Feature Pyramid Transformer

Dong Zhang[1] Hanwang Zhang[2] Jinhui Tang[1⋆] Meng Wang[3]
Xiansheng Hua[4] Qianru Sun[5]
{dongzhang, jinhuitang}@njust.edu.cn hanwangzhang@ntu.edu.sg eric.mengwang@gmail.com
xiansheng.hxs@alibaba-inc.com qianrusun@smu.edu.sg

[1]School of Computer Science and Engineering, Nanjing University of Science and Technology
[2]Nanyang Technological University [3]Hefei University of Technology
[4]Damo Academy, Alibaba Group [5]Singapore Management University

Abstract. Feature interactions across space and scales underpin modern visual recognition systems because they introduce beneficial visual contexts. Conventionally, spatial contexts are passively hidden in the CNN's increasing receptive fields or actively encoded by non-local convolution. Yet, the non-local spatial interactions are not across scales, and thus they fail to capture the non-local contexts of objects (or parts) residing in different scales. To this end, we propose a fully active feature interaction across both space and scales, called Feature Pyramid Transformer (FPT). It transforms any feature pyramid into another feature pyramid of the same size but with richer contexts, by using three specially designed transformers in self-level, top-down, and bottom-up interaction fashion. FPT serves as a generic visual backbone with fair computational overhead. We conduct extensive experiments in both instance-level (*i.e.*, object detection and instance segmentation) and pixel-level segmentation tasks, using various backbones and head networks, and observe consistent improvement over all the baselines and the state-of-the-art methods[1].

Keywords: Feature pyramid; Visual context; Transformer; Object detection; Instance segmentation; Semantic segmentation

1 Introduction

Modern visual recognition systems stand in context. Thanks to the hierarchical structure of Convolutional Neural Network (CNN), as illustrated in Fig. 1 (a), contexts are encoded in the gradually larger receptive fields (the green dashed rectangles) by pooling [14,19], stride [30] or dilated convolution [37]. Therefore, the prediction from the last feature map is essentially based on the rich contexts — even though there is only one "feature pixel" for a small object, *e.g.*, `mouse`, its recognition will be still possible, due to the perception of larger contexts, *e.g.*, `table` and `computer` [11,29].

Scale also matters — the `mouse` recognition deserves more feature pixels, not only the ones from the last feature map, which easily overlooks small objects. A

⋆ Corresponding author.
[1] Code is open-sourced at https://github.com/ZHANGDONG-NJUST

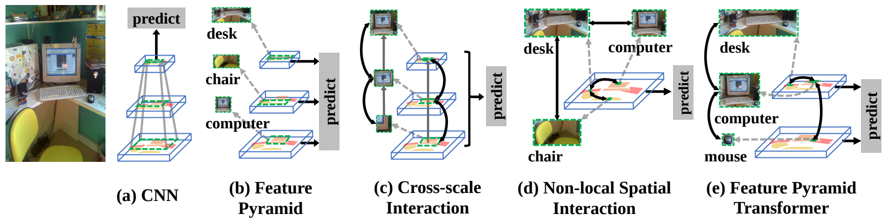

Fig. 1. The evolution of feature interaction across space and scale in feature pyramid for visual context. Transparent cubes: feature maps. Shaded `predict`: task-specific head networks. The proposed Feature Pyramid Transformer is inspired by the evolution.

conventional solution is to pile an *image pyramid* for the same image [1], where the higher/lower levels are images of lower/higher resolutions. Thus, objects of different scales are recognized in their corresponding levels, *e.g.*, `mouse` in lower levels (high resolution) and `table` in higher levels (low resolution). However, the image pyramid multiplies the time-consuming CNN forward pass as each image requires a CNN for recognition. Fortunately, CNN offers an in-network *feature pyramid* [39], *i.e.*, lower/higher-level feature maps represent higher/lower-resolution visual content without computational overhead [25,28]. As shown in Fig. 1 (b), we can recognize objects of different scales by using feature maps of different levels, *i.e.*, small objects (`computer`) are recognized in lower-levels and large objects (`chair` and `desk`) are recognized in higher-levels [16,22,24].

Sometimes the recognition — especially for *pixel-level* labeling such as semantic segmentation — requires to combine the contexts from multiple scales [5,44]. For example in Fig. 1 (c), to label pixels in the frame area of the `monitor`, perhaps the local context of the object itself from lower levels is enough; however, for the pixels in the screen area, we need to exploit both of the local context and the global context from higher levels, because the local appearance of `monitor` screen is close to `TV` screen, and we should use scene context such as `keyboard` and `mouse` to distinguish between the two types.

The spirit of the above non-local context is recently modeled in a more explicit and active fashion — as opposed to the above passive feature map pile — by using the non-local convolution [34] and self-attention [33,3]. Such spatial feature interaction is expected to capture the reciprocal co-occurring patterns of multiple objects [41,16]. As shown in Fig. 1 (d), it is more likely that there is a `computer` on the `desk` rather than on `road`, thus, the recognition of either is helpful to the other.

The tale of context and scale should continue, and it is our key motivation. In particular, we are inspired by the omission of the cross-scale interactions (Fig. 1 (c)) in the non-local spatial interactions (Fig. 1 (d)). Moreover, we believe that the non-local interaction *per se* should happen in the corresponding scales of the interacted objects (or parts), but not just in one uniform scale as in existing methods [33,34,41]. Fig. 1 (e) illustrates the expected non-local

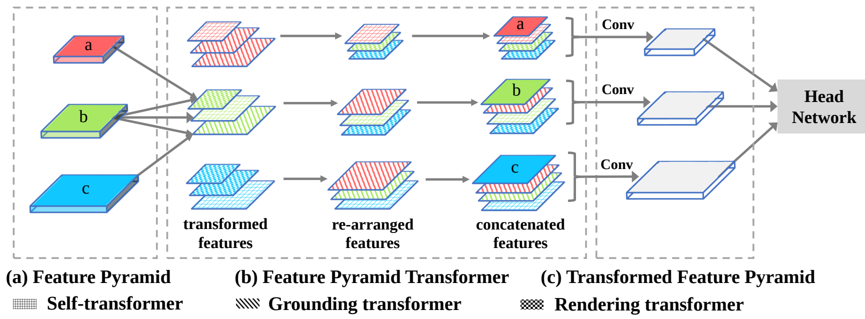

Fig. 2. Overall structure of our proposed FPT network. Different texture patterns indicate different feature transformers, and different color represents feature maps with different scales. "Conv" denotes a 3×3 convolution with the output dimension of 256. Without loss of generality, the top/bottom layer feature maps has no rendering/grounding transformer.

interactions across scales: the low-level `mouse` is interacting with the high-level `computer`, which is interacting with `desk` at the same scale.

