Representative Graph Neural Network

Changqian Yu[1,2][0000−0002−4488−4157], Yifan Liu[2][0000−0002−2746−8186], Changxin Gao[1][0000−0003−2736−3920], Chunhua Shen[2][0000−0002−8648−8718], and Nong Sang[1⋆][0000−0002−9167−1496]

[1] Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science & Technology, China
[2] The University of Adelaide, Australia
{changqian_yu,cgao,nsang}@hust.edu.cn

Abstract. Non-local operation is widely explored to model the long-range dependencies. However, the redundant computation in this operation leads to a prohibitive complexity. In this paper, we present a Representative Graph (RepGraph) layer to dynamically sample a few representative features, which dramatically reduces redundancy. Instead of propagating the messages from all positions, our RepGraph layer computes the response of one node merely with a few representative nodes. The locations of representative nodes come from a learned spatial offset matrix. The RepGraph layer is flexible to integrate into many visual architectures and combine with other operations. With the application of semantic segmentation, without any bells and whistles, our RepGraph network can compete or perform favourably against the state-of-the-art methods on three challenging benchmarks: ADE20K, Cityscapes, and PASCAL-Context datasets. In the task of object detection, our RepGraph layer can also improve the performance on the COCO dataset compared to the non-local operation. Code is available at https://git.io/RepGraph.

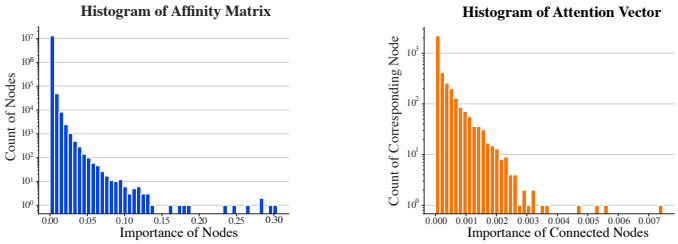
Keywords: Representative Graph; Dynamic Sampling; Semantic Segmentation; Deep Learning

1 Introduction

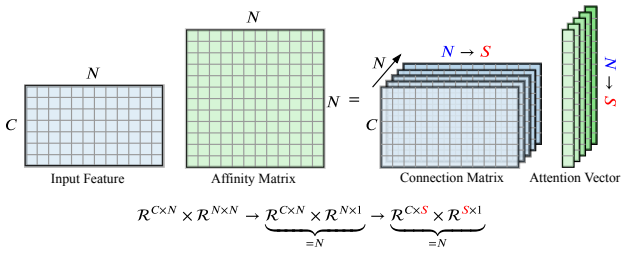
Modelling long-range dependencies is of vital importance for visual understanding, e.g., semantic segmentation [11,18,44] and object detection/segmentation [2, 16, 37]. The previous dominant paradigm is dependent on the deep stacks of local operators, e.g., convolution operators, which are yet limited by inefficient computation, hard optimization and insufficient effective receptive field [29].

To solve this issue, a family of non-local methods [6, 16, 37, 41, 44, 47] is proposed to capture the long-range relations in feature representation. The non-local operation computes the response at a position as a weighted sum of the

⋆ Corresponding author. Part of the work was done when C. Yu was visiting The University of Adelaide.



(a) Statistical analysis of the affinity matrix in the non-local operation



(b) Computation diagram of the non-local operation

Fig. 1. Illustration of the affinity matrix. (a) The statistical analysis of the affinity matrix. In Non-local, the affinity matrix models the importance of all other positions to each position. The histogram of the weights of all the positions is on the left, while that of one position is on the right. These statistical results indicate some *representative* positions exhibit principal importance. (b) The computation diagram of Non-local. The computation in the left can be decoupled into N groups of connection matrix and attention vector. The connection matrix determines which positions require connecting to the current position, while the attention vector assigns the weight to the connection edge. Here, the N above the connection matrix represents N nodes are connected to the current position. Therefore, for each position, we can select S representative nodes to connect and compute a corresponding attention vector to reduce the complexity

features at all positions, whose weights are assigned by a dense affinity matrix. This affinity matrix models the importance of all other positions to each position. Intuitively, there is some redundancy in this affinity matrix, leading to a prohibitive computation complexity. For each position, some other positions may contribute little to its response. This assumption drives us to study further the distributions of importance in the affinity matrix. We perform the statistical analysis on this affinity matrix to help to understand in Figure 1 (a). It is surprising to see that the distribution is extremely imbalanced, which implies that *some representative positions contribute principal impact, while the majority of positions have little contribution.* Therefore, if the affinity matrix can only compute the response with only a few representative positions, the computation redundancy can be dramatically reduced.

Based on previous observations, we rethink Non-local methods from the graphical model perspective and propose an efficient, flexible, and generic operation to capture the long-range dependencies, termed **Rep**resentative **Graph** (RepGraph). Considering each feature position as a node, Non-local constructs a complete graph (fully-connected graph) and assigns a weight to each connection edge via the affinity map. Thus, for each feature node, the computation of non-local graph can be decoupled into two parts: (i) a connection matrix; (ii) a corresponding attention vector, as illustrated in Figure 1 (b). The connection matrix determines which nodes make contributions to each node, while the attention vector assigns the weight for each connection edge.

To reduce the redundant computation, for each node, the *RepGraph* layer dynamically selects a few representative nodes as the neighbourhoods instead of all nodes. The corresponding weight is assigned to the edge to propagate the long-range dependencies. Specifically, the *RepGraph* layer first regresses an offset matrix conditioned on the current node. With the offset matrix, this layer samples the representative nodes on the graph, and then compute an affinity map as the weight to aggregate these representative features.

