Feature Pyramid Grids

Kai Chen[1,2*] Yuhang Cao[2] Chen Change Loy[3] Dahua Lin[2] Christoph Feichtenhofer[4]

[1]SenseTime Research [2]The Chinese University of Hong Kong
[3]Nanyang Technological University [4]Facebook AI Research (FAIR)

Abstract. Feature pyramid networks have been widely adopted in the object detection literature to improve feature representations for better handling of variations in scale. In this paper, we present Feature Pyramid Grids (FPG), a deep multi-pathway feature pyramid, that represents the feature scale-space as a regular grid of parallel bottom-up pathways which are fused by multi-directional lateral connections. FPG can improve single-pathway feature pyramid networks by significantly increasing its performance at similar computation cost, highlighting importance of deep pyramid representations. In addition to its general and uniform structure, over complicated structures that have been found with neural architecture search, it also compares favorably against such approaches without relying on search. We hope that FPG with its uniform and effective nature can serve as a strong component for future work in object recognition.

1 Introduction

It seems trivial how human perception can simultaneously recognize visual information across various levels of different resolution. For machine perception, recognizing objects at various scales has been a classical challenge in visual recognition over decades [2,1,15,27,29]. Numerous methods have been developed to build pyramid representations [2,1,15] as an effective way to model the scale-space, by building a hierarchical pyramid ranging from large to small image scales. Such classical pyramid representations are typically built by subsequent filtering (blurring) and subsampling operations applied to the image.

In recent deep learning approaches, a bottom-up pathway is inherently built by ConvNets [18] that hierarchically abstract information from higher to lower resolution in deeper layers, also by hierarchical filtering and subsampling. For object detection tasks, Feature Pyramid Networks (FPN) [19], an effective representation for multi-scale features has become popular. FPN augments ConvNets with a second top-down pathway and lateral connections to enrich high-resolution features with semantic information from deeper lower-resolution features.

There exist recent efforts [7,38] of applying Neural Architecture Search (NAS) to find deeper feature pyramid representations that have shown strong experimental results. NAS-FPN [7] defines a search space for the modular pyramidal architecture and adopts reinforcement learning to search the best performing one and Auto-FPN [38] proposed new search spaces for both FPN and the box head.

*Work done during an internship at Facebook AI Research

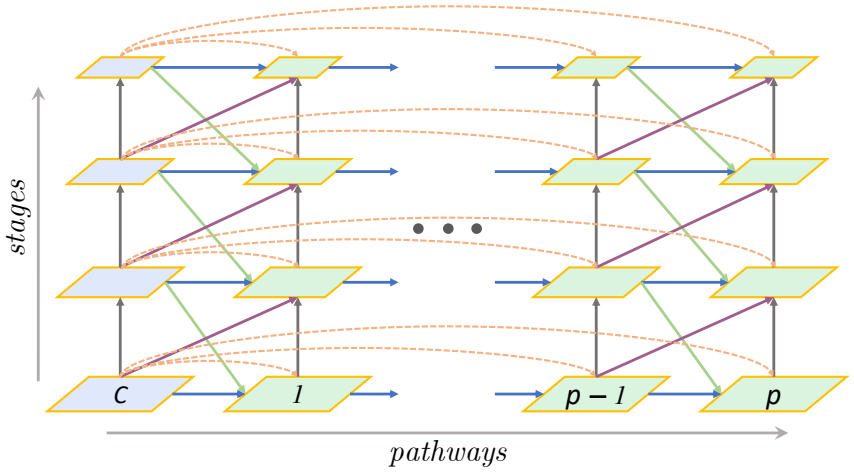

Fig. 1: **A Feature Pyramid Grid (FPG)** connects the *backbone features*, C, of a ConvNet with a regular structure of p parallel top-down pyramid *pathways* which are fused by multi-directional *lateral connections*, AcrossSame →, AcrossUp ↗, AcrossDown ↘, and AcrossSkip ↷. AcrossSkip are direct connections while all other types use convolutional and ReLU layers.

While search-based approaches have indeed shown new levels of performance that outperform conventional, manually designed FPN structures, there are several implications *w.r.t.* NAS in the context of finding improved architectures.

(i) The final network structure is often complicated hence not very comprehensible, limiting the adoption of the models by the community.

(ii) The search cost incurred by NAS is large, implicating up to thousands of TPU hours [7] to find an optimal architecture.

(iii) The discovered architecture may not generalize well to other detection frameworks. To give an example, NAS-FPN achieves a good performance on RetinaNet (which it was searched for), but it is unclear if it also performs similarly on other detection architectures.

In this paper, we present Feature Pyramid Grids (FPG), a deep multi-pathway feature pyramid network that represents the feature scale-space as a regular grid of parallel pathways fused by multi-directional lateral connections between them, as shown in Fig. 1. FPG enriches the hierarchical feature representation built internally in the backbone pathway of a ConvNet with *multiple* pyramid pathways in parallel. On a high-level, FPG is a deep generalization of FPN [19] from one to p pathways under a dense lateral connectivity structure.

Different from FPN, all the individual pathways are built in a bottom-up manner, similar to the backbone pathway that goes from the input image to a prediction output. To form a deep *grid of feature pyramids*, the pyramid pathways are intertwined with various lateral connections, both across-scale as well as within-scale to enable information exchange across all levels. We categorize these *lateral connections* into four types, AcrossSame →, AcrossUp ↗, AcrossDown ↘, and AcrossSkip ↷, as illustrated in Fig. 1.

Conceptually, the idea of FPG is generic, and it can be instantiated with different pathway and lateral connection design as well as implementation specifics. Our experiments systematically analyze the importance of key components (parallel pathways and lateral connections) in feature pyramid design using our unified FPG framework. Our aim is to strike a good trade-off between accuracy and computation, and we approach it by starting with a densely connected pyramid grid that is subsequently contracted based on systematic ablation.

In our evaluation, we are interested in the questions (1) if FPG can improve over FPN under *similar* complexity cost, and (2) if FPG could compete with NAS-optimized pyramid structures, even though being systematically designed.

