What is object detection?

이름 : 김지원

전공 : 항공우주공학 / 컴퓨터공학

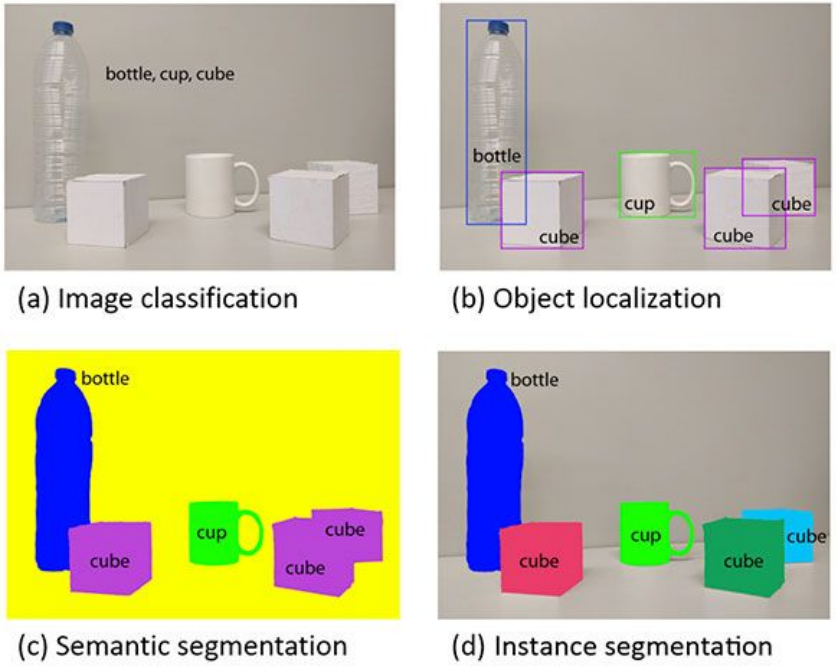
연구분야 : **Object Detection**

연구실 : [지능기전공학부 RCV연구실]

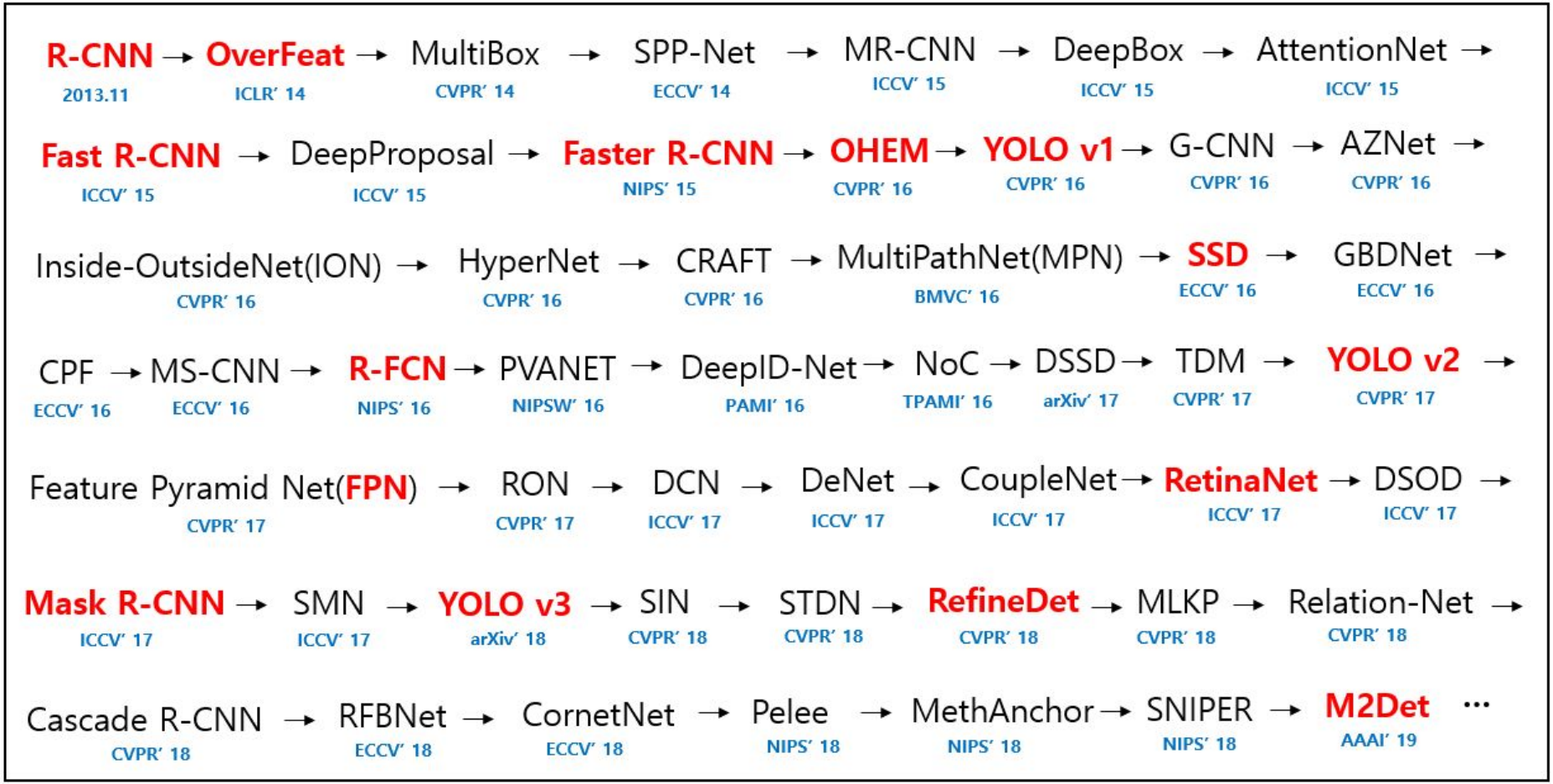
메일 : **jwkim@rcv.sejong.ac.kr**

Github : https://github.com/socome

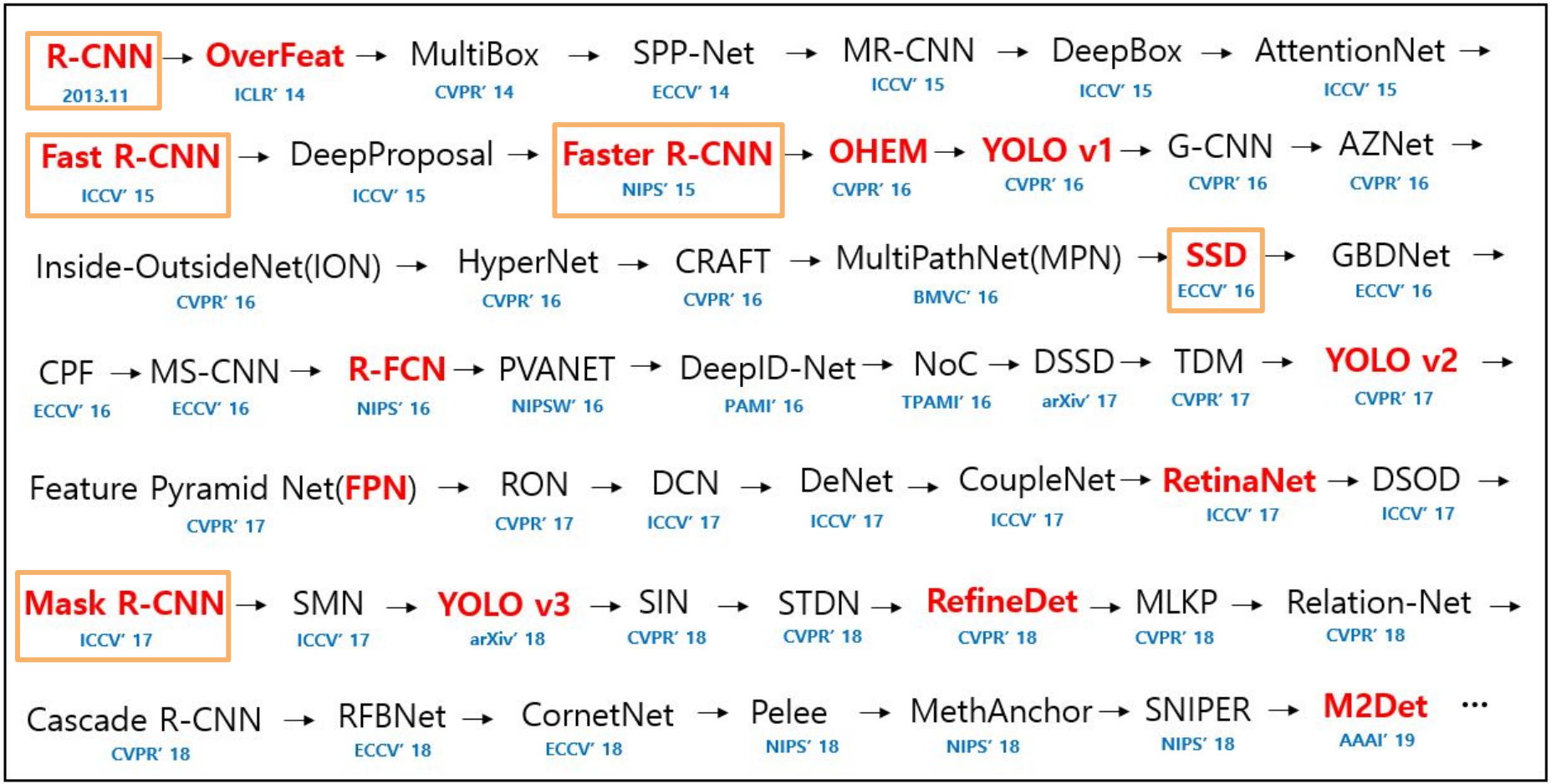
Blog : https://socome.github.io/about/



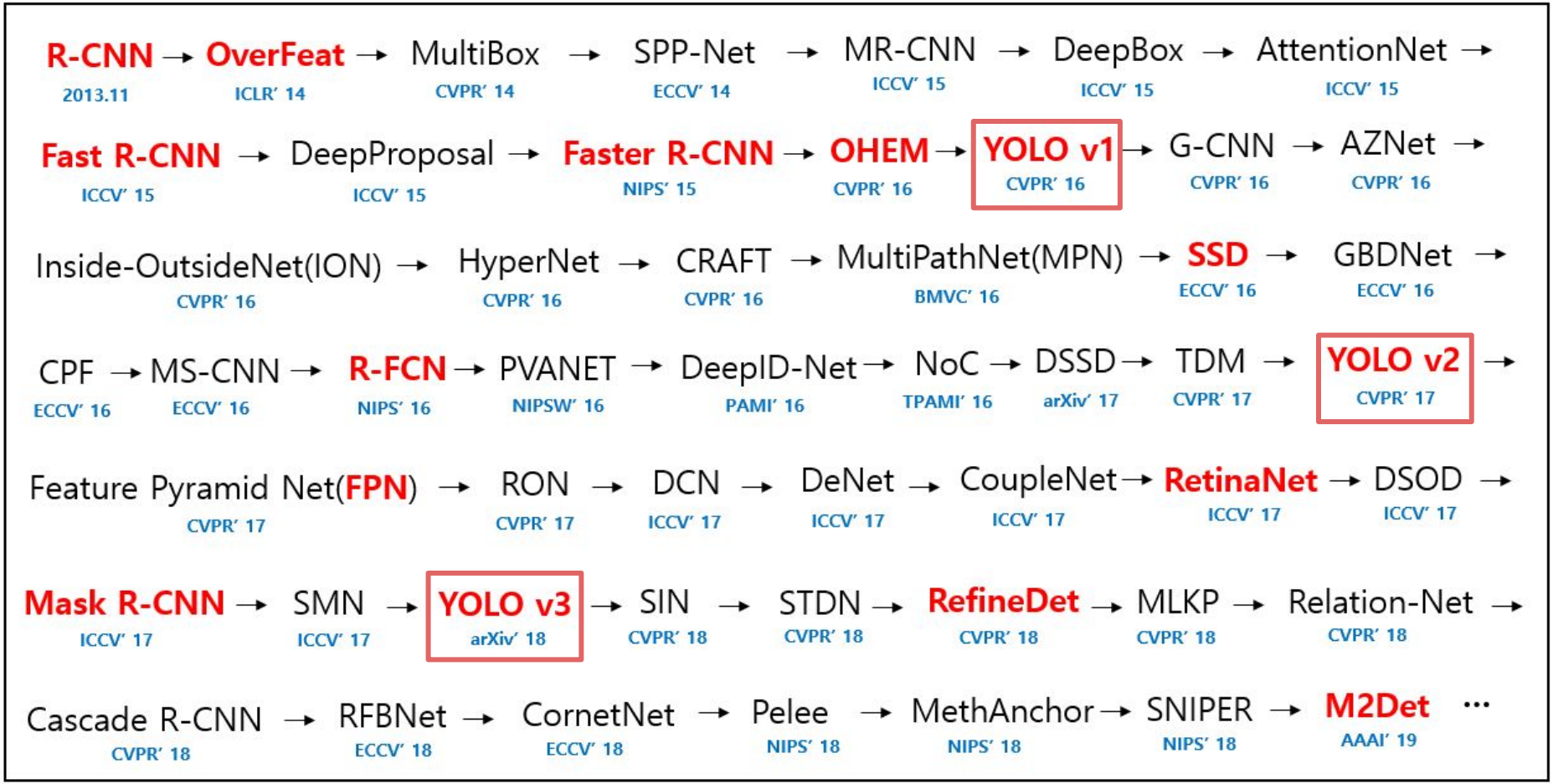
https://www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/



https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-what-is-object-detection/



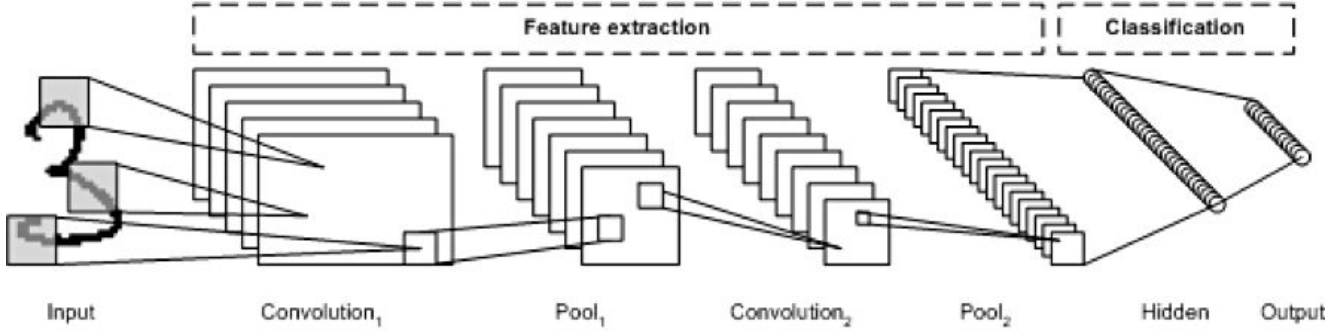
https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-what-is-object-detection/