To this end, we propose a novel feature pyramid network called **Feature Pyramid Transformer** (FPT) for visual recognition, such as instance-level (*i.e.*, object detection and instance segmentation) and pixel-level segmentation tasks. In a nutshell, as illustrated in Fig. 2, the input of FPT is a feature pyramid, and the output is a transformed one, where each level is a *richer* feature map that encodes the non-local interactions across space and scales. Then, the feature pyramid can be attached to any task-specific head network. As its name implies, FPT's interaction adopts the transformer-style [33,3]. It has the neat *query*, *key* and *value* operation (cf. Section 3.1) that is shown effective in selecting informative long-range interaction, which tailors our goal: non-local interaction at proper scales. In addition, the computation overhead (cf. Section 4.1) can be alleviated by using TPUs like any other transformer models [18].

Our technical contributions, as illustrated in the FPT breakdown in Fig. 2, are the designs of three transformers: 1) **Self-Transformer** (ST). It is based on the classic non-local interaction within the same level feature map [34], and the output has the same scale as its input. 2) **Grounding Transformer** (GT). It is in a top-down fashion, and the output has the same scale as the lower-level feature map. Intuitively, we ground the "concept" of the higher-level feature maps to the "pixels" of the lower-level ones. In particular, as it is unnecessary to use the global information to segment objects, and the context within a local region is empirically more informative, we also design a *locality-constrained* GT for both efficiency and accuracy of semantic segmentation. 3) **Rendering Transformer** (RT). It is in a bottom-up fashion, and the output has the same scale as the higher-level feature map. Intuitively, we render the higher-level "concept" with the visual attributes of the lower-level "pixels". Note that this is a *local* interaction as it is meaningless to render an "object" with the "pixels" of another distant one. The transformed feature maps of each level (the red, blue and green) are re-arranged to its corresponding map size and then concatenated with the original map, before feeding into the conv-layer that resize them to the original "thickness".

Extensive experiments show that FPT can greatly improve conventional detection/segmentation pipelines by the following absolute gains: 1) 8.5% box-AP for object detection and 6.0% mask-AP for instance segmentation over baseline on the MS-COCO [23] *test-dev*; 2) for semantic segmentation, 1.6% and 1.2% mIoU on Cityscapes [7] and PASCAL VOC 2012 [9] test sets, respectively; 1.7% and 2.0% mIoU on ADE20K [45] and LIP [12] validation sets, respectively.

2 Related Work

FPT is generic to apply in a wide range of computer vision tasks. This paper focuses on two instance-level tasks: object detection, instance segmentation, and one pixel-level task: semantic segmentation. Object detection aims to predict a bounding box for each object and then assigns the bounding box a class label [29], while instance segmentation is additionally required to predict a pixel-level mask of the object [13]. Semantic segmentation aims to predict a class label to each pixel of the image [26].

Feature pyramid. The in-network feature pyramid (*i.e.*, the Bottom-up Feature Pyramid (BFP) [22]) is one of the most commonly used methods, and has been shown useful for boosting object detection [25], instance segmentation [24] and semantic segmentation [43]. Another popular method of constructing feature pyramid uses feature maps of the scale while processing the maps through pyramidal pooling or dilated/atrous convolutions. For example, atrous spatial pyramid pooling [5] and pyramid pooling module [14,44] leverages output feature maps of the last convolution layer in the CNN backbone to build the four-level feature pyramid, in which different levels have the same resolution but different information granularities. Our approach is based on the existing BFP (for the instance-level) and unscathed feature pyramid [27] (for the pixel-level). Our contribution is the novel feature interaction approach.

Feature interaction. An intuitive approach to the cross-scale feature interaction is gradually summing the multi-scale feature maps, such as Feature Pyramid Network (FPN) [22] and Path Aggregation Network (PANet) [24]. In particular, both FPN and PANet are based on BFP, where FPN adds a top-down path to propagate semantic information into low-level feature maps, and PANet adds a bottom-up path augmentation on the basis of FPN. Another approach is to concatenate multi-scale feature maps along the channel dimension. The specific examples for semantic segmentation are DeepLab [4] and pyramid scene parsing network [44]. Besides, a more recent work proposed the ZigZagNet [20] which exploits the addition and convolution to enhance the cross-scale feature interaction. Particularly, for the within-scale feature interaction, some recent works exploited non-local operation [34] and self-attention [33] to capture the co-occurrent object features in the same scene. Their models were evaluated in a wide range of visual tasks [16,38,41,48]. However, we argue that the non-local interaction performed in just one uniform scale feature map is not enough to represent the contexts. In this work, we aim to conduct the non-local interaction *per se* in the corresponding scales of the interacted objects (or parts).

3 Feature Pyramid Transformer

Given an input image, we can formally extract a feature pyramid, where the fine-/coarse-grained feature maps are in low/high levels, respectively. Without loss of generality, we express a low-level fine-grained feature map as $\mathbf{X}^f$ and a high-level coarse-grained feature map as $\mathbf{X}^c$. **Feature Pyramid Transformer** (FPT) enables features to interact across space and scales. It specifically includes three transformers: self-transformer (cf. Section 3.2), grounding transformer (cf. Section 3.3) and rendering transformer (cf. Section 3.4). The transformed feature pyramid is in the same size but with richer contexts than the original.

