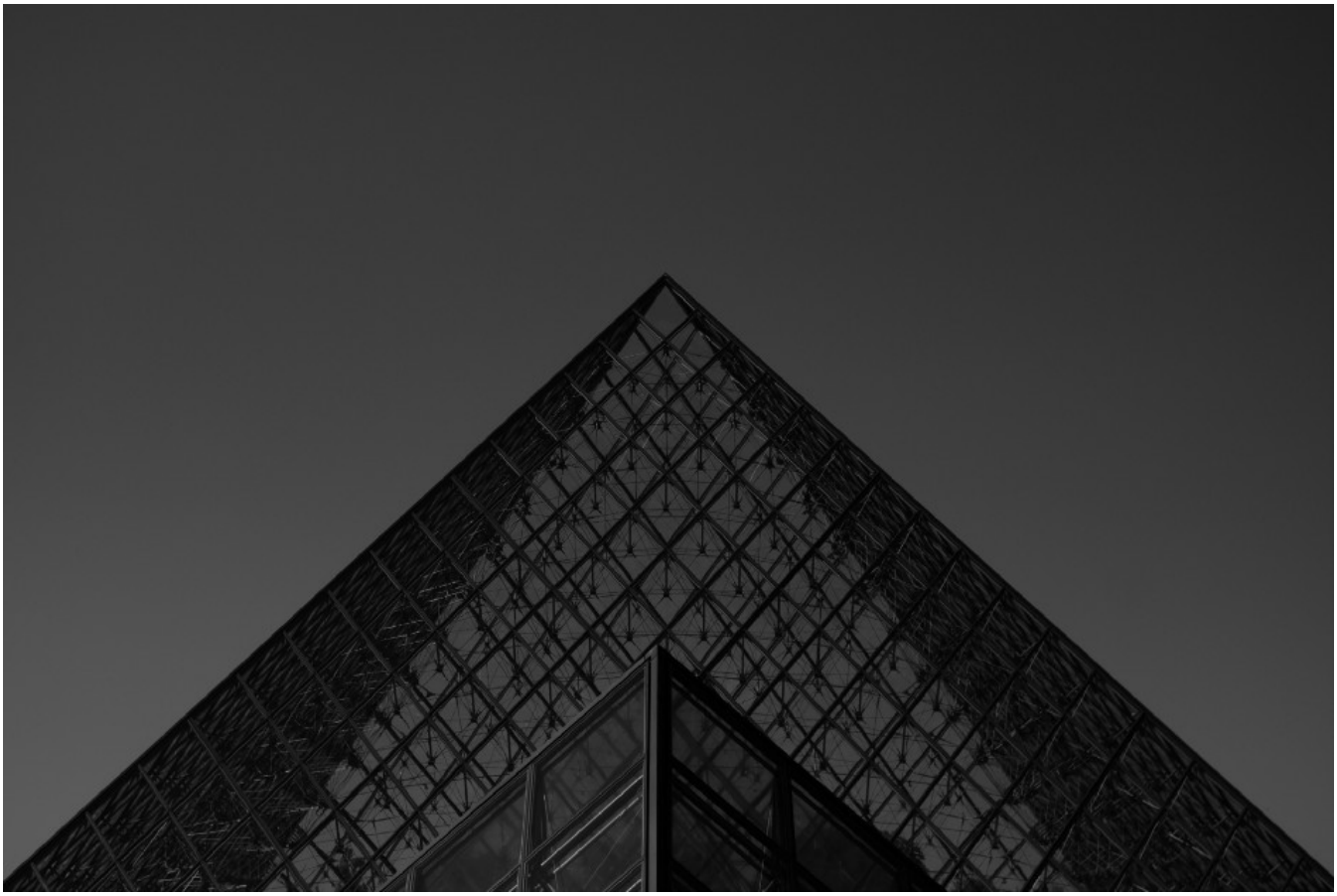
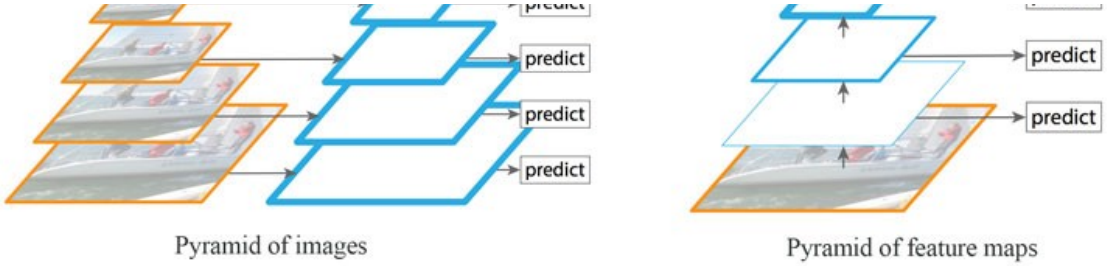