We apply FPG to single-stage (RetinaNet [20]), two-stage (Faster R-CNN [31], Mask R-CNN [9] and cascaded (Cascade R-CNN [4]) detectors. Our findings are that, under similar computational cost, FPG performs better than FPN and NAS-FPN on various models and settings. Concretely, adopting the same setting as in NAS-FPN, our FPG achieves 0.2%, 1.5%, 2.3% and 2.2% higher mean Average Precision (mAP) than NAS-FPN on RetinaNet, Faster R-CNN, Mask R-CNN and Cascade R-CNN, respectively.

Our ablations reveal that a straightforward extension of FPN from one to many pathways does not succeed, but FPG is able to consistently increase accuracy for deeper pyramid representations. Overall, our experiments show that FPG is efficient and generalizes well across detectors, providing better performance than architecture searched pyramid networks. Given its systematic nature, we hope that FPG can serve as a strong component for future work in object recognition.

2 Related Work

Handcrafted FPN architectures. Scale variation is a well-known challenge for instance-level recognition tasks, and building pyramidal representations have been an effective way to process visual information across various image resolutions, in classical computer vision applications [2,1,15,29], and also in deep learning based approaches [25,3,19,16,13,24,39,40].

SSD [25] and MS-CNN [3] utilize multi-level feature maps to make predictions, but no aggregation is performed across different feature levels. FPN [19] is the current leading paradigm for learning feature representation of different levels through the top-down pathway and lateral connections. Similarly, RON [16] introduces reverse connections to pass information from high-level to low-level features. Although MSDNet [13] is not designed for FPN, it maintains coarse and fine level features throughout the network with a two-dimensional multi-scale network architecture. HRNet [33] also maintains high-resolution representations through the whole backbone feedforward process. PANet [24] extends FPN by introducing an extra bottom-up pathway to boost information flow. DLA [39] further deepens the representation by nonlinear and progressive fusion. M2Det [40] employs multiple U-Net to pursue a more suitable feature representation for object detection.

In relation to these previous efforts, FPG tries to formalize a unified grid structure that could potentially generalize many of these FPN variations within a systematic multi-pathway structure.

NAS-based FPN architectures. NAS automatically searches for efficient and effective architectures on a specific task. It shows promising results on image classification [30,23,35], and is also applied to other downstream tasks [5,28,26,7,38,22]. Some methods aim at discovering better FPN architectures. NAS-FPN [7] searches the construction of merging cells, *i.e.*, how to merge features at different scales. It achieves significant accuracyf improvements with a highly complicated wiring pattern (*i.e.* architecture). Auto-FPN [38] searches the architecture of both the FPN and head. It defines a fully connected search space with various dilated convolution operations, resulting in a more lightweight solution than NAS-FPN. Unlike NAS-FPN or FPG, the pathways in Auto-FPN is fixed and the connections of different stacks are not the same, thus making it not easily scalable.

In contrast to NAS-based FPN architectures, FPG can be seen as a more unified approach to feature pyramid representations, which is simple, intuitive and easy to extend.

3 Feature Pyramid Grids

Our objective in this paper is to design a unified and general multi-pathway feature pyramid representation. We aim to use the hierarchical feature representation built internally by ConvNets and enrich it with multiple pathways and lateral connections between them, to form a regular *Feature Pyramid Grid* (FPG). The concept is illustrated in Fig. 1.

Our generic grid has a backbone pathway (§3.1) and multiple pyramid pathways (§3.2), which are fused by lateral connections (§3.3) to define FPG.

3.1 Backbone pathway

The backbone pathway can be the hierarchical feature representation of any ConvNet for image classification (*e.g.*, [17,32,34,10]). This pathway is identical to what is used as the bottom-up pathway in FPN [19]. It has feature maps of progressively smaller scales from the input image to the output. As in [10,19], feature tensors with the same scale belong to a network stage and the last feature map of each stage is denoted as C_i, where i corresponds to the stage within the backbone hierarchy. The spatial stride of feature tensors *w.r.t.* the input increases from early to deeper stages, as is common in image classification [17,32,34,10].

3.2 Pyramid pathways

The deeper backbone stages, closer to the classification layer of the network represent high-level semantics, but at low resolution, while the features in early stages are only weakly related to semantics, but, on the other hand, have high

localization accuracy due to their fine resolution. The objective of the pyramid pathways is to build fine resolution features with strong semantic information. A single pyramid pathway consecutively upsamples deeper features of lower resolution to higher resolution of early stages, aiming to propagate semantic information backwards towards the network input, in parallel to the backbone (*i.e.*, feedforward) pathway.

Multiple pyramid pathways. FPG extends this idea by having *multiple*, $p > 1$, pyramid pathways in parallel. Our intention here is to enrich the capacity of the network to build a powerful representation with fine resolution across spatial dimensions and high discriminative ability, by employing multiple pyramid pathways in parallel. A typical value is $p = 9$ parallel pathways in our experiments. We build the pyramid pathways in a *bottom-up* manner, in parallel to the backbone pathway (and the first highest resolution pyramid feature is taken from the corresponding backbone stage). Connections in pyramid pathways are denoted as *SameUp*. The presence of multiple pathways is key to the FPG concept (Fig. 1) since it allows the network to build stronger pyramid features as will be demonstrated in our experiments. To form a deep Feature Pyramid Grid, the p individual pyramid pathways are intertwined with various lateral connections introduced in the next section.

Low channel capacity. Following the efficient design of FPN [19], we aim to make the pyramid pathways lightweight by reducing their channel capacity. Concretely, the pyramid pathways use a significantly lower channel capacity than the number of channels of the final stage in the backbone pathway. The typical value is 256 in FPN. Notice that the computation cost (floating-number operations, or FLOPs) of a weight layer scales *quadratically* with its channel dimensions (*i.e.* width). Therefore, reducing the channel capacity in the pyramid pathways can make multiple pathways very computationally-effective as we will demonstrate in our experiments.

3.3 Lateral connections

The aim of lateral connections is to enrich features with multi-directional (semantic) information flow in the scale space, and allow complex hierarchical feature learning across different scales.

We are using across-scale as well as within-scale connections between adjacent pathways. In relation to this, our p parallel pyramid pathways with the lateral connections between define a Feature Pyramid Grid.