3.1 Non-Local Interaction Revisited

A typical non-local interaction [34] operates on *queries*(Q), *keys*(K) and *values*(V) within a single feature map $\mathbf{X}$, and the output is the transformed version $\tilde{\mathbf{X}}$ with the same scale as $\mathbf{X}$. This non-local interaction is formulated as:

$$\begin{aligned}
\textbf{Input:}&\quad \mathbf{q}_i, \mathbf{k}_j, \mathbf{v}_j \\
\textbf{Similarity:}&\quad s_{i,j} = F_{sim}(\mathbf{q}_i, \mathbf{k}_j) \\
\textbf{Weight:}&\quad w_{i,j} = F_{nom}(s_{i,j}) \\
\textbf{Output:}&\quad \tilde{\mathbf{X}}_i = F_{mul}(w_{i,j}, \mathbf{v}_j),
\end{aligned} \tag{1}$$

where $\mathbf{q}_i = f_q(\mathbf{X}_i) \in \mathrm{Q}$ is the i^{th} *query*; $\mathbf{k}_j = f_k(\mathbf{X}_j) \in \mathrm{K}$ and $\mathbf{v}_j = f_v(\mathbf{X}_j) \in \mathrm{V}$ are the j^{th} *key*/*value* pair; $f_q(\cdot)$, $f_k(\cdot)$ and $f_v(\cdot)$ denote the *query*, *key* and *value* transformer functions [3,33], respectively. $\mathbf{X}_i$ and $\mathbf{X}_j$ are the i^{th} and j^{th} feature positions in $\mathbf{X}$, respectively. F_{sim} is the similarity function (default as *dot product*); F_{nom} is the normalizing function (default as *softmax*); F_{mul} is the weight aggregation function (default as *matrix multiplication*); and $\tilde{\mathbf{X}}_i$ is the i^{th} feature position in the transformed feature map $\tilde{\mathbf{X}}$.

3.2 Self-Transformer

Self-Transformer (ST) aims to capture the co-occurring object features on one feature map. As illustrated in Fig. 3 (a), ST is a modified non-local interaction [34] and the output feature map $\hat{\mathbf{X}}$ has the same scale as its input $\mathbf{X}$. A main difference with [33,34] is that we deploy the Mixture of Softmaxes (MoS) [35] as the normalizing function F_{mos}, which turns out to be more effective than the standard *Softmax* [41] on images. Specifically, we first divide $\mathbf{q}_i$ and $\mathbf{k}_j$ into $\mathcal{N}$

parts. Then, we calculate a similarity score $s^n_{i,j}$ for every pair, *i.e.*, $\mathbf{q}_{i,n}$, $\mathbf{k}_{j,n}$, using F_{sim}. The MoS-based normalizing function F_{mos} is as follows:

$$F_{mos}(s^n_{i,j}) = \sum_{n=1}^{\mathcal{N}} \pi_n \frac{\exp(s^n_{i,j})}{\sum_j \exp(s^n_{i,j})}, \tag{2}$$

where $s^n_{i,j}$ is the similarity score of the n^{th} part. π_n is the n^{th} aggregating weight that is equal to $\mathit{Softmax}\left(\mathbf{w}_n^T \bar{\mathbf{k}}\right)$, where $\mathbf{w}_n$ is a learnable linear vector for normalization and $\bar{\mathbf{k}}$ is the arithmetic mean of all positions of $\mathbf{k}_j$. Based on F_{mos}, we then can reformulate Eq. 1 to elaborate our proposed ST as follows:

$$\begin{aligned} \textbf{Input:} \quad & \mathbf{q}_i, \mathbf{k}_j, \mathbf{v}_j, \mathcal{N} \\ \textbf{Similarity:} \quad & s^n_{i,j} = F_{sim}(\mathbf{q}_{i,n}, \mathbf{k}_{j,n}) \\ \textbf{Weight:} \quad & w_{i,j} = F_{mos}(s^n_{i,j}) \\ \textbf{Output:} \quad & \hat{\mathbf{X}}_i = F_{mul}(w_{i,j}, \mathbf{v}_j), \end{aligned} \tag{3}$$

where $\hat{\mathbf{X}}_i$ is the i^{th} transformed feature position in $\hat{\mathbf{X}}$.

3.3 Grounding Transformer

Grounding Transformer (GT) can be categorized as a top-down non-local interaction [34], which grounds the "concept" in the higher-level feature maps $\mathbf{X}^c$ to the "pixels" in the lower-level feature maps $\mathbf{X}^f$. The output $\hat{\mathbf{X}}^f$ has the same scale as $\mathbf{X}^f$. Generally, image features at different scales extract different semantic or contextual information or both [39,43]. Moreover, it has been empirically shown that the negative value of the *euclidean distance* F_{eud} is more effective in computing the similarity than *dot product* when the semantic information of two feature maps is different [42]. So we prefer to use F_{eud} as the similarity function, which is expressed as:

$$F_{eud}(\mathbf{q}_i, \mathbf{k}_j) = -||\mathbf{q}_i - \mathbf{k}_j||^2, \tag{4}$$

where $\mathbf{q}_i = f_q(\mathbf{X}^f_i)$ and $\mathbf{k}_j = f_k(\mathbf{X}^c_j)$; $\mathbf{X}^f_i$ is the i^{th} feature position in $\mathbf{X}^f$, and $\mathbf{X}^c_j$ is the j^{th} feature position in $\mathbf{X}^c$. We then replace the similarity function in Eq. 3 with F_{eud}, and get the formulation of the proposed GT as follows:

$$\begin{aligned} \textbf{Input:} \quad & \mathbf{q}_i, \mathbf{k}_j, \mathbf{v}_j, \mathcal{N} \\ \textbf{Similarity:} \quad & s^n_{i,j} = F_{eud}(\mathbf{q}_{i,n}, \mathbf{k}_{j,n}) \\ \textbf{Weight:} \quad & w_{i,j} = F_{mos}(s^n_{i,j}) \\ \textbf{Output:} \quad & \hat{\mathbf{X}}^f_i = F_{mul}(w_{i,j}, \mathbf{v}_j), \end{aligned} \tag{5}$$

where $\mathbf{v}_j = f_v(\mathbf{X}^c_j)$; $\hat{\mathbf{X}}^f_i$ is the i^{th} transformed feature position in $\hat{\mathbf{X}}^f$. Based on Eq. 5, each pair of $\mathbf{q}_i$ and $\mathbf{k}_j$ with a closer distance will be given a larger weight as in [33,34]. Compared to the results of *dot product*, using F_{eud} brings clear improvements in the top-down interactions[2].

[2] More details are given in *Section A* of the supplementary.

