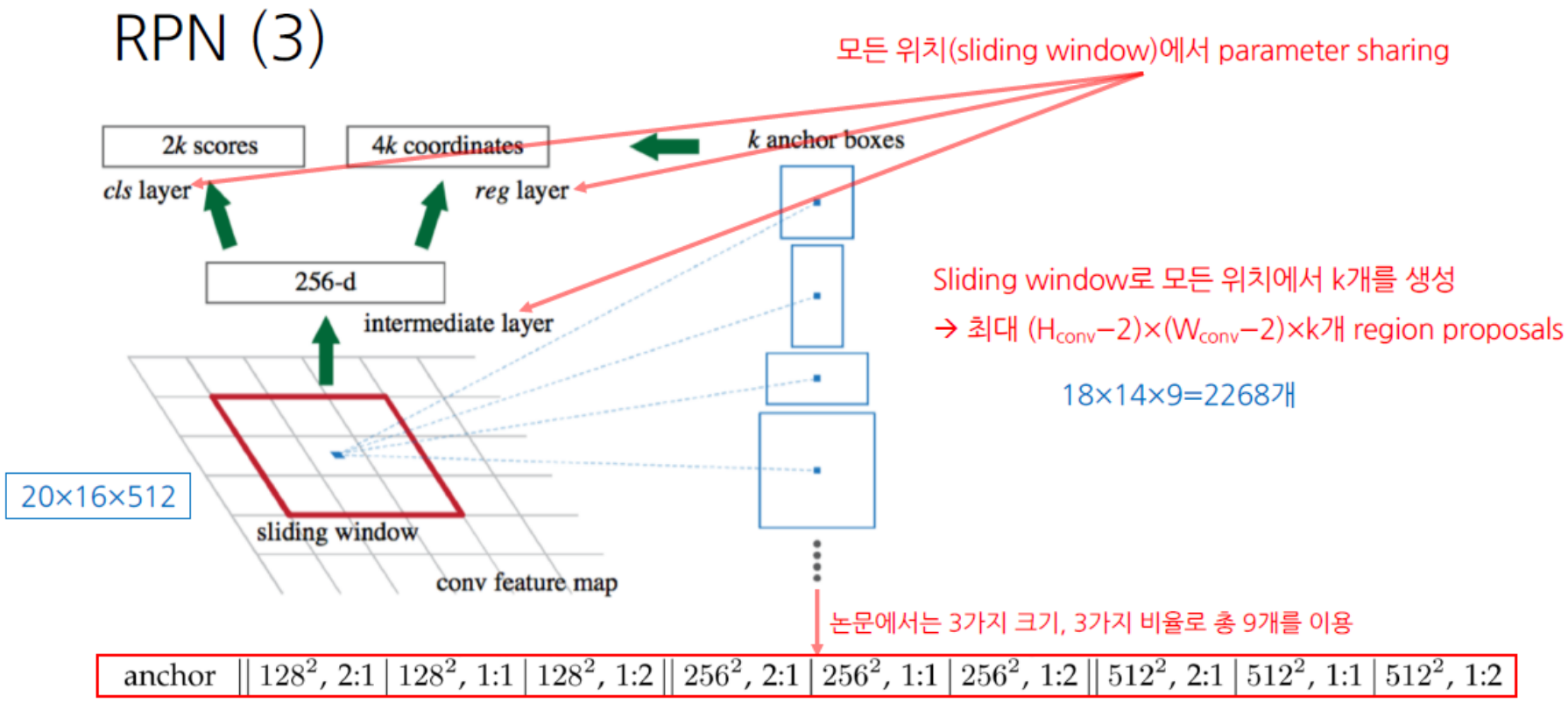
RPN(Region proposal network)

- Input
  - Image (H×W×C)
- Output
  - Region proposals (N_{boxes}×4): bounding box의 좌표
  - Objectness scores (N_{boxes}×2): 물체인지 아닌지를 one-hot으로 표현

