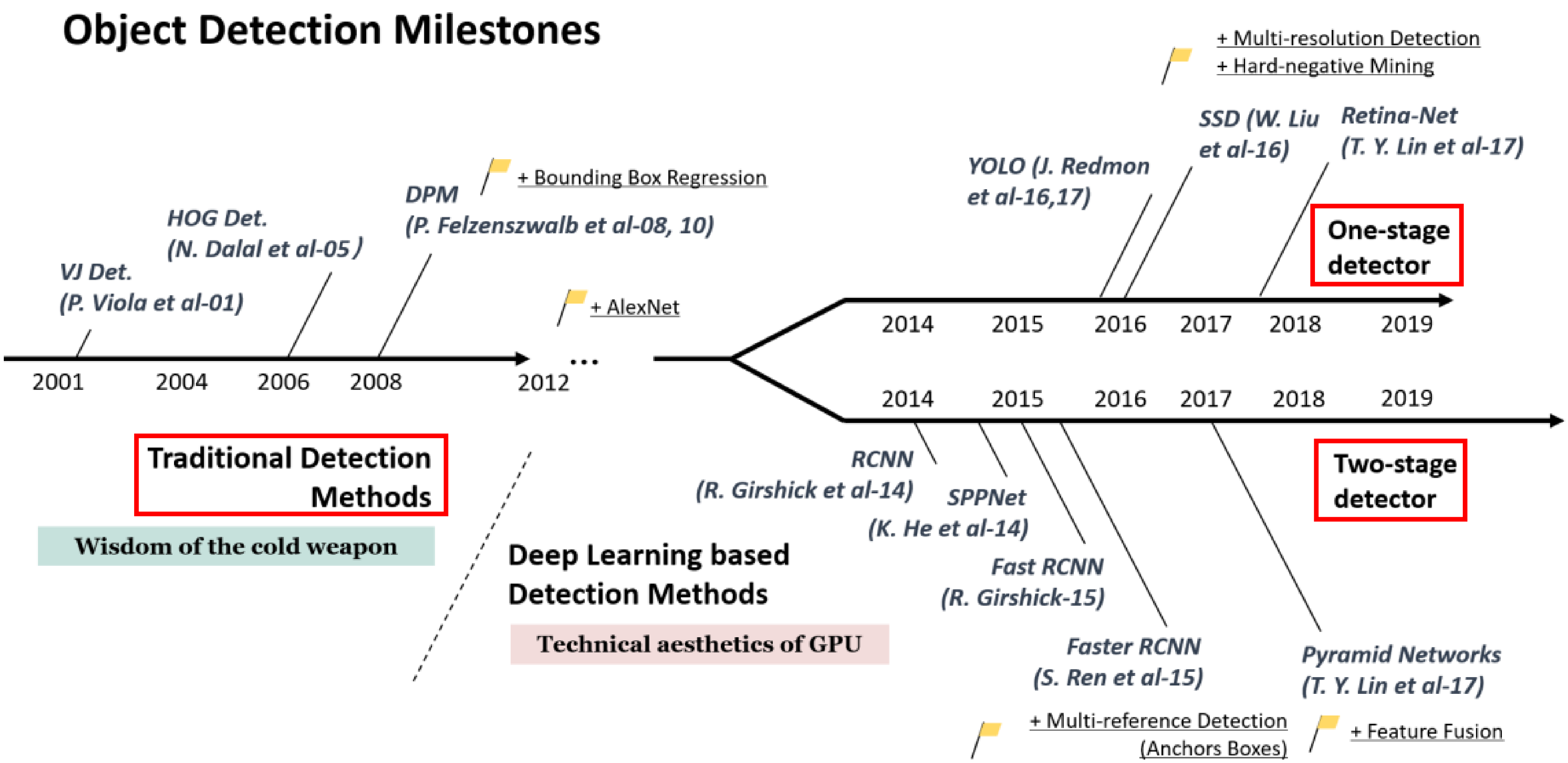
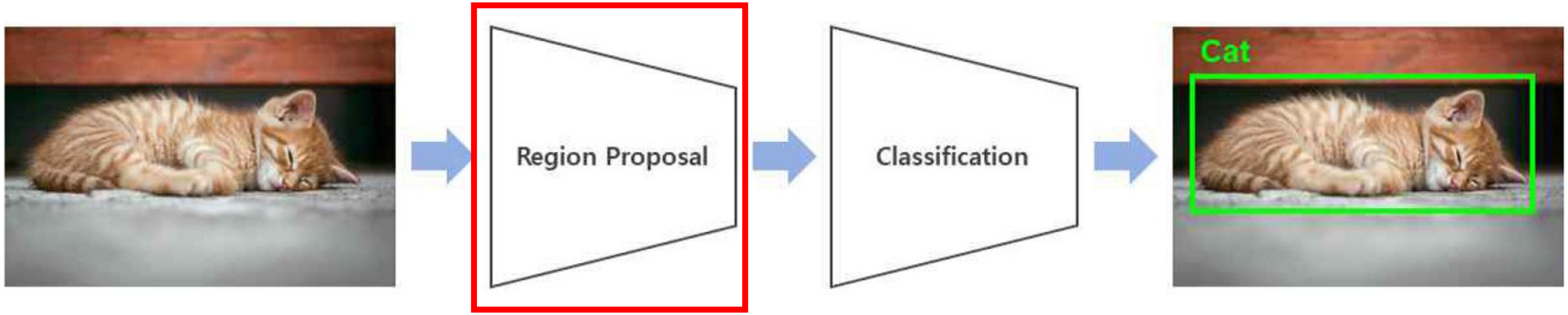
객체 검출 Ⅰ
(Two stage detector)

강사: 김 남 범 교수

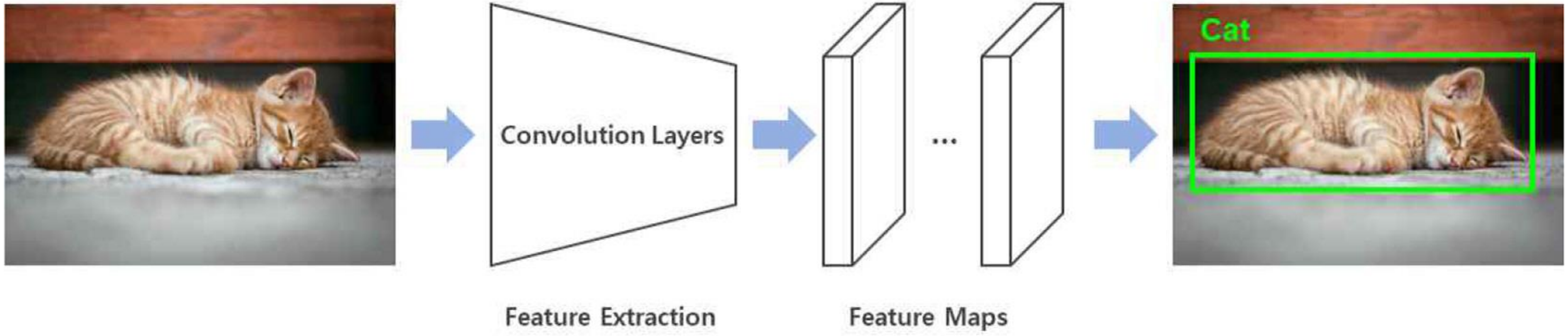
Object detection Milestones



Zou et al. 2019. Object Detection in 20 Years: A Survey



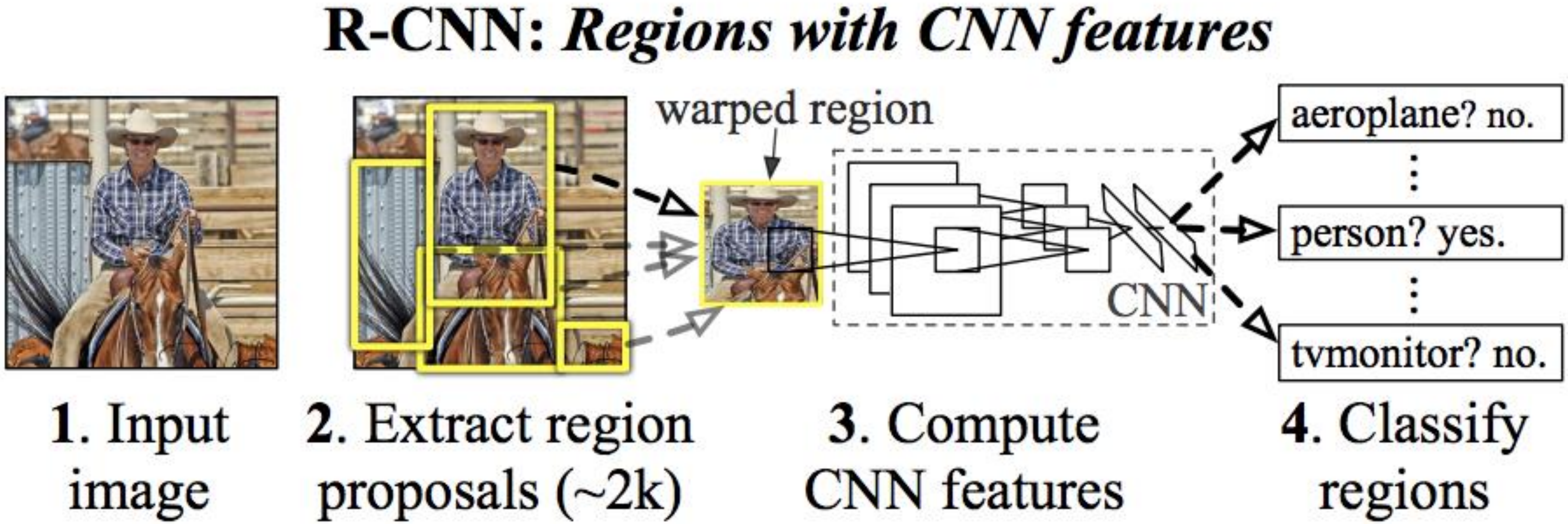
(a) 2-Stage detector



(b) 1-Stage detector

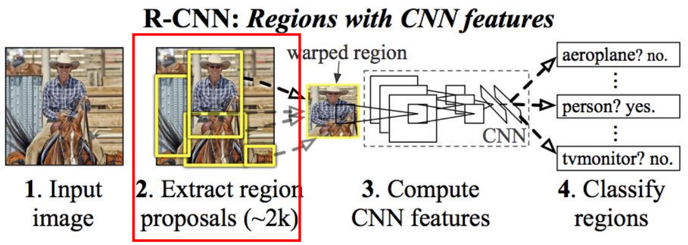
Stage에 따른 모델분류

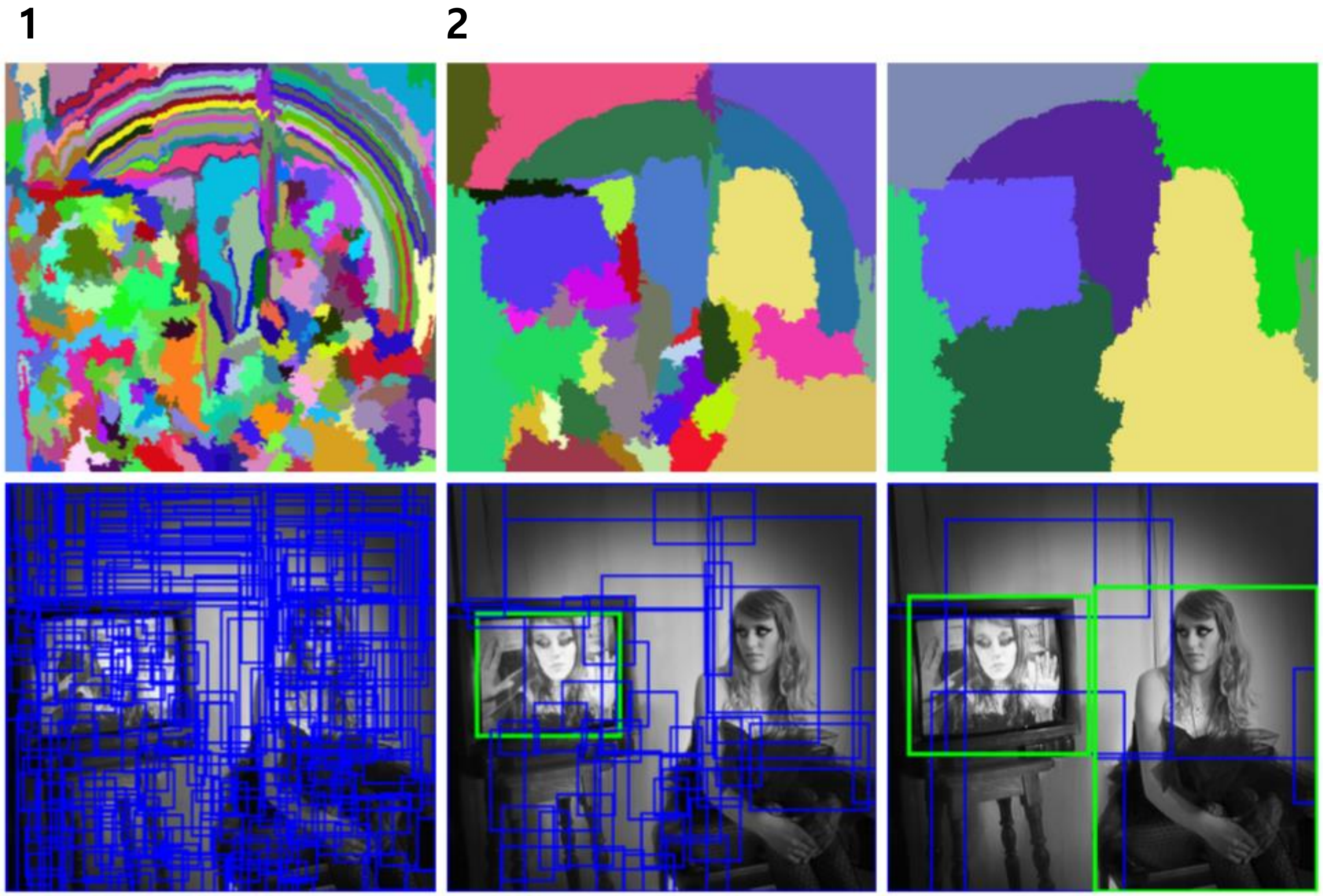
- Two stage detector
  - R-CNN (2014)
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN (Instance segmentation)
- One stage detector
  - Yolo series
    - Yolo1 – Yolo11 (Object detection)
  - SSD series
    - SSD
    - RetinaNet