Understanding Feature Pyramid Networks for object detection (FPN)

Jonathan Hui
Mar 27, 2018 · 6 min read

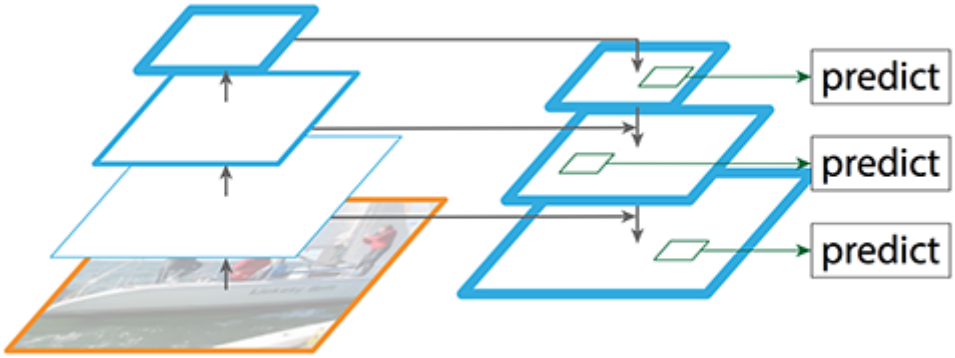
Detecting objects in different scales is challenging in particular for small objects. We can use a pyramid of the same image at different scale to detect objects (the left diagram below). However, processing multiple scale images is time consuming and the memory demand is too high to be trained end-to-end simultaneously. Hence, we may only use it in inference to push accuracy as high as possible, in particular for competitions, when speed is not a concern. Alternatively, we create a pyramid of feature and use them for object detection (the right diagram). However, feature maps closer to the image layer composed of low-level structures that are not effective for accurate object detection.



Source

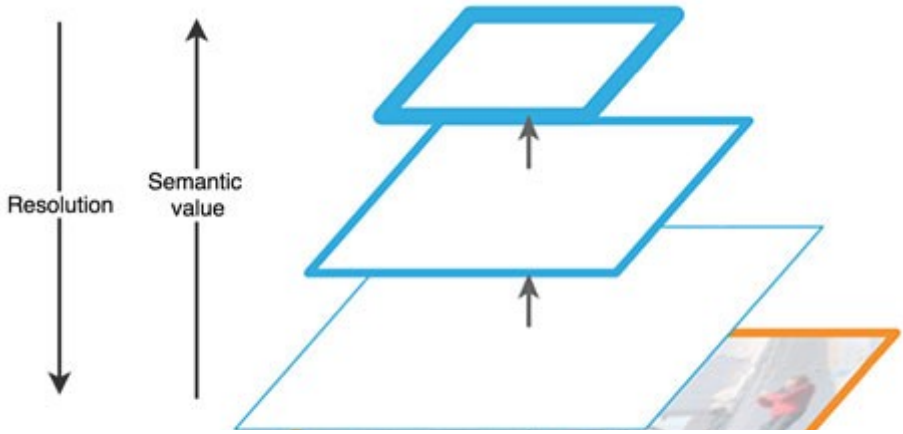
Feature Pyramid Network (**FPN**) is a feature extractor designed for such pyramid concept with accuracy and speed in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (**multi-scale feature maps**) with better quality information than the regular feature pyramid for object detection.

Data Flow



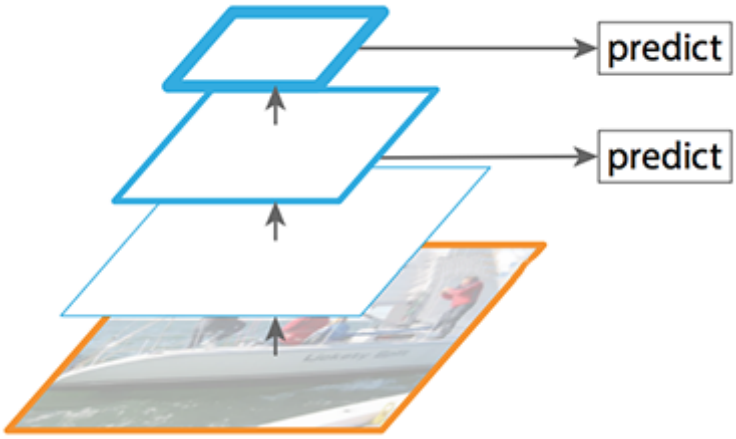
FPN (Source)

FPN composes of a **bottom-up** and a **top-down** pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the **semantic value** for each layer increases.



