Exploring Scale-Aware Features for Real-Time Semantic Segmentation of Street Scenes

Kaige Li, Qichuan Geng, and Zhong Zhou

***Abstract*— Real-time semantic segmentation of street scenes is an essential and challenging task for autonomous driving systems, which needs to achieve both high accuracy and efficiency. Moreover, numerous objects and stuff at different scales in street scenes further increase the difficulty of this task. To address this challenge, we develop a lightweight and high-accuracy network termed Scale-Aware Network (SANet), which aims to selectively aggregate multi-scale features while maintaining high efficiency. In SANet, we first design a Selective Context Encoding (SCE) module, which considers the intrinsic differences of various pixels to selectively encode private contexts for each pixel, thus learning more desirable contextual features while reducing redundancy. With the context embedding in hand, we then design a Selective Feature Fusion (SFF) module to recursively fuses them with multiple features at different levels or scales to generate scale-aware features, where each feature map contains scale-specific information. Extensive experiments on challenging street scene datasets, i.e., Cityscapes and CamVid, illustrate that our SANet achieves a leading trade-off between segmentation accuracy and speed. Concretely, our method yields 78.1% mIoU at 109.0 FPS on the Cityscapes test set and 77.2% mIoU at 250.4 FPS on the CamVid test set. Code will be available at https://github.com/kaigelee/SANet.**

***Index Terms*— Street scene understanding, real-time semantic segmentation, selective context encoding, selective feature fusion.**

I. Introduction

SEMANTIC segmentation is one of the fundamental tasks of computer vision, which aims to label each pixel in an image with a predefined category. It is a crucial step to realize an in-depth understanding of urban street scenes, and has been extensively applied in various intelligent transportation systems [1], such as video surveillance, stereo reconstruction, and autonomous driving [2], [3], [4]. In general, these tasks require high accuracy and real-time response.

Street scene images (e.g., Cityscapes [5]) contain extensive multi-scale data [3], [6]. Various objects of different scales, such as cars, persons, and roads, or the same object with different distances, are multi-scale inputs. Hence, street semantic segmentation requires multi-scale inference because particular objects are best predicted only at specific scales [4], [7]. Therefore, numerous methods have been proposed to introduce multi-scale features [8], [9], [10]. Nevertheless, they rely heavily on large-scale backbone networks (e.g., ResNet-101 [11]), which leads to huge computation costs. Consequently, these accuracy-oriented methods are prohibitive for real-time applications, such as autonomous driving systems. To this end, real-time methods, such as ENet [12], ERFNet [13], FRNet [14] and FASSD-Net [15], have become the focus of research. They mainly accelerate network inference by designing lightweight networks [13], [16], [17]. However, lightweight networks struggle to extract abundant spatial and contextual information, resulting in a great decrease in accuracy [7]. Fig. 1 shows the speed and accuracy comparison of some state-of-the-art (SOTA) methods. Obviously, making a better speed-accuracy trade-off is still a challenging problem.

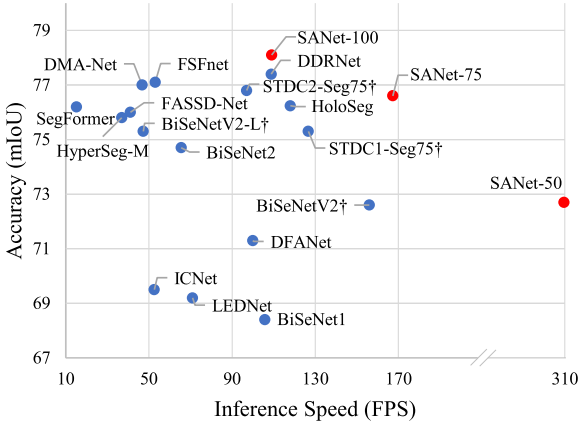


Fig. 1. Inference speed and accuracy (reported) performance comparison on the Cityscapes test dataset [5]. Our SANet achieves a better speed/accuracy trade-off.

Recently, most real-time methods employ U-shape [18] or multi-branch structures [3], [7], [19], [20] with context modules to improve performance. Specifically, for context modeling, they mostly adopt modified Pyramid Pooling or Atrous Convolution modules. However, they either lack the modeling of multi-scale (e.g., local and global) contexts [3], [7], or ignore the inherent differences of different pixels (e.g., scale properties) [7], [20], [21]. Unfortunately, the demands of each pixel for contexts are discrepant and dynamic according to the input. Thus, the per-pixel unified modeling

Manuscript received 19 July 2022; revised 20 December 2022, 14 May 2023, 28 July 2023, and 26 September 2023; accepted 30 October 2023. This work was supported by the National Natural Science Foundation of China under Grant 62272018. The Associate Editor for this article was J. Li. *(Corresponding author: Zhong Zhou.)*

Kaige Li is with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China (e-mail: lkg@buaa.edu.cn).