We categorize our lateral connections into 4 different categories according to their starting and ending feature stages, which are denoted as:

- Across-pathway same-stage (*AcrossSame*, $\rightarrow$)
- Across-pathway bottom-up connection (*AcrossUp*, $\nearrow$)
- Across-pathway top-down (*AcrossDown*, $\searrow$)
- Across-pathway skip connection (*AcrossSkip*, $\curvearrowright$)

We describe how these connections are implemented within the context of a concrete instantiation of FPG next.

3.4 Instantiations

Our idea of FPG is generic, and it can be instantiated with different pathway and lateral connection designs as well as implementation specifics. Here, we describe concrete instantiations of the FPG network architectures.

Backbone pathway. The backbone 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 (*i.e.* the spatial stride between stages). Taking ResNet [10] for example, we adopt the same scheme as in FPN and use the output feature map of each stage's last residual block to represent the pyramid levels, denoted as $\{C_2, C_3, C_4, C_5\}$.

Pyramid pathways. Similar to the backbone pathway, pyramid pathways represent information across scales. We follow a simple design for building these in a bottom-up manner, starting from the highest resolution stage to the lowest. The first feature map of the pathway is formed by a 1×1 lateral connection from the corresponding high-resolution backbone or pyramid stage. Then, we use sub-sampling to create each lower-level feature map in the pyramid pathway by using a 3×3 convolution width stride 2, Therefore, in each pyramid pathway, the feature hierarchy consists of multi-scale feature maps with the same spatial resolution of the individual stages as in the backbone pathway.

Lateral connections. Our lateral connections fuse between the pathways into multiple directions. We employ across-pathway lateral, bottom-up, and top-down connections between adjacent pathways, and skip connections between the first pathway and all other pathways. Concrete instantiations of the lateral connections are as follows.

- *AcrossSame*, $\rightarrow$
  These lateral connections to connect the same-level features across pathways. We use a 1×1 lateral convolution on each feature map to project the features and fuse them with the corresponding features in the adjacent pathway.
- *AcrossUp*, $\nearrow$
  In order to shorten the path from low-level features in shallow pathways to high-level features in deep pathways, we introduce direct connections to build the across-level bottom-up pathway. The low-level feature map is downsampled to half size by a 3×3 stride-2 convolution and then fused with the higher-level one.
- *AcrossDown*, $\searrow$
  Similar to our bottom-up information stream within each pathway, we aim for a top-down flow of information by incorporating *AcrossDown* connections. Firstly we upsample the high-level feature maps by a scaling factor of 2 with nearest interpolation, and then use a 3×3 convolution to make *AcrossDown* learnable. The upsampled features are fused with the low-level features.

– *AcrossSkip*, ↷
  To ease the training of such a wide feature pyramid grid, we add skip connections, *e.g.*, 1×1 convolution, between same level of the first pathway and each later pathway.

Identically as for building the p parallel pathways, we employ element-wise *Sum* as the fusion function for all lateral connections.

3.5 FPG implementation details

Given a feature hierarchy in the backbone pathway, *e.g.*, $\{C_2, C_3, C_4, C_5\}$ with strides of $\{4, 8, 16, 32\}$ respectively, as in FPN, we first use 1×1 convolutions to uniformly reduce the channel capacity by β times the width of the highest feature map in the backbone pathway (*e.g.*, 256 for a ResNet with a maximum of width of 2048 and $\beta = 1/8$), producing $\{P_2^1, P_3^1, P_4^1, P_5^1\}$. Similar to FCOS [36], we produce P_6^1 from P_5^1. Then we apply the same topology to each following pathway, until the last one $\{P_2^p, P_3^p, P_4^p, P_5^p, P_6^p\}$. We follow the standard approach of using $P_2 \sim P_6$ for Faster R-CNN and Mask R-CNN, and $P_3 \sim P_7$ for RetinaNet [20], everything else is identical across detectors.

Following [7], each convolution block in above lateral connections consists of a ReLU [17], a convolution layer, and a BatchNorm [14] layer. Those connections are not shared across different pathways. After the last pathway, we append a vanilla 3×3 convolution layer after each merged feature map to output the final feature map.

We aim for the simplest possible design of FPG. We think that adopting advanced upsampling and downsampling operators such as [37,6], separable-convolution [11,38], designing more advanced blocks or fusion strategies (*e.g.* using attention [12,7]), may further boost the system-level performance, but is not the focus of this work.

Grid contraction. Our ablation studies in §4.4 reveal that the regular design of FPG can be simplified for better computation/accuracy trade-off. This will be demonstrated in our experiments, but for now, we show a more efficient version of FPG that reduces some stages without sacrificing significant accuracy. First, there are two bottom-up streams in our design: *SameUp* and *AcrossUp*. Our ablation analysis in §4.4 reveals that removing *AcrossUp* has no significant impact and *SameUp* is sufficient to provide the information flow from low-level to high-level features which is expected to be less rich in semantic information, and therefore might require lower representation capacity. Second, we found that contracting the "lower triangle" connections for high resolution feature maps can be done without sacrificing performance. Our hypothesis is that low-level feature maps need first to be enriched from top-down propagation before benefiting from deep pathways structures. Specifically, the lower triangular part of the grid can be truncated to conserve computation while preserving accuracy. The contracted FPG architecture is illustrated in Fig. 2 and used by default in the experiments.

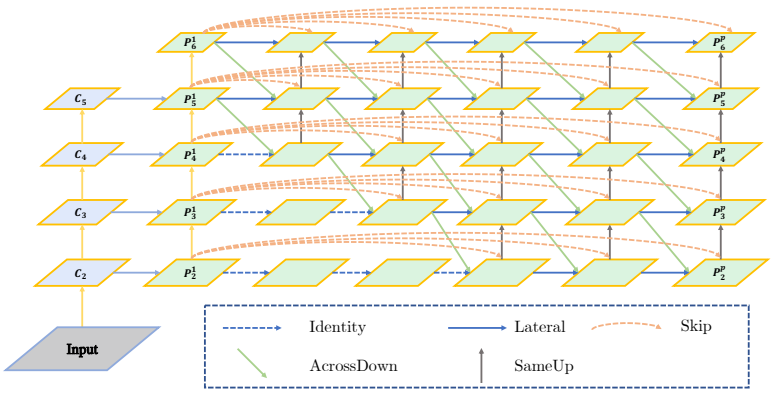

Fig. 2: The contracted FPG, in which *AcrossUp* connections are ablated and the lower triangular connections are truncated. This conserves computation while preserving accuracy (§4.4). The illustration shows 5/9 pathways of the grid.