[Paper] Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN: Region proposal





Initial segmentation   After some iterations   After more iterations

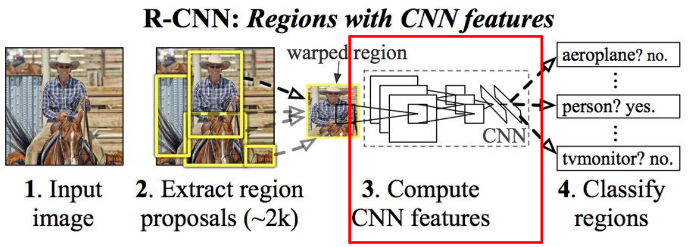
Selective search:

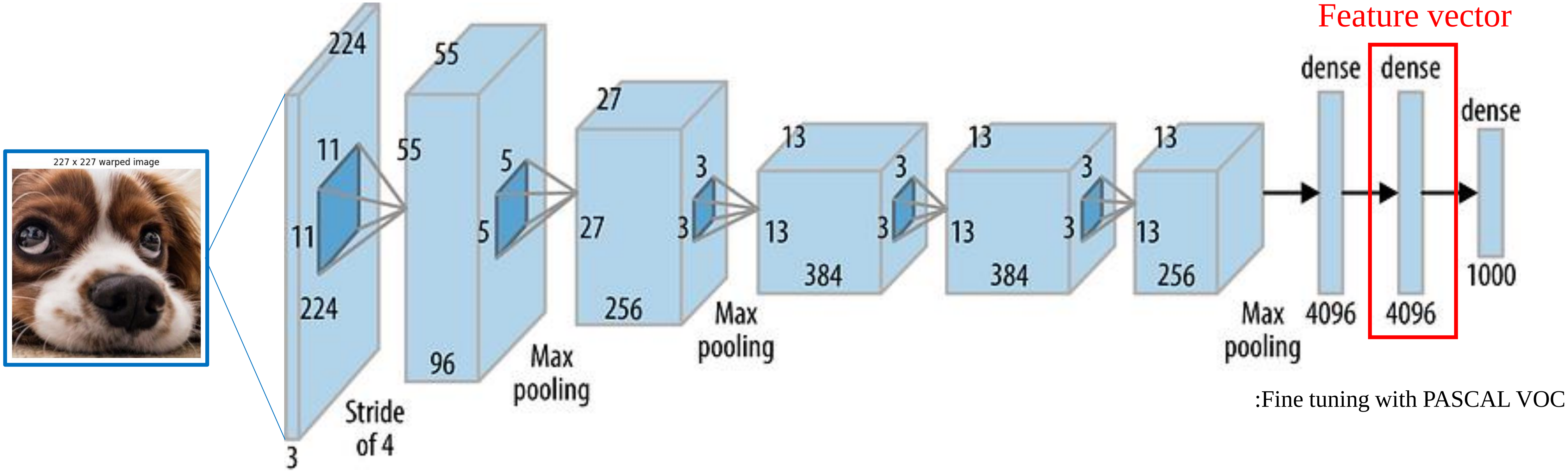
1. Image segmentation
   (Graph-based segmentation)
   - felzenszwalb segmentation

2. Merge regions
   - Greedy algorithm

3. Candidate object location (~2)

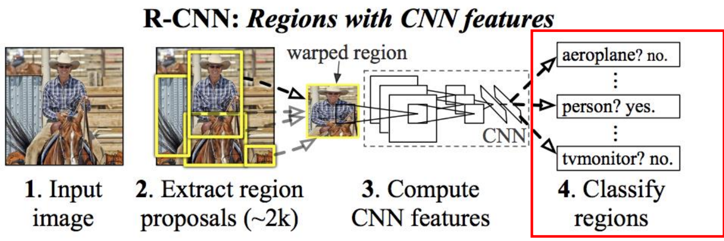
R-CNN: Feature extraction

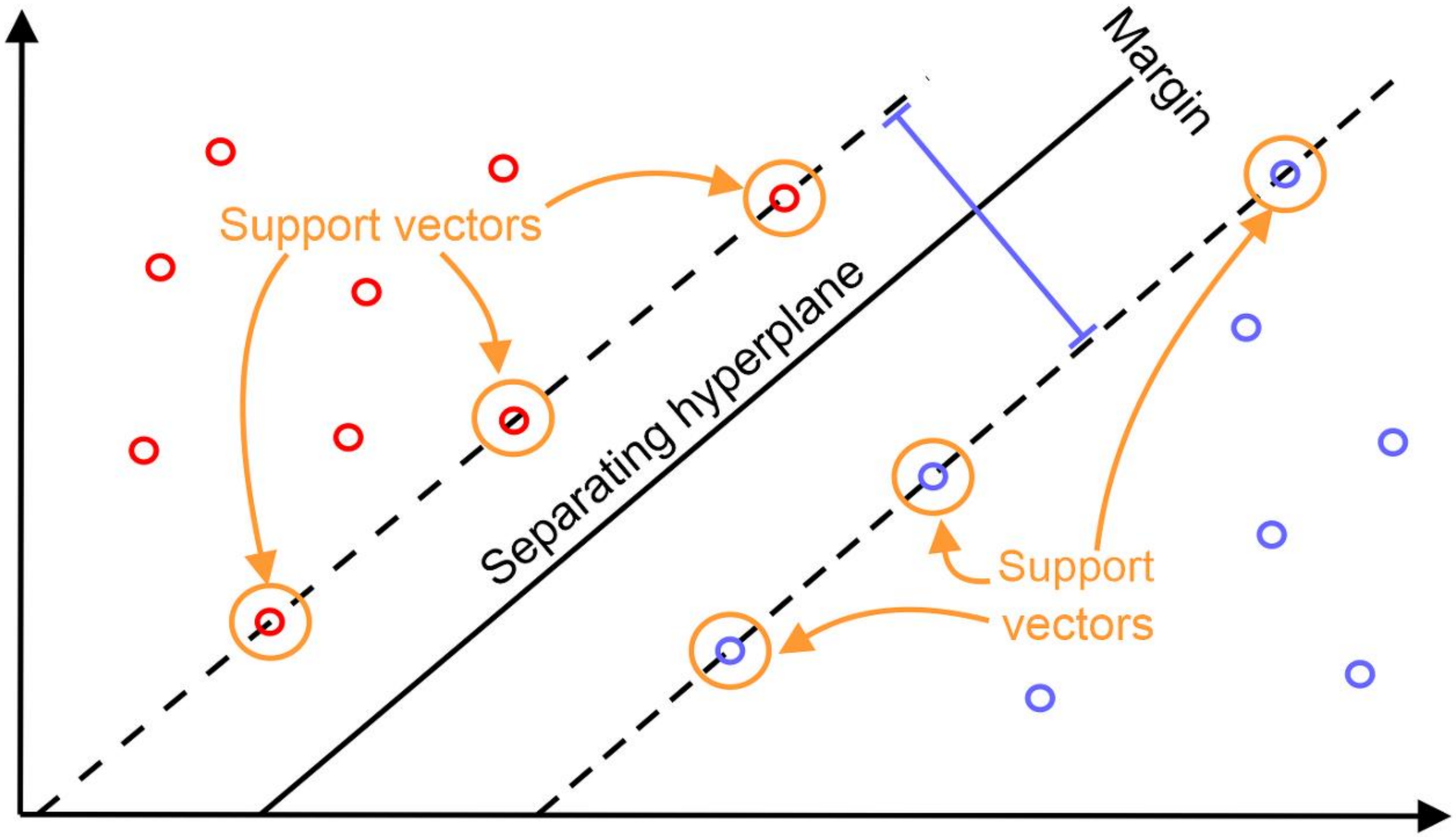




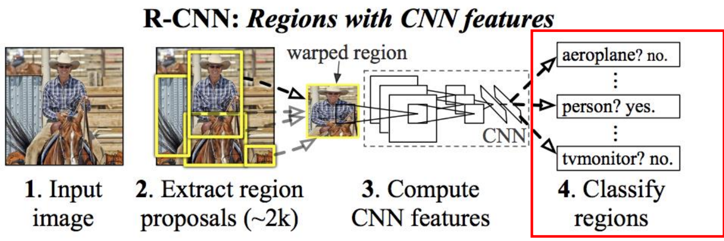
Alexnet (2012)