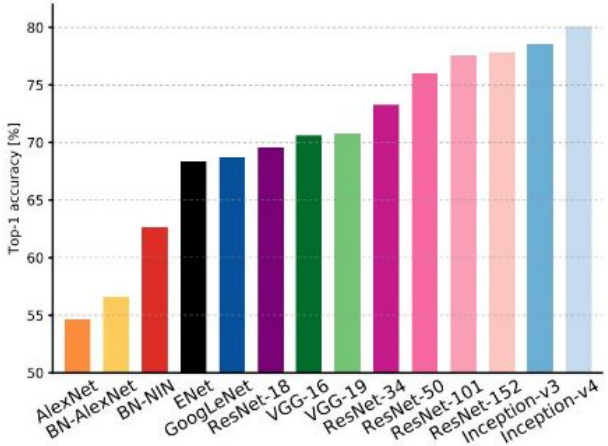


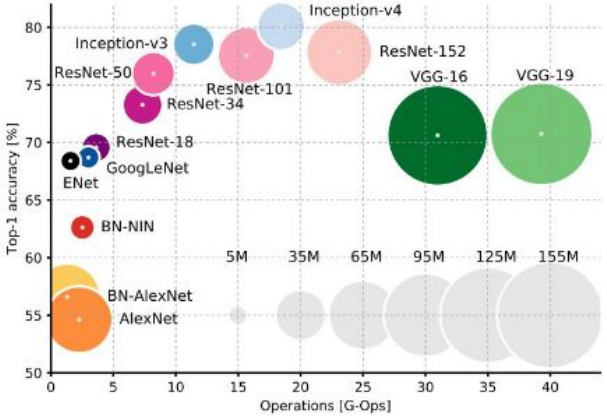
https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-what-is-object-detection/

About CNN

(a) Image classification



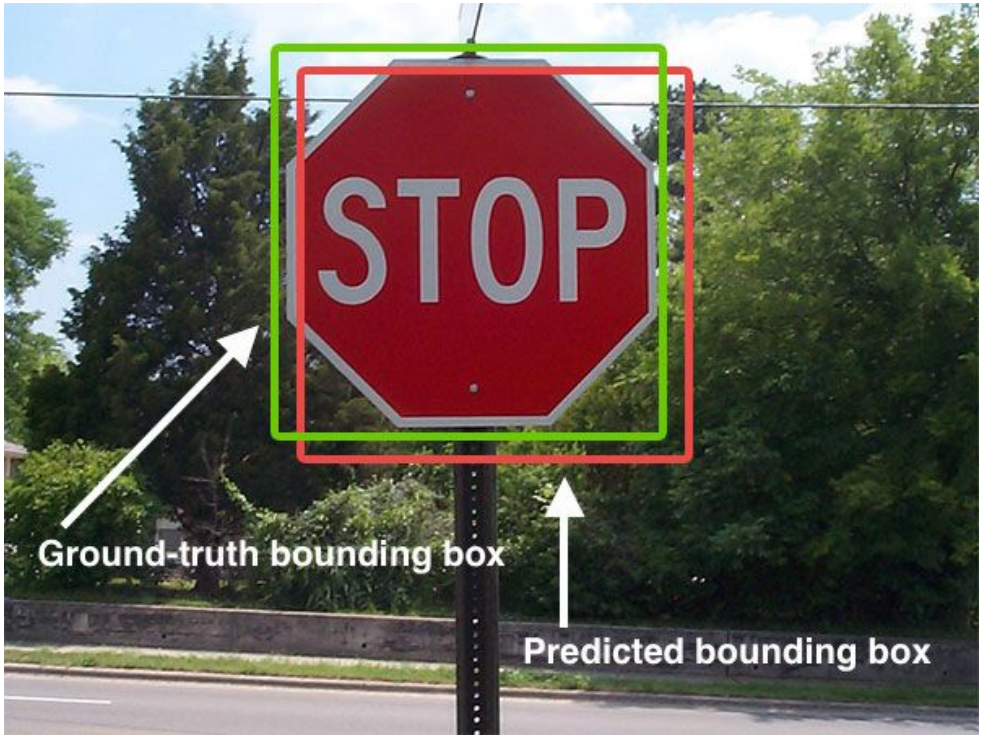




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Object Detection

IOU



https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

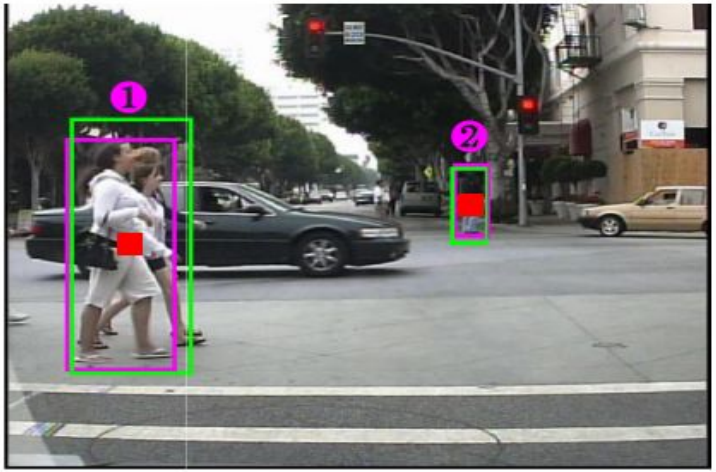
Box Regression

R-CNN: Pipeline

R-CNN in detail

- Bounding box regression*
  - Proposal $P = (P_x, P_y, P_w, P_h)$ / Ground-truth $G = (G_x, G_y, G_w, G_h)$
  - **For box-scale invariance**, regression target t for the training pair (P, G)

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_y$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h)$$



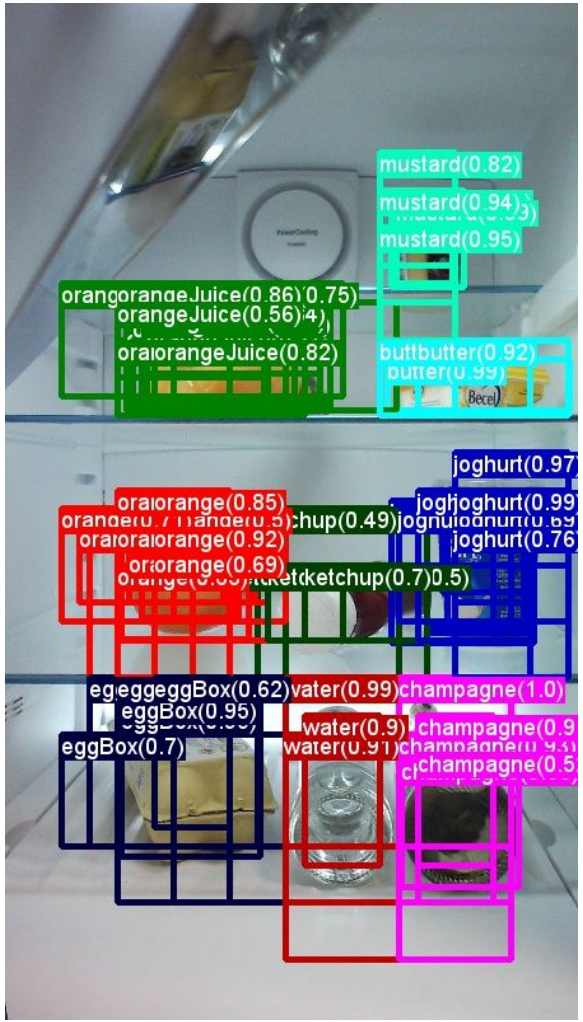
① $G = (57.4, 141.0, 97.2, 237.0)$
$P = (61.6, 120.0, 107.3, 261.7)$

$t_x = (57.4 - 61.6)/107.3 = -0.039$
$t_y = (141.0 - 120.0)/261.7 = 0.080$
$t_w = \log(97.2/107.3) = -0.043$
$t_h = \log(237.0/261.7) = -0.043$

② $G = (408.0, 167.0, 29.9, 73.0)$
$P = (405.8, 170.8, 31.5, 76.9)$

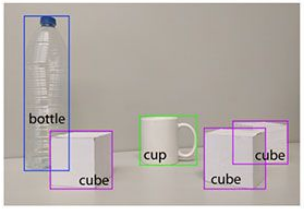
$t_x = (408.0 - 405.8)/31.5 = 0.070$
$t_y = (167.0 - 170.8)/76.9 = -0.049$
$t_w = \log(29.9/31.5) = -0.023$
$t_h = \log(73.0/76.9) = -0.023$

NMS

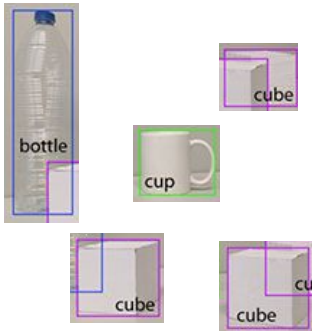

mustard(0.82)
mustard(0.94)
mustard(0.95)
orangeJuice(0.56)
orangeJuice(0.82)
butter(0.92)
joghurt(0.97)
joghurt(0.99)
joghurt(0.76)
orange(0.92)
orange(0.69)
ketchup(0.49)
ketchup(0.7)
eggBox(0.95)
eggBox(0.7)
water(0.99)
champagne(1.0)
champagne(0.5)

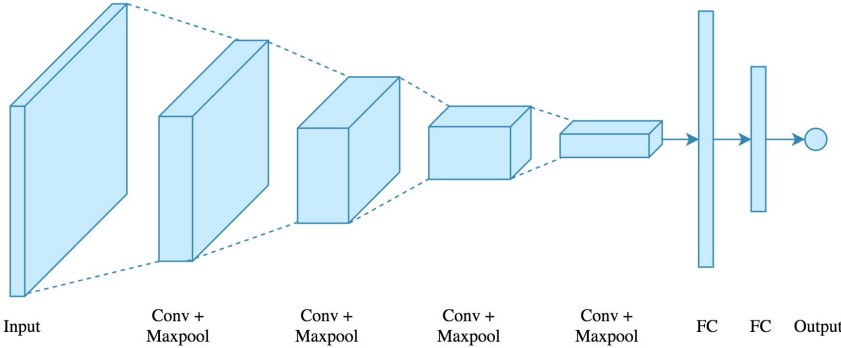

mustard(0.99)
orangeJuice(0.91)
butter(0.99)
joghurt(0.99)
orange(0.97)
ketchup(0.77)
eggBox(0.95)
water(0.99)
champagne(1.0)

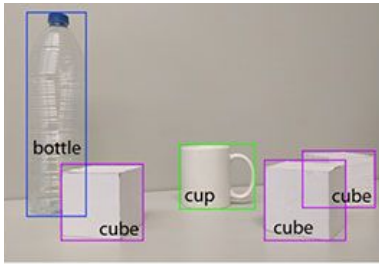
RCNN , Fast RCNN, Faster RCNN, Mask RCNN