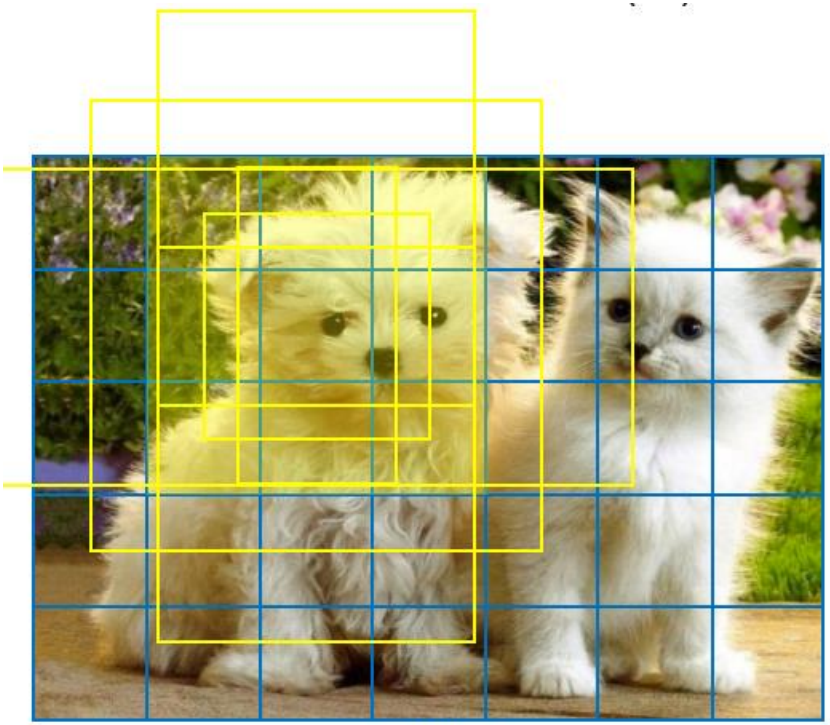
Faster R-CNN: RPN(2/3)

- Anchor boxes
  - Region proposal의 시작점 역할을 수행
    - 1개의 anchor box로부터 1개의 region proposal이 만들어짐.
    - Region proposal은 4개의 box parameter와 2개의 objectness score로 구성됨.
  - 일정 stride마다 동일한 anchor boxes가 존재
    - Image 상에서 object의 위치에 따라 region proposal이 달라지지 않음. (Translation invariance)
    - 모든 위치에서 region proposal을 위한 weight를 공유 → Memory 절약

Faster R-CNN: RPN(3/3)



Faster R-CNN: Anchor Boxes



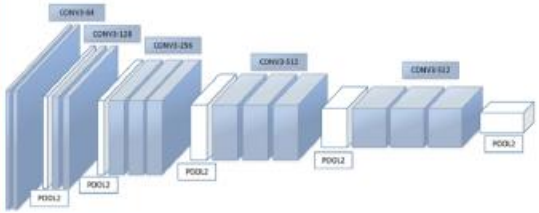
224×160

Anchor box setting:

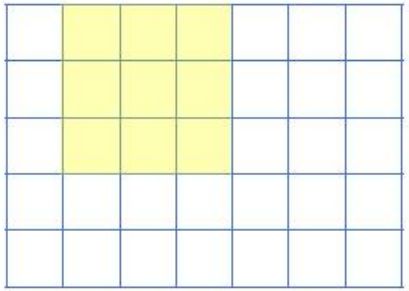
Sizes: 64^2, 128^2

Ratios: 1:1, 1:2, 2:1

→ 64×64, 45×90, 90×45, … (6 anchor boxes)



CNN
(total stride=32)



