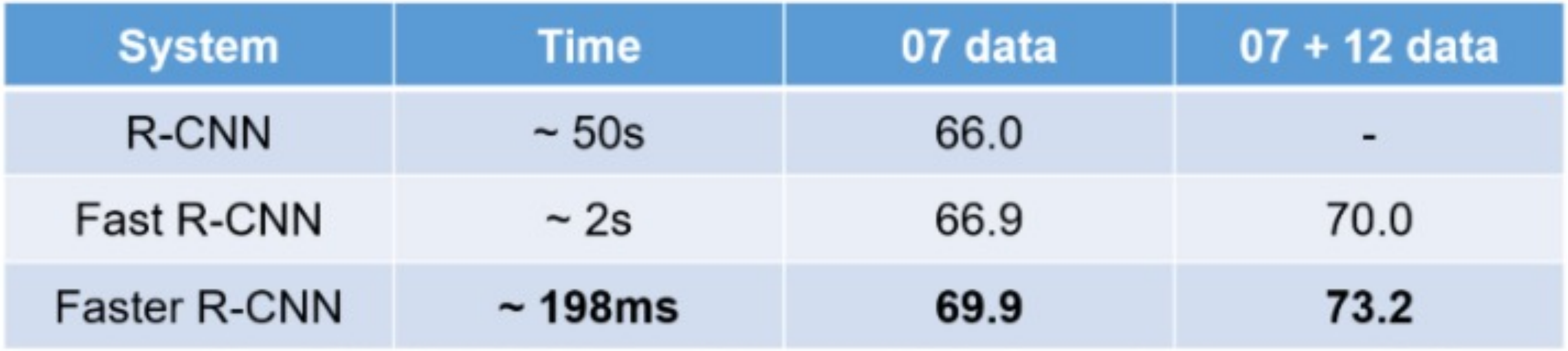
인공지능 심화 과정

R-CNN
SPP-Net
Fast-RCNN
Faster RCNN

R-CNN VS Fast R-CNN VS Faster R-CNN

However, there is a technical issue in the training and testing of the CNNs: the prevalent CNNs require a fixed input image size (e.g., 224×224), which limits both the aspect ratio and the scale of the input image. When applied to images of arbitrary sizes, current methods mostly fit the input image to the fixed size, either via cropping [3], [4] or via warping [13], [7], as shown in Figure 1 (top). But the cropped region may not contain the entire object, while the warped content may result in unwanted geometric distortion. Recognition accuracy can be compromised due to the content loss or distortion.



System	Time	07 data	07 + 12 data
R-CNN	~ 50s	66.0	-
Fast R-CNN	~ 2s	66.9	70.0
Faster R-CNN	**~ 198ms**	**69.9**	**73.2**

Detection mAP on PASCAL VOC 2007 and 2012, with VGG-16 pre-trained on ImageNet Dataset

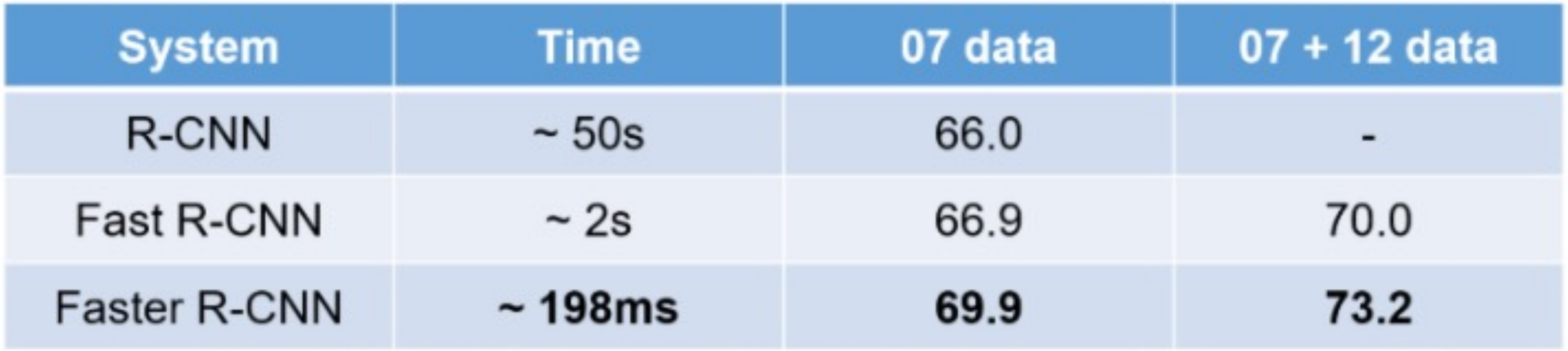
SPP-Net

Introduction

We are witnessing a rapid, revolutionary change in our vision community, mainly caused by deep convolutional neural networks (CNNs)

…

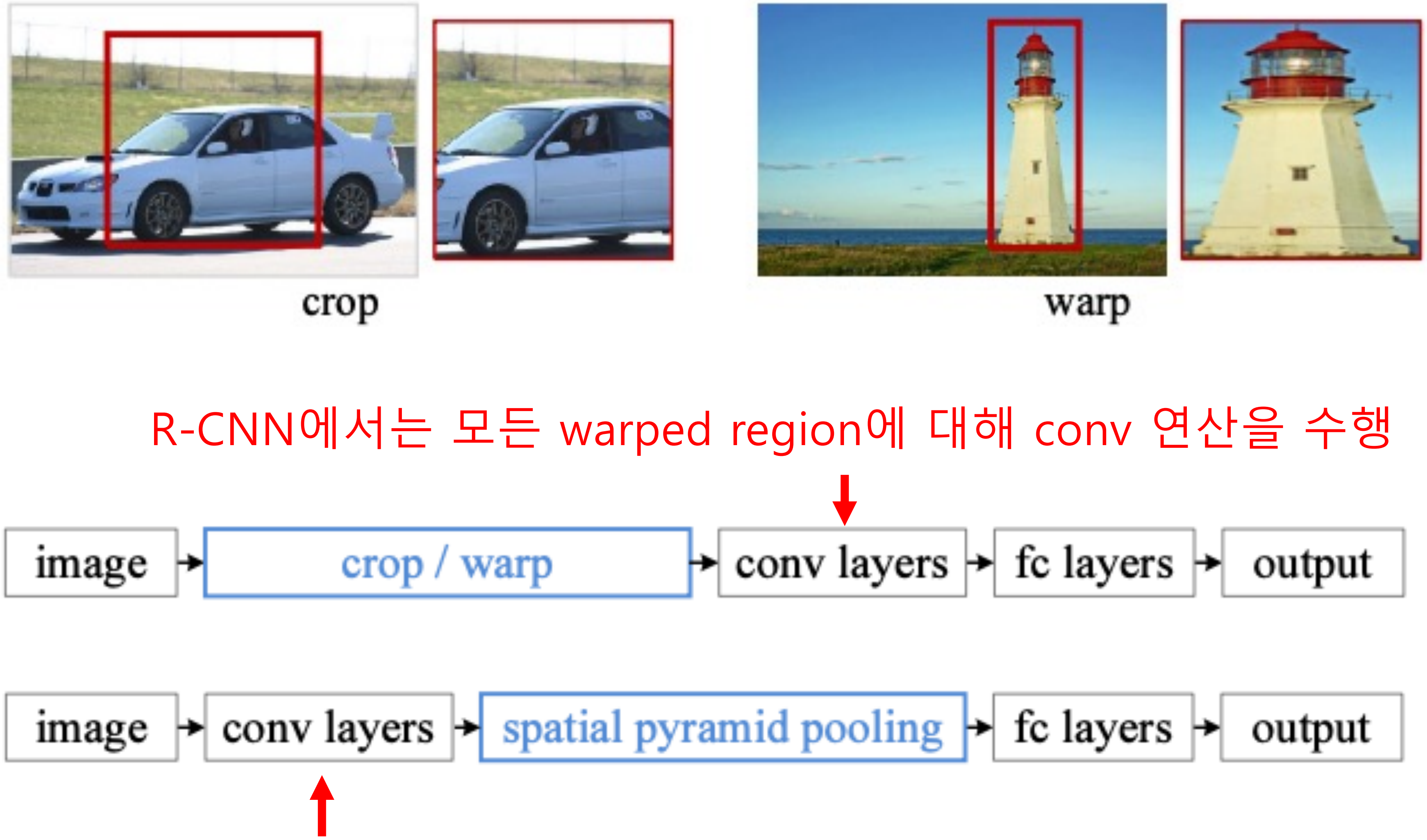
However, there is a technical issue in the training and testing of the CNNs: the prevalent CNNs require a fixed input image size (e.g., 224×224), which limits both the aspect ratio and the scale of the input image. When applied to images of arbitrary sizes, current methods mostly fit the input image to the fixed size, either via cropping [3], [4] or via warping [13], [7], as shown in Figure 1 (top). But the cropped region may not contain the entire object, while the warped content may result in unwanted geometric distortion. Recognition accuracy can be compromised due to the content loss or distortion.



System	Time	07 data	07 + 12 data
R-CNN	~ 50s	66.0	-
Fast R-CNN	~ 2s	66.9	70.0
Faster R-CNN	**~ 198ms**	**69.9**	**73.2**

Detection mAP on PASCAL VOC 2007 and 2012, with VGG-16 pre-trained on ImageNet Dataset