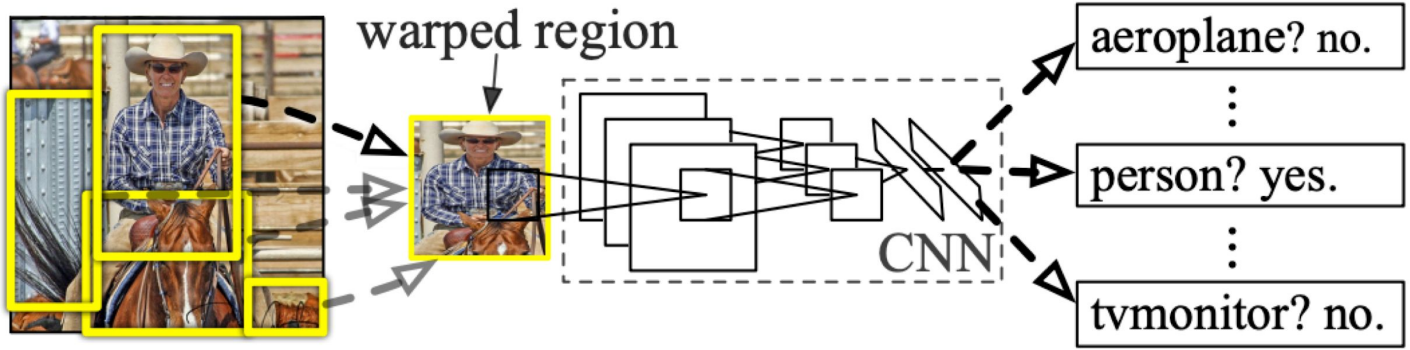


(b) Object localization

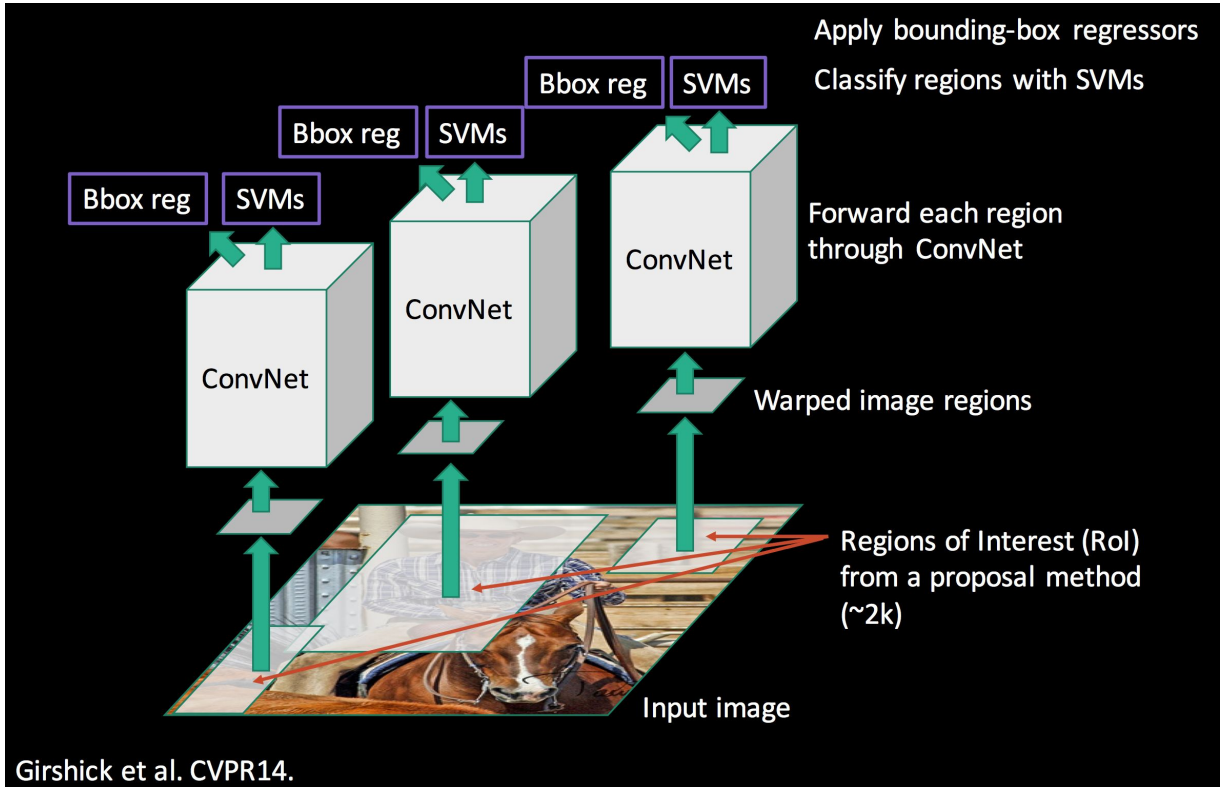




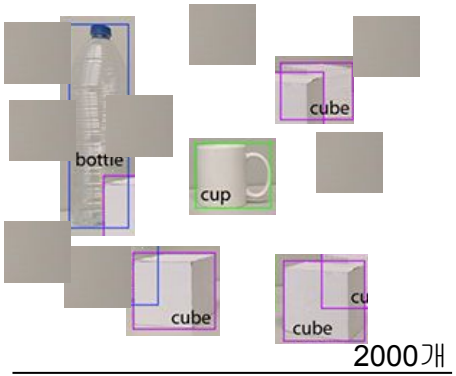


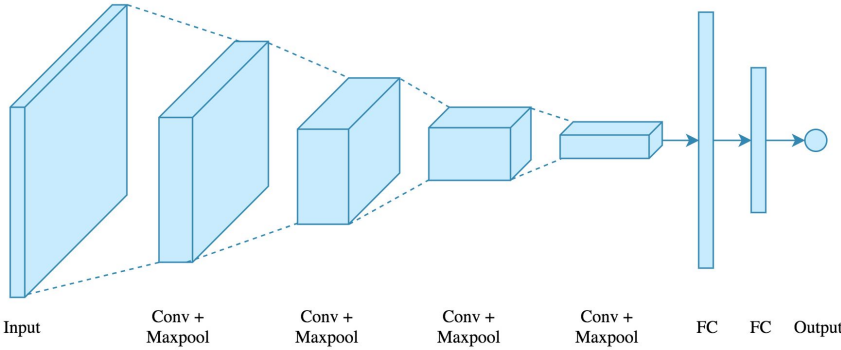


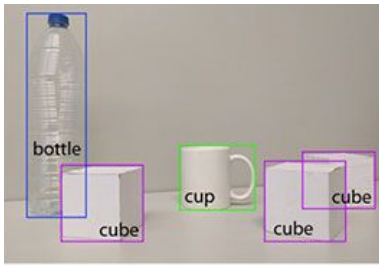
RCNN


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.

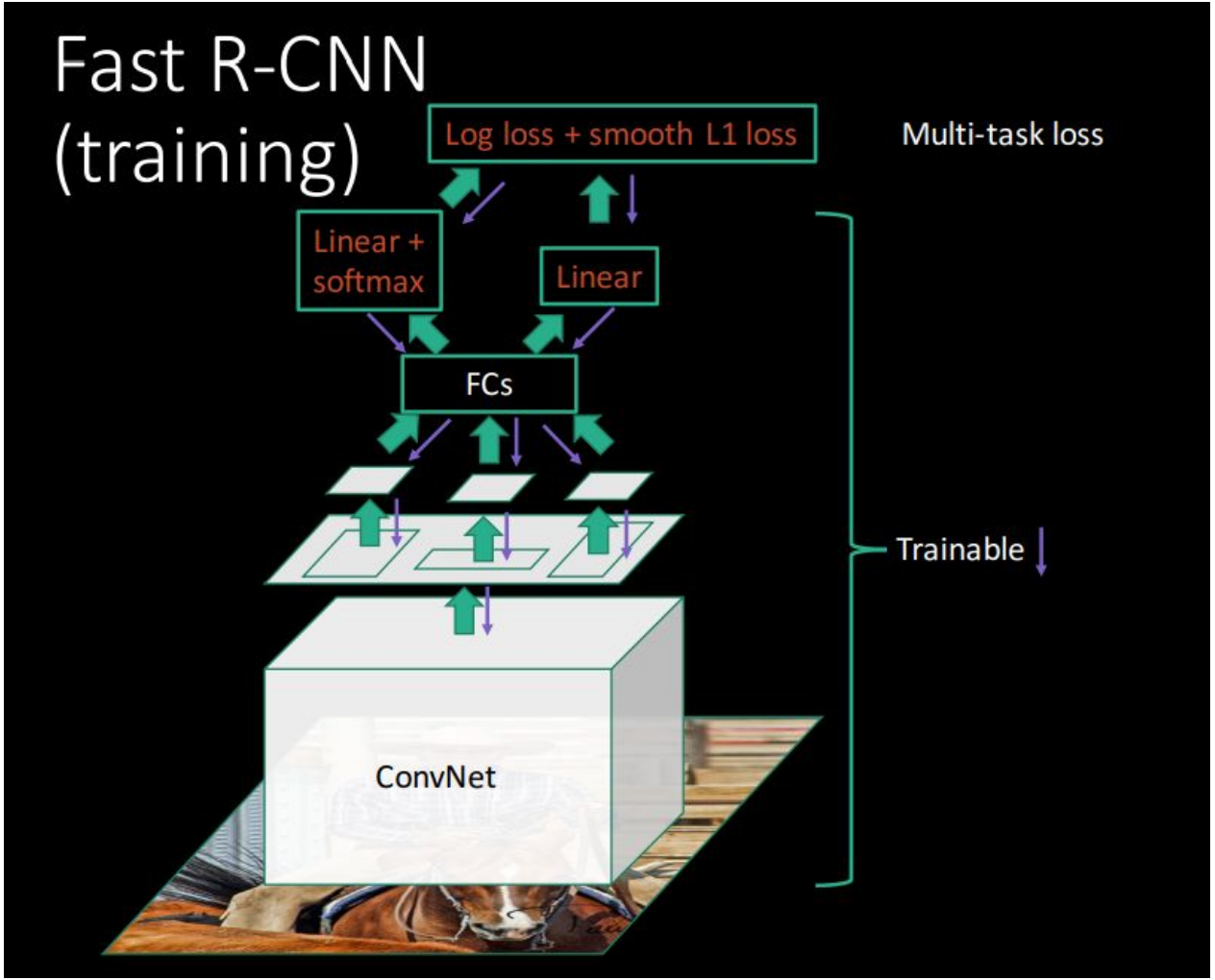
RCNN , Fast RCNN, Faster RCNN, Mask RCNN







2000개의 모든 **proposal**을 처리하기에는 너무 많은 시간 소모



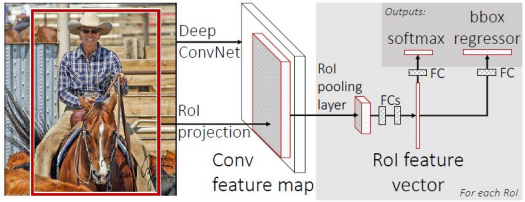
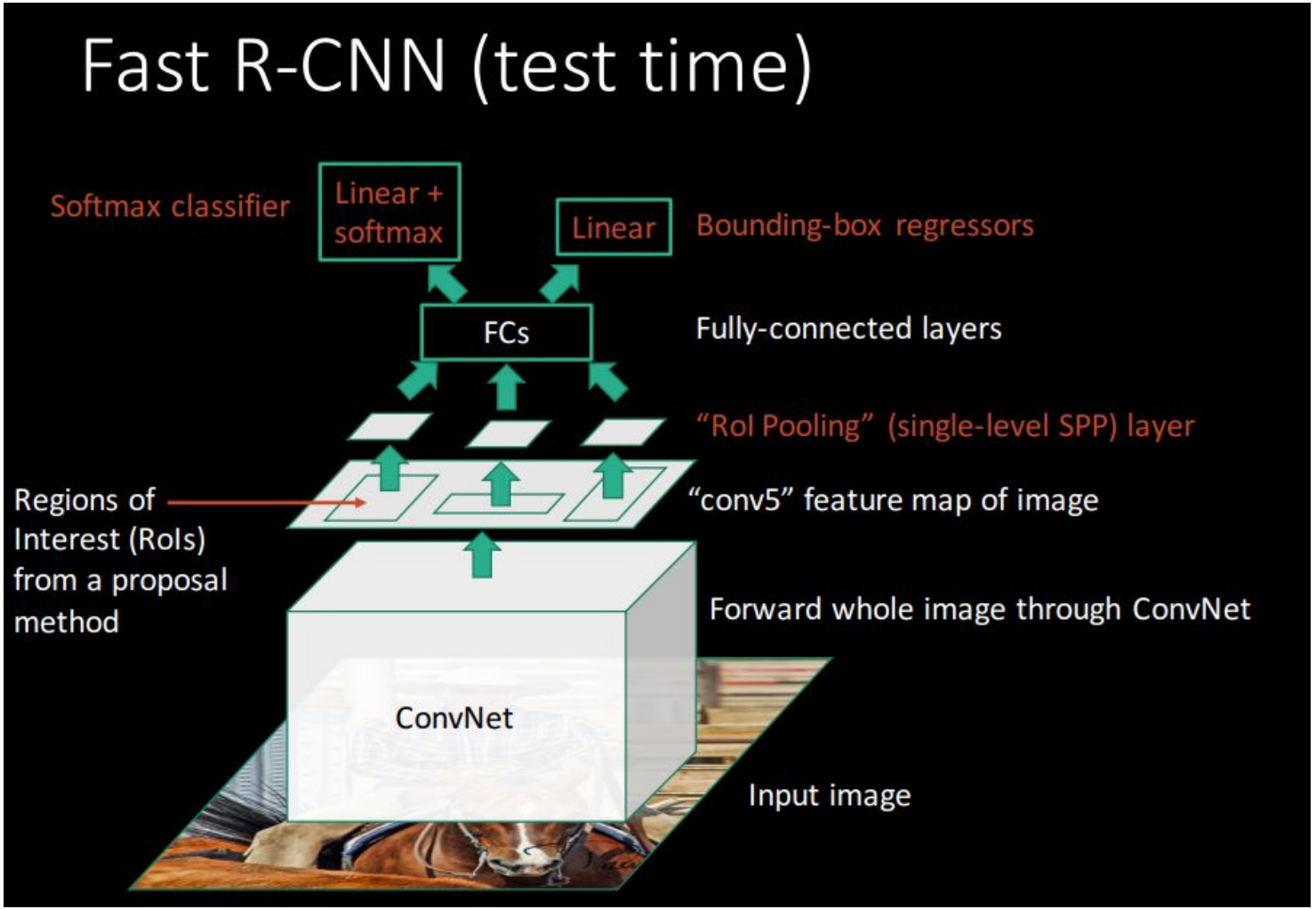
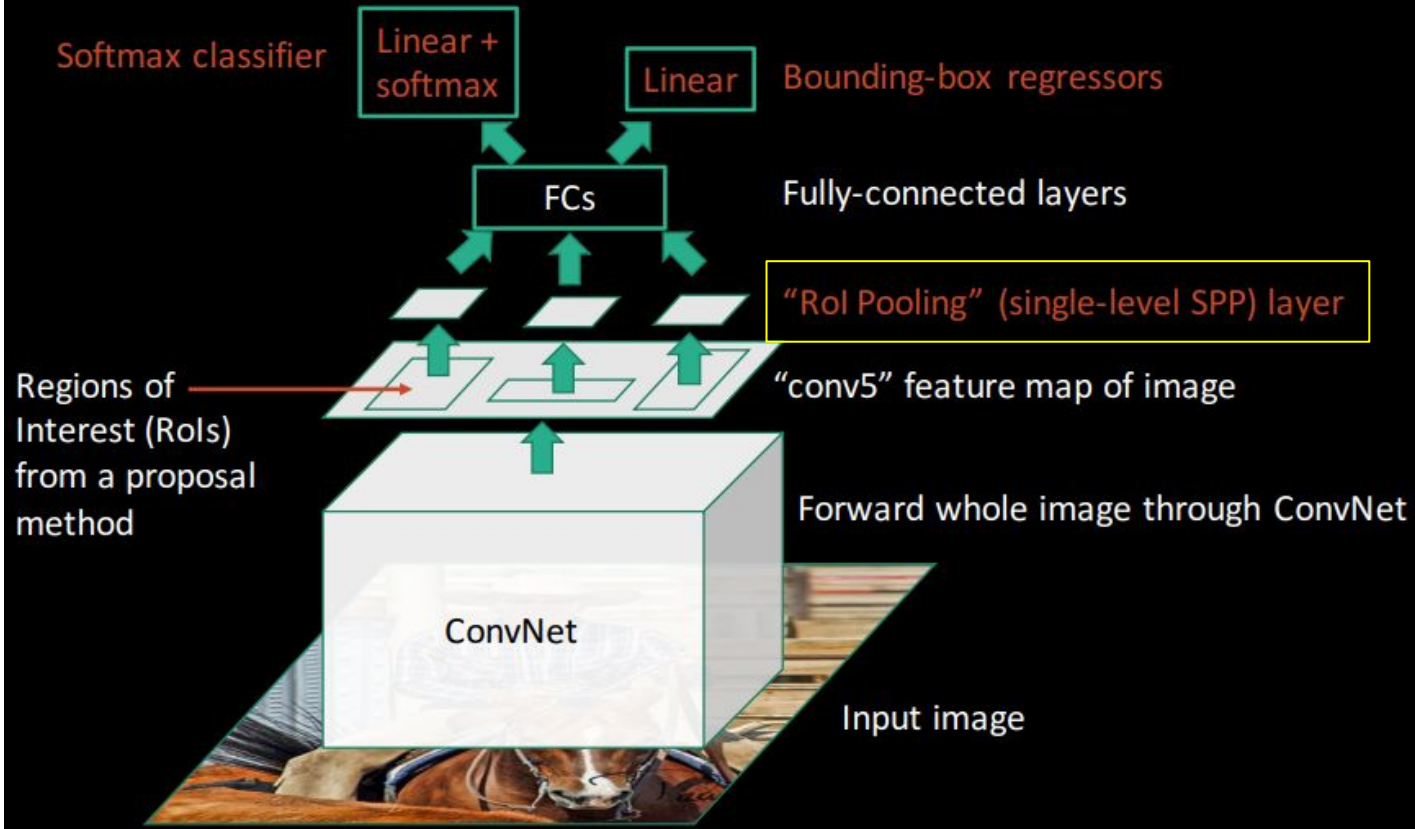


