Feature Pyramid Networks for Object Detection

Tsung-Yi Lin[1,2], Piotr Dollár[1], Ross Girshick[1],
Kaiming He[1], Bharath Hariharan[1], and Serge Belongie[2]

[1]Facebook AI Research (FAIR)
[2]Cornell University and Cornell Tech

Abstract

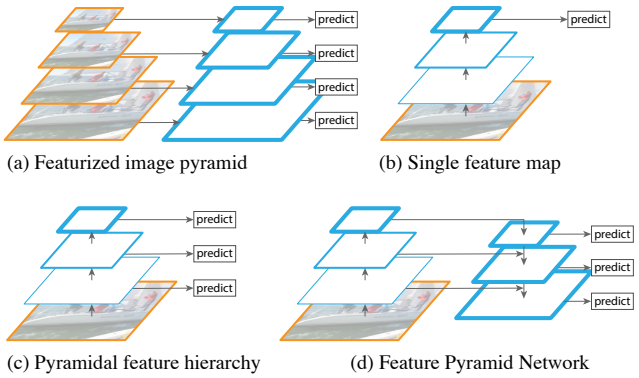
Feature pyramids are a basic component in recognition systems for detecting objects at different scales. But recent deep learning object detectors have avoided pyramid representations, in part because they are compute and memory intensive. In this paper, we exploit the inherent multi-scale, pyramidal 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. Using FPN in a basic Faster R-CNN system, our method achieves state-of-the-art single-model results on the COCO detection benchmark without bells and whistles, surpassing all existing single-model entries including those from the COCO 2016 challenge winners. In addition, our method can run at 6 FPS on a GPU and thus is a practical and accurate solution to multi-scale object detection. Code will be made publicly available.

1. Introduction

Recognizing objects at vastly different scales is a fundamental challenge in computer vision. *Feature pyramids built upon image pyramids* (for short we call these *featurized image pyramids*) form the basis of a standard solution [1] (Fig. 1(a)). These pyramids are scale-invariant in the sense that an object's scale change is offset by shifting its level in the pyramid. Intuitively, this property enables a model to detect objects across a large range of scales by scanning the model over both positions and pyramid levels.

Featurized image pyramids were heavily used in the era of hand-engineered features [5, 25]. They were so critical that object detectors like DPM [7] required dense scale sampling to achieve good results (*e.g.*, 10 scales per octave). For recognition tasks, engineered features have largely been replaced with features computed by deep convolutional networks (ConvNets) [19, 20]. Aside from being capable of representing higher-level semantics, ConvNets are also more robust to variance in scale and thus facilitate recognition from features computed on a single input scale [15, 11, 29] (Fig. 1(b)). But even with this robustness, pyramids are still needed to get the most accurate results. All recent top entries in the ImageNet [33] and COCO [21] detection challenges use multi-scale testing on featurized image pyramids (*e.g.*, [16, 35]). The principle advantage of featurizing each level of an image pyramid is that it produces a multi-scale feature representation in which *all levels are semantically strong*, including the high-resolution levels.

Nevertheless, featurizing each level of an image pyramid has obvious limitations. Inference time increases considerably (*e.g.*, by four times [11]), making this approach impractical for real applications. Moreover, training deep



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

用于目标检测的特征金字塔网络

林致远[1,2]、彼得·多拉尔[1]、罗斯·吉斯克[1]、何恺明[1]、巴拉特·哈里哈兰[1]与谢尔盖·贝洛吉[2]

1Facebook AI Research (FAIR) [2]
康奈尔大学和康奈尔理工学院

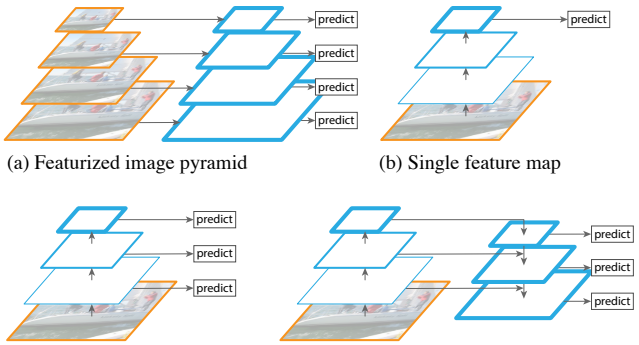
摘要

Feature pyramids are a basic component in recognition systems for detecting objects at different scales. But 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. Using FPN in a basic Faster R-CNN system, our method achieves state-of-the-art single-model results on the COCO detection benchmark without bells and whistles, surpassing all existing single-model entries including those from the COCO 2016 challenge winners. In addition, our method can run at 6 FPS on a GPU and thus is a practical and accurate solution to multi-scale object detection. Code will be made publicly available.

1. 引言

在计算机视觉中，识别尺度差异极大的物体是一个基础性挑战。*Feature pyramids built upon image pyramids* (为简便起见，我们称这些*featur- ized image pyramids*)构成了标准解决方案的基础[1]（图1(a)）。这些金字塔具有尺度不变性，即物体尺度的变化可以通过在金字塔中移动其所在层级来抵消。直观上，这一特性使模型能够通过在位置和金字塔层级上进行扫描，来检测大范围尺度内的物体。

特征化图像金字塔在手工程特征时代被大量使用[5, 25]。它们至关重要，以至于像DPM[7]这样的目标检测器需要密集的尺度采样才能获得良好结果（{v*}，每个八度10个尺度）。对于识别任务，工程特征已经很大程度上已被深度卷积网络（ConvNets）计算的特征所取代[19, 20]。除了能够表示更高层次的语义外，ConvNets对尺度变化也更为鲁棒，从而有助于在单一输入尺度上计算的特征进行识别[15, 11, 29]（图1(b)）。但即使具备这种鲁棒性，为了获得最准确的结果，仍然需要金字塔结构。近期ImageNet [33]和COCO [21]检测挑战中所有顶尖参赛方案都在特征化的图像金字塔上使用了多尺度测试（*e.g*，如[16, 35]）。对图像金字塔每一层级进行特征化的主要优势在于，它能生成一种多尺度特征表示，其中*all levels are semantically strong*，包括高分辨率层级。



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

图1. (a) 使用图像金字塔构建特征金字塔。特征在每个图像尺度上独立计算，速度较慢。(b) 近期的检测系统倾向于仅使用单尺度特征以加快检测速度。(c) 另一种方案是复用卷积网络计算出的金字塔特征层次，将其视为特征化的图像金字塔。(d) 我们提出的特征金字塔网络（FPN）兼具(b)和(c)的速度优势，且精度更高。图中蓝色轮廓表示特征图，轮廓越粗代表语义特征越强。

然而，对图像金字塔的每个层级进行特征化具有明显的局限性。推理时间显著增加（{v*}，例如增加四倍[11]），使得这种方法在实际应用中不切实际。此外，训练深度

networks end-to-end on an image pyramid is infeasible in terms of memory, and so, if exploited, image pyramids are used only at test time [15, 11, 16, 35], which creates an inconsistency between train/test-time inference. For these reasons, Fast and Faster R-CNN [11, 29] opt to not use featurized image pyramids under default settings.

However, image pyramids are not the only way to compute a multi-scale feature representation. A deep ConvNet computes a *feature hierarchy* layer by layer, and with subsampling layers the feature hierarchy has an inherent multi-scale, pyramidal shape. This in-network feature hierarchy produces feature maps of different spatial resolutions, but introduces large semantic gaps caused by different depths. The high-resolution maps have low-level features that harm their representational capacity for object recognition.

The Single Shot Detector (SSD) [22] is one of the first attempts at using a ConvNet's pyramidal feature hierarchy as if it were a featurized image pyramid (Fig. 1(c)). Ideally, the SSD-style pyramid would reuse the multi-scale feature maps from different layers computed in the forward pass and thus come free of cost. But to avoid using low-level features SSD foregoes reusing already computed layers and instead builds the pyramid starting from high up in the network (*e.g.*, conv4_3 of VGG nets [36]) and then by adding several new layers. Thus it misses the opportunity to reuse the higher-resolution maps of the feature hierarchy. We show that these are important for detecting small objects.

The goal of this paper is to naturally leverage the pyramidal shape of a ConvNet's feature hierarchy while creating a feature pyramid that has strong semantics at all scales. To achieve this goal, we rely on an architecture that combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections (Fig. 1(d)). The result is a feature pyramid that has rich semantics at all levels and is built quickly from a single input image scale. In other words, we show how to create in-network feature pyramids that can be used to replace featurized image pyramids without sacrificing representational power, speed, or memory.

Similar architectures adopting top-down and skip connections are popular in recent research [28, 17, 8, 26]. Their goals are to produce a single high-level feature map of a fine resolution on which the predictions are to be made (Fig. 2 top). On the contrary, our method leverages the architecture as a feature pyramid where predictions (*e.g.*, object detections) are independently made on each level (Fig. 2 bottom). 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]. Without bells and whistles, we report a state-of-the-art single-model result on the challenging COCO detection benchmark [21] simply based on FPN and a basic Faster R-CNN detector [29], surpassing all existing heavily-engineered single-model entries of competition winners. In ablation experiments, we find that for bounding box proposals, FPN significantly increases the Average Recall (AR) by 8.0 points; for object detection, it improves the COCO-style Average Precision (AP) by 2.3 points and PASCAL-style AP by 3.8 points, over a strong single-scale baseline of Faster R-CNN on ResNets [16]. 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.

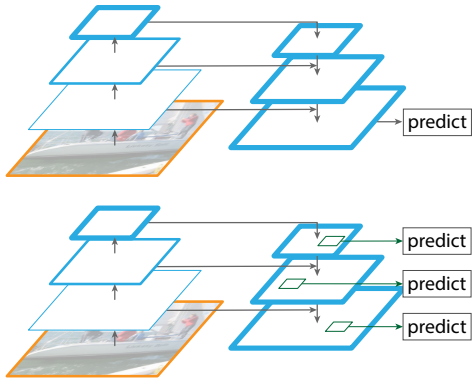


Figure 2. Top: a top-down architecture with skip connections, where predictions are made on the finest level (*e.g.*, [28]). Bottom: our model that has a similar structure but leverages it as a *feature pyramid*, with predictions made independently at all levels.

