Towards Efficient Scene Understanding via Squeeze Reasoning

Xiangtai Li, Xia Li, Ansheng You, Li Zhang, Guangliang Cheng, Kuiyuan Yang, Yunhai Tong, Zhouchen Lin

Abstract—Graph-based convolutional model such as non-local block has shown to be effective for strengthening the context modeling ability in convolutional neural networks (CNNs). However, its pixel-wise computational overhead is prohibitive which renders it unsuitable for high resolution imagery. In this paper, we explore the efficiency of context graph reasoning and propose a novel framework called Squeeze Reasoning. Instead of propagating information on the spatial map, we first learn to squeeze the input feature into a channel-wide global vector and perform reasoning within the single vector where the computation cost can be significantly reduced. Specifically, we build the node graph in the vector where each node represents an abstract semantic concept. The refined feature within the same semantic category results to be consistent, which is thus beneficial for downstream tasks. We show that our approach can be modularized as an end-to-end trained block and can be easily plugged into existing networks. Despite its simplicity and being lightweight, the proposed strategy allows us to establish the considerable results on different semantic segmentation datasets and shows significant improvements with respect to strong baselines on various other scene understanding tasks including object detection, instance segmentation and panoptic segmentation. Code is available at https://github.com/lxtGH/SFSegNets.

Index Terms—Channel Attention, Efficient Global Context Modeling, Scene Understanding

I. INTRODUCTION

CONVOLUTIONAL neural networks have proven to be effective and useful to learn visual representations in an end-to-end fashion with a certain objective task such as, semantic segmentation [1], image classification [2], object detection [3], instance segmentation [4] and panoptic segmentation [5]. However, the effective receptive field [6] of CNNs grows slowly if we simply stack local convolutional layers. Thus, the limited receptive field prevents the model from taking all the contextual information into account and thus renders the model insufficiently covering all the regions of interest.

A broad range of prior research has investigated network architecture designs to increase the receptive field-of-view such as self-attention/Non-local [7], channel attention [8], and graph convolution network (GCN) [9]. Although they have been shown to be effective in strengthening the representations produced by CNNs, modeling the inter-dependencies in a high-dimensional space prevents them from fully exploiting the sparse property required by the final classifier. Furthermore, they suffer from the prohibitively expensive computation overhead during training and inference, e.g., the large affinity matrix at each spatial position [8] or channel position [9], which renders these methods unsuitable for high-resolution inputs. Although recent methods reduce such cost by involving fewer pixels [10] or selecting representative nodes [11], their computation is still huge given high resolution image inputs.

Could we find another way to eliminate the limitation of high-cost spatial information while capturing global context information? We first carry out toy experiments using a pretrained Deeplabv3+ model [12]. We count the pixels on the final normalized feature (512 dimensions before classification) given ground truth masks whose activation values are beyond 0.8. As shown in Fig 1(a), we find different classes lie in different groups along channels sparsely. We only show three classes for simplicity. This motivates us to build an information propagation module on channel solely where each group represents one specific semantic class while the cost of spatial resolution can be avoided. Inspired by SE-networks [8], we first squeeze the feature into a compact global vector and then perform reasoning operation on such compact vector. Benefit from squeezing, the computation cost...
can be significantly reduced compared with previous works. The schematic illustration of our proposed method is shown in Fig 1(b). Compared with previous work modeling pair-wised affinity maps over the input pixels [10], [13]–[17], our method is totally different by building node graph conditionally on the whole image statistics and also results in efficient inference. After reasoning, the most representative channels of input features can be selected and enhanced which solves the inconsistent segmentation results on large objects.

Our framework mainly contains three steps. First, we perform the node squeezing operation to obtain the global vector. This can be done by simply a global average pooling or using Hadamard product pooling to capture the second-order statistics. Then we carry out node reasoning by dividing such vector into different groups, and the inter-dependencies can be diffused through the reasoning process. Finally, we reconstruct the original feature map by multiplying the reasoned vector with the original input. Our approach can serve as a lightweight module and can be easily plugged into existing networks. Compared to Non-local [7] or graph convolution network [9], which model the global relationship on feature spatial or channel dimension, our approach instead models the inter-dependencies on the squeezed global vector space, and notably, each node consists of a group of atom/channel. Therefore, our method uses substantially fewer floating-point operations and fewer parameters and memory costs. Moreover, our method achieves the best speed and accuracy trade-off on the Cityscapes test set, which shows its practical usage. In particular, our method achieves 77.5% mIoU on Cityscapes test set while running at 65 FPS on single 1080-TI device.

Besides its efficiency, our method is also verified to be effective on long-range context modeling. As shown in Tab. I, our method results in a significant gain on dilated FCN with ResNet50 backbone. In particular, the segmentation results of larger objects in the scene can be improved significantly by over 10% mIoU per category. Moreover, our methods only require a few extra computation cost (1.5% relative increase over the baseline models). Fig 2 gives the visualization results on the corresponding models in Tab. I. As shown in that figure, the inconsistent noise on the train and truck can be removed by our proposed SR module.

Moreover, our proposed method is also verified to other tasks including instance level tasks on instance segmentation and panoptic segmentation on COCO datasets [18], image classification on ImageNet [19]. Our method can obtain consistent improvements over mask-rcnn baseline models [4] with negligible cost and considerable results on ImageNet classification with SE-net [8]. More detailed information can be found in the experiment parts. Those experiments further demonstrate the generality of our proposed approach.

The contributions of this work are as follows:

(i) We propose a novel squeeze reasoning framework for highly efficient deep feature representation learning for scene understanding tasks.

(ii) An efficient node graph reasoning is introduced to model the inter-dependencies between abstract semantic nodes. This enables our method to serve as a lightweight and effective module and can be easily deployed in existing networks.

(iii) Extensive experiments demonstrate that the proposed approach can establish new state-of-the-arts on four major semantic segmentation benchmark datasets including Cityscapes [20], Pascal Context [21], ADE20K [22] and Camvid [23] while keeping efficiency, and show consistent improvement with respect to strong baselines on several scene understanding tasks with negligible cost. More experiments on different tasks through datasets [18], [19] including instance segmentation and image classification prove the generality of proposed SR module.

II. RELATED WORK

In this section, we will review the related work in two aspects: global context aggregation and semantic segmentation.