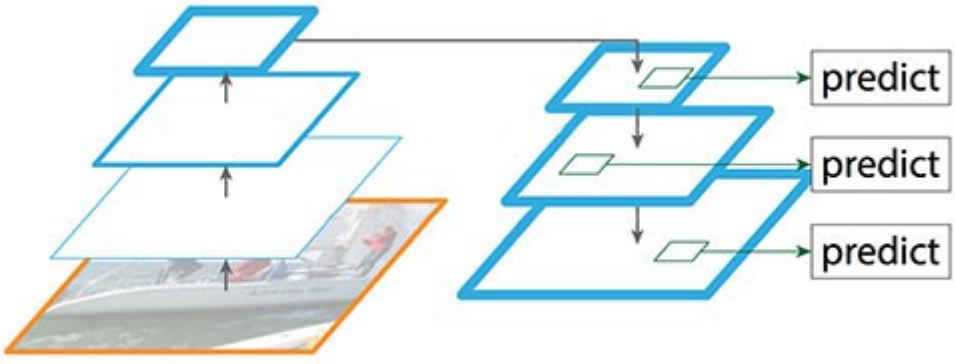
Feature extraction in FPN (Modified from source)

SSD makes detection from multiple feature maps. However, the bottom layers are not selected for object detection. They are in high resolution but the semantic value is not high enough to justify its use as the speed slow-down is significant. So SSD only uses upper layers for detection and therefore performs much worse for small objects.



Modified from source

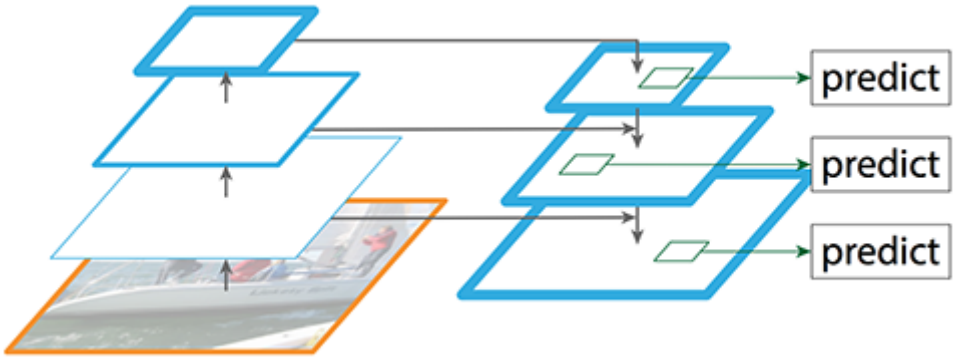
FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer.



Reconstruct spatial resolution in the top-down pathway. (Modified from source)

While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling. We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector

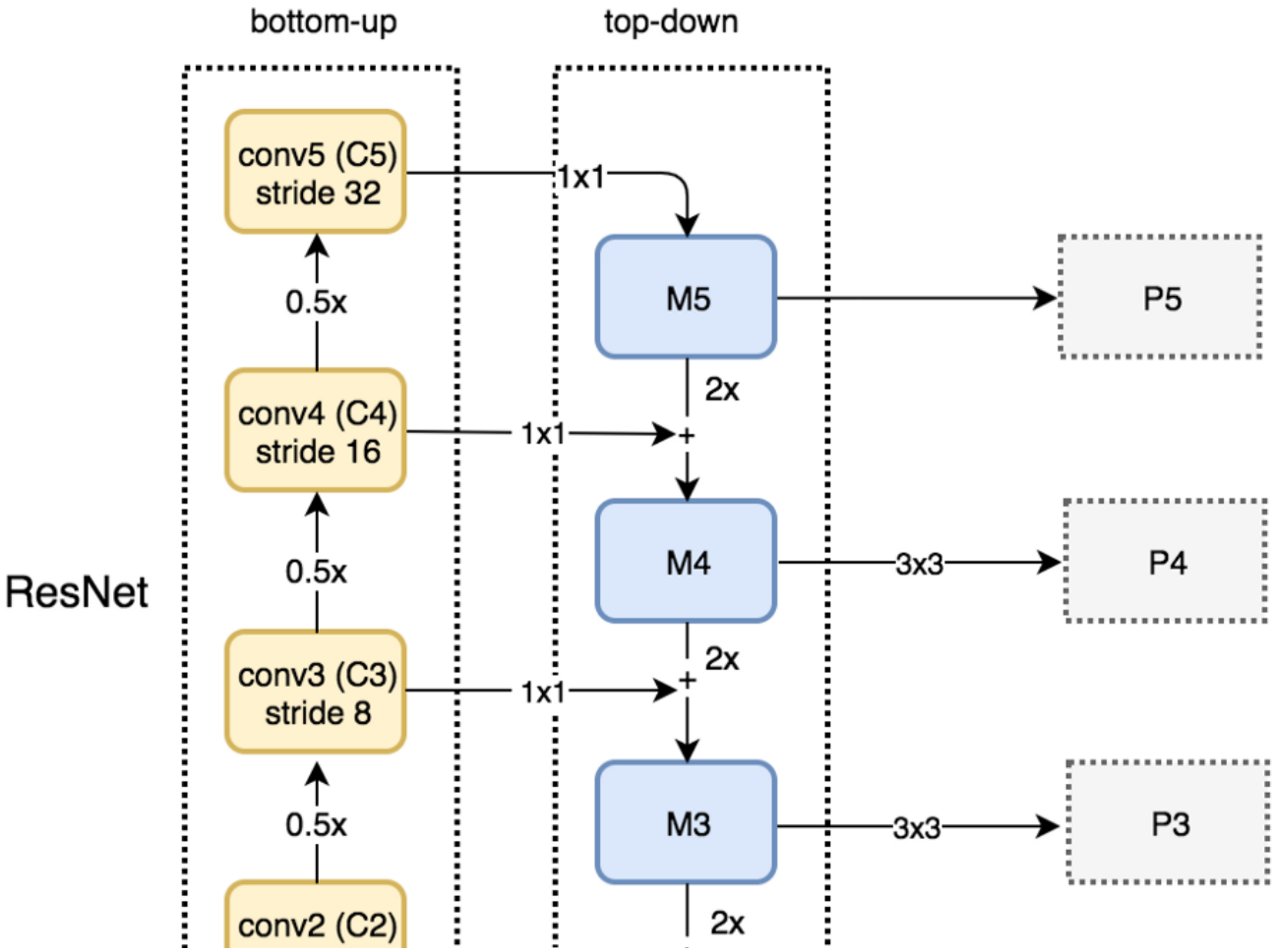
to predict the location betters. It also acts as skip connections to make training easier (similar to what ResNet does).

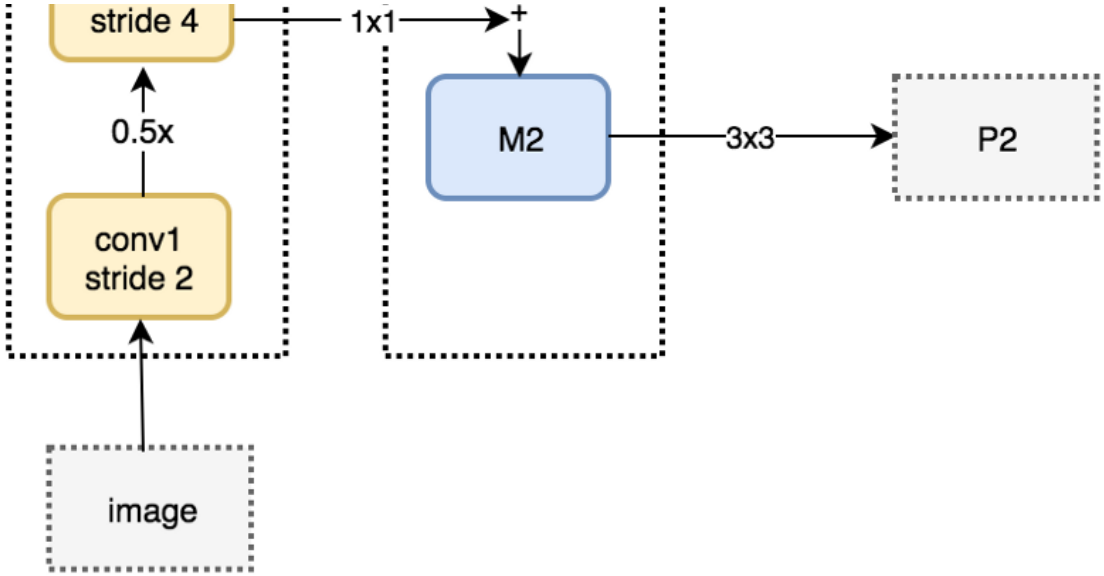


Add skip connections (Source)

Bottom-up pathway

The bottom-up pathway uses ResNet to construct the bottom-up pathway. It composes of many convolution modules (conv*i* for i equals 1 to 5) each has many convolution layers. As we move up, the spatial dimension is reduced by 1/2 (i.e. double the stride). The output of each convolution module is labeled as C*i* and later used in the top-down pathway.