5×7

Faster R-CNN: Training RPN

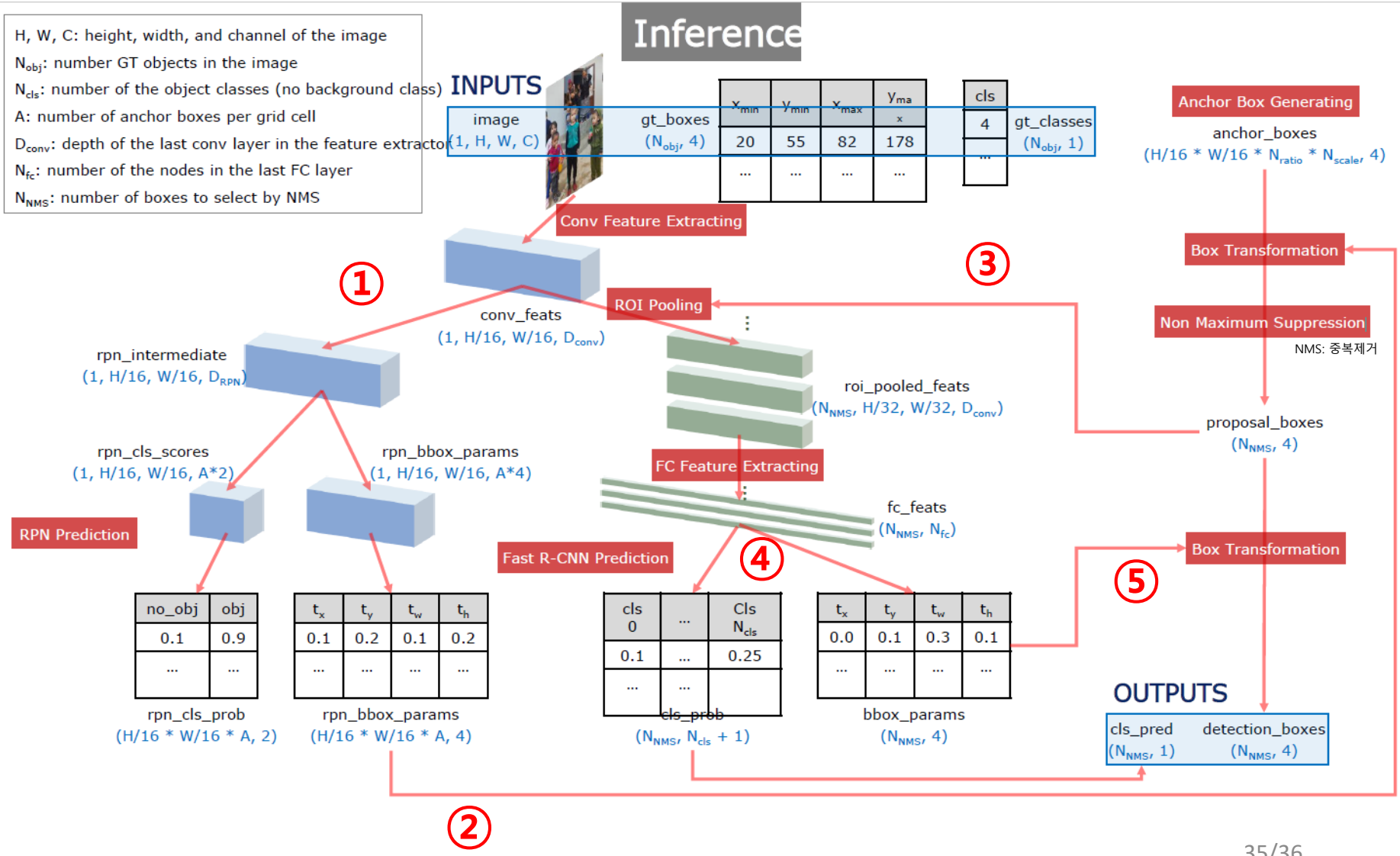
이 조건이 없으면 RPN이 학습하지 못하는 유형의 object가 발생할 수 있음

- Anchor box assignment
  - IoU with GT box ≥ 0.7인 anchor boxes → positive
  - IoU with GT box < 0.3인 anchor boxes → negative
  - Else(Gray zone) → ignore
  - 할당된 anchor box가 없는 GT box에게는 가장 높은 IoU를 갖는 anchor box를 할당
- Image-centric sampling strategy
  - 한 image로부터 1개 batch를 구성해서 update한다.
  - Positive/Negative ratio의 비율은 1:1로 맞춘다.