4 Experiments

We perform experiments on standard object recognition tasks of object detection and instance segmentation. We compare to FPN [19] and NAS-FPN [7] as they represent the closest related shallow and deep feature pyramid networks, respectively. Comparisons to other other pyramid networks (PANet [24], HRNet [33]) are provided in appendix B.

4.1 Experimental Setup

Dataset and evaluation metric. We conduct experiments on the MSCOCO 2017 dataset [21]. For all tasks, models are trained on the `train` split and results are reported on `val` splits. We also show our main results on `test-dev`. Evaluation follows standard COCO mAP metrics [21].

Implementation details. For object detection and instance segmentation, we experiment with two different augmentation settings, denoted as `crop-aug` and `std-aug`. `crop-aug` is the setting introduced in NAS-FPN [7], images are firstly rescaled with a ratio randomly selected from the range [0.8, 1.2] and then cropped to a fixed size of 640×640. We utilize 8 GPUs for training and use a batch-size of 8 images on each GPU, resulting in a total batch size of 64. The training lasts for 50 epochs. The initial learning rate is set to 0.08, and then decays by 0.1 after 30 and 40 epochs. BN layers are not frozen in the Pyramid but frozen in other components. `std-aug` is the standard augmentation procedure in the original publications of the detectors used [31,9,20,4]. It resizes input images to a maximum size of 1333×800 without changing the aspect ratio. Therefore it requires more GPU memory for training. We use 16 GPUs for training and the mini-batch size is 1 image per GPU (so the total mini-batch size is 16). Models are trained for 12 epochs with an initial learning rate of 0.02 and decreased by 0.1 after 8 and 11 epochs. BN statistics are synchronized across GPUs for the Pyramid, and frozen in the ImageNet pre-trained backbone, pathways.

We use ResNet-50 for the `crop-aug` setting experiments and ResNet-50/101 for the `std-aug` setting. We use a weight decay of 0.0001 and momentum of 0.9.

For inference we exactly follow the standard settings in the original architectures [31,9,20,4,7]. For the `crop-aug` setting, the longer side of the images is resized to 640 pixels, while preserving aspect ratio [7]. For the `std-aug` setting, we resize the images to a maximum size of 1333×800 without changing the aspect ratio [9]. For object detection, the number of RoI (Region of Interest) proposals is 1000 for Faster R-CNN. The box prediction branch is applied on the proposals, followed by non-maximum suppression [8] with an IoU threshold of 0.5. We keep at most 100 bounding boxes for each image after NMS. For instance segmentation, the mask branch is run on the 100 highest scoring detection boxes [9]. As in Mask R-CNN [9], the mask output is resized to the RoI size, and binarized at a threshold of 0.5.

We denote the combination of number of pathways p and the common pathway channel width w as $p@w$. For example, using this terminology 9@256 indicates 9 pathways of channel width 256 for all pyramid layers.

We report single image floating point operations (FLOPs) as a basic *unit* of measuring computational cost agnostic to implementation and hardware specifics. We are also showing number of overall parameters of the detection systems used.

Pyramid details. We apply FPG to various detectors with the `crop-aug` and `std-aug` settings. We report the performance of 2 different architectures: *FPG (9@256)* and *FPG (9@128)*. FPG (9@256) has comparable FLOPs with NAS-FPN (7@256), and FPG (9@128) is as lightweight version of FPG that roughly matches the computational cost of the default FPN (1@256).

4.2 Main Results on Object Detection

The results of the `crop-aug` setting are shown in Table 1, for four different detection architectures. For all detection systems, *i.e.*, single-stage (RetinaNet) [20], two-stage (Faster R-CNN, Mask R-CNN) [31,9] and cascaded (Cascade R-CNN) [4], **FPG** outperforms FPN [19] by a strong margin, and also achieves better performance than NAS-FPN [7] , without relying on architecture search.

More specifically, compared to FPN, the **FPG** (9@128) improves the box AP by **+2.0**, **+2.4**, **+2.3**, and **+1.9** mAP on RetinaNet, Faster R-CNN, Mask R-CNN, and Cascade R-CNN, respectively. This result is achieved at roughly the same computational cost (FLOPs), showing that deeper and densely connected feature pyramid representations (FPG), that are thinner in width (128 *vs.* 256 channels), are significantly better than wider but shallower ones in FPN.

Compared with NAS-FPN (7@256), **FPG** (9@256) achieves +0.2, **+1.5**, **+2.3** and **+2.2** higher mAP on those four detectors, while maintaining slightly less FLOPs. Without any bells and whistles, FPG (9@256) obtains an mAP of 41.4% on Faster R-CNN and 43.8% on Cascade R-CNN using a ResNet-50 backbone, significantly better than NAS-FPN [7] under *identical* settings.

Interestingly, NAS-FPN performs comparably to FPG on RetinaNet (which it was optimized for), but achieves inferior results on the other detection systems.

Table 1: **Object detection** mAP on COCO `test-dev`. Results of different detectors on COCO with the *crop-aug* setting and inference FLOPs are reported on a single image of size 640x640. Backbone: ResNet-50 [10]. FPG achieves significant accuracy gains at similar complexity.

