Feature Pyramid Networks for Object Detection (2017)

Tsung-Yi Lin • Piotr Dollar • Ross Girshick
Kaiming He • Bharath Hariharan • Serge Belongie

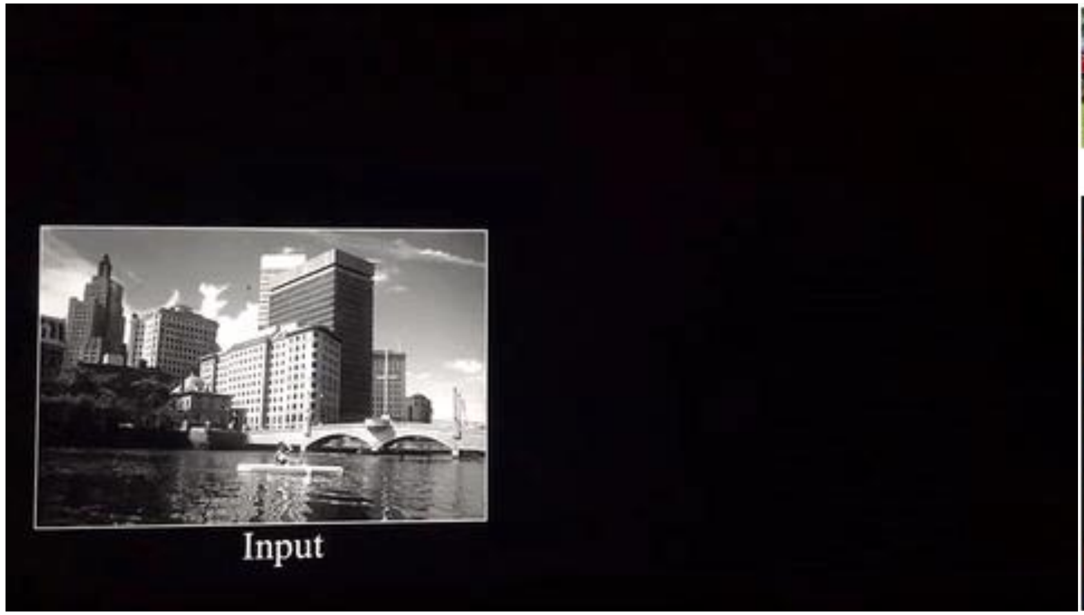
Facebook AI Research
Cornell University and Cornell Tech

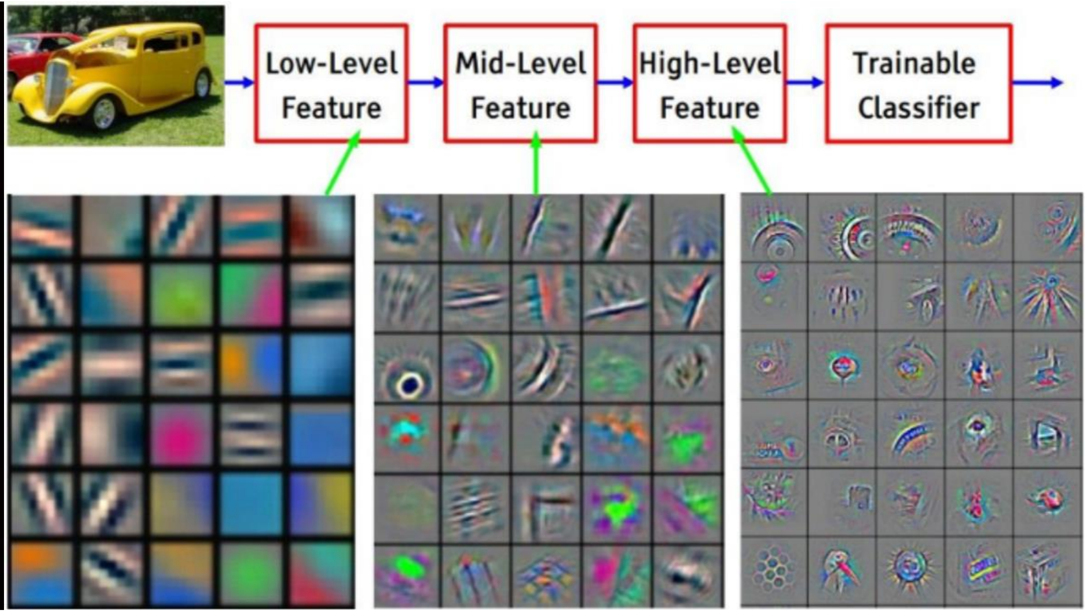
Presented by
Selahaddin HONİ

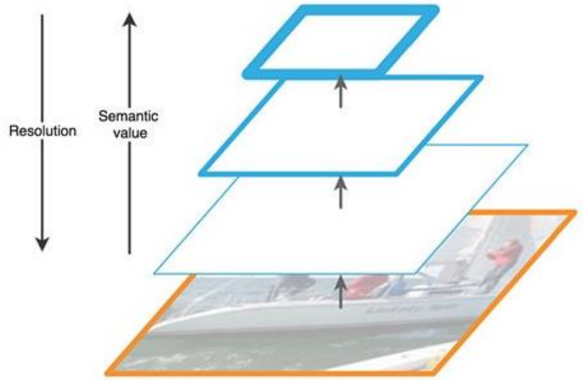
Digital Video Processing
Spring 2021

Content

Introduction | Feature Maps Refresh :: Resolution & Semantic





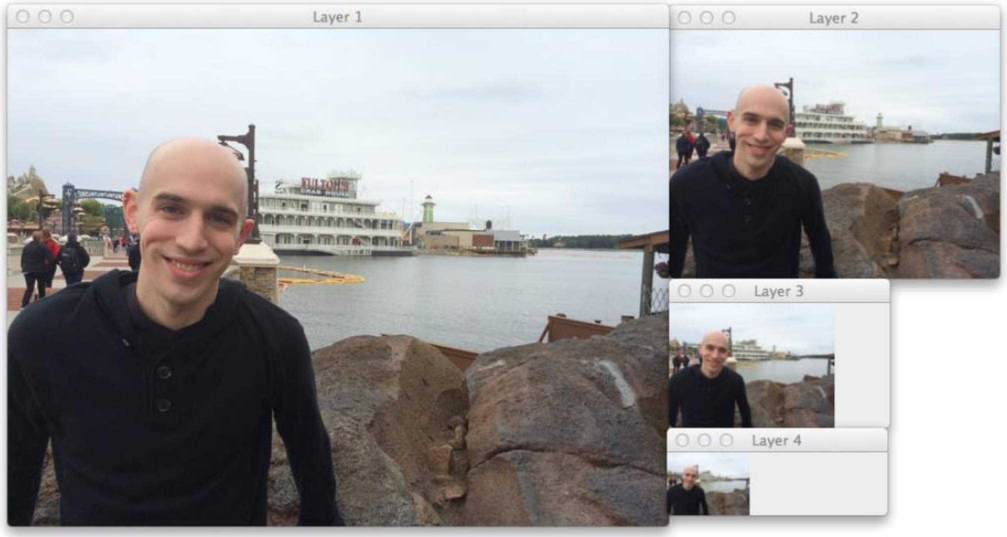


Semantic value increases with higher level features

Reference: [Feature Maps. Feature Map is also called as… | by Chris Kevin | Medium]

Introduction | Image Pyramid & Sliding Windows for Object Detection

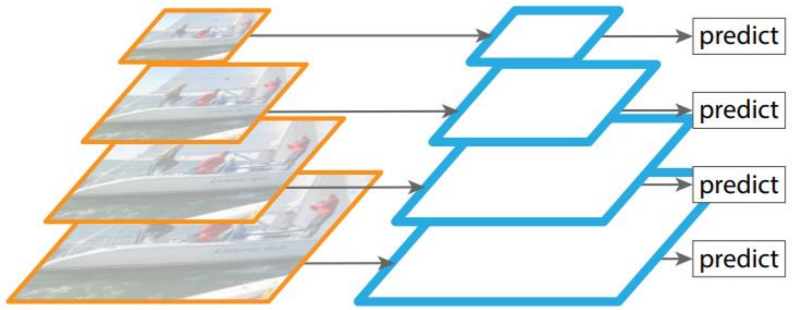
Image pyramid is simply the scale variations of an image. Main purpose is detecting an object in different scales. **Sliding windows** is a basic search mechanism.