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

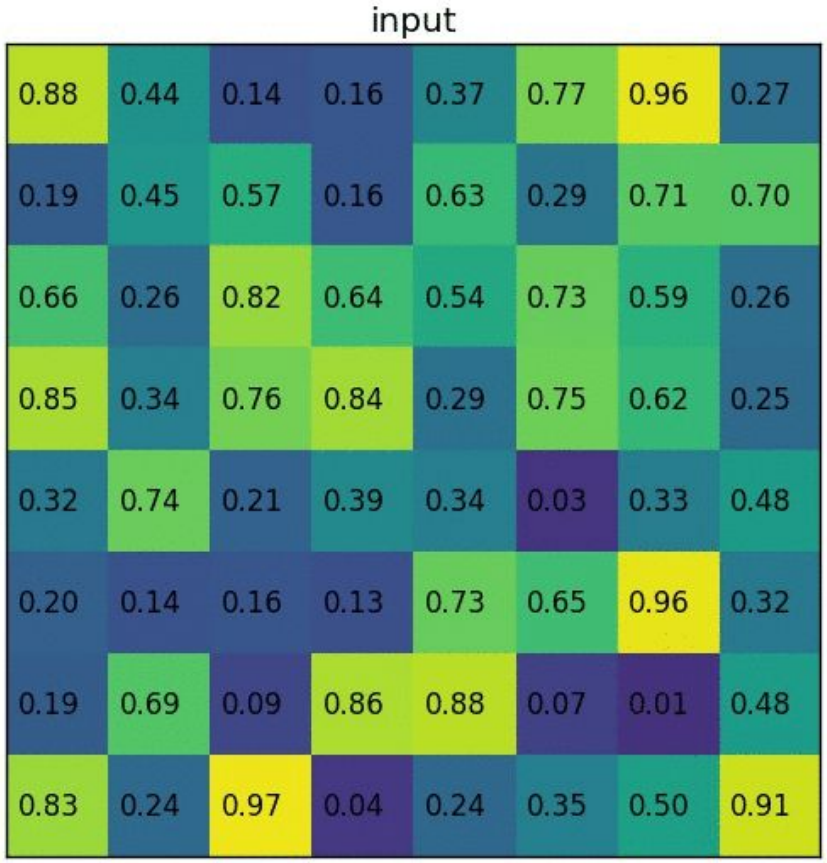
Fast R-CNN (test time)

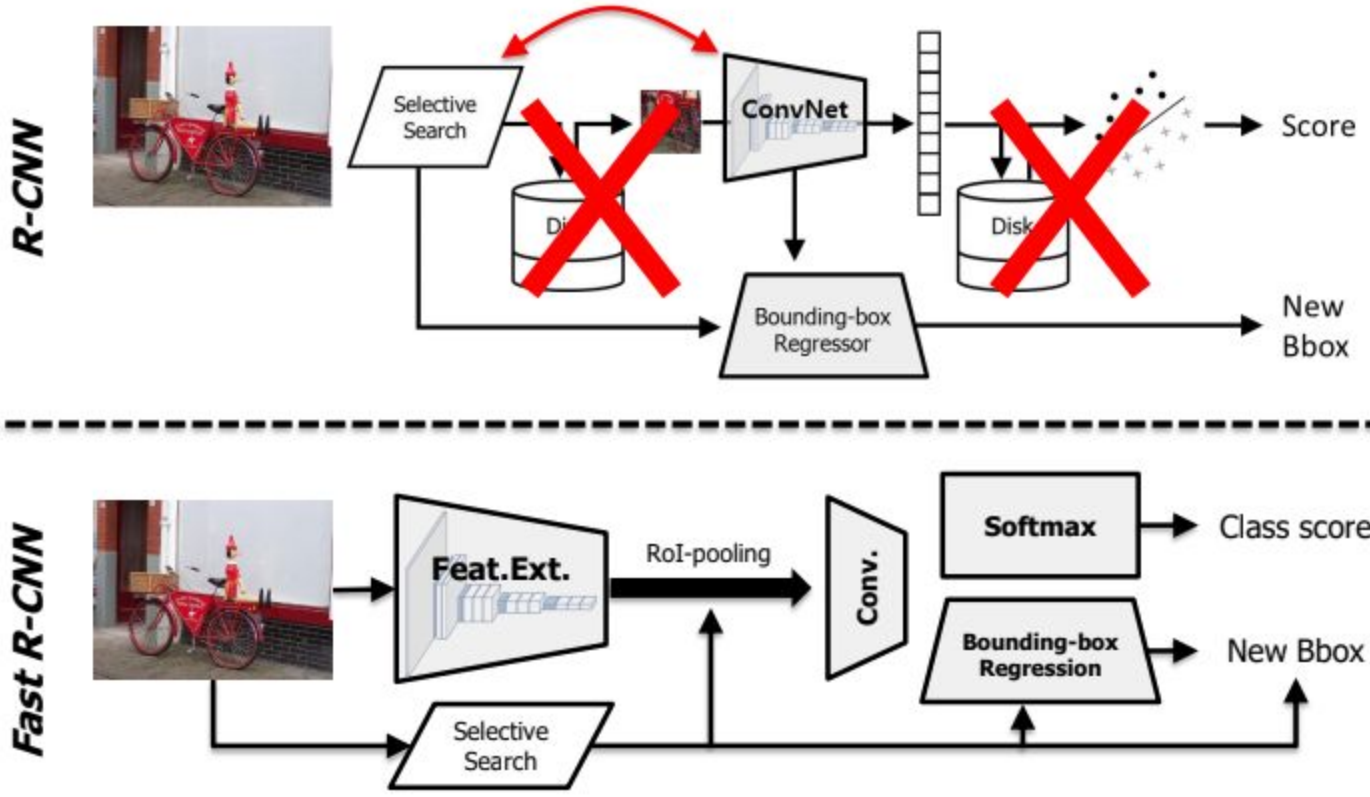


Fast R-CNN (test time)



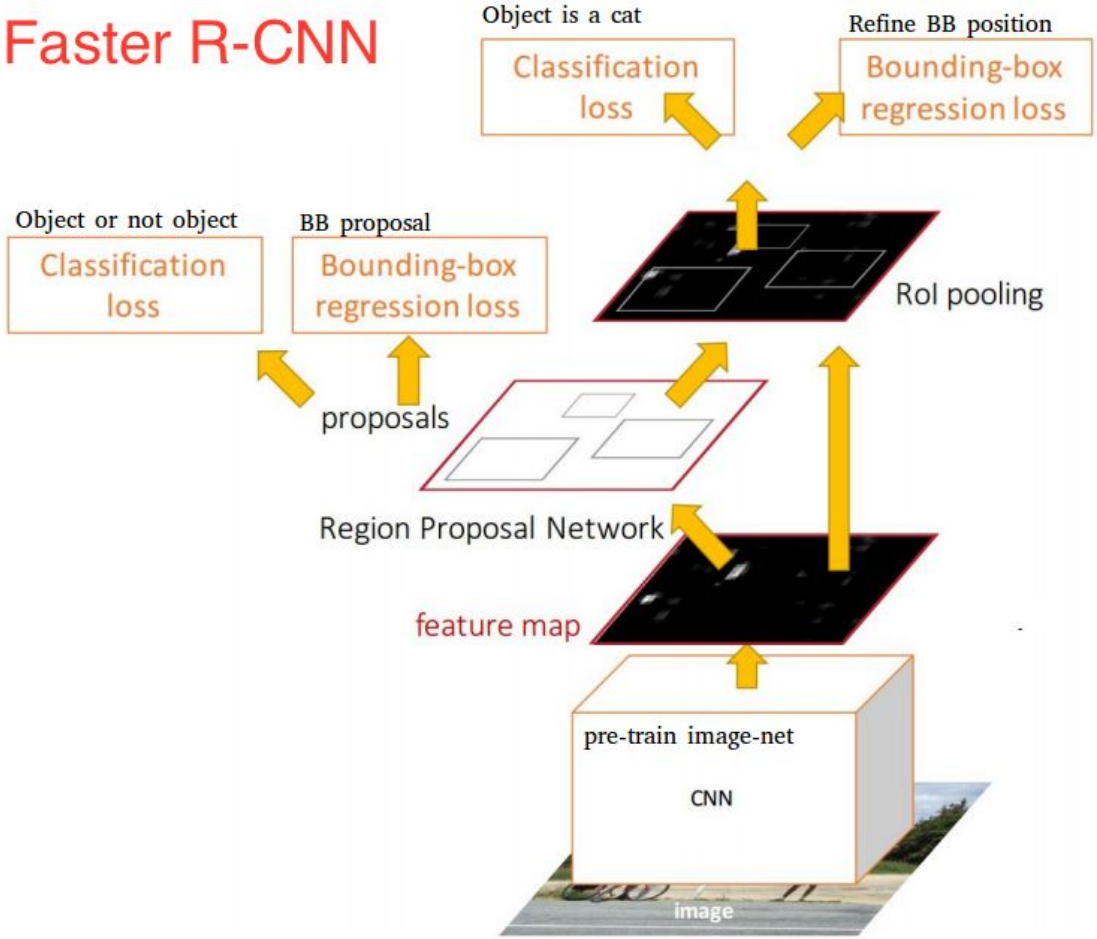
ROI Pooling


input
0.88 0.44 0.14 0.16 0.37 0.77 0.96 0.27
0.19 0.45 0.57 0.16 0.63 0.29 0.71 0.70
0.66 0.26 0.82 0.64 0.54 0.73 0.59 0.26
0.85 0.34 0.76 0.84 0.29 0.75 0.62 0.25
0.32 0.74 0.21 0.39 0.34 0.03 0.33 0.48
0.20 0.14 0.16 0.13 0.73 0.65 0.96 0.32
0.19 0.69 0.09 0.86 0.88 0.07 0.01 0.48
0.83 0.24 0.97 0.04 0.24 0.35 0.50 0.91

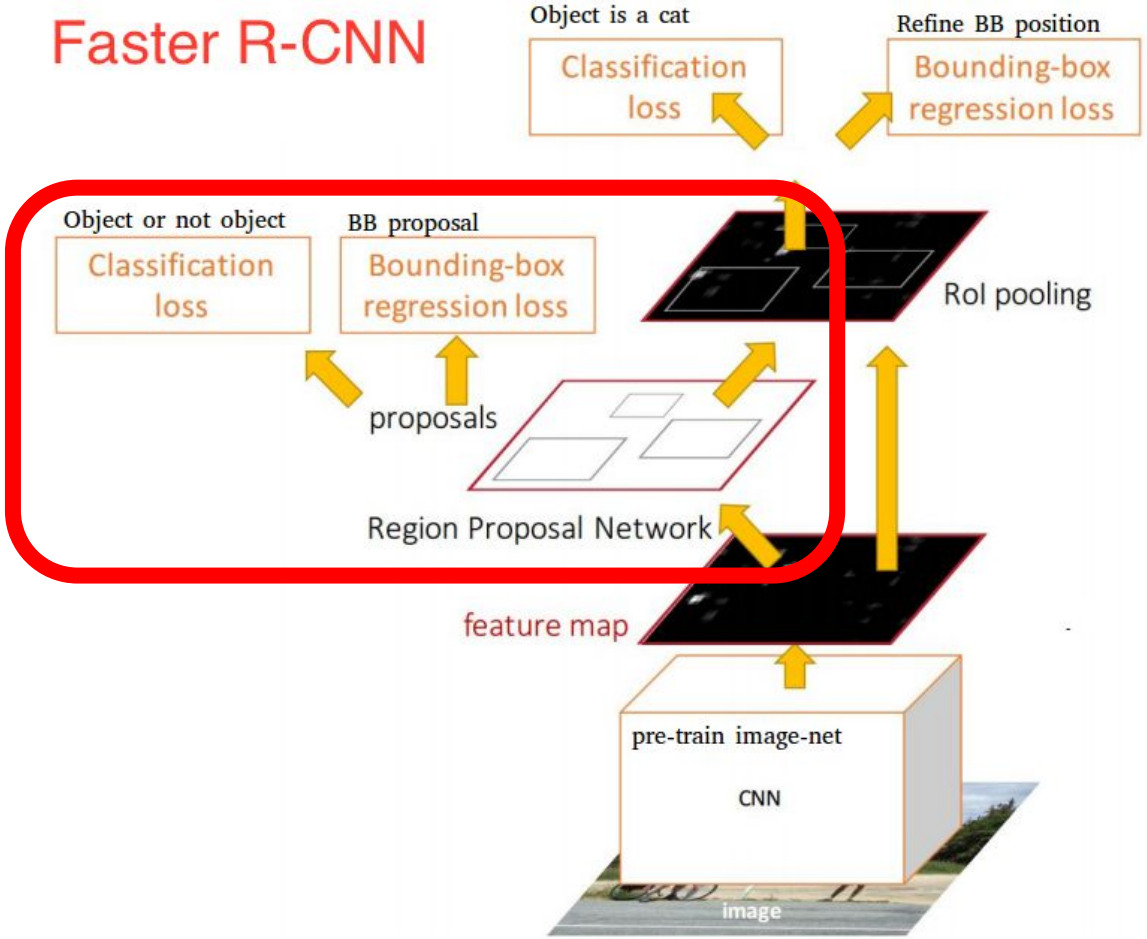

R-CNN
Selective Search
ConvNet
Score
Bounding-box Regressor
New Bbox
Fast R-CNN
Feat.Ext.
RoI-pooling
Conv.
Softmax
Class score
Bounding-box Regression
New Bbox
Selective Search