A. Global context aggregation

Beyond the standard convolutional operator used for short-range modeling, many long-range operators are proposed to aggregate information from large image regions, even the whole image. Global Average Pooling (GAP) [2], which bridges local feature maps and global classifiers, is widely used for long-range modeling. In Squeeze-and-Excitation network [8], GAP is used in more intermediate layers for coupling global information and local information more thoroughly. In Pyramid Pooling Module (PPM) [23], a pyramid of average pooling operators is used to harvest features. In addition to first-order statistics captured by GAP, bilinear pooling [25] extracts image-level second-order statistics as complementary of convolutional features. Besides pooling-based operators, generalized convolutional operators [26] are also used for
long-range modeling. Atrous convolution enlarges kernels by inserting zeros in between \([27]\), which is further used by stacking kernels with multiple atrous rates pyramidally \([28]–[30]\) or densely \([31]\). Deformable convolution \([32], [33]\) generalizes atrous convolution by learning the offsets for convolution sampling locations. Global average pooled features are concatenated into existing feature maps on \([34]\). In PSPNet \([24]\), average pooled features of multiple window sizes including global average pooling are upsampled to the same size and concatenated together to enrich global information. The DeepLab series of papers \([28]–[30]\) propose atrous or dilated convolutions and atrous spatial pyramid pooling (ASPP) to increase the effective receptive field. DenseASPP \([31]\) improves on \([29]\) by densely connecting convolutional layers with different dilation rates to further increase the receptive field of network. In addition to concatenating global information into feature maps, multiplying global information into feature maps also shows better performance \([15], [35]–[37]\). In particular, EncNet \([15]\) and SqueezeSeg \([38]\) use attention along the channel dimension of the convolutional feature map to account for global context such as the co-occurrences of different classes in the scene. CBAM \([35]\) explores channel and spatial attention in a cascade way to learn task specific representation.

Recently, advanced global information modeling approaches initiated from non-local network \([7]\) are showing promising results on scene understanding tasks. In contrast to convolutional operator where information is aggregated locally defined by local filters, non-local operators aggregate information from the whole image based on an affinity matrix calculated among all positions around the image. Using non-local operator, impressive results are achieved in OCNet \([13]\), CoCurNet \([39]\), DANet \([40]\), A2Net \([41]\), CCNet \([10]\) and Compact Generalized Non-Local Net \([36]\). OCNet \([13]\) uses non-local blocks to learn pixel-wise relationship while CoCurNet \([39]\) adds extra global average pooling path to learn whole scene statistic. DANet \([40]\) explores orthogonal relationships in both channel and spatial dimension using non-local operator. CCNet \([10]\) models the long range dependencies by considering its surrounding pixels on the cross-crop path through a recurrent way to save both computation and memory cost. Compact Generalized non-local Net \([36]\) considers channel information into affinity matrix. Another similar work to model the pixel-wised relationship is PSANet \([42]\). It captures pixel-to-pixel relations using an attention module that takes the relative location of each pixel into account. EMANet \([43]\) proposes to adopt expectation-maximization algorithm \([44]\) for the self-attention mechanism.

Another way to get global representation is using graph convolutional networks, and do reasoning in a non-euclidean space \([45]–[47]\) where messages are passing between each node before projection back to each position. Glorot \([9]\) projects the feature map into interaction space using learned projection matrix and does graph convolution on projected fully connected graph. BeyondGrids \([45]\) learns to cluster different graph nodes and does graph convolution in parallel. SPGNet \([46]\) performed spatial pyramid graph reasoning while DGMNet \([47]\) proposed dynamic graph reasoning framework for more efficient learning. In our work, a global vector squeezed from the whole image is organized as a small graph for reasoning, where each node contains rich global information. Thus reasoning is carried on a high level, which is more efficient and robust to noises than previous methods.

B. Semantic segmentation

Recent years have seen a lot of work on semantic segmentation using deep neural network. FCN \([1]\) removes global information aggregation layers such as global average pooling layer and fully-connected layers for semantic segmentation. Later, FCN-based methods dominate the area of image semantic segmentation. We review related methods in two different settings: non-real-time methods for better segmentation results and real-time models for fast inference. The work \([27]\) removed the last two downsample layers to obtain dense prediction and utilized dilated convolutions to enlarge the receptive field. Meanwhile, both SAC \([48]\) and DCN \([32]\) improved the standard convolutional operator to handle the deformation and various scales of objects, which also enlarge the receptive fields of CNN operator. Several works \([49]–[52]\) adopted encoder-decoder structures that fuses the information in low-level and high-level layers to make dense prediction results. In particular, following such architecture design, GFFnet \([53]\), CCLNet \([54]\) and G-SCNN \([55]\) use gates for feature fusing to avoid noise and feature redundancy. CRF-RNN \([56]\) used graph model such CRF, MRF for semantic segmentation. AAF \([57]\) used adversarial learning to capture and match the semantic relations between neighboring pixels in the label space. DenseDecoder \([58]\) built multiple long-range skip connections on cascaded architecture. DPC \([59]\) and auto-deeplab \([60]\) utilized architecture search techniques to build multi-scale architectures for semantic segmentation. Besides, there are also several works aiming for real time application. ICNet \([61]\), BiSegNet \([62]\) and SFNet \([63]\) were designed for real-time semantic segmentation by fusing multi scale inputs or feature pyramids. DFFNet \([64]\) utilizes a light-weight backbone to speed up its network and proposes a cross-level feature aggregation to boost accuracy, while SwiftNet \([65]\) uses lateral connections as the cost-effective solution to restore the prediction resolution while maintaining the speed. There are also specially designed video semantic segmentation works for boosting accuracy \([66], [67]\) and saving inference time \([68], [69]\). Our proposed module can work in both real-time setting to obtain the best speed and accuracy trade-off due to its efficiency or non-real-time setting to achieve the better consistent segmentation results.

III. Method

In this section, we first review related works \([7], [8], [70]\) as preliminary knowledge. Then detailed description and formulation of our SR module are introduced. Finally, we elaborate on how to apply it to several different computer vision tasks.

A. Preliminaries

**Graph convolution.** Assume an input matrix \(X \in \mathbb{R}^{D \times N}\), where \(D\) is the feature dimension and \(N = H \times W\) is the...
Graph Reasoning: perform GCN reasoning in node space. Feature Reconstruction: reconstruct the feature by the reasoned global vector.

Fig. 3. Schematic illustration of our proposed SR module. Our module contains three steps. Node Squeezing: squeeze the feature into separate nodes. Node Graph Reasoning: perform GCN reasoning in node space. Feature Reconstruction: reconstruct the feature by the reasoned global vector.

number of locations defined on regular grid coordinates \( \Omega = \{1, \ldots, H\} \times \{1, \ldots, W\} \). In standard convolution, information is only exchanged among positions in a small neighborhood defined by the filter size (typically \( 3 \times 3 \)). In order to create a large receptive field and capture long-range dependencies, one needs to stack numerous layers after each other, as done in common architectures \([2]\). Graph convolution \([70]\), is a highly efficient, effective and differentiable module that generalizes the neighborhood definition used in standard convolution and allows long-range information exchange in a single layer. This is done by defining edges \( E \) among nodes \( V \) in a graph \( G \). Formally, the graph convolution is defined as

\[
\tilde{X} = \sigma(W X A),
\]

where \( \sigma(\cdot) \) is the non-linear activation function, \( A \in \mathbb{R}^{N \times N} \) is the adjacency matrix characterising the neighbourhood relations of the graph and \( W \in \mathbb{R}^{D \times D} \) is the weight matrix. So the graph definition and structure play a key role in determining the information propagation.