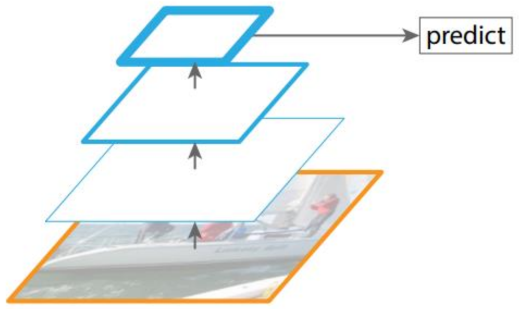
Reference: Sliding Windows for Object Detection with Python and OpenCV - PyImageSearch

Literature Overview

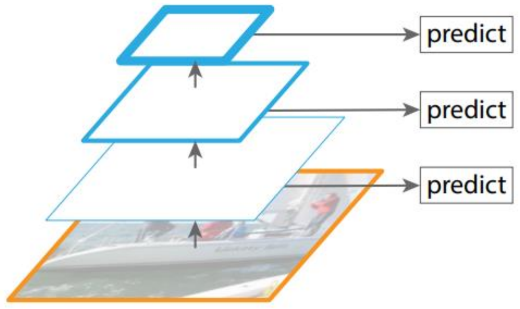
Proposed network architectures for detecting in different scales



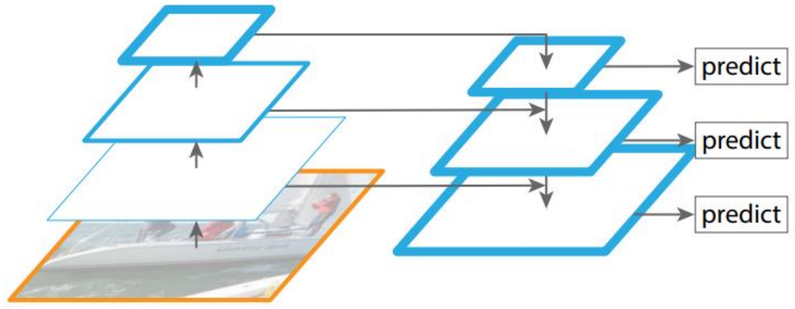
(a) Featurized image pyramid



(b) Single feature map

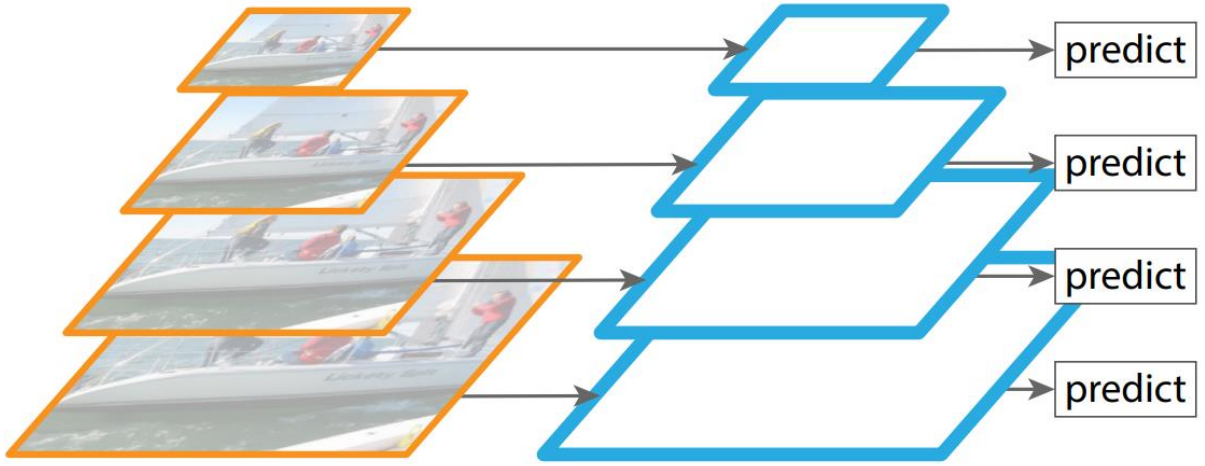


(c) Pyramidal feature hierarchy

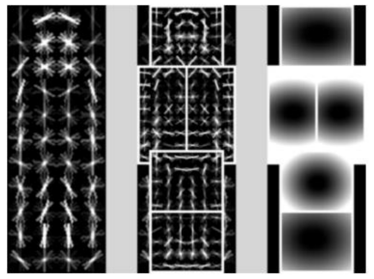


(d) Feature Pyramid Network

Literature Review | (a) Featurized Image Pyramid



Problem is too slow.


model

Deformable Parts Model

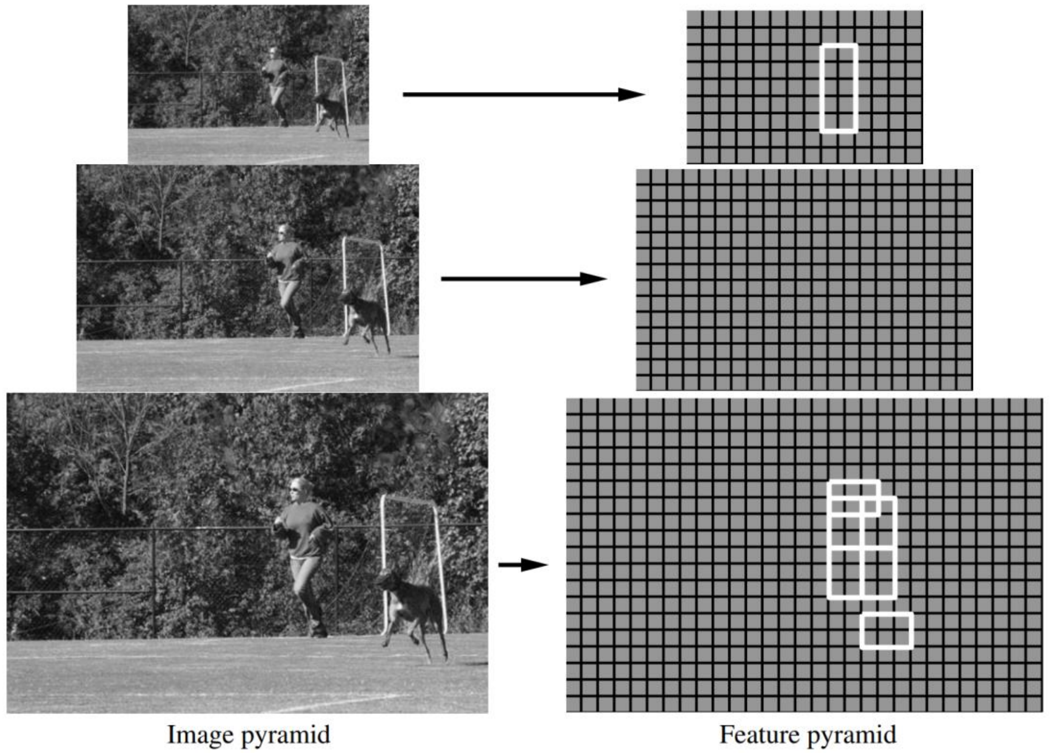
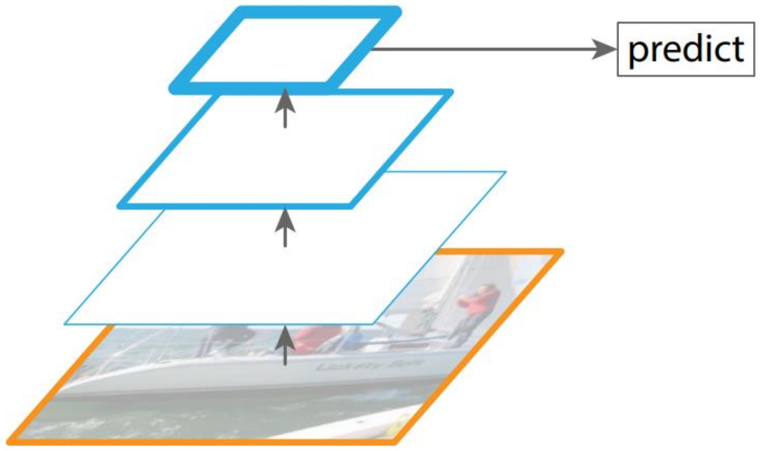


Fig. 3. A feature pyramid and an instantiation of a person model within that pyramid. The part filters are placed at twice the spatial resolution of the placement of the root.

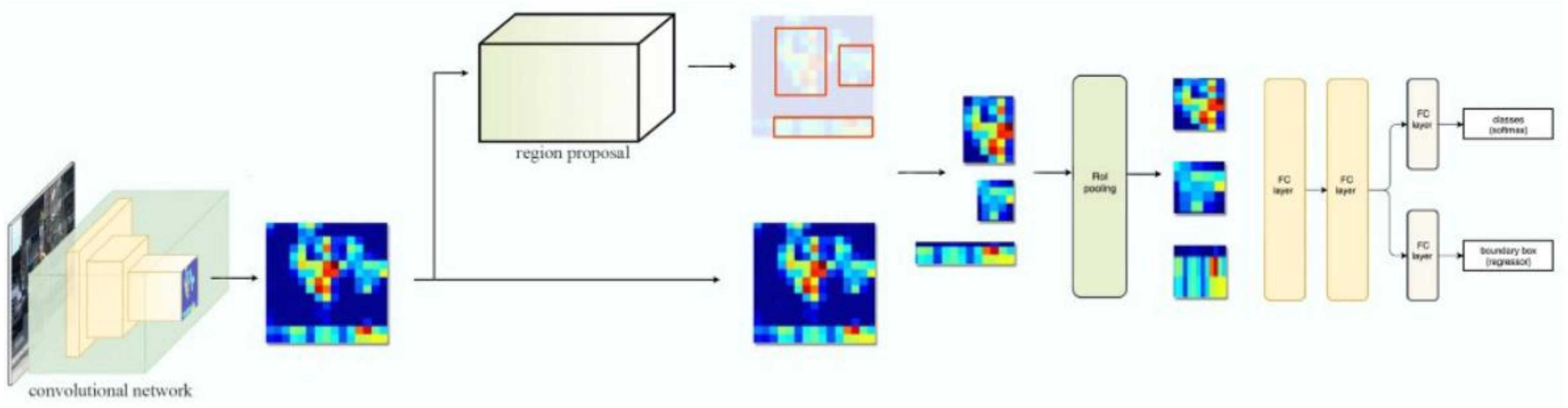
Reference: Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester and Deva Ramanan. (2009) “Object Detection with Discriminatively Trained Part Based Models”

Literature Review | (b) Single Feature Map



Problem
RPN only use the final feature map of the backbone to propose the regions.