SPP-Net



R-CNN에서는 모든 warped region에 대해 conv 연산을 수행

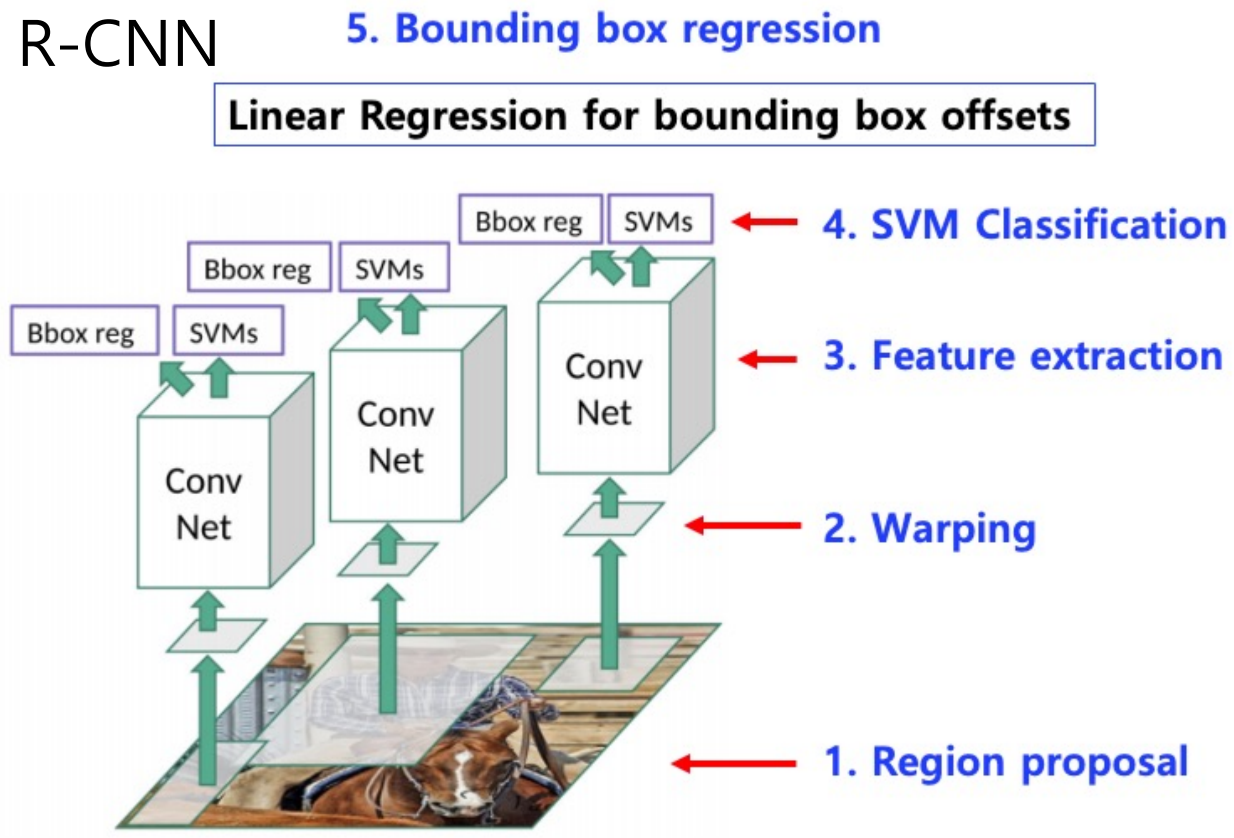
전체 이미지에 대해 conv 연산을 한 번만 수행

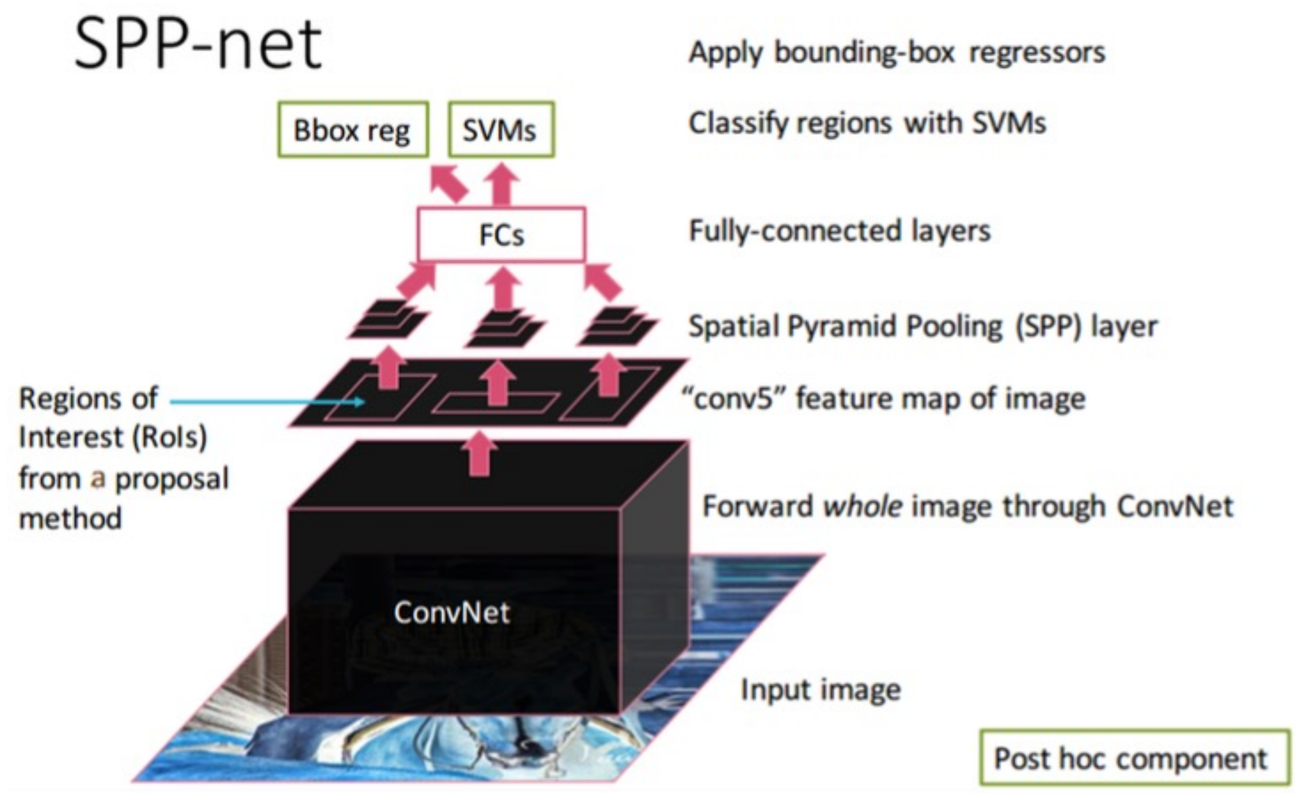
Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

https://arxiv.org/pdf/1406.4729.pdf

SPP-Net

R-CNN




SPP-net
Bbox reg
SVMs
Apply bounding-box regressors
Classify regions with SVMs
FCs
Fully-connected layers
Spatial Pyramid Pooling (SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image
Post hoc component

https://arxiv.org/pdf/1406.4729.pdf

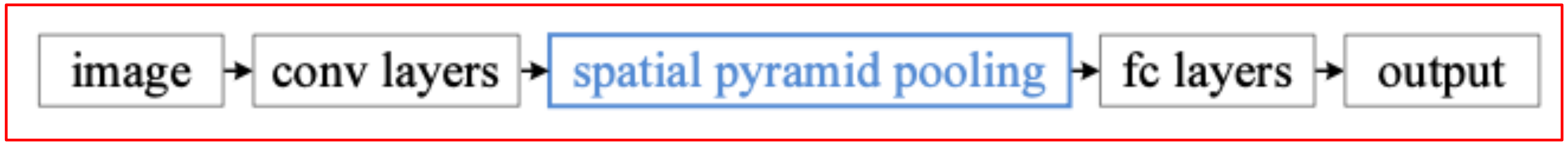
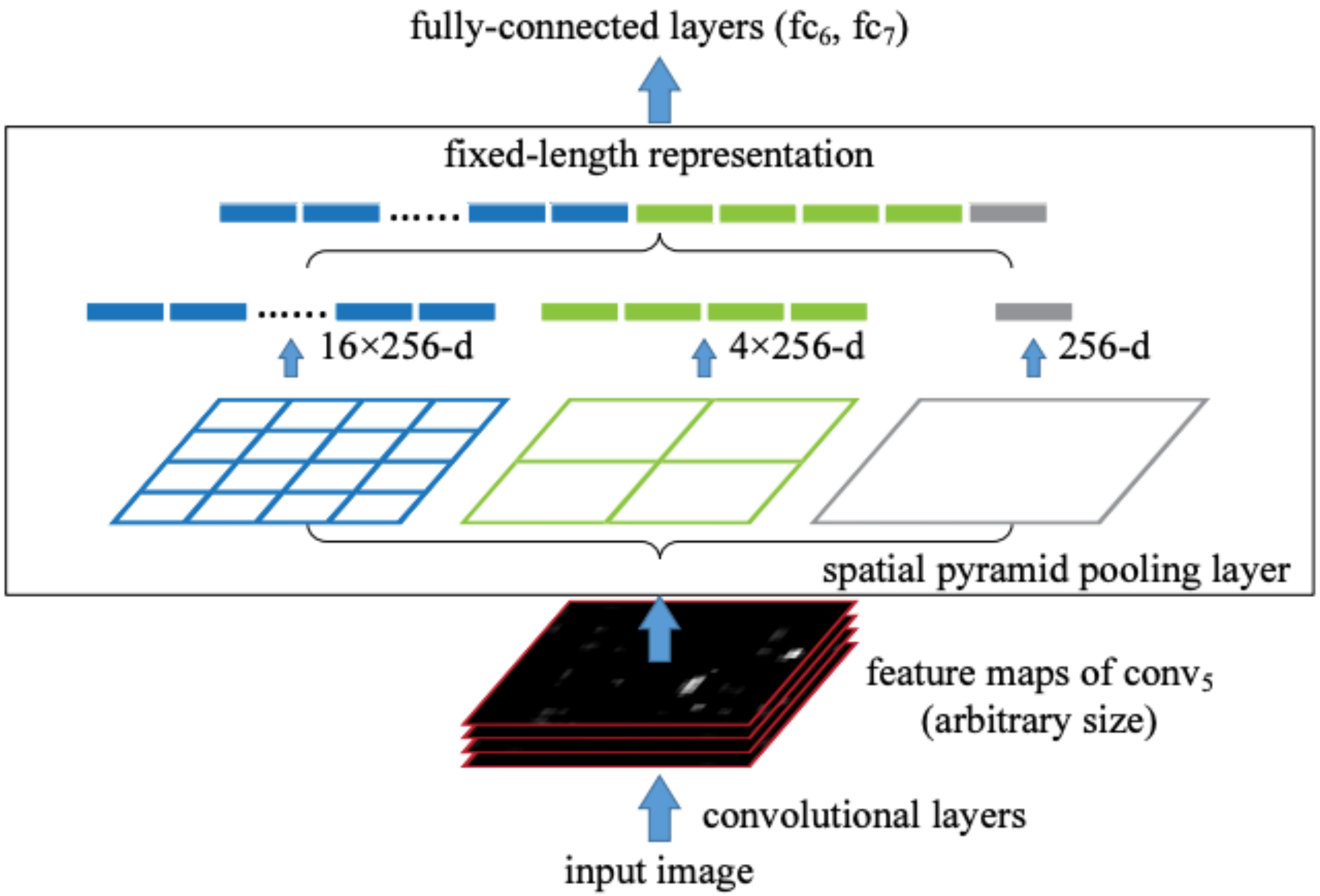


image → conv layers → spatial pyramid pooling → fc layers → output

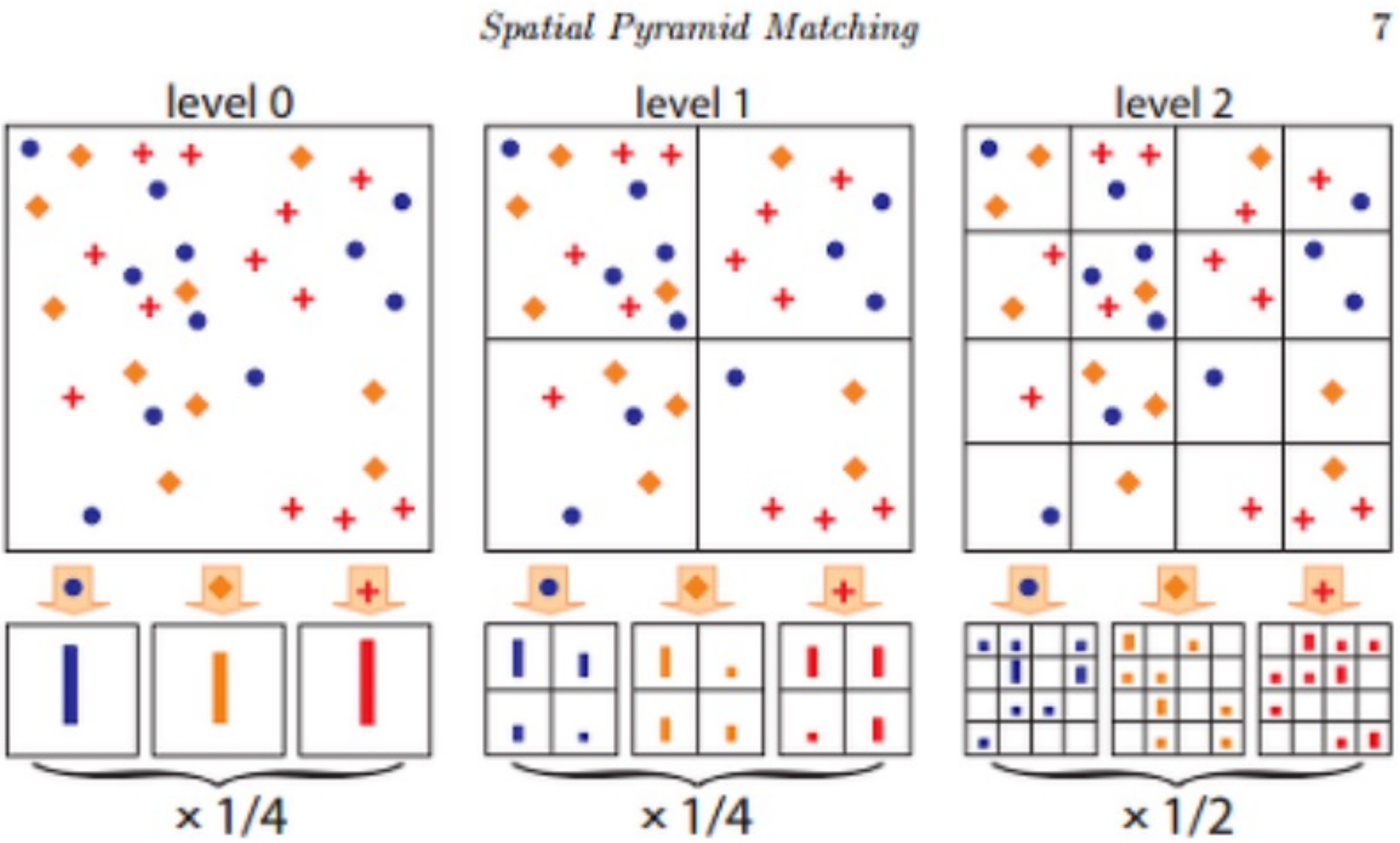
SPP-Net

다양한 feature에서 pooling의 window size와 stride 만을 조절하여 출력 크기를 결정



2006년도에 발표된 Spatial Pyramid Matching을 참조

SPP-Net



Spatial Pyramid Matching 7

[Spatial Pyramid Matching]

SPP-Net

SPP-Net의 개선점

✓ SPPnet은 CNN을 이미지에 한 번만 적용하여 속도가 빠르다.

✓ CNN으로 생성된 feature map에서 selective search에서 추천된 region을 window로 잘라 특징을 추출한다.