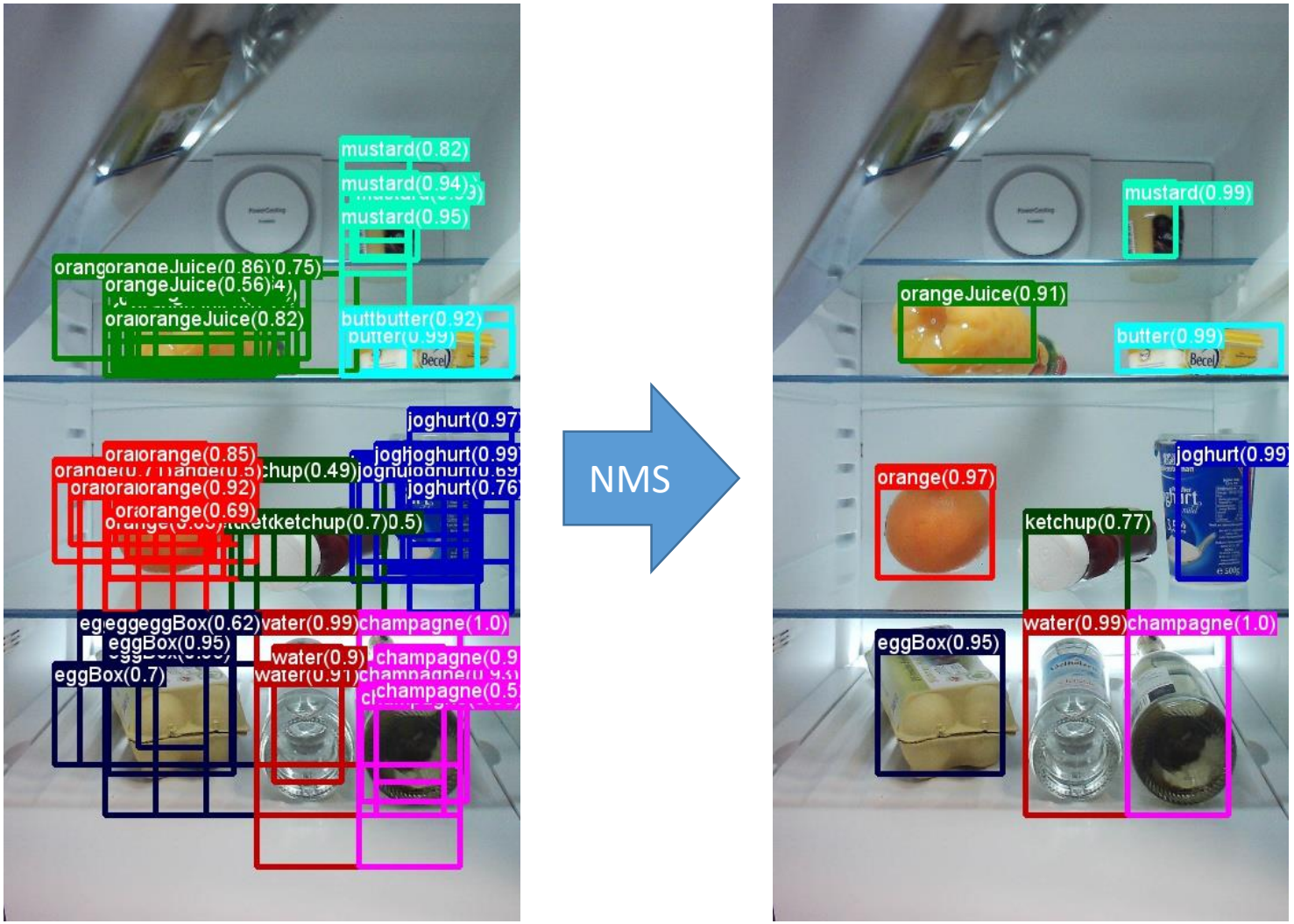
Faster R-CNN: Training

- 4-step alternating training — 이 4단계를 반복해서 수행할 수도 있지만 최종 성능에 큰 차이는 없음
  1. Train RPN
     - ImageNet pretrained model로부터 시작
  2. Train Fast R-CNN
     - ImageNet pretrained model로부터 시작, 1의 RPN이 생성한 RoIs 이용
  3. Train RPN
     - 2의 Fast R-CNN의 CNN을 고정된 상태로 가져와 이용, RPN의 FC layers 학습 ← CNN을 공유
  4. Train Fast R-CNN
     - 3의 CNN을 계속 고정된 상태로 이용, 3의 RPN이 생성한 RoIs 이용