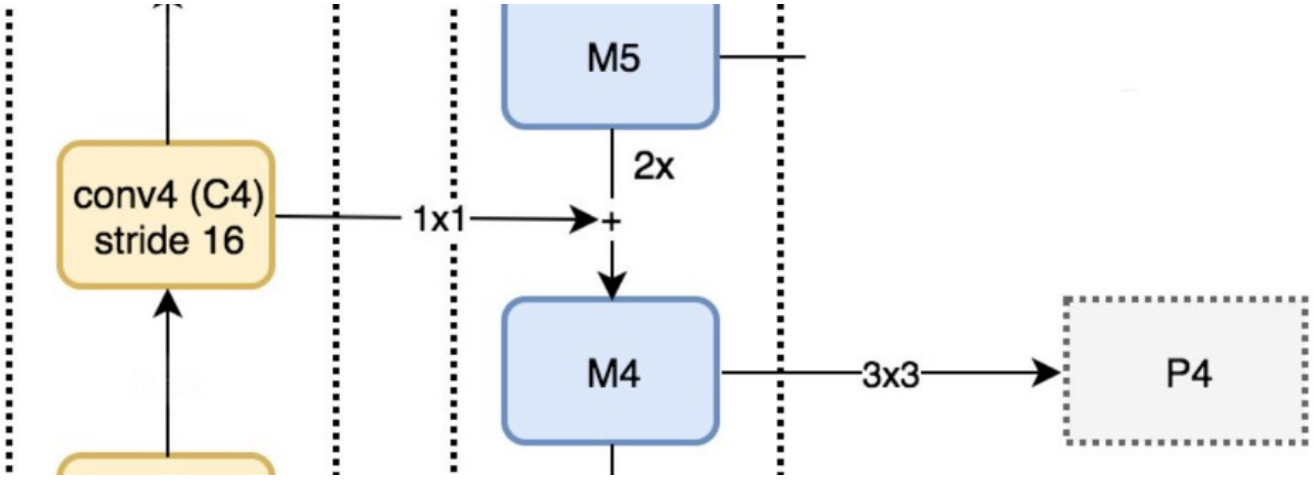




Top-down pathway

We apply a 1 × 1 convolution filter to reduce C5 channel depth to 256-d to create M5. This becomes the first feature map layer used for object prediction.

As we go down the top-down path, we upsample the previous layer by 2 using nearest neighbors upsampling. We again apply a 1 × 1 convolution to the corresponding feature maps in the bottom-up pathway. Then we add them element-wise. We apply a 3 × 3 convolution to all merged layers. This filter reduces the aliasing effect when merged with the upsampled layer.