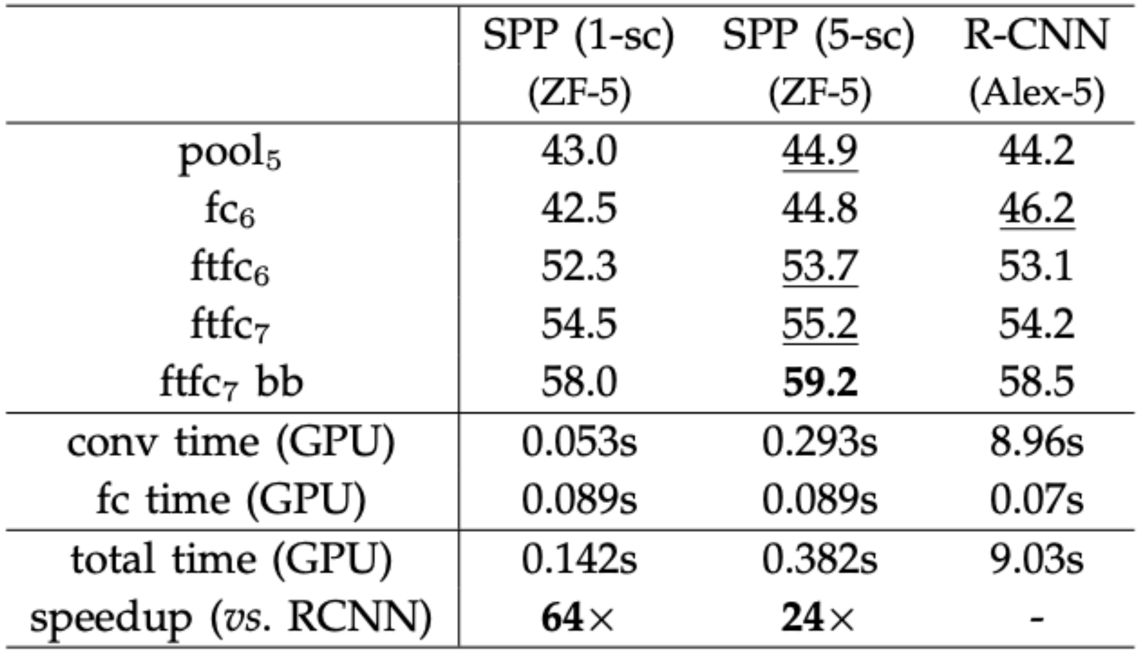
✓ feature map에서 임의의 크기 window로 특징 추출이 가능하다.

✓ 입력 이미지의 scale, size, aspect ratio에 영향을 받지 않는다.

그러나, 여전히 학습이 여러 단계로 나누어져 있다.

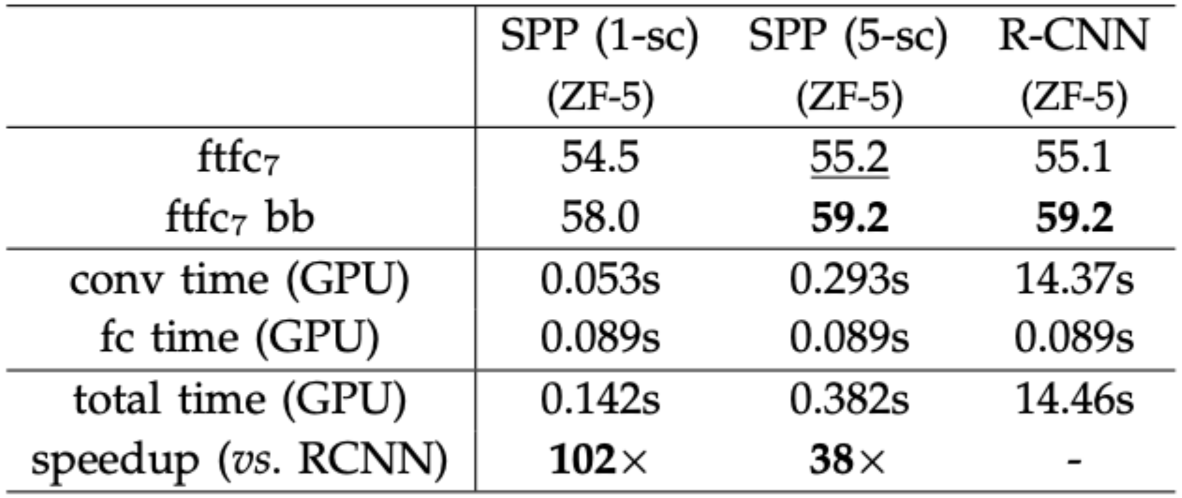
Feature extraction, SVM Classification, bounding box regression

SPP-Net



	SPP (1-sc) (ZF-5)	SPP (5-sc) (ZF-5)	R-CNN (Alex-5)
pool$_5$	43.0	44.9	44.2
fc$_6$	42.5	44.8	46.2
ftfc$_6$	52.3	53.7	53.1
ftfc$_7$	54.5	55.2	54.2
ftfc$_7$ bb	58.0	**59.2**	58.5
conv time (GPU)	0.053s	0.293s	8.96s
fc time (GPU)	0.089s	0.089s	0.07s
total time (GPU)	0.142s	0.382s	9.03s
speedup (*vs.* RCNN)	**64**×	**24**×	-

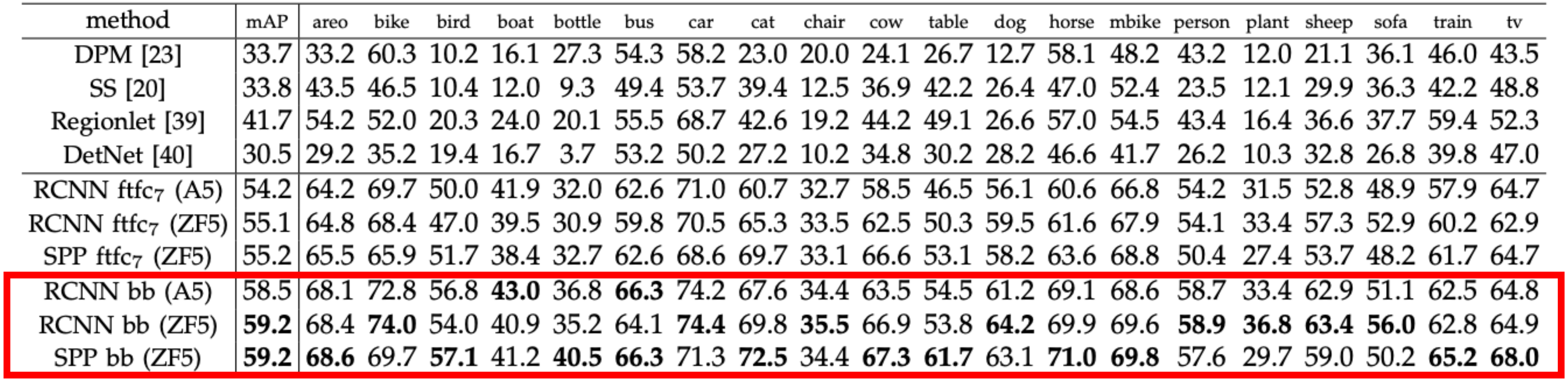
Table 9: Detection results (mAP) on Pascal VOC 2007. "ft" and "bb" denote fine-tuning and bounding box regression.



	SPP (1-sc) (ZF-5)	SPP (5-sc) (ZF-5)	R-CNN (ZF-5)
ftfc$_7$	54.5	55.2	55.1
ftfc$_7$ bb	58.0	**59.2**	**59.2**
conv time (GPU)	0.053s	0.293s	14.37s
fc time (GPU)	0.089s	0.089s	0.089s
total time (GPU)	0.142s	0.382s	14.46s
speedup (*vs.* RCNN)	**102**×	**38**×	-

Table 10: Detection results (mAP) on Pascal VOC 2007, **using the same pre-trained model** of SPP (ZF-5).

SPP-Net



method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
DPM [23]	33.7	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5
SS [20]	33.8	43.5	46.5	10.4	12.0	9.3	49.4	53.7	39.4	12.5	36.9	42.2	26.4	47.0	52.4	23.5	12.1	29.9	36.3	42.2	48.8
Regionlet [39]	41.7	54.2	52.0	20.3	24.0	20.1	55.5	68.7	42.6	19.2	44.2	49.1	26.6	57.0	54.5	43.4	16.4	36.6	37.7	59.4	52.3
DetNet [40]	30.5	29.2	35.2	19.4	16.7	3.7	53.2	50.2	27.2	10.2	34.8	30.2	28.2	46.6	41.7	26.2	10.3	32.8	26.8	39.8	47.0
RCNN ftfc$_7$ (A5)	54.2	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7
RCNN ftfc$_7$ (ZF5)	55.1	64.8	68.4	47.0	39.5	30.9	59.8	70.5	65.3	33.5	62.5	50.3	59.5	61.6	67.9	54.1	33.4	57.3	52.9	60.2	62.9
SPP ftfc$_7$ (ZF5)	55.2	65.5	65.9	51.7	38.4	32.7	62.6	68.6	69.7	33.1	66.6	53.1	58.2	63.6	68.8	50.4	27.4	53.7	48.2	61.7	64.7
RCNN bb (A5)	58.5	68.1	72.8	56.8	**43.0**	36.8	**66.3**	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8
RCNN bb (ZF5)	**59.2**	68.4	**74.0**	54.0	40.9	35.2	64.1	**74.4**	69.8	**35.5**	66.9	53.8	**64.2**	69.9	69.6	**58.9**	**36.8**	**63.4**	**56.0**	62.8	64.9
SPP bb (ZF5)	**59.2**	**68.6**	69.7	**57.1**	41.2	**40.5**	**66.3**	71.3	**72.5**	34.4	**67.3**	**61.7**	63.1	**71.0**	**69.8**	57.6	29.7	59.0	50.2	**65.2**	**68.0**

Table 11: Comparisons of detection results on Pascal VOC 2007.

인공지능 심화 과정

R-CNN
SPP-Net
Fast-RCNN
Faster RCNN

Fast R-CNN

SPP-Net에서의 성능 개선점을 Fast R-CNN에 적용

Fast R-CNN도 동일하게

1. softmax classification(pre-training)
2. SVM classification
3. boundingbox regression

의 3단계로 진행된다.