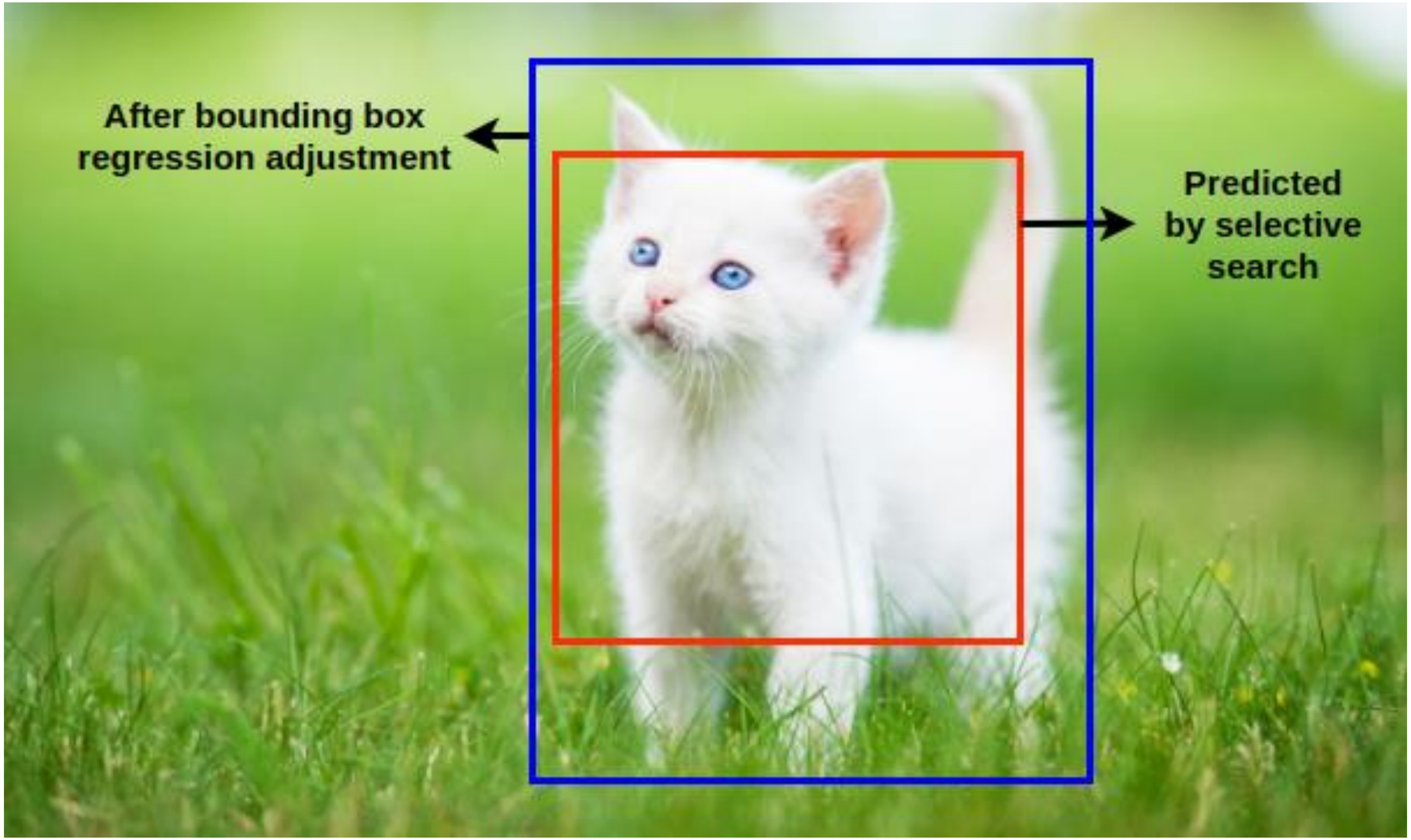
R-CNN: SVM Classification



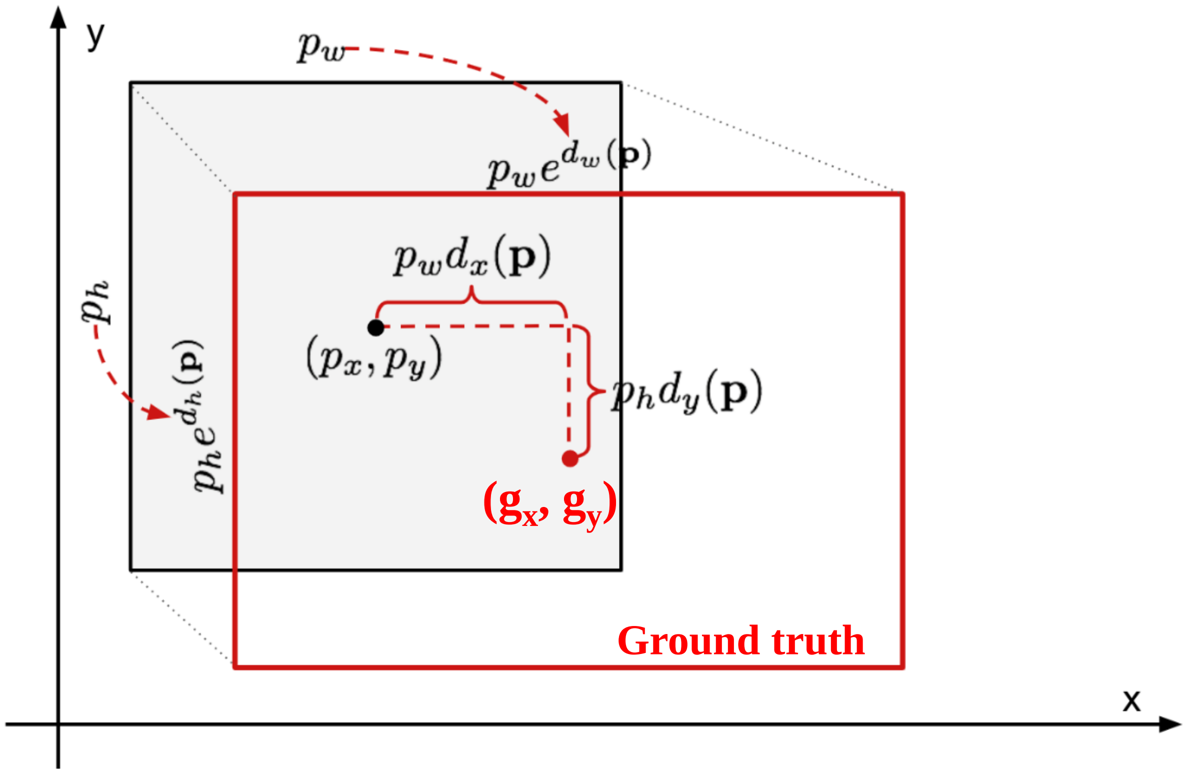


R-CNN: Bounding box regression





Bounding box regression

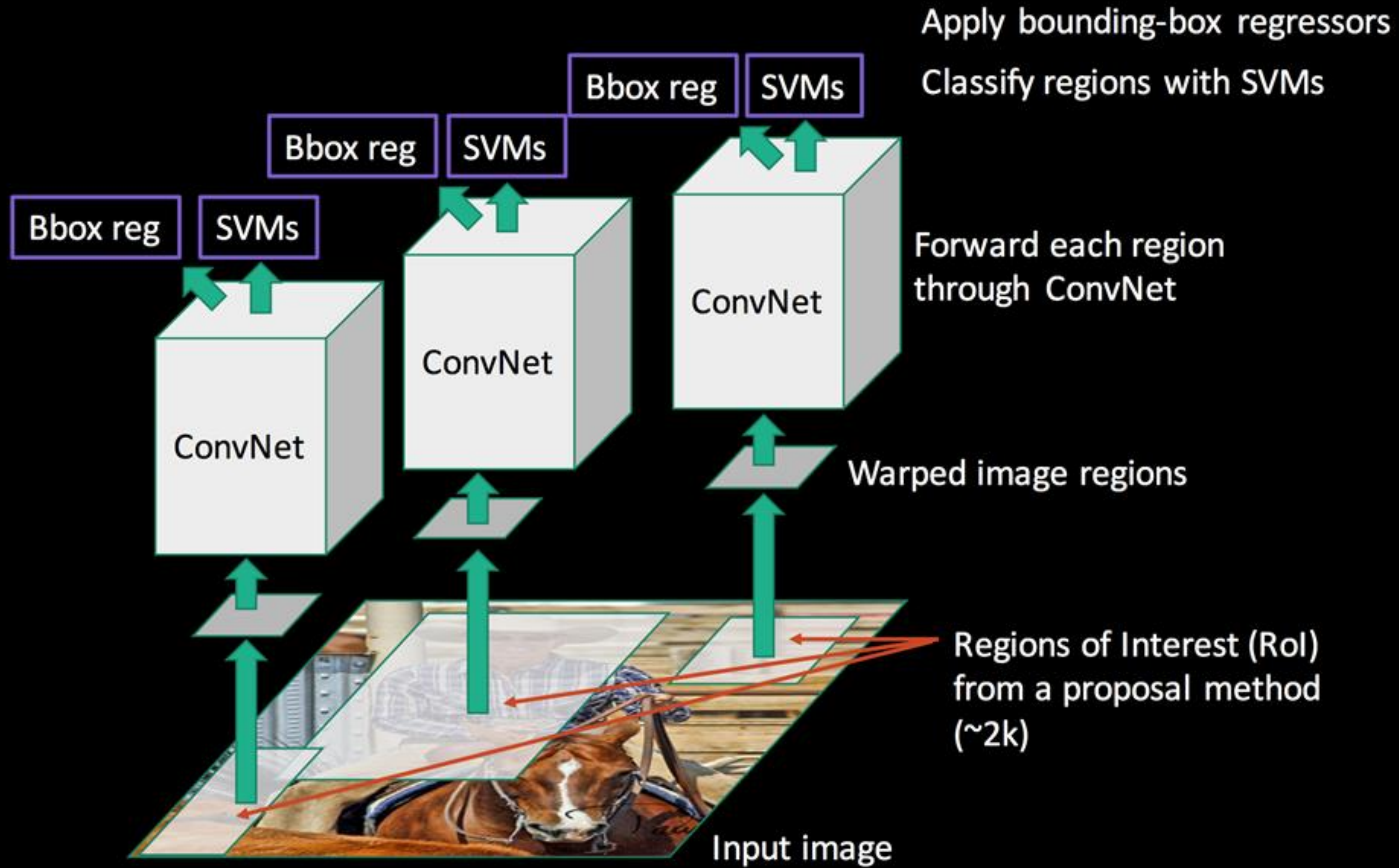


$$\hat{g}_x = p_w d_x(\mathbf{p}) + p_x$$
$$\hat{g}_y = p_h d_y(\mathbf{p}) + p_y$$
$$\hat{g}_w = p_w \exp(d_w(\mathbf{p}))$$
$$\hat{g}_h = p_h \exp(d_h(\mathbf{p}))$$

$$t_x = (g_x - p_x)/p_w$$
$$t_y = (g_y - p_y)/p_h$$
$$t_w = \log(g_w/p_w)$$
$$t_h = \log(g_h/p_h)$$

Minimizing SSE loss =

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$



Girshick et al. CVPR14.

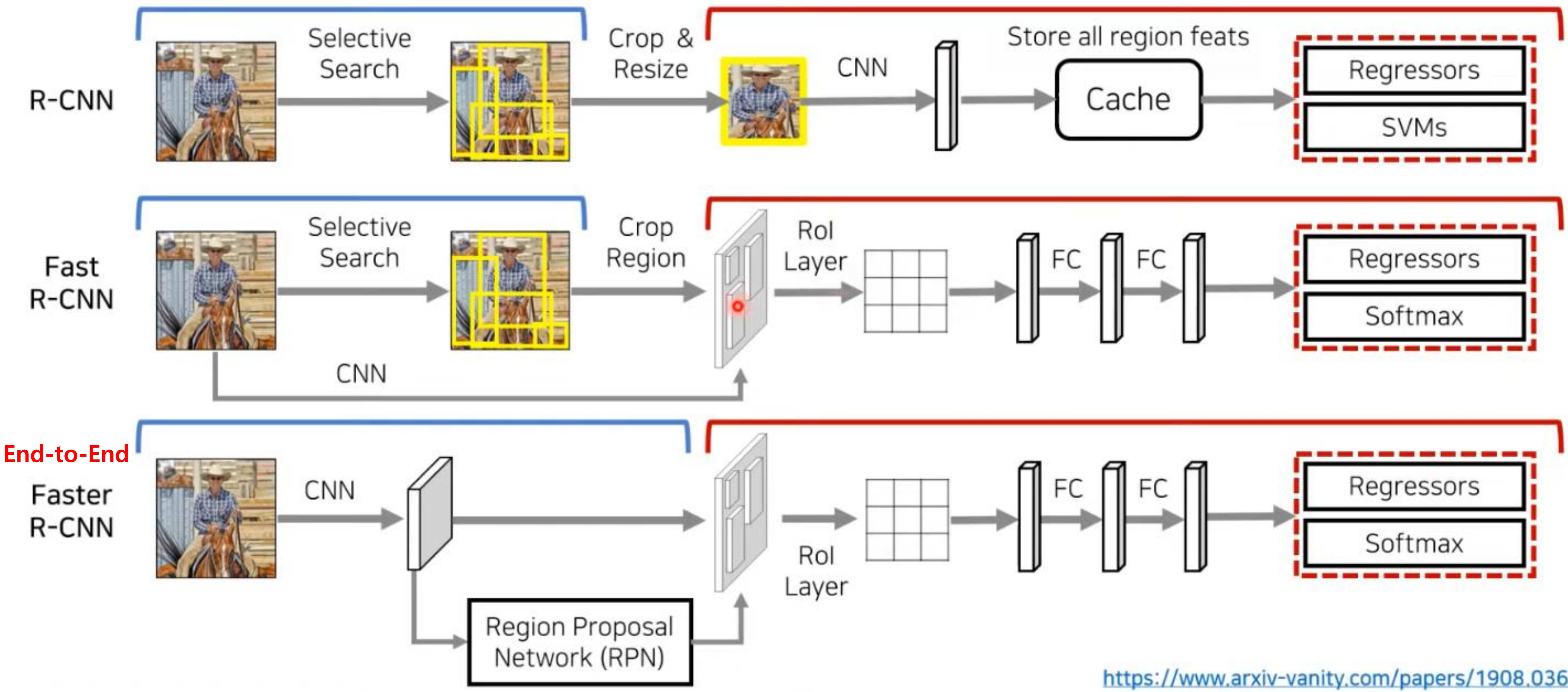
R-CNN의 장단점

- 높은 Detection 정확도

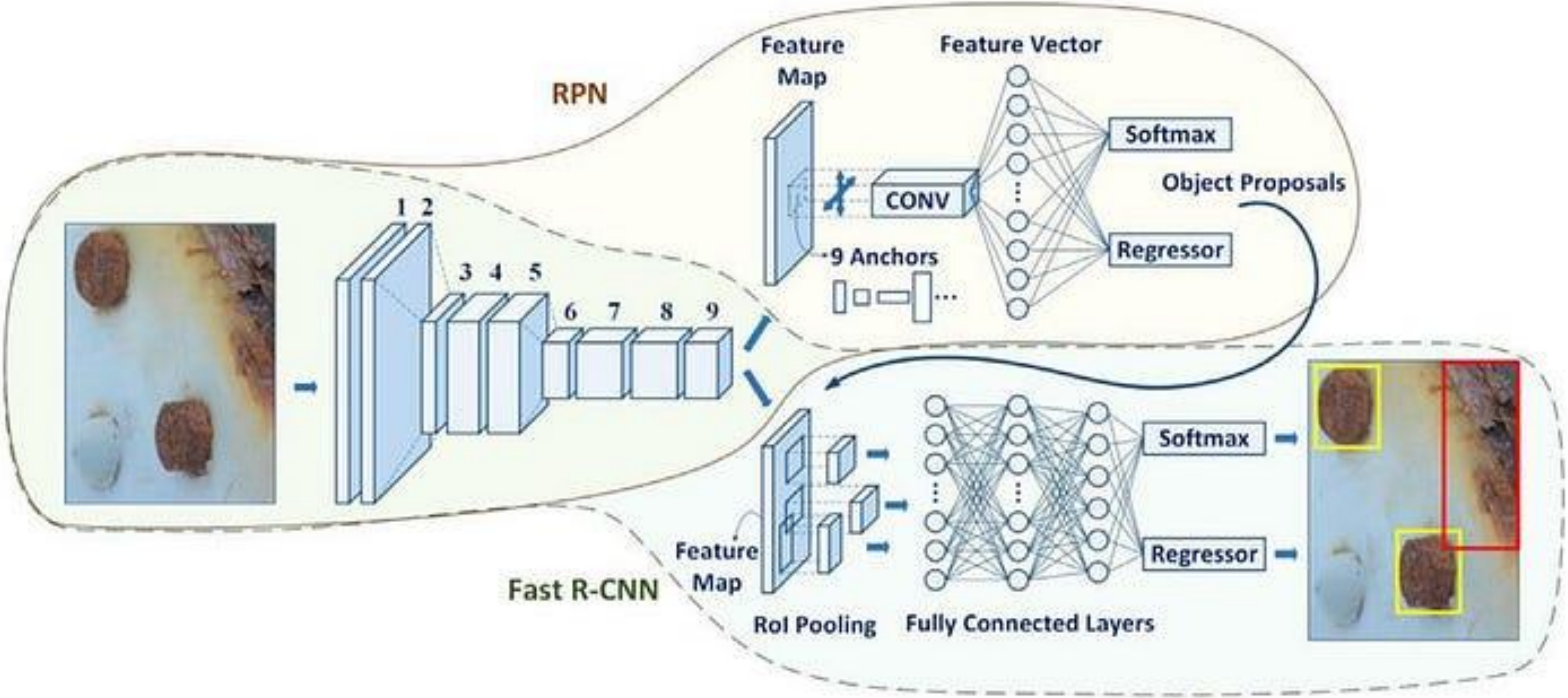
- 2000개씩 생성된 region 이미지를 CNN Feature map 생성
  - 너무 느림
  - 이미지 한 장당 50초 소요