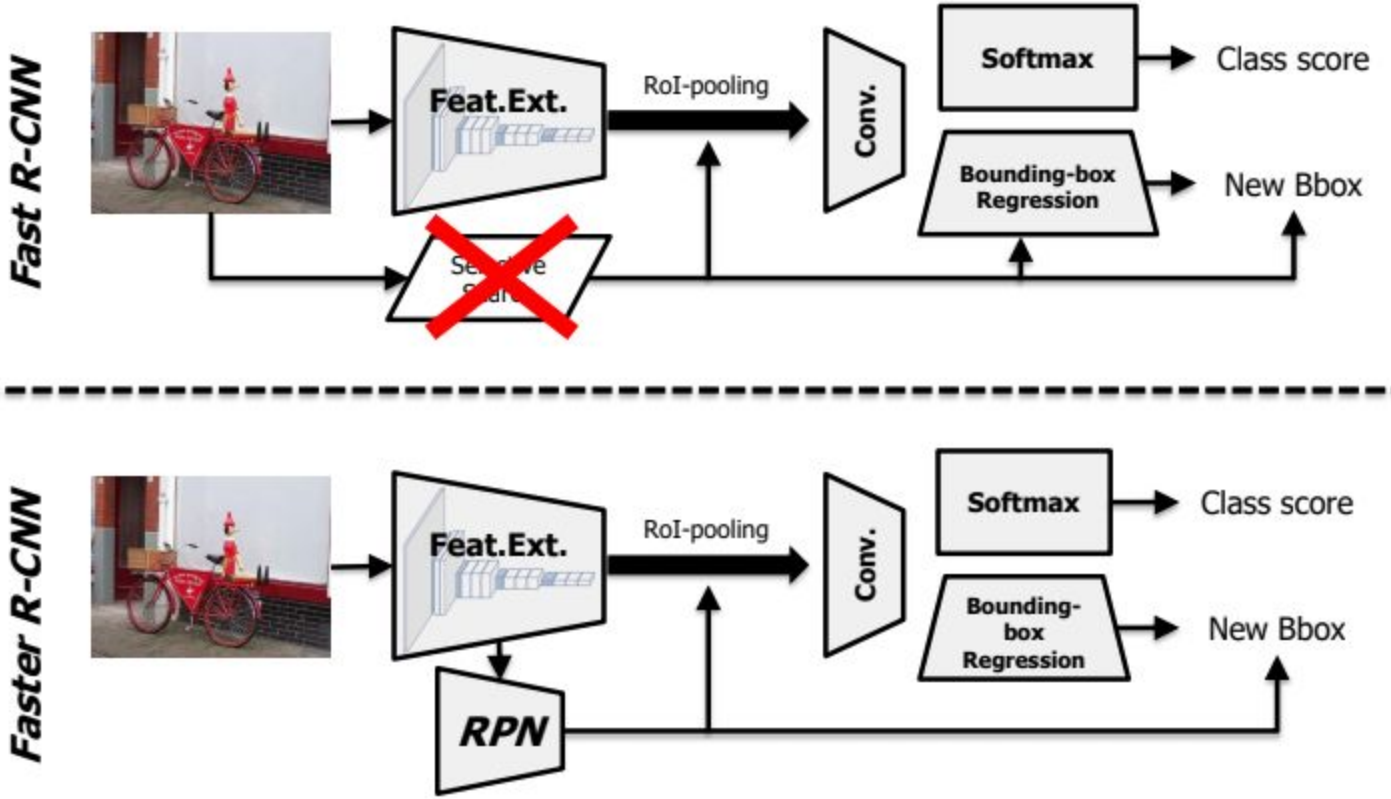
RCNN , Fast RCNN, Faster RCNN, Mask RCNN

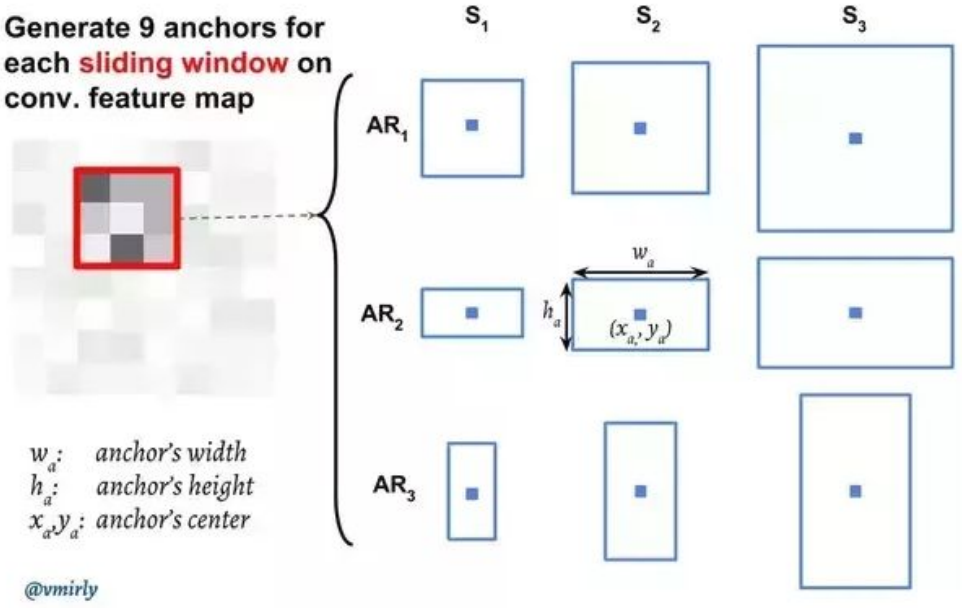
Faster R-CNN

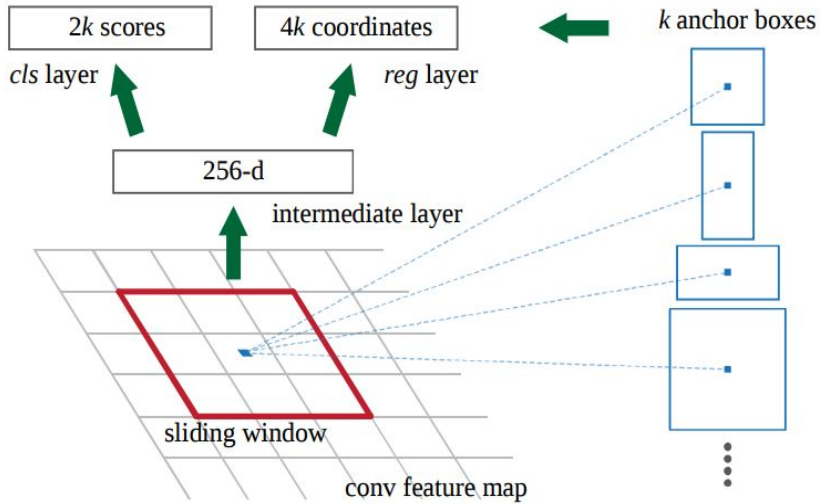

Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image

Faster R-CNN

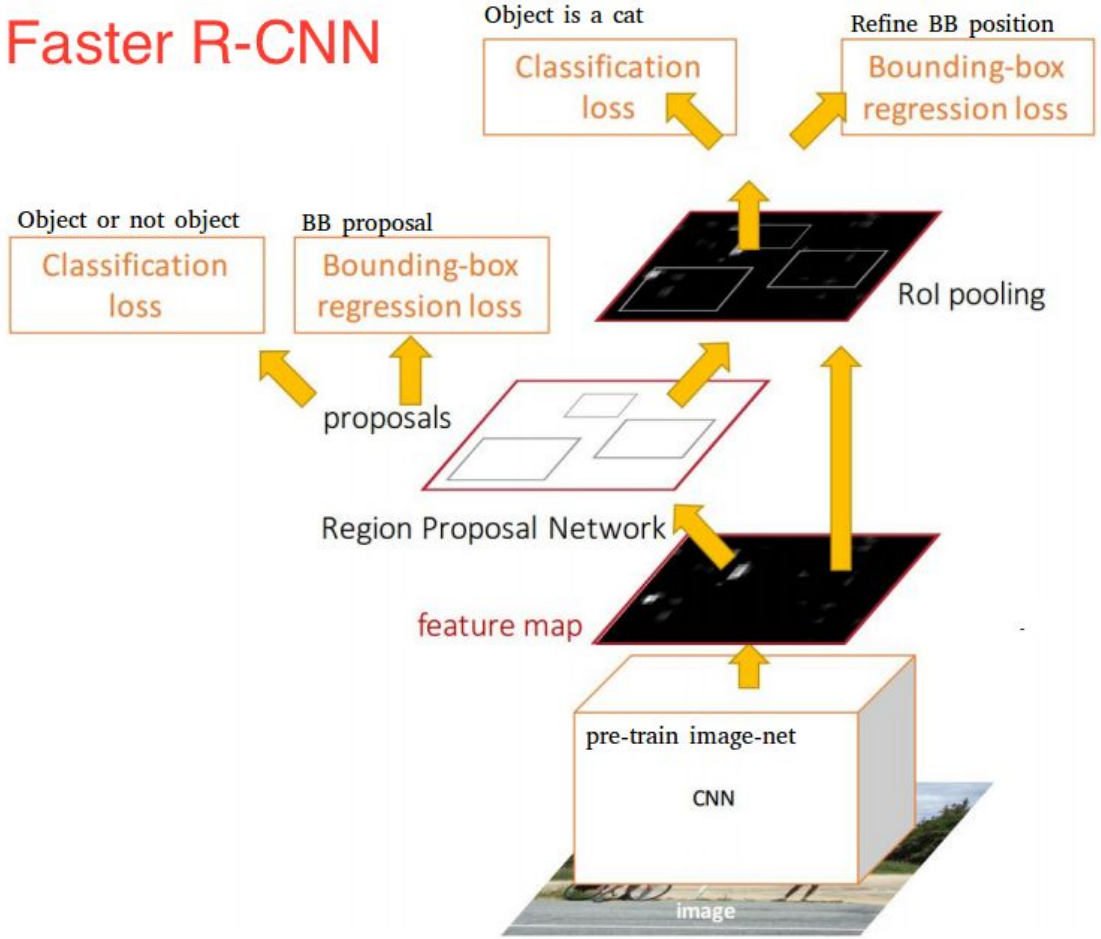

Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image


Fast R-CNN
Feat.Ext.
RoI-pooling
Conv.
Softmax
Class score
Bounding-box Regression
New Bbox
Faster R-CNN
Feat.Ext.
RoI-pooling
Conv.
Softmax
Class score
Bounding-box Regression
New Bbox
RPN

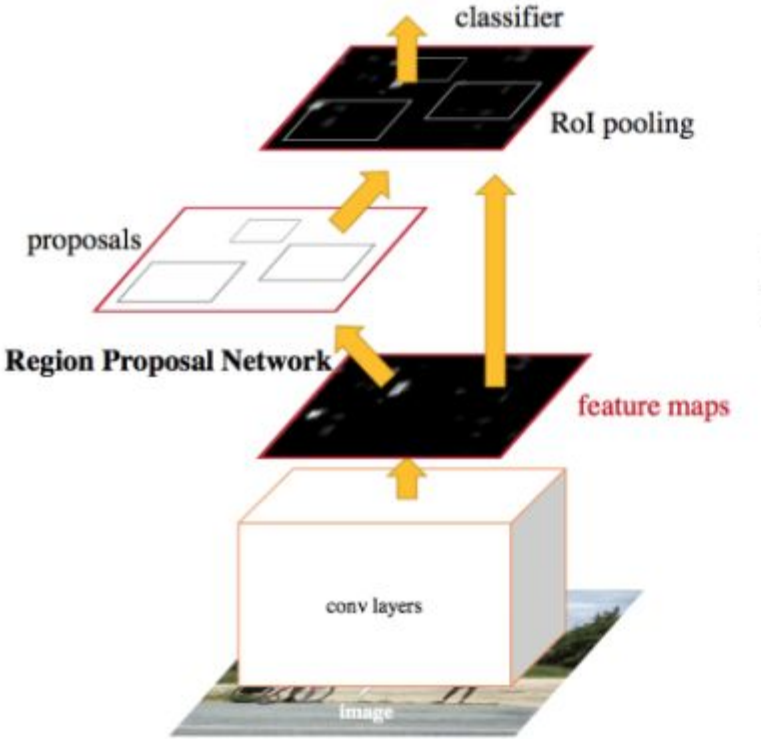

Generate 9 anchors for each sliding window on conv. feature map
S1
S2
S3
AR1
AR2
AR3
wa
ha
(xa, ya)
wa: anchor's width
ha: anchor's height
xa,ya: anchor's center
@vmirly