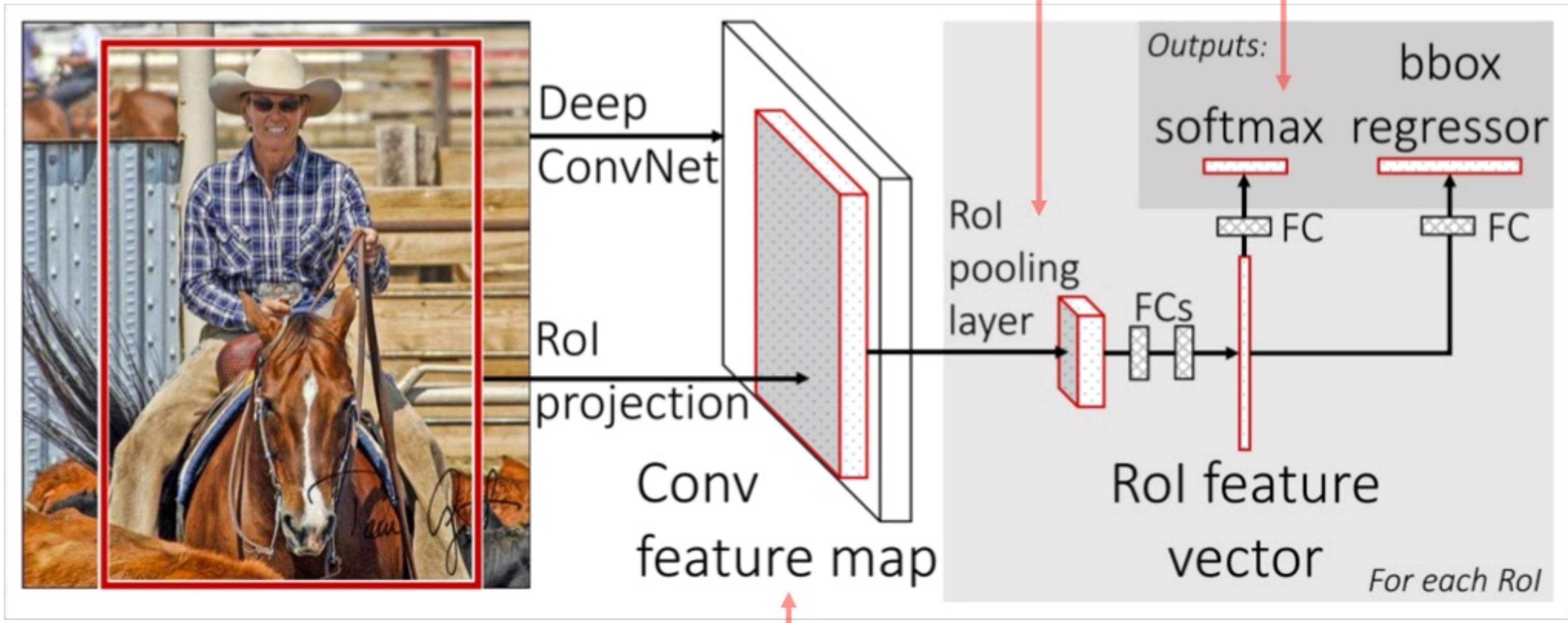
Multi-task Loss 구성

Pre-Training이후 2번과 3번 과정이 동시에 진행되게 함

Fast R-CNN Overview

RoI pooling = Spatial pyramid pooling의 특수한 형태

SVM 대신 softmax layer로 classification



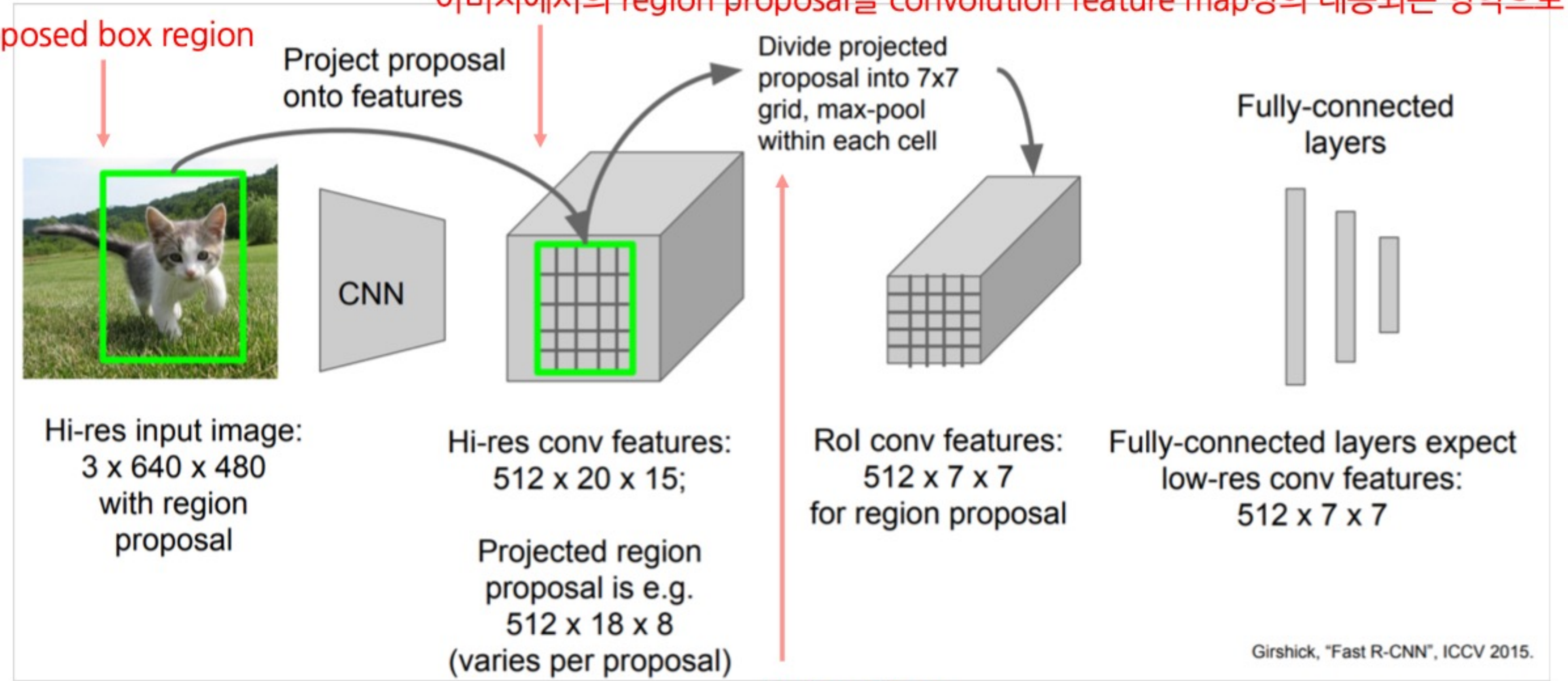
(SPP-Net에서와 같이) 전체 이미지에 대해 conv layers 한 번만 계산

Fast R-CNN

RoI Projection:

이미지에서의 region proposal을 convolution feature map상의 대응되는 영역으로 mapping

Proposed box region



Project proposal onto features

Divide projected proposal into 7x7 grid, max-pool within each cell

Fully-connected layers

Hi-res input image:
3 x 640 x 480
with region
proposal

Hi-res conv features:
512 x 20 x 15;

Projected region
proposal is e.g.
512 x 18 x 8
(varies per proposal)

RoI conv features:
512 x 7 x 7
for region proposal

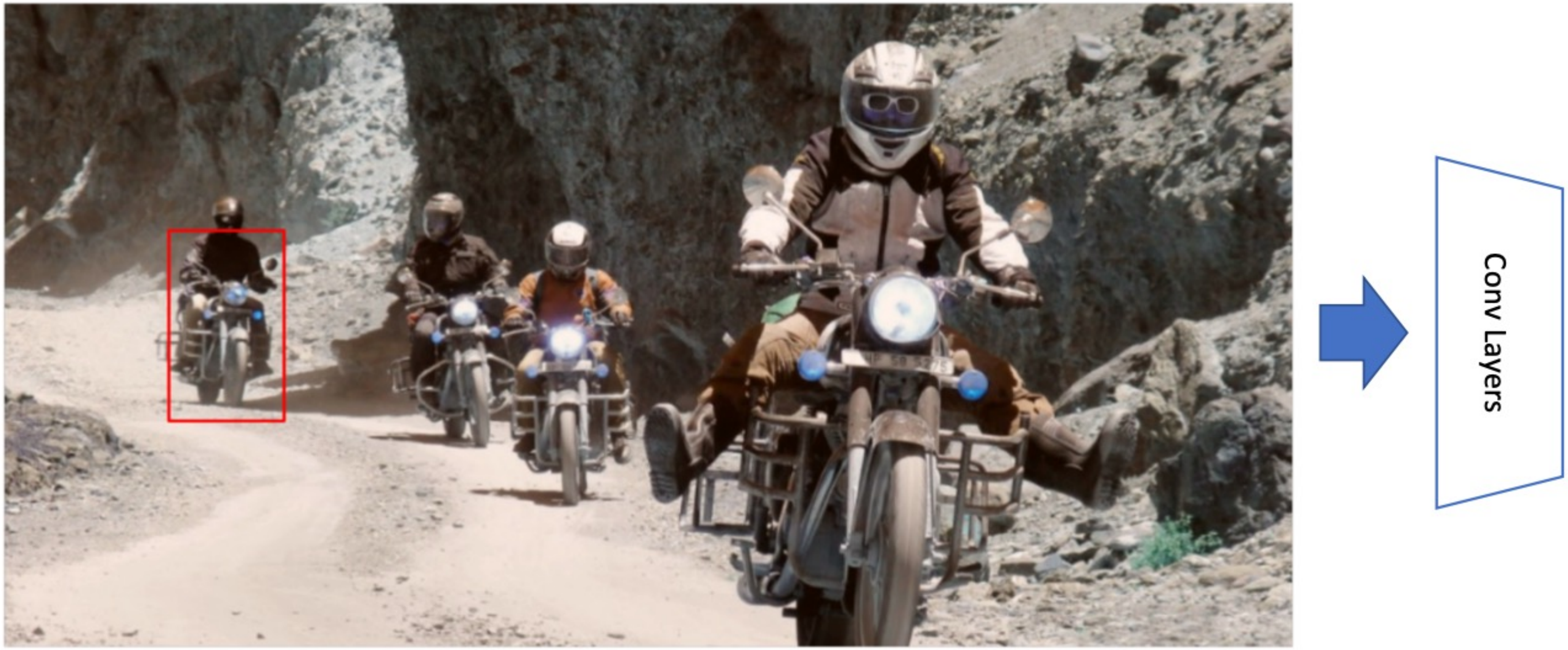
Fully-connected layers expect
low-res conv features:
512 x 7 x 7

Girshick, "Fast R-CNN", ICCV 2015.

RoI pooling:

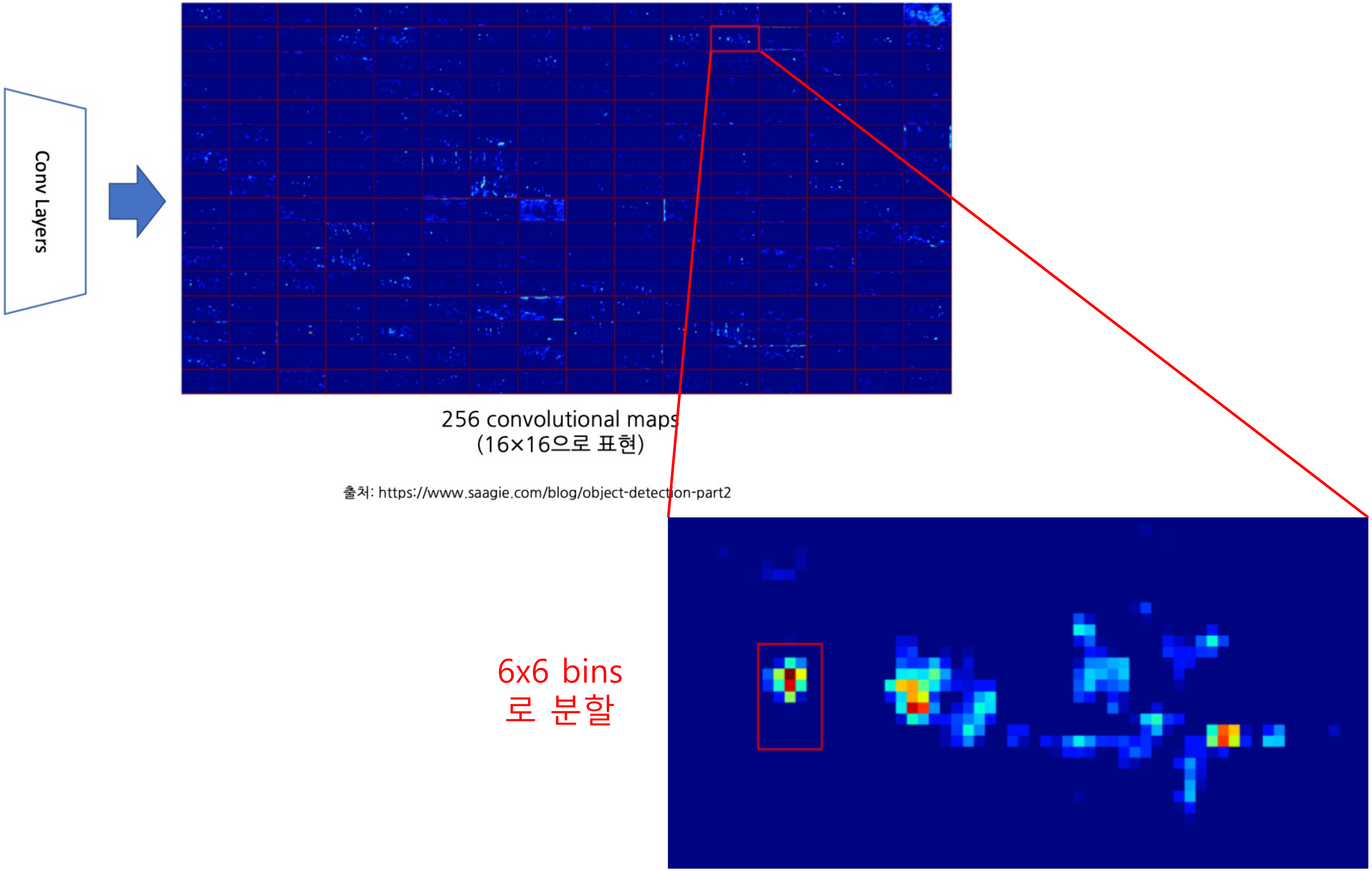
임의 크기의 feature map을 고정 크기로 변환

Fast R-CNN : ROI Pooling



출처: https://www.saagie.com/blog/object-detection-part2

Fast R-CNN : ROI Pooling



256 convolutional maps
(16×16으로 표현)

