# Feature Pyramid Networks for Object Detection

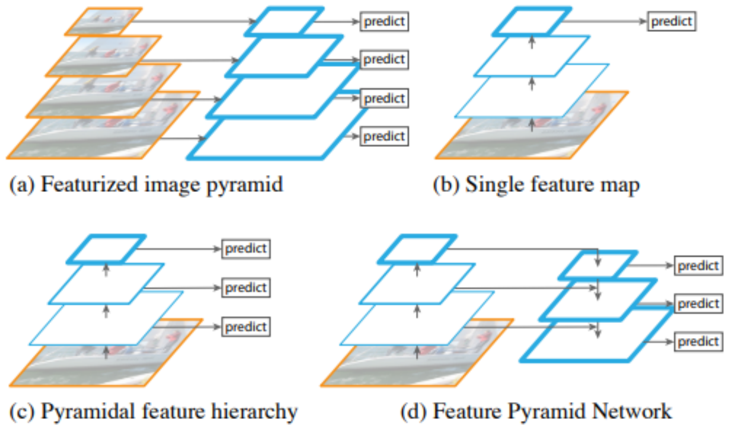
## Abstract

Feature pyramids are a basic component in recognition systems for detecting objects at different scales.
Recent deep learning object detectors have avoided pyramid representations, in part because they are compute and memory intensive.

In this paper, we exploit the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost.
A top-down architecture with lateral connections is developed for building high-level semantic feature maps at all scales.
This architecture, called a Feature Pyramid Network (FPN), shows significant improvement as a generic feature extractor in several applications.

In addition, our method can run at 6 FPS on a GPU and thus is a practical and accurate

## Introduction



Our model echoes a featurized image pyramid, which has not been explored in these works.
We evaluate our method, called a Feature Pyramid Network (FPN), in various systems for detection and segmentation [11, 29, 27].

Our method is also easily extended to mask proposals and improves both instance segmentation AR and speed over state-of-the-art methods that heavily depend on image pyramids.
In addition, our pyramid structure can be trained end-to-end with all scales and is used consistently at train/test time, which would be memory-infeasible using image pyramids.
Moreover, this improvement is achieved without increasing testing time over the single-scale baseline.

## Related Work

**Hand-engineered features and early neural networks**
These HOG and SIFT pyramids have been used in numerous works for image classification, object detection, human pose estimation, and more.
Dollar´ et al. [6] demonstrated fast pyramid computation by first computing a sparsely sampled (in scale) pyramid and then interpolating missing levels.
Before HOG and SIFT, early work on face detection with ConvNets [38, 32] computed shallow networks over image pyramids to detect faces across scales.

**Deep ConvNet object detectors.**
Recent and more accurate detection methods like Fast R-CNN [11] and Faster R-CNN [29] advocate using features computed from a single scale, because it offers a good trade-off between accuracy and speed.
Multi-scale detection, however, still performs better, especially for small objects.

**Methods using multiple layers.**

Although these methods adopt architectures with pyramidal shapes, they are unlike featurized image pyramids [5, 7, 34] where predictions are made independently at all levels, see Fig. 2.
In fact, for the pyramidal architecture in Fig. 2 (top), image pyramids are still needed to recognize objects across multiple scales [28].

## Feature Pyramid Networks

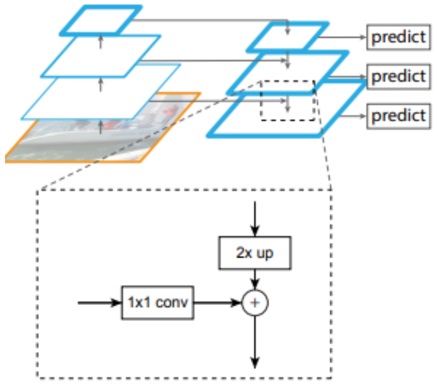
Our goal is to leverage a ConvNet's pyramidal feature hierarchy, which has semantics from low to high levels, and build a feature pyramid with high-level semantics throughout.

Our method takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion.
This process is independent of the backbone convolutional architectures (e.g., [19, 36, 16]), and in this paper we present results using ResNets [16].
The construction of our pyramid involves a bottom-up pathway, a top-down pathway, and lateral connections, as introduced in the following.

**Bottom-up pathway**

The bottom-up pathway is the feedforward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2.
We choose the output of the last layer of each stage as our reference set of feature maps, which we will enrich to create our pyramid.
Specifically, for ResNets [16] we use the feature activations output by each stage's last residual block.
We denote the output of these last residual blocks as {C2, C3, C4, C5} for conv2, conv3, conv4, and conv5 outputs, and note that they have strides of {4, 8, 16, 32} pixels with respect to the input image.
We do not include conv1 into the pyramid due to its large memory footprint.