**Non-local network.** We describe non-local network \([7]\) in view of a fully connected graphical model. For a 2D input feature with the size of \( C \times H \times W \), where \( C, H, W \) denote the channel dimension, height, and width respectively, it can be interpreted as a set of features, \( X = [x_1, x_2, \cdots, x_N]^T \), \( x_i \in \mathbb{R}^C \), where \( N \) is the number of nodes \((e.g. N = H \times W)\), and \( C \) is the node feature dimension.

\[
\tilde{X} = \delta(X_\theta X_\phi^T)X_g = A(X)X_g,
\]

where \( A(X) \in \mathbb{R}^{N \times N} \) indicates the affinity matrix, \( X_\theta \in \mathbb{R}^{N \times C'} \), \( X_\phi \in \mathbb{R}^{N \times C'} \), and \( X_g \in \mathbb{R}^{N \times C'} \) which are projection matrix. In summary, according to Equ. 1 and Equ. 2 both the computation and affinity cost are highly dependent on the number of node \( N \).

**‘Squeeze’ operation.** The ‘squeeze’ operation is commonly used in networks for image classification, and adopted in SE-Net \([8]\) to summarize the global contexts for intermediate layers for channel weights re-calibration. One simple implementation of this operation is the Global Average Pooling (GAP).

**B. SR module formulation**

As discussed, pixels or nodes’ choices are essential for reducing computation cost for both graph convolution models and non-local models. Recent works \([10, 14]\) follow this idea to achieve less computation cost. However, both the affinity memory and computation cost are still linearly dependent on the input resolution. In particular, this will limit their usage for some applications such as road scene understanding with high-resolution image inputs. Different from their approaches, we propose a simple yet effective framework named Squeeze Reasoning. Our approach mainly contains three steps. First, we squeeze the input feature map into a compact global vector. We then split such vector into different groups or nodes and perform graph reasoning operation on such input node graphs. Finally, we reconstruct the original feature map by multiplying the reasoned vector back into the input feature. We specify the details of these three steps in the following parts. Fig. 3 shows the detailed pipeline of our SR module.

1) **Node squeezing:** It is well known that the global vector describes whole image statistics, which is a key component in many modern convolutional network architectures for different tasks such as object detection, scene parsing and image generation. The simplest way to calculate the global vector is the global average pooling, which calculates the first-order whole image statistics.

Recent works \([25, 41]\) use the bilinear pooling to capture second-order statistics of features and generate global representations. Compared with the conventional average and max pooling, which only computes first-order statistics, bilinear pooling can better capture and preserve complex relations. In particular, bilinear pooling gives a sum pooling of second-order features from the outer product of all the feature vector pairs \((b_i, c_i)\) within two input feature maps \(B\) and \(C\). It can be expressed as follows:

\[
G_{\text{bilinear}}(B, C) = BC^T = \sum_{i=1}^{HW} b_i c_i^T,
\]

where \( B = [b_1, \ldots, b_i, \ldots, b_{HW}] \in \mathbb{R}^{C \times HW} \) and \( C = [c_1, \ldots, c_i, \ldots, c_{HW}] \in \mathbb{R}^{C \times HW} \). The output size is \( C \times C \).

To get a more compact vector for each node, in this paper, instead of generating outer product of all feature pairs from \( b_i \) and \( c_i \) within two input feature maps \( B \) and \( C \), we calculate
the Hadamard product:
\[
G_{\text{global}}(B, C) = \sum_{i=1}^{HW} B_i \circ C_i, \tag{4}
\]
where \(\circ\) means the Hadamard product operation. To be noted that, we first reduce channel dimension of input feature \(X\) by \(1 \times 1\) convolution layer into \(\tilde{X}\) and then we perform pooling operation using Eq. \([4]\) or simple global average pooling.

2) Node graph reasoning: To form a node graph, we divide vector \(g\) into \(k\) different groups with each group size of \(d\) where \(C = k \times d\). We use the graph convolution to model the relationship between nodes and consider it a fully-connected graph. As for the transformation \(W\) in Eq. \([1]\) we adopt a \(1 \times 1\) convolutional layer to implement it. Moreover, for the adjacency matrix \(A\), we will show by experiments that our Squeeze Reasoning mechanism is not sensitive to these choices, indicating that the generic graph reasoning behavior is the main reason for the observed improvements. We will describe two specific choices in the following part:

- Learned matrix: We follow the same settings in GloRe \([9]\), a simple choice of \(A\) is a \(1 \times 1\) convolutional layer that can be updated by the general backpropagation. Similar to previous works \([9, 45]\), we consider adopting the Laplacian matrix \((I - A_g)\) to propagate the node features over the graph, where the identity matrix \(I\) serves as a residual sum connection. In this setting, the Graph Reasoning can be formulated as follows:
\[
G_{\text{output}} = \sigma(W_g G_{\text{input}} (I - A_g)), \tag{5}
\]
where \(\sigma\) is the ReLU operation.

- Correlation matrix: Another choice is to adopt the self-attention mechanism for information exchange \([7, 40]\) where the correlation matrix (or dense affinity) is calculated by the projection of node feature itself, by which the reasoning process can be written as follows:
\[
G_{\text{output}} = \sigma(\rho_g(G_{\text{input}})[\phi_g(G_{\text{input}})^T \theta_g(G_{\text{input}})]), \tag{6}
\]
where \(\phi_g, \theta_g\) and \(\rho_g\) are three \(1 \times 1\) convolutions. \(\phi_g\) and \(\theta_g\) are named 'query' and 'key', respectively. The \(\rho_g\) operation here, named 'value', functions the same as the \(W_g\) in the 'Learned matrix' mechanism. The results generated by operations inside the \([\cdot]\) form the adjacency matrix \(A\). To be noted that, either reasoning process can be adopted in our framework and more detailed results can be referred to the experiment part.

3) Feature reconstruction: The final step is to generate the representation \(R\). To reconstruct the feature map, we first reshape the reasoned vector and multiply it with \(X\) to highlight different channels according to the input scene where \(\tilde{X} = XG_{\text{output}}\). Then we adopt another \(1 \times 1\) convolution layers \(W_R\) to project the \(\tilde{X}\) into original shape. Following the same idea of residual learning \([2, 7]\), we get the final output \(Y\) by adding original input \(X\). Then the feature map can be reconstructed as follows:
\[
Y = W_R \tilde{X} + X \tag{7}
\]
where \(W_R\) is a learnable linear transformation and \(Y\) is the final output feature.

C. Analysis and discussion

Relationship with Previous Operators: Compared with the Non-local block \([7]\), instead of affinity modeling on pixel-level, our SR extends the reasoning on channel dimension, which captures statistics of whole feature map space. Compared with the SE block \([71]\), our SR module captures more relational information and performs channel diffusion more efficiently than the fixed fully connected layers. Moreover, the experiment results show the advantages of our module.