Detector	Pyramid	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet	FPN 1@256	95.7	37.8	37.0	55.9	39.7	16.1	41.1	51.5
	FPG 9@128	95.9	40.1	**39.0(+2.0)**	58.2	41.9	17.3	43.9	53.3
	NAS-FPN 7@256	138.8	59.8	39.8	58.5	42.6	17.6	44.8	54.4
	FPG 9@256	136.0	72.5	**40.0**(+0.2)	59.1	42.9	18.2	45.0	54.7
Faster R-CNN	FPN 1@256	91.4	41.5	37.6	58.4	40.7	18.4	40.7	50.8
	FPG 9@128	99.1	42.1	**40.0(+2.4)**	59.9	43.5	20.0	43.6	53.8
	NAS-FPN 7@256	265.3	68.2	39.9	58.8	43.3	18.8	43.8	54.4
	FPG 9@256	254.1	79.8	**41.4(+1.5)**	61.4	45.1	21.5	44.8	54.8
Mask R-CNN	FPN 1@256	159.9	44.2	38.6	59.2	41.9	18.7	41.4	52.4
	FPG 9@128	161.8	44.4	**40.9(+2.3)**	60.5	44.6	20.9	44.4	54.6
	NAS-FPN 7@256	333.8	70.8	40.1	57.9	44.3	19.0	45.7	58.1
	FPG 9@256	322.6	82.4	**42.4(+2.3)**	62.1	46.3	22.5	45.8	56.0
Cascade R-CNN	FPN 1@256	119.1	69.2	40.6	58.5	43.9	19.5	43.4	55.5
	FPG 9@128	113.9	56.9	**42.5(+1.9)**	60.0	46.0	21.4	45.9	57.3
	NAS-FPN 7@256	292.9	95.8	41.6	58.3	45.1	19.1	45.8	57.3
	FPG 9@256	281.8	107.4	**43.8(+2.2)**	61.5	47.6	23.2	47.2	58.2

As we observe higher gains over NAS-FPN in multi-stage detectors, this suggests that the NAS-FPN architecture found for the single-stage detection architecture (*i.e.* RetinaNet) might not generalize well to multi-stage detectors. Our systematic multi-pathway approach in FPG exhibits good generalization across all detection systems with strong gains over NAS-FPN for Faster R-CNN, Mask R-CNN, and Cascade R-CNN, even though it does not benefit from architecture search. The results of the `std-aug` setting are shown in Table 2 and show slightly lower gains, but are consistent with our findings for the `crop-aug` setting.

Table 2: **Object detection** mAP on COCO `test-dev`. Results of different detectors on COCO with the *std-aug* setting and inference FLOPs are reported on a single image of size 1280x832.

Detector	Pyramid	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN R-50	FPN 1@256	214.8	41.5	36.6	58.8	39.6	21.6	39.8	45.0
	FPG 9@128	245.0	42.1	**38.0(+1.4)**	59.4	41.2	22.1	40.7	46.4
	NAS-FPN 7@256	666.9	68.2	39.0	59.5	42.4	22.4	42.6	47.8
	FPG 9@256	637.8	79.8	**39.2**(+0.2)	60.8	42.7	22.7	41.9	48.4
Faster R-CNN R-101	FPN 1@256	294.0	60.5	38.8	60.9	42.3	22.3	42.2	48.6
	FPG 9@128	324.2	61.1	**39.5**(+0.7)	61.0	43.0	22.9	42.4	49.2
	NAS-FPN 7@256	746.0	87.2	40.3	61.2	43.8	23.1	43.9	50.1
	FPG 9@256	716.9	98.8	**40.6**(+0.3)	62.2	44.3	23.4	43.5	50.6
Mask R-CNN R-50	FPN 1@256	283.4	44.2	37.4	59.3	40.7	22.0	40.6	46.3
	FPG 9@128	307.8	44.5	**39.0(+1.6)**	59.9	42.4	22.8	41.8	48.4
	NAS-FPN 7@256	735.4	70.8	39.6	59.8	43.3	22.8	42.7	48.4
	FPG 9@256	706.3	82.4	**40.3**(+0.7)	61.2	44.2	23.7	42.8	49.7
Mask R-CNN R-101	FPN 1@256	362.5	63.2	39.7	61.6	43.2	23.0	43.2	49.7
	FPG 9@128	386.9	63.5	**40.5**(+0.7)	61.5	44.3	23.5	43.6	50.2
	NAS-FPN 7@256	814.5	89.8	40.5	60.8	44.2	23.4	43.7	50.2
	FPG 9@256	785.4	101.4	**41.6(+1.1)**	62.7	45.5	24.1	44.5	51.6

Efficiency *vs.* accuracy trade-off. Feature pyramid architectures allow to easily configure the model capacity. By adjusting the number of pyramid pathways p (depth) and the pathway width w we can trade off the efficiency with accuracy and obtain a set of FPG networks, from lightweight to heavy computational cost. We apply the same principle to FPN [19] and NAS-FPN [7], which also uses this strategy to stack capacity, and investigate the compute/accuracy trade-off.

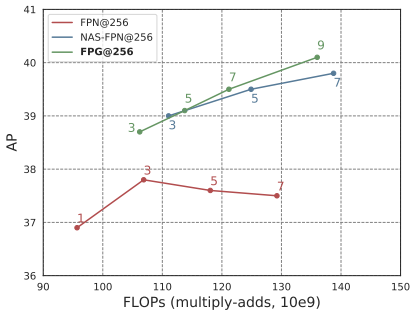

Fig. 3: The efficiency-accuracy trade-off by increasing number of pathways. Extending FPN does not work beyond 3. Detector: RetinaNet (which NAS-FPN is searched on). Backbone: ResNet-50.

Figure 3 shows the comparison on RetinaNet, which NAS-FPN was optimized for. First we notice, that extending FPN from one to many pathways does not succeed. We observe that for extending FPN with more than 3 top-down pathways accuracy stops increasing and it will instead decrease. This verifies that a simple extension of FPN to multiple top-down pathways does not lead to similar performance as we can achieve with FPG. By changing the pyramid depth, we observe that increasing the FPG pyramid pathways or NAS-FPN stacks is beneficial in terms of accuracy (vertical axis, AP). Overall, FPG achieves a better trade-off than FPN and NAS-FPN. For example, FPG (9@256) achieves higher accuracy than NAS-FPN (7@256) with fewer FLOPs.

Next we investigate the Faster R-CNN detector and vary the number of pyramid pathways p (depth) *and* the pathway width w for studying the computation/accuracy trade-off.

Figure 4 shows the effects of multiple (*i.e.* 3, 5, 7, 9) FPG pyramid pathways, or NAS-FPN stacks, as well as varying the channel width (128, 256). For both NAS-FPN and FPG, adopting a larger channel width of 256 improves the accuracy (vertical axis) while resulting in higher FLOPs (horizontal axis). We observe that FPG achieves significantly better efficiency-accuracy trade-off for Faster R-CNN detectors for which NAS-FPN was not searched (it was optimized on RetinaNet), illustrating better generalization of FPG across detectors.