**Top-down pathway and lateral connections**



The topdown pathway hallucinates higher resolution features by upsampling spatially coarser, but semantically stronger, feature maps from higher pyramid levels.
These features are then enhanced with features from the bottom-up pathway via lateral connections.

With a coarser-resolution feature map, we upsample the spatial resolution by a factor of 2 (using nearest neighbor upsampling for simplicity).
The upsampled map is then merged with the corresponding bottom-up map (which undergoes a 1×1 convolutional layer to reduce channel dimensions) by element-wise addition.
This process is iterated until the finest resolution map is generated.

There are no non-linearities in these extra layers, which we have empirically found to have minor impacts.

Simplicity is central to our design and we have found that our model is robust to many design choices.

Applications

In the following we adopt our method in RPN [29] for bounding box proposal generation and in Fast R-CNN [11] for object detection.

### Feature Pyramid Networks for RPN

We adapt RPN by replacing the single-scale feature map with our FPN.
We assign training labels to the anchors based on their Intersection-over-Union (IoU) ratios with ground-truth bounding boxes as in [29].
An anchor is assigned a positive label if it has the highest IoU for a given groundtruth box or an IoU over 0.7 with any ground-truth box, and a negative label if it has IoU lower than 0.3 for all ground-truth boxes.
The parameters of the heads are shared across all feature pyramid levels; we have also evaluated the alternative without sharing parameters and observed similar accuracy.

### Feature Pyramid Networks for Fast R-CNN

To use it with our FPN, we need to assign RoIs of different scales to the pyramid levels.
Formally, we assign an RoI of width w and height h (on the input image to the network) to the level Pk of our feature pyramid by:

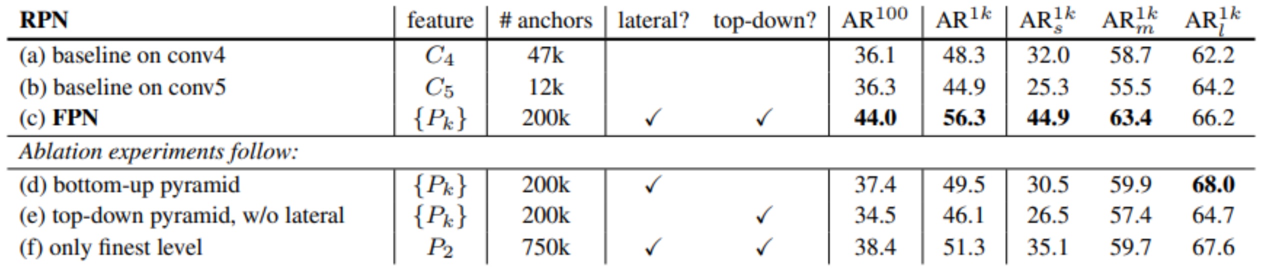
$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor .$$

We attach predictor heads (in Fast R-CNN the heads are class-specific classifiers and bounding box regressors) to all RoIs of all levels.
Again, the heads all share parameters, regardless of their levels.

Experiments on Object Detection

### Region Proposal with RPN

**Implementation details.**



| **RPN** | feature | # anchors | lateral? | top-down? | $\mathrm{AR}^{100}$ | $\mathrm{AR}^{1k}$ | $\mathrm{AR}_s^{1k}$ | $\mathrm{AR}_m^{1k}$ | $\mathrm{AR}_l^{1k}$ |
|---|---|---|---|---|---|---|---|---|---|
| (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 |

### Ablation Experiments

**Comparisons with baselines**

a single higher-level feature map is not enough because there is a trade-off between coarser resolutions and stronger semantics.
Our pyramid representation greatly improves RPN’s robustness to object scale variation.

**How important is top-down enrichment?**

The results in Table 1(d) are just on par with the RPN baseline and lag far behind ours.
We conjecture that this is because there are large semantic gaps between different levels on the bottom-up pyramid (Fig. 1(b)), especially for very deep ResNets.

We have also evaluated a variant of Table 1(d) without sharing the parameters of the heads, but observed similarly degraded performance.
This issue cannot be simply remedied by level-specific heads.

**How important are lateral connections?**
This top-down pyramid has strong semantic features and fine resolutions.
But we argue that the locations of these features are not precise, because these maps have been downsampled and upsampled several times.
More precise locations of features can be directly passed from the finer levels of the bottom-up maps via the lateral connections to the top-down maps.