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. As a result, FPNs are able to achieve higher accuracy than all existing state-of-the-art methods. Moreover, this improvement is achieved without increasing testing time over the single-scale baseline. We believe these advances will facilitate future research and applications. Our code will be made publicly available.

2. Related Work

Hand-engineered features and early neural networks. SIFT features [25] were originally extracted at scale-space extrema and used for feature point matching. HOG features [5], and later SIFT features as well, were computed densely over entire image pyramids. These HOG and SIFT pyramids have been used in numerous works for image classification, object detection, human pose estimation, and more. There has also been significant interest in computing featurized image pyramids quickly. Dollár *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.

在图像金字塔上端到端地训练网络在内存方面是不可行的，因此，即使利用图像金字塔，也仅在测试时使用[15, 11, 16, 35]，这导致了训练与测试时推断的不一致。出于这些原因，Fast和Faster R-CNN[11, 29]在默认设置下选择不使用特征化的图像金字塔。

然而，图像金字塔并非计算多尺度特征表示的唯一方式。深度卷积网络通过逐层计算生成*feature hierarchy*，并借助下采样层使特征层次结构天然具备多尺度金字塔形态。这种网络内部的特征层次结构会生成不同空间分辨率的特征图，但同时也因深度差异造成了显著的语义鸿沟。高分辨率特征图包含的低层级特征会削弱其在目标识别任务中的表征能力。

单次检测器（SSD）[22] 是首次尝试将卷积神经网络的金字塔特征层级结构当作特征化图像金字塔使用的范例之一（图1(c)）。理想情况下，SSD风格的金字塔会复用前向传播中计算出的不同层多尺度特征图，从而实现零成本构建。但为了避免使用低层特征，SSD放弃了复用已计算层，而是从网络较高层开始构建金字塔（例如VGG网络[36]的{v*} conv4_3层），随后添加若干新层。这导致其错失了复用特征层级中高分辨率特征图的机会。我们证明这些特征图对于检测小目标至关重要。

本文的目标是自然地利用卷积网络特征层次的金字塔形状，同时在所有尺度上创建具有强语义的特征金字塔。为实现这一目标，我们依赖一种架构，该架构通过自上而下的路径和横向连接（图1(d)），将低分辨率、语义强的特征与高分辨率、语义弱的特征相结合。其结果是一个在所有层级都具有丰富语义的特征金字塔，并且可以从单一输入图像尺度快速构建。换句话说，我们展示了如何创建网络内特征金字塔，用以替代特征化的图像金字塔，而无需牺牲表征能力、速度或内存。

采用自上而下和跳跃连接的类似架构在近期研究中颇为流行[28, 17, 8, 26]。这些方法旨在生成单一的高分辨率高级特征图，并基于此进行预测（图2顶部）。相反，我们的方法将该架构作为特征金字塔使用，在每个层级上独立进行预测（*e.g.*，如目标检测）（图2底部）。我们的模型呼应了特征化的图像金字塔结构，而这一点在现有研究中尚未得到充分探索。

我们评估了名为特征金字塔网络（FPN）的方法，在多种检测与分割系统中进行了测试[11, 29, 27]。无需复杂技巧，仅基于FPN与{v*}，我们就在极具挑战性的COCO检测基准[21]上报告了当前最先进的单模型结果。

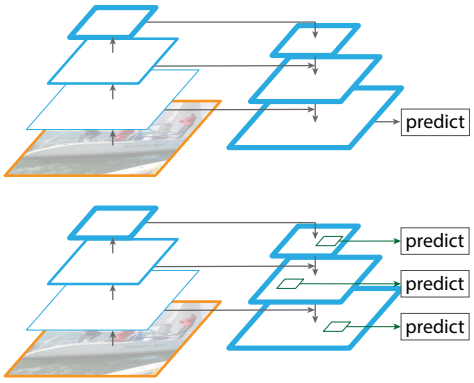


图2。顶部：一种带有跳跃连接的自顶向下架构，在最精细级别（*e.g.*，[28]）进行预测。底部：我们的模型具有类似结构，但将其用作*featurepyramid*，并在所有层级独立进行预测。

一个基础的Faster R-CNN检测器[29]，超越了所有现有的、经过大量工程优化的竞赛获胜者单模型方案。在消融实验中，我们发现对于边界框提议，FPN将平均召回率(AR)显著提升了8.0个点；对于目标检测，相较于基于ResNet[16]的强单尺度Faster R-CNN基线，它将COCO风格平均精度(AP)提升了2.3个点，PASCAL风格AP提升了3.8个点。我们的方法也能轻松扩展到掩码提议，并在实例分割AR和速度上超越了严重依赖图像金字塔的现有最优方法。

此外，我们的金字塔结构能够以端到端的方式在所有尺度上进行训练，并在训练/测试时保持一致使用，而使用图像金字塔则在内存上不可行。因此，FPN能够比所有现有的最先进方法达到更高的准确度。此外，这一改进是在不增加单尺度基线测试时间的情况下实现的。我们相信这些进展将促进未来的研究和应用。我们的代码将公开提供。

2. 相关工作

手工设计的特征与早期神经网络。SIFT特征[25]最初在尺度空间极值点处提取，并用于特征点匹配。HOG特征[5]以及后来的SIFT特征，均在完整的图像金字塔上密集计算。这些HOG和SIFT金字塔已被广泛应用于图像分类、物体检测、人体姿态估计等众多领域。快速计算特征化图像金字塔也一直备受关注。Dollár *et al*等人[6]通过先计算稀疏采样（在尺度上）的金字塔，再插值缺失的层级，展示了快速金字塔计算方法。在HOG和SIFT之前，早期基于卷积神经网络的人脸检测研究[38, 32]通过在图像金字塔上运行浅层网络，实现了跨尺度的人脸检测。

Deep ConvNet object detectors. With the development of modern deep ConvNets [19], object detectors like OverFeat [34] and R-CNN [12] showed dramatic improvements in accuracy. OverFeat adopted a strategy similar to early neural network face detectors by applying a ConvNet as a sliding window detector on an image pyramid. R-CNN adopted a region proposal-based strategy [37] in which each proposal was scale-normalized before classifying with a ConvNet. SPPnet [15] demonstrated that such region-based detectors could be applied much more efficiently on feature maps extracted on a single image scale. 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. A number of recent approaches improve detection and segmentation by using different layers in a ConvNet. FCN [24] sums partial scores for each category over multiple scales to compute semantic segmentations. Hypercolumns [13] uses a similar method for object instance segmentation. Several other approaches (HyperNet [18], ParseNet [23], and ION [2]) concatenate features of multiple layers before computing predictions, which is equivalent to summing transformed features. SSD [22] and MS-CNN [3] predict objects at multiple layers of the feature hierarchy without combining features or scores.

There are recent methods exploiting lateral/skip connections that associate low-level feature maps across resolutions and semantic levels, including U-Net [31] and SharpMask [28] for segmentation, Recombinator networks [17] for face detection, and Stacked Hourglass networks [26] for keypoint estimation. Ghiasi *et al*. [8] present a Laplacian pyramid presentation for FCNs to progressively refine segmentation. 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].

3. 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. The resulting *Feature Pyramid Network* is general-purpose and in this paper we focus on sliding window proposers (Region Proposal Network, RPN for short) [29] and region-based detectors (Fast R-CNN) [11]. We also generalize FPNs to instance segmentation proposals in Sec. 6.

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.

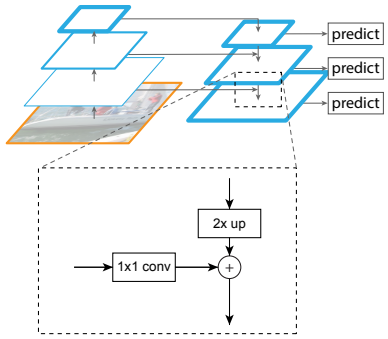


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

Bottom-up pathway. The bottom-up pathway is the feed-forward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2. There are often many layers producing output maps of the same size and we say these layers are in the same network *stage*. For our feature pyramid, we define one pyramid level for each stage. 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. This choice is natural since the deepest layer of each stage should have the strongest features.

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 $\{C_2, C_3, C_4, C_5\}$ 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 top-down 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. Each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway. The bottom-up feature map is of lower-level semantics, but its activations are more accurately localized as it was subsampled fewer times.