Qichuan Geng is with the Information Engineering College, Capital Normal University, Beijing 100048, China (e-mail: gengqichuan1989@cnu.edu.cn).

Zhong Zhou is with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China, and also with the Zhongguancun Laboratory, Beijing 100191, China (e-mail: zz@buaa.edu.cn).

Digital Object Identifier 10.1109/TITS.2023.3330498

methods [3], [7], [21] will introduce undesirable contexts and thus cause prediction confusion. For multi-scale feature extraction, a common way [14], [18], [19], [22] is to fuse high-level and low-level features by concatenating or summing them. However, indiscriminately fusing features [18], [19] weakens information propagation effectiveness due to the representation gaps between features at different levels and the different demands for spatial details and semantics at different positions. This issue is more obvious in street scenes, which usually contain numerous objects with different scales. Given the above issues, we propose a novel Scale-Aware Network (SANet), which constructs features containing scale-specific information for each position via selective context encoding and feature fusion, achieving high-performance real-time semantic segmentation of street scenes.

To collect rich contexts, previous methods adopt either Pooling-based [7], [8], [21] or Atrous convolution-based [9], [15], [21] ways. However, they aggregate homogeneous contexts for each pixel in a fixed manner, which will cause context misalignment and bring unexpected prediction errors. Specifically, the local context is essential for the edges or small objects, while the global context is beneficial to the large objects or dominant stuff, and encoding inappropriate contexts for a pixel may cause ambiguity. For example, global statistics are easily biased towards large objects that occupy more pixels. Hence, encoding global contexts for small objects will lead to over-smoothing results for them as the local information is overwhelmed by dominant global contexts. To this end, we propose a Selective Context Encoding (SCE) module to perform more flexible context encoding. Our SCE first predicts the demand coefficients of each pixel for different scale contexts and then customizes its private contexts according to these coefficients. Compared to previous methods, we encode the contexts across all pixels differently, considering their inherent differences, which they ignore. Despite being a minor change, SCE greatly improves the network capacity with minimal computation.

Further, we investigate how to fuse multi-level features more reasonably. Existing methods [20], [23], [24] propose to reweight fused features by generating channel-wise weights via global pooling and a fully connected layer. SKNet [25] also generates the channel weights to reweight the multiple-input features before summing them, but SKNet focuses on fusing features of different scales rather than different levels. Despite achieving better performance, the above feature fusion method using a fully connected way to generate the weight vectors is inefficient and has high model redundancy. Concretely, the activation and inhibition of a feature are correlated with its neighboring features [26], so it is unnecessary to aggregate information from all channels when computing attention weights. Based on this insight, we propose a Selective Feature Fusion (SFF) module that performs adaptive feature fusion by considering every channel and its k neighbors in adjacent levels to generate cross-level weights. With this local interaction, our SFF reduces the model complexity from $\mathcal{O}\left(C^2\right)$ to $\mathcal{O}(1)$. In SANet, SFF selectively fuses the favorable feature maps at each level with those at other levels to produce scale-aware features with only marginal extra computation.

Extensive experiments show the effectiveness of our method, and it performs excellently against other SOTA real-time methods, as shown in Fig. 1. In summary, our main contributions are as follows:

1) We propose a Selective Context Encoding module to encode pixel-sensitive contexts by generating the context demand coefficients of each pixel. SCE obtains clearer contexts and avoids prediction confusion caused by the falsely introduced contexts.
2) We propose a Selective Feature Fusion module with cross-level local feature interaction, which generates attention weights by considering only each feature and its cross-level neighbors to calibrate features before fusion. SFF brings clear improvement while increasing minimal model complexity.
3) With two proposed modules, SANet can be more compact and achieves a SOTA trade-off between accuracy and speed on two challenging street scene benchmarks. Specifically, we obtain 78.1% mIoU at 109.0 FPS on the Cityscapes [5] test set and 77.2% mIoU at 250.4 FPS on the CamVid [27] test set.