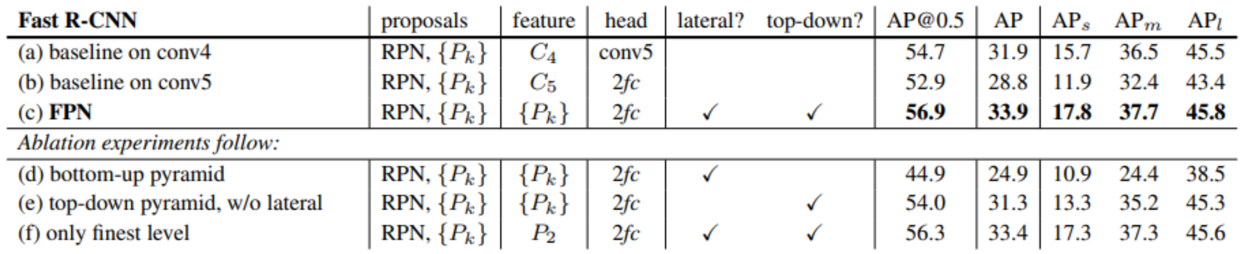
**How important are pyramid representations?**
Instead of resorting to pyramid representations, one can attach the head to the highest-resolution, strongly semantic feature maps of P2 (i.e., the finest level in our pyramids).
This result suggests that a larger number of anchors is not sufficient in itself to improve accuracy.

## Object Detection with Fast/Faster R-CNN

### Implementation details

The input image is resized such that its shorter side has 800 pixels

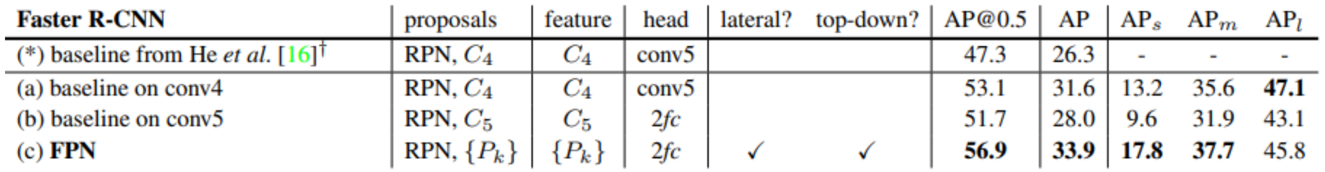
### Fast R-CNN (on fixed proposals)



| **Fast R-CNN** | proposals | feature | head | lateral? | top-down? | AP@0.5 | AP | AP$_s$ | AP$_m$ | AP$_l$ |
|---|---|---|---|---|---|---|---|---|---|---|
| (a) baseline on conv4 | RPN, $\{P_k\}$ | $C_4$ | conv5 | | | 54.7 | 31.9 | 15.7 | 36.5 | 45.5 |
| (b) baseline on conv5 | RPN, $\{P_k\}$ | $C_5$ | 2*fc* | | | 52.9 | 28.8 | 11.9 | 32.4 | 43.4 |
| (c) **FPN** | RPN, $\{P_k\}$ | $\{P_k\}$ | 2*fc* | ✓ | ✓ | **56.9** | **33.9** | **17.8** | **37.7** | **45.8** |
| *Ablation experiments follow:* | | | | | | | | | | |
| (d) bottom-up pyramid | RPN, $\{P_k\}$ | $\{P_k\}$ | 2*fc* | ✓ | | 44.9 | 24.9 | 10.9 | 24.4 | 38.5 |
| (e) top-down pyramid, w/o lateral | RPN, $\{P_k\}$ | $\{P_k\}$ | 2*fc* | | ✓ | 54.0 | 31.3 | 13.3 | 35.2 | 45.3 |
| (f) only finest level | RPN, $\{P_k\}$ | $P_2$ | 2*fc* | ✓ | ✓ | 56.3 | 33.4 | 17.3 | 37.3 | 45.6 |

We argue that this is because RoI pooling is a warping-like operation, which is less sensitive to the region's scales.
Despite the good accuracy of this variant, it is based on the RPN proposals of {Pk} and has thus already benefited from the pyramid representation.

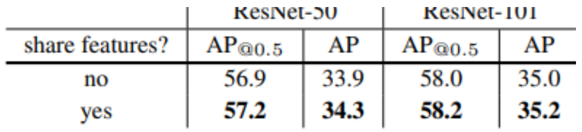
### Faster R-CNN (on consistent proposals)



| **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]$^\dagger$ | 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 |