Faster R-CNN



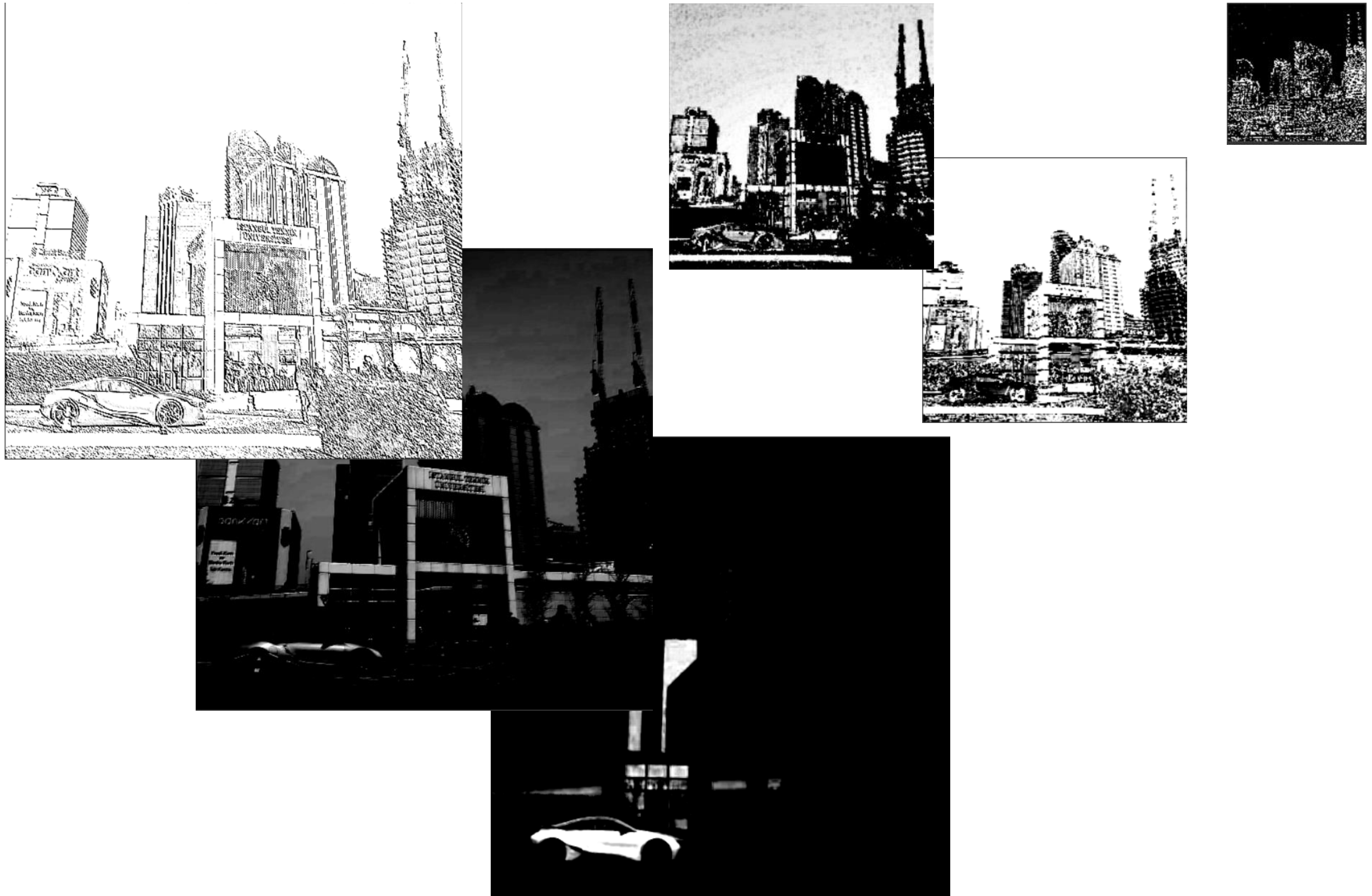
Literature Review | (b) Single Feature Map :: Faster R-CNN (ResNet50 – COCO)

Input Image

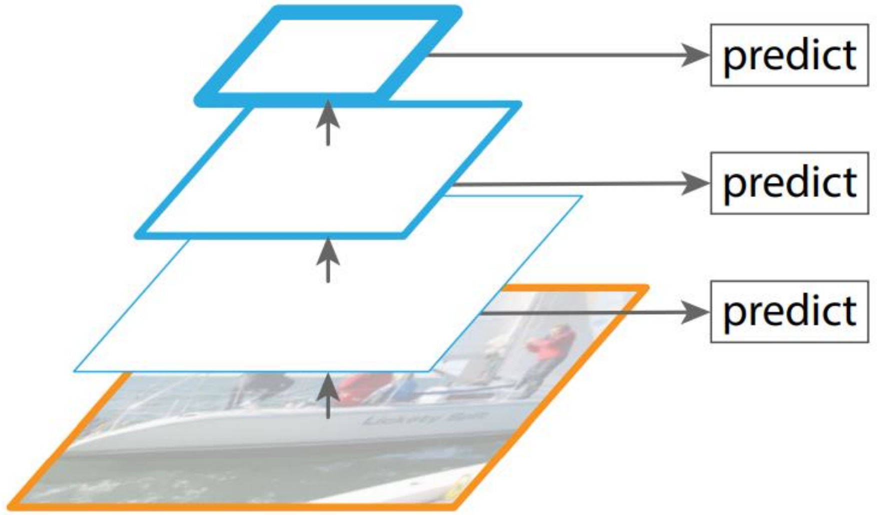
First Conv Block (512x512)

Second Conv Block (256x256)

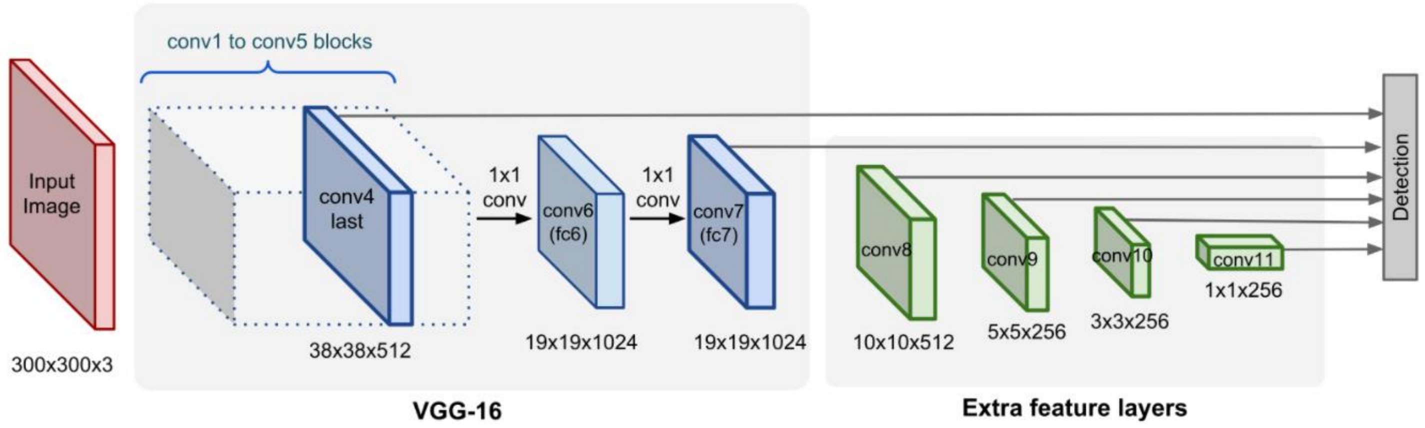
Third Conv Block (128x128)



Literature Review | (c) Pyramidal Feature Hierarchy



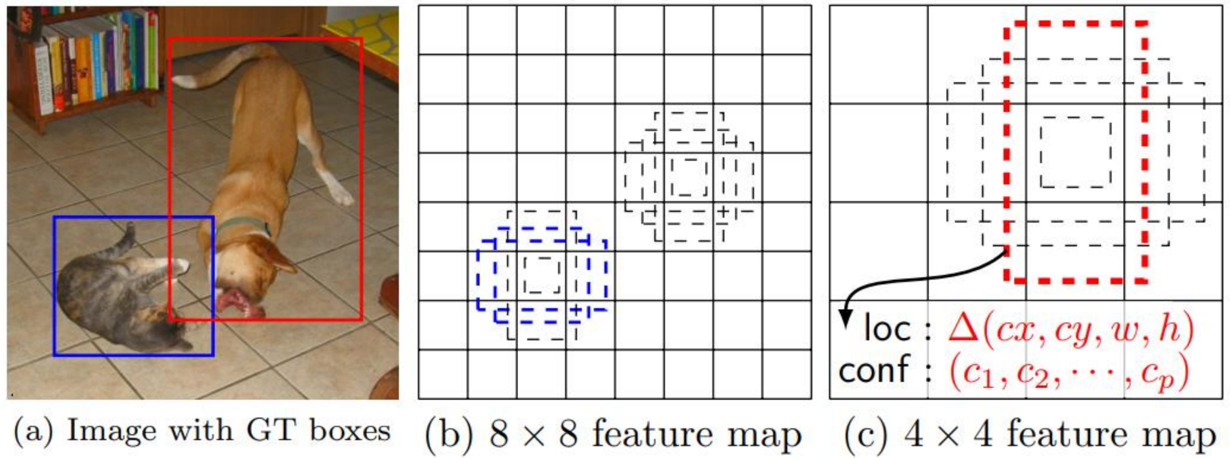
Single Shot Detector



SSD predicts offset of predefined anchor boxes ('default boxes' in the paper[1]) for every location of the feature map.

Problem is SSD misses to use the low-level features or high resolution map. It is important for detecting small objects.