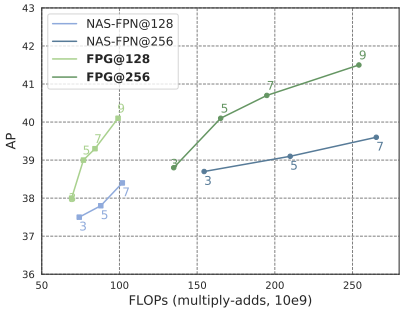

Fig. 4: Efficiency-accuracy trade-off. FPG shows consistent improvement over NAS-FPN for varying pathway width (128/256) and depth (3, 5, 7, 9). The results are under the `crop-aug` setting and show the performance on COCO *val2017*. Detector: Faster R-CNN. Backbone: ResNet-50.

Table 3: **Instance segmentation** *mask* AP on COCO `val2017`. FPG provides significant improvements over the FPN and NAS-FPN variants. Backbone: R-50.

Pyramid	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN	34.5	55.2	36.8	13.4	37.2	54.0
NAS-FPN	35.6	55.2	38.5	13.1	39.3	56.9
FPG	**37.2**	**58.4**	**39.8**	**15.9**	**40.3**	**57.0**

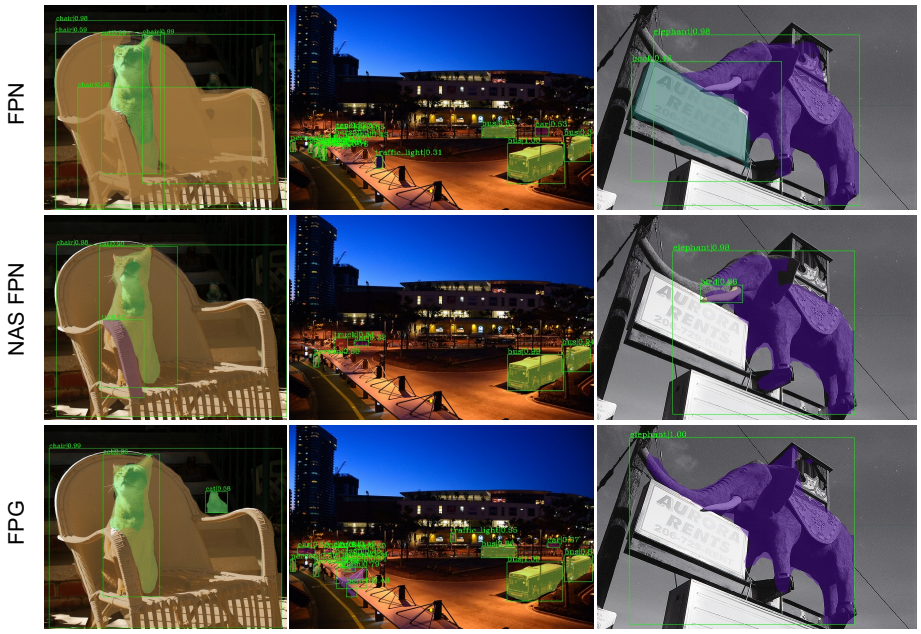

Fig. 5: **Example results for Instance Segmentation** with Mask R-CNN [9] using ResNet-50 [10] with FPN [19], NAS-FPN [7] and our FPG. Note how FPG is able to produce correct mask predictions for small-scale objects and has fewer misclassifications.

4.3 Main Results on Instance Segmentation

Here, we compare the instance segmentation results of Mask R-CNN in Table 3. The setting is the same as in Table 1, but for mask- instead of box-level prediction. **FPG** (9@256) achieves **+2.7** higher mask AP than FPN and **+1.6** higher mask AP than NAS-FPN (7@256), demonstrating generalization of FPG across different tasks. In general, our results show that a systematically designed feature pyramid grid can rival (RetinaNet) or even surpass (Faster R-CNN, Mask R-CNN and Cascade R-CNN) neural architecture search based optimization. We show qualitative results in Fig. 5, where we compare our FPG with FPN and NAS-FPN. The visualizations show that FPN and NAS-FPN are challenged by misclassification of overlapping instances, as well as small-scale objects. More examples are available in appendix A.

Table 4: **Ablations**. We study the effectiveness of each component of FPG on `val2019` and report *box* AP. Last row: Our default, contracted (Cont) instantiation.

AD	AU	SU	AS	Cont	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
✓	✓	✓	✓		173.3	104.5	**40.1**	59.0	43.0	19.9	45.6	56.7
	✓	✓	✓		128.1	83.3	35.5	52.9	38.2	14.0	39.7	53.7
✓		✓	✓		162.0	83.3	40.1	59.0	42.9	19.6	46.1	56.4
✓	✓		✓		162.0	83.3	39.6	58.6	42.3	18.7	45.4	55.9
✓	✓	✓			162.2	101.5	39.5	58.4	42.2	18.6	45.3	56.6
✓		✓	✓	✓	136.0	72.5	**40.1**	59.2	42.7	19.4	45.7	57.1

4.4 Ablation Study

We perform a thorough study of the design of FPG on COCO `val2017`, and explore different implementations of lateral connections within the grid. Our ablation experiments are conducted on RetinaNet with the `crop-aug` setting.

Component Analysis. Firstly, we investigate the necessity of pyramid pathways and lateral connections. Starting from a complete version of FPG with all connections and pathways, we remove each component respectively to see the effects. From Table 4 we see that *AcrossDown* (AD) is essential for FPG, since it is the only connection that contribute to top-down pathways. Removing these connections leads to a $-$**4.6** point mAP decrease.

On the contrary, *AcrossUp* (AU) appears to be redundant, which only adds to more FLOPs and Parameters but does not improve the performance.

Next, the connections *SameUp* (SU) and *AcrossSkip* (AS) are around equally beneficial, with a less severe impact on accuracy, as ignoring each of them results in a -0.5 mAP and -0.6 mAP decrement, respectively.

Finally, our grid contraction (Cont in Table 4) which truncates the lower-triangle low-level feature maps of the first 3 pathways (described in §3.5 and illustrated in Fig. 2) significantly reduces FLOPs and parameters, while maintaining the same level of performance. This shows that the lower, large-resolution features can use more shallow lateral structure, without sacrificing performance. Our hypothesis is that low-level feature maps need first to be enriched by top-down propagation before expanding into a deeper high-resolution pathway structure.