Fig. 3 shows the building block that constructs our top-down feature maps. With a coarser-resolution feature map, we upsample the spatial resolution by a factor of 2 (using nearest neighbor upsampling for simplicity). The upsam-

深度卷积网络目标检测器。随着现代深度卷积网络的发展[19]，诸如OverFeat[34]和R-CNN[12]等目标检测器在精度上展现出显著提升。OverFeat采用了与早期神经网络人脸检测器类似的策略，将卷积网络作为滑动窗口检测器应用于图像金字塔。R-CNN则采用基于区域建议的策略[37]，在通过卷积网络分类前对每个建议区域进行尺度归一化。SPPnet[15]证明了此类基于区域的检测器能够在单图像尺度提取的特征图上更高效地运行。近期更精确的检测方法如Fast R-CNN[11]和Faster R-CNN[29]主张使用单尺度计算的特征，因其在精度与速度间实现了良好平衡。然而多尺度检测仍表现更优，尤其对于小目标检测任务。

使用多层的方法。最近的一些方法通过使用卷积神经网络中的不同层来改进检测和分割。FCN [24] 在多个尺度上对每个类别的部分分数求和以计算语义分割。Hypercolumns [13] 使用类似的方法进行对象实例分割。其他几种方法（HyperNet [18]、ParseNet [23] 和 ION [2]）在计算预测之前连接多个层的特征，这相当于对变换后的特征求和。SSD [22] 和 MS-CNN [3] 在特征层次结构的多个层上预测对象，而不结合特征或分数。

最近有一些方法利用横向/跳跃连接，将低层次特征图在不同分辨率和语义级别之间关联起来，包括用于分割的U-Net [31]和SharpMask [28]、用于人脸检测的Recombinator网络 [17]，以及用于关键点估计的Stacked Hourglass网络 [26]。Ghiasi {v*}等人[8]提出了一种用于全卷积网络的拉普拉斯金字塔表示法，以逐步优化分割结果。尽管这些方法采用了金字塔形架构，但它们不同于特征化图像金字塔[5, 7, 34]——后者在所有层级上独立进行预测（见图2）。实际上，对于图2（顶部）的金字塔架构，仍然需要图像金字塔来识别多尺度物体[28]。

3. 特征金字塔网络

我们的目标在于利用卷积网络的金字塔特征层级结构（其语义信息由低层向高层递增），并构建一个具有高层语义贯穿始终的特征金字塔。由此得到的*Feature Pyramid Network*具有通用性，本文重点研究滑动窗口提议生成器（简称区域提议网络RPN）[29]以及基于区域的检测器（Fast R-CNN）[11]。我们还在第6节中将FPN推广至实例分割提议的生成。

我们的方法以任意尺寸的单尺度图像作为输入，并输出按比例缩放的{v*}特征图。

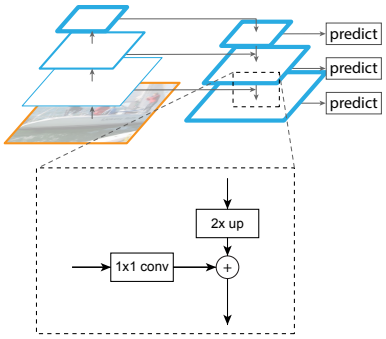


图3. 一个展示横向连接和自上而下通路的构建模块，通过加法合并。

在多个层次上，以全卷积的方式进行。这一过程独立于主干卷积架构（*e.g*，如[19, 36, 16]），本文中我们展示了使用ResNet[16]的结果。我们金字塔结构的构建包含自下而上的路径、自上而下的路径以及横向连接，具体如下所述。

自底向上路径。自底向上路径是骨干卷积网络的前馈计算，它计算出一个特征层次结构，包含多个尺度的特征图，缩放步长为2。通常有许多层产生相同尺寸的输出图，我们称这些层处于同一网络*stage*。对于我们的特征金字塔，我们为每个阶段定义一个金字塔层级。我们选择每个阶段最后一层的输出作为我们的参考特征图集，我们将丰富这些特征图以构建我们的金字塔。这种选择是自然的，因为每个阶段的最深层应具有最强的特征。

具体而言，对于ResNets[16]，我们采用每个阶段最后一个残差块输出的特征激活值。我们将这些最后残差块的输出记为$\{C_2, C_3, C_4, C_5\}$，分别对应conv2、conv3、conv4和conv5的输出，并注意到它们相对于输入图像的步长为{4、8、16、32}像素。由于conv1占用内存较大，我们未将其纳入金字塔结构中。

自上而下的路径与横向连接。自上而下的路径通过对更高金字塔层级中空间上更粗糙但语义更强的特征图进行上采样，来生成更高分辨率的特征。这些特征随后通过横向连接与自下而上路径的特征进行增强。每个横向连接会合并自下而上路径和自上而下路径中具有相同空间尺寸的特征图。自下而上的特征图具有较低层级的语义，但由于其被下采样的次数较少，其激活位置的定位更为准确。

图3展示了构建我们自上而下特征图的基本模块。我们通过一个更粗糙分辨率的特征图，将空间分辨率上采样2倍（为简便起见，使用最近邻上采样）。上采

pled 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. To start the iteration, we simply attach a 1×1 convolutional layer on C_5 to produce the coarsest resolution map. Finally, we append a 3×3 convolution on each merged map to generate the final feature map, which is to reduce the aliasing effect of upsampling. This final set of feature maps is called $\{P_2, P_3, P_4, P_5\}$, corresponding to $\{C_2, C_3, C_4, C_5\}$ that are respectively of the same spatial sizes.

Because all levels of the pyramid use shared classifiers/regressors as in a traditional featurized image pyramid, we fix the feature dimension (numbers of channels, denoted as d) in all the feature maps. We set $d = 256$ in this paper and thus all extra convolutional layers have 256-channel outputs. 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. We have experimented with more sophisticated blocks (*e.g.*, using multi-layer residual blocks [16] as the connections) and observed marginally better results. Designing better connection modules is not the focus of this paper, so we opt for the simple design described above.

4. Applications

Our method is a generic solution for building feature pyramids inside deep ConvNets. In the following we adopt our method in RPN [29] for bounding box proposal generation and in Fast R-CNN [11] for object detection. To demonstrate the simplicity and effectiveness of our method, we make minimal modifications to the original systems of [29, 11] when adapting them to our feature pyramid.

4.1. Feature Pyramid Networks for RPN

RPN [29] is a sliding-window class-agnostic object detector. In the original RPN design, a small subnetwork is evaluated on dense 3×3 sliding windows, on top of a single-scale convolutional feature map, performing object/non-object binary classification and bounding box regression. This is realized by a 3×3 convolutional layer followed by two sibling 1×1 convolutions for classification and regression, which we refer to as a network *head*. The object/non-object criterion and bounding box regression target are defined with respect to a set of reference boxes called *anchors* [29]. The anchors are of multiple pre-defined scales and aspect ratios in order to cover objects of different shapes.

We adapt RPN by replacing the single-scale feature map with our FPN. We attach a head of the same design (3×3 conv and two sibling 1×1 convs) to each level on our feature pyramid. Because the head slides densely over all locations in all pyramid levels, it is not necessary to have multi-scale anchors on a specific level. Instead, we assign anchors of a single scale to each level. Formally, we define the anchors to have areas of $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ pixels on $\{P_2, P_3, P_4, P_5, P_6\}$ respectively.[1] As in [29] we also use anchors of multiple aspect ratios $\{1{:}2,\ 1{:}1,\ 2{:}1\}$ at each level. So in total there are 15 anchors over the pyramid.

We assign training labels to the anchors based on their Intersection-over-Union (IoU) ratios with ground-truth bounding boxes as in [29]. Formally, an anchor is assigned a positive label if it has the highest IoU for a given ground-truth 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. Note that scales of ground-truth boxes are not explicitly used to assign them to the levels of the pyramid; instead, ground-truth boxes are associated with anchors, which have been assigned to pyramid levels. As such, we introduce no extra rules in addition to those in [29].

We note that 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. The good performance of sharing parameters indicates that all levels of our pyramid share similar semantic levels. This advantage is analogous to that of using a featurized image pyramid, where a common head classifier can be applied to features computed at any image scale.

With the above adaptations, RPN can be naturally trained and tested with our FPN, in the same fashion as in [29]. We elaborate on the implementation details in the experiments.

4.2. Feature Pyramid Networks for Fast R-CNN

Fast R-CNN [11] is a region-based object detector in which Region-of-Interest (RoI) pooling is used to extract features. Fast R-CNN is most commonly performed on a single-scale feature map. To use it with our FPN, we need to assign RoIs of different scales to the pyramid levels.

We view our feature pyramid as if it were produced from an image pyramid. Thus we can adapt the assignment strategy of region-based detectors [15, 11] in the case when they are run on image pyramids. Formally, we assign an RoI of width w and height h (on the input image to the network) to the level P_k of our feature pyramid by:

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor. \tag{1}$$

Here 224 is the canonical ImageNet pre-training size, and k_0 is the target level on which an RoI with $w \times h = 224^2$ should be mapped into. Analogous to the ResNet-based Faster R-CNN system [16] that uses C_4 as the single-scale feature map, we set k_0 to 4. Intuitively, Eqn. (1) means that if the RoI's scale becomes smaller (say, 1/2 of 224), it should be mapped into a finer-resolution level (say, $k = 3$).

[1] Here we introduce P_6 only for covering a larger anchor scale of 512^2. P_6 is simply a stride two subsampling of P_5. P_6 is not used by the Fast R-CNN detector in the next section.

然后，pled图与相应的自底向上图（经过1×1卷积层以减少通道维度）通过逐元素相加进行合并。这个过程不断迭代，直到生成最高分辨率的图。为了开始迭代，我们简单地在C_5上附加一个1×1卷积层以生成最粗糙分辨率的图。最后，我们在每个合并后的图上添加一个3×3卷积来生成最终的特征图，目的是减少上采样的混叠效应。这组最终的特征图被称为$\{P_2, P_3, P_4, P_5\}$，对应于$\{C_2, C_3, C_4, C_5\}$，它们分别具有相同的空间尺寸。

由于金字塔的所有层级都使用共享的分类器/回归器，就像传统的特征化图像金字塔一样，我们在所有特征图中固定了特征维度（通道数，记为d）。在本文中，我们设定d =为256，因此所有额外的卷积层都具有256通道的输出。这些额外层中没有非线性激活函数，我们通过实验发现这影响较小。

简洁性是我们设计的核心，我们发现模型对多种设计选择都表现出鲁棒性。我们尝试过更复杂的模块（例如，使用多层残差块[16]作为连接），并观察到略微更好的结果。设计更优的连接模块并非本文重点，因此我们选择了上述的简单设计。

4. 应用

我们的方法是一种在深度卷积网络内部构建特征金字塔的通用解决方案。接下来，我们将该方法应用于RPN[29]以生成边界框建议，并应用于Fast R-CNN[11]以进行目标检测。为展示本方法的简洁性与有效性，在将[29, 11]的原系统适配至我们的特征金字塔时，我们仅对其进行了最小程度的修改。