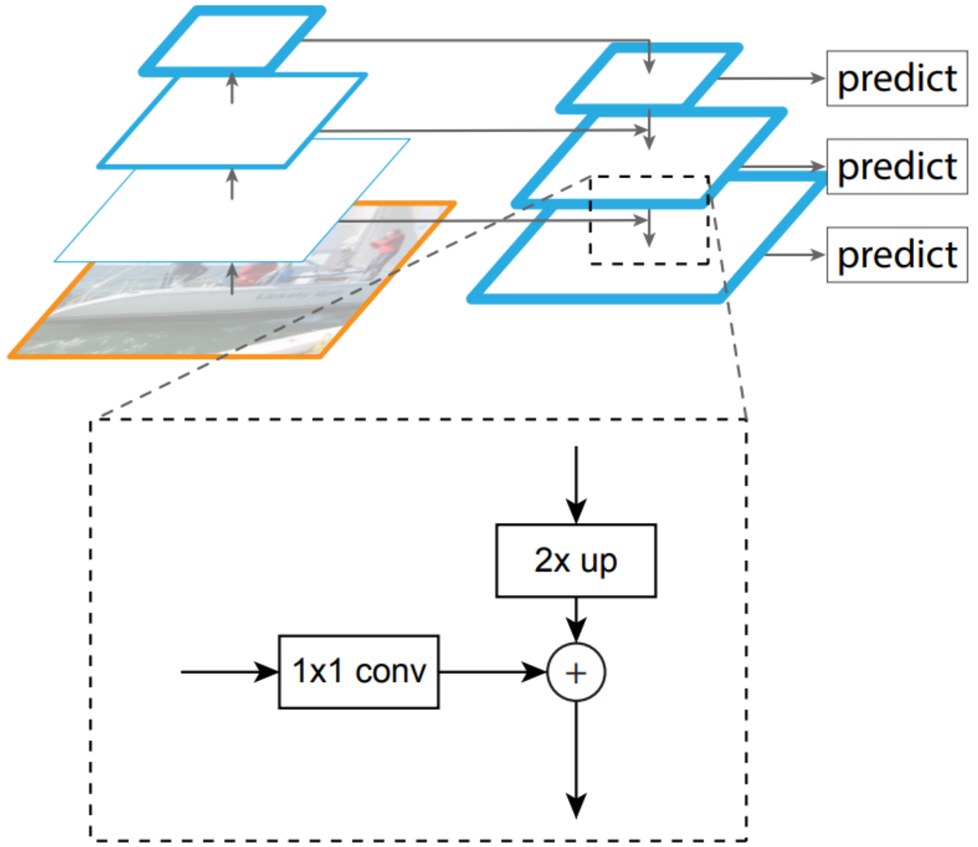
Hierarchy starts from at the end of the backbone.


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

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

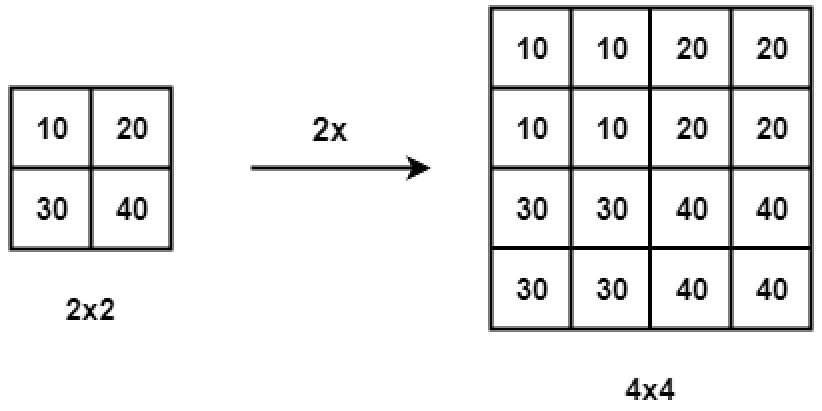
[1] Wei Liu , Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg (2016). "SSD: Single Shot MultiBox Detector"
[2] Object Detection Part 4: Fast Detection Models (lilianweng.github.io)

Feature Pyramid Networks | Theory

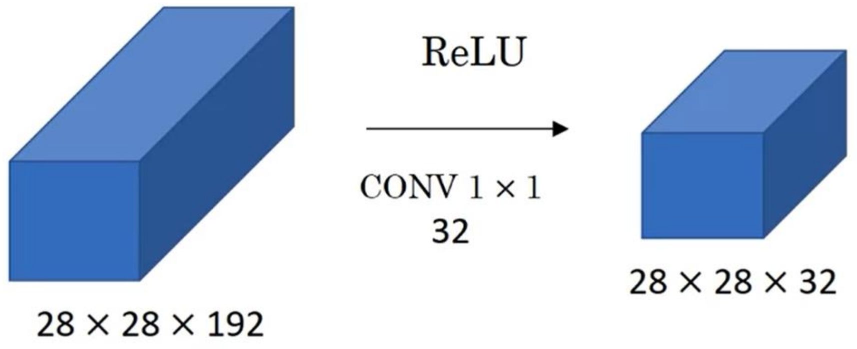


Combination of Strong Semantics & High Resolution

Upsampling (Nearest Neighbor)

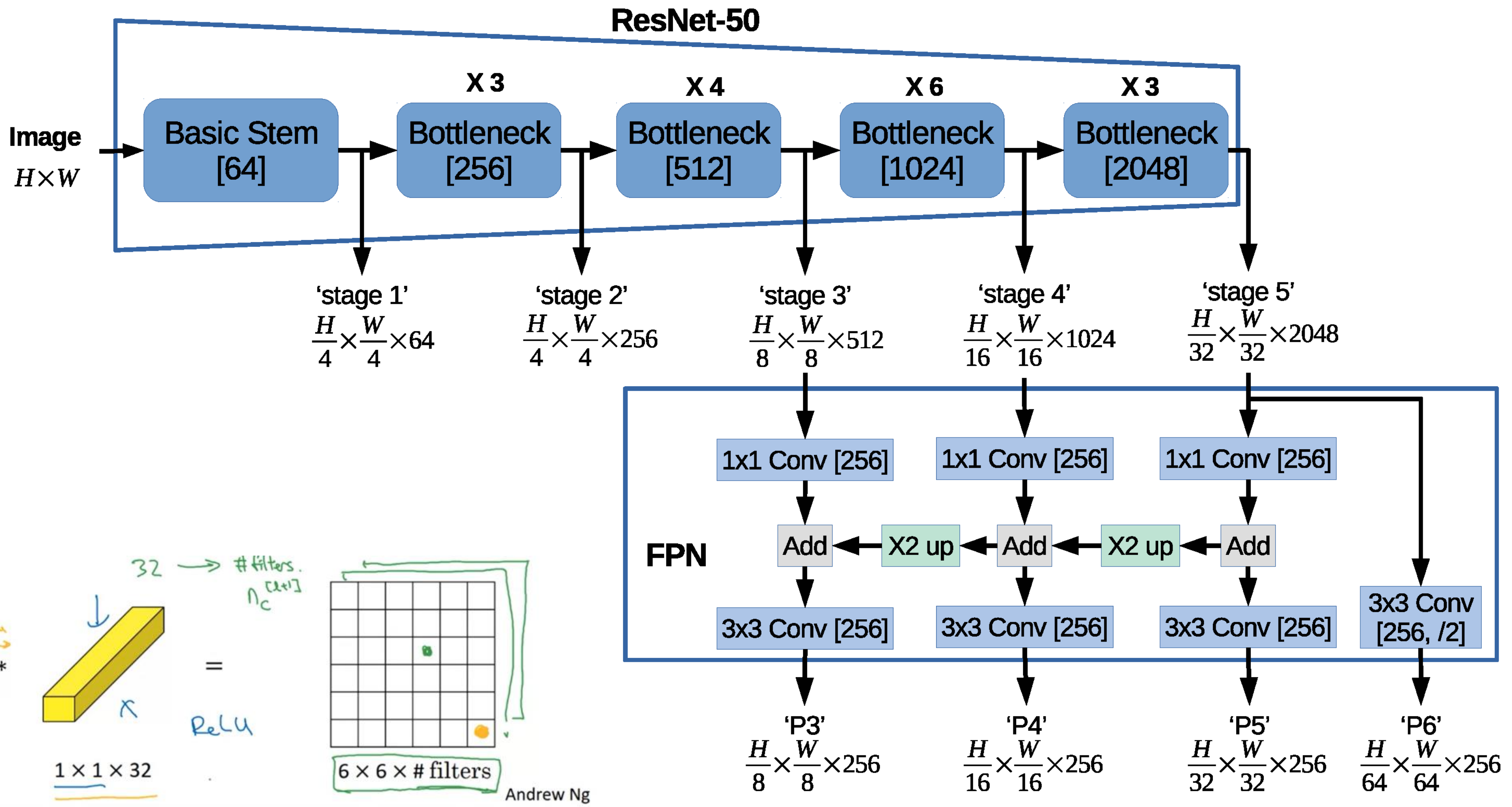


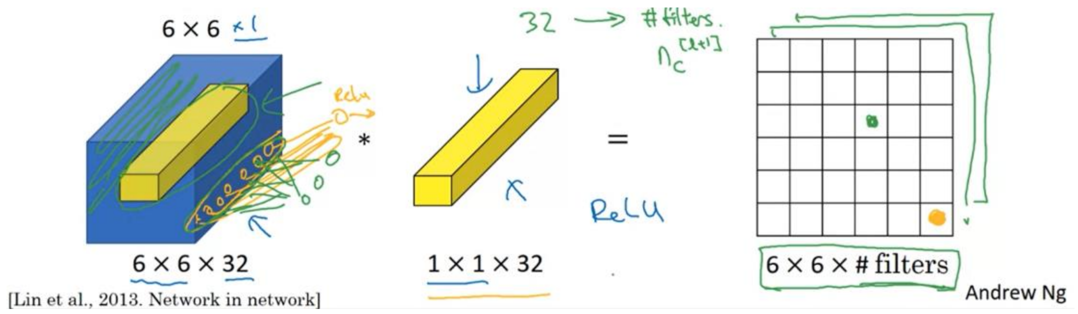
1x1 conv adjusts the channel dimension and adds non-linearity



[1] YasinEnigma/Image_Interpolation: Image interpolation implementation using pure python and compare result with Opencv. (github.com)
[2] Networks in Networks and 1x1 Convolutions - Deep Convolutional Models: Case Studies | Coursera