Table 5: Comparison of different designs of *SameUp*. **Bold**: Default.

	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
AvgPool	128.1	54.8	39.0	58.4	41.8	19.1	44.4	54.7
MaxPool	128.1	54.8	39.6	58.8	42.6	18.8	45.3	55.9
Conv	136.0	72.5	**40.1**	59.2	42.7	19.4	45.7	57.1

SameUp (↑). Table 5 shows the ablation results of the SameUp connection in the pyramid pathway. We compare three commonly used downsampling methods: average pooling, max pooling and 3×3 convolution with a stride of 2. Max pooling outperforms average pooling by $+0.6$ mAP which is further improved by using Conv ($+0.5$ mAP).

Table 6: Comparison of different designs of *AcrossSkip*. **Bold**: Default.

	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
identity	133.0	70.2	39.5	58.1	42.3	19.1	45.3	56.2
k1	136.0	72.5	**40.1**	59.2	42.7	19.4	45.7	57.1

AcrossSkip (◠). Skip connection ease the training of deeper pyramid structures by propagating information across a direct connection path. We compare two lightweight designs, an identity connection and 1×1 convolutional projections. As shown in Table 6, 1×1 convolution (k1) outperforms identity connection (identity) by +0.6 mAP with only marginal extra cost.

Table 7: Comparison of different designs of *AcrossDown*. **Bold**: Default.

	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
intp	109.3	57.2	31.9	49.6	34.0	14.1	36.3	45.8
intp + k1	112.3	58.9	39.2	58.0	42.1	18.5	44.6	55.8
intp + k3	136.0	72.5	**40.1**	59.2	42.7	19.4	45.7	57.1

AcrossDown (↘). Finally, we ablate the structure of the across-pathway top-down connections. The simplest implementation is nearest interpolation (intp), as used in FPN. Hypothesizing that a naïve interpolation may not be enough to build strong top-down pathways, we ablate either a 1×1 (k1) or 3×3 (k3) convolution to improve the FPGs capacity to project features for downsampling.

Table 7 shows the result in comparison. The accuracy is as low as 31.9 mAP with direct interpolation (intp). Adding an additional 1×1 convolution (k1) improves it by **+7.3** mAP and adopting a larger kernel size (k3) leads to a further +0.9 mAP improvement. Suggesting that a convolutional layer after interpolation, that can adapt the features and re-align the receptive field for further processing, is critical for the implementation of FPG and to achieve good performance.

5 Conclusion

This paper has presented Feature Pyramid Grids (FPG), a deep multi-pathway feature pyramid network, that represents the feature scale-space as a regular grid of parallel pyramid pathways. The pathways are intertwined by multi-directional lateral connections, forming a unified grid of feature pyramids. On instance detection and segmentation tasks, FPG provides significant improvements over both FPN and NAS-FPN with advantageous accuracy to computation trade-off. Given its unified and intuitive nature, we hope that FPG can serve as a strong component for future research and applications in instance-level recognition.

Acknowledgments. We are grateful for discussions with Kaiming He and Ross Girshick.

Appendix

A Qualitative Results

As mentioned in the main paper, we show more qualitative results for comparing FPN[19], NAS-FPN [7] and FPG here. In Fig. 6, we observe that FPG is more accurate for predicting small-scale objects and partly occluded objects. For example, in the left column of Fig. 6, it is seen that FPG is able to correctly predict masks for partly-occluded people who are spectating cross-country skiing, or in the second column of Fig. 6, there are correct FPG predictions of 'bench' instances in the background, while *e.g.* FPN misclassifies these as 'car' and NAS-FPN does not detect them. We hypothesize that this is due to the deep feature pyramid representation of FPG which allows the network to build strong features for classifying small-scale objects in high-resolution features.

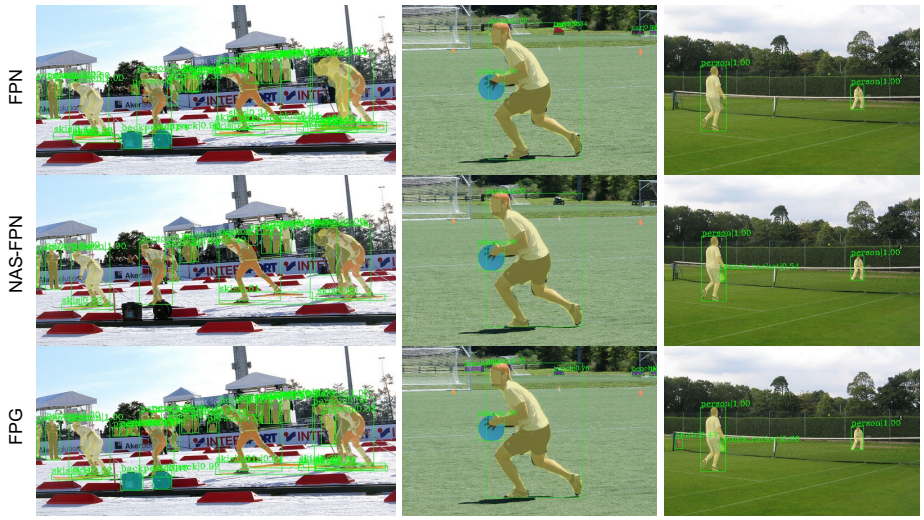

Fig. 6: **More examples for instance segmentation** with Mask R-CNN [9] using ResNet-50 [10] with FPN [19], NAS-FPN [7] and our FPG. Note how FPG is able to produce correct mask predictions for small-scale objects and has fewer misclassifications. Please view electronically, with zoom.

B Comparison with other pyramid networks

As referenced in the main paper, this section compares FPG 9@128 with related FPN structures and backbones from the literature: PANet [24] and HRNet [33] on both Faster R-CNN and Mask R-CNN detectors. All experiments are conducted with the `crop-aug` setting, which is described in the implementation details, §4.1.

Table A.1: **Object detection** mAP based on Faster R-CNN and Mask R-CNN with different pyramids.