2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

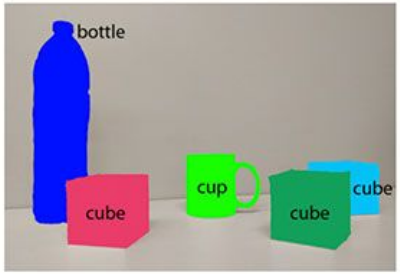
Train


Faster R-CNN
Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image

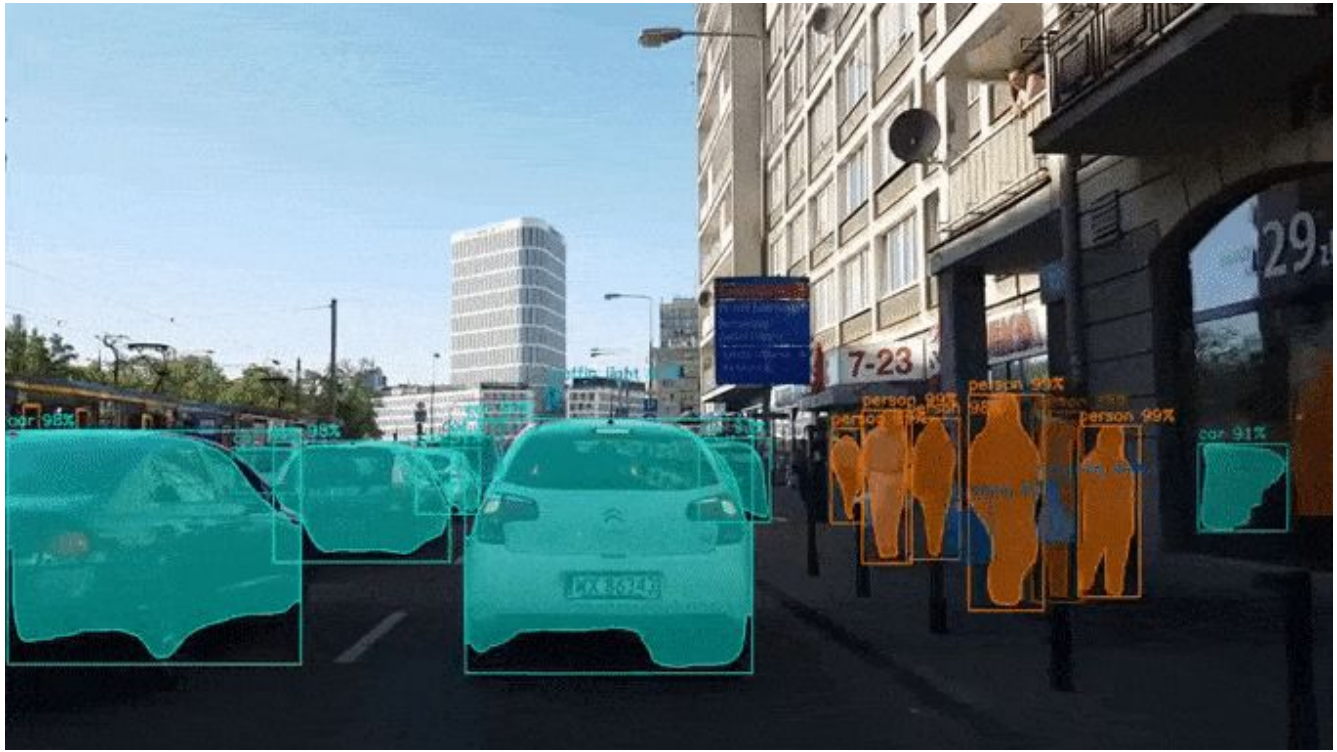
Test


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

RCNN , Fast RCNN, Faster RCNN, Mask RCNN



(d) Instance segmentation



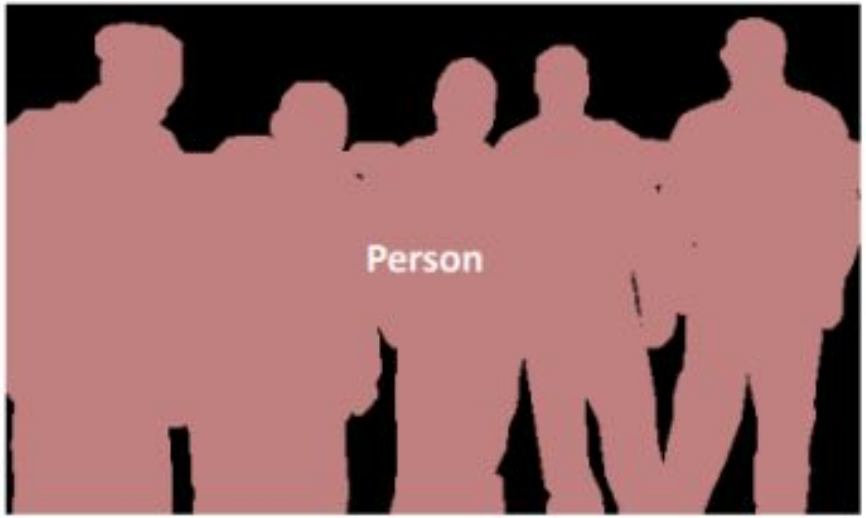


Image 1

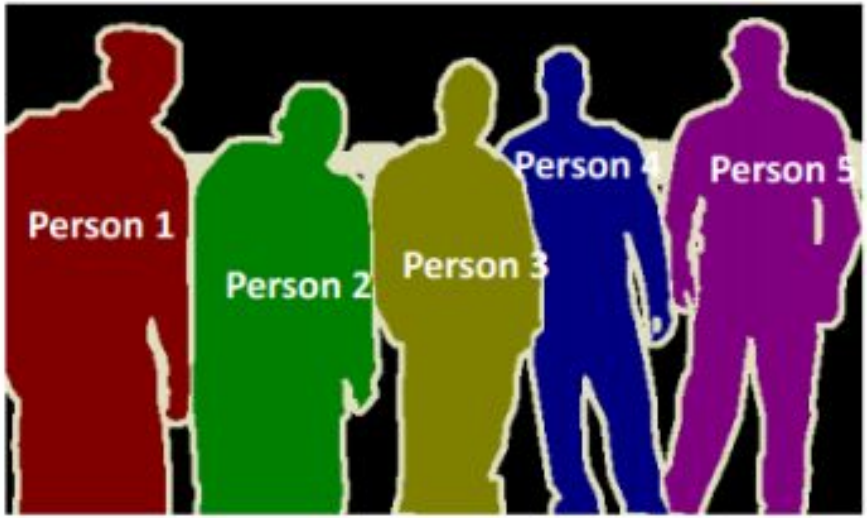
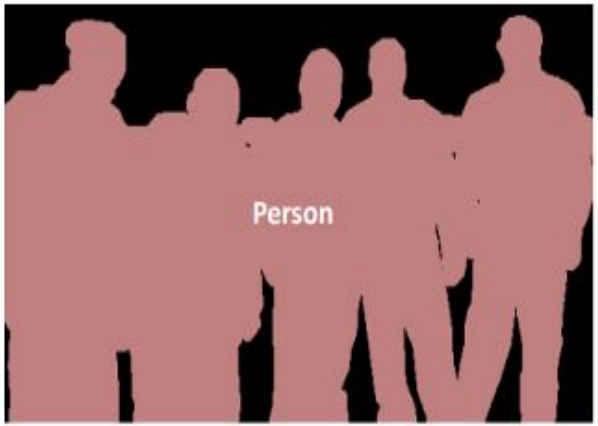
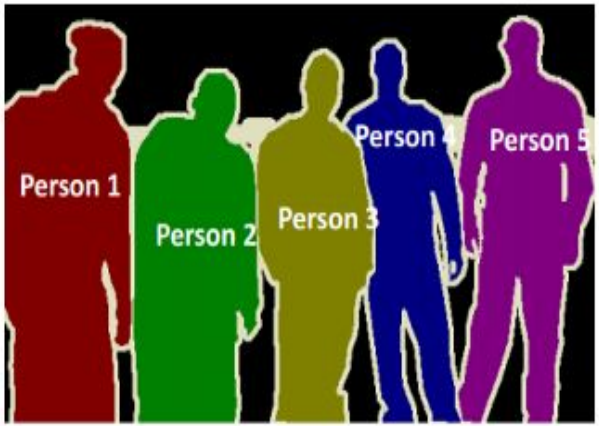
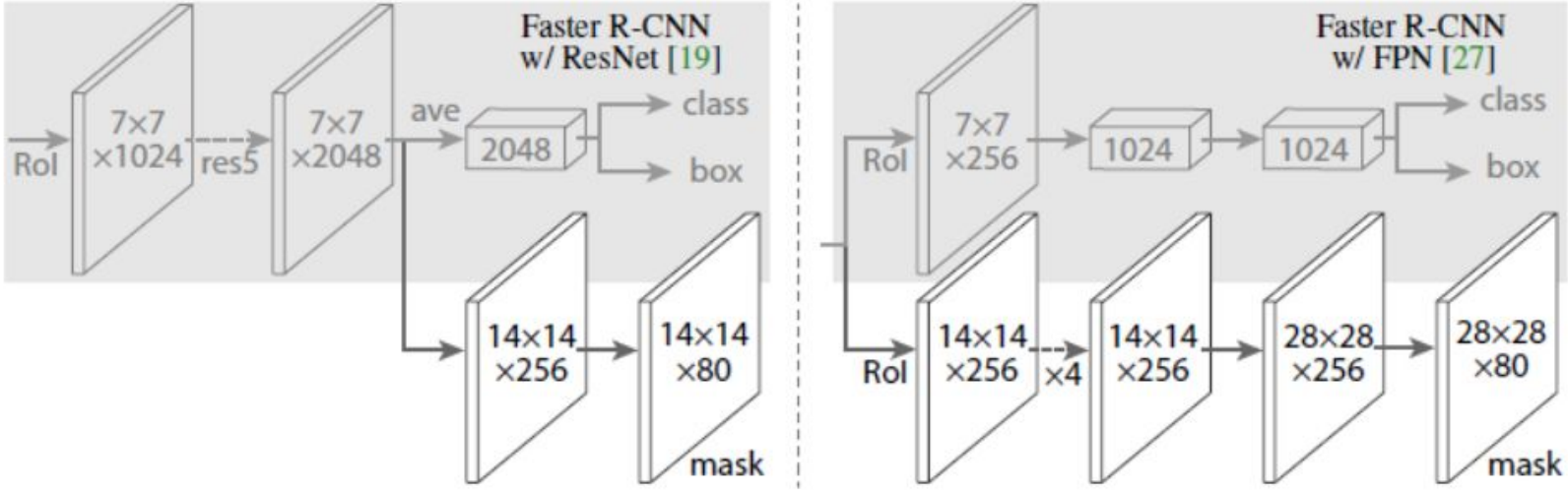
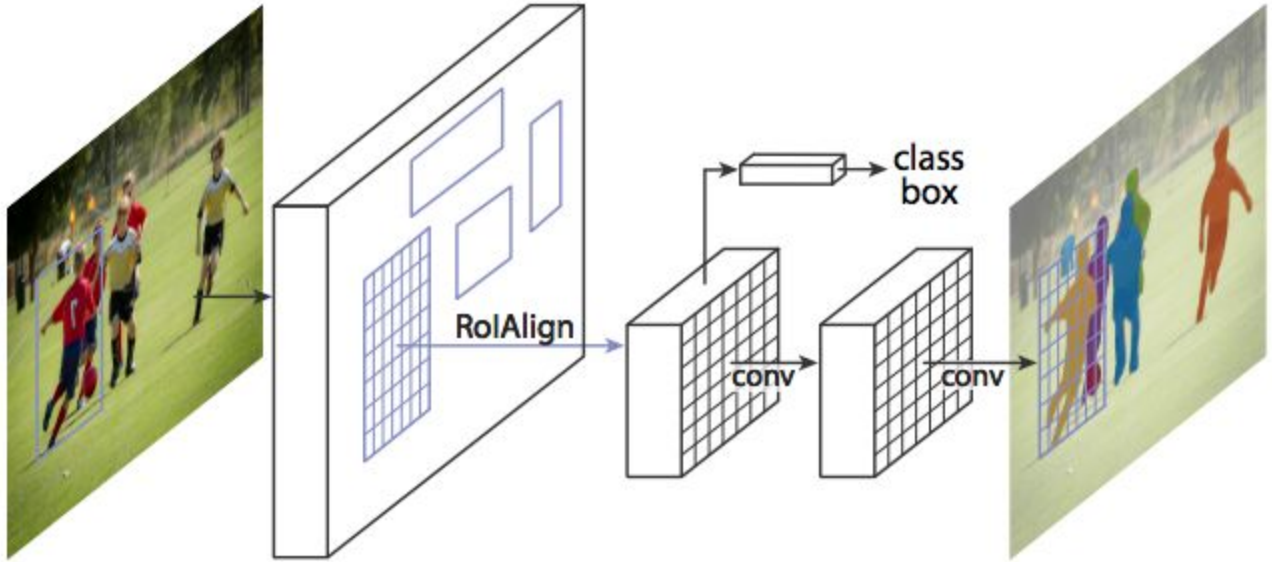


Image 2


Person


Person 1
Person 2
Person 3
Person 4
Person 5


Faster R-CNN
w/ ResNet [19]
RoI
7×7
×1024
res5
7×7
×2048
ave
2048
class
box
14×14
×256
14×14
×80
mask
Faster R-CNN
w/ FPN [27]
RoI
7×7
×256
1024
1024
class
box
RoI
14×14
×256
×4
14×14
×256
28×28
×256
28×28
×80
mask