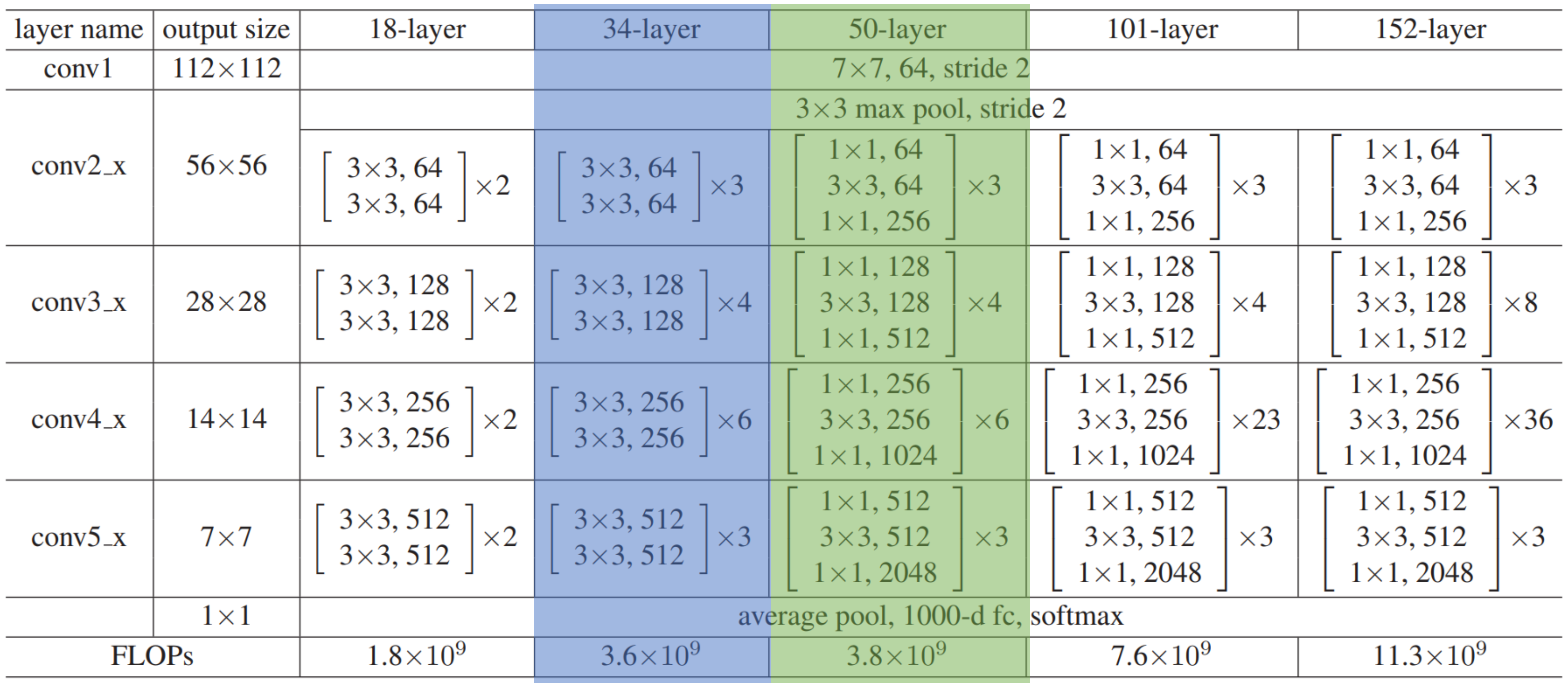
Feature Pyramid Networks | Theory :: FPN on ResNet-50


ResNet-50
Image
$H \times W$
Basic Stem [64]
X 3
Bottleneck [256]
X 4
Bottleneck [512]
X 6
Bottleneck [1024]
X 3
Bottleneck [2048]
'stage 1' $\frac{H}{4} \times \frac{W}{4} \times 64$
'stage 2' $\frac{H}{4} \times \frac{W}{4} \times 256$
'stage 3' $\frac{H}{8} \times \frac{W}{8} \times 512$
'stage 4' $\frac{H}{16} \times \frac{W}{16} \times 1024$
'stage 5' $\frac{H}{32} \times \frac{W}{32} \times 2048$
FPN
1x1 Conv [256]
1x1 Conv [256]
1x1 Conv [256]
Add
X2 up
Add
X2 up
Add
3x3 Conv [256]
3x3 Conv [256]
3x3 Conv [256]
3x3 Conv [256, /2]
'P3' $\frac{H}{8} \times \frac{W}{8} \times 256$
'P4' $\frac{H}{16} \times \frac{W}{16} \times 256$
'P5' $\frac{H}{32} \times \frac{W}{32} \times 256$
'P6' $\frac{H}{64} \times \frac{W}{64} \times 256$



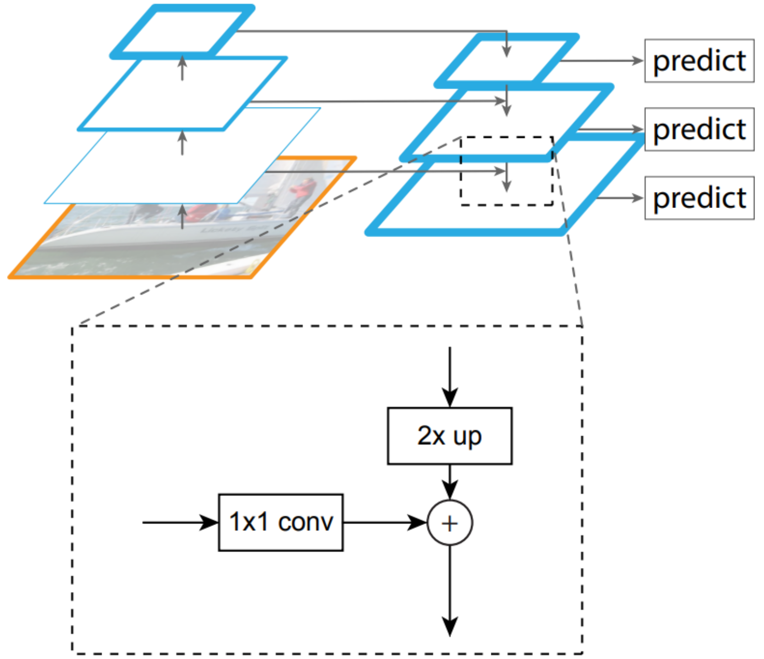
Reference: Panero Martinez, Ruben; Schiopu, Ionut; Cornelis, Bruno; Munteanu, Adrian. 2021. "Real-Time Instance Segmentation of Traffic Videos for Embedded Devices" Sensors 21, no. 1: 275. https://doi.org/10.3390/s21010275

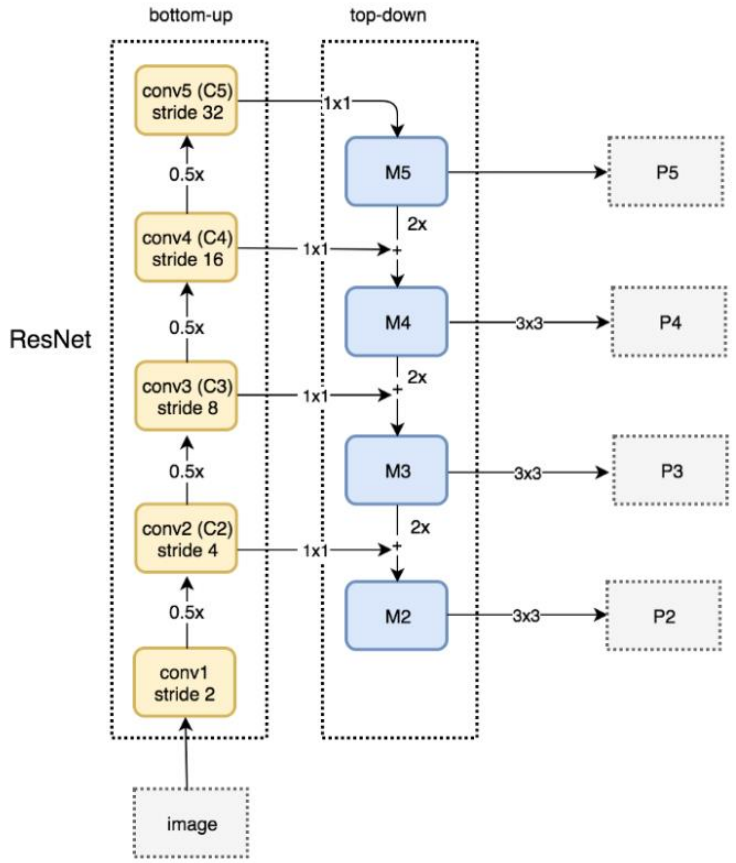
Feature Pyramid Networks | Theory :: ResNet Details

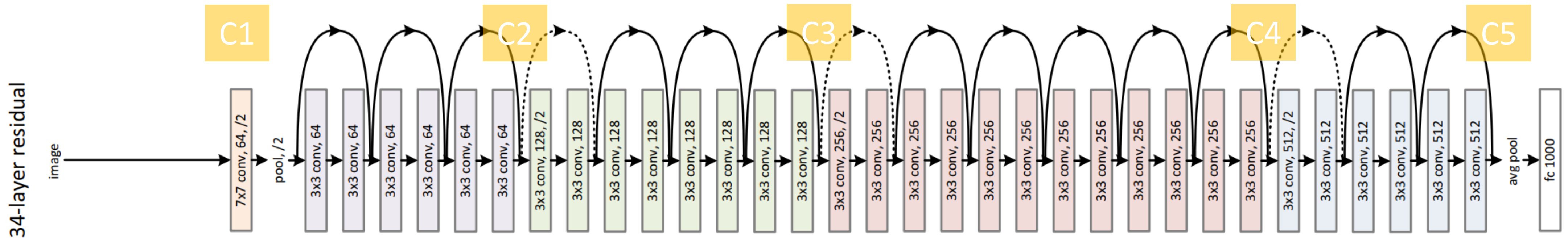


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Feature Pyramid Networks | Theory :: FPN on ResNet-34