- End-to-End 학습이 안됨
  - CNN Feature Extractor + SVM and Bounding box regressor

R-CNN, Fast R-CNN, Faster R-CNN

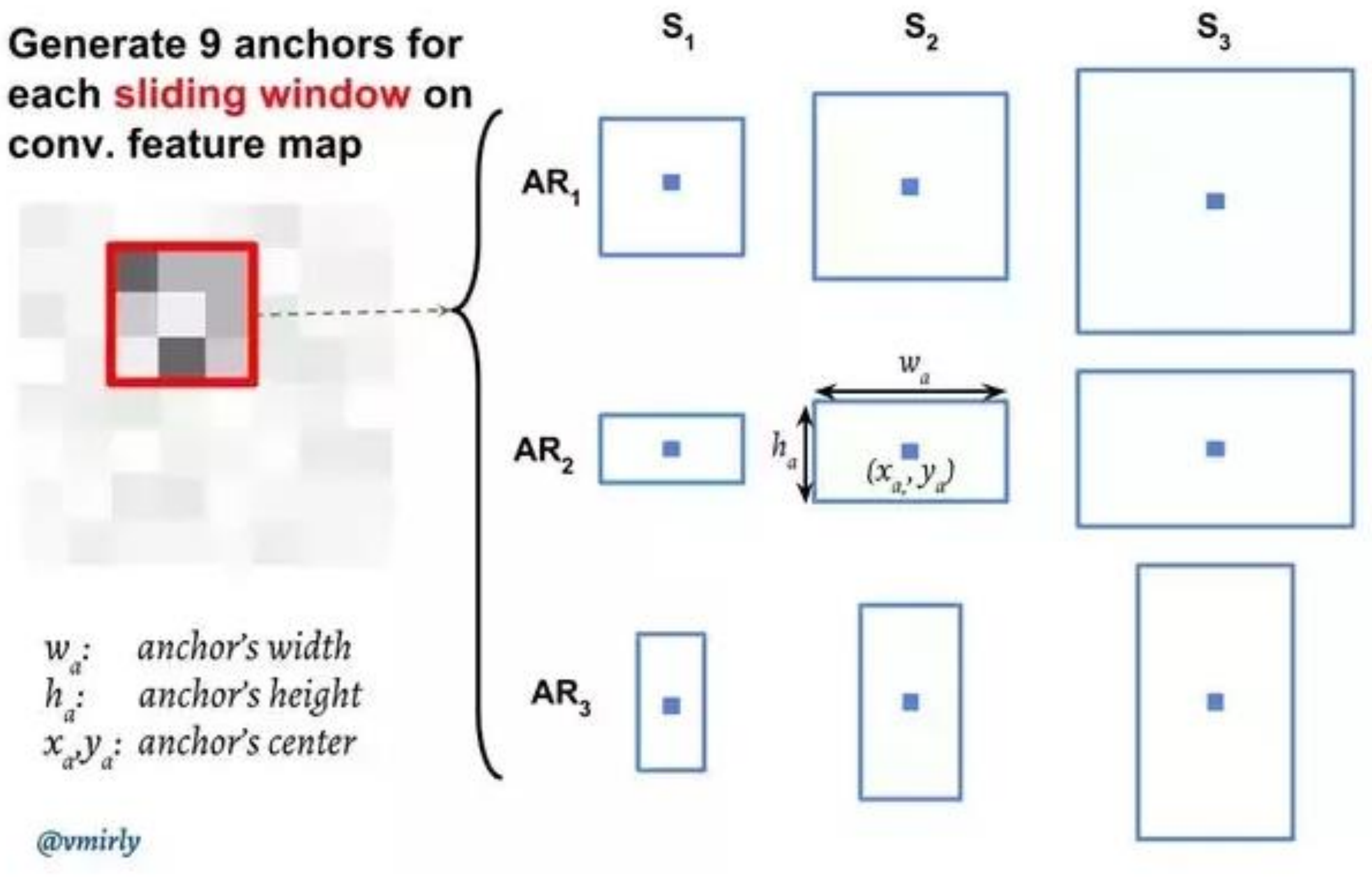


https://www.arxiv-vanity.com/papers/1908.03673/



RPN
Feature Map
Feature Vector
CONV
9 Anchors
Softmax
Regressor
Object Proposals
1 2
3 4 5
6 7 8 9
Feature Map
RoI Pooling
Fully Connected Layers
Softmax
Regressor
Fast R-CNN

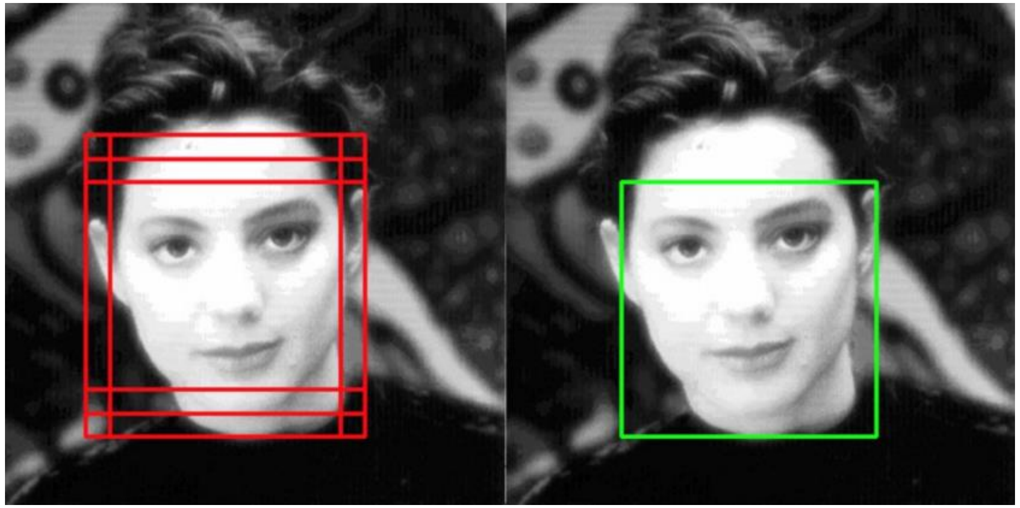
RPN (Region proposal network)

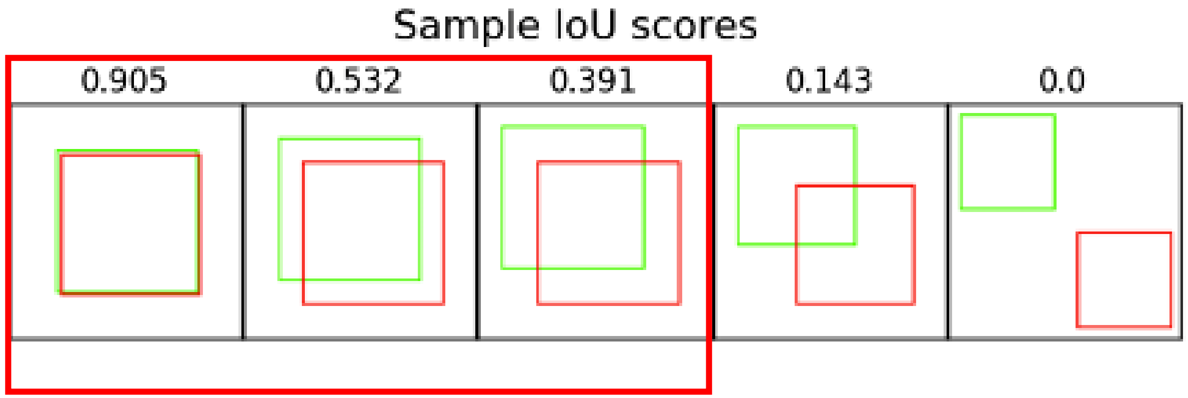


- **Anchor target generation**
- **Calculate the IOU of GT boxes**
- **Proposal generation**

https://www.researchgate.net/publication/325385109_Yield_Estimation_using_faster_R-CNN