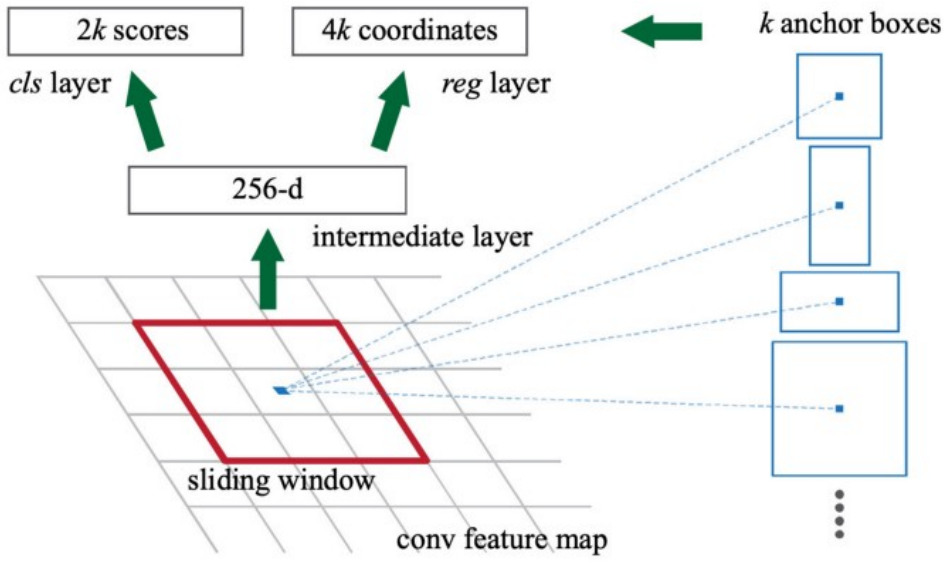


We repeat the same process for P3 and P2. However, we stop at P2 because the spatial dimension of C1 is too large. Otherwise, it will slow down the process too much. Because we share the same classifier and box regressor of every output feature maps, all pyramid feature maps (P5, P4, P3 and P2) have 256-d output channels.

FPN with RPN (Region Proposal Network)

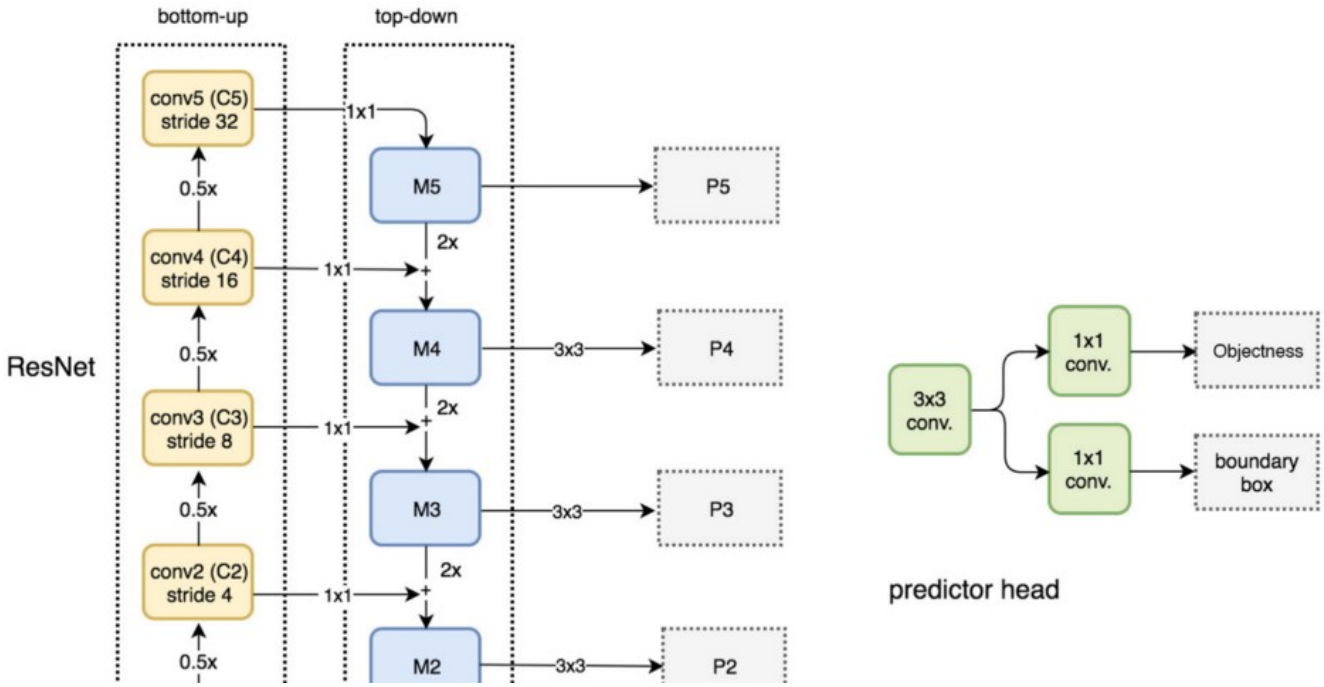
> *FPN is not an object detector by itself. It is a feature extractor that works with object detectors.*

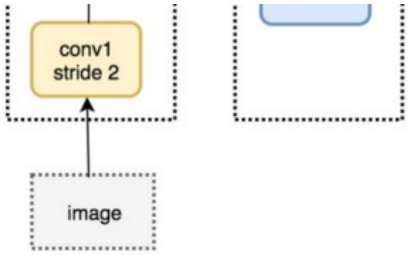
FPN extracts feature maps and later feeds into a detector, says RPN, for object detection. RPN applies a sliding window over the feature maps to make predictions on the objectness (has an object or not) and the object boundary box at each location.