4.1. 用于RPN的特征金字塔网络

RPN [29] 是一种滑动窗口的类别无关目标检测器。在原始RPN设计中，一个小子网络在单尺度卷积特征图上密集的3×3滑动窗口上进行评估，执行目标/非目标的二分类和边界框回归。这是通过一个3×3卷积层，后接两个并行的1×1卷积层（分别用于分类和回归）来实现的，我们将其称为网络*head*。目标/非目标的判定准则和边界框回归目标都是相对于一组称为*anchors*[29]的参考框来定义的。这些锚框具有多种预定义的尺度和长宽比，以覆盖不同形状的目标。

我们通过将单尺度特征图替换为我们的FPN来改进RPN。我们在特征金字塔的每一层都附加了一个相同设计的头部（3×3卷积和两个并行的1×1卷积）。由于该头部在金字塔所有层级的所有位置上密集滑动，因此无需采用多尺度在特定层级上的锚点。相反，我们将单一尺度的锚点分配给每个层级。形式上，我们定义锚点在$\{P_2, P_3, P_4, P_5, P_6\}$上分别具有$\{32^2, 64^2, 128^2, 256^2, 512^2\}$像素的面积。如[29]中所述，我们在每个层级也使用多种宽高比的锚点{1:2, 1:1, 2:1}。因此，整个金字塔上总共有15个锚点。

我们根据锚框与真实边界框的交并比（IoU）为其分配训练标签，方法如[29]所述。具体而言，若一个锚框与某个真实框的IoU最高，或与任意真实框的IoU超过0.7，则将其标记为正样本；若其与所有真实框的IoU均低于0.3，则标记为负样本。需要注意的是，真实框的尺度并未被显式用于将其分配到金字塔的特定层级；相反，真实框是与已分配到金字塔各层级的锚框相关联的。因此，除了[29]中的规则外，我们未引入额外规则。

我们注意到，所有特征金字塔层级的头部参数是共享的；我们也评估了不共享参数的替代方案，并观察到相似的准确度。共享参数的良好性能表明，我们的金字塔所有层级都具有相似的语义水平。这一优势类似于使用特征化图像金字塔的情况，其中通用的头部分类器可以应用于任何图像尺度上计算得到的特征。

通过上述调整，RPN可以自然地与我们的FPN一起进行训练和测试，方式与[29]中相同。我们将在实验部分详细阐述实现细节。

4.2. 用于Fast R-CNN的特征金字塔网络

Fast R-CNN [11] 是一种基于区域的目标检测器，其中使用感兴趣区域（RoI）池化来提取特征。Fast R-CNN 通常在单尺度特征图上执行。为了将其与我们的 FPN 结合使用，我们需要将不同尺度的 RoI 分配到金字塔的不同层级。

我们将特征金字塔视为由图像金字塔生成。因此，我们可以借鉴基于区域的检测器[15, 11]在图像金字塔上运行时的分配策略。形式上，我们将输入图像中宽度为w、高度为h (的感兴趣区域（RoI）分配给网络)，并通过以下方式将其映射到特征金字塔的层级P_k:

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor. \quad (1)$$

这里224是ImageNet预训练的标准尺寸，k_0是目标层级，一个$w \times h$ =224^2的RoI应映射到该层级。类似于基于ResNet的Faster R-CNN系统[16]使用C_4作为单尺度特征图，我们将k_0设为4。直观上，公式(1)意味着如果RoI的尺度变小（例如224的1/2），它应被映射到更精细的分辨率层级（例如k =3）。

[1]Here we introduce P_6 only for covering a larger anchor scale of 512^2. P_6 is simply a stride two subsampling of P_5. P_6 is not used by the Fast R-CNN detector in the next section.

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. In [16], a ResNet's conv5 layers (a 9-layer deep subnetwork) are adopted as the head on top of the conv4 features, but our method has already harnessed conv5 to construct the feature pyramid. So unlike [16], we simply adopt RoI pooling to extract 7×7 features, and attach two hidden 1,024-d fully-connected (*fc*) layers (each followed by ReLU) before the final classification and bounding box regression layers. These layers are randomly initialized, as there are no pre-trained *fc* layers available in ResNets. Note that compared to the standard conv5 head, our 2-*fc* MLP head is lighter weight and faster.

Based on these adaptations, we can train and test Fast R-CNN on top of the feature pyramid. Implementation details are given in the experimental section.

5. Experiments on Object Detection

We perform experiments on the 80 category COCO detection dataset [21]. We train using the union of 80k train images and a 35k subset of val images (`trainval35k` [2]), and report ablations on a 5k subset of val images (`minival`). We also report final results on the standard test set (`test-std`) [21] which has no disclosed labels.

As is common practice [12], all network backbones are pre-trained on the ImageNet1k classification set [33] and then fine-tuned on the detection dataset. We use the pre-trained ResNet-50 and ResNet-101 models that are publicly available.[2] Our code is a reimplementation of `py-faster-rcnn`[3] using Caffe2.[4]

5.1. Region Proposal with RPN

We evaluate the COCO-style Average Recall (AR) and AR on small, medium, and large objects (AR_s, AR_m, and AR_l) following the definitions in [21]. We report results for 100 and 1000 proposals per images (AR^{100} and AR^{1k}).

Implementation details. All architectures in Table 1 are trained end-to-end. The input image is resized such that its shorter side has 800 pixels. We adopt synchronized SGD training on 8 GPUs. A mini-batch involves 2 images per GPU and 256 anchors per image. We use a weight decay of 0.0001 and a momentum of 0.9. The learning rate is 0.02 for the first 30k mini-batches and 0.002 for the next 10k. For all RPN experiments (including baselines), we include the anchor boxes that are outside the image for training, which is unlike [29] where these anchor boxes are ignored. Other implementation details are as in [29]. Training RPN with FPN on 8 GPUs takes about 8 hours on COCO.

[2] https://github.com/kaiminghe/deep-residual-networks

[3] https://github.com/rbgirshick/py-faster-rcnn

[4] https://github.com/caffe2/caffe2

5.1.1 Ablation Experiments

Comparisons with baselines. For fair comparisons with original RPNs [29], we run two baselines (Table 1(a, b)) using the single-scale map of C_4 (the same as [16]) or C_5, both using the same hyper-parameters as ours, including using 5 scale anchors of $\{32^2, 64^2, 128^2, 256^2, 512^2\}$. Table 1 (b) shows no advantage over (a), indicating that a single higher-level feature map is not enough because there is a trade-off between coarser resolutions and stronger semantics.

Placing FPN in RPN improves AR^{1k} to 56.3 (Table 1 (c)), which is **8.0** points increase over the single-scale RPN baseline (Table 1 (a)). In addition, the performance on small objects (AR_s^{1k}) is boosted by a large margin of 12.9 points. Our pyramid representation greatly improves RPN's robustness to object scale variation.

How important is top-down enrichment? Table 1(d) shows the results of our feature pyramid without the top-down pathway. With this modification, the 1×1 lateral connections followed by 3×3 convolutions are attached to the bottom-up pyramid. This architecture simulates the effect of reusing the pyramidal feature hierarchy (Fig. 1(b)).

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? Table 1(e) shows the ablation results of a top-down feature pyramid without the 1×1 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. As a results, FPN has an AR^{1k} score 10 points higher than Table 1(e).

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 P_2 (*i.e.*, the finest level in our pyramids). Similar to the single-scale baselines, we assign all anchors to the P_2 feature map. This variant (Table 1(f)) is better than the baseline but inferior to our approach. RPN is a sliding window detector with a fixed window size, so scanning over pyramid levels can increase its robustness to scale variance.

In addition, we note that using P_2 alone leads to more anchors (750k, Table 1(f)) caused by its large spatial resolution. This result suggests that a larger number of anchors is not sufficient in itself to improve accuracy.

我们在所有层级的全部RoI上附加了预测头（在Fast R-CNN中，预测头是类别特定的分类器和边界框回归器）。同样，所有预测头共享参数，无论它们处于哪个层级。在[16]中，ResNet的conv5层（一个9层深的子网络）被用作conv4特征之上的预测头，但我们的方法已经利用conv5构建了特征金字塔。因此与[16]不同，我们仅采用RoI池化提取7×7特征，并在最终的分类和边界框回归层之前附加两个隐藏的1,024维全连接（*fc*）层（每层后接ReLU激活函数）。由于ResNet中没有预训练的*fc*层可用，这些层采用随机初始化。值得注意的是，与标准的conv5预测头相比，我们的2层*fc*MLP预测头更轻量且速度更快。

基于这些调整，我们可以在特征金字塔上训练和测试Fast R-CNN。具体实现细节将在实验部分给出。

5. 目标检测实验

我们在80个类别的COCO检测数据集[21]上进行了实验。训练时使用了80k训练图像与35k验证图像子集（trainval35k [2]）的联合数据集，并在5k验证图像子集（minival）上进行消融实验。同时，我们在未公开标签的标准测试集（test-std）[21]上报告了最终结果。

按照惯例[12]，所有网络骨干均在ImageNet1k分类集[33]上进行预训练，随后在检测数据集上进行微调。我们使用公开可用的预训练ResNet-50和ResNet-101模型。[2] 我们的代码是基于Caffe2对py-faster-rcnn[3]的重新实现。[4]

5.1. 基于RPN的区域提议

我们按照[21]中的定义，评估了COCO风格的平均召回率（AR）以及小、中、大物体上的AR（AR_s、AR_m和AR_l）。我们报告了每张图像100和1000个提议的结果（AR^{100}和AR^{1k}）。

实现细节。表1中的所有架构均采用端到端训练。输入图像被调整大小，使其短边为800像素。我们在8个GPU上采用同步SGD训练。每个小批量包含每GPU 2张图像，每张图像256个锚点。我们使用的权重衰减为0.0001，动量为0.9。前30k个小批量的学习率为0.02，随后10k个为0.002。在所有RPN实验（包括基线）中，我们将在图像外的锚框也纳入训练，这与[29]中忽略这些锚框的做法不同。其他实现细节与[29]一致。在COCO数据集上，使用FPN训练RPN在8个GPU上大约需要8小时。