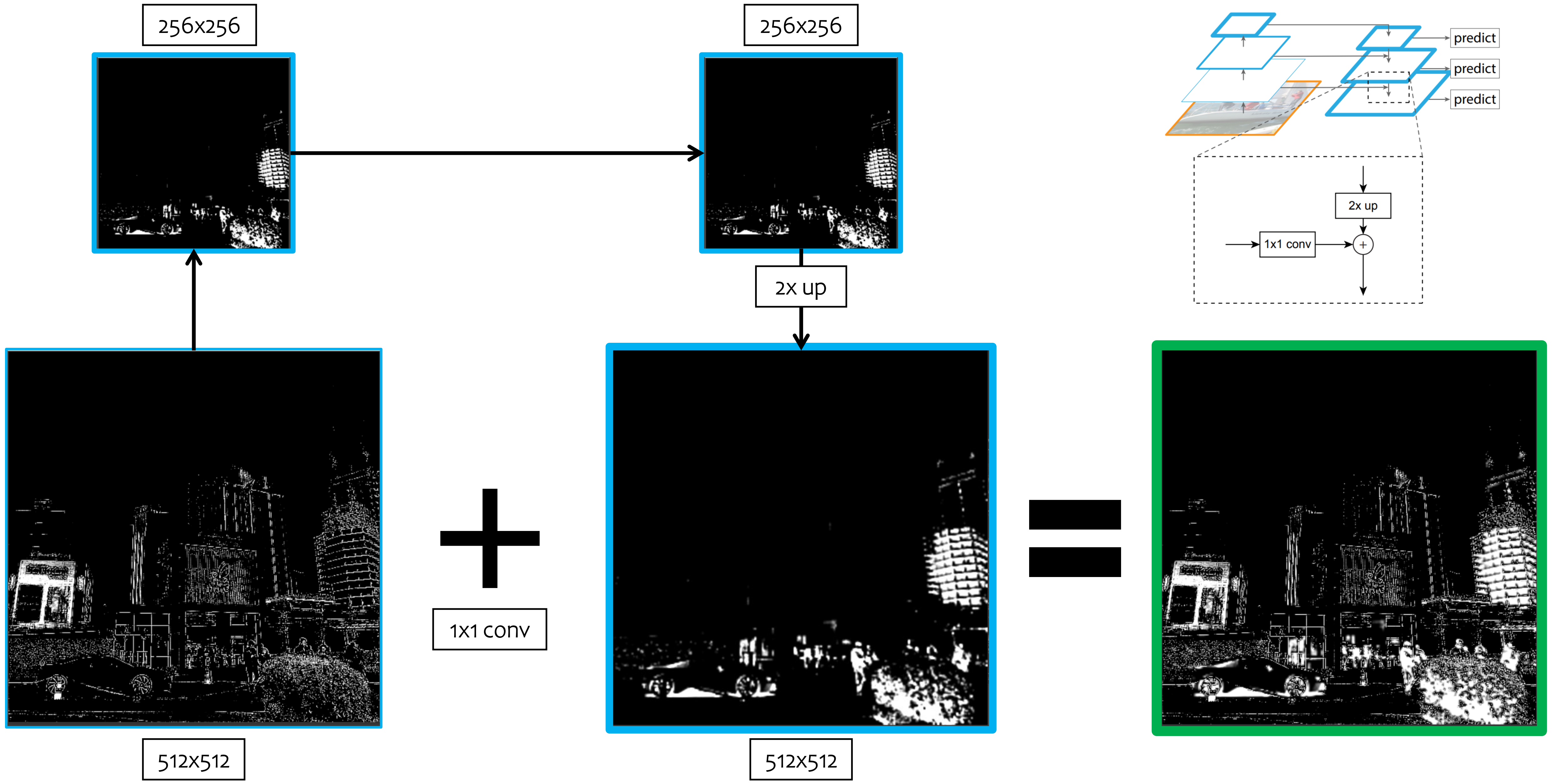


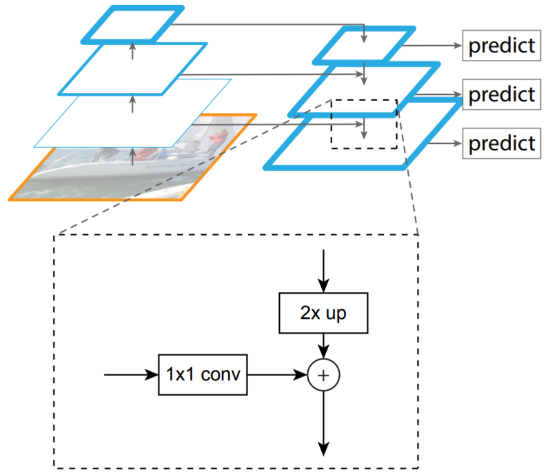




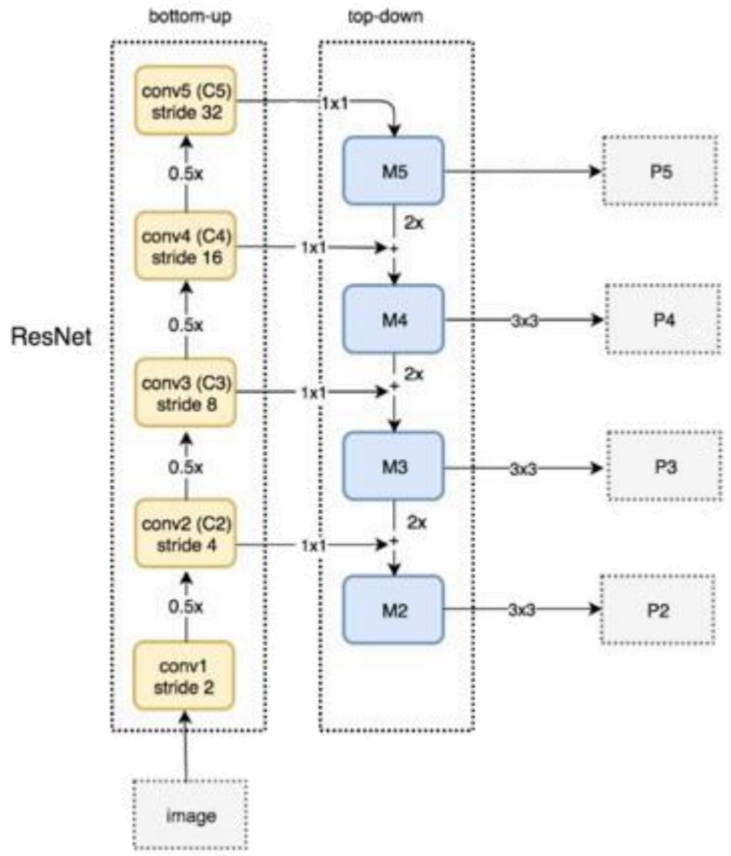
Reference: [Understanding Feature Pyramid Networks for object detection (FPN) | by Jonathan Hui | Medium]

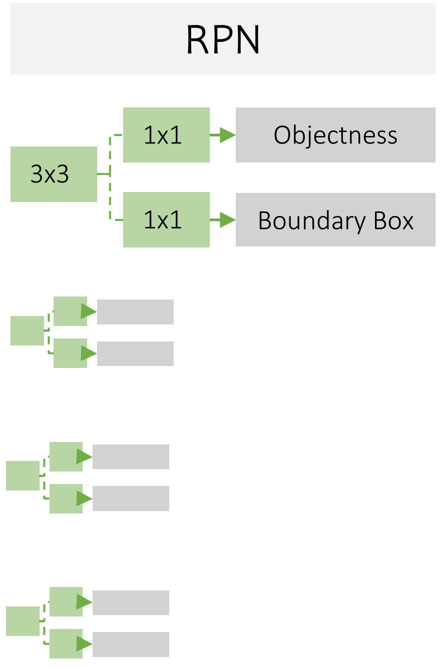
Feature Pyramid Networks | Illustration :: FPN on ResNet-50 (Not Actual Implementation)





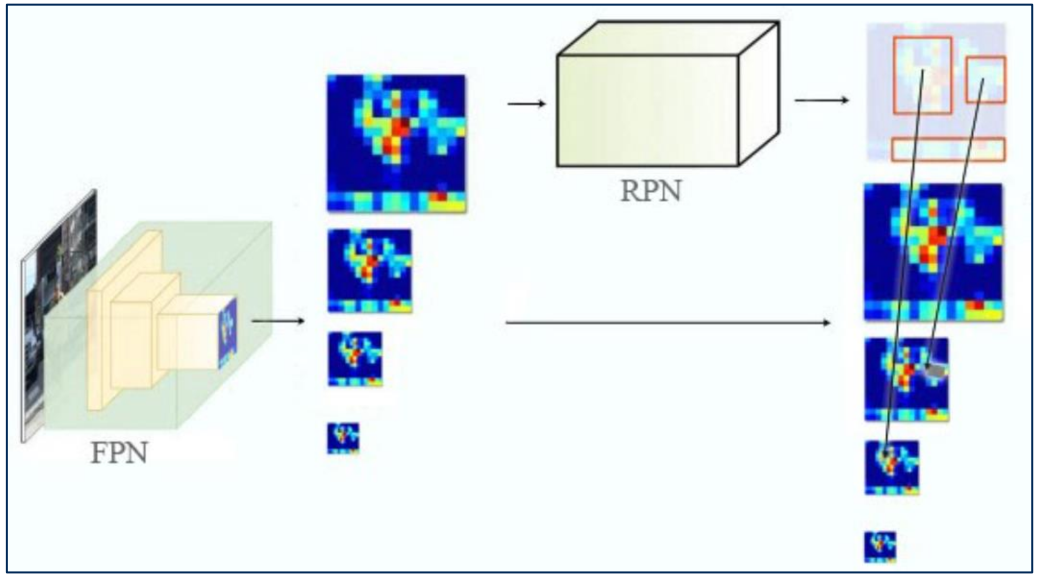
Feature Pyramid Networks | Region Proposal Network (RPN)