Non Maximum suppression

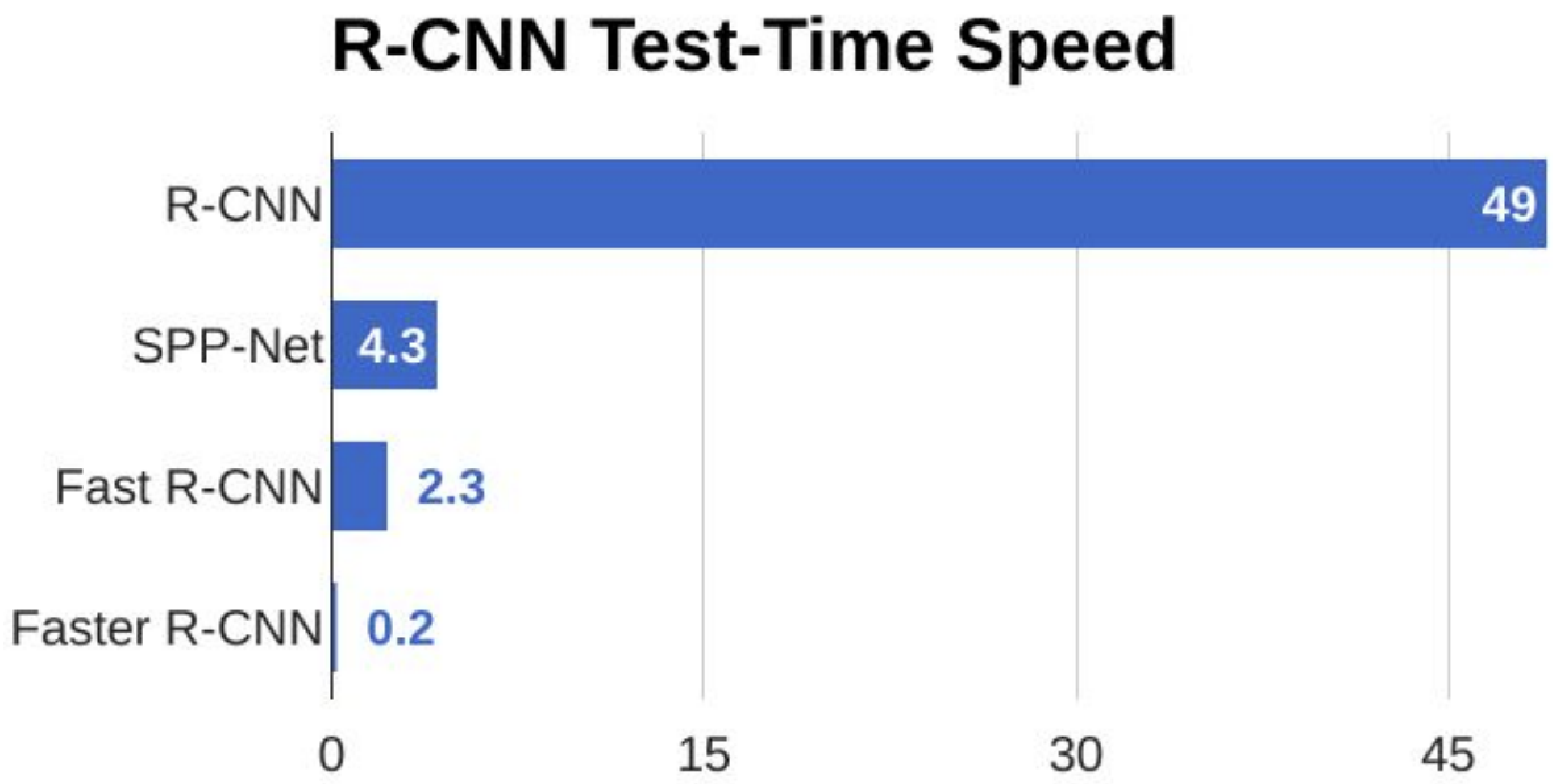




R-CNN 계열 성능비교

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**

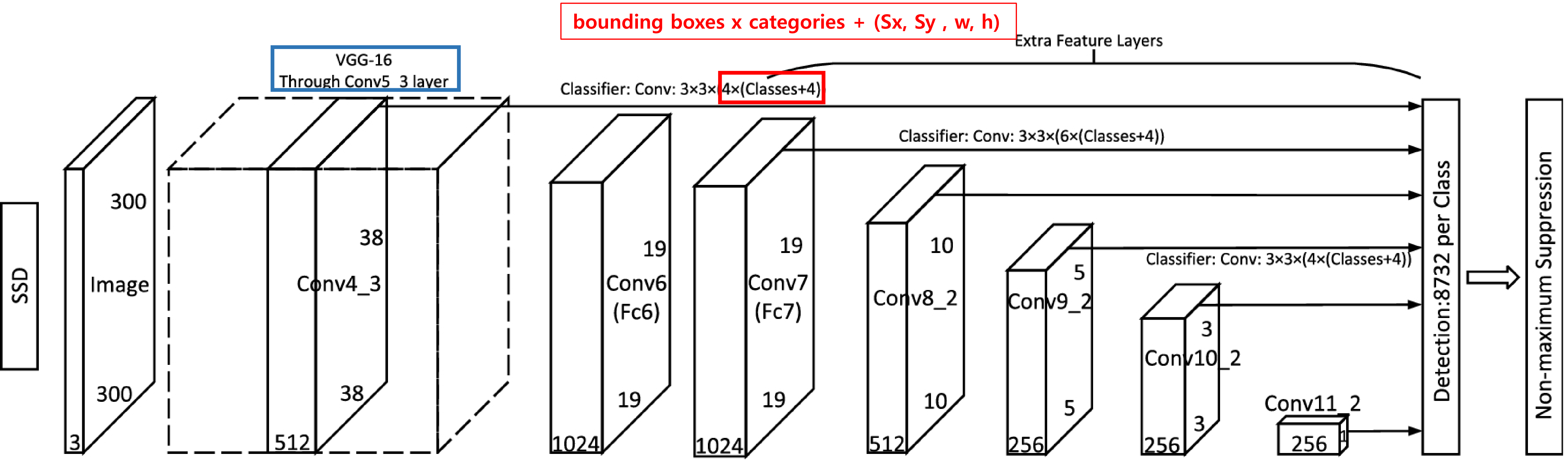
Detection mAP on PASCAL VOC 2007 and 2012, with VGG-16 pre-trained on ImageNet Dataset


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

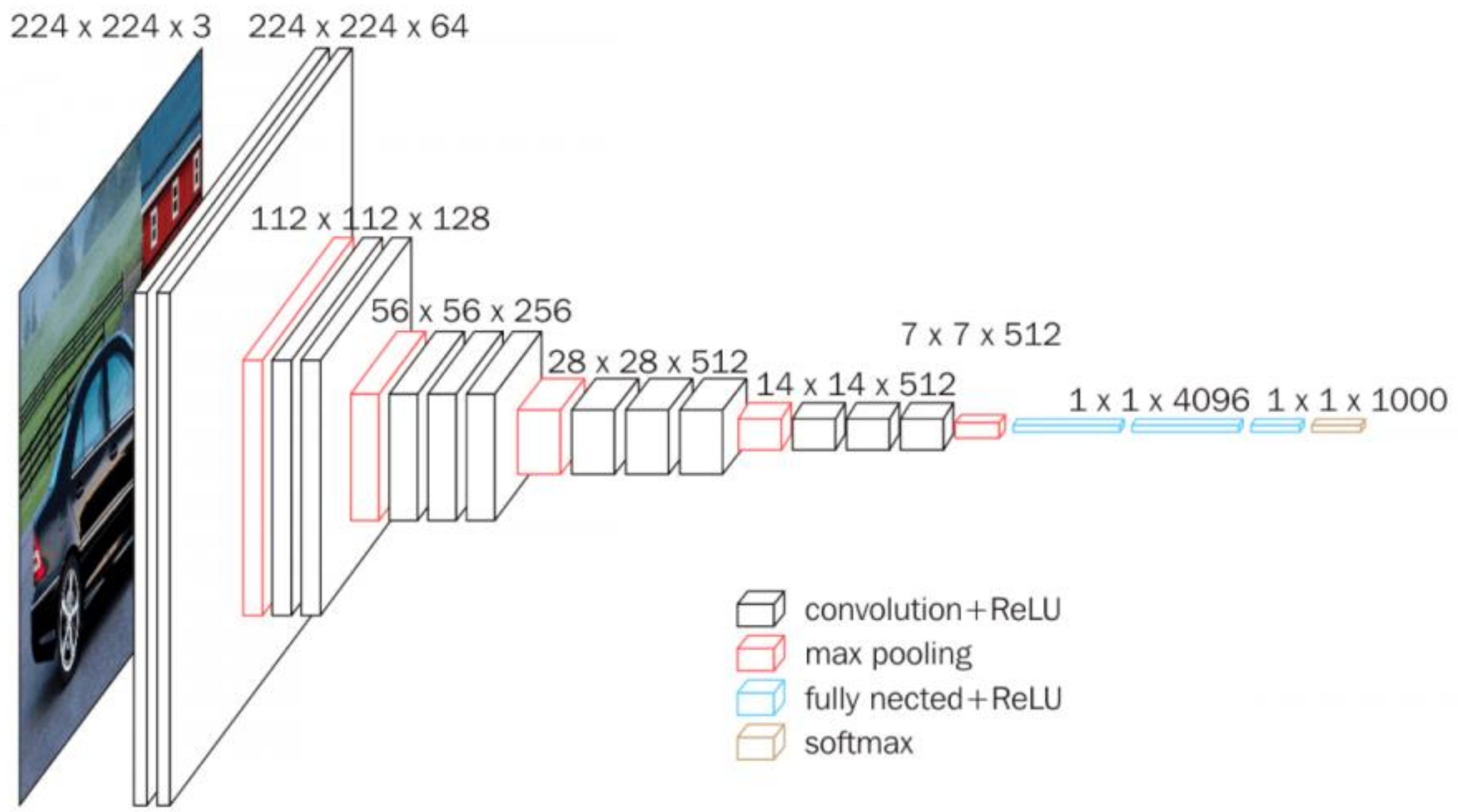
객체 검출 Ⅱ
(Real time detector)

강사: 김 남 범 교수

Single shot multibox detector (SSD, 2016)

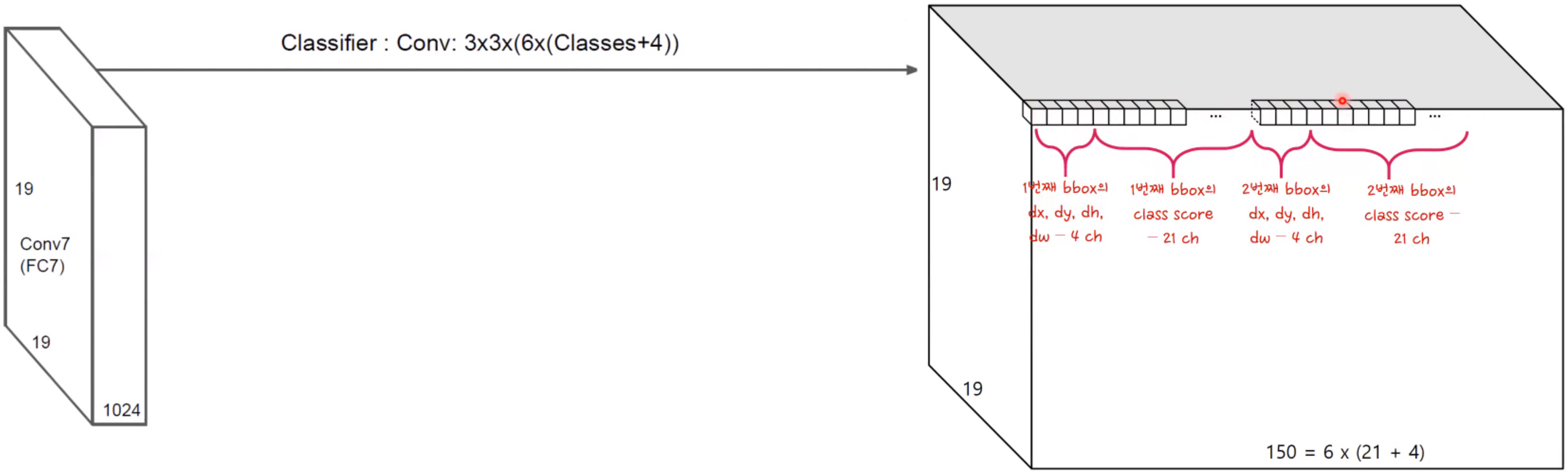


VGG16 (2014)