Meanwhile, the *RepGraph* layer is easy to combine with other mechanisms. Motivated by the pyramid methods [3, 51], we can spatially group some feature elements as a graph node instead of only considering one element, named *Grid Representative Graph (Grid RepGraph)*. Besides, inspired by the channel group mechanism [15, 33, 39, 45, 50], we can also divide the features into several groups, and conduct computation in each corresponding group, termed *Group RepGraph*.

There are several merits of our *RepGraph* layer: (i) The *RepGraph* layer can dramatically reduce the redundant computation of Non-local. Specifically, the RepGraph layer is around 17 times smaller in computation complexity and over 5 times faster than Non-local with a 256×128 input. (ii) The *RepGraph* layer learns a compact and representative feature representation, which is more efficient and effective than non-local operation; (iii) Our *RepGraph* layer can be flexibly integrated into other neural networks or operations.

We showcase the effectiveness of *RepGraph* operations in the application of semantic segmentation. Extensive evaluations demonstrate that the *RepGraph* operations can compete or perform favorably against the *state-of-the-art* semantic segmentation approaches. To demonstrate the generality of *RepGraph* operations, we further conduct the experiments of object detection/segmentation tasks on the COCO dataset [28]. In the comparison of non-local blocks, our *RepGraph* operations can increase the accuracy further.

2 Related Work

Non-local methods and compact representation. Motivated by the non-local means [1], [37] proposes the non-local neural network to model the long-range dependences in the application of video classification [5, 37], object detection and segmentation [16, 37, 47]. The relation network [16] embeds the geometry feature with the self-attention manner [34] to model the relationship

between object proposals. OCNet [44] extends the non-local block with pyramid methods [4, 51]. CFNet [47] explores the co-occurrent context to help the scene understanding, while DANet [11] applies the self-attention on both the spatial and channel dimension. Meanwhile, CPNet [41] explores the supervised self-attention matrix to capture the intra-context and inter-context.

The redundant computation in these nonlocal methods leads to a prohibitive computational complexity, which hinders its application, especially in some dense prediction tasks. Therefore, there are mainly two ways to explore the compact representation of non-local operations: (i) Matrix factorization. A^2-Net [5] computes the affinity matrix between channels. LatentGNN [49] embeds the features into a latent space to get a low-complexity matrix. CGNL [45] groups the channels and utilizes the Taylor expansion reduce computation. CCNet [18] introduces a recurrent criss-cross attention. (ii) Input restricting. Non-local has a quadratic complexity with the size of the input feature. Therefore, restricting the input size is a straightforward approach to reduce complexity. [55] utilizes a pyramid pooling manner to compress the input of the key and value branch. ISA [17] adopts the interlacing mechanism to spatially group the input.

In this paper, we explore a compact and general representation to model the dependencies. Non-local can be a special case of our work, as illustrated in Figure 1 (b). In contrast to previous compact methods, our work dynamically sample the representative nodes to efficiently reduces the spatial redundancy.

Graph neural network. Our work is also related to the graphical neural network [22, 23, 53]. Non-local [34, 37] can be viewed as a densely-connected graph, which models the relationships between any two nodes. Meanwhile, GAT [35] introduces a graph attentional layer, which performs self-attention on each node. In contrast to both dense graphical models, our work constructs a sparse graph, on which each node is simply connected to a few representative nodes.

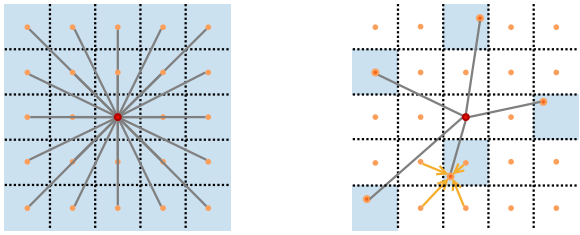
Deformable convolution. Our work needs to learn an offset matrix to locate some representative nodes, which is related to the deformable convolution [8]. The learned offset matrix in DCN is applied on the regular grid positions of convolution kernels, and the number of the sampled positions requires matching with the kernel size. However, our work applies the learned offset to each node position directly. The number of sampled nodes can be unlimited theoretically.

3 Representative Graph Neural Networks

In this section, we first revisit Non-local from the graphical model perspective in Section 3.1. Next, we introduce the motivation, formulation, and several instantiations of the representative graph layer in Section 3.2. Finally, the extended instantiations of the representative graph layer are illustrated in Section 3.3.

3.1 Revisiting Non-local Graph Neural Network

Following the formulation of the non-local operator in [37], we describe a fully-connected graphical neural network.



(a) Non-local graph (b) Representative graph

Fig. 2. Comparison of non-local graph and representative graph. (a) The non-local graph is a fully-connected graph, leading to prohibitive complexity. (b) The representative graph only computes the relationships with some representative nodes, e.g., five in the figure, for each output node. The sampled nodes with fractional positions are interpolated with the nodes at four integral neighbourhood positions

For a 2D input feature with the size of $C \times H \times W$, where C, H, and W denote the channel dimension, height, and width respectively, it can be interpreted as a set of features, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^\intercal$, $\mathbf{x}_i \in \mathbb{R}^C$, where N is the number of nodes (e.g., $N = H \times W$), and C is the node feature dimension.