5.1.1 消融实验

与基线的比较。为了与原始RPNs [29]进行公平比较，我们运行了两个基线（表1(a, b)），使用与[16]相同的单尺度映射C_4 (或C_5，两者均采用与我们相同的超参数，包括使用5个尺度锚点$\{32^2, 64^2, 128^2, 256^2, 512^2\}$。表1(b)未显示出优于(a)的优势，这表明单一高层特征映射不足，因为较粗的分辨率与更强的语义之间存在权衡。

将FPN置于RPN中可将AR^{1k}提升至56.3（表1(c)），相比单尺度RPN基线（表1(a)）提高了8.0个点。此外，小物体检测性能（AR_s^{1k}）大幅提升了12.9个点。我们的金字塔表征显著增强了RPN对物体尺度变化的鲁棒性。

自上而下富集有多重要？表1(d)展示了我们不带自上而下路径的特征金字塔结果。通过这一修改，将1×1横向连接及随后的3×3卷积附加到自下而上的金字塔上。该架构模拟了重用金字塔特征层次结构（图1(b)）的效果。

表1(d)中的结果仅与RPN基线持平，远落后于我们的方法。我们推测这是因为自底向上金字塔（图1(b)）的不同层级间存在较大的语义鸿沟，对于极深的ResNet尤为明显。我们还评估了表1(d)中不共享头部参数的变体，但观察到性能同样下降。这一问题无法通过简单的层级专用头部解决。

横向连接有多重要？表1(e)展示了不带1×1横向连接的自顶向下特征金字塔的消融实验结果。这种自顶向下的金字塔具有强语义特征和精细分辨率。但我们认为这些特征的位置不够精确，因为这些特征图经历了多次下采样和上采样。通过横向连接，更精确的特征位置可以直接从自底向上特征图的更精细层级传递到自顶向下特征图中。因此，FPN的AR^{1k}得分比表1(e)高出10个百分点。

金字塔表示有多重要？与其依赖金字塔表示，不如将头部连接到最高分辨率、强语义的特征图P_2 (*i.e*上，这是我们金字塔中最精细的层级)。类似于单尺度基线，我们将所有锚点分配给P_2特征图。这种变体（表1(f)）优于基线但不及我们的方法。RPN是一种具有固定窗口大小的滑动窗口检测器，因此在金字塔层级上进行扫描可以增强其对尺度变化的鲁棒性。

此外，我们注意到，单独使用P_2会因其较大的空间分辨率导致更多锚点（750k，表1(f)）。这一结果表明，仅靠增加锚点数量本身并不足以提升精度。

[2] https://github.com/kaiminghe/deep-residual-networks
[3] https://github.com/rbgirshick/py-faster-rcnn
[4] https://github.com/caffe2/caffe2

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

Table 1. Bounding box proposal results using RPN [29], evaluated on the COCO `minival` set. All models are trained on `trainval35k`. The columns "lateral" and "top-down" denote the presence of lateral and top-down connections, respectively. The column "feature" denotes the feature maps on which the heads are attached. All results are based on ResNet-50 and share the same hyper-parameters.

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	*2fc*			52.9	28.8	11.9	32.4	43.4
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	**45.8**
Ablation experiments follow:										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	*2fc*	✓	✓	56.3	33.4	17.3	37.3	45.6

Table 2. Object detection results using **Fast R-CNN** [11] on *a fixed set of proposals* (RPN, $\{P_k\}$, Table 1(c)), evaluated on the COCO `minival` set. Models are trained on the `trainval35k` set. All results are based on ResNet-50 and share the same hyper-parameters.

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

Table 3. Object detection results using **Faster R-CNN** [29] evaluated on the COCO `minival` set. *The backbone network for RPN are consistent with Fast R-CNN.* Models are trained on the `trainval35k` set and use ResNet-50. †Provided by authors of [16].

5.2. Object Detection with Fast/Faster R-CNN

Next we investigate FPN for region-based (non-sliding window) detectors. We evaluate object detection by the COCO-style Average Precision (AP) and PASCAL-style AP (at a single IoU threshold of 0.5). We also report COCO AP on objects of small, medium, and large sizes (namely, AP_s, AP_m, and AP_l) following the definitions in [21].

Implementation details. The input image is resized such that its shorter side has 800 pixels. Synchronized SGD is used to train the model on 8 GPUs. Each mini-batch involves 2 image per GPU and 512 RoIs per image. We use a weight decay of 0.0001 and a momentum of 0.9. The learning rate is 0.02 for the first 60k mini-batches and 0.002 for the next 20k. We use 2000 RoIs per image for training and 1000 for testing. Training Fast R-CNN with FPN takes about 10 hours on the COCO dataset.

5.2.1 Fast R-CNN (on fixed proposals)

To better investigate FPN's effects on the region-based detector alone, we conduct ablations of Fast R-CNN on *a fixed set of proposals*. We choose to freeze the proposals as computed by RPN on FPN (Table 1(c)), because it has good performance on small objects that are to be recognized by the detector. For simplicity we do not share features between Fast R-CNN and RPN, except when specified.

As a ResNet-based Fast R-CNN baseline, following [16], we adopt RoI pooling with an output size of 14×14 and attach all conv5 layers as the hidden layers of the head. This gives an AP of 31.9 in Table 2(a). Table 2(b) is a baseline exploiting an MLP head with 2 hidden *fc* layers, similar to the head in our architecture. It gets an AP of 28.8, indicating that the 2-*fc* head does not give us any orthogonal advantage over the baseline in Table 2(a).

Table 2(c) shows the results of our FPN in Fast R-CNN. Comparing with the baseline in Table 2(a), our method improves AP by 2.0 points and small object AP by 2.1 points. Comparing with the baseline that also adopts a 2*fc* head (Table 2(b)), our method improves AP by 5.1 points.[5] These comparisons indicate that our feature pyramid is superior to single-scale features for a *region-based* object detector.

Table 2(d) and (e) show that removing top-down con-

[5]We expect a stronger architecture of the head [30] will improve upon our results, which is beyond the focus of this paper.

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR_s^{1k}	AR_m^{1k}	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

表1. 使用RPN[29]的边界框提议结果，在COCO minival集上评估。所有模型均在trainval35k上训练。“lateral”和“top-down”列分别表示是否存在横向和自上而下的连接。“feature”列表示头部所连接的特征图。所有结果均基于ResNet-50，并共享相同的超参数。

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

表2. 在COCO minival集上评估的使用Fast R-CNN [11]在*a fixed set of proposals* (RPN、$\{P_k\}$、表1(c))上的目标检测结果。模型在trainval35k集上训练。所有结果均基于ResNet-50并共享相同的超参数。

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

表3. 使用Faster R-CNN [29]在COCO minival集上评估的目标检测结果。*The backbone network for RPN are consistent with Fast R-CNN.*模型在trainval35k集上训练，并使用ResNet-50。†由[16]的作者提供。

5.2. 基于Fast/Faster R-CNN的目标检测

接下来我们研究基于区域（非滑动窗口）检测器的FPN。我们通过COCO风格的平均精度（AP）和PASCAL风格的AP（在单一IoU阈值为0.5时）来评估目标检测。我们还按照[21]中的定义，报告了小、中、大尺寸物体的COCO AP（即AP_s、AP_m和AP_l）。

实现细节。输入图像被调整大小，使其短边为800像素。使用同步SGD在8个GPU上训练模型。每个小批次包含每个GPU 2张图像，每张图像512个RoI。我们使用的权重衰减为0.0001，动量为0.9。前60k个小批次的学习率为0.02，随后20k个为0.002。训练时每张图像使用2000个RoI，测试时使用1000个。在COCO数据集上，使用FPN训练Fast R-CNN大约需要10小时。

5.2.1 快速R-CNN（基于固定候选区域）

为了更好地单独研究FPN对基于区域检测器的影响，我们在*a fixed set of proposals*上对Fast R-CNN进行了消融实验。我们选择固定候选框，如由RPN在FPN上计算得出（表1(c)），因为它在检测器需要识别的小物体上具有良好的性能。为简化起见，除非特别说明，我们不共享Fast R-CNN和RPN之间的特征。

作为基于ResNet的Fast R-CNN基线，我们遵循[16]的方法，采用输出尺寸为14×14的RoI池化，并将所有conv5层作为头部的隐藏层。这在表2(a)中实现了31.9的AP。表2(b)是一个使用具有2个隐藏*fc*层的MLP头部的基线，类似于我们架构中的头部设计。其AP为28.8，表明这种2-*fc*头部相比表2(a)的基线并未带来任何正交优势。

表2(c)展示了我们的FPN在Fast R-CNN中的结果。与表2(a)中的基线相比，我们的方法将AP提升了2.0个百分点，小目标AP提升了2.1个百分点。与同样采用2*fc*头结构的基线（表2(b)）相比，我们的方法将AP提升了5.1个百分点[5]。这些比较表明，对于*region-based*目标检测器而言，我们的特征金字塔优于单尺度特征。

表2(d)和(e)显示，移除自上而下的连

[5]We expect a stronger architecture of the head [30] will improve upon our results, which is beyond the focus of this paper.

method	backbone	competition	image pyramid	test-dev AP@.5	test-dev AP	test-dev AP_s	test-dev AP_m	test-dev AP_l	test-std AP@.5	test-std AP	test-std AP_s	test-std AP_m	test-std 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

Table 4. Comparisons of **single-model** results on the COCO detection benchmark. Some results were not available on the test-std set, so we also include the test-dev results (and for Multipath [40] on minival). †: http://image-net.org/challenges/talks/2016/GRMI-COCO-slidedeck.pdf. ‡: http://mscoco.org/dataset/#detections-leaderboard. §: This entry of AttractioNet [10] adopts VGG-16 for proposals and Wide ResNet [39] for object detection, so is not strictly a single-model result.

nections or removing lateral connections leads to inferior results, similar to what we have observed in the above subsection for RPN. It is noteworthy that removing top-down connections (Table 2(d)) significantly degrades the accuracy, suggesting that Fast R-CNN suffers from using the low-level features at the high-resolution maps.