Source

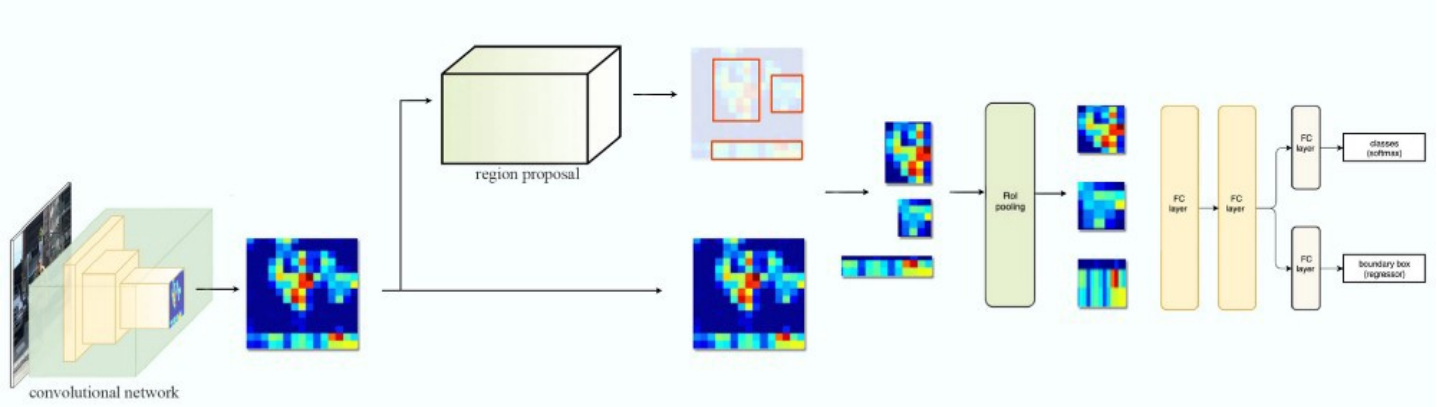
In the FPN framework, for each scale level (say P4), a 3×3 convolution filter is applied over the feature maps followed by separate 1×1 convolution for objectness predictions and boundary box regression. These 3×3 and 1×1 convolutional layers are called the RPN **head**. The same head is applied to all different scale levels of feature maps.



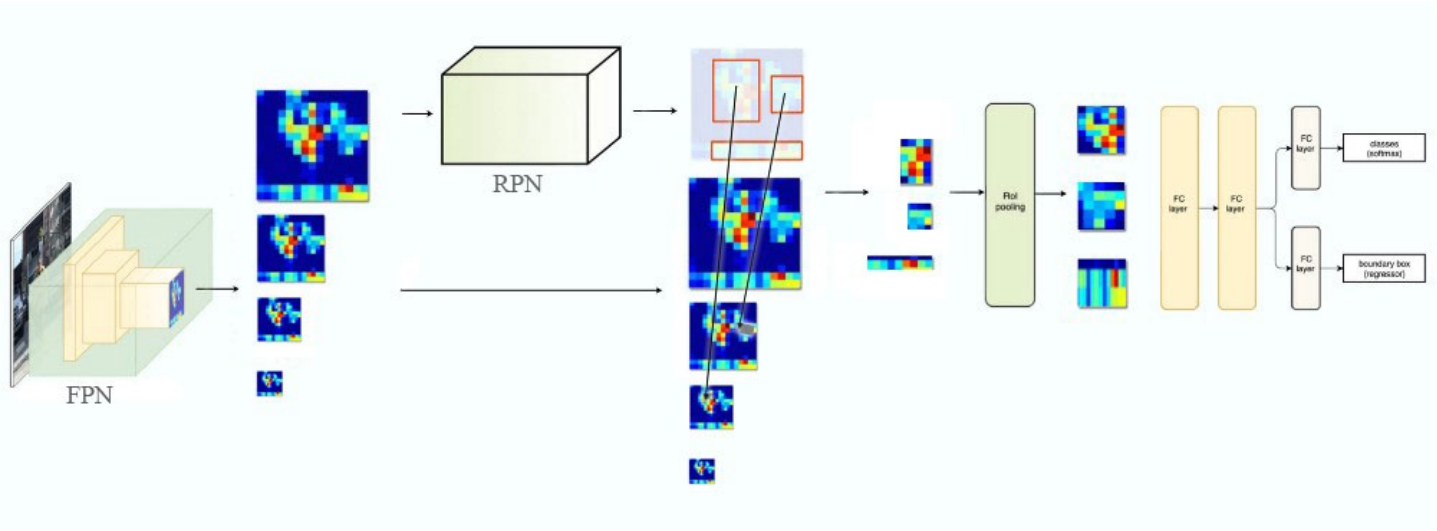


FPN with Fast R-CNN or Faster R-CNN

Let's take a quick look at the Fast R-CNN and Faster R-CNN data flow below. It works with one feature map layer to create ROIs. We use the ROIs and the feature map layer to create feature patches to be fed into the ROI pooling.