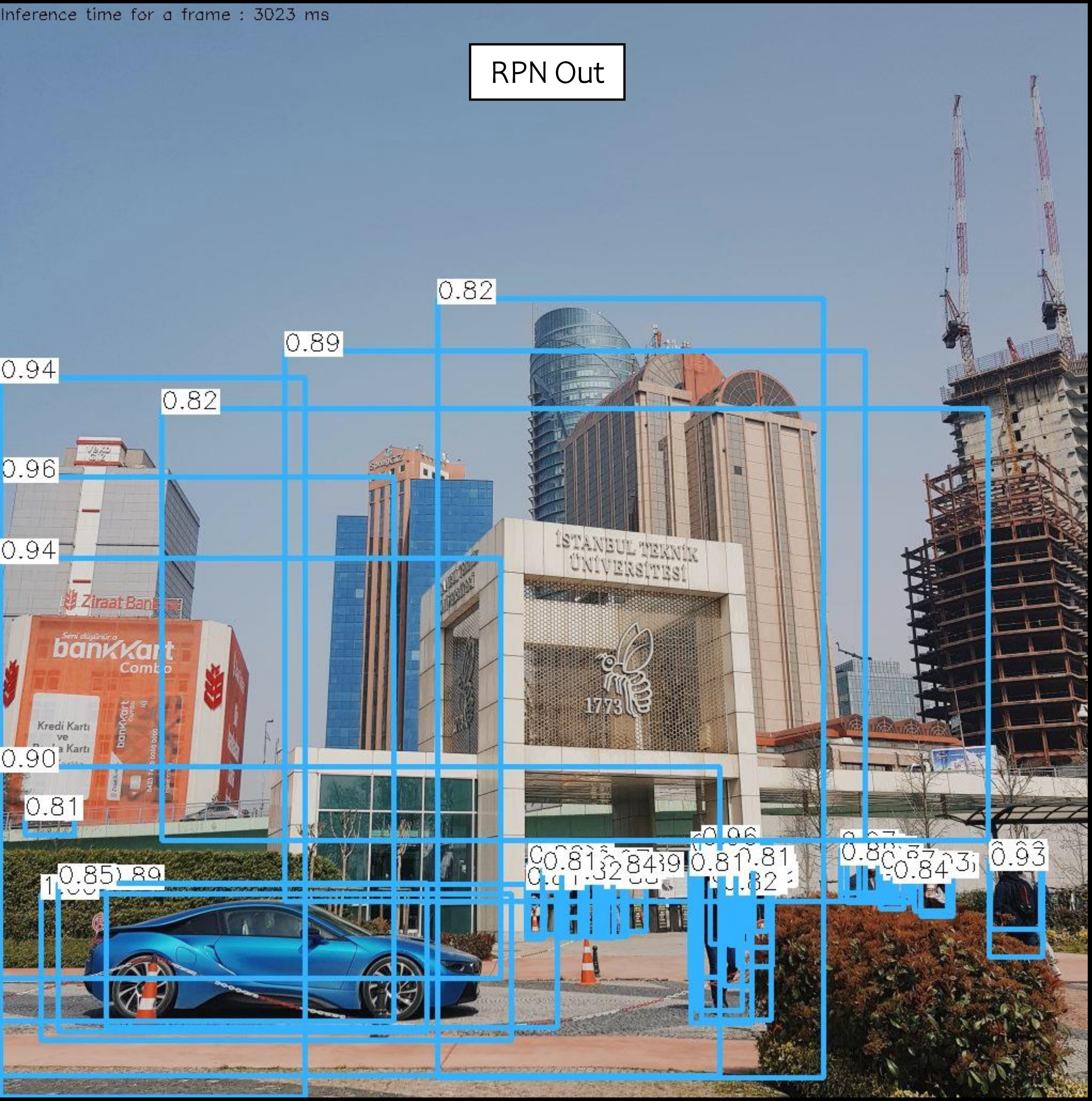
FPN-based RPN generates more anchors because more than one feature maps are fed into RPN input.

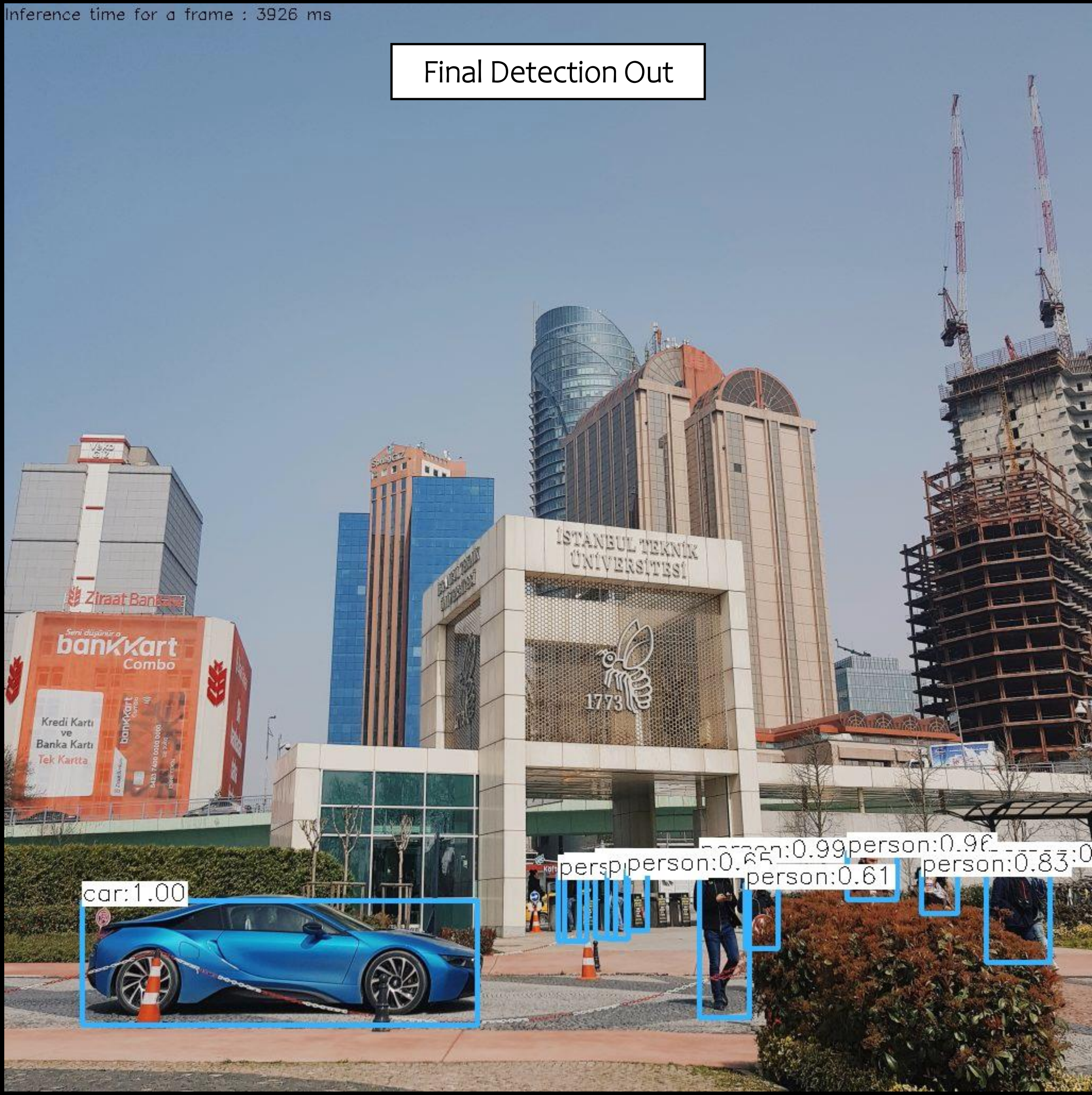
First Stage of Faster R-CNN



Reference: [Understanding Feature Pyramid Networks for object detection (FPN) | by Jonathan Hui | Medium]

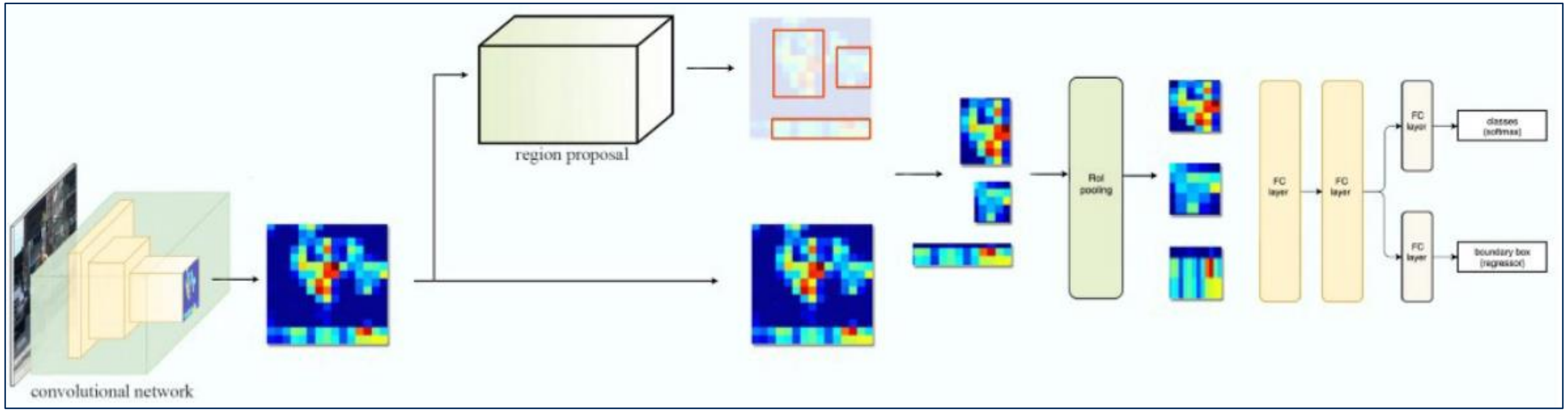
Feature Pyramid Networks | RPN & Faster R-CNN Detection Out