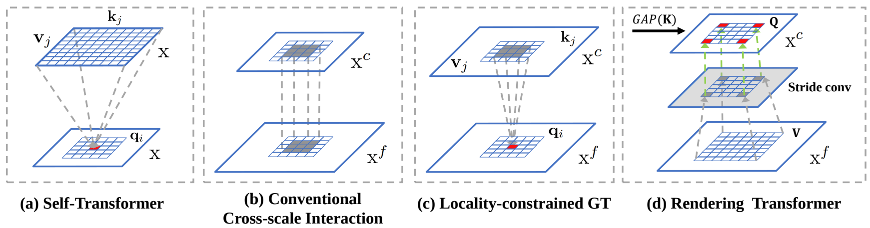

Fig. 3. Self-Transformer(ST), Conventional Cross-Scale Interaction in existing methods, Locality-constrained Grounding Transformer (GT), and Rendering Transformer. The red grid in low-level is a *query* position; grids in high-level are the *key* and the *value* positions (within a local square area in (b)); $\mathbf{Q}$ are the high-level feature maps, $\mathbf{K}$ and $\mathbf{V}$ are the low-level feature maps. Grey square is the down-sampled $\mathbf{V}$.

In feature pyramid, high-/low-level feature maps contain much global/local image information. However, for semantic segmentation by cross-scale feature interactions, it is unnecessary to use global information to segment two objects in an image. The context within a local region around the *query* position is empirically more informative. That is why the conventional cross-scale interaction (*e.g.*, summation and concatenation) is effective in existing segmentation methods [4,44]. As shown in Fig. 3 (b), they are essentially the implicit local style. However, our default GT is the global interaction.

Locality-constrained Grounding Transformer. We therefore introduce a *locality-constrained* version of GT called Locality-constrained GT (LGT) for semantic segmentation, which is an explicit local feature interaction. As illustrated in Fig. 3 (c), each $\mathbf{q}_i$ (*i.e.*, the red grid on the low-level feature map) interacts with a portion of $\mathbf{k}_j$ and $\mathbf{v}_j$ (*i.e.*, the blue grids on the high-level feature map) within the local square area where the center coordinate is the same with $\mathbf{q}_i$ and the side length is *square_size*. Particularly, for positions of $\mathbf{k}_j$ and $\mathbf{v}_j$ that exceed the index, we use 0 value instead.

3.4 Rendering Transformer

Rendering Transformer (RT) works in a bottom-up fashion, aiming to render the high-level "concept" by incorporating the visual attributes in the low-level "pixels". As illustrated in Fig. 3 (d), RT is a *local* interaction where the *local* is due to the fact that it is meaningless to render an "object" with the features or attributes from another distant one.

In our implementation, RT is not performed by pixel but the entire feature maps. Specifically, the high-level feature map is defined as $\mathbf{Q}$; the low-level feature map is defined as $\mathbf{K}$ and $\mathbf{V}$. To highlight the rendering target, the interaction between $\mathbf{Q}$ and $\mathbf{K}$ is conducted in a channel-wise attention manner [6]. $\mathbf{K}$ first computes a weight $\mathbf{w}$ for $\mathbf{Q}$ through Global Average Pooling (GAP) [21]. Then, the weighted $\mathbf{Q}$ (*i.e.*, $\mathbf{Q}_{att}$) goes through a 3×3 convolution for refinement [36]. $\mathbf{V}$ goes through a 3×3 convolution with stride to reduce the feature scale (the gray square in Fig. 3 (d)). Finally, the refined $\mathbf{Q}_{att}$ and the down-sampled $\mathbf{V}$ (*i.e.*, $\mathbf{V}_{dow}$) are summed-up, and processed by another 3×3 convolution for refinement. The proposed RT can be formulated as follows:

$$\begin{aligned}
\textbf{Input:} \quad & \mathbf{Q}, \mathbf{K}, \mathbf{V} \\
\textbf{Weight:} \quad & \mathbf{w} = GAP(\mathbf{K}) \\
\textbf{Weight Query:} \quad & \mathbf{Q}_{att} = F_{att}(\mathbf{Q}, \mathbf{w}) \\
\textbf{Down-sampled Value:} \quad & \mathbf{V}_{dow} = F_{sconv}(\mathbf{V}) \\
\textbf{Output:} \quad & \hat{\mathbf{X}}^{c} = F_{add}(F_{conv}(\mathbf{Q}_{att}), \mathbf{V}_{dow}),
\end{aligned} \tag{6}$$

where $F_{att}(\cdot)$ is an *outer product* function; $F_{sconv}(\cdot)$ is a 3×3 stride convolution, in particular, where $stride = 1$ if the scales of $\mathbf{Q}$ and $\mathbf{V}$ are equal; $F_{conv}(\cdot)$ is a 3×3 convolution for refinement; $F_{add}(\cdot)$ is the feature map summation function with a 3×3 convolution; and $\hat{\mathbf{X}}^{c}$ denotes the output feature map of RT.

3.5 Overall Architecture

We build specific FPT networks for tackling object detection [16,22], inatance segmentation [13,24], and semantic segmentation [5,44]. Each FPT network is composed of four components: a backbone for feature extraction; a feature pyramid construction module; our proposed FPT for feature transformer; and a task-specific head network. In the following, we detail the proposed architectures.

FPT for object detection and instance segmentation. We follow [22,24] to deploy the ResNet as the backbone, and pre-train it on the ImageNet [8]. BFP [22] is used as the pyramid construction module. Then the proposed FPT is applied to BFP, for which the number of divided parts of $\mathcal{N}$ is set to 2 for ST and 4 for GT[3]. Then, the transformed feature maps (by FPT) are concatenated with the original maps along the channel dimension. The concatenated maps go through a 3×3 convolution to reduce the feature dimension into 256. On the top of the output feature maps, we apply the head networks for handling specific tasks, *e.g.*, the Faster R-CNN [29] head for object detection and the Mask R-CNN [13] head for instance segmentation. To enhance the feature generalization, we apply the DropBlock [10] to each output feature map. We set the drop block size as 5 and the feature keep probability as 0.9.

FPT for semantic segmentation. We use dilated ResNet-101 [37] as the backbone (pre-trained on the ImageNet) following [5,40]. We then apply the Unscathed Feature Pyramid (UFP) as the feature pyramid construction module, which basically contains a pyramidal global convolutional network [27] with the internal kernel size of 1, 7, 15 and 31, and each scale with the output dimension of 256. Then, the proposed FPT (including LGT) is applied to UFP with the same number of divided parts $\mathcal{N}$ as in the instance-level tasks. In particular, the *square_size* of LGT is set to 5. On the top of the transformed feature pyramid, we apply the semantic segmentation head network, as in [5,41]. We also deploy the DropBlock [10] on the output feature maps with the drop block size as 3 and the feature keep probability as 0.9.