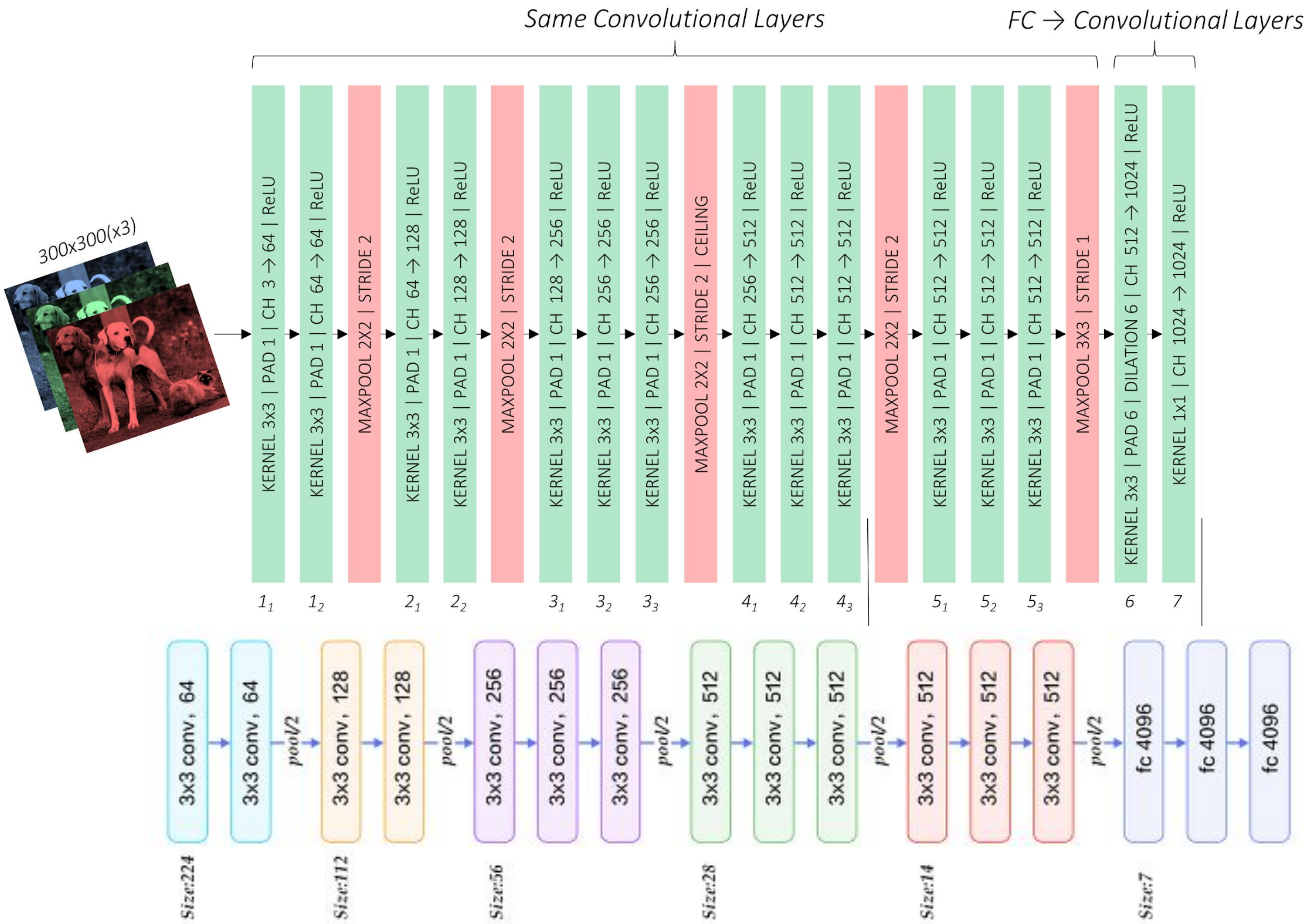

RoIAlign
class
box
conv
conv

https://github.com/chenyuntc/simple-faster-rcnn-pytorch

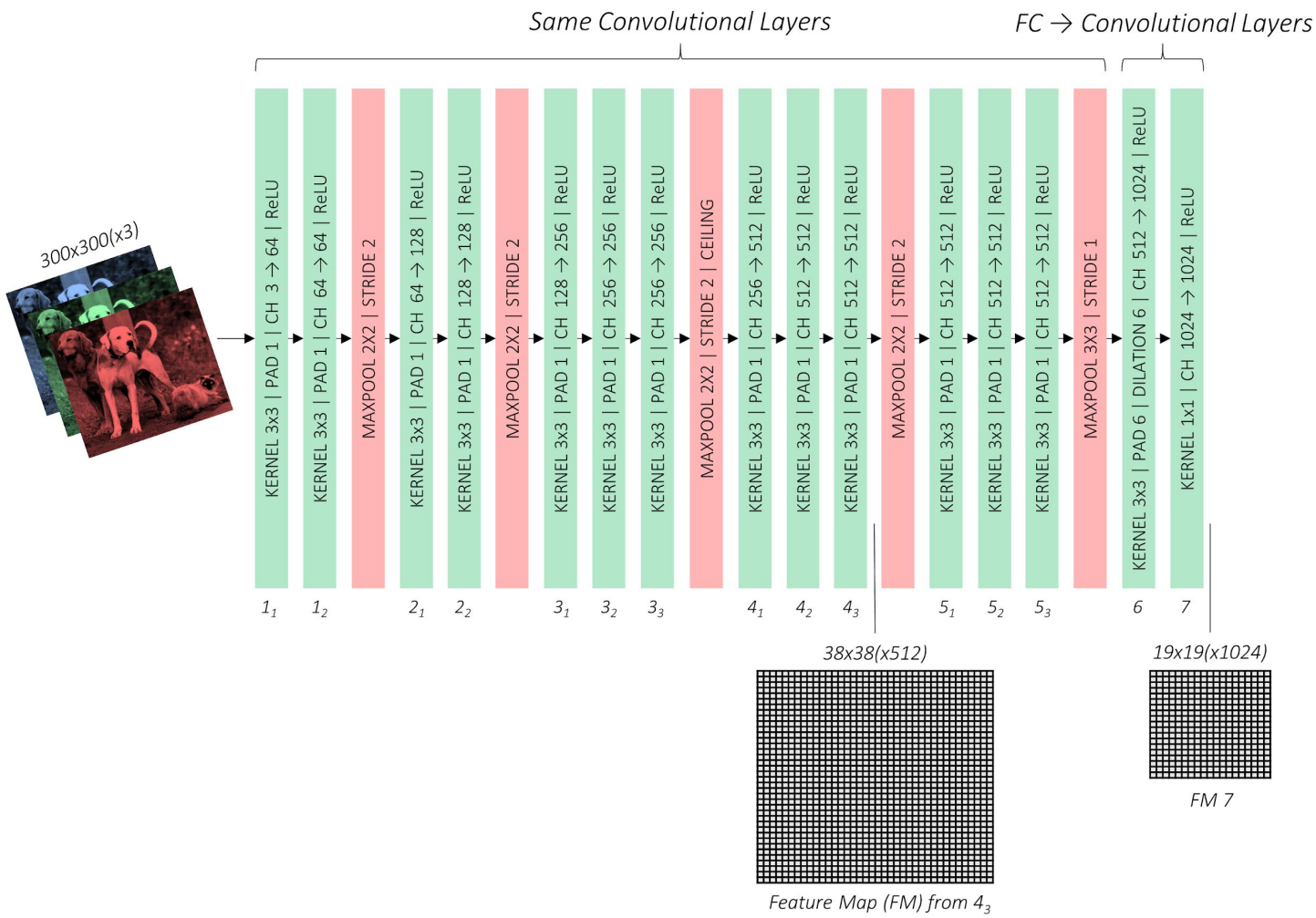
https://github.com/facebookresearch/maskrcnn-benchmark

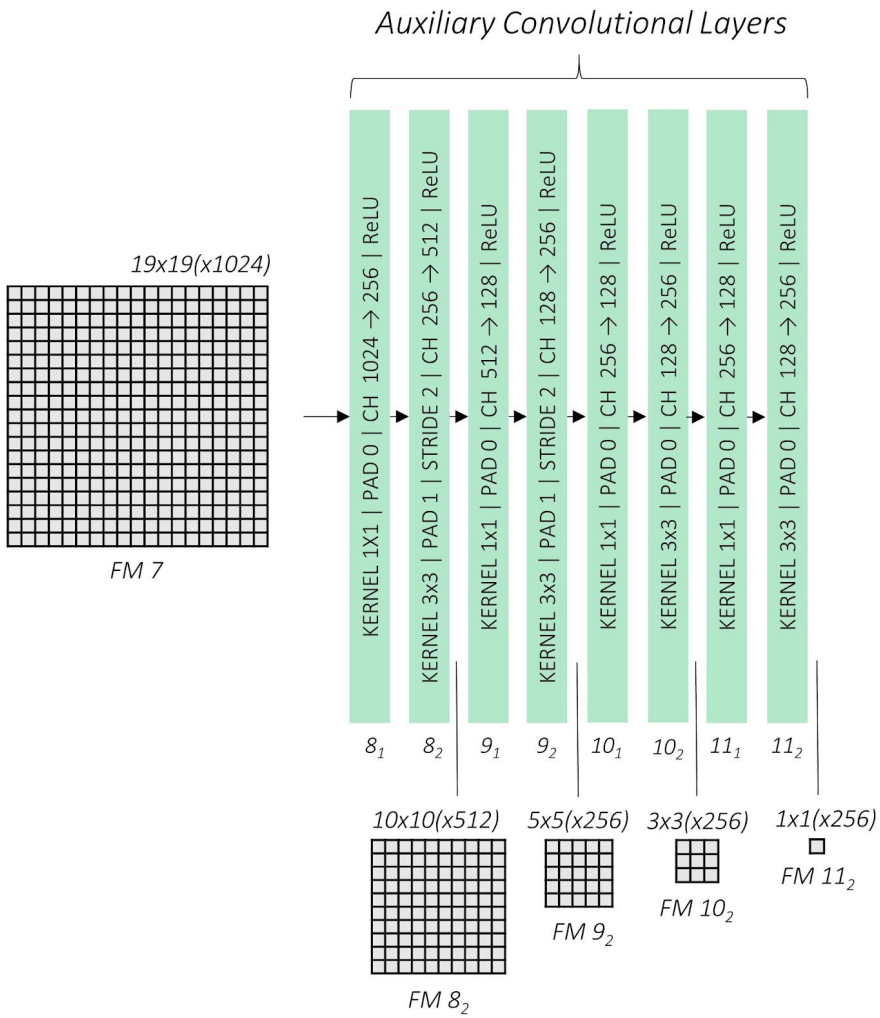
Single ShotmultiBox Detector

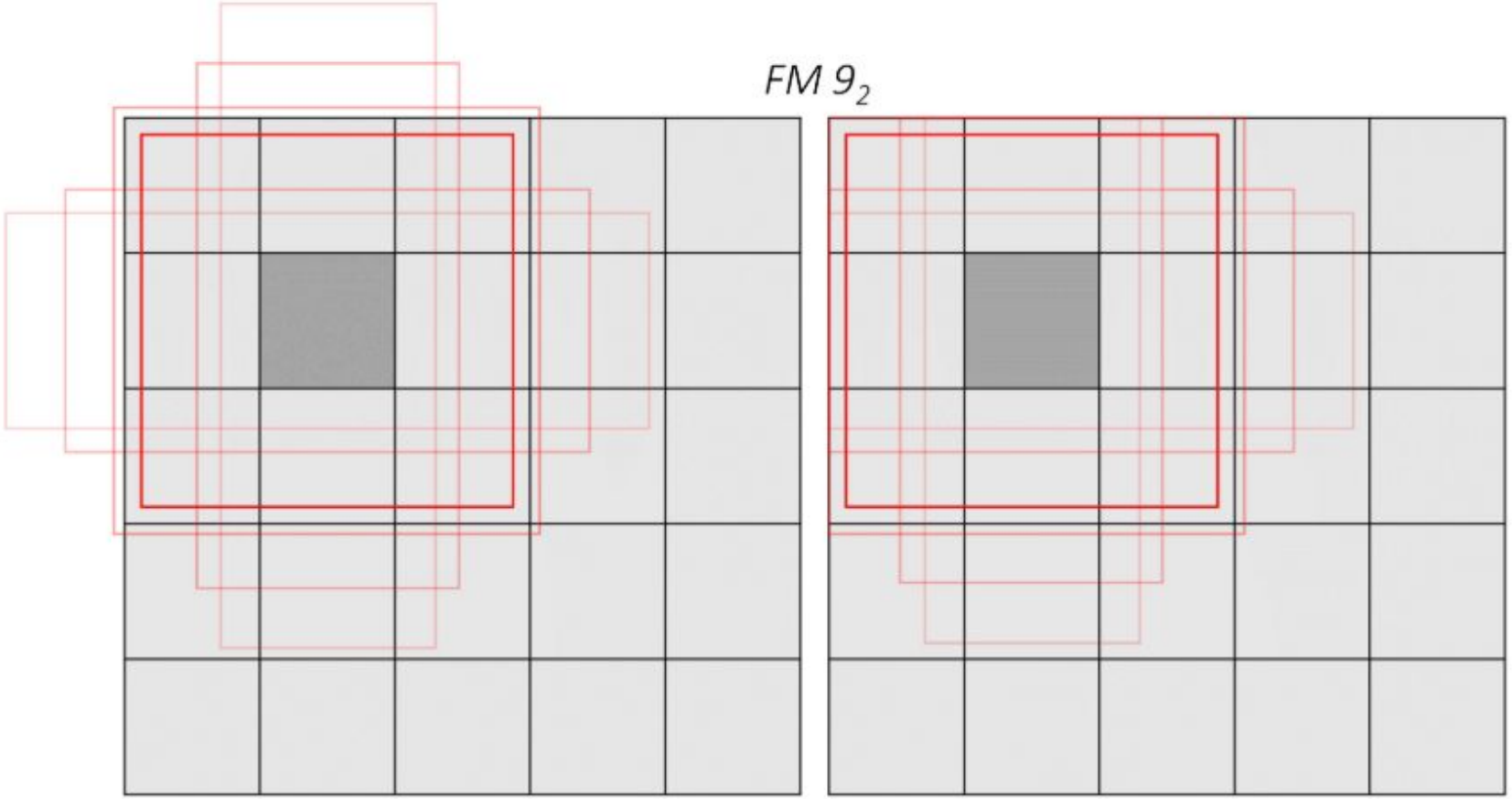
https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection



Same Convolutional Layers
FC → Convolutional Layers
300x300(x3)
KERNEL 3x3 | PAD 1 | CH 3 → 64 | ReLU
KERNEL 3x3 | PAD 1 | CH 64 → 64 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 64 → 128 | ReLU
KERNEL 3x3 | PAD 1 | CH 128 → 128 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 128 → 256 | ReLU
KERNEL 3x3 | PAD 1 | CH 256 → 256 | ReLU
KERNEL 3x3 | PAD 1 | CH 256 → 256 | ReLU
MAXPOOL 2X2 | STRIDE 2 | CEILING
KERNEL 3x3 | PAD 1 | CH 256 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
MAXPOOL 3X3 | STRIDE 1
KERNEL 3x3 | PAD 6 | DILATION 6 | CH 512 → 1024 | ReLU
KERNEL 1x1 | CH 1024 → 1024 | ReLU
1_1 1_2 2_1 2_2 3_1 3_2 3_3 4_1 4_2 4_3 5_1 5_2 5_3 6 7
3x3 conv, 64
3x3 conv, 64
pool/2
3x3 conv, 128
3x3 conv, 128
pool/2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
fc 4096
fc 4096
fc 4096
Size:224
Size:112
Size:56
Size:28
Size:14
Size:7

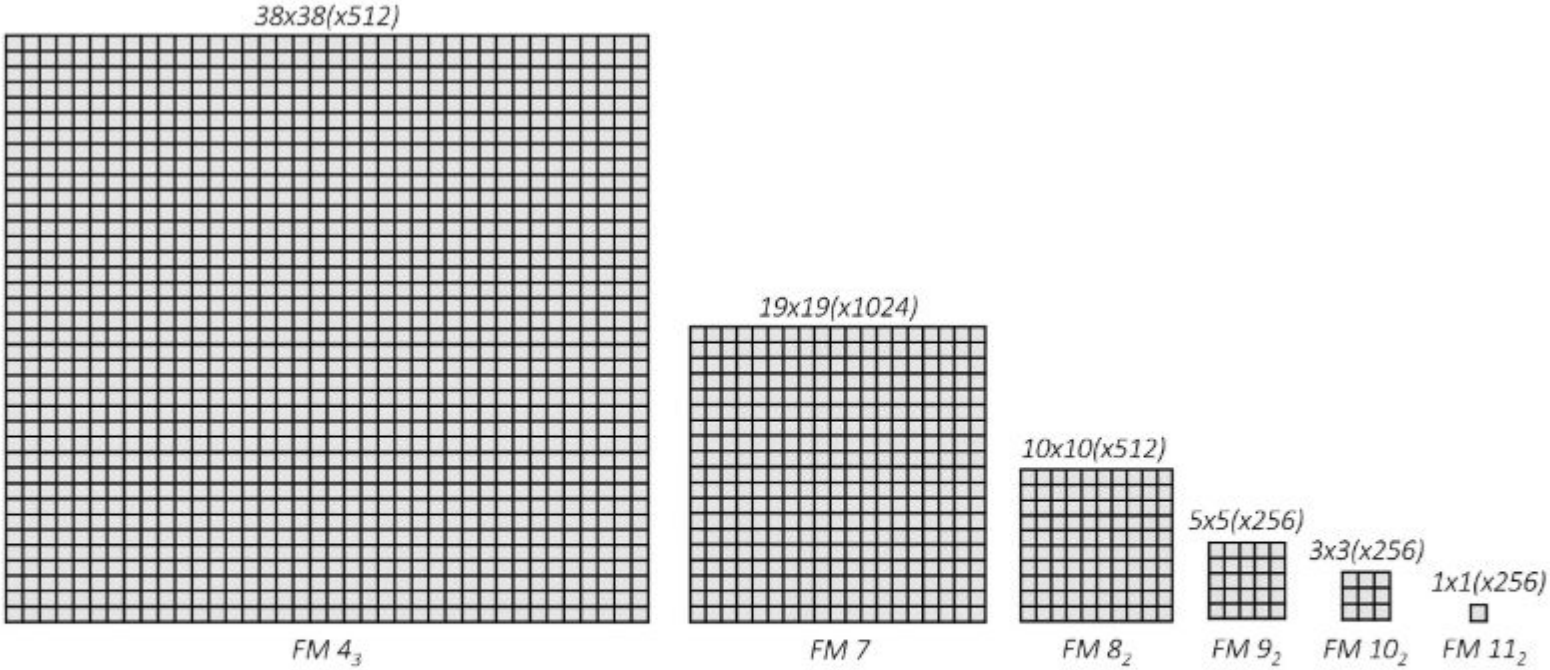

Same Convolutional Layers
FC → Convolutional Layers
300x300(x3)
KERNEL 3x3 | PAD 1 | CH 3 → 64 | ReLU
KERNEL 3x3 | PAD 1 | CH 64 → 64 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 64 → 128 | ReLU
KERNEL 3x3 | PAD 1 | CH 128 → 128 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 128 → 256 | ReLU
KERNEL 3x3 | PAD 1 | CH 256 → 256 | ReLU
KERNEL 3x3 | PAD 1 | CH 256 → 256 | ReLU
MAXPOOL 2X2 | STRIDE 2 | CEILING
KERNEL 3x3 | PAD 1 | CH 256 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
MAXPOOL 2X2 | STRIDE 2
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
KERNEL 3x3 | PAD 1 | CH 512 → 512 | ReLU
MAXPOOL 3X3 | STRIDE 1
KERNEL 3x3 | PAD 6 | DILATION 6 | CH 512 → 1024 | ReLU
KERNEL 1x1 | CH 1024 → 1024 | ReLU
1_1 1_2 2_1 2_2 3_1 3_2 3_3 4_1 4_2 4_3 5_1 5_2 5_3 6 7
38x38(x512)
19x19(x1024)
FM 7
Feature Map (FM) from 4_3