With the input features, we can construct a fully-connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ as the nodes, and $\mathcal{E}$ as the edges, as illustrated in Figure 2 (a). The graphical model assigns each feature element $\mathbf{x}_i$ as a node $v_i \in \mathcal{V}$, while the edge $(v_i, v_j) \in \mathcal{E}$ encodes the relationship between node v_i and node v_j. Three linear transformation, parameterized by three weight matrices $W_\phi \in \mathbb{R}^{C' \times C}$, $W_\theta \in \mathbb{R}^{C' \times C}$, and $W_g \in \mathbb{R}^{C' \times C}$ respectively, are applied on each node. Therefore, the formulation of the non-local graph network can be interpreted as:

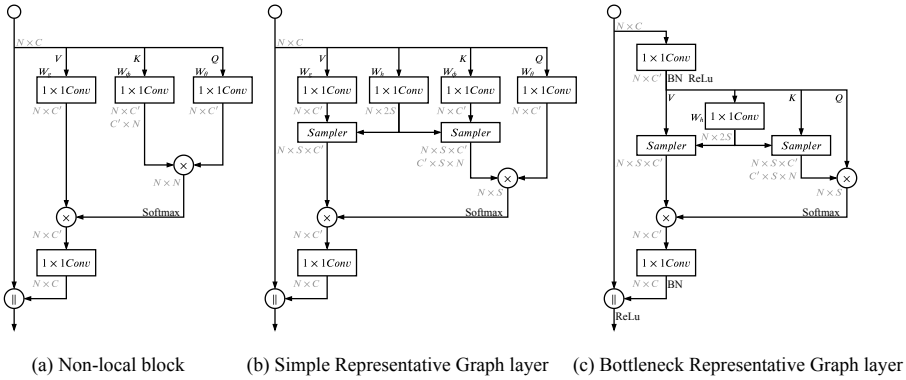
$$\tilde{\mathbf{x}}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j) = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} \delta(\theta(\mathbf{x}_i)\phi(\mathbf{x}_j)^\intercal) g(\mathbf{x}_j), \tag{1}$$

where j enumerates all possible positions, δ is the softmax function, and $\mathcal{C}(\mathbf{x})$ is a normalization factor.

We can rewrite the formulation in Equation 1 in a matrix form:

$$\tilde{\mathbf{X}} = \delta(\mathbf{X}_\theta \mathbf{X}_\phi{}^\intercal)\mathbf{X}_\mathbf{g} = \mathbf{A}(\mathbf{X})\mathbf{X}_\mathbf{g}, \tag{2}$$

where $\mathbf{A}(\mathbf{X}) \in \mathbb{R}^{N \times N}$ indicates the affinity matrix, $X_\theta \in \mathbb{R}^{N \times C'}$, $X_\phi \in \mathbb{R}^{N \times C'}$, and $X_g \in \mathbb{R}^{N \times C'}$. The matrix multiplication results in a prohibitive computation complexity: $\mathcal{O}(C' \times N^2)$. In some visual understanding tasks, e.g., semantic segmentation and object detection, the input usually has a large resolution, which is unfeasible to compute the dense affinity matrix. It is thus desirable to explore a more compact and efficient operation to model the long-range dependencies.



(a) Non-local block (b) Simple Representative Graph layer (c) Bottleneck Representative Graph layer

Fig. 3. Instantiations of Representative Graph layer. (a) is the structure of the non-local block; (b) is the structure of our Simple RepGraph layer, which utilizes learned offset matrix to sample the representative features of the value and key branches; (c) is the structure of our Bottleneck RepGraph layer, following the bottleneck design. Here, $\|$ means the aggregation operation, e.g., *summation* or *concatenation*

3.2 Representative Graph Layer

Motivation. As showed in Figure 1 (b), we can reconstruct the matrix multiplication as a new form:

$$\mathcal{R}^{N\times N} \times \mathcal{R}^{N\times C'} \rightarrow \underbrace{\mathcal{R}^{1\times N} \times \mathcal{R}^{N\times C'}}_{=N}, \tag{3}$$

where $\mathcal{R}^{N\times C'}$, as the connection matrix, determines which nodes are connected to current node, while $\mathcal{R}^{1\times N}$ as the attention vector assigns the weight to corresponding edge.

Based on our observation discussed in Section 1, we can select a few representative nodes (e.g., S) for each node instead of propagating the messages from all nodes. Therefore, the number of connected nodes for each node can be reduced from N to S (usually $S \ll N$), which dramatically reduces the prohibitive computation complexity. The new pipeline can be transformed as:

$$\underbrace{\mathcal{R}^{1\times N} \times \mathcal{R}^{N\times C'}}_{=N} \rightarrow \underbrace{\mathcal{R}^{1\times S} \times \mathcal{R}^{S\times C'}}_{=N}, \tag{4}$$

where S is the number of the representative nodes. This reconstruction reduces the computation cost from $\mathcal{O}(C'\times N^2)$ to $\mathcal{O}(C'\times N\times S)$, usually $S \ll N$ (e.g., for a 65×65 input feature, $N = 65\times 65 = 4225$, while $S = 9$ in our experiments).

Formulation. Based on the observations, we can dynamically sample some nodes to construct the Representative Graph, as illustrated Figure 2 (b).