II. Related Work

A. Real-Time Semantic Segmentation

Existing semantic segmentation methods based on Fully Convolutional Networks (FCNs) [28] achieved promising performance on various benchmarks [8], [10], [29]. Nevertheless, they heavily rely on dilated backbone [9], which generally suffers from a huge computational load and cannot be directly used in real-time applications [13], such as automotive driving [7], [30]. For example, PSPNet [8] only achieves 0.78 FPS for a 1024×2048 input, which is problematic for real-time scenarios. To improve efficiency, numerous real-time methods have been proposed. First, since the pre-trained lightweight backbones (e.g., ResNet-18 [11]) can provide decent feature encoding capabilities, many methods [7], [18], [23], [30] adopt them to address real-time segmentation tasks. For example, BiSeNet [23] proposes a two-branch framework to separately encode semantics and preserve spatial details through a lightweight backbone and several convolutional layers. Like ENet [12] and ERFNet [13], BiSeNet also reduces the overall computational complexity by resizing the input. Reference [20] proposes a modified MobileNetV2 [31] and a distinctive ASPP to effectively address the multi-scale problem of semantic segmentation in street scenes.

Second, instead of utilizing the pre-trained backbone, some methods tend to redesign a lightweight network using convolution factorization, such as depth-wise separable convolution (DW Conv), group convolution, and factorized convolution. Fast-SCNN [32] uses DW Conv for real-time segmentation. LEDNet [16] adopts channel split and shuffle operations to reduce computational complexity. ERFNet [13] and FRNet [14] employ factorized convolutions to build compact segmentation networks. Although these redesigned networks mostly have smaller parameters, they cannot benefit from the ImageNet [33] pretraining. Moreover, the actual speed of factorization convolution is often lower than standard

convolution due to memory access cost or other reasons [34]. Therefore, this paper adopts the pre-trained ResNet-18 as the backbone and focuses on innovative decoders to improve performance.

Thirdly, some methods utilize Knowledge Distillation (KD) [35], [36], [37], [38] to lighten the network. They transfer knowledge from a large-scale network to a lightweight model to improve performance. Liu et al. [35] propose to use pairwise and holistic distillation to distill structured knowledge from large networks to compact ones. TransKD [37] proposes a Transformer-based Knowledge Distillation (TransKD) framework, which distills both feature maps and patch embeddings from large teacher transformers to compact student transformers. KD has great potential to further optimize the model as a post-processing technique. However, we mainly focus on building a novel lightweight network, SANet. Without KD, our SANet still outperforms these KD-based methods.

Finally, witnessing the success of Transformer in vision tasks, several methods also applied it to semantic segmentation. For example, Trans4Trans [39] designs a Transformer-based encoder and develops a Transformer Parsing Module (TPM) as a decoder to generate the final output. SegFormer [40] unifies a hierarchical Transformer encoder with a lightweight multilayer perceptron (MLP) decoder to perform semantic segmentation. In this paper, we mainly focus on how to design an efficient decoder based on pre-trained lightweight convolutional neural networks.

B. Context Encoding

Contexts can provide rich scene category priors, thus improving semantic segmentation performance. DMA-Net [7] uses a Global Context Block (GCB) to encode global contextual information. PSPNet [8] exploits the Pyramid Pooling Module (PPM) to partition features into different regions to encode multi-scale contexts. ASPP [9] and DenseASPP [41] adopt a series of atrous convolution layers to harvest different levels of contexts. Dong et al. [20] design a Distinctive ASPP to exploit multi-scale information more effectively. DANet [29] and CCNet [42] introduce self-attention mechanism [43] to model long-range context dependency. Further, FASSD-Net [15] propose a Dilated Asymmetric Pyramidal Fusion Module (DAPF) to reduce the computational load of ASPP. DDRNet [21] proposes a Deep Aggregation Pyramid Pooling Module (DAPPM) to capture context information by combining feature aggregation with pyramid pooling. However, since these methods either ignore the multi-scale contexts [7], [29] or ignore the inherent differences of different pixels [15], [20], [21], they may adversely affect segmentation of objects of different sizes.