[3] More details are given in *Section B* of the supplementary.

4 Experiments

Our experiments were conducted on three interesting and challenging tasks: *i.e.*, instance-level object detection and segmentation, and pixel-level semantic segmentation. In each task, we evaluated our approach with careful ablation studies, extensive comparisons to the state-of-the-arts and representative visualizations.

4.1 Instance-Level Recognition

Dataset. Experiments on object detection and instance segmentation were conducted on MS-COCO 2017 [23] which has 80 classes and includes 115k, 5k and 20k images for training, validation and test, respectively.
Backbone. In the ablation study, ResNet-50 [15] was used as the backbone. To compare to state-of-the-arts, we also employed ResNet-101 [15], Non-local Network (NL-ResNet-101) [34], Global Context Network (GC-ResNet-101) [2] and Attention Augmented Network (AA-ResNet-101) [17] as the backbone networks.
Setting. As in [22,24], the backbone network was pre-trained on the ImageNet [8], then the whole network was fine-tuned on the training data while freezing the backbone parameters. For fair comparisons, input images were resized into 800 pixels/$1,000$ pixels for the shorter/longer edge [20].
Training details. We adopted SGD training on 8 GPUs with the Synchronized Batch Norm (SBN) [40]. Each mini-batch involved one image per GPU and 512 Region of Interest (ROI) per image. The positive-to-negative ratio was set to 1 : 3. The weight decay and momentum were set to 0.0001 and 0.9, respectively. For object detection, the learning rate was 0.05 in the first 80k iterations, and 0.005 in the remaining 20k iterations. For instance segmentation, the learning rate was 0.05 for the first 120k iterations, and 0.005 in the remaining 40k iterations. An end-to-end region proposal network was used to generate proposals, as in [34].
Comparison methods. We compared our FPT to the state-of-the-art cross-scale feature pyramid interactions including FPN [22], Bottom-up Path Aggregation (BPA) in PANet [24], and Bi-direction Feature Interaction (BFI) in ZigZagNet [20]. We also reported the experimental results of using the Augmented Head (AH) [24] and Multi-scale Training (MT) [24], where the AH specifically includes the adaptive feature pooling, fully-connected fusion, and heavier head.
Metrics. We evaluated the model performance using the standard Average Precision (AP), AP_{50}, AP_{75}, AP_S, AP_M and AP_L.
Ablation study. Our ablation study aims to (1) evaluate the performance of three individual transformers (in our FPT) and combinations, for which the base pyramid method BFP [22] is the baseline (in Table 1), and (2) investigate the effects of SBN [40] and DropBlock [10] on our FPT (in Table 2).
Comparing to the baseline. Table 1 show that three transformers bring consistent improvements over the baseline. For example, ST, GT and RT respectively brings 0.4%, 3.5% and 3.1% improvements for the bounding box AP in object detection. The improvements are higher as 0.7%, 4.0% and 3.2% for the mask AP in instance segmentation. The gain by ST is not as much as the gains by the

BFP	ST	GT	RT	AP		AP_{50}		AP_{75}		AP_S		AP_M		AP_L		Params	GFLOPs
✓	✗	✗	✗	31.6	29.9	54.1	50.7	35.9	34.7	16.1	14.2	32.5	31.6	48.8	48.5	34.6 M	172.3
✓	✓	✗	✗	32.0	30.6	54.9	51.4	36.9	35.5	16.5	15.1	34.0	32.1	49.1	49.7	55.0 M	248.2
✓	✗	✓	✗	35.1	33.9	55.2	52.4	38.1	37.7	17.4	16.9	36.3	33.3	50.3	51.7	63.9 M	265.1
✓	✗	✗	✓	34.7	33.1	55.5	52.0	37.5	37.7	17.0	15.3	36.6	34.9	52.0	52.1	39.8 M	187.9
✓	✓	✓	✗	35.7	34.6	55.7	54.1	38.3	37.9	18.0	17.4	36.5	34.0	52.1	50.5	82.5 M	322.9
✓	✓	✗	✓	35.9	34.4	56.8	55.1	38.8	38.0	19.1	17.9	37.0	34.8	53.1	52.2	61.2 M	256.7
✓	✗	✓	✓	36.9	35.1	56.6	54.5	38.2	38.5	18.8	17.7	37.7	**35.3**	54.3	53.2	69.6 M	281.6
✓	✓	✓	✓	**38.0**	**36.8**	**57.1**	**55.9**	**38.9**	**38.6**	**20.5**	**18.8**	**38.1**	**35.3**	**55.7**	**54.2**	88.2 M	346.2
improvements				↑ **6.4**	↑ **6.9**	↑ **3.0**	↑ **5.2**	↑ **3.0**	↑ **3.9**	↑ **4.4**	↑ **4.6**	↑ **5.6**	↑ **3.7**	↑ **6.9**	↑ **5.7**		

Table 1. Ablation study on MS-COCO 2017 val set [23]. "BFP" is Bottom-up Feature Pyramid [22]; "ST" is Self-Transformer; "GT" is Grounding Transformer; "RT" is Rendering Transformer. Results on the left and right of the dashed are of bounding box detection and instance segmentation.

FPT	SBN	DropBlock	AP		AP_{50}		AP_{75}		AP_S		AP_M		AP_L	
✓	✗	✗	37.2	35.9	56.0	54.3	37.7	36.9	19.0	17.2	37.7	34.8	53.1	51.3
✓	✓	✗	37.8	36.5	56.7	55.2	38.4	38.2	19.6	18.0	37.9	35.1	54.0	52.1
✓	✗	✓	37.5	36.2	56.5	54.8	38.0	37.3	19.5	17.8	37.8	35.0	53.8	51.9
✓	✓	✓	**38.0**	**36.8**	**57.1**	**55.9**	**38.9**	**38.6**	**20.5**	**18.8**	**38.1**	**35.3**	**55.7**	**54.2**

Table 2. Ablation study of SBN [40] and DropBlock [10] on the MS-COCO 2017 val set [23]. Results on the left and right of dashed lines are respectively for bounding box detection and instance segmentation.

other two transformers. An intuitive reason is that, compared to self-interaction (*i.e.*, ST), the cross-scale interactions (*i.e.*, GT and RT) capture more diverse and richer inter-object contexts to achieve better object recognition and detection performances, which is consistent with the conclusion of instance-level recognition works [46,47]. The middle blocks in Table 1 show that the combination of transformers improves the performance over individuals in most of cases. In particular, the full combination of ST, GT and RT results the best performance, *i.e.*, 38.0% bounding box AP (6.4% higher than BFP) on object detection and 36.8% mask AP (6.9% higher than BFP) on instance segmentation.