Auxiliary Convolutional Layers
19x19(x1024)
FM 7
KERNEL 1X1 | PAD 0 | CH 1024 → 256 | ReLU
KERNEL 3x3 | PAD 1 | STRIDE 2 | CH 256 → 512 | ReLU
KERNEL 1x1 | PAD 0 | CH 512 → 128 | ReLU
KERNEL 3x3 | PAD 1 | STRIDE 2 | CH 128 → 256 | ReLU
KERNEL 1x1 | PAD 0 | CH 256 → 128 | ReLU
KERNEL 3x3 | PAD 0 | CH 128 → 256 | ReLU
KERNEL 1x1 | PAD 0 | CH 256 → 128 | ReLU
KERNEL 3x3 | PAD 0 | CH 128 → 256 | ReLU
8_1 8_2 9_1 9_2 10_1 10_2 11_1 11_2
10x10(x512)
FM 8_2
5x5(x256)
FM 9_2
3x3(x256)
FM 10_2
1x1(x256)
FM 11_2



When priors at a location overshoot the edges of the feature map, they are clipped

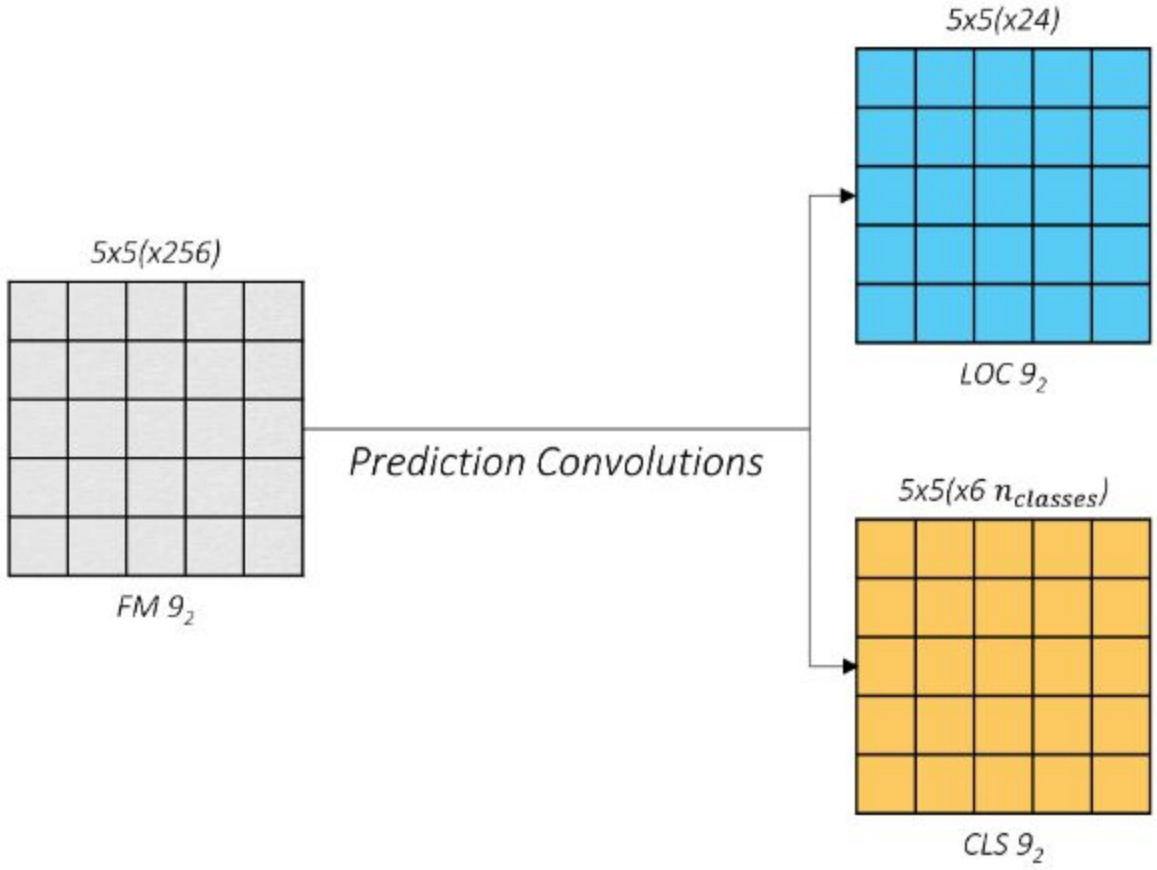
Feature Map From	Feature Map Dimensions	Prior Scale	Aspect Ratios	Number of Priors per Position	Total Number of Priors on this Feature Map
`conv4_3`	38, 38	0.1	1:1, 2:1, 1:2 + an extra prior	4	5776
`conv7`	19, 19	0.2	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	2166
`conv8_2`	10, 10	0.375	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	600
`conv9_2`	5, 5	0.55	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	150
`conv10_2`	3, 3	0.725	1:1, 2:1, 1:2 + an extra prior	4	36
`conv11_2`	1, 1	0.9	1:1, 2:1, 1:2 + an extra prior	4	4
Grand Total	–	–	–	–	**8732 priors**

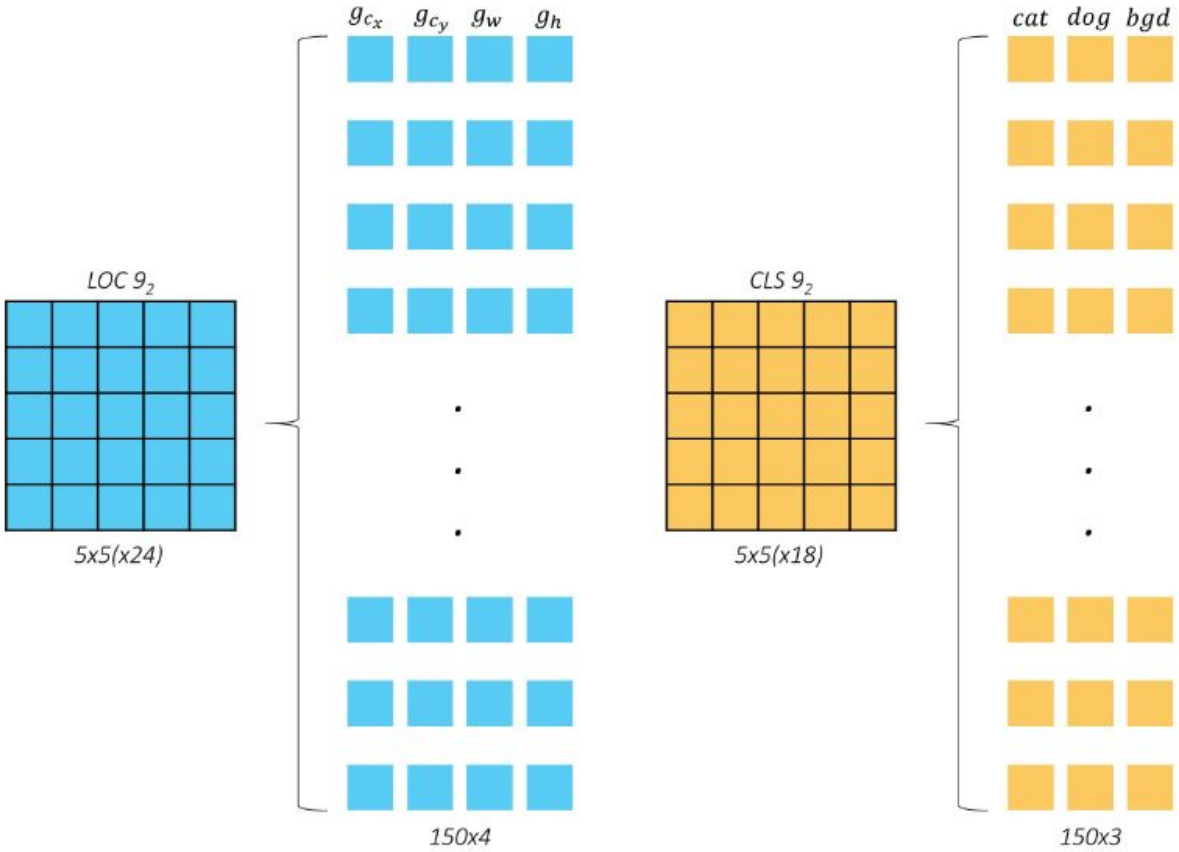


Prediction Convolutions

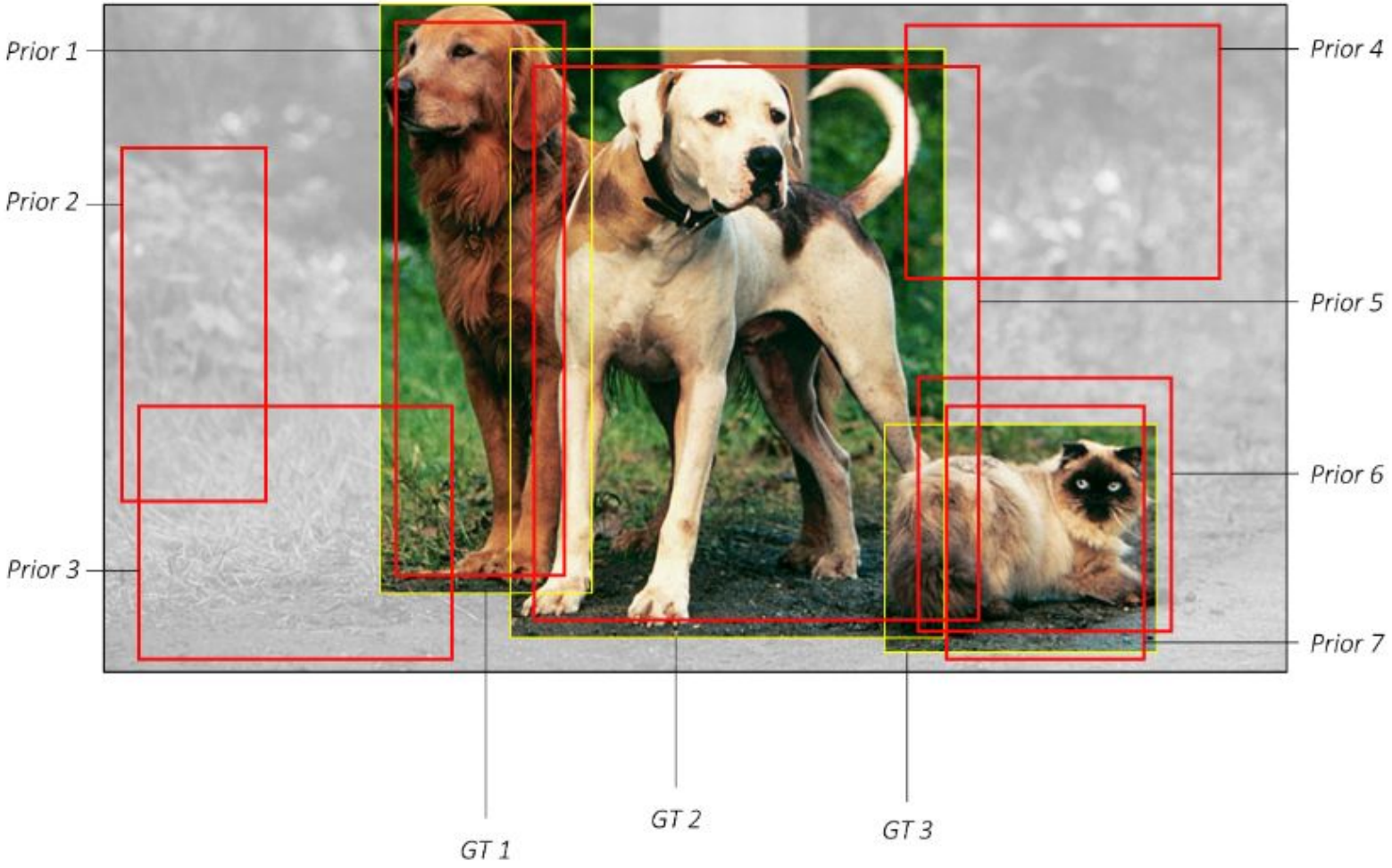
KERNEL 3X3 | PAD 1 |
CH __ → number of priors per location for this feature map * 4

KERNEL 3X3 | PAD 1 |
CH __ → number of priors per location for this feature map * number of object classes


5x5(x24)
5x5(x256)
Prediction Convolutions
5x5(x6 $n_{classes}$)
LOC 9_2
FM 9_2
CLS 9_2



Reshape predictions from FM 9_2 to represent offsets and class scores for the 150 predicted boxes


Prior 1
Prior 2
Prior 3
Prior 4
Prior 5
Prior 6
Prior 7
GT 1
GT 2
GT 3

IoU	*GT 1*	*GT 2*	*GT 3*
Prior 1	0.75	0.07	0
Prior 2	0	0	0
Prior 3	0.05	0	0
Prior 4	0	0.03	0
Prior 5	0.05	0.88	0.06
Prior 6	0	0.03	0.65
Prior 7	0	0	0.68

	Matched GT	*Type*	*Label*	*Coordinates*
Prior 1	GT 1	Positive	*dog*	of GT 1
Prior 2	GT 1	Negative	*bgd*	-
Prior 3	GT 1	Negative	*bgd*	-
Prior 4	GT 2	Negative	*bgd*	-
Prior 5	GT 2	Positive	*dog*	of GT 2
Prior 6	GT 3	Positive	*cat*	of GT 3
Prior 7	GT 3	Positive	*cat*	of GT 3