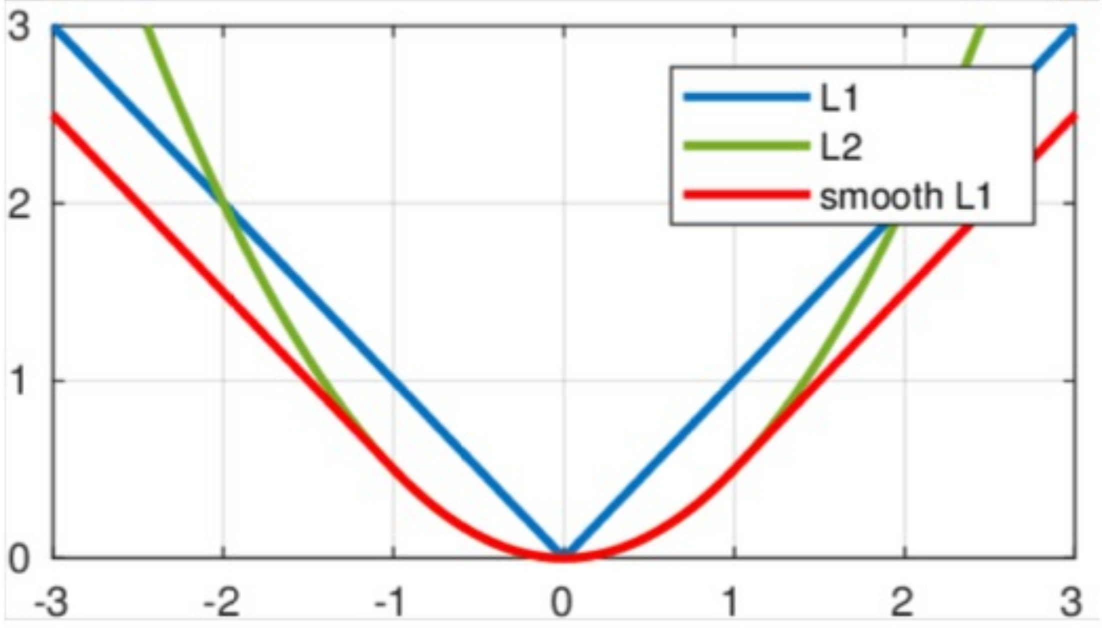
출처: https://www.saagie.com/blog/object-detection-part2

6x6 bins
로 분할

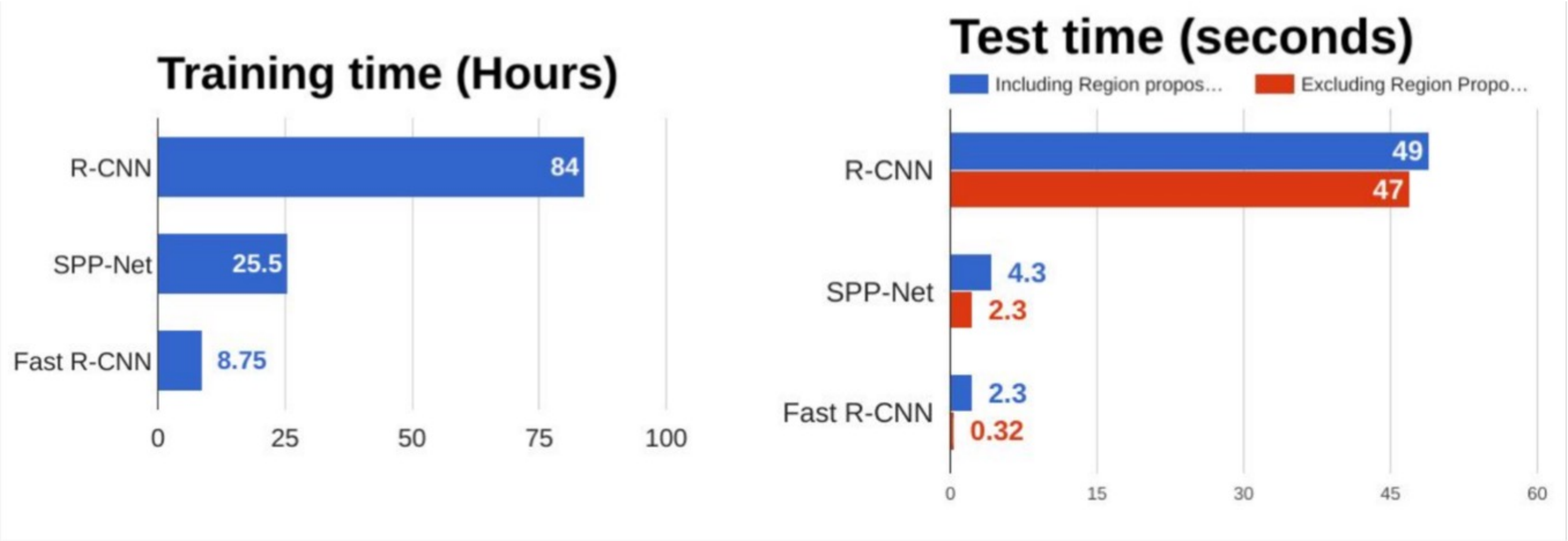
Fast R-CNN : Smooth L1 Loss

- Less sensitive to outliers than the L2 loss(R-CNN, SPP-Net)

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$



R-CNN VS SPP-Net VS Fast R-CNN

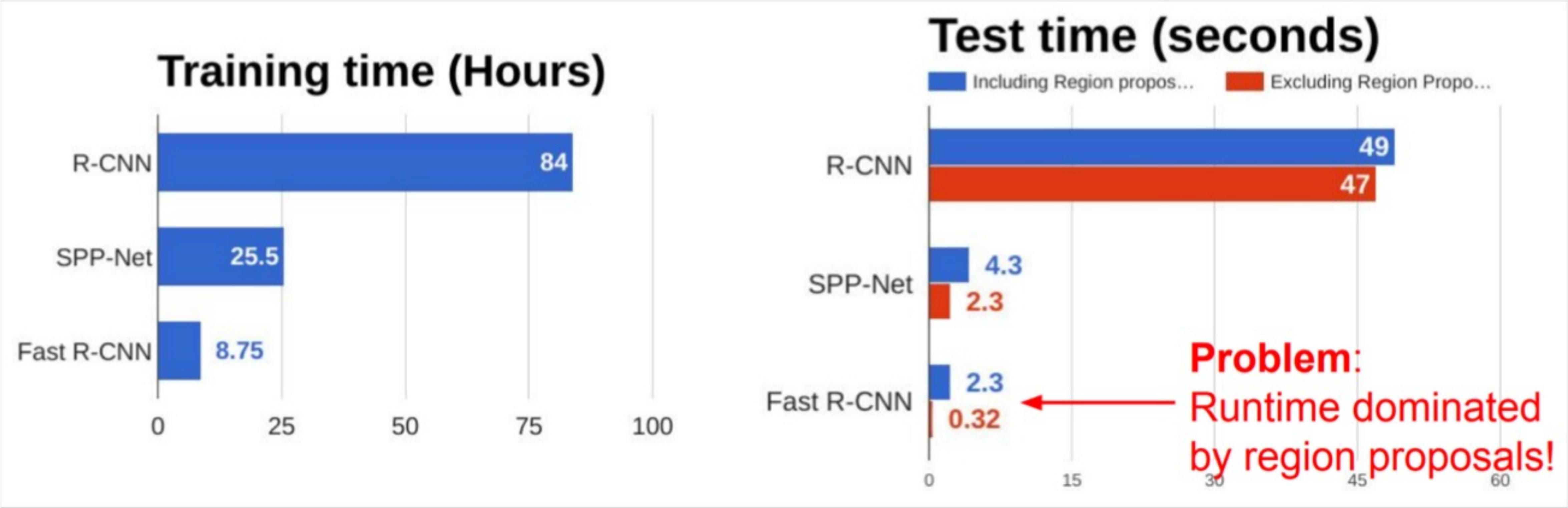


출처: http://cs231n.stanford.edu

인공지능 심화 과정

R-CNN
SPP-Net
Fast-RCNN
Faster RCNN

Faster R-CNN



Problem:
Runtime dominated
by region proposals!

출처: http://cs231n.stanford.edu

Selective Search는 CPU연산을 통해 수행되므로 속도가 느리다.

CNN을 통해 이루어지는 다른 파트에 비해 Bottle-neck이 되고 있다.

Faster R-CNN

Selective Search를 대신할 region proposal 방법을 찾음

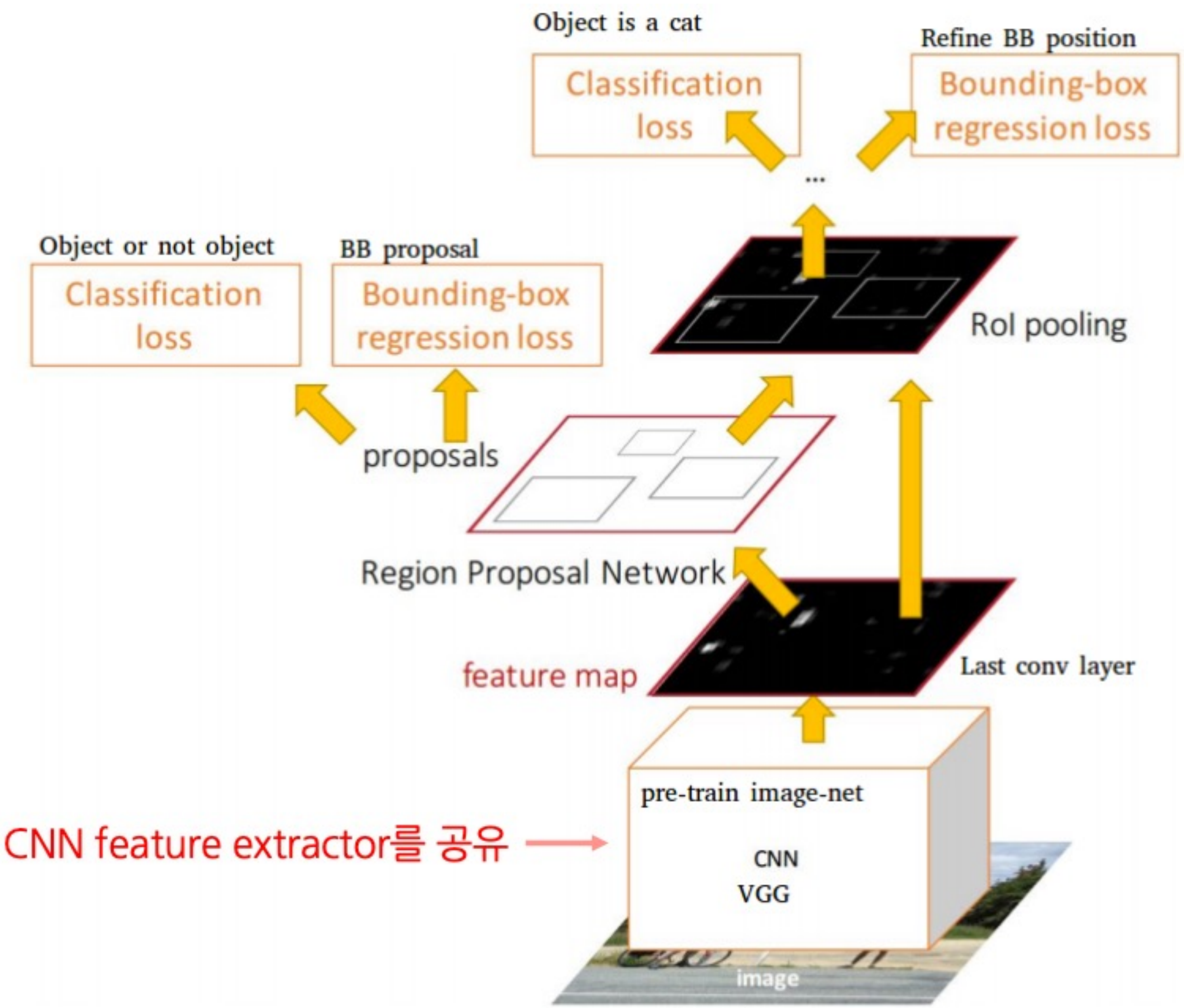
Region proposal Network (RPN). -> 수행 속도, 성능 모두 향상됨

Faster R-CNN = RPN + Fast R-CNN

Region proposal 뒤에는 기존의 Fast R-CNN을 그대로 사용한다.

단, RPN과 Fast R-CNN은 CNN Feature extractor를 공유 한다.