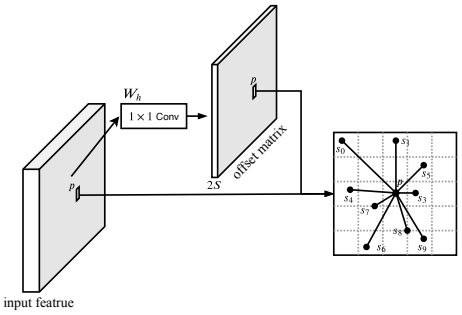


Fig. 4. Illustration of representative nodes sampler. For each position p, the layer adopts one 1×1 convolution operation to regress an offset matrix to sample S representative nodes. The offset matrix has $2S$ channels, the values of which are typically fractional

For each position i of the node feature, we sample a set of representative node features:

$$\mathcal{F}(\mathbf{x}(i)) = \{\mathbf{s}(n)|n = 1, 2, \cdots, S\}, \tag{5}$$

where $s(n) \in \mathcal{R}^{C'}$ is the sampled representative node feature.

Given the sampling function $\mathcal{F}(\mathbf{x}_i)$, Equation 1 can be reformulated as:

$$\tilde{\mathbf{x}}(i) = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall n} \delta(\mathbf{x}_\theta(i)\mathbf{s}_\phi(n)^\intercal)\mathbf{s}_g(n), \tag{6}$$

where n only enumerates the sampled positions, δ is the softmax function, $\mathbf{x}_\theta(i) = W_\theta \mathbf{x}(i), \mathbf{s}_\phi(i) = W_\phi \mathbf{s}(i), \mathbf{s}_g(i) = W_g \mathbf{s}(i)$.

Representative nodes sampler. Motivated by [8], we can instantiate Equation 5 via offset regression. Conditioned on the node features, we can learn an offset matrix to dynamically select nodes. Therefore, for each position p, Equation 5 can be reformulated as:

$$\mathcal{F}(\mathbf{x}(p)) = \{\mathbf{x}(p + \Delta p_n)|n = 1, 2, \cdots, S\}, \tag{7}$$

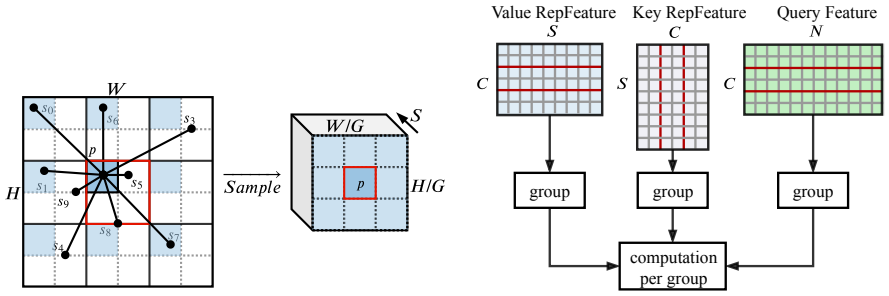
where Δp_n is the regressed offset.

Due to the regression manner, the offset Δp_n is commonly fractional. Thus, we utilize bilinear interpolation [20] to compute the correct values of the fractional position with the node feature at four integral neighbourhood positions:

$$\mathbf{x}(p_s) = \sum_{\forall t} G(t, p_s)\mathbf{x}(t), \tag{8}$$

where $p_s = p + \Delta p_n$, t is four neighbourhood integral positions, and G is bilinear interpolation kernel.

As illustrated in Figure 5, we adopt a 1×1 convolutional layer to regress the offset matrix for each node feature, which has $2S$ channel dimensions. After bilinear interpolation, the RepGraph layer can sample S representative nodes.



(a) Grid RepGraph Layer (b) Group RepGraph Layer

Fig. 5. Illustration of extended instantiations of the RepGraph layer. (a) is Grid RepGraph layer, which spatially grids the input features into several groups, e.g., the features in the red box. Each group has an anchor coordinate, showed in blue cube. The learned offset matrix is applied on the anchor coordinate to sample S representative node features. G indicates how many elements to group along a dimension, e.g., $G = 2$ in the top figure. (b) is Group RepGraph layer, which divides the feature of query branch, sampled representative feature of value branch and key branch into several channel groups respectively. Then, the same computation shown in Equation 6 is conducted in each corresponding group

Instantiations. We can instantiate Equation 6 with a residual structure [14], as illustrated in Figure 3 (b). We define the RepGraph layer as:

$$\mathbf{y}_i = W_y \tilde{\mathbf{x}}_i + \mathbf{x}_i, \tag{9}$$

where $\tilde{\mathbf{x}}_i$ is given in Equation 6, and $\mathbf{x}_i$ is the original input feature. This residual structure enables the RepGraph layer can be inserted to any pre-trained model. We note that when applied into the pre-trained model, W_y should be initialized to zero in avoid of changing the initial behavior of the pre-trained model.

As shown in Figure 3 (b), the RepGraph layer adopts a 1×1 convolution layer to regress the offset matrix, and sample the representative nodes of the key and value branch. The features of the query branch conduct the matrix multiplication with the sampled representative node features of the key branch to obtain the attention matrix. Then the attention matrix assigns corresponding weights and aggregates the representative node features of the value branch.

Meanwhile, motivated by the bottleneck design [14, 39], we re-design the RepGraph layer as a bottleneck structure, as illustrated in Figure 3 (c). We note that when applied to the pre-trained architectures, the weights and biases of last convolution and batch normalization should be initialized to zero, and the ReLU function should be removed.

3.3 Extended Instantiations

Motivated by pyramid methods [3,51], and channel group mechanism [15,39,45], it is easy to instantiate the RepGraph layer with diverse structures.