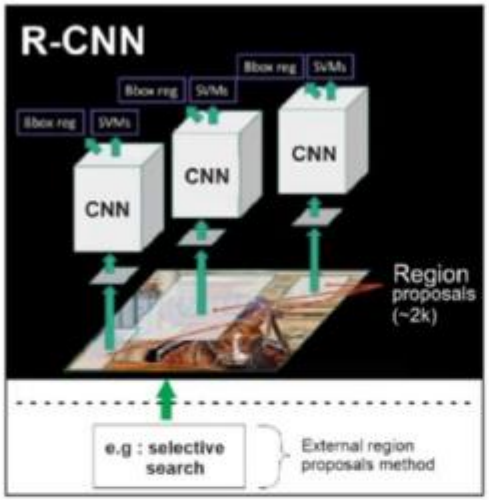
Faster R-CNN: Inference

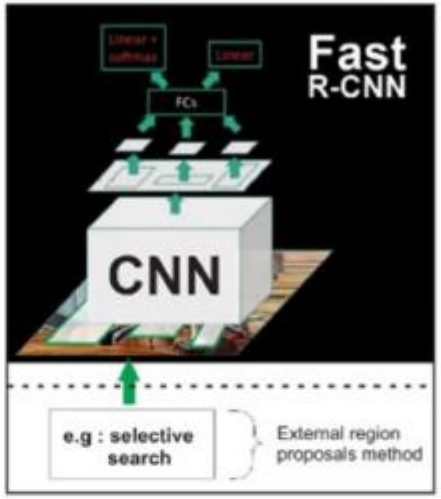


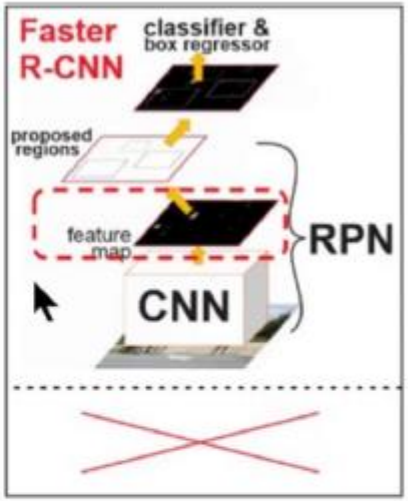
(참고) NMS(None Maximum Suppression)