Faster R-CNN : Overview



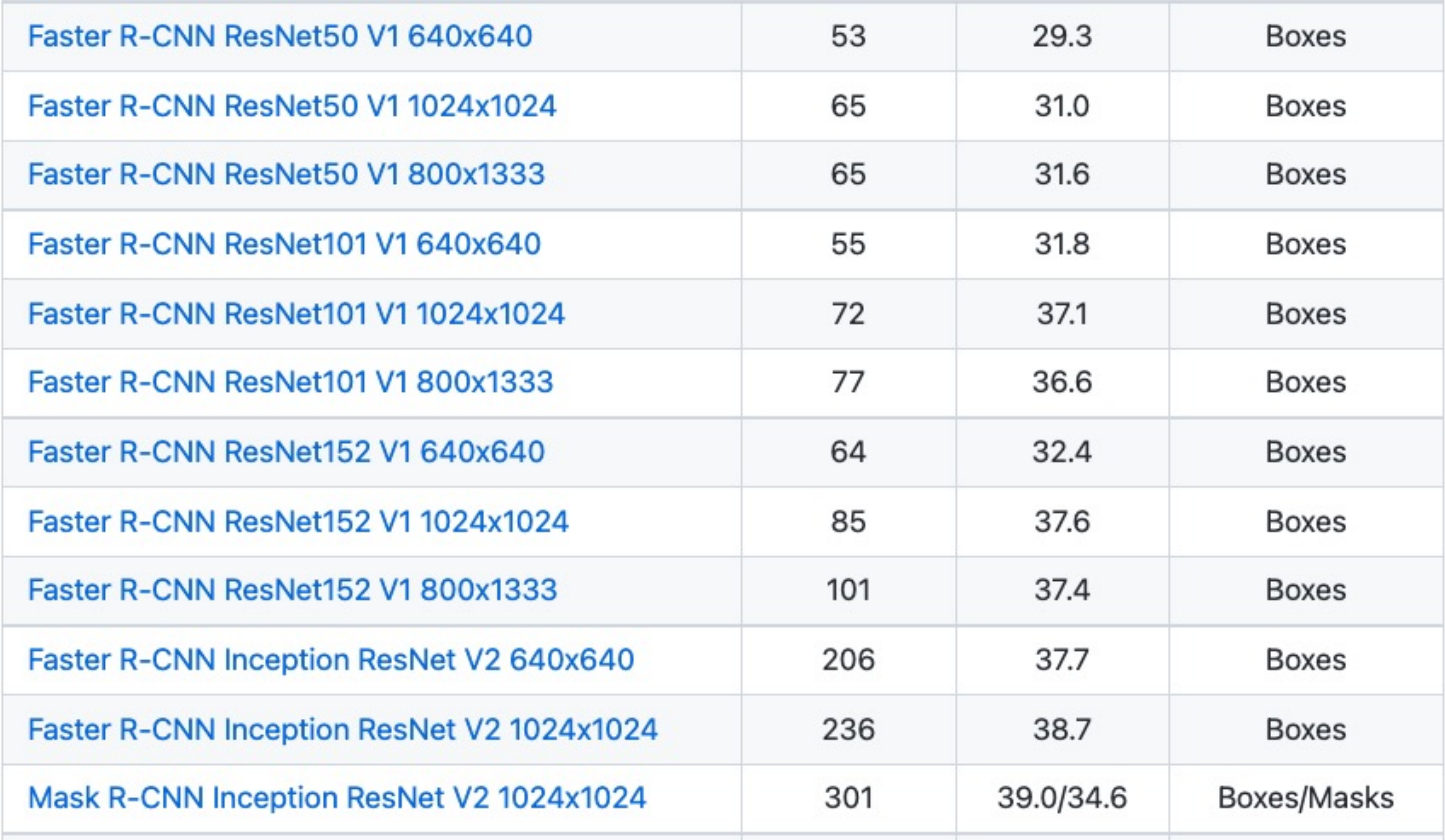
RPN과 classifier/regressor가 CNN feature extractor를 공유 ⟶

출처: https://www.saagie.com/blog/object-detection-part2

Faster R-CNN : Feature Extractor

- Architecture
  - 논문에서는 ZF와 VGG로 실험
  - ResNet, Inception, NAS 등으로도 대체 가능
  - Tensorflow pretrained models
    - Classification models
      - Feature extractor 부분만 가져와서 fine-tuning
    - Detection models
      - Classification 부분이 동일하면 바로 inference 가능

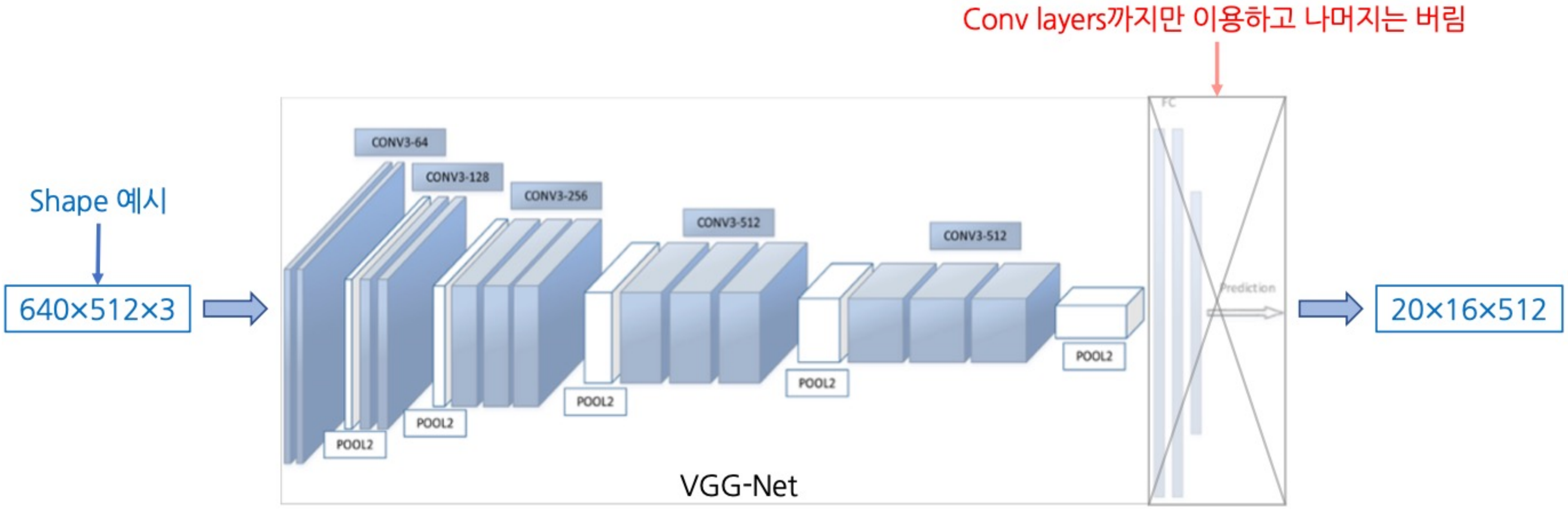
Faster R-CNN : Tensorflow object detection Model Zoo



Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Faster R-CNN



Conv layers까지만 이용하고 나머지는 버림

Shape 예시

640×512×3

20×16×512

VGG-Net

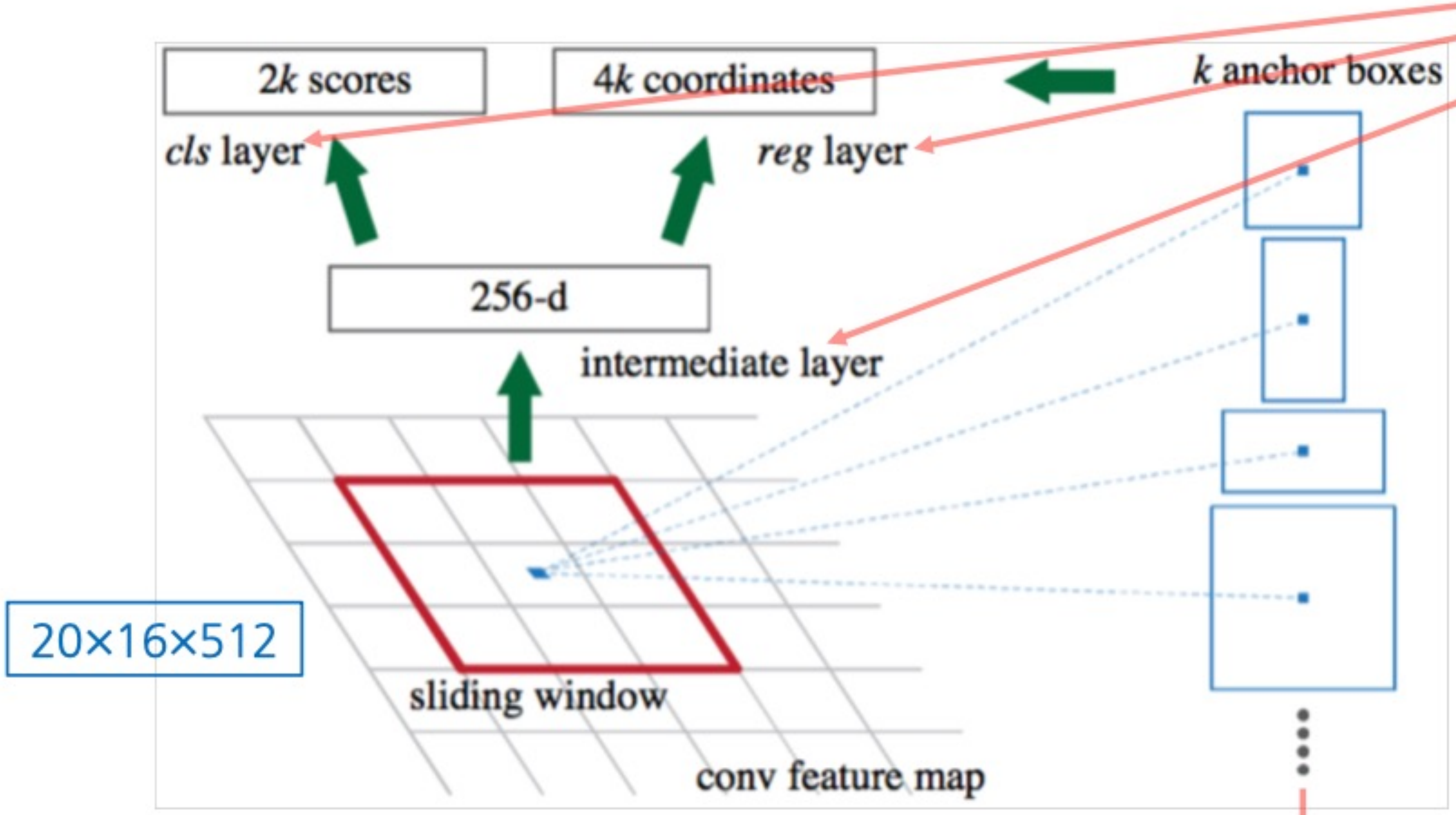
(total stride=32)