Detector	Backbone	Pyramid	FLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-50	PA-FPN [24]	117.1	52.2	37.7	58.4	40.8	18.4	40.6	50.8
	ResNet-50	**FPG**	99.1	42.1	**40.0(+2.3)**	59.9	43.5	20.0	43.6	53.8
	HRNet-W18	HR-FPN [33]	83.6	27.5	37.5	57.7	40.9	19.4	39.7	49.8
	HRNet-W18	**FPG**	91.2	28.0	**39.4(+1.9)**	59.3	42.9	20.9	41.8	51.4
Mask R-CNN	ResNet-50	PA-FPN [24]	185.6	54.8	38.2	58.5	41.5	17.8	41.2	52.7
	ResNet-50	**FPG**	161.8	44.4	**40.9(+2.7)**	60.5	44.6	20.9	44.4	54.6
	HRNet-W18	HR-FPN [33]	152.1	30.1	38.4	58.4	41.8	19.6	40.7	50.7
	HRNet-W18	**FPG**	153.9	30.3	**40.3(+1.9)**	59.8	44.0	21.5	42.9	52.4

PANet [24] extends FPN with a path-aggregation pyramid structure (PA-FPN), and HRNet [33] is a newly designed backbone that maintains high-resolution through the whole feedforward process. It achives better performance than ResNet backbones [10] in several recognition tasks. Our FPG is aimed at better pyramidal feature representation and therefore could be complementary to HRNet backbone, if used instead of the pyramid in HRNet (HR-FPN).

Results are shown in Table A.1. We first compare to the pyramid of PANet [24]. The table shows that **FPG** achieves **+2.3** and **+2.7** higher mAP than PA-FPN [24] on Faster R-CNN and Mask R-CNN, respectively, while being lighter in terms of Floating Point Operations (FLOPs) and parameters. This result is achieved under *identical settings*, by just changing the feature pyramids of the detectors.

We note the original PANet publication [24] reports higher performance by introducing extra components other than PA-FPN, such as adaptive pooling, fully connected fusion, synchronized BN in the backbone, and heavier heads than the original ones used in R-CNN variants. These improvements to the detection architectures are orthogonal to the FPN structure and expected to be complementary. For direct comparison, we only compare FPG to PA-FPN, *i.e.*, the path aggregation feature pyramid structure, holding everything else constant.

Second, we evaluate replacing the feature pyramid used in HRNet [33] with FPG (or equivalently, changing the backbone of FPG from ResNet to HRNet) in Table A.1. HRNet-W18 + **FPG** improves box AP by **+1.9** over HRNet-W18 + HR-FPN on both Faster R-CNN and Mask R-CNN, showing complementary of FPG with the underlying backbones used for detection and superiority to the default HR-FPN under similar FLOPs and parameters.

References

1. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on communications **31**(4), 532–540 (1983) 1, 3
2. Burt, P.J., Hong, T.H., Rosenfeld, A.: Segmentation and estimation of image region properties through cooperative hierarchial computation. IEEE Transactions on Systems, Man, and Cybernetics **11**(12), 802–809 (1981) 1, 3
3. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: Proc. ECCV. Springer (2016) 3
4. Cai, Z., Vasconcelos, N.: Cascade R-CNN: Delving into high quality object detection. In: Proc. CVPR (2018) 3, 8, 9
5. Chen, Y., Yang, T., Zhang, X., Meng, G., Pan, C., Sun, J.: Detnas: Neural architecture search on object detection. arXiv preprint arXiv:1903.10979 (2019) 4
6. Gao, Z., Wang, L., Wu, G.: Lip: Local importance-based pooling. In: Proc. ICCV (2019) 7
7. Ghiasi, G., Lin, T.Y., Le, Q.V.: NAS-FPN: Learning scalable feature pyramid architecture for object detection. In: Proc. CVPR (2019) 1, 2, 4, 7, 8, 9, 11, 12, 15
8. Girshick, R.: Fast R-CNN. In: Proc. ICCV (2015) 9
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proc. ICCV (2017) 3, 8, 9, 12, 15
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016) 4, 6, 10, 12, 15, 16
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 7
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. CVPR (2018) 7
13. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844 (2017) 3
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. ICML (2015) 7
15. Koenderink, J.J.: The structure of images. Biological cybernetics **50**(5), 363–370 (1984) 1, 3
16. Kong, T., Sun, F., Yao, A., Liu, H., Lu, M., Chen, Y.: Ron: Reverse connection with objectness prior networks for object detection. In: Proc. CVPR (2017) 3
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012) 4, 7
18. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation (1989) 1
19. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proc. CVPR (2017) 1, 2, 3, 4, 5, 8, 9, 11, 12, 15
20. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proc. ICCV (2017) 3, 7, 8, 9
21. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proc. ECCV (2014) 8

22. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proc. CVPR (2019) 4
23. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018) 4
24. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proc. CVPR (2018) 3, 8, 15, 16
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Proc. ECCV. Springer (2016) 3
26. Liu, Z., Zheng, T., Xu, G., Yang, Z., Liu, H., Cai, D.: Training-time-friendly network for real-time object detection. arXiv preprint arXiv:1909.00700 (2019) 4
27. Mallat, S.G.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis & Machine Intelligence (7), 674–693 (1989) 1
28. Peng, J., Sun, M., Zhang, Z., Tan, T., Yan, J.: Efficient neural architecture transformation searchin channel-level for object detection. arXiv preprint arXiv:1909.02293 (2019) 4
29. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on pattern analysis and machine intelligence **12**(7), 629–639 (1990) 1, 3
30. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33 (2019) 4
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015) 3, 8, 9
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. ICLR (2015) 4
33. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: Proc. CVPR (2019) 3, 8, 15, 16
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. CVPR (2015) 4
35. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: Proc. ICML (2019) 4
36. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proc. ICCV (2019) 7
37. Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D.: Carafe: Content-aware reassembly of features. In: Proc. ICCV (2019) 7
38. Xu, H., Yao, L., Zhang, W., Liang, X., Li, Z.: Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In: Proc. ICCV (2019) 1, 4, 7
39. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proc. CVPR (2018) 3
40. Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H.: M2det: A single-shot object detector based on multi-level feature pyramid network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33 (2019) 3