Grid RepGraph layer. Instead of considering one input feature element as a node, we can spatially group varying quantity of elements as a node. First, we spatially grid input feature into several groups. The left-top element in each group is the anchor position. Then, we utilize the average pooling to group the input features to regress the offset matrix spatially. The learned offset coordinates are applied to the anchor positions to sample some representative nodes for each group. Finally, we conduct the matrix multiplication on the grid features.

Group RepGraph layer. Channel group mechanism is widely used in lightweight recognition architectures, which can reduce the computation and increase the capacity of the model [15,33,39,45,50]. It is easy to be applied on the RepGraph layer. With the input features $\mathbf{x}_\theta(i), \mathcal{F}(\mathbf{x}_\phi(j)), \mathcal{F}(\mathbf{x}_g(j))$, we can divide all C' channels into G groups, each of which has $\tilde{C} = C'/G$ channels. Then, the RepGraph computation of Equation 6 can be performed in each group independently. Finally, we concatenate the output features of all the groups along the dimension of channels as the final output features.

4 Experiments on Semantic Segmentation

We perform comprehensive ablative evaluation on challenging ADE20K [54] dataset. We also report performance on Cityscapes [7] and PASCAL-Context [30] to investigate the effectiveness of our work.

Datasets. The ADE20K dataset contains 20K training images and 2K validation images. It is a challenging scene understanding benchmark due to the complex scene and up to 150 category labels.

Cityscapes is a large urban street scene parsing benchmark, which contains 2,975, 500, 1,525 fine-annotation images for training, validation, and testing, respectively. Besides, there are additional 20,000 coarse-annotation images for training. In our experiments, we only use the fine-annotation set. The images of this benchmark are all in 2048×1024 resolution with 19 semantic categories.

PASCAL-Context [30] augments 10,103 images from PASCAL VOC 2010 dataset [10] for scene understanding, which considers both the stuff and thing categories. This dataset can be divided into 4,998 images for training and 5,105 images for testing. The most common 59 categories are used for evaluation.

Training. We utilize the SGD algorithm with 0.9 momentum to fine-tune the RepGraph network. For the ADE20K and PASCAL-Context datasets, we train our model starting with the initial learning rate of 0.02, the weight decay of 0.0001, and the batch size of 16. For the Cityscapes dataset, the initial rate is 0.01 with the weight decay of 0.0005, while the batch size is 8. We note that we adopt the "poly" learning rate strategy [3], in which the initial learning rate is multiplied by $(1 - \frac{iter}{iter_{max}})^{0.9}$. Besides, the synchronized batch normalization [19, 31, 46] is applied to train our models. We train our models for 100K, 40K, 80K iterations on ADE20K, Cityscapes, PASCAL-Context datasets, respectively.

For the data augmentation, we randomly horizontally flip, randomly scale and crop the input images to a fixed size for training, which scales include

Table 1. Ablations on ADE20K. We show mean IoU (%) and pixel accuracy (%) as the segmentation performance

model, R50	mIoU	pixAcc
R50 baseline	36.48	77.57
NL baseline	40.97	79.96
$S = 1$	42.53	80.08
9	**43.12**	80.27
12	42.60	80.43
15	42.99	80.45
18	42.85	80.44
27	43.06	**80.64**

(a) **Representative nodes**: 1 RepGraph layer of diverse nodes is inserted after the last stage of R50 baseline

model, R50		mIoU	pixAcc
baseline		36.48	77.57
+NL	sum	40.3	79.96
	concate	40.97	80.01
+Simple	sum	42.61	80.191
	concate	42.98	80.41
+Bottleneck	sum	42.67	**80.46**
	concate	**43.12**	80.27

(b) **Instantiations**: 1 RepGraph layer of different structure is inserted after the last stage of ResNet-50 baseline

model, R50	mIoU	pixAcc
baseline	36.48	77.57
res_2	41.52	79.67
res_3	**41.59**	**79.97**
res_4	41.25	79.69
res_5	41.34	79.89

(c) **Stages**: 1 Bottleneck RepGraph layer is inserted into different stages of ResNet-50 baseline

model, R50	mIoU	pixAcc
baseline	36.48	77.57
1-layer	41.59	**79.97**
3-layer	**41.86**	79.85
5-layer	41.76	79.89

(d) **Deeper non-local models**: we insert 1, 3, and 5 Bottleneck RepGraph layer into the ResNet-50 baseline

$\{0.75, 1, 1.25, 1.5, 1.75, 2.0\}$. Meanwhile, the cropped resolutions are 520×520 for ADE20K and PASCAL-Context, and 769×769 for Cityscapes dataset.

Inference. In the inference phase, we adopt the sliding-window evaluation strategy [43,51,52]. Moreover, multi-scale and flipped inputs are employed to improve the performance, which scales contain {0.5, 0.75, 1.0, 1.25} for the ADE20K and PASCAL-Context datasets, and {0.5, 0.75, 1, 1.5} for the Cityscapes dataset.

4.1 Ablative Evaluation on ADE20K

This section provides an ablative evaluation on ADE20K comparing segmentation accuracy and computation complexity. We train all models on the training set and evaluate on the validation set. We adopt the pixel accuracy (pixAcc) and mean intersection of union (mIoU) as the evaluation metric.