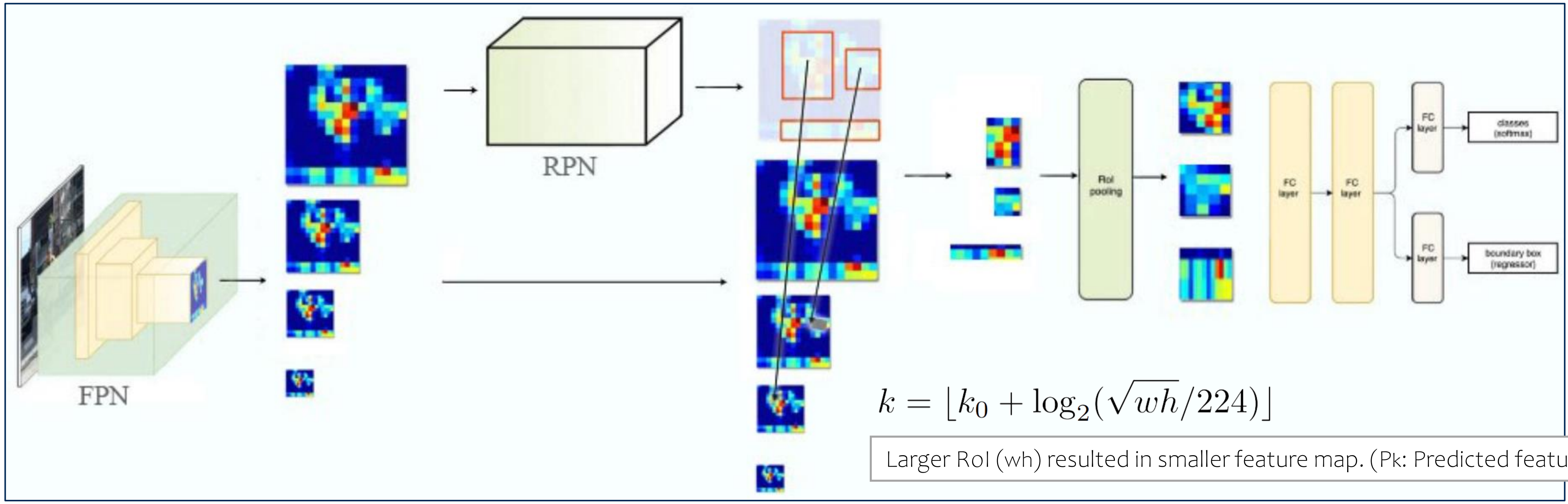


Feature Pyramid Networks | Faster R-CNN

Faster R-CNN



FPN-based Faster R-CNN

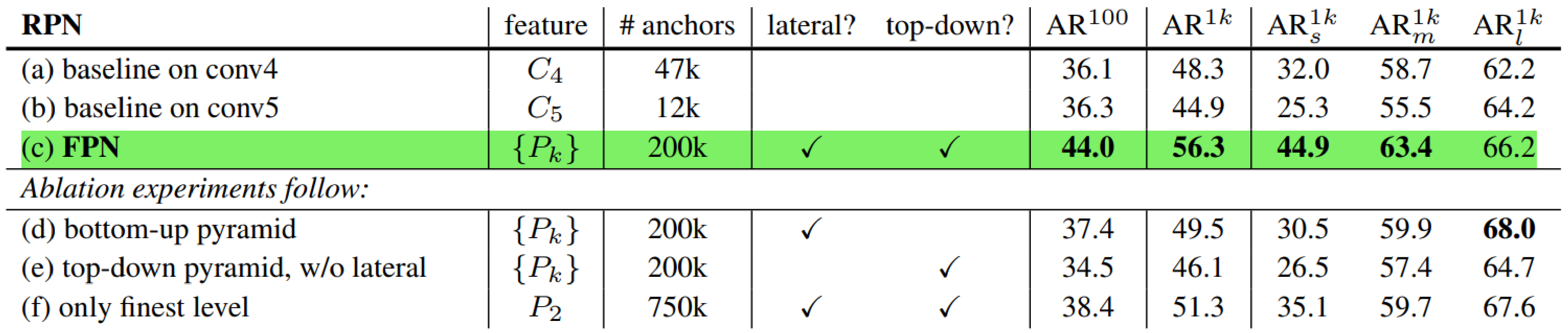


$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$$

Larger RoI (wh) resulted in smaller feature map. (Pk: Predicted feature map)

Reference: [Understanding Feature Pyramid Networks for object detection (FPN) | by Jonathan Hui | Medium]

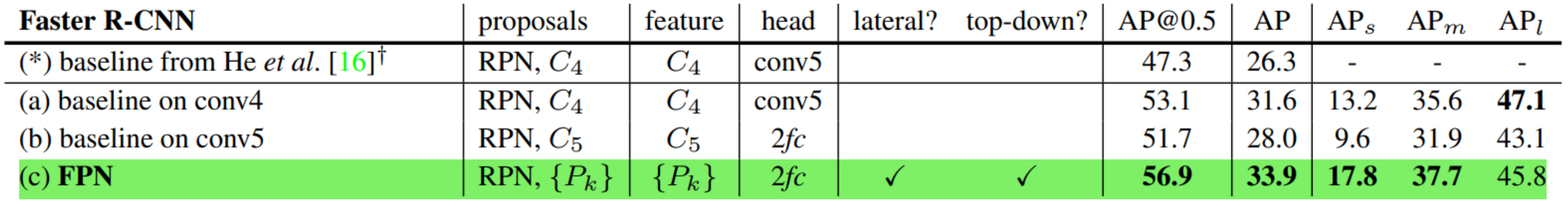
Benchmark



RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_{s}	AR^{1k}_{m}	AR^{1k}_{l}
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

FPN increases the #anchors generated in RPN even though scale variations are eliminated. (RPN anchor parameter *k* is less)

Benchmark



Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16][†]	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	*2fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

Inference Time on NVIDIA M40 GPU

0.148 sec	FPN-based Faster R-CNN (ResNet-50)
0.172 sec	Standard Faster R-CNN (ResNet-50)

FPN has extra layers but has a lighter weight head.

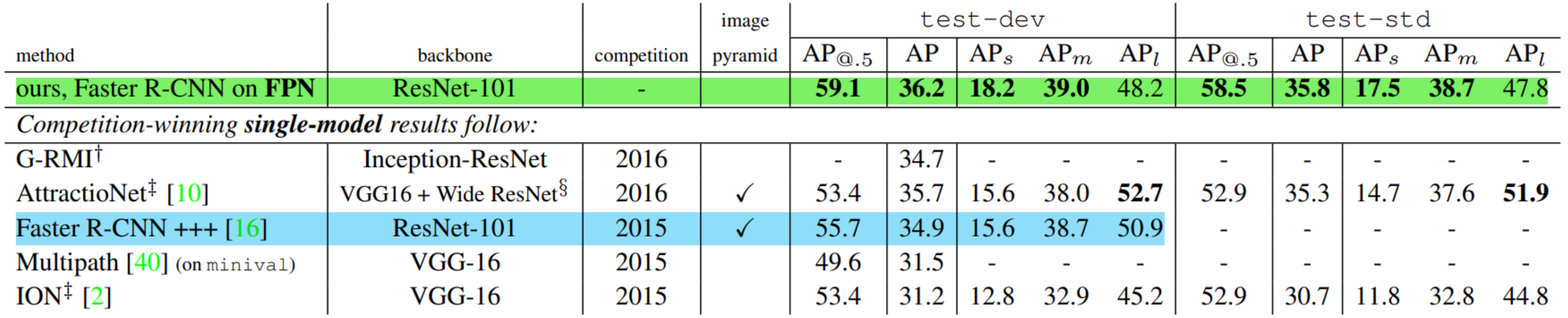


			image	test-dev					test-std				
method	backbone	competition	pyramid	$AP_{@.5}$	AP	AP_s	AP_m	AP_l	$AP_{@.5}$	AP	AP_s	AP_m	AP_l
ours, Faster R-CNN on **FPN**	ResNet-101	-		**59.1**	**36.2**	**18.2**	**39.0**	48.2	**58.5**	**35.8**	**17.5**	**38.7**	47.8
Competition-winning ***single-model*** *results follow:*													
G-RMI[†]	Inception-ResNet	2016		-	34.7	-	-	-	-	-	-	-	-
AttractioNet[‡] [10]	VGG16 + Wide ResNet[§]	2016	✓	53.4	35.7	15.6	38.0	**52.7**	52.9	35.3	14.7	37.6	**51.9**
Faster R-CNN +++ [16]	ResNet-101	2015	✓	55.7	34.9	15.6	38.7	50.9	-	-	-	-	-
Multipath [40] (on `minival`)	VGG-16	2015		49.6	31.5	-	-	-	-	-	-	-	-
ION[‡] [2]	VGG-16	2015		53.4	31.2	12.8	32.9	45.2	52.9	30.7	11.8	32.8	44.8

Thank you