Computation and memory analysis: Compared with previous self-attention methods \([7, 40, 41]\), we compare our module computation in Tab. \([1]\) which shows our module is both lightweight on both computation cost and affinity memory. The network linearly depends on the input resolution in terms of computation and has no relation with the input resolution in terms of affinity memory. More analysis can be found in the experimental part.

Discussion with EncNet \([15]\): Despite both EncNet and our SR both adopt the channel attention framework \([8]\), our method is different from EncNet. For squeezing process, EncNet learns an inherent dictionary to represent the semantic context of the dataset which is dataset dependent, while our method squeezes the entire feature map into one compact vector via learnable convolutions which is data dependent. The dataset dependent dictionary captures the class relation prior on specific dataset and thus it is hard for generalization. Our approach is more general and it can be a plug-in-play module.

In addition, we have proven this point on many other tasks including object detection, instance segmentation and panoptic segmentation. For reasoning process, our method divides the global feature into different groups. Each group represents a latent class and the information diffusion is achieved by one graph convolution layer while EncNet adopts SE-like architecture. Our divided and reasoning process shows better results than SENet in Table-X. Moreover, EncNet also uses a semantic loss to achieve better results while our method only uses the cross-entropy loss as naive FCN with much less tricks. Finally, we show the full advantages over different datasets under different settings over EncNet in the experimental part.
D. Network architecture

The proposed SR module can be easily incorporated into the existing CNN architectures. We detail our network design in the task of semantic segmentation and instance level segmentation.

Semantic Segmentation: We adopt the Fully Convolution Networks (FCNs) [1] as the backbone model. In particular, we choose ImageNet [19] pretrained ResNet [2], remove the last two down-sampling operations and adopt the multi-grid [27] dilated convolutions. We remove the last two down-sampling operations and use the dilatation convolutions instead to hold the feature maps from the last two stages. We remove the last two down-sampling operations and adopt the multi-grid [27] dilated convolutions. We remove the last two down-sampling operations and use the dilatation convolutions instead to hold the feature maps from the last two stages. Concretely, all the feature maps in the last three stages have the same spatial size. Following the same setting [10], [11], we insert our proposed module between two 3 × 3 convolution layers (both layers output D = 512 channels ), which are appended at the end of the FCN. Following [24], our model has two supervisions: one after the final output of our model while another at the output layer of Stage4 as auxiliary cross-entropy loss. For real-time segmentation models, we choose DF-seg models [72] as a baseline and we replace their head with our SR module.

Instance level segmentation: For instance segmentation and panoptic segmentation, we choose two-stage mask-rcnn-like architectures [4], [51]. We insert our module on the outputs of bottleneck in ResNet [2] layer4 for context modeling.

IV. Experiment

We verify the proposed module on four scene understanding tasks, including semantic segmentation, object detection, instance segmentation, panoptic segmentation and image classification. Our method outperforms several state-of-the-art methods on four benchmarks for semantic segmentation, including Cityscapes, ADE20K, Pascal Context and Camvid, with much less computation cost. Experiments on the other four vision tasks also demonstrate the effectiveness of the proposed module. All the experiments are under the same setting for each task and each dataset for a fair comparison.

A. Ablation Experiments on Cityscapes dataset

Experimental settings on ablation studies: We carry out detailed ablation studies and visual analysis on proposed approaches. We implement our method based on the PyTorch framework [73]. For the Cityscapes dataset, following the same settings in PSPNet [24] where momentum and weight decay coefficients are set to 0.9 and 5e-4 respectively, and “poly” learning rate schedule is used. For ablation studies, we choose ResNet-50 as the backbone where momentum and weight decay coefficients are set to 0.9 and 5e-4 respectively, and “poly” learning rate schedule is used which decays initial learning rate of 0.01 by multiplying (1 − iter / total_iter)0.9. Synchronized batch normalization is used [15]. For data augmentation, random cropping with size 769 and random left-right flipping are adopted. For the ablation studies, all models are trained by 50,000 iterations and evaluated by sliding-window crop inference. For the real time models, we use single scale full image inference.

Ablation on SR framework design: We first present a detailed analysis of each component of SR through ablation study and report results in Tab. III(a). Comparing with the baseline, all SR versions equipped with different components achieve considerable improvements. By switching different squeezing operations, we find Global Hadamard Pooling (GHP) performs consistently better than Global Average Pooling (GAP) across different used reasoning methods. Moreover, reasoning with Graph Convolutional Network and Self-Attention brings more improvements, even comparing with methods using global information from squeezing operation and further transformed by fully-connected layers (FC). Our segmentation models are trained under the best setting in this table.

Ablation on hyper-parameter settings: To select the best hyper-parameter K, we also carry out an ablation study on the number of groups. To control independent variables, we fix K M = C / 2, and only adjust K. We also explore the effect of channel reduction ratio with fixed K. The results are shown in Fig 4. From which, we can see that the selection of K doesn’t influence too much while reducing channel leads to inferior results. We set K = 16 and ratio to 2 as default for the remaining experiments.

Comparisons with context aggregation approaches: In Tab. IV we compare the performance of different context aggregation approaches, where SR achieves the best mIoU with ResNet-50 as the backbone. We give detailed and fair comparisons in terms of flops, parameters and memory cost. In particular, SR performs even better than all the non-local methods [7], [10], [15], [56], which aggregates long-range contextual information in a pixel-wise manner. This indicates the effectiveness of cross-channel relationships in building
TABLE IV
Comparison experiments using different context modeling methods on the Cityscapes validation set, where dilated FCN (ResNet-50) serves as the baseline method. The FLOPS and the Memory (Mem) are computed over the input image of size 768 × 768 × 3 and input feature map of size 96 × 96 × 2048. For models other than ASPP and PSP, the overhead of the 3 × 3 convolutions before and behind the modules are shown separately as the row +2 × 3 × 3. † means the relative overhead over those of +2 × 3 × 3. All the methods are evaluated under the same setting for the fair comparison.

| Method       | mIoU (%) | FLOPS      | Params | Mem      |
|--------------|----------|------------|--------|----------|
| dilated FCN | 74.8     | 241.05G    | 23.63M | 32.49M   |
| +ASPP [30]   | 77.4     | +148.37G   | +15.54M| +191.20M |
| +PSP [24]    | 77.2     | +174.09G   | +23.07M| +221.05M |
| +2 × 3 × 3   | -        | +10.74G    | +11.80M| +108.00M |
| +SE [8]      | 74.6     | 9.47M †    | 0.03M †| 18.88M † |
| +NL [7]      | 78.0     | 48.36G†    | 0.53M †| 865.52M† |
| +A2NL [41]   | 78.1     | 4.94G†     | 0.53M †| 183.32M† |
| +CGNL [36]   | 78.2     | 4.91G†     | 0.53M †| 201.94M† |
| +RCCA [10]   | 78.5     | 11.55G†    | 0.53M †| 394.28M† |
| +Encoding [15] | 77.5     | 12.31G†    | 0.59M †| 257.12M† |
| +ANN [11]    | 78.4     | 10.41G†    | 0.68M †| 421.12M† |
| +EMAU [16]   | 77.9     | 6.97G†     | 0.54M †| 132.64M† |
| +SR (GAP)    | 79.1     | 2.43G†     | 0.26M †| 82.31M†  |
| +SR (GHP)    | 79.9     | 3.64G†     | 0.40M †| 110.75M† |

Fig. 5. (a) Speed Comparison with Non-local and its variants. (b) Speed and Accuracy trade-off on Cityscapes Test Set for real time models. Best view it in color and zoom in.