Faster R-CNN : RPN

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

모든 위치(sliding window)에서 parameter sharing



Sliding window로 모든 위치에서 k개를 생성

→ 최대 $(H_{conv}-2)\times(W_{conv}-2)\times k$개 region proposals

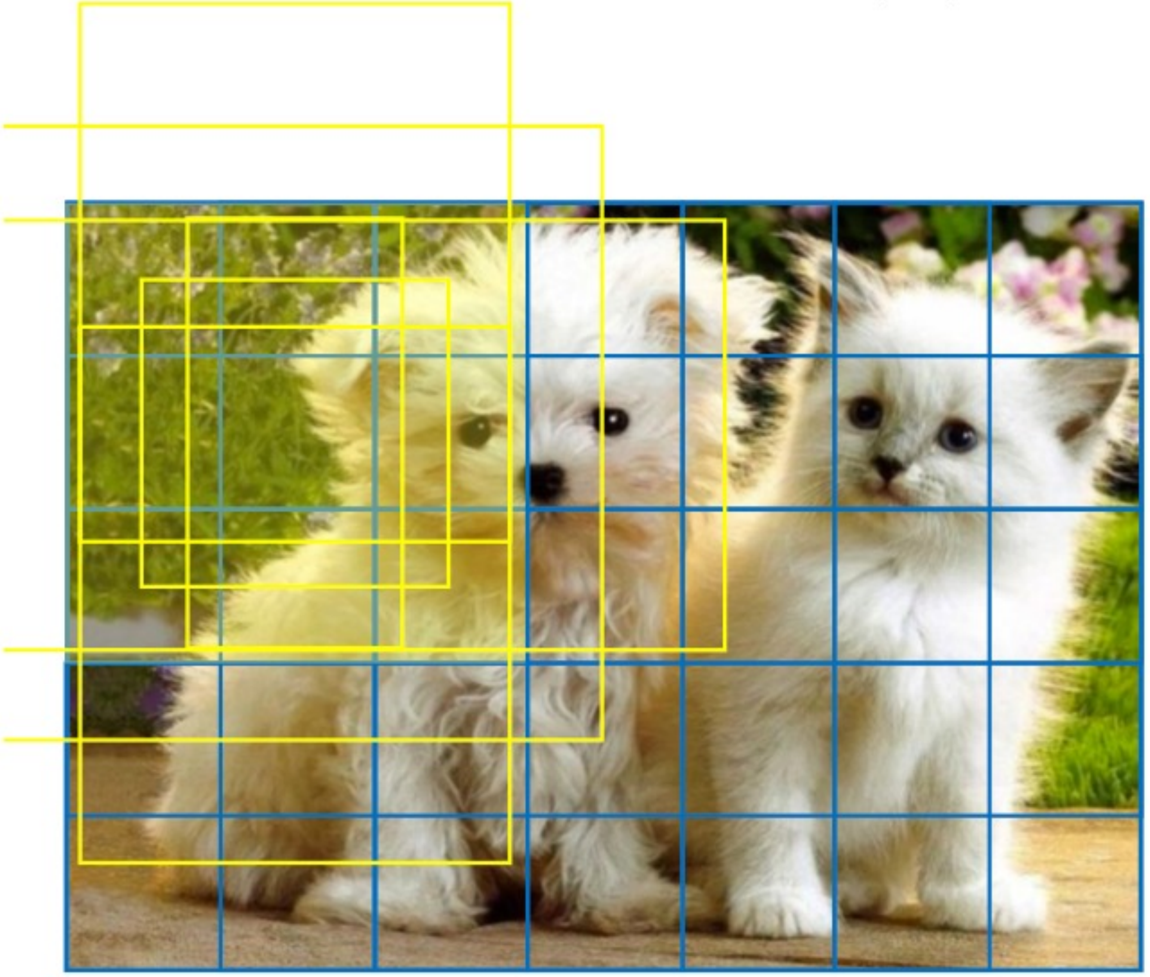
18×14×9=2268개

논문에서는 3가지 크기, 3가지 비율로 총 9개를 이용

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2

출처: Ren et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

Faster R-CNN : Anchor Boxes



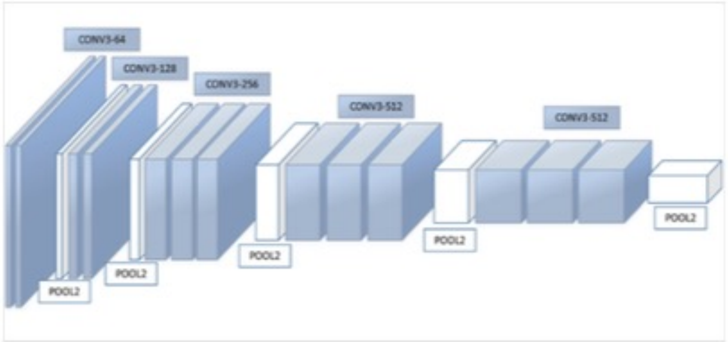
224×160

Anchor box setting:

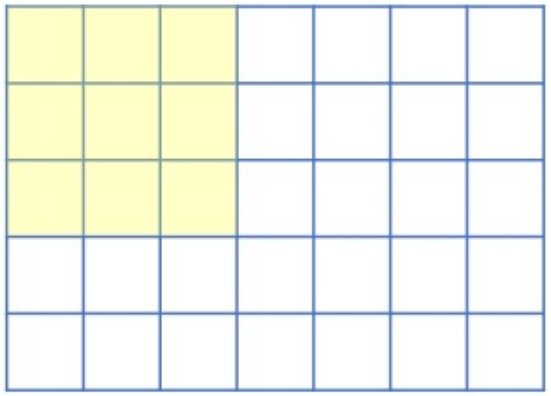
Sizes: 64^2, 128^2

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

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

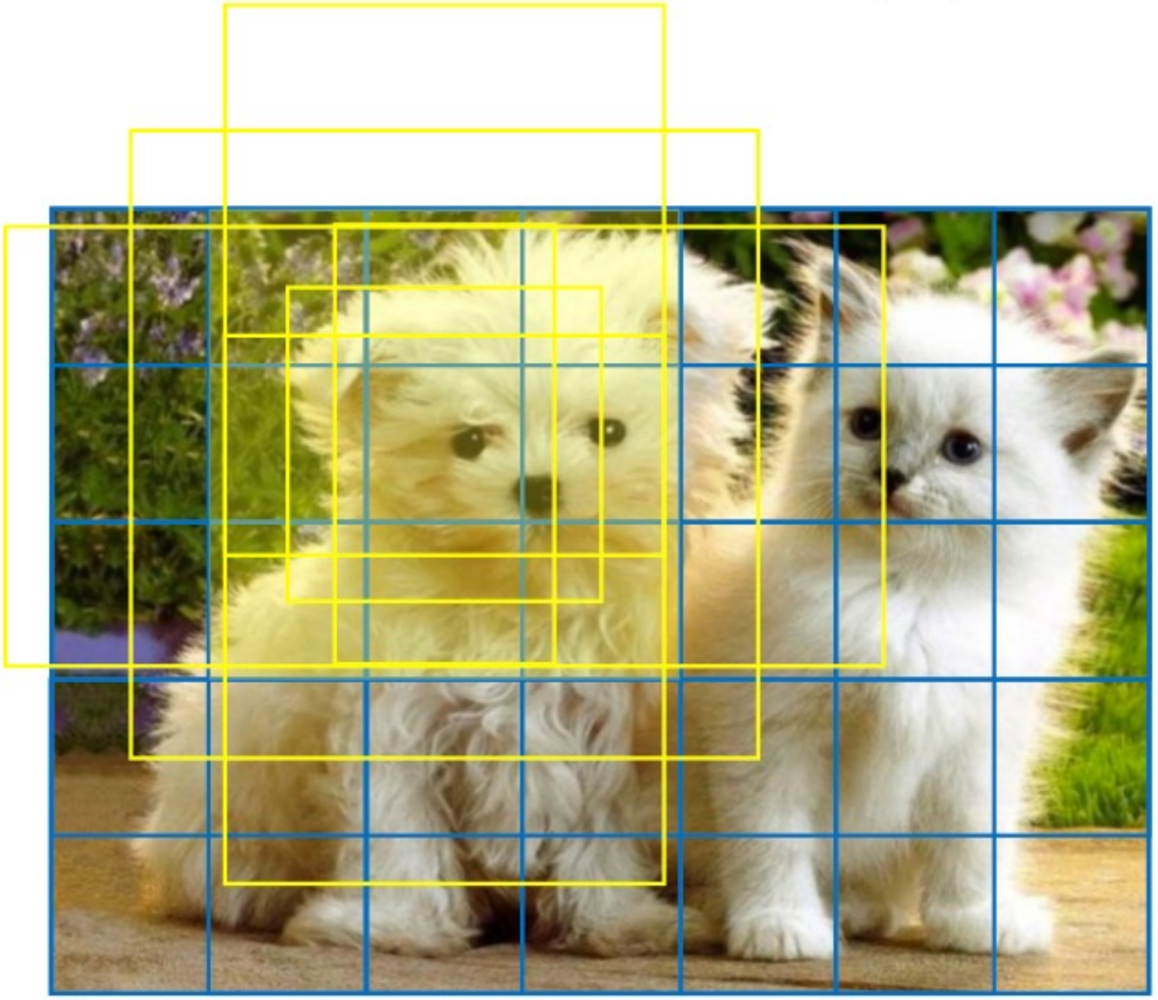


CNN
(total stride=32)



5×7

Faster R-CNN



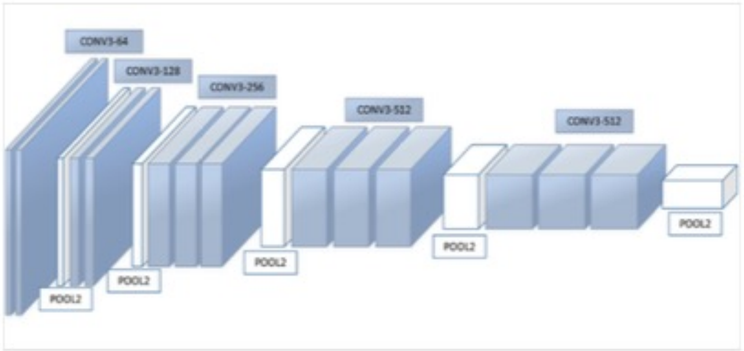
224×160

Anchor box setting:

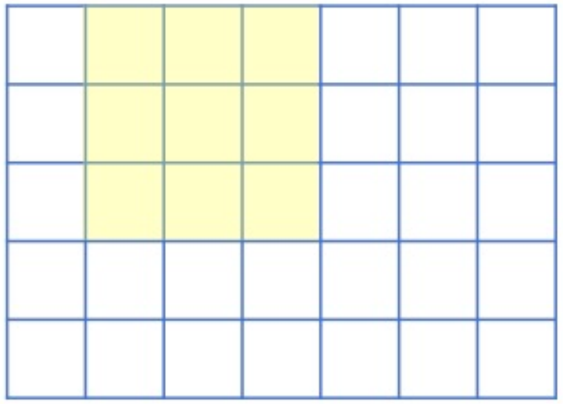
Sizes: 64^2, 128^2

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

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

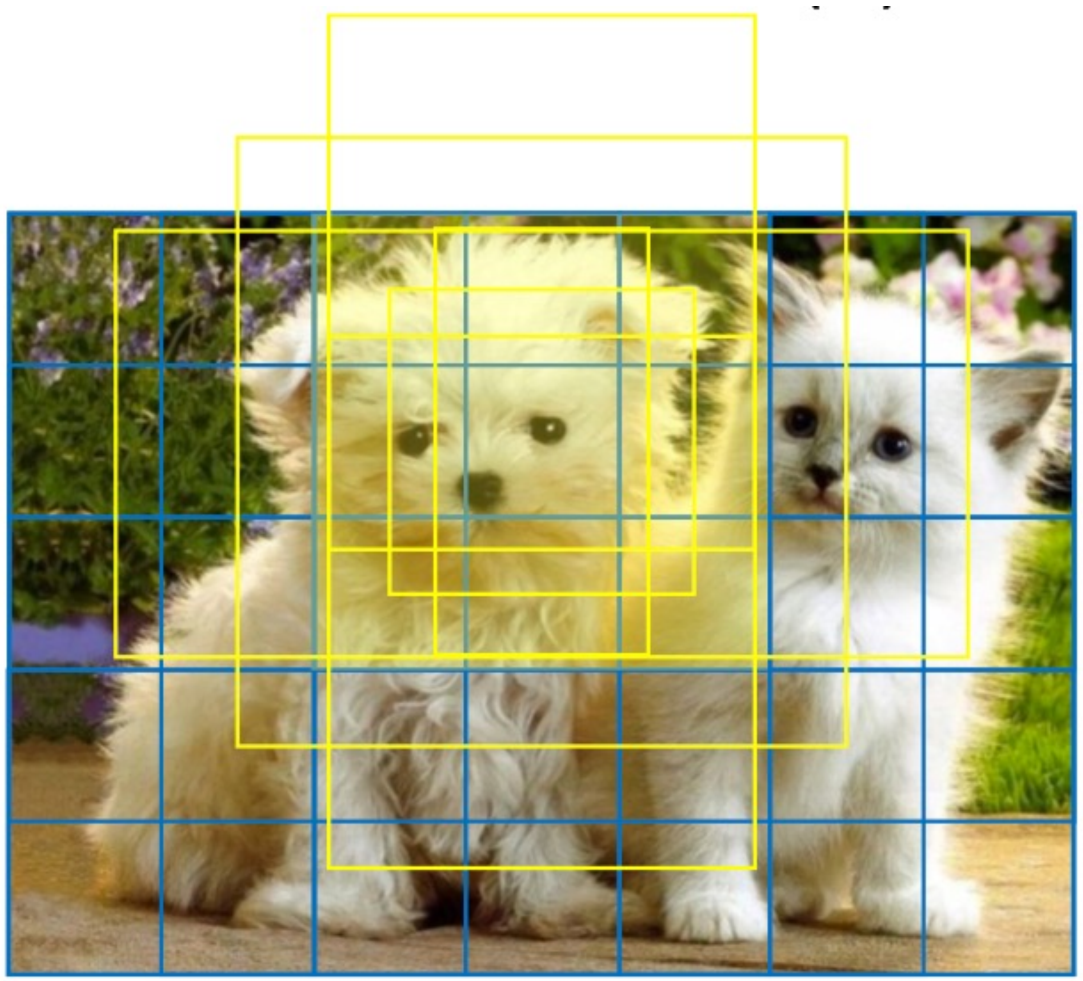


CNN
(total stride=32)