In Table 2(f), we adopt Fast R-CNN on the single finest scale feature map of P_2. Its result (33.4 AP) is marginally worse than that of using all pyramid levels (33.9 AP, Table 2(c)). 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 $\{P_k\}$ and has thus already benefited from the pyramid representation.

5.2.2 Faster R-CNN (on consistent proposals)

In the above we used a fixed set of proposals to investigate the detectors. But in a Faster R-CNN system [29], the RPN and Fast R-CNN must use *the same network backbone* in order to make feature sharing possible. Table 3 shows the comparisons between our method and two baselines, all using *consistent* backbone architectures for RPN and Fast R-CNN. Table 3(a) shows our reproduction of the baseline Faster R-CNN system as described in [16]. Under controlled settings, our FPN (Table 3(c)) is better than this strong baseline by **2.3** points AP and **3.8** points AP@0.5.

Note that Table 3(a) and (b) are baselines that are much stronger than the baseline provided by He *et al*. [16] in Table 3(*). We find the following implementations contribute to the gap: (i) We use an image scale of 800 pixels instead of 600 in [11, 16]; (ii) We train with 512 RoIs per image which accelerate convergence, in contrast to 64 RoIs in [11, 16]; (iii) We use 5 scale anchors instead of 4 in [16] (adding 32^2); (iv) At test time we use 1000 proposals per image instead of 300 in [16]. So comparing with He *et al*.'s ResNet-50 Faster R-CNN baseline in Table 3(*), our method improves AP by 7.6 points and AP@0.5 by 9.6 points.

Sharing features. In the above, for simplicity we do not share the features between RPN and Fast R-CNN. In Table 5, we evaluate sharing features following the 4-step training described in [29]. Similar to [29], we find that sharing features improves accuracy by a small margin. Feature sharing also reduces the testing time.

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

Table 5. More object detection results using Faster R-CNN and our FPNs, evaluated on minival. Sharing features increases train time by 1.5× (using 4-step training [29]), but reduces test time.

Running time. With feature sharing, our FPN-based Faster R-CNN system has inference time of 0.148 seconds per image on a single NVIDIA M40 GPU for ResNet-50, and 0.172 seconds for ResNet-101.[6] As a comparison, the single-scale ResNet-50 baseline in Table 3(a) runs at 0.32 seconds. Our method introduces small extra cost by the extra layers in the FPN, but has a lighter weight head. 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.

5.2.3 Comparing with COCO Competition Winners

We find that our ResNet-101 model in Table 5 is not sufficiently trained with the default learning rate schedule. So we increase the number of mini-batches by 2× at each learning rate when training the Fast R-CNN step. This increases AP on minival to 35.6, without sharing features. This model is the one we submitted to the COCO detection leaderboard, shown in Table 4. We have not evaluated its feature-sharing version due to limited time, which should be slightly better as implied by Table 5.

Table 4 compares our method with the *single-model* results of the COCO competition winners, including the 2016 winner G-RMI and the 2015 winner Faster R-CNN+++. *Without adding bells and whistles*, our single-model entry has surpassed these strong, heavily engineered competitors.

[6] These runtimes are updated from an earlier version of this paper.

method	backbone	competition	image pyramid	test-dev AP@.5	AP	AP_s	AP_m	AP_l	test-std 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

表4. COCO检测基准上单模型结果的比较。部分结果在test-std集上不可用，因此我们同时纳入了test-dev集的结果（以及Multipath [40]在minival集上的结果）。†: http://image-net.org/challenges/talks/2016/GRMI-COCO-slidedeck.pdf。‡: http://mscoco.org/dataset/#detections-leaderboard。§: AttractioNet [10]的此项采用VGG-16生成候选区域，并采用宽残差网络[39]进行目标检测，故严格来说并非单模型结果。

移除横向连接或移除侧向连接会导致较差的结果，这与我们在上述RPN子章节中观察到的情况相似。值得注意的是，移除自上而下的连接（表2(d)）会显著降低准确率，这表明Fast R-CNN在使用高分辨率地图中的低层特征时存在问题。

在表2(f)中，我们在P_2的单一最精细尺度特征图上采用Fast R-CNN。其结果（33.4 AP）略逊于使用所有金字塔层级的结果（33.9 AP，表2(c)）。我们认为这是因为RoI池化是一种类似形变的操作，对区域尺度较不敏感。尽管该变体精度良好，但它基于$\{P_k\}$的RPN建议框，因此已受益于金字塔表示。

5.2.2 Faster R-CNN（基于一致建议框）

在上述内容中，我们使用了一组固定的提议来研究检测器。但在Faster R-CNN系统[29]中，RPN和Fast R-CNN必须使用*the same network back- bone*，以实现特征共享。表3展示了我们的方法与两个基线的比较，所有方法均采用*consistent*骨干架构用于RPN和Fast R-CNN。表3(a)展示了我们根据[16]复现的基线Faster R-CNN系统。在受控设置下，我们的FPN（表3(c)）比这一强基线高出2.3个AP点和3.8个AP@0.5点。

请注意，表3(a)和(b)的基线远强于He *et al*在表3(*)中提供的基线[16]。我们发现以下实现细节导致了性能差距：(i) 我们使用的图像尺度为800像素，而非[11, 16]中的600像素；(ii) 我们每张图像使用512个RoI进行训练，这加速了收敛，而[11, 16]中仅使用64个RoI；(iii) 我们采用5尺度锚框而非[16]中的4尺度（增加了32^2）；(iv) 测试时每张图像使用1000个候选区域，而非[16]中的300个。因此，与He *et al*在表3(*)中的ResNet-50 Faster R-CNN基线相比，我们的方法将AP提升了7.6个百分点，AP@0.5提升了9.6个百分点。

共享特征。在上述中，为简化起见，我们未在RPN和Fast R-CNN之间共享特征。在Ta-

	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**

表5. 使用Faster R-CNN和我们的FPN在minival上评估的更多目标检测结果。特征共享将训练时间增加了1.5×（使用四步训练法[29]），但减少了测试时间。

在表5中，我们按照[29]中描述的四步训练法评估了特征共享的效果。与[29]类似，我们发现共享特征能小幅提升准确率。特征共享同时减少了测试所需时间。

运行时间。通过特征共享，我们基于FPN的Faster R-CNN系统在单个NVIDIA M40 GPU上，使用ResNet-50时每张图像的推理时间为0.148秒，使用ResNet-101时为0.172秒。[6] 作为对比，表3(a)中的单尺度ResNet-50基线运行时间为0.32秒。我们的方法因FPN中的额外层引入了少量额外开销，但头部结构更轻量。总体而言，我们的系统比基于ResNet的Faster R-CNN对应版本更快。我们相信本方法的高效性和简洁性将有益于未来的研究和应用。

5.2.3 与COCO竞赛优胜者对比

我们发现表5中的ResNet-101模型在默认学习率调度下训练不足。因此，在训练Fast R-CNN步骤时，我们在每个学习率阶段将小批量数量增加了2×。这使得在不共享特征的情况下，minival数据集上的AP提升至35.6。该模型是我们提交至COCO检测排行榜的版本，如表4所示。由于时间有限，我们尚未评估其特征共享版本，但根据表5的暗示，该版本性能应略优于当前模型。

表4将我们的方法与COCO竞赛获胜者的*single-model*结果进行了比较，包括2016年冠军G-RMI和2015年冠军Faster R-CNN+++。*Without adding bells and whistles*，我们的单模型参赛结果已超越这些实力强劲、经过深度优化的竞争对手。

[6] These runtimes are updated from an earlier version of this paper.

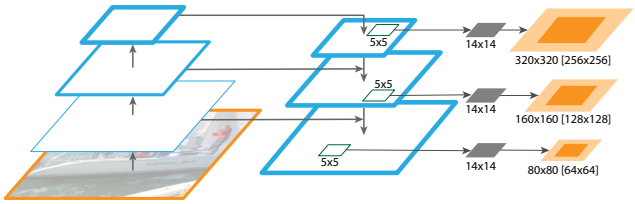


Figure 4. FPN for object segment proposals. The feature pyramid is constructed with identical structure as for object detection. We apply a small MLP on 5×5 windows to generate dense object segments with output dimension of 14×14. Shown in orange are the size of the image regions the mask corresponds to for each pyramid level (levels P_{3-5} are shown here). Both the corresponding image region size (light orange) and canonical object size (dark orange) are shown. Half octaves are handled by an MLP on 7x7 windows ($7 \approx 5\sqrt{2}$), not shown here. Details are in the appendix.

On the `test-dev` set, our method increases over the existing best results by **0.5** points of AP (36.2 *vs*. 35.7) and **3.4** points of AP@0.5 (59.1 *vs*. 55.7). 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. This could only be achieved by high-resolution image inputs with previous methods.

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*. These improvements are complementary to FPNs and should boost accuracy further.

Recently, FPN has enabled new top results in *all* tracks of the COCO competition, including detection, instance segmentation, and keypoint estimation. See [14] for details.

6. Extensions: Segmentation Proposals

Our method is a generic pyramid representation and can be used in applications other than object detection. In this section we use FPNs to generate segmentation proposals, following the DeepMask/SharpMask framework [27, 28].

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].

It is easy to adapt FPN to generate mask proposals. We use a fully convolutional setup for both training and inference. We construct our feature pyramid as in Sec. 5.1 and set $d = 128$. 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.

	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
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

Table 6. Instance segmentation proposals evaluated on the first 5k COCO `val` images. All models are trained on the `train` set. DeepMask, SharpMask, and FPN use ResNet-50 while InstanceFCN uses VGG-16. DeepMask and SharpMask performance is computed with models available from https://github.com/facebookresearch/deepmask (both are the 'zoom' variants). †Runtimes are measured on an NVIDIA M40 GPU, except the InstanceFCN timing which is based on the slower K40.