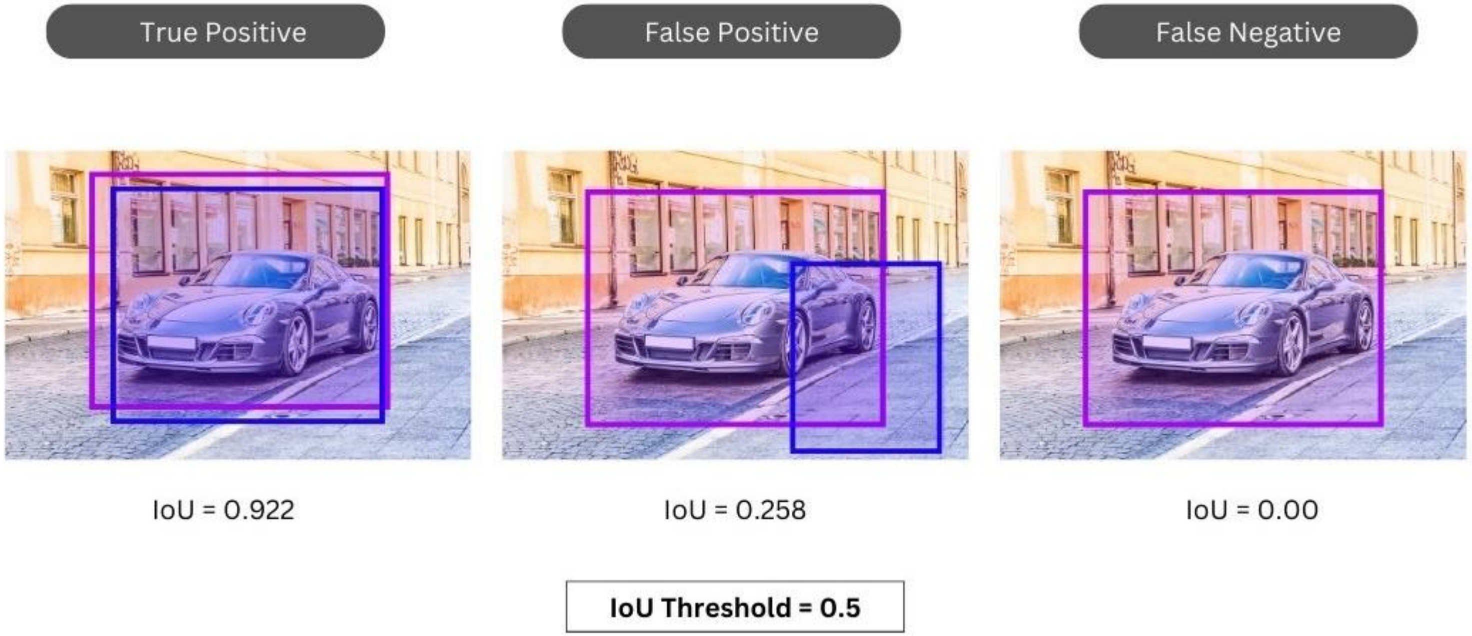


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

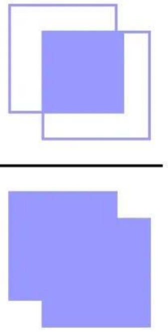
Convolutional predictors

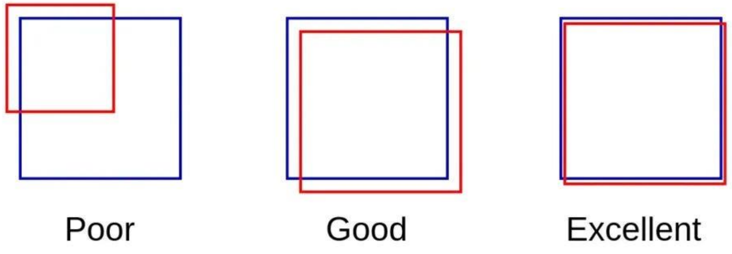


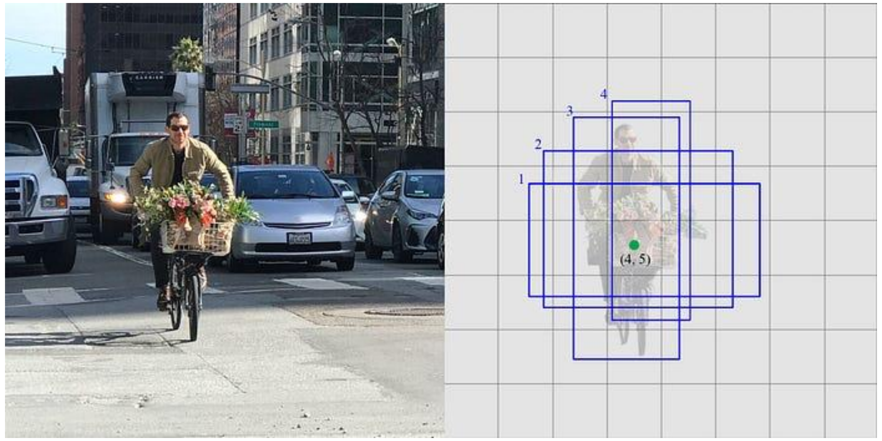
Jaccard overlap



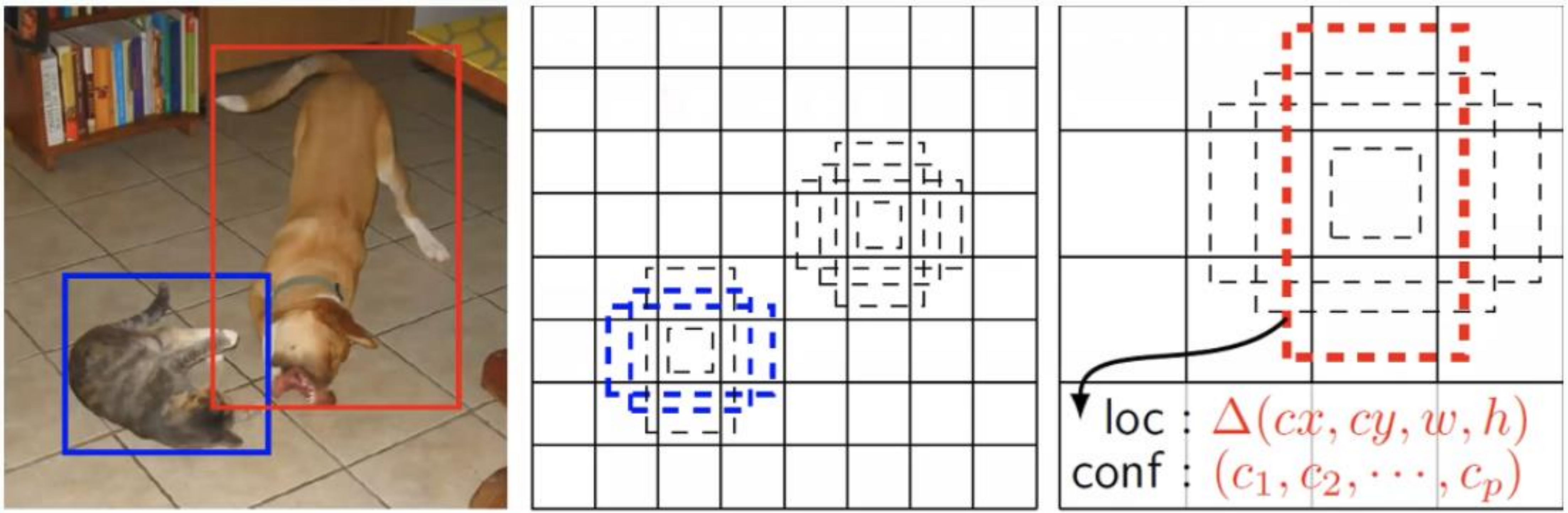
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



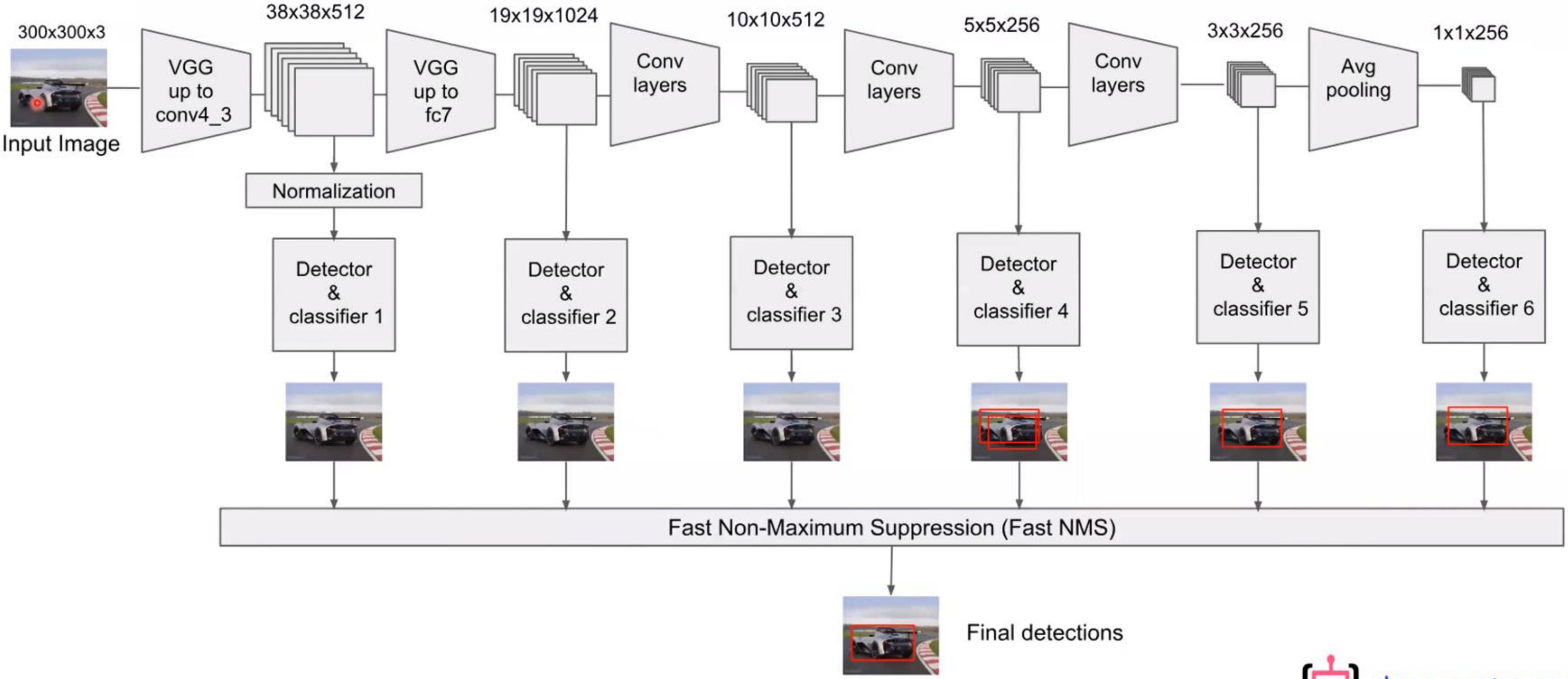




Scales and aspect ratios for default boxes

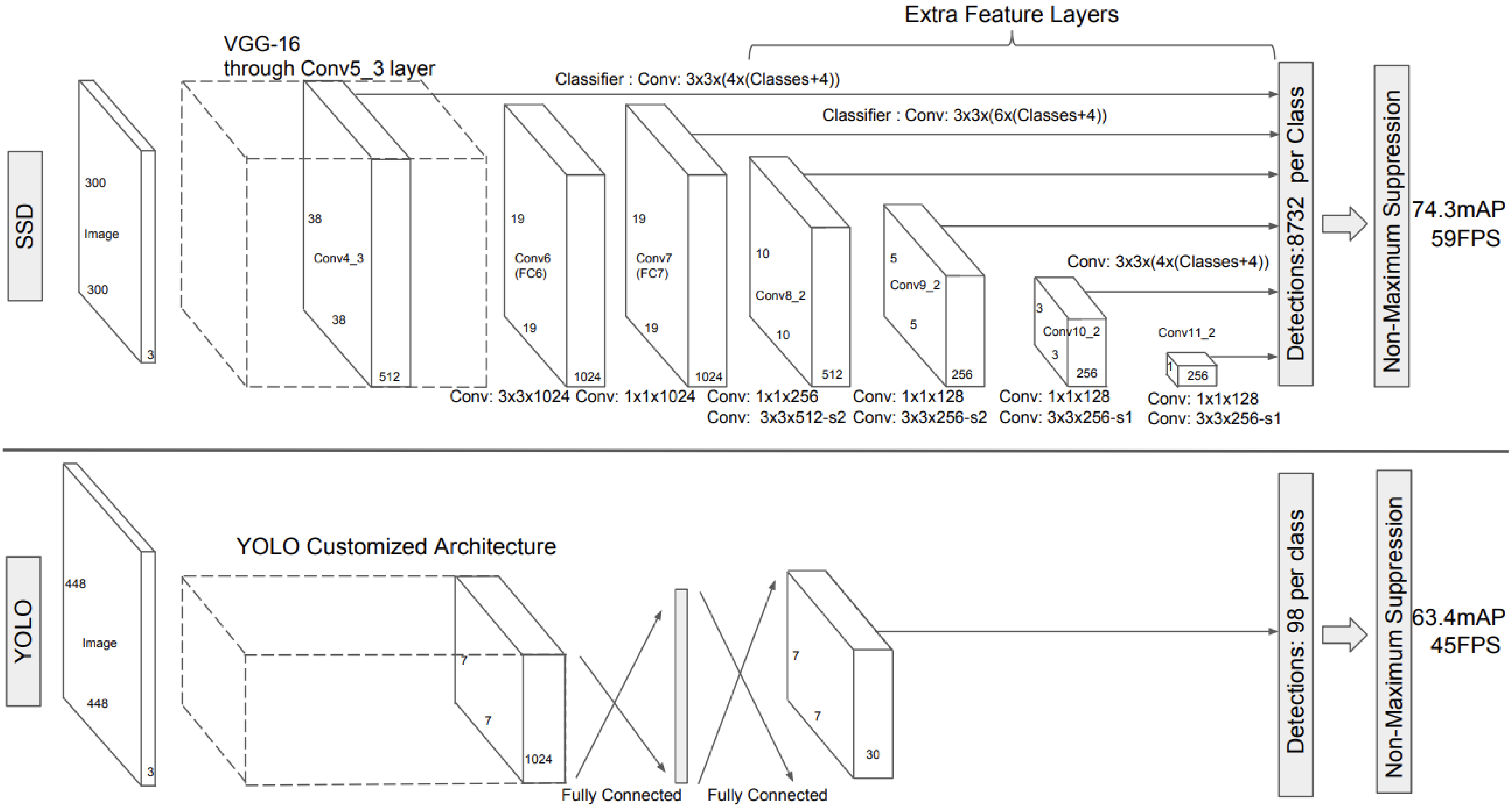


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map



deepsystems.io

Slide from https://goo.gl/NsP6Wg



SSD
Extra Feature Layers
VGG-16
through Conv5_3 layer
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
300
Image
300
3
38
Conv4_3
38
512
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
Conv11_2
1
256
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
YOLO
YOLO Customized Architecture
448
Image
448
3
7
7
1024
Fully Connected
Fully Connected
7
7
30
Detections: 98 per class
Non-Maximum Suppression
63.4mAP
45FPS

Yolo architecture (You Only Look Once, 2015)