Effects of SBN and DropBlock. Table 2 shows that both SBN and DropBlock improve the model performance. Their combination yields 0.8% AP improvement for object detection, and 0.9% AP improvement for instance segmentation.

Model efficiency[4]**.** We reported the model Parameters (Params) and GFLOPs with the Mask R-CNN [13]. Adding +ST/+GT/+RT to the baseline respectively increase Params by 0.59×/0.85×/0.15× (with mask AP improvements of 0.7%/4.0%/3.2%). Correspondingly, GFLOPs are increased by 0.44×, 0.54× and 0.09×. Compared to related works [20,22,34], these are relatively fair overheads on average.

[4] More details are given in the *Section C* of the supplementary.

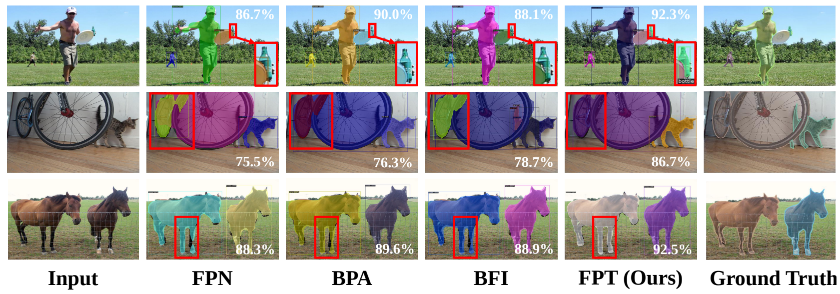

Fig. 4. Visualization results in instance segmentation. The red rectangle highlights the better predicted area of FPT. Samples are from MS-COCO 2017 validation set [23]. The value on each image represents the corresponding segmentation mIoU.

Comparing to the state-of-the-arts. Table 3 show that applying the cross-scale interaction methods, *e.g.*, FPN [22], BPA [24], BFI [20] and FPT, results consistent improvements over the baseline [22]. In particular, our FPT achieves the highest gains, *i.e.*, 8.5% AP in object detection and 6.0% mask AP in instance segmentation, with ResNet-101 [15]. Besides, the consistent improvements are also achieved on the stronger NL-, GC- and AA- ResNet-101, and validate that BFP+FPT can generalize well to stronger backbones, which make more senses in *the age of results*[5]. Two bottom blocks in Table 3 show that adding efficient training strategies such as AH, MT, and both (denoted as "[all]") to BFP+FPT yields performance boosts. For example, BFP+FPT [all] achieves a higher bounding box AP and the same mask AP, compared to the best performance of BFP+FPT (with stronger GC-ResNet-101). Besides, BFP+FPT [all] achieves the average 1.5% AP in object detection and 2.1% mask AP in instance segmentation (over BFP+BFI) using ResNet-101, which further verifies the robust plug-and-play ability of our FPT. The visualization results in instance segmentation are given in Fig. 4. Compared to other feature interaction methods, the results of FPT show more precise predictions for both small (*e.g.*, bottle) and large objects (*e.g.*, bicycle). Moreover, it shows the gracile parts in the object (*e.g.*, the horse legs) are also well predicted using our FPT.

4.2 Experiments on Pixel-Level Recognition

Dataset. Our pixel-level segmentation experiments were conducted on four benchmarks: (1) *Cityscapes* [7] contains 19 classes, and includes $2,975$, 500 and $1,525$ images for training, validation and test, respectively; (2) *ADE20K* [45] has 150 classes, and uses $20k$, $2k$, and $3k$ images for training, validation and test, respectively; (3) *LIP* [12] contains $50,462$ images with 20 classes, and includes $30,462$, $10k$ and $10k$ images for training, validation and test, respectively; (4) *PASCAL VOC 2012* [9] contains 21 classes, and includes $1,464$, $1,449$ and $1,456$ images for training, validation and test, respectively.

[5] More results are given in *Section D* of the supplementary.

Methods	Backbone	AP		AP_{50}		AP_{75}		AP_S		AP_M		AP_L	
BFP [22]	ResNet-101	33.1	32.6	53.8	51.7	34.6	33.3	12.6	11.4	35.3	34.4	49.5	48.9
	NL-ResNet-101	34.4	33.7	54.3	53.6	35.8	33.9	15.1	13.7	37.1	36.0	50.7	49.7
	GC-ResNet-101	35.0	34.2	55.8	54.1	36.5	35.3	14.8	13.9	38.6	37.3	50.9	50.5
	AA-ResNet-101	33.8	32.8	54.2	52.3	35.4	33.8	13.0	12.3	35.5	34.5	50.0	49.0
BFP+FPN [22]	ResNet-101	36.2	35.7	59.1	58.0	39.0	37.8	18.2	15.5	39.0	38.1	52.4	49.2
BFP+BPA [24]	ResNet-101	37.3	36.3	60.4	59.0	39.9	38.3	18.9	16.3	39.7	39.0	53.0	50.5
BFP+BFI [20]	ResNet-101	39.5	-	-	-	-	-	-	-	-	-	-	-
BFP+FPT	ResNet-101	41.6	38.6	60.9	58.2	44.0	43.3	23.4	19.0	41.5	39.2	53.1	50.8
	NL-ResNet-101	42.0	39.5	62.1	60.7	46.5	45.4	25.1	20.8	42.6	41.0	53.7	53.0
	GC-ResNet-101	42.5	**40.3**	62.0	61.0	46.1	45.8	**25.3**	21.1	42.7	**41.8**	53.1	52.7
	AA-ResNet-101	42.1	40.1	61.5	60.1	46.5	45.2	25.2	20.6	42.6	41.2	53.5	52.0
BFP+FPT [AH]	ResNet-101	41.1	40.0	62.0	59.9	46.6	45.5	24.2	20.5	42.1	41.0	53.3	52.5
BFP+FPT [MT]	ResNet-101	41.2	39.8	62.1	60.1	46.0	45.1	24.1	20.9	41.9	40.8	53.2	51.9
BFP+FPN [22] [all]	ResNet-101	37.9	36.3	59.6	58.8	40.1	39.1	19.5	16.7	41.0	40.3	53.5	51.1
BFP+BPA [24] [all]	ResNet-101	39.0	37.7	60.8	59.4	41.7	40.1	20.2	18.5	41.5	40.1	54.1	52.4
BFP+BFI [20] [all]	ResNet-101	40.1	38.2	61.2	60.0	42.6	42.4	21.9	19.6	42.4	40.8	54.3	52.5
BFP+FPT [all]	ResNet-101	**42.6**	**40.3**	**62.4**	**61.1**	**46.9**	**45.9**	24.9	**21.3**	**43.0**	41.2	**54.5**	**53.3**