6.1. Segmentation Proposal Results

Results are shown in Table 6. We report segment AR and segment AR on small, medium, and large objects, always for 1000 proposals. Our baseline FPN model with a single 5×5 MLP achieves an AR of 43.4. Switching to a slightly larger 7×7 MLP leaves accuracy largely unchanged. Using both MLPs together increases accuracy to 45.7 AR. Increasing mask output size from 14×14 to 28×28 increases AR another point (larger sizes begin to degrade accuracy). Finally, doubling the training iterations increases AR to 48.1.

We also report comparisons to DeepMask [27], SharpMask [28], and InstanceFCN [4], the previous state of the art methods in mask proposal generation. We outperform the accuracy of these approaches by over **8.3** points AR. In particular, we nearly double the accuracy on small objects.

Existing mask proposal methods [27, 28, 4] are based on densely sampled image pyramids (*e.g.*, scaled by $2^{\{-2:0.5:1\}}$ in [27, 28]), making them computationally expensive. Our approach, based on FPNs, is substantially faster (our models run at 6 to 7 FPS). These results demonstrate that our model is a generic feature extractor and can replace image pyramids for other multi-scale detection problems.

7. Conclusion

We have presented a clean and simple framework for building feature pyramids inside ConvNets. Our method shows significant improvements over several strong baselines and competition winners. Thus, it provides a practical solution for research and applications of feature pyramids, without the need of computing image pyramids. Finally, our study suggests that despite the strong representational power of deep ConvNets and their implicit robustness to scale variation, it is still critical to explicitly address multi-scale problems using pyramid representations.

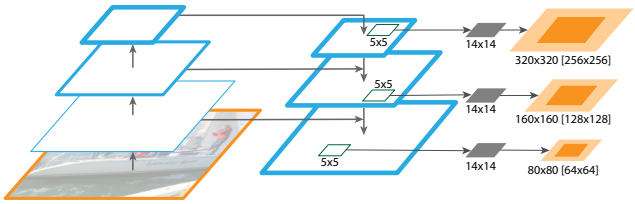


图4. 用于物体分割提议的FPN。特征金字塔的构建结构与物体检测相同。我们在5×5窗口上应用小型MLP，生成输出维度为14×14的密集物体分割片段。橙色部分展示了每个金字塔层级（此处展示了P_{3-5}层级）中掩码对应的图像区域尺寸。同时显示了对应的图像区域尺寸（浅橙色）和规范物体尺寸（深橙色）。半八度通过7x7窗口（$7\approx 5\sqrt{2}$）上的MLP处理，此处未展示。详见附录。

在测试开发集上，我们的方法将现有最佳结果提升了0.5个AP点（36.2 *vs*对比35.7）和3.4个AP@0.5点（59.1 *vs*对比55.7）。值得注意的是，我们的方法不依赖图像金字塔，仅使用单一输入图像尺度，但在小尺度物体上仍具有突出的AP表现。这在此前的方法中仅能通过高分辨率图像输入实现。

此外，我们的方法并未采用许多流行的改进技术，例如迭代回归[9]、难负样本挖掘[35]、上下文建模[16]、更强的数据增强[22]等。这些改进与FPN是互补的，应能进一步提升准确度。

最近，FPN在COCO竞赛的多个赛道中实现了新的最佳成绩，包括目标检测、实例分割和关键点估计。详情请参阅[14]。

6. 扩展：分割提案

我们的方法是一种通用的金字塔表示，可用于除目标检测之外的其他应用。在本节中，我们遵循DeepMask/SharpMask框架[27, 28]，使用FPN生成分割候选区域。

DeepMask/SharpMask在图像裁剪上进行训练，用于预测实例分割以及物体/非物体的得分。在推理阶段，这些模型以卷积方式运行，以在图像中生成密集的候选区域。为了生成多尺度的分割结果，需要使用图像金字塔[27, 28]。

将FPN调整为生成掩码提议非常简单。我们在训练和推理中均采用全卷积设置。如第5.1节所述构建特征金字塔，并设定$d=$为128。在特征金字塔的每一层之上，我们应用一个5×5的小型MLP，以全卷积方式预测14×14的掩码和物体得分，详见图4。此外，受[27, 28]图像金字塔中每八度使用2个尺度的启发，我们采用输入尺寸为7×7的第二个MLP来处理半八度。这两个MLP的作用类似于RPN中的锚点。该架构采用端到端训练；完整实现细节见附录。

	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
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

表6. 在COCO验证集前5千张图像上评估的实例分割提案。所有模型均在训练集上训练。DeepMask、SharpMask和FPN采用ResNet-50，而Instance-FCN采用VGG-16。DeepMask与SharpMask的性能通过https://github.com/facebookresearch/deepmask提供的模型计算（两者均为'zoom'变体）。†运行时间在NVIDIA M40 GPU上测得，其中InstanceFCN的计时基于较慢的K40。

6.1. 分割提案结果

结果如表6所示。我们报告了在1000个提议下，针对小、中、大物体的分段平均召回率（segment AR）。使用单层5×5 MLP的基线FPN模型实现了43.4的AR。切换到稍大的7×7 MLP后，准确率基本保持不变。同时使用两个MLP将准确率提升至45.7 AR。将掩码输出尺寸从14×14增加到28×28，使AR再提升一个点（更大尺寸会开始降低准确率）。最后，将训练迭代次数加倍使AR提升至48.1。

我们还与DeepMask [27]、SharpMask [28]和InstanceFCN [4]进行了比较，这些是之前掩模提议生成领域的最先进方法。我们的方法在准确率上超过这些方法超过8.3个AR点。特别是，我们在小物体上的准确率几乎翻了一番。

现有的掩码提议方法[27, 28, 4]基于密集采样的图像金字塔（*e.g.*，如在[27, 28]中按$2^{\{-2:0.5:1\}}$缩放），导致计算成本高昂。我们基于FPN的方法则显著更快（我们的模型运行速度为6至7 FPS）。这些结果表明，我们的模型是一个通用特征提取器，能够替代图像金字塔用于其他多尺度检测问题。

7. 结论

我们提出了一个在卷积网络内部构建特征金字塔的简洁框架。我们的方法相较于多个强大的基线模型和竞赛优胜方案均展现出显著改进。因此，该方法为特征金字塔的研究与应用提供了实用解决方案，且无需计算图像金字塔。最后，我们的研究表明：尽管深度卷积网络具备强大的表征能力，并隐式具备尺度变化的鲁棒性，但利用金字塔表征显式处理多尺度问题仍然至关重要。

A. Implementation of Segmentation Proposals

We use our feature pyramid networks to efficiently generate object segment proposals, adopting an image-centric training strategy popular for object detection [11, 29]. Our FPN mask generation model inherits many of the ideas and motivations from DeepMask/SharpMask [27, 28]. However, in contrast to these models, which were trained on image crops and used a densely sampled image pyramid for inference, we perform fully-convolutional training for mask prediction on a feature pyramid. While this requires changing many of the specifics, our implementation remains similar in spirit to DeepMask. Specifically, to define the label of a mask instance at each sliding window, we think of this window as being a crop on the input image, allowing us to inherit definitions of positives/negatives from DeepMask. We give more details next, see also Fig. 4 for a visualization.

We construct the feature pyramid with P_{2-6} using the same architecture as described in Sec. 5.1. We set $d = 128$. Each level of our feature pyramid is used for predicting masks at a different scale. As in DeepMask, we define the scale of a mask as the max of its width and height. Masks with scales of $\{32, 64, 128, 256, 512\}$ pixels map to $\{P_2, P_3, P_4, P_5, P_6\}$, respectively, and are handled by a 5×5 MLP. As DeepMask uses a pyramid with half octaves, we use a second slightly larger MLP of size 7×7 ($7 \approx 5\sqrt{2}$) to handle half-octaves in our model (*e.g.*, a $128\sqrt{2}$ scale mask is predicted by the 7×7 MLP on P_4). Objects at intermediate scales are mapped to the nearest scale in log space.

As the MLP must predict objects at a range of scales for each pyramid level (specifically a half octave range), some padding must be given around the canonical object size. We use 25% padding. This means that the mask output over $\{P_2, P_3, P_4, P_5, P_6\}$ maps to $\{40, 80, 160, 320, 640\}$ sized image regions for the 5×5 MLP (and to $\sqrt{2}$ larger corresponding sizes for the 7×7 MLP).

Each spatial position in the feature map is used to predict a mask at a different location. Specifically, at scale P_k, each spatial position in the feature map is used to predict the mask whose center falls within 2^k pixels of that location (corresponding to ± 1 cell offset in the feature map). If no object center falls within this range, the location is considered a negative, and, as in DeepMask, is used only for training the score branch and not the mask branch.

The MLP we use for predicting the mask and score is fairly simple. We apply a 5×5 kernel with 512 outputs, followed by sibling fully connected layers to predict a 14×14 mask (14^2 outputs) and object score (1 output). The model is implemented in a fully convolutional manner (using 1×1 convolutions in place of fully connected layers). The 7×7 MLP for handling objects at half octave scales is identical to the 5×5 MLP except for its larger input region.

During training, we randomly sample 2048 examples per mini-batch (128 examples per image from 16 images) with a positive/negative sampling ratio of 1:3. The mask loss is given 10× higher weight than the score loss. This model is trained end-to-end on 8 GPUs using synchronized SGD (2 images per GPU). We start with a learning rate of 0.03 and train for 80k mini-batches, dividing the learning rate by 10 after 60k mini-batches. The image scale is set to 800 pixels during training and testing (we do not use scale jitter). During inference our fully-convolutional model predicts scores at all positions and scales and masks at the 1000 highest scoring locations. We do not perform any non-maximum suppression or post-processing.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 1984.

[2] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[3] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016.

[4] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *TPAMI*, 2014.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.