C. Feature Fusion

DFANet [19] and SwiftNet [18] fuse the features from different levels or scales to refine segmentation predictions. However, depending on the input, the demand for information contained in multiple feature maps is dynamic and different. Simply concatenating [19] or summing [18] multiple features is very naive. To fuse multi-level features more reasonably, inspired by SENet [44], BiSeNet [23], MsNet [3] and MFNet [4] propose to use global average pooling to generate a channel-wise weight vector to recalibrate the fused features. SKNet [25] also uses a SENet-like [44] way to generate channel-wise weights to calibrate multiple inputs before summing them. Further, FSFNet [30] proposes to adaptively fuse different level features by generating weight maps in both spatial and channel-wise to ease the spatial information loss caused by global pooling. However, it inevitably incurs more computation. In this paper, we find that SENet-like ways [23], [25] are unsuitable for multi-level feature fusion because they consider all levels of information to build attention weights and inevitably introduce redundant information. Thus, we introduce a local cross-level interaction way to generate attention weights, which improves performance with minimal computation.

III. Methodology

In this section, we first introduce the complete pipeline of the proposed SANet for real-time semantic segmentation of street scenes, then we elaborate on the details of the two main components for learning scale-aware features.

A. Overview

Given a street scene image, stuff or objects, are diverse in many ways, especially in scales. Previous methods [18], [21] usually overlook that different pixels have different demands for spatial details or semantic contexts. Therefore, they suffer from the intractable multi-scale issue, which will be even more pronounced in street scenes that contain considerable objects of varying scales. To this end, we explicitly explore the scale-aware features, where each feature map contains scale-specific information, via selective context encoding and feature fusion. We instantiate our method into two modules: Selective Context Encoding (SCE) module and Selective Feature Fusion (SFF) module, and build a Scale-Aware Network (SANet) based on them. The overall architecture of SANet is illustrated in Fig. 2. SANet uses an asymmetric encoder-decoder architecture, where the encoder employs a backbone network to extract features, and the decoder employs the proposed modules to restore spatial resolution progressively.

B. Selective Context Encoding Module

Contexts can provide rich prior information of scene images to improve performance. Pooling-based methods [8], [21], [45], [46] are very effective and efficient in aggregating contextual information. However, they encode contexts for each pixel homogeneously, which inevitably leads to pixel-context mismatch. Usually, the global context has semantic guidance for dominant stuff and large objects (e.g., “road” or “truck”), while small objects and edges (e.g., “traffic sign” or “pole”) prefer the local context. Intuitively, encoding suitable contexts for each pixel will significantly improve the modeling ability of the network. Based on this intuition, we propose a SCE module to encode contexts selectively for each pixel by predicting its context demand coefficients.

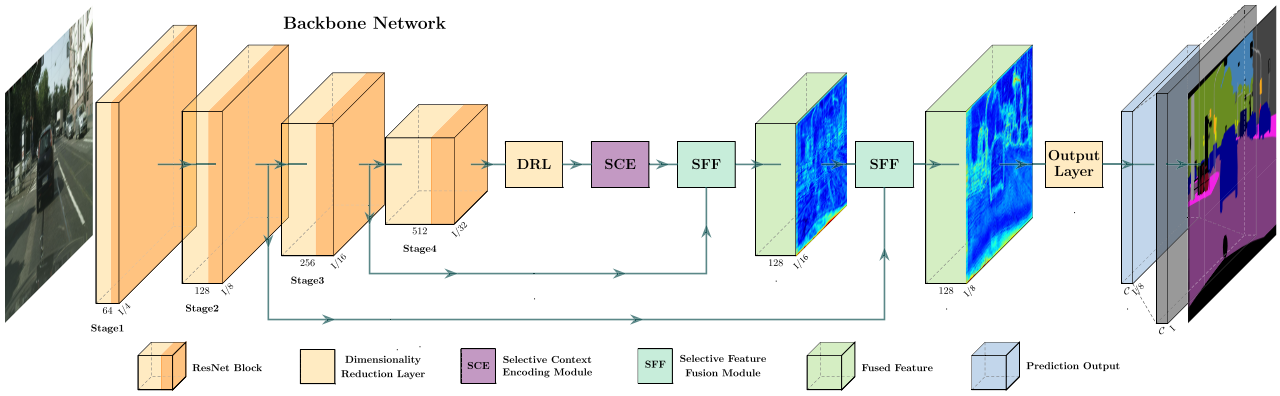


Fig. 2. Overview of our SANet. Given an input image $I \in \mathbb{R}^{3\times h\times w}$, we first employ the backbone network to extract features of different stages. Secondly, we feed features of the last stage into a dimensionality-reduction layer (DRL) with a reduction ratio ρ ($= 4$ by default) to decrease the computational cost of succeeding layers. Then a Selective Context Encoding (SCE) module is applied to encode the semantic contexts for each pixel and enhance its discrimination. Afterward, the enhanced features are input into two consecutive Selective Feature Fusion (SFF) modules to remedy spatial details and generate scale-aware features. Finally, we adopt several convolutional layers and an upsampling layer to produce the final segmentation results. In this figure, "I/N" means the feature size is $\frac{1}{N}$ of the input size. Best viewed in color.