Table 3. Experimental results on MS-COCO 2017 *test-dev* [23]. "AH" is Augmented Head, and "MT" is Multi-scale Training [24]; "all" means that both the AH and MT are used. Results on the left and right of the dashed are of bounding box detection and instance segmentation. "-" means that there is no reported result in its paper.

Backbone. We used dilated ResNet-101 [37] as the backbone as in [41].

Setting. We first pre-trained the backbone network on the ImageNet [8], then fine-tuned the whole network on the training data while fixing the parameters of backbone as in [40]. Before input, we cropped the image into 969×969 for *Cityscapes*, 573×573 for *LIP*, and 521×521 for *PASCAL VOC 2012*. Because images in ADE20K are of various sizes, we cropped the shorter-edge images to an uniform size $\{269, 369, 469, 569\}$ as that in [38].

Training details. We followed [38] to use the learning rate scheduling $lr = baselr \times (1 - \frac{iter}{total_{iter}})^{power}$. On *Cityscapes*, *LIP* and *PASCAL VOC 2012*, the base learning rate was 0.01, and the power is 0.9. The weight decay and momentum were set to 0.0005 and 0.9, respectively. On *ADE20K*, the base learning rate was 0.02 and the power was 0.9. The weight decay and momentum were 0.0001 and 0.9, respectively. We trained models on 8 GPUs with SBN [40]. The model was trained for 120 epochs on *Cityscapes* and *ADE20K*, 50 on *LIP*, and 80 on *PASCAL VOC 2012*. For data augmentation, the training images were flipped left-right and randomly scaled between a half and twice as in [41].

Comparison methods. Our FPT was applied to the feature pyramids constructed by three methods: UFP [27], PPM [14,44] and ASPP [5]. Based on each of these methods, we compared our FPT to the state-of-the-art pixel-level feature pyramid interaction method, *i.e.*, Object Context Network (OCNet) [38].

Metrics. We used the standard mean Intersection of Union (mIoU) as a uniform metric. We showed the results of ablation study by reporting the mIoU of training set (*i.e.*, Tra.mIoU) and validation set (*i.e.*, Val.mIoU) on the *Cityscapes*.

Methods	Tra.mIoU	Val.mIoU	Params	GFLOPs
UFP [27]	86.0	79.1	71.3 M	916.1
UFP+ST [27]	86.9	80.7	91.2 M	948.4
UFP+LGT [27]	86.5	80.3	102.8 M	1008.3
UFP+RT [27]	86.3	80.1	77.4 M	929.3
UFP+LGT+ST [27]	87.2	80.9	121.3 M	1052.6
UFP+RT+ST [27]	87.0	80.8	96.2 M	985.2
UFP+LGT+RT [27]	86.6	80.4	107.0 M	1014.8
UFP+LGT+ST+RT [27]	**87.4**	**81.7**	127.2 M	1063.9
the improvement	↑ **1.4**	↑ **2.6**		

Table 4. Ablation study on the Cityscapes validation set [7]. "LGT" is Locality-constrained Grounding Transformer; "RT" is Rendering Transformer; "ST" is Self-Transformer. "+" means building the method on the top of UFP.

Methods	Backbone	Cityscapes	ADE20K	LIP	PASCAL VOC 2012
baseline	ResNet-101	65.3	40.9	42.7	62.2
CFNet [41]	ResNet-101	80.6	44.9	54.6	84.2
AFNB [48]	ResNet-101	81.3	45.2	-	-
HRNet [31]	HRNetV2-W48	81.6	44.7	**55.9**	**84.5**
OCNet [38]	ResNet-101	81.7	**45.5**	54.7	84.3
GSCNN [32]	Wide-ResNet-101	**82.8**	-	55.2	-
PPM [44]+OC [38]	ResNet-101	79.9	43.7	53.0	82.9
ASPP [5]+OC [38]	ResNet-101	80.0	44.1	53.3	82.7
UFP [27]+OC [38]	ResNet-101	80.6	44.7	54.5	83.2
PPM [44]+FPT	ResNet-101	80.4(↑ **0.5**)	44.8(↑ **1.1**)	54.2(↑ **1.2**)	83.2(↑ **0.3**)
ASPP [5]+FPT	ResNet-101	80.7(↑ **0.7**)	45.2(↑ **1.1**)	54.4(↑ **1.1**)	83.1(↑ **0.4**)
UFP [27]+FPT	ResNet-101	**82.2**(↑ **1.6**)	**45.9**(↑ **1.2**)	**56.2**(↑ **1.7**)	**85.0**(↑ **1.8**)

Table 5. Comparisons with state-of-the-art on test sets of Cityscapes [7] and PASCAL VOC 2012 [9], validation sets of ADE20K [45] and LIP [12]. Results in this table refer to mIoU; "-" means that there is no reported result in its paper. The **best** and **second best** models under each setting are marked with corresponding formats.

Ablation study. Results are given in Table 4. Applying our transformers (*i.e.*, +ST, +LGT and +RT) to UFP respectively achieves the improvements of 0.9%, 0.5% and 0.3% Tr.mIoU, and the more impressive 1.6%, 1.2% and 1.0% Val.mIoU. Moreover, any component combinations of our transformers yields concretely better results than individual ones. Our best model achieves 1.4% and 2.6% improvements (over UFP) for Tr.mIoU and Val.mIoU, respectively.