Baselines. Similar to [3,4,46,51,52], we adopt dilated ResNet (ResNet-50) [14] with pre-trained weights as our *backbone baseline*. An auxiliary loss function with the weight of 0.4 is integrated into the fourth stage of the backbone network [41, 51,52]. We utilize one 3×3 convolution layer followed by batch normalization [19] and ReLU activation function on the output of the last backbone stage to reduce the channel dimension to 512. Based on this output, we apply one non-local block (NL) as the *NL baseline*. Table 1 (a) shows the segmentation performance of backbone baseline and NL baseline.

Instantiations. Table 1 (b) shows the different structures of RepGraph layer, as illustrated in Figure 3. The simple RepGraph layer (SRG) has a similar structure with the non-local operation, while the bottleneck RepGraph layer (BRG) combines the residual bottleneck [14, 39] with the simple RepGraph layer. The number of sampled representative nodes is 9. Meanwhile, we also compare the different fusion methods (*summation* and *concatenation*) of diverse structures. The bottleneck RepGraph layer has stronger representation ability, which achieves better performance than the simple RepGraph layer. Therefore, in the rest of this paper, we use the bottleneck RepGraph layer version by default.

Table 2. Practical GFLOPs of different blocks with the input feature of 256×128 resolution (1/8 of the 1024×2048 image). The batch size of the input feature is 1, while the input channel $C = 2048$ and middle channel $C' = 256$. The inference time is measured on one NVIDIA RTX 2080Ti card. The decrease of our methods in term of computation and inference time is compared with NL

input size	model	GFLOPs	Inference Time(ms)
256×128	NL [44]	601.4	146.65
	DANet [11]	785.01	279.56
	SRG [ours]	45.31 ($\downarrow$ 556.09)	60.89 ($\downarrow$ 85.76)
	BRG [ours]	34.96 ($\downarrow$ 566.44)	25.96 ($\downarrow$ 120.69)

Table 3. Extended instantiations of RepGraph layer. We can spatially group the spatial nodes, termed *Grid RepGraph*, or divide the channel into several groups, termed *Group RepGraph*. We show mean IoU (%) and pixel accuracy (%) as the segmentation performance

model, R50	mIoU	pixAcc
baseline	36.48	77.57
Grid RG($g_s = 1$)	43.12	80.27
$g_s = 5$	42.40	80.17
13	41.82	80.01
65	41.23	79.73

(a) **Spatial group**: g_s is the number of spatially grouped elements in one dimension, (e.g., for a 2D input, $g_s = 5$ indicates groups 5×5 elements as a graph node)

model, R50	mIoU	pixAcc
baseline	36.48	77.57
Group RG($g_c = 1$)	43.12	80.27
$g_c = 4$	42.78	80.20
8	43.01	80.32
16	42.96	80.19
32	42.38	80.18

(b) **Channel group**: g_c indicates how many channels require dividing into on group

How many representative nodes to sample? Table 1 (a) compares the performance of choosing different number of representative nodes. It shows *all* the models can improve the performance over the ResNet-50 baseline and are better than the NL baseline, which validates the effectiveness and robustness of our RepGraph layer. We employ $S = 9$ as our default. Interestingly, even only choosing one representative node ($s = 1$) for each position can also lead to a 1.56% performance improvement. This validates reducing redundancy in non-local helps to more effective representation. For better understanding, we show some visualization of learned sampling positions in the supplementary material.

Next, we investigate the combination with the pre-trained model (e.g., ResNet). Due to insertion into the pre-trained model, we can not change the initial behaviour of the pre-trained model. Therefore, we have to choose the summation version of bottleneck RepGraph layer and remove the last ReLU function of this layer. Meanwhile, the parameters of the last convolution layer and batch normalization require to initialize as zero.

Table 4. Quantitative evaluations on the ADE20K validation set. The proposed RGNet performs favorably against the *state-of-the-art* segmentation algorithms

model	reference	backbone	*mIoU*	*picAcc*
RefineNet [26]	CVPR2017	ResNet-152	40.7	-
UperNet [38]	ECCV2018	ResNet-101	42.66	81.01
PSPNet [51]	CVPR2017	ResNet-269	44.94	81.69
DSSPN [25]	CVPR2018	ResNet-101	43.68	81.13
PSANet [52]	ECCV2018	ResNet-101	43.77	81.51
SAC [48]	ICCV2017	ResNet-101	44.30	**81.86**
EncNet [46]	CVPR2018	ResNet-101	44.65	81.69
CFNet [47]	CVPR2019	ResNet-101	44.89	-
CCNet [18]	ICCV2019	ResNet-101	45.22	-
ANL [55]	ICCV2019	ResNet-101	45.24	-
DMNet [12]	ICCV2019	ResNet-101	<u>45.50</u>	
RGNet	-	ResNet-50	44.02	81.12
RGNet	-	ResNet-101	**45.8**	<u>81.76</u>

Which stage to insert RepGraph layer? We insert the bottleneck RepGraph layer before the last block of different backbone stages, as shown in Table 1 (c). The improvements over the backbone baseline validate the RepGraph layer can be a generic component to extract features.

Going deeper with RepGraph layer. Table 1 (d) shows the performance with more RepGraph layers. We add 1 (to res_3), 3 (to res_3, res_4, res_5 respectively), and 5 (1 to res_3, 2 to res_4, 2 to res_5). With more layers, the performance can be improved further. This improvement validates the RepGraph layer can model some complementary information not encoded in the pre-trained model.