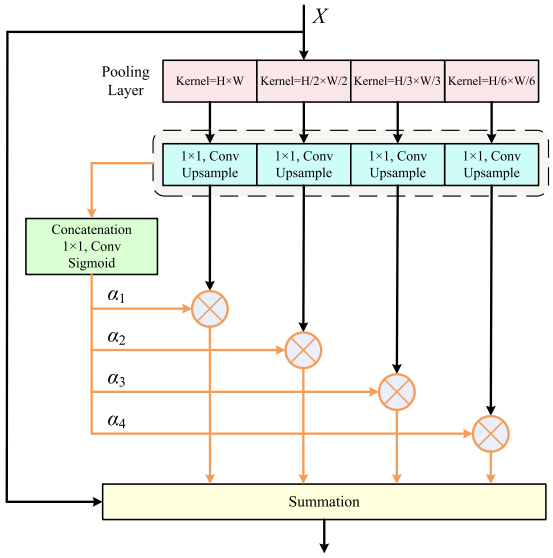


Fig. 3. The details of the Selective Context Encoding Module. Orange arrows indicate the unique processes in SCE. For simplicity, we omit the BN and ReLU layers in the figure. Best viewed in color.

Before introducing SCE, we first introduce a Reduced Context Encoding (RCE) module based on the framework of PPM [8] to capture contexts more efficiently. RCE reduces intermediate, output channels, and pooling output size, and replaces concatenation with summation. As shown in Fig. 3, given an input feature map $\mathbf{X} \in \mathbb{R}^{C\times H\times W}$, RCE first uses spatial pyramid pooling (SPP) layer to collect levels of the information under four different pyramid scales. Afterwards, RCE performs convolution and upsampling operations on the output features. For convolution operations, the kernel size is 1×1, and the number of output channels is smaller than the number of input channels. Finally, RCE performs element-wise summation between the original input $\mathbf{X}$ and four upsampled features to produce the final output. Compared with PPM, RCE is more efficient and suitable for real-time applications. More importantly, RCE promotes the performance greatly with less extra computation.

Since both RCE and previous methods [8], [21] ignore the scale difference between pixels, it is difficult for them to handle the multi-scale issue in street scene segmentation well. Thus, on the basis of RCE, we further develop a SCE module, which considers the context demand coefficients of different pixels to construct more distinctive contexts. As in Fig. 3, the upsampled pooled contextual features, are concatenated and fed into a dedicated fully convolutional head to generate a pixel-wise context demand coefficients $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4] \in \mathbb{R}^{4\times H\times W}$, formulated as:

$$\alpha = 1 + \sigma\left(\mathcal{W}_2\delta(\mathcal{W}_1\mathbf{E})\right) \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{(4\times C)\times H\times W}$ denotes concatenated upsampled pooled features, σ and δ refer to the $Sigmoid$ and ReLU function, respectively. $\mathcal{W}_1 \in \mathbb{R}^{C\times 4C\times 1\times 1}$ and $\mathcal{W}_2 \in \mathbb{R}^{4\times C\times 1\times 1}$ denote 1×1 convolutional layer. Then, we multiply each context $\mathbf{E}_i \in \mathbb{R}^{C\times H\times W}$ by the demand coefficients $\alpha_i \in \mathbb{R}^{H\times W}$ to calibrate the contexts. Finally, we perform an element-wise summation between the original feature map X and the calibrated contexts to derive the final output. Formally, the SCE module can be formulated as:

$$\mathbf{F} = \mathbf{X} + \textstyle\sum_{i=1}^{4} \alpha_i \cdot P^{n_i}(\mathbf{X}) \tag{2}$$

where $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$ is the refined output of SCE, $P^{n_i}(\cdot)$ represents the SPP layer, where the superscript n_i indicates the height (or width) of the output size of the pooling layer. We set $n \in [1, 2, 3, 6]$ by default.

See (2), SCE enables each pixel to integrate specific contexts according to its corresponding coefficients α, which allows SCE to mitigate multi-scale issue effectively.