TABLE V
Comparison on Cityscapes test set with state-of-the-art real-time models. For fair comparison, input size is also considered, and all models use single scale inference.

| Method       | InputSize | mIoU (%) | FPS  | Params |
|--------------|-----------|----------|------|--------|
| ESPNet [74]  | 1024×2048 | 70.5     | 1024 | 1536   |
| ESPNetv2 [75]| 1024×2048 | 71.3     | 1024 | 1536   |
| ERFNet [76]  | 1024×2048 | 69.5     | 1024 | 1536   |
| BiSeNet [78] | 1024×2048 | 71.3     | 1024 | 1536   |
| BiSeNet-ResNet [78] | 1024×2048 | 75.5     | 1024 | 1536   |
| BiSeNetception [39] | 1024×2048 | 74.8     | 1024 | 1536   |
| ICNet [60]   | 1024×2048 | 69.5     | 1024 | 1536   |
| DF-Seg [72]  | 1024×2048 | 73.0     | 1024 | 1536   |
| SwiftNet [65]| 1024×2048 | 75.5     | 1024 | 1536   |
| SwiftNet-E [65]| 1024×2048 | 76.5     | 1024 | 1536   |
| DFANet [67]  | 1024×2048 | 71.3     | 1024 | 1536   |
| CellNet [76] | 768 × 1536| 70.5     | 1024 | 1536   |
| DFV2(baseline) [72] | 1024×2048 | 74.8     | 1024 | 1536   |
| SRNet(DFv2)  | 1024×2048 | 77.5     | 1024 | 1536   |

Visualizations and analysis on learned node representation: Here, we provide visualization analysis on different channel activation on the reasoned feature map. From Fig. 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.

B. Experiments on Cityscapes in real-time settings

Experiment settings: Due to the efficiency of the proposed approach, we extend our method into real-time training settings. We mainly follow the DFANet [64]. The networks with SR head are trained with the same setting, where stochastic gradient descent (SGD) with batch size of 16 is used as optimizer, with momentum of 0.9 and weight decay of 5e-4. All models are trained for 50K iterations with an initial learning rate of 0.01. As a common practice, the “polo” learning rate policy is adopted to decay the initial learning rate by multiplying \((1 - \frac{iter}{total\ iter})^{0.9}\) during training. Data augmentation contains random horizontal flip, random resizing with scale range of \([0.75, 2.0]\), and random cropping with crop size of \(1024 \times 1024\). During inference, we use the whole picture as input to report performance. To be more specific, we replace PPM head [24] in DF-Seg-v2 [72] with our module. Tab. VI shows the results of our real-time model. Compared with baseline DFSeg [72], Our method has similar parameters but with more accurate and faster speed.

Comparisons with real time models on Cityscapes: We set a new record on best speed and accuracy trade-off in Cityscapes test set with 77.5 %mIoU and 70 FPS shown in Fig. 5 with full image resolution inputs \((1024 \times 2048)\), which indicates the practical usage of our method. During inference, we use the whole picture as input to report performance. Tab. VI shows the results of our real-time model. Compared with previous real time models, our method achieves the best speed and accuracy trade-off. Compared with baseline DFSeg [72], Our method has similar parameters but with more accurate and faster speed.

C. Comparisons with state-of-the-art methods in non-real-time setting:

In this section, we compare our method with state-of-the-art methods on four semantic segmentation benchmarks

maps, where SR significantly improves the consistency of features inside objects, which is also the reason for consistent semantic map prediction. After SR, the features inner the objects have similar color and clearer boundaries shown the second column in Fig 8.

Fig. 7. Visualization on predictions and feature maps: group-10 are more sensitive to the small objects like poles and cars while the second column shows that items focus on the reasoned feature map. From Fig 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.

Visualizations and analysis on learned node representation: Here, we give visualization analysis on different channel activation on the reasoned feature map. From Fig 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.

Visualizations and analysis on learned node representation: Here, we give visualization analysis on different channel activation on the reasoned feature map. From Fig 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.

Visualizations and analysis on learned node representation: Here, we give visualization analysis on different channel activation on the reasoned feature map. From Fig 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.

Visualizations and analysis on learned node representation: Here, we give visualization analysis on different channel activation on the reasoned feature map. From Fig 6, we can see that each item corresponds to some abstract conceptions in the image. For example, the third-row item attends on the trucks and the cars while the second column shows that items focus on the stuff and background, and the fourth column items in group-10 are more sensitive to the small objects like poles and boundaries.
Fig. 6. Visualization on learned group representation. We select the most salient channel from each node group. Such items capture specific concepts in the images. Best view in color and zoom in.

Fig. 7. Comparison of our results from cropped images where dilated-FCN with GAP- Squeeze operation as the baseline model. Best view it in color and zoom in.

Fig. 8. The input and the output feature maps of the SR module. They are projected from 512-d to 3-d by PCA. Best view it in color and zoom in.

Our ResNet-50 based model achieves 81.0% mIoU and outperforms DenseASPP [31] by 0.4% with much larger backbone [78], which shows the effectiveness of our method. After replacing stronger backbone Wider-ResNet [79], we achieve 83.3% mIoU with only fine annotated data, which outperforms previous state-of-the-art methods by a large margin. Note that we follow the G-SCNN [55] setting by using Deeplabv3+ based Wider-ResNet pretrained on Mapillary [80]. The detailed results are shown in Tab. VII for reference.

Results on Cityscapes: We train our model for 120K iterations using only the finely annotated data (trainval set), online hard negative mining is used following [40]. Multi-scale and horizontal flip testing is used as previous works [10]. Tab. VII(a) compares the results, where our methods achieves 82.2% mIoU and outperforms all previous state-of-the-art models by a large margin. In particular, our method is 0.7% mIoU higher than DANet [40], which uses non-local-like operator and is much efficient in both computation and memory cost due to the design of squeeze and reasoning.

Results on CamVid: is another road driving dataset. Camvid involves 367 training images, 101 validation images and 233 testing images with resolution of 960 × 720. We use a crop size of 640 and training with 100 epochs. The results are shown in Tab. VII(b). We report results with ResNet-50 and ResNet-101 backbone. With ResNet-101 as backbone, our method achieves 78.3% mIoU, outperforming the state-of-the-art approach [84] by a large margin (3.1%).