Computation complexity. The theoretical computational complexity of non-local operation and RepGraph layer is $\mathcal{O}(C \times N^2)$ and $\mathcal{O}(C \times S \times N)$ respectively. Meanwhile, Table 2 shows the practical GFLOPs and inference time of non-local operation [37], DANet [11] and RepGraph layer with the input size of 256×128 (1/8 of the 1024×2048 image). Here, we use the concatenation fusion method in each block. The RepGraph layer can dramatically reduce the computation complexity and have fewer inference time compared to the non-local operation.

Then, we show some extended instantiations of RepGraph layer.

Extension. Table 3 shows some extended instantiations of the RepGraph layer. Instead of considering one feature element as a graph node, we can spatially group a few pixels as a graph node to construct the RepGraph layer, termed *Grid RepGraph*. We argue that the Grid RepGraph layer computes the relationships between representative nodes and local nodes in one group. The sampling of representative nodes can capture long-range information, while the spatial grouping enables the short-range contextual modelling.

Inspired by [39,45], the channel of RepGraph layer can be divided into a few groups, called *Group RepGraph* layer. This structure can increase the cardinality

Table 5. Quantitative evaluations on Cityscapes test set. The proposed RGNet performs favorably against the *state-of-the-art* segmentation methods. We train our model with *trainval-fine* set, and evaluate on the *test* set

model	reference	backbone	*mIoU*
GCN [32]	CVPR2017	ResNet-101	76.9
DUC [36]	WACV2018	ResNet-101	77.6
DSSPN [25]	CVPR2018	ResNet-101	77.8
SAC [48]	ICCV2017	ResNet-101	78.1
PSPNet [51]	CVPR2017	ResNet-101	78.4
BiSeNet [42]	ECCV2018	ResNet-101	78.9
AAF [21]	ECCV2018	ResNet-101	79.1
DFN [43]	CVPR2018	ResNet-101	79.3
PSANet [52]	ECCV2018	ResNet-101	80.1
DenseASPP [40]	CVPR2018	DenseNet-161	80.6
ANL [55]	ICCV2019	ResNet-101	81.3
CPNet [41]	CVPR2020	ResNet-101	81.3
CCNet [18]	ICCV2019	ResNet-101	81.4
DANet [11]	CVPR2019	ResNet-101	**81.5**
RGNet	-	ResNet-101	**81.5**

and capture the correlation in diverse channel groups. Although there is a little performance decrease, the extended instantiations are more efficient.

4.2 Performance Evaluation

In this section, we compare the RepGraph network (RGNet) with other *state-of-the-art* methods on three datasets: ADE20K, Cityscapes, and PASCAL-Context.

ADE20K. Table 4 shows the comparison results with other *state-of-the-art* algorithms on ADE20K dataset. *Without any bells and whistles*, our RGNet with ResNet-101 as backbone achieves mean IoU of 45.8% and pixel accuracy of 81.76%, which outperforms previous *state-of-the-art* methods. Our RGNet with ResNet-50 obtains mean IoU of 44.04% and pixel accuracy of 81.12%, even better than the PSANet [52], PSPNet [51], UperNet [38], and RefineNet [26] with deeper backbone networks.

Cityscapes. Table 5 shows the comparison with previous results on Cityscapes [7] dataset. We train our model with *trainval* set of merely the fine annotation images, and evaluate on the *test* set. The compared methods only use the fine-annotation images as well. The RGNet achieves mean IoU of 81.5%, which competes with previous *state-of-the-art* methods. However, as shown in Table 5, the RGNet is more efficient than the DANet, which applies the self-attention mechanism on the spatial and channel dimension respectively.

PASCAL-Context. Table 6 shows the results on the PASCAL-Context dataset compared with other methods. The RGNet achieves mean IoU of 53.9% on the *val* set, which sets *state-of-the-art* result.

Table 6. Quantitative evaluations on the PASCAL-Context validation set. The proposed RGNet performs favorably against the *state-of-the-art* segmentation methods.

model	reference	backbone	*mIoU*
CRF-RNN [53]	ICCV2015	VGG-16	39.3
RefineNet [26]	CVPR2017	ResNet-152	47.3
PSPNet [51]	CVPR2017	ResNet-101	47.8
CCL [9]	CVPR2018	ResNet-101	51.6
EncNet [46]	CVPR2018	ResNet-101	51.7
DANet [11]	CVPR2019	ResNet-101	52.6
ANL [55]	ICCV2019	ResNet-101	52.8
EMANet [24]	ICCV2019	ResNet-101	<u>53.1</u>
CPNet [41]	CVPR2020	ResNet-101	**53.9**
RGNet	-	ResNet-101	**53.9**

Table 7. Adding 1 RepGraph layer to Mask R-CNN for COCO **object detection** and **instance segmentation**. The backbone is ResNet-50 with FPN [27]

method		AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R50	baseline	38.0	59.6	41.0	34.6	56.4	36.5
	+1 NL	39.0	61.1	41.9	35.5	**58.0**	37.4
	+1 RGL	**39.6**	**61.4**	**42.1**	**36.0**	57.9	**37.9**

5 Experiments on Detection

To investigate the generalization ability of our work, we conduct experiments on object detection. Following [37], we set the Mask R-CNN [13] as our baseline. All experiments are trained on COCO [28] *train*2017 and tested on *test*2017.

We add one RepGraph layer before the last block of res_4 of ResNet backbone network in the Mack R-CNN. Table 7 shows the box AP and mask AP on COCO dataset. As we can see, using just one RepGraph layer can improve the performance over the baseline. Meanwhile, adding one RepGraph layer achieves *better* performance than adding one non-local operation.