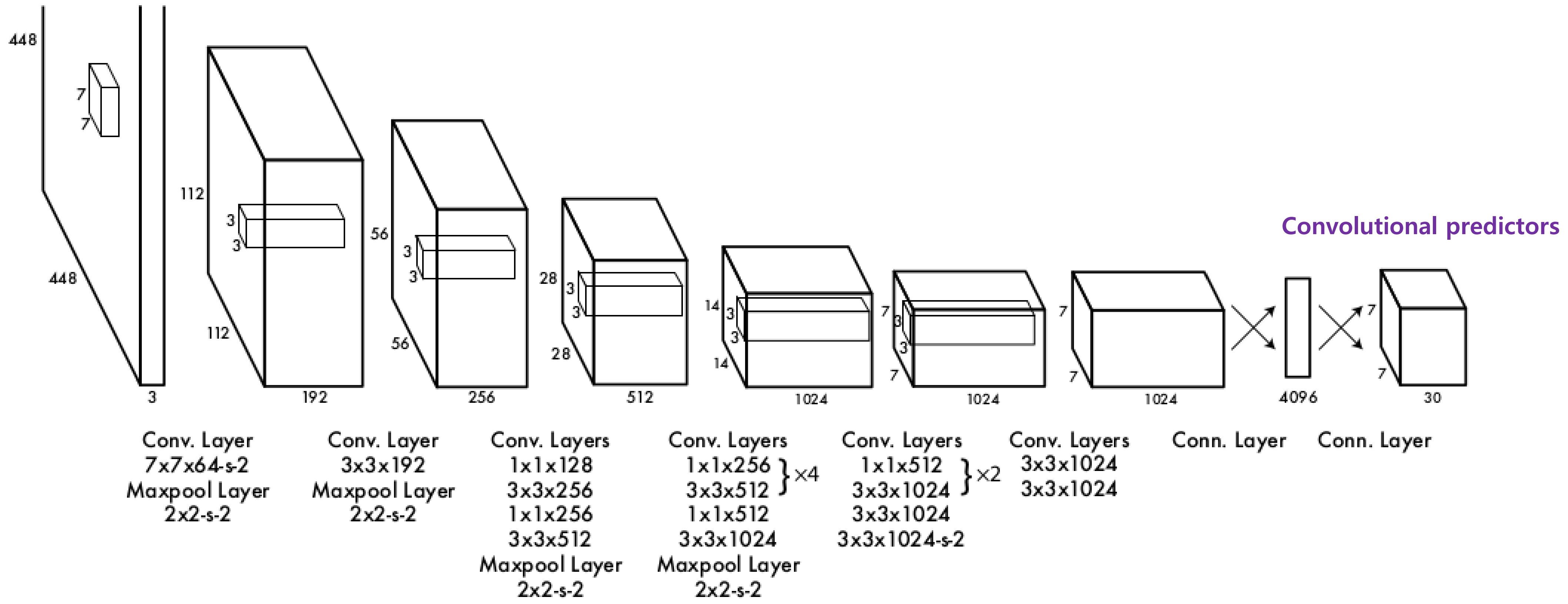


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Unified detection

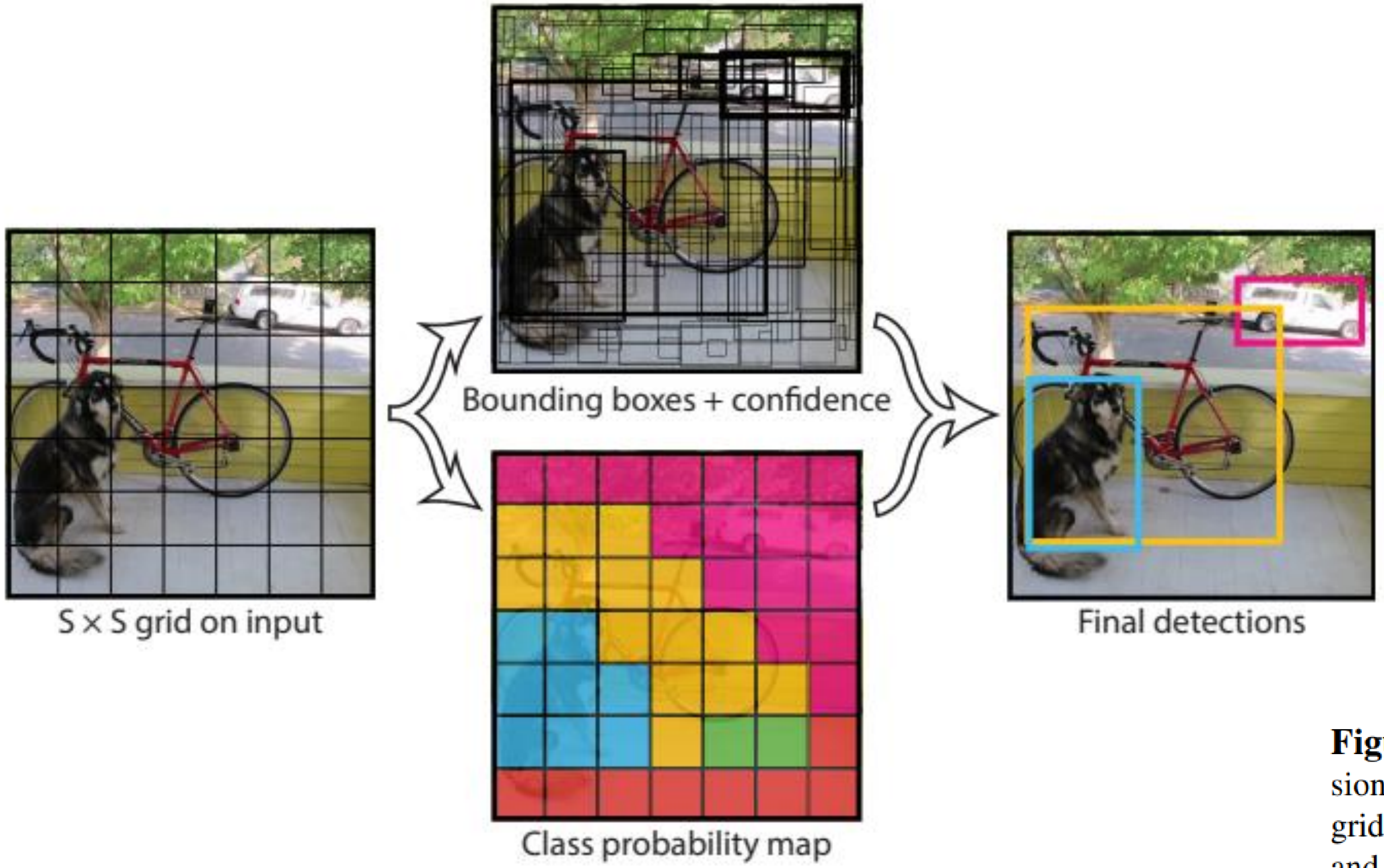
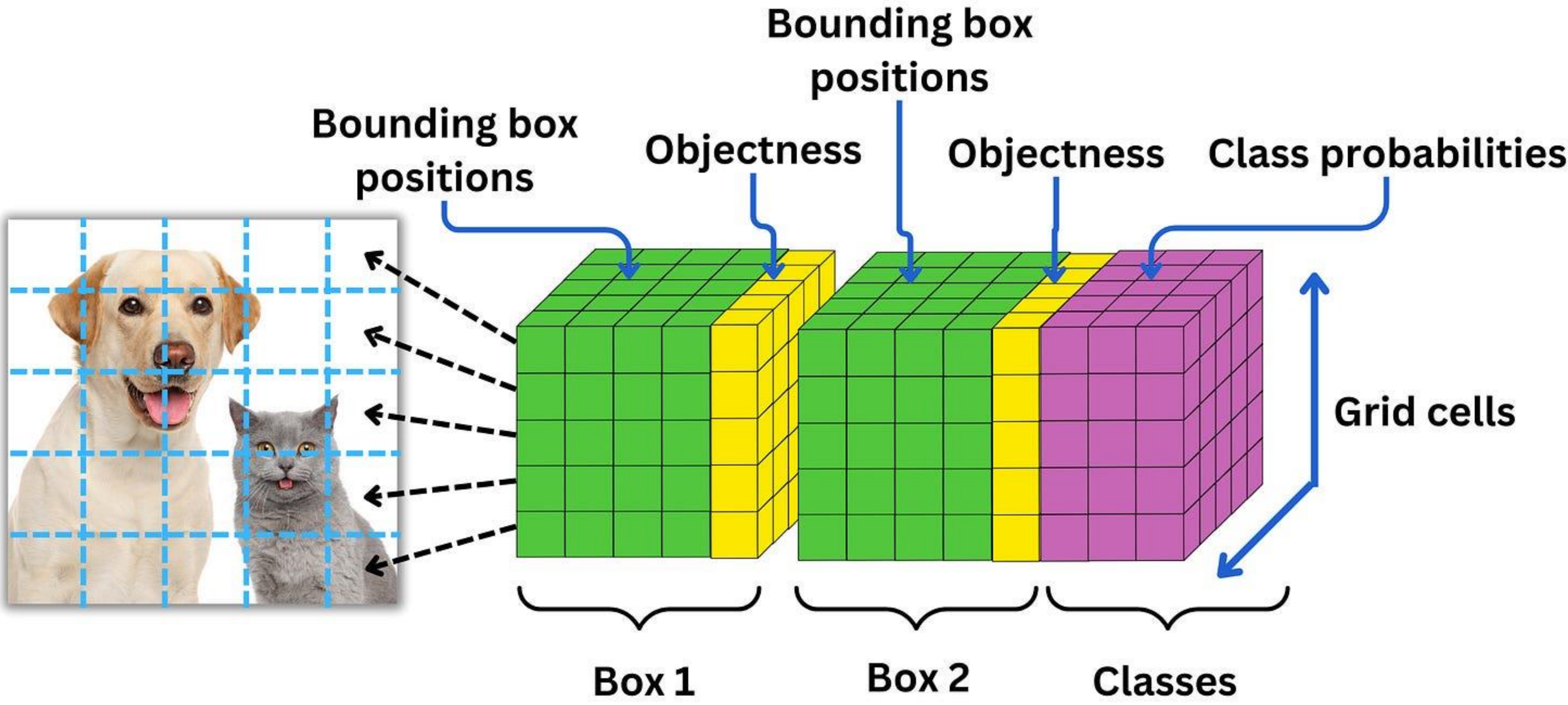
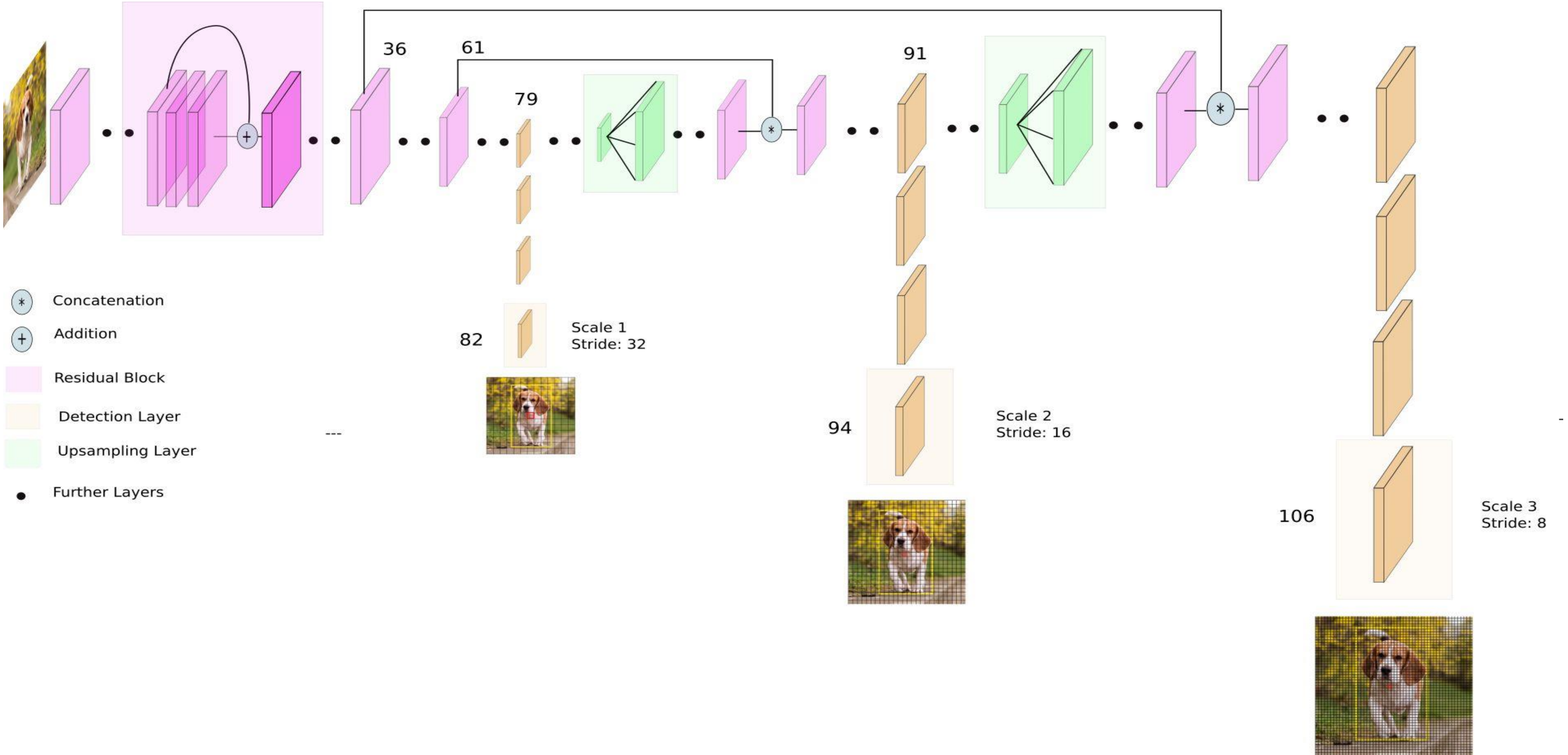