Model efficiency. In Table 4, we reported the model Params and GFLOPs. It is clear that using our transformers increases a fair computational overhead. For example, +ST, +LGT and +RT respectively add Params 0.28×, 0.44× and 0.09×, and increase GFLOPs by 0.04×, 0.10× and 0.01×, compared to UFP.

Comparing to the state-of-the-arts. From Table 5, we can observe that our FPT can achieve a new state-of-the-art performance over all the previous methods based on ResNet-101. It obtains improvements as 1.6%, 1.2%, 1.7% and 1.8% mIoU on *Cityscapes* [7], *ADE20K* [45], *LIP* [12] and *PASCAL VOC*

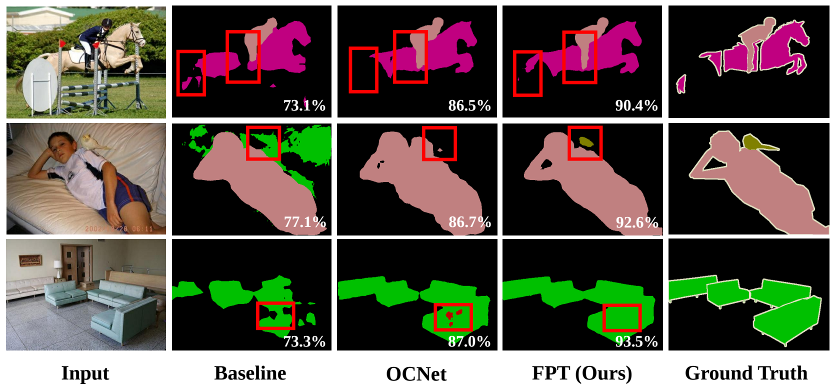

Fig. 5. Visualization results. Samples are from the validation set of *PASCAL VOC 2012* [9]. The value on each image represents the corresponding segmentation mIoU.

2012 [9], respectively. Besides, compared to OCNet, FPT obtains gain by 0.9%, 1.1%, 1.3% and 0.8% mIoU in these four datasets on average. In Fig. 5, we provide the qualitative results[6]. Compared to the baseline [27] and OCNet [38], results of FPT show more precise segmentation for smaller and thinner objects, *e.g.*, the guardrail, person's leg and bird. Moreover, FPT can also achieve more integrated segmentation on some larger objects, *e.g.*, the horse, person and sofa.

5 Conclusion

We proposed an efficient feature interaction approach called FPT, composed of three carefully-designed transformers to respectively encode the explicit self-level, top-down and bottom-up information in the feature pyramid. Our FPT does not change the size of the feature pyramid, and is thus *generic* and easy to plug-and-play with modern deep networks. Our extensive quantitative and qualitative results on three challenging visual recognition tasks showed that FPT achieves consistent improvements over the baselines and the state-of-the-arts, validating its high effectiveness and strong application capability.

Acknowledgements

Wed like to thank all the anonymous reviewers for their constructive comments. This work was partially supported by the National Key Research and Development Program of China under Grant 2018AAA0102002, the National Natural Science Foundation of China under Grant 61925204, the China Scholarships Council under Grant 201806840058, the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant, and the NTU-Alibaba JRI.

[6] More visualization results are given in the *Section E* of the supplementary.

References

1. Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA Engineer **29**(6), 33–41 (1984)
2. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In: ICCV (2019)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. TPAMI **40**(4), 834–848 (2017)
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. In: arXiv (2017)
6. Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., Chua, T.S.: Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: CVPR (2017)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
9. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV **111**(1), 98–136 (2015)
10. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. In: NeurIPS (2018)
11. Girshick, R.: Fast r-cnn. In: ICCV (2015)
12. Gong, K., Liang, X., Zhang, D., Shen, X., Lin, L.: Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In: CVPR (2017)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. TPAMI **37**(9), 1904–1916 (2015)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: CVPR (2018)
17. Irwan, B., Barret, Z., Ashish, V., Jonathon, S., Quoc, V.L.: Attention augmented convolutional networks. In: ICCV (2019)
18. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: ISCA (2017)
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
20. Lin, D., Shen, D., Shen, S., Ji, Y., Lischinski, D., Cohen-Or, D., Huang, H.: Zigzagnet: Fusing top-down and bottom-up context for object segmentation. In: CVPR (2019)
21. Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR (2014)
22. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)

23. 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: ECCV (2014)
24. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: CVPR (2018)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)
26. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
27. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. In: CVPR (2017)
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015)
30. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: ICLR (2015)
31. Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., Wang, J.: High-resolution representations for labeling pixels and regions. In: arXiv (2019)
32. Takikawa, T., Acuna, D., Jampani, V., Fidler, S.: Gated-scnn: Gated shape cnns for semantic segmentation. In: ICCV (2019)
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
34. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
35. Yang, Z., Dai, Z., Salakhutdinov, R., Cohen, W.W.: Breaking the softmax bottleneck: A high-rank rnn language model. In: ICLR (2018)
36. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Learning a discriminative feature network for semantic segmentation. In: CVPR (2018)
37. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
38. Yuan, Y., Wang, J.: Ocnet: Object context network for scene parsing. In: arXiv (2018)
39. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV (2014)
40. Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A.: Context encoding for semantic segmentation. In: CVPR (2018)
41. Zhang, H., Zhang, H., Wang, C., Xie, J.: Co-occurrent features in semantic segmentation. In: CVPR (2019)
42. Zhang, Y., Hare, J., Prügel-Bennett, A.: Learning to count objects in natural images for visual question answering. In: ICLR (2018)
43. Zhang, Z., Zhang, X., Peng, C., Xue, X., Sun, J.: Exfuse: Enhancing feature fusion for semantic segmentation. In: ECCV (2018)
44. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
45. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)
46. Zhou, Y., Zhu, Y., Ye, Q., Qiu, Q., Jiao, J.: Weakly supervised instance segmentation using class peak response. In: CVPR (2018)
47. Zhu, L., Wang, T., Aksu, E., Kamarainen, J.K.: Portrait instance segmentation for mobile devices. In: ICME (2019)

48. Zhu, Z., Xu, M., Bai, S., Huang, T., Bai, X.: Asymmetric non-local neural networks for semantic segmentation. In: ICCV (2019)