6 Concluding Remarks

We present a Representative Graph (RepGraph) layer to model long-range dependencies via dynamically sample a few representative nodes. The RepGraph layer is compact and general component for visual understanding. Meanwhile, the RepGraph layer is easy to integrate into any pre-trained model or combined with other designs. On the semantic segmentation and object detection task, the RepGraph layer can achieve promising improvement over baseline and non-local operation. We believe the *RepGraph* layer can be an efficient and general block to the visual understanding community.

Acknowledgment: This work is supported by the National Natural Science Foundation of China (No.61433007 and 61876210).

References

1. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 2, pp. 60–65. IEEE (2005) 3
2. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Gcnet: Non-local networks meet squeeze-excitation networks and beyond. arXiv (2019) 1
3. 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. arXiv (2016) 3, 8, 9, 10
4. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv (2017) 4, 10
5. Chen, Y., Kalantidis, Y., Li, J., Yan, S., Feng, J.: Aˆ 2-nets: Double attention networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 352–361 (2018) 3, 4
6. Chen, Y., Rohrbach, M., Yan, Z., Shuicheng, Y., Feng, J., Kalantidis, Y.: Graph-based global reasoning networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 433–442 (2019) 1
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: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 9, 13
8. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 764–773 (2017) 4, 7
9. Ding, H., Jiang, X., Shuai, B., Qun Liu, A., Wang, G.: Context contrasted feature and gated multi-scale aggregation for scene segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2393–2402 (2018) 14
10. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html 9
11. Fu, J., Liu, J., Tian, H., Fang, Z., Lu, H.: Dual attention network for scene segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 1, 4, 11, 12, 13, 14
12. He, J., Deng, Z., Qiao, Y.: Dynamic multi-scale filters for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (October 2019) 12
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2961–2969 (2017) 14
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 8, 10
15. 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 (2017) 3, 8, 9
16. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 1, 3

17. Huang, L., Yuan, Y., Guo, J., Zhang, C., Chen, X., Wang, J.: Interlaced sparse self-attention for semantic segmentation. arXiv (2019) 4
18. Huang, Z., Wang, X., Huang, C., Wei, Y., Liu, W.: Ccnet: Criss-cross attention for semantic segmentation. Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2019) 1, 4, 12, 13
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 448–456 (2015) 9, 10
20. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2017–2025 (2015) 7
21. Ke, T.W., Hwang, J.J., Liu, Z., Yu, S.X.: Adaptive affinity fields for semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 587–602 (2018) 13
22. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: Advances in Neural Information Processing Systems (NeurIPS) (2011) 4
23. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML) (2001) 4
24. Li, X., Zhong, Z., Wu, J., Yang, Y., Lin, Z., Liu, H.: Expectation-maximization attention networks for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (October 2019) 14
25. Liang, X., Zhou, H., Xing, E.P.: Dynamic-structured semantic propagation network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 752–761 (2018) 12, 13
26. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 12, 13, 14
27. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2125 (2017) 14
28. 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: Proceedings of the European Conference on Computer Vision (ECCV) (2014) 3, 14
29. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 4898–4906 (2016) 1
30. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014) 9
31. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6181–6189 (2018) 9
32. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 13
33. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Inverted residuals and linear bottlenecks: Mobile networks for classification. arXiv **1801** (2018) 3, 9

34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS) (2017) 3, 4
35. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. Proceedings of the International Conference on Learning Representations (ICLR) (2018) 4
36. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV) (2018) 13
37. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 1, 3, 4, 12, 14
38. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 418–434 (2018) 12, 13
39. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1492–1500 (2017) 3, 8, 9, 10, 12
40. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: Denseaspp for semantic segmentation in street scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3684–3692 (2018) 13
41. Yu, C., Wang, J., Gao, C., Yu, G., Shen, C., Sang, N.: Context prior for scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12416–12425 (2020) 1, 4, 10, 13, 14
42. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 325–341 (2018) 13
43. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Learning a discriminative feature network for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 10, 13
44. Yuan, Y., Wang, J.: Ocnet: Object context network for scene parsing. arXiv (2018) 1, 4, 11
45. Yue, K., Sun, M., Yuan, Y., Zhou, F., Ding, E., Xu, F.: Compact generalized non-local network. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 6510–6519 (2018) 3, 4, 8, 9, 12
46. Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A.: Context encoding for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7151–7160 (2018) 9, 10, 12, 14
47. Zhang, H., Zhang, H., Wang, C., Xie, J.: Co-occurrent features in semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 548–557 (2019) 1, 3, 4, 12
48. Zhang, R., Tang, S., Zhang, Y., Li, J., Yan, S.: Scale-adaptive convolutions for scene parsing. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2031–2039 (2017) 12, 13
49. Zhang, S., Yan, S., He, X.: Latentgnn: Learning efficient non-local relations for visual recognition. In: Proceedings of the International Conference on Machine Learning (ICML) (2019) 4
50. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6848–6856 (2018) 3, 9

51. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 3, 4, 8, 10, 12, 13, 14
52. Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C.C., Lin, D., Jia, J.: PSANet: Point-wise spatial attention network for scene parsing. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018) 10, 12, 13
53. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015) 4, 14
54. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. CoRR **abs/1608.05442** (2016) 9
55. Zhu, Z., Xu, M., Bai, S., Huang, T., Bai, X.: Asymmetric non-local neural networks for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 593–602 (2019) 4, 12, 13, 14