Results on Pascal Context: This dataset provides detailed semantic labels for the whole scenes [21]. It contains 4998 images for training and 5105 images for validation. We train the network for 100 epochs with a batch size of 16, a crop size of 480. For evaluation, we perform multi-scale testing with the flip operation, which boosts the results by about 1.2% in mIoU. Fig. 9 shows the results of our method and EncNet. Compared with EncNet [15], our method achieves better consistent results on the object inner parts, benefited from better reasoned features. Tab. VII(a) reports results on Pascal Context. With ResNet-101 as the backbone, our method
This is a more challenging scene

**TABLE VI**

| Method          | Backbone         | mIoU (%) |
|-----------------|------------------|----------|
| DFN [37]        | ResNet-101       | 97.3     |
| CFNet [39]      | ResNet-101       | 97.6     |
| DenseASPP [51]  | DenseNet-161     | 80.6     |
| G loreN [52]    | ResNet-101       | 80.9     |
| BAFNet [83]     | ResNet-101       | 81.4     |
| CCNet [10]      | ResNet-101       | 81.4     |
| ANNNet [11]     | ResNet-101       | 81.3     |
| DANet [40]      | ResNet-101       | 81.5     |
| RGNNet [17]     | ResNet-101       | 81.5     |
| DGMN [47]       | ResNet-101       | 81.6     |
| OCRNet [14]     | ResNet-101       | 81.8     |
| SRNet           | ResNet-50        | 81.0     |
| SRNet           | ResNet-101       | 82.2     |
| G-SCNN [55]     | Wider-ResNet-38  | 82.8     |
| SRNet           | Wider-ResNet-38  | 83.3     |

(a) Results on the Cityscapes test set. All methods use only finely annotated data.

| Method          | Backbone         | mIoU (%) |
|-----------------|------------------|----------|
| SegNet [71]     | VGG-16           | 60.1     |
| R TA [85]       | VGG-16           | 62.5     |
| BiSeg [62]      | ResNet-18        | 68.7     |
| PSPNet [24]     | ResNet-50        | 69.1     |
| SRNet           | ResNet-50        | 74.3     |
| DilatedNet [27] | ResNet-101       | 65.3     |
| Dense-Decoder [58] | ResNet-101   | 70.9     |
| BFP [63]        | ResNet-101       | 74.1     |
| VideoGCRF [84]  | ResNet-101       | 75.2     |
| SRNet           | ResNet-101       | 78.3     |

(b) Results on the CamVid test set.

achieves 54.7% in mIoU with multi-scale inference, surpassing state-of-the-art alternatives by a large margin. Additionally, using the ResNet-50 backbone, we achieve 50.8% mIoU, which also outperforms the previous work [40] under the same setting.

**TABLE VII**

| Method          | Backbone         | mIoU (%) |
|-----------------|------------------|----------|
| SRNet-ResNet101 | ResNet-50        | 81.0     |
| SRNet-ResNet101 | ResNet-101       | 82.2     |
| G-SCNN [55]     | Wider-ResNet-38  | 82.8     |
| SRNet           | Wider-ResNet-38  | 83.3     |

(a) Results on the Cityscapes test set. All methods use only finely annotated data.

**TABLE VIII**

| Method          | Backbone         | mIoU (%) |
|-----------------|------------------|----------|
| EncNet [86]     | ResNet-50        | 49.2     |
| DANet [40]      | ResNet-50        | 50.1     |
| SRNet           | ResNet-50        | 50.8     |
| EncNet [86]     | ResNet-101       | 51.7     |
| Ding et al. [54]| ResNet-101       | 51.6     |
| DANet [40]      | ResNet-101       | 52.6     |
| SGR [87]        | ResNet-101       | 52.5     |
| ANN [11]        | ResNet-101       | 52.8     |
| BAFPNet [83]    | ResNet-101       | 53.6     |
| EMANet [16]     | ResNet-101       | 53.1     |
| GFFNet [55]     | ResNet-101       | 54.2     |
| CFNet [88]      | ResNet-101       | 54.1     |
| SPNet [89]      | ResNet-101       | 54.5     |
| APCNet [90]     | ResNet-101       | 54.7     |
| SRNet           | ResNet-101       | 54.7     |

(a) Results on Pascal Context dataset.

| Method          | Backbone         | mIoU (%) |
|-----------------|------------------|----------|
| PSPNet [24]     | ResNet-50        | 42.78    |
| PSANet [42]     | ResNet-50        | 42.97    |
| UperNet [50]    | ResNet-50        | 41.55    |
| EncNet [86]     | ResNet-50        | 41.11    |
| GCUNet [45]     | ResNet-50        | 42.60    |
| SRNet           | ResNet-101       | 43.42    |
| PSPNet [24]     | ResNet-101       | 43.29    |
| PSANet [42]     | ResNet-101       | 43.77    |
| SAC [48]        | ResNet-101       | 44.30    |
| EncNet [86]     | ResNet-101       | 44.65    |
| GCUNet [45]     | ResNet-101       | 44.81    |
| ANN [11]        | ResNet-101       | 45.24    |
| SRNet           | ResNet-101       | 45.53    |

(b) Results under the common setting in [24]. The visible results are shown in Fig. 10. Compared to PSPNet [24], our method better handles inconsistent results and missing objects in the scene.

In Tab. VIII (b), Both results using ResNet-50 and ResNet-101 backbone are reported. As shown in Tab. VIII (b), Our method with ResNet-101 achieves the best results and comparable results with ResNet-50 backbone. Our method has less computation cost in the head part. Thus it results in an efficient
To verify our module’s generality, we further conduct experiments on MS COCO [18] for more tasks, including object detection, instance segmentation and panoptic segmentation. The trainval set has about 115k images, the minival set has 5k images. We perform training on trainval set and report results on minival set. For the first two tasks, our model is based on the state-of-the-art method Mask R-CNN [4] and its variants [32], [91]. For panoptic segmentation, we choose Panoptic FPN as our baseline [51]. We use open-source tools [92] to carry out all the experiments and report results on the MS COCO validation dataset. The GFlops are measured with 1200×800 inputs. Our models and all baselines are trained with the typical ‘1x’ training schedule and setting from the public mmmdetection [92] for all experiments on COCO.

Results on object detection and instance segmentation: Tab. IX(a) compares results of both object detection and instance segmentation with various backbone networks [93] and advanced method [91], where our method achieves consistently better performance on all backbones with much less computation cost compared with Non-Local blocks [7].

Results on panoptic segmentation: Panoptic Segmentation [5] uses the PQ metric to capture the performance for all classes (stuff and things) in a unified way. We use the Panoptic-FPN [51] as our baseline model and follow the standard settings in mmmdetection [92] for all experiments on COCO.

Visualization Results on COCO: Fig. 11 shows the qualitative results on COCO validation set. We use Mask-RCNN [94] with ResNet101 as the baseline model. The first two rows show our SR module can handle small missing objects (red boxes) while the last two rows show our method can also avoid false positives (blue boxes) in the scene.