Faster R-CNN: Result

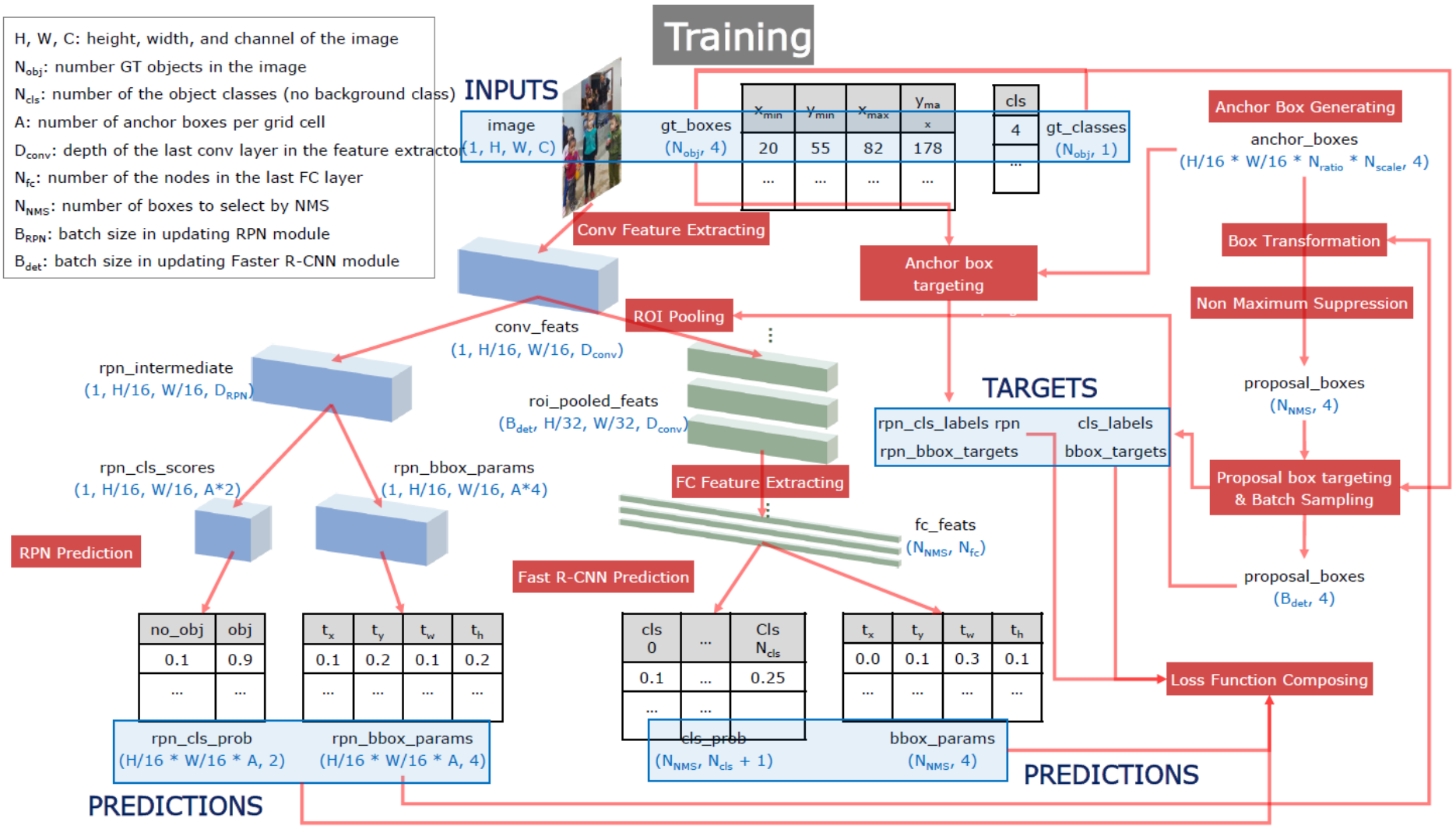






	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

Q&A


H, W, C: height, width, and channel of the image
N_{obj}: number GT objects in the image
N_{cls}: number of the object classes (no background class)
A: number of anchor boxes per grid cell
D_{conv}: depth of the last conv layer in the feature extractor
N_{fc}: number of the nodes in the last FC layer
N_{NMS}: number of boxes to select by NMS
B_{RPN}: batch size in updating RPN module
B_{det}: batch size in updating Faster R-CNN module
Training
INPUTS
image
(1, H, W, C)
gt_boxes
(N_{obj}, 4)
x_{min} y_{min} x_{max} y_{max}
20 55 82 178
gt_classes
(N_{obj}, 1)
cls
4
Conv Feature Extracting
conv_feats
(1, H/16, W/16, D_{conv})
Anchor box targeting
Anchor Box Generating
anchor_boxes
(H/16 * W/16 * N_{ratio} * N_{scale}, 4)
Box Transformation
Non Maximum Suppression
proposal_boxes
(N_{NMS}, 4)
Proposal box targeting & Batch Sampling
proposal_boxes
(B_{det}, 4)
ROI Pooling
rpn_intermediate
(1, H/16, W/16, D_{RPN})
roi_pooled_feats
(B_{det}, H/32, W/32, D_{conv})
TARGETS
rpn_cls_labels rpn
rpn_bbox_targets
cls_labels
bbox_targets
FC Feature Extracting
fc_feats
(N_{NMS}, N_{fc})
rpn_cls_scores
(1, H/16, W/16, A*2)
rpn_bbox_params
(1, H/16, W/16, A*4)
RPN Prediction
Fast R-CNN Prediction
no_obj obj
0.1 0.9
t_x t_y t_w t_h
0.1 0.2 0.1 0.2
cls 0 ... Cls N_{cls}
0.1 ... 0.25
t_x t_y t_w t_h
0.0 0.1 0.3 0.1
Loss Function Composing
rpn_cls_prob
(H/16 * W/16 * A, 2)
rpn_bbox_params
(H/16 * W/16 * A, 4)
cls_prob
(N_{NMS}, N_{cls} + 1)
bbox_params
(N_{NMS}, 4)
PREDICTIONS
PREDICTIONS