In FPN, we generate a pyramid of feature maps. We apply the RPN (described in the previous section) to generate ROIs. Based on the size of the ROI, we select the feature map layer in the most proper scale to extract the feature patches.



The formula to pick the feature maps is based on the width w and height h of the ROI.

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

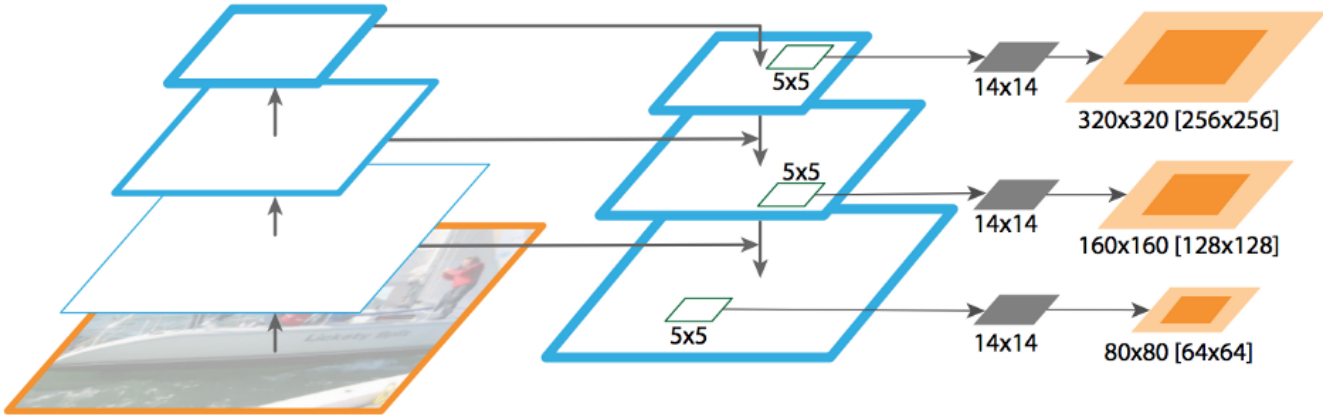
where

$k_0 = 4$
k is the P_k layer in the FPN used to generate the feature patch.

So if k = 3, we select P3 as our feature maps. We apply the ROI pooling and feed the result to the Fast R-CNN head (Fast R-CNN and Faster R-CNN have the same head) to finish the prediction.

Segmentation

Just like Mask R-CNN, FPN is also good at extracting masks for image segmentation. Using MLP, a 5×5 window is slide over the feature maps to generate an object segment of dimension 14×14 segments. Later, we merge masks at a different scale to form our final mask predictions.



Source

Results

Placing FPN in RPN improves AR (average recall: the ability to capture objects) to 56.3, an 8.0 points improvement over the RPN baseline. The performance on small objects is increased by a large margin of 12.9 points.

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

Source

The FPN based Faster R-CNN achieves an inference time of 0.148 second per image on a single NVIDIA M40 GPU for ResNet-50 while the single-scale ResNet-50 baseline runs at 0.32 seconds. Here is the baseline comparison for FPN using Faster R-CNN. (FPN introduces a small extra cost for those extra layers in the FPN, but has a lighter weight head in the FPN implementation.)

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	2*fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

Source

FPN is very competitive with state-of-the-art detectors. In fact, it beats the winners for the COCO 2016 and 2015 challenge.

			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

Source

Lessons learned

Here are some lessons learned from the experimental data.

- Adding more anchors on a single high-resolution feature map layer is not sufficient to improve accuracy.
- Top-down pathway restores resolution with rich semantic information.
- But we need lateral connections to add more precise object spatial information back.
- Top-down pathway plus lateral connections improve accuracy by 8 points on COCO dataset. For small objects, it improves 12.9 points.

Machine Learning Deep Learning Computer Vision Object Detection Artificial Intelligence

About Help Legal

Get the Medium app