E. Extension on ImageNet classification

We also perform experiments for image classification on ImageNet dataset [19], where SENet [8] is used as our baseline model. To be noted, we only verify the effectiveness and generalization of SR framework for classification task. Our model is designed by replacing fully-connected layer with our proposed graph reasoning module, where the global hadamard pooling is not used for both saving computation and fair comparison with SENet. All models are trained under the same setting, and results are shown in Table X. All networks are trained following the same strategy as [2] using Pytorch [73] with 8 GTX 1080Ti GPUs. In particular, cosine learning rate schedule with warm up strategy is adopt [25], and weight decay is set to 1e-4. SGD with mini-batch size 256 is used for updating weights. Top-1 and top-5 classification accuracy on validation set using single 224 × 224 central crop are reported for performance comparison. Due to the usage of cosine learning rate schedule [25], our baseline models on ImageNet are higher than the original paper [2], [8]. Compared with both SENet and CBAM [35], our SRNet improves the strong.

---

**TABLE IX**

EXPERIMENTS ON COCO DATASET: (a) DETECTION RESULTS ON THE COCO 2017 VALIDATION SET. R-X: RES-NET-X, X-101: RESNEXT-X-101, +SR: NON-LOCAL BLOCKS, +NL: NON-LOCAL BLOCKS. (b) PANOPTIC SEGMENTATION RESULTS ON THE COCO 2017 VALIDATION SET.

| Backbone | Detector | AP-box | AP-mask | GFlops |
|----------|----------|--------|---------|--------|
| R-50     | Mask-RCNN| 37.2   | 33.8    | 275.58 |
| R-50 +NL | 38.0     | 34.7   | +0.5   |
| R-50 +SR | 38.4     | 34.9   | +0.56  |
| X-101    | Mask-RCNN| 39.8   | 36.0    | 351.65 |
| R-101    | +NL      | 40.5   | 36.7    | +5.7   |
| R-101 +SR| 40.8     | 36.9   | +1.32  |
| X-101    | +SR      | 41.2   | 37.3    | 355.37 |
| X-101    | Cascaded-Mask-RCNN | 42.0 | 37.8 | +1.32 |
| X-101 +SR| 45.5     | 39.0   | +1     |

(a) Experiments on COCO Object Detection and Instance Segmentation using various baseline models.

| Method | Backbone | PQ | PQ (things) | PQ (stuff) | GFLOPs |
|--------|----------|----|-------------|------------|--------|
| Base   | ResNet-50| 39.0| 45.9        | 28.7       | 270.8  |
| +SR    | ResNet-50| 40.3| 47.2        | 29.9       | +0.56  |
| Base   | ResNet-101| 40.3| 47.5        | 29.5       | 346.87 |
| +NL    | ResNet-101| 40.8| 48.5        | 30.4       | +45.7  |
| +SR    | ResNet-101| 41.8| 48.7        | 31.3       | +1.32  |

(b) Experiments on COCO Panoptic Segmentation using PanopticFPN as baseline model.

**TABLE X**

EXPERIMENTS RESULTS ON IMAGENET. OUR METHOD ACHIEVES BETTER TOP-1 ACCURACY WITH LOWER PARAMETER AND COMPUTATION COST ON STRONG BASELINE SETTINGS. R50 MEANS RESNET50 AS BACKBONE WHILE R101 MEANS RESNET101 AS BACKBONE.

| Method    | Top-1 | Params | GFLOPs |
|-----------|-------|--------|--------|
| ResNet-r50| 77.5  | 25.56M | 4.122  |
| SENet-r50 | 77.8  | 28.09M | 4.130  |
| CBAM-r50  | 78.0  | 28.09M | 4.139  |
| SRNet-r50 | 78.1  | 25.64M | 4.127  |
| ResNet-r101| 78.7 | 44.55M | 7.849  |
| SENet-r101| 79.1  | 49.33M | 7.863  |
| SRNet-r101| 79.3  | 44.70M | 7.858  |
baseline SENet [8] by 0.3 in Top-1 accuracy with much less parameter and GFlops. Our method leads to higher accuracy with fewer parameters and FLOPs, which demonstrates both effectiveness and efficiency of the proposed method.

V. CONCLUSION

This paper proposes a novel Squeezing and Reasoning framework for highly efficient deep feature representation learning for the scene understanding tasks. It learns to squeeze the feature to a node graph space where each node represents an abstract semantic concept while both memory and computation costs are significantly reduced. Extensive experiments demonstrate that our method can establish considerable results on semantic segmentation while keeping efficiency with much less computation. It also verified to be effective on image classification task and better results over SENet. The further work can be exploring cross layer reasoning over entire network.

ACKNOWLEDGMENT

We gratefully acknowledge the support of SenseTime Research for providing the computing resources. Z. Lin is supported by NSF China (grant no.s 61625301 and 61731018) and Major Scientific Research Project of Zhejiang Lab (grant no.s 2019KB0AC01 and 2019KB0AB02). This work was also supported by the National Key Research and Development Program of China (No.2020YFB2103402).

REFERENCES

[1] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in CVPR, 2015.
[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016.
[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in NIPS, 2015.
[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in ICCV, 2017.
[5] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, “Panoptic segmentation,” in CVPR, 2019.
[6] W. Luo, Y. Li, R. Urtasun, and R. Zemel, “Understanding the effective receptive field in deep convolutional neural networks,” in NIPS, 2016.
[7] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in CVPR, 2018.
[8] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in CVPR, 2018.
[9] Y. Chen, M. Rohrbach, Z. Yan, Y. Shuicheng, J. Feng, and Y. Kalantidis, “Graph-based global reasoning networks,” in CVPR, 2019.
[10] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, “Cnet: Criss-cross attention for semantic segmentation,” arXiv preprint arXiv:1811.11721, 2018.
[11] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, “Asymmetric non-local neural networks for semantic segmentation,” in ICCV, 2019.
[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in ECCV, 2018.
[13] Y. Yuan and J. Wang, “Ocnet: Object context network for scene parsing,” arXiv preprint arXiv:1808.00916, 2018.
[14] Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” arXiv preprint arXiv:1909.11065, 2020.
[15] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, “Context encoding for semantic segmentation,” in CVPR, June 2018.
[16] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, “Expectation-maximization attention networks for semantic segmentation,” in ICCV, 2019.
[17] C. Yu, Y. Liu, C. Gao, C. Shen, and N. Sang, “Representative graph neural network,” in Proceedings of the European Conference on Computer Vision (ECCV), 2020.
[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in ECCV, 2014.
[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” IJCV, 2015.
[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in CVPR, 2016.
[21] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in CVPR, 2014.
[22] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Semantic understanding of scenes through the ADE20K dataset,” arXiv preprint arXiv:1608.05442, 2016.
[23] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” Pattern Recognition Letters, 2008.
[24] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in CVPR, 2017.
[25] T.-Y. Lin, A. RoyChowdhury, and S. Maji, “Bilinear cnn models for fine-grained visual recognition,” in ICCV, 2015.
[26] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, “Attention augmented convolutional networks,” in ICCV, 2019.
[27] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in ICLR, 2016.
[28] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected CRFs,” in ICLR, 2015.
J. Li, and A. Wong, “Squeeze-and-attention networks for semantic segmentation,” in *CVPR*, 2018.