**Sharing features.**

| | ResNet-50 | ResNet-101 |
|---|---|---|



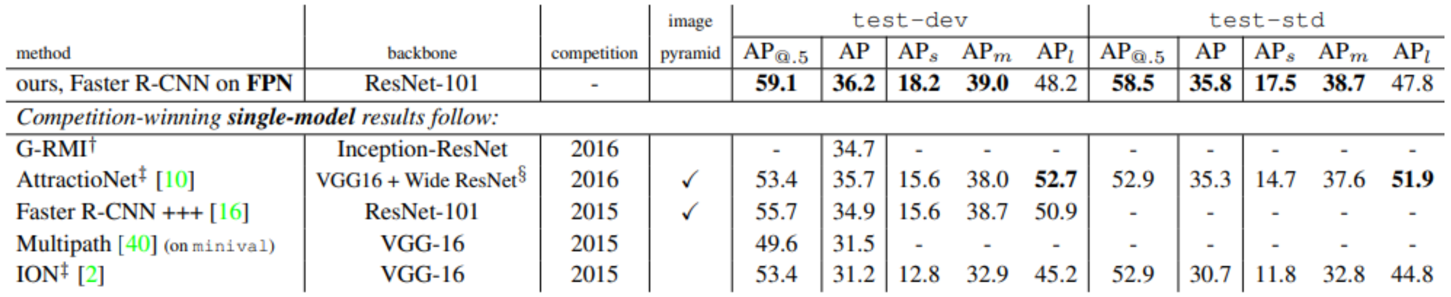
| | ResNet-50 | | ResNet-101 | |
|---|---|---|---|---|
| share features? | AP$_{@0.5}$ | AP | AP$_{@0.5}$ | AP |
| no | 56.9 | 33.9 | 58.0 | 35.0 |
| yes | **57.2** | **34.3** | **58.2** | **35.2** |

**Running time**

Overall our system is faster than the ResNet-based Faster R-CNN counterpart.

We believe the efficiency and simplicity of our method will benefit future research and applications.

### Comparing with COCO Competition Winners



| | | | 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 |

It is worth noting that our method does not rely on image pyramids and only uses a single input image scale, but still has outstanding AP on small-scale objects.

Moreover, our method does not exploit many popular improvements, such as iterative regression [9], hard negative mining [35], context modeling [16], stronger data augmentation [22], etc.

Extensions: Segmentation Proposals

Our method is a generic pyramid representation and can be used in applications other than object detection.

DeepMask/SharpMask were trained on image crops for predicting instance segments and object/non-object scores.

At inference time, these models are run convolutionally to generate dense proposals in an image.

To generate segments at multiple scales, image pyramids are necessary [27, 28].

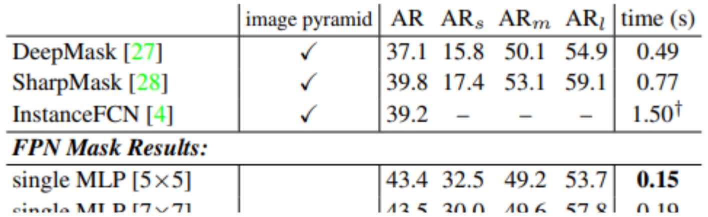
On top of each level of the feature pyramid, we apply a small 5×5 MLP to predict 14×14 masks and object scores in a fully convolutional fashion, see Fig. 4.

Additionally, motivated by the use of 2 scales per octave in the image pyramid of [27, 28], we use a second MLP of input size 7×7 to handle half octaves.

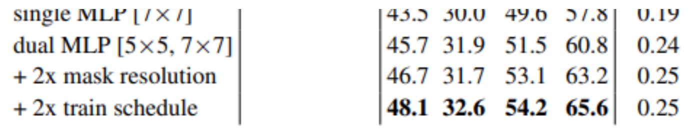
The two MLPs play a similar role as anchors in RPN.

The architecture is trained end-to-end; full implementation details are given in the appendix.

### Segmentation Proposal Results



| | image pyramid | AR | AR$_s$ | AR$_m$ | AR$_l$ | time (s) |
|---|---|---|---|---|---|---|
| DeepMask [27] | ✓ | 37.1 | 15.8 | 50.1 | 54.9 | 0.49 |
| SharpMask [28] | ✓ | 39.8 | 17.4 | 53.1 | 59.1 | 0.77 |
| InstanceFCN [4] | ✓ | 39.2 | – | – | – | 1.50[†] |
| ***FPN Mask Results:*** | | | | | | |
| single MLP [5×5] | | 43.4 | 32.5 | 49.2 | 53.7 | **0.15** |
| single MLP [7×7] | | 43.5 | 30.0 | 49.6 | 57.8 | 0.19 |



| | | | | | | |
|---|---|---|---|---|---|---|
| single MLP [7×7] | | 43.5 | 30.0 | 49.6 | 57.8 | 0.19 |
| dual MLP [5×5, 7×7] | | 45.7 | 31.9 | 51.5 | 60.8 | 0.24 |
| + 2x mask resolution | | 46.7 | 31.7 | 53.1 | 63.2 | 0.25 |
| + 2x train schedule | | **48.1** | **32.6** | **54.2** | **65.6** | 0.25 |

These results demonstrate that our model is a generic feature extractor and can replace image pyramids for other multi-scale detection problems.

Conclusion

Our method shows significant improvements over several strong baselines and competition winners.
despite the strong representational power of deep ConvNets and their implicit robustness to scale variation, it is still critical to explicitly address multiscale problems using pyramid representations.