[8] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016.

[9] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. In *ICCV*, 2015.

[10] S. Gidaris and N. Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. In *BMVC*, 2016.

[11] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[13] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv:1703.06870*, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*. 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

A. 分割提案的实现

我们利用特征金字塔网络高效生成对象分割提议，采用了在目标检测中流行的以图像为中心的训练策略[11, 29]。我们的FPN掩码生成模型继承了DeepMask/SharpMask[27, 28]的诸多理念与动机。然而，与这些在图像裁剪上训练并使用密集采样图像金字塔进行推理的模型不同，我们在特征金字塔上执行全卷积训练以预测掩码。虽然这需要调整许多具体细节，但我们的实现在本质上仍与DeepMask相似。具体而言，为定义每个滑动窗口处掩码实例的标签，我们将该窗口视为输入图像上的裁剪区域，从而能够沿用DeepMask中正负样本的定义标准。下文将提供更多细节，可视化示意图可参见图4。

我们采用与第5.1节所述相同的架构，利用P_{2-6}构建特征金字塔。我们将$d=$设为128。特征金字塔的每一层级用于预测不同尺度的掩码。如DeepMask所述，我们将掩码的尺度定义为其宽度和高度的最大值。尺度为{32, 64, 128, 256, 512}像素的掩码分别对应$\{P_2, P_3, P_4, P_5, P_6\}$，并由5×5的多层感知机处理。由于DeepMask使用半八度的金字塔结构，我们在模型中采用第二个稍大的7×7（$7 \approx 5\sqrt{2}$）多层感知机来处理半八度情况（*e.g*例如，$128\sqrt{2}$尺度的掩码由7×7多层感知机在P_4上预测）。中间尺度的对象在对数空间中被映射到最接近的尺度。

由于MLP必须在每个金字塔层级预测一系列尺度的物体（具体为半个八度范围），因此需要在规范物体尺寸周围提供一定的填充。我们使用25%的填充。这意味着在5×5 MLP中，$\{P_2, P_3, P_4, P_5, P_6\}$上的掩码输出映射到{40, 80, 160, 320, 640}尺寸的图像区域（对于7×7 MLP则映射到$\sqrt{2}$倍更大的对应尺寸）。

特征图中的每个空间位置用于预测不同位置的掩码。具体来说，在尺度P_k下，特征图中的每个空间位置用于预测其中心落在该位置2^k像素范围内的掩码（对应特征图中±1个单元的偏移）。如果没有物体中心落在此范围内，该位置被视为负样本，并且与DeepMask中一样，仅用于训练得分分支，而不用于掩码分支。

我们用于预测掩码和分数的MLP相当简单。我们应用一个5×5卷积核，输出512维特征，随后通过并行的全连接层来预测一个14×14的掩码（14^2个输出）和目标分数（1个输出）。该模型以全卷积方式实现（使用1×1卷积替代全连接层）。用于处理半八度尺度目标的7×7 MLP与5×5 MLP结构相同，仅输入区域更大。

在训练过程中，我们每个小批量随机采样2048个样本（从16张图像中每张图像选取128个样本），并正负样本采样比例为1:3。掩码损失的权重比得分损失高10×。该模型使用同步SGD（每GPU处理2张图像）在8个GPU上进行端到端训练。初始学习率为0.03，训练8万个小批量，在6万个小批量后将学习率除以10。训练和测试阶段图像尺度均设置为800像素（未使用尺度抖动）。推理过程中，我们的全卷积模型会在所有位置和尺度上预测得分，并在得分最高的1000个位置生成掩码。未进行任何非极大值抑制或后处理操作。

参考文献

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, 与 J. M. Ogden。图像处理中的金字塔方法。*RCA engineer*，1984年。[2] S. Bell, C. L. Zitnick, K. Bala, 与 R. Girshick。内外网：通过跳跃池化和循环神经网络在上下文中检测物体。载于 *CVPR*，2016年。[3] Z. Cai, Q. Fan, R. S. Feris, 与 N. Vasconcelos。一种用于快速目标检测的统一多尺度深度卷积神经网络。载于 *ECCV*，2016年。[4] J. Dai, K. He, Y. Li, S. Ren, 与 J. Sun。实例敏感的完全卷积网络。载于 *ECCV*，2016年。[5] N. Dalal 与 B. Triggs。用于人体检测的定向梯度直方图。载于 *CVPR*，2005年。[6] P. Dollár, R. Appel, S. Belongie, 与 P. Perona。用于目标检测的快速特征金字塔。*TPAMI*，2014年。[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, 与 D. Ramanan。使用基于判别性训练的部件模型进行目标检测。*TPAMI*，2010年。[8] G. Ghiasi 与 C. C. Fowlkes。用于语义分割的拉普拉斯金字塔重建与优化。载于 *ECCV*，2016年。[9] S. Gidaris 与 N. Komodakis。通过多区域及语义分割感知的CNN模型进行目标检测。载于 *ICCV*，2015年。

[10] S. Gidaris 和 N. Komodakis。Attend Refine Repeat：通过内外定位进行主动候选框生成。发表于 *BMVC*，2016年。

[11] R. Girshick. Fast R-CNN. 发表于 *ICCV*, 2015年。

[12] R. Girshick, J. Donahue, T. Darrell 和 J. Malik。用于精确目标检测和语义分割的丰富特征层次结构。发表于 *CVPR*，2014年。

[13] B. Hariharan, P. Arbeláez, R. Girshick, 和 J. Malik。用于目标分割和细粒度定位的超列方法。发表于*CVPR*，2015年。

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *arXiv:1703.06870*, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. 用于视觉识别的深度卷积网络中的空间金字塔池化。见 *ECCV*。2014。

[[16] K. He, X. Zhang, S. Ren, and J. Sun. 用于图像识别的深度残差学习。 发表于 *CVPR*, 2016.

[17] S. Honari, J. Yosinski, P. Vincent, and C. Pal. Recombinator networks: Learning coarse-to-fine feature aggregation. In *CVPR*, 2016.

[18] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. SSD: Single shot multibox detector. In *ECCV*, 2016.

[23] W. Liu, A. Rabinovich, and A. C. Berg. ParseNet: Looking wider to see better. In *ICLR workshop*, 2016.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[26] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[27] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015.

[28] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[30] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *PAMI*, 2016.

[31] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[32] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Carnegie Mellon University, 1995.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[35] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[37] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[38] R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. *IEE Proc. on Vision, Image, and Signal Processing*, 1994.

[39] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[40] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. In *BMVC*, 2016.

[17] S. Honari, J. Yosinski, P. Vincent, 与 C. Pal。重组网络：学习从粗到细的特征聚合。发表于 *CVPR*，2016年。[18] T. Kong, A. Yao, Y. Chen, 与 F. Sun。Hypernet：迈向精确的区域提议生成与联合目标检测。发表于 *CVPR*，2016年。[19] A. Krizhevsky, I. Sutskever, 与 G. Hinton。使用深度卷积神经网络进行ImageNet分类。发表于 *NIPS*，2012年。[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, 与 L. D. Jackel。反向传播应用于手写邮政编码识别。*Neural compu- tation*，1989年。[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, 与 C. L. Zitnick。Microsoft COCO：上下文中的常见物体。发表于 *ECCV*，2014年。[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, 与 S. Reed。SSD：单次多框检测器。发表于 *ECCV*，2016年。[23] W. Liu, A. Rabinovich, 与 A. C. Berg。ParseNet：看得更广以见得更好。发表于 *ICLR workshop*，2016年。[24] J. Long, E. Shelhamer, 与 T. Darrell。用于语义分割的全卷积网络。发表于 *CVPR*，2015年。[25] D. G. Lowe。来自尺度不变关键点的独特图像特征。*IJCV*，2004年。[26] A. Newell, K. Yang, 与 J. Deng。用于人体姿态估计的堆叠沙漏网络。发表于 *ECCV*，2016年。[27] P. O. Pinheiro, R. Collobert, 与 P. Dollar。学习分割候选物体。发表于 *NIPS*，2015年。[28] P. O. Pinheiro, T.-Y. Lin, R. Collobert, 与 P. Dollár。学习优化物体分割。发表于 *ECCV*，2016年。[29] S. Ren, K. He, R. Girshick, 与 J. Sun。Faster R-CNN：利用区域提议网络实现实时目标检测。发表于 *NIPS*，2015年。[30] S. Ren, K. He, R. Girshick, X. Zhang, 与 J. Sun。卷积特征图上的目标检测网络。*PAMI*，2016年。[31] O. Ronneberger, P. Fischer, 与 T. Brox。U-Net：用于生物医学图像分割的卷积网络。发表于 *MIC- CAI*，2015年。[32] H. Rowley, S. Baluja, 与 T. Kanade。视觉场景中的人脸检测。技术报告 CMU-CS-95-158R，卡内基梅隆大学，1995年。[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, 与 L. Fei-Fei。ImageNet大规模视觉识别挑战赛。*IJCV*，2015年。[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, 与 Y. LeCun。Overfeat：使用卷积网络进行集成识别、定位与检测。发表于 *ICLR*，2014年。[35] A. Shrivastava, A. Gupta, 与 R. Girshick。使用在线难例挖掘训练基于区域的目标检测器。发表于 *CVPR*，2016年。[36] K. Simonyan 与 A. Zisserman。用于大规模图像识别的极深卷积网络。发表于 *ICLR*，2015年。[37] J. R. Uijlings, K. E. van de Sande, T. Gevers, 与 A. W. Smeulders。用于目标识别的选择性搜索。*IJCV*，2013年。[38] R. Vaillant, C. Monrocq, 与 Y. LeCun。图像中物体定位的原创方法。*IEE Proc. on Vision, Image, and Signal Processing*, 1994。[39] S. Zagoruyko 与 N. Komodakis。宽残差网络。发表于 *BMVC*, 2016。[40] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, 与 P. Dollár。用于物体检测的多路径网络。发表于 *BMVC*, 2016。