Z. Li, Z. Huang, Y. Q. Zhao, Z. Chen, and Y. Yang, “Deep residue network for semantic segmentation,” in *CVPR*, 2018.

Z. Zhang, Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, Z. Li, W.-S. Zheng, J. Li, and A. Wong, “Squeeze-and-attention networks for semantic segmentation,” in *CVPR*, 2020.

Y. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu, “Dual attention network for scene segmentation,” in *arXiv preprint arXiv:1809.02983*, 2018.

Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, “A2-nets: Double attention networks,” in *NIPS*, 2018.

H. Zhou, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, and J. Jia, “Peanut: Pointwise spatial attention network for scene parsing,” in *ECCV*, 2018.

X. Li, Z. Zhong, J. Yu, Y. Yang, Z. Lin, and H. Liu, “Expectation-maximization attention networks for semantic segmentation,” in *ICCV*, 2019.

T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal Processing magazine*, 1996.

Y. Li and A. Gupta, “Beyond grids: Learning graph representations for visual recognition,” in *NIPS*, 2018.

X. Li, Y. Yang, Q. Zhao, T. Shen, Z. Lin, and H. Liu, “Spatial pyramid based graph reasoning for semantic segmentation,” in *CVPR*, 2020.

Z. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan, “Scale-adaptive convolutions for scene parsing,” in *ECCV*, 2017.

O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.

T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, “Unified perceptual parsing for scene understanding,” in *ECCV*, 2018.

A. Kirillov, R. Girshick, K. He, and P. Dollar, “Panoptic feature pyramid networks,” in *CVPR*, 2019.

G. Lin, A. Milan, C. Shen, and I. D. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” in *CVPR*, 2017.

X. Li, Z. Houlong, H. Lei, T. Yunhai, and Y. Kuiyuan, “Gff: Gated fully fusion for semantic segmentation,” in *AAAI*, 2020.

H. Ding, X. Jiang, B. Shuai, A. Qun Liu, and G. Wang, “Context contrasted feature and gated multi-scale aggregation for scene segmentation,” in *CVPR*, 2018.

T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, “Gated-scnn: Gated shorcut cnns for semantic segmentation,” in *ICCV*, 2019.

S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, “Conditional random fields as recurrent neural networks,” in *ICCV*, 2015.

T.-W. Ke, J.-J. Hwang, Z. Liu, and S. X. Yu, “Adaptive affinity fields for semantic segmentation,” in *ECCV*, 2018.

B. Bilinski and B. Leibe, “Dense decoder shortcut connections for single-pass semantic segmentation,” in *CVPR*, 2018.

L.-C. Chen, M. Collins, Y. Zhu, G. Papandrou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, “Searching for efficient multi-scale architectures for dense image prediction,” in *NIPS*, 2018.

C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation,” in *CVPR*, 2019.

H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “Lcnet for real-time semantic segmentation on high-resolution images,” in *ECCV*, September 2018.

C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, “Bisenet: Bilateral segmentation network for real-time semantic segmentation,” in *ECCV*, 2018.

X. Li, A. You, Z. Zho, H. Zhao, M. Yang, K. Yang, and Y. Tong, “Semantic flow for fast and accurate scene parsing,” in *CVPR*, 2020.

H. Li, P. Xiong, H. Fan, and J. Sun, “Difnet: Deep feature aggregation for real-time semantic segmentation,” in *CVPR*, June 2019.

M. Orsic, A. Ratulovic, and A. Crul, “Semantic video segmentation by gated recurrent flow propagation,” in *CVPR*, 2018.

X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, “Deep feature flow for video recognition,” in *CVPR*, 2017.

Y. Li, J. Shi, and D. Lin, “Low-latency video semantic segmentation,” in *CVPR*, 2018.

T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *ICLR*, 2017.

V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *TPAMI*, 2017.

X. Li, Y. Zhou, Z. Pan, and J. Feng, “Partial order pruning: for best accuracy trade-off in neural architecture search,” in *CVPR*, 2019.

A. Paszke, G. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NeurIPS workshops*, 2017.

S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, “Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation,” in *CVPR*, 2019.

S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, “Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network,” in *CVPR*, June 2019.

E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, “Erfnet: Efficient residual factorized convnet for real-time semantic segmentation,” *IEEE Trans. Intelligent Transportation Systems*, pp. 263–272, 2018.

V. V. Krasov, H. Chen, C. Liu, Z. Zhang, and I. Reid, “Fast neural architecture search of compact semantic segmentation models via auxiliary cells,” in *CVPR*, June 2019.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *CVPR*, 2017.

Z. Wu, C. Shen, and A. van den Hengel, “Wider or deeper: Revisiting the resnet model for visual recognition.” *arXiv preprint arXiv:1611.08090*, 2016.

G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, “The maipulary vistas dataset for semantic understanding of street scenes,” in *ICCV*, 2017.

P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, “Understanding convolution for semantic segmentation,” in *WACV*, 2018.

S. Y. Falong Shen, Gan Rui and G. Zeng, “Semantic segmentation via structured patch prediction, context crf and guidance crf,” in *CVPR*, 2017.

H. Ding, X. Jiang, A. Q. Liu, N. Magenat-Thalmann, and G. Wang, “Boundary-aware feature propagation for scene segmentation,” in *ICCV*, 2019.

S. Chandra, C. Couprie, and I. Kokkinos, “Deep spatio-temporal random fields for efficient video segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun, “Efficient uncertainty estimation for semantic segmentation in videos,” in *The European Conference on Computer Vision (ECCV)*, September 2018.

H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, “Context encoding for semantic segmentation,” in *CVPR*, 2018.

X. Liang, Z. Hu, H. Zhang, L. Lin, and E. P. Xing, “Symbolic graph convolutional networks for biomedical image segmentation,” in *TPAMI*, 2015.

M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, “In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images,” in *CVPR*, June 2019.

R. Galde, V. Jampani, and P. V. Gehler, “Semantic video cnns through representation warping,” in *ICCV*, 2017.

B. Nilsson and C. Ranesesceu, “Semantic video segmentation by gated recurrent flow propagation,” in *CVPR*, 2018.

J. He, Z. Deng, L. Zhou, Y. Yang, and Y. Qiao, “Adaptive pyramid context network for semantic segmentation,” in *CVPR*, 2019.
[91] Z. Cai and N. Vasconcelos, “Cascade r-cnn: Delving into high quality object detection,” in CVPR, 2018.

[92] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, “mmdetection,” [https://github.com/open-mmlab/mmdetection], 2018.

[93] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in CVPR, 2017.

[94] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask r-cnn,” in ICCV, 2017.

[95] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in CVPR, June 2019.