5×7

Faster R-CNN



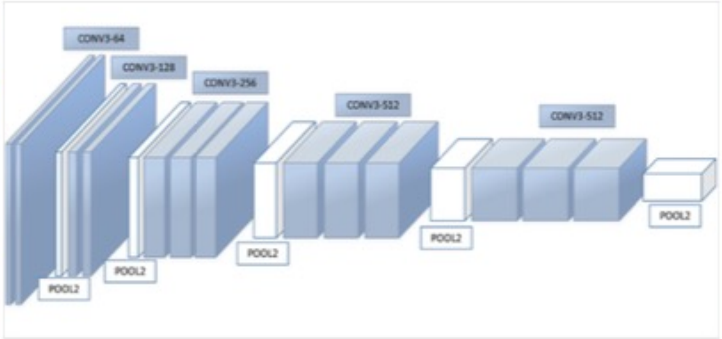
224×160

Anchor box setting:

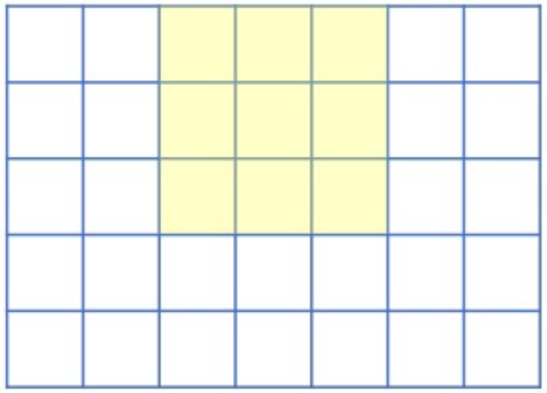
Sizes: 64^2, 128^2

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

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



CNN
(total stride=32)



5×7

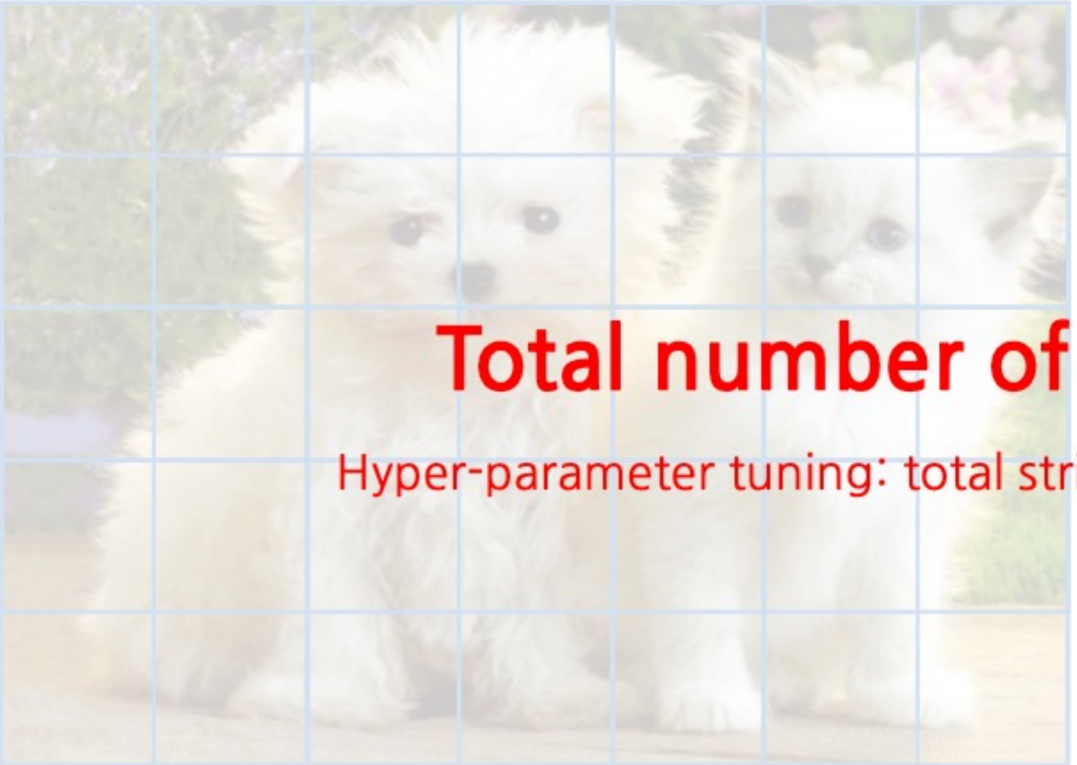
Faster R-CNN

Anchor box setting:

Sizes: 64^2, 128^2

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

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



224×160

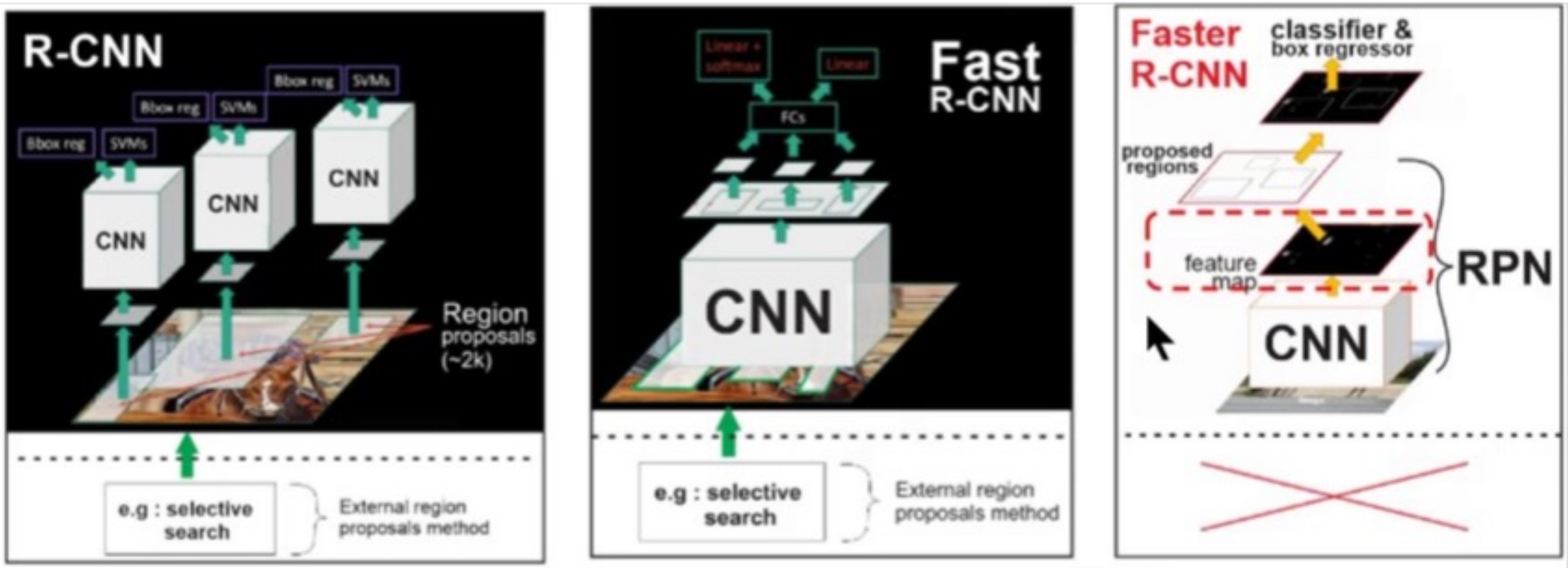
CNN
(total stride=32)

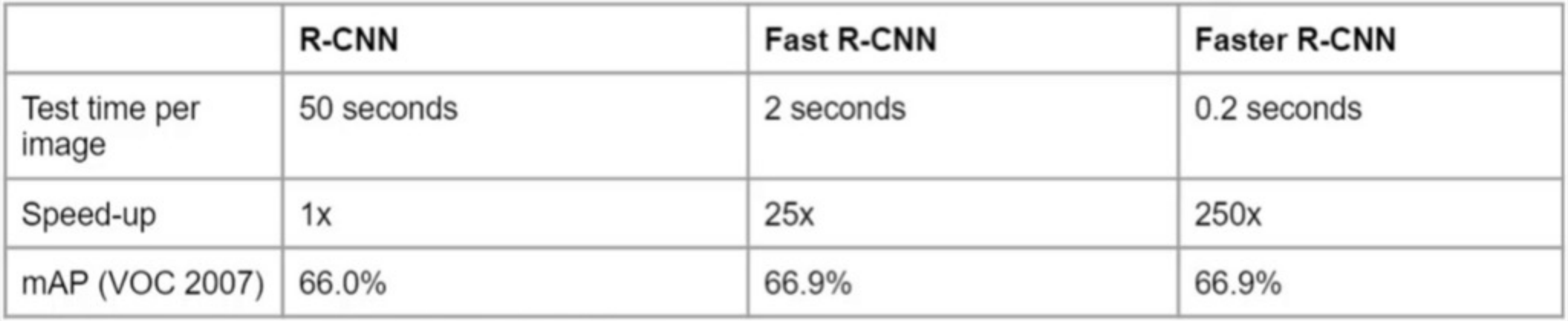
5×7

Total number of anchors: 5×7×6 = 210

Hyper-parameter tuning: total stride, anchor box setting, assigning strategy, etc.

Faster R-CNN

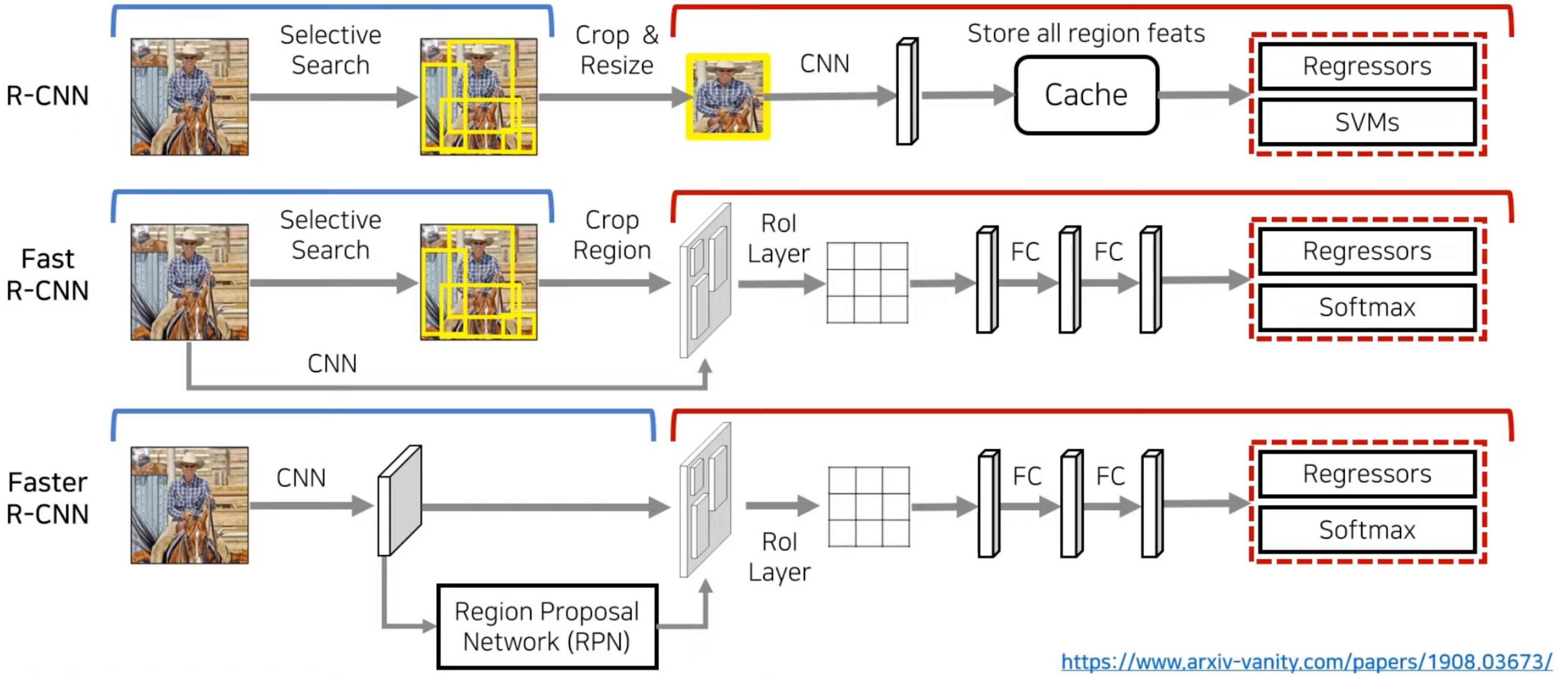




	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

출처: http://cs231n.stanford.edu

R-CNN부터 Faster RCNN까지 비교



https://www.arxiv-vanity.com/papers/1908.03673/