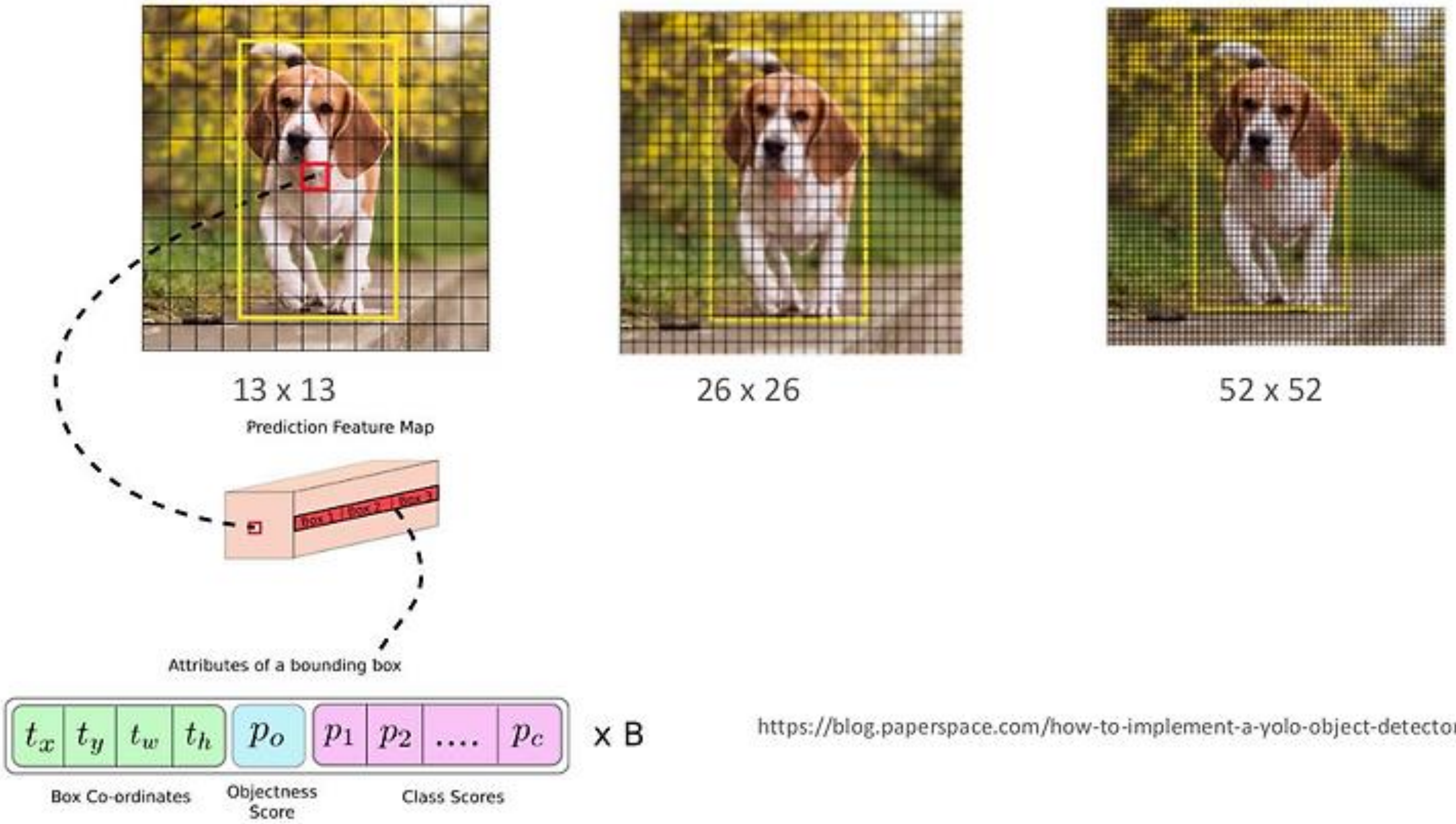
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.


Bounding box positions
Objectness
Bounding box positions
Objectness
Class probabilities
Grid cells
Box 1
Box 2
Classes

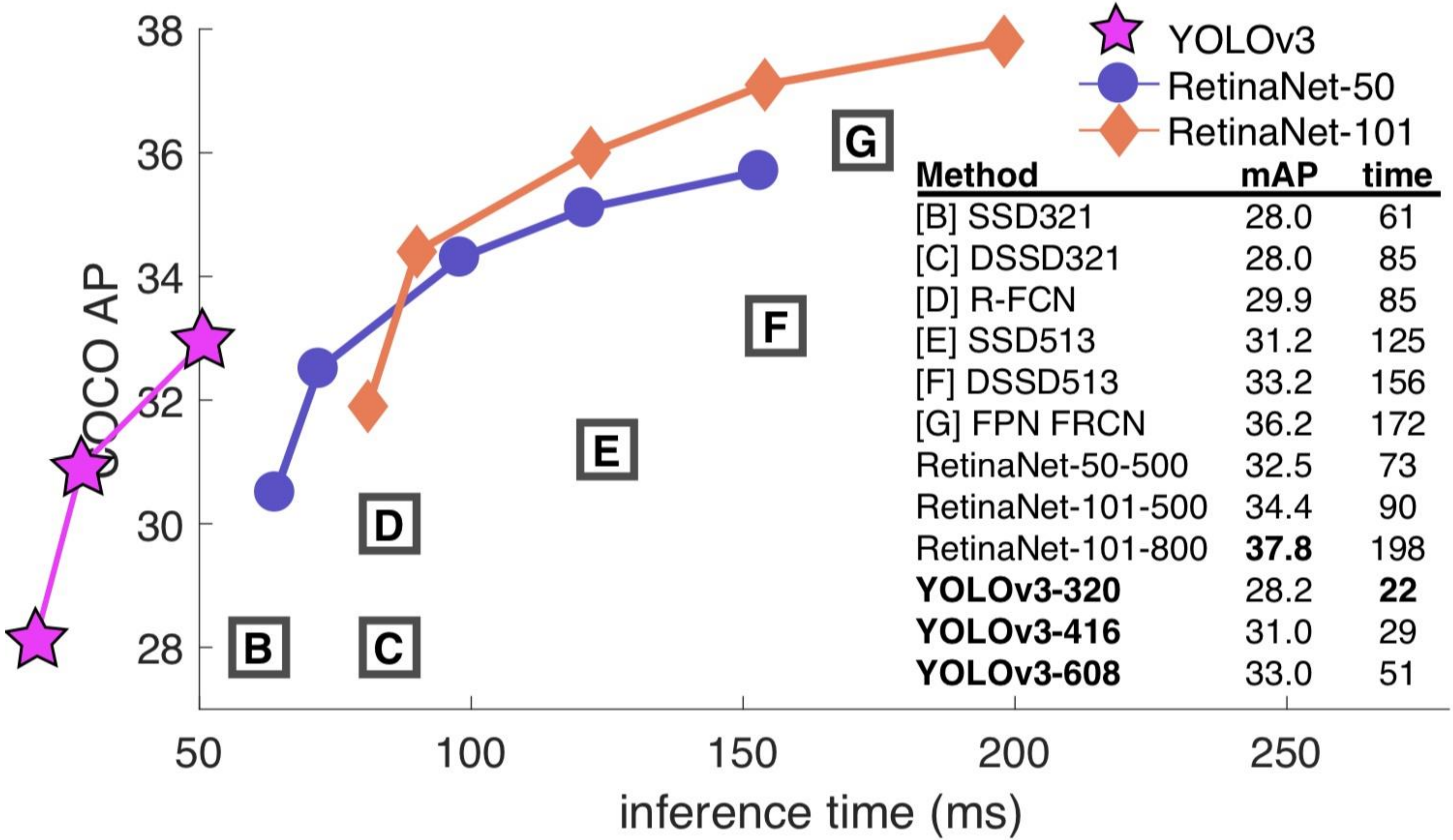
Yolo-3 architecture (2018)



Yolo-3 output layers



https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

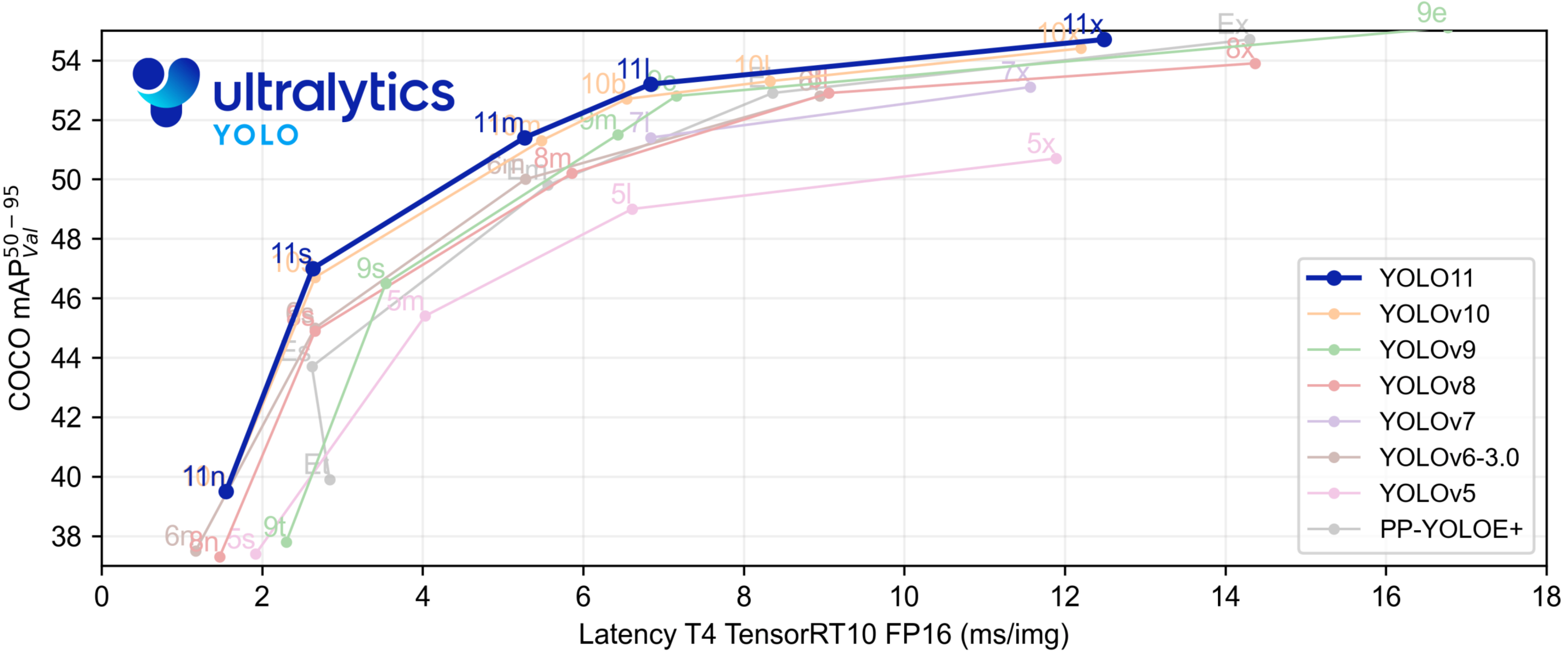


YOLOv3
RetinaNet-50
RetinaNet-101
Method mAP time
[B] SSD321 28.0 61
[C] DSSD321 28.0 85
[D] R-FCN 29.9 85
[E] SSD513 31.2 125
[F] DSSD513 33.2 156
[G] FPN FRCN 36.2 172
RetinaNet-50-500 32.5 73
RetinaNet-101-500 34.4 90
RetinaNet-101-800 37.8 198
YOLOv3-320 28.2 22
YOLOv3-416 31.0 29
YOLOv3-608 33.0 51
COCO AP
38
36
34
32
30
28
B
C
D
E
F
G
50
100
150
200
250
inference time (ms)

	저자	발표 시점	단체	출간처	특징	정확도 (mAP)	속도 (fps)	논문/github	코드 베이스	라이선스
Yolov1 ~ YOLOv3	Joseph Redmon, Ali Farhadi	2018년 4월	University of Washington	arXiv	Feature Pyramid Network(FPN), Multi-Scale Prediction	33.0% (COCO)	35 fps (Titan X)	paper github	Darknet	"YOLO" license
YOLOv4	Alexey Bochkovskiy, et al.	2020년 4월	(2,3저자) National Taiwan University	arXiv (YOLOv4)/ CVPR 2021 (Scaled-YOLOv4)	Bag of Freebies (BoF), Bag of Specials (BoS)	43.5% (COCO)	65 fps (V100)	paper paper(scaled) github (darknet) github(pytorch)	Darknet	"YOLO" license
YOLOv5	Glenn Jocher	2020년 6월	Ultralytics	-	PyTorch Implementation, CSP backbone	50% (COCO)	93 fps (RTX4090)	github	PyTorch	AGPL 3.0
YOLOv6	Chuyi Li, et al.	2022년 6월	Meituan	arXiv	Bi-directional Concatenation(BiC) module, Anchor-aided training(AAT) strategy	52.8% (COCO)	116 fps (T4)	paper(v1.0) paper(v3.0) github	PyTorch	GPL 3.0
YOLOv7	Chien-Yao Wang*, et al. (*v4 논문의 2저자)	2022년 7월	Academia Sinica	CVPR 2023	Extended-ELAN, Model Reparameterization	55.9% (COCO)	56 fps (V100)	paper github	YOLOv5 (PyTorch)	GPL 3.0
YOLOv8	Glenn Jocher, et al.	2023년 1월	Ultralytics	-	Anchor-free, Multi-task application	53.9% (COCO)	78 fps (T4)	website	PyTorch	AGPL 3.0
YOLOv9	Chien-Yao Wang, et al.	2024년 2월	Academia Sinica	arXiv	Programmable Gradient Information (PGI), Generalized ELAN	55.6% (COCO)	73 fps (T4)	paper github	YOLOv5	(PyTorch)
YOLOv10	Ao Wang, et al.	2024년 5월	Tsinghua University	arXiv / NeurIPS 2024 poster	NMS-free training, spatial-channel separate downsampling	54.4% (COCO)	94 fps(T4)	paper github	Ultralytics(YOLOv8/PyTorch)	AGPL 3.0

출처: https://devocean.sk.com/blog/techBoardDetail.do?ID=166976&boardType=techBlog

Yolo version comparison



https://docs.ultralytics.com/ko/models/yolov8/#overview