C. Selective Feature Fusion Module

Multi-level feature fusion can recover lost spatial details and obtain multi-scale information, which makes it possible to handle the huge-scale variations of objects or stuff in the scene. Unfortunately, simply combining (e.g., summation [18], [47] or concatenation [22]) multi-level features has been proved

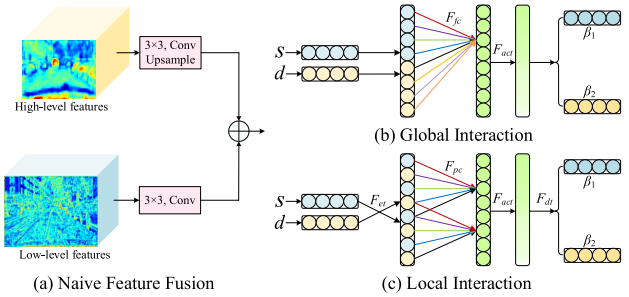


Fig. 4. Illustration of different feature fusion methods. (a) Naive fusion method. (b) Weight generation based on global interaction. (c) Weight generation based on local interaction. Best viewed in color.

to be very naïve [20], [24]. As shown in Fig. 4 (a), high-level features contain more global semantics while low-level features contain abundant spatial details. Naive Feature Fusion (NFF) suffers from the semantic representation gap among them. Here, we analyze in detail the advantages and disadvantages of existing feature fusion architectures [24], [30] and propose a Selective Feature Fusion (SFF) module to perform selective fusion of different feature maps.

Given a high-level feature $\mathbf{S} \in \mathbb{R}^{C' \times H' \times W'}$ and low-level feature $\mathbf{D} \in \mathbb{R}^{C \times H \times W}$, NFF first uses convolution and bilinear interpolation to unify their size to $C'' \times H \times W$. Then, NFF simply sums or concatenates them to obtain multi-scale features, as shown in Fig. 4 (a). To improve the feature fusion effect, we integrate ideas from previous methods (e.g., SKNet [25] and BiSeNet [24]) and carefully modify them to build SFF module, which can better adapt to real-time segmentation tasks. Specifically, SFF first uses global pooling $\mathbf{F}_{gp}$ to process the unified $\mathbf{S}$ and $\mathbf{D}$, respectively, to obtain global statistics $\mathbf{s}, \mathbf{d} \in \mathbb{R}^{C''}$:

$$\mathbf{s}, \mathbf{d} = \mathbf{F}_{gp}(\mathbf{S}), \mathbf{F}_{gp}(\mathbf{D}) \tag{3}$$

where $\mathbf{F}_{gp}$ denotes Generalized Mean pooling (GeM) instead of global average pooling (GAP) in the previous methods [24], [25]. This is because GAP performs spatially uniform pooling across all locations, which suffers from non-informative image regions that play a negative role in global modeling. In contrast, we use GeM to highlight favorable information and compress irrelevant information within each feature map, thereby obtaining more comprehensive global representation. GeM can be expressed as:

$$\mathbf{F}_{GeM}(\mathbf{X}) = \left(\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{X}(i,j)^p \right)^{\frac{1}{p}} \tag{4}$$

Then, SFF concatenates $\mathbf{s}$ and $\mathbf{d}$ to obtain $\mathbf{g}$, and inputs $\mathbf{g}$ into two fully-connected layers $\mathbf{F}_{fc}$ and the activation function $\mathbf{F}_{act}$ to generate the channel-wise weight vectors β:

$$\beta = \mathbf{F}_{act}(\mathbf{F}_{fc}(\mathbf{g})) = 1 + tanh(\mathcal{W}_\psi \delta(\mathcal{B}(\mathcal{W}_\theta \mathbf{g}))) \tag{5}$$

where $\beta = [\beta_1, \beta_2] \in \mathbb{R}^{2C'' \times 1}$, denote weight vectors for S and D, F_{fc} contains two fully connected layers $\mathcal{W}_\theta \in \mathbb{R}^{\frac{C''}{r} \times 2C''}$ and $\mathcal{W}_\psi \in \mathbb{R}^{2C'' \times \frac{C''}{r}}$, $\mathcal{B}$ represents Batch Normalization. r denotes reduction ratio and we set $r = 16$ by default. Here we use $1 + tanh$ function instead of $sigmoid$ since its wider range of values enables it to simulate more diverse inter-feature relationships, e.g., cooperation (if both β_1 and β_2 are greater than 1) and competition (if β_1 is greater than 1 and β_2 is less than 1, and vice versa). Finally, the low-level and high-level features are weighted fusion:

$$\mathbf{O} = \beta_1 \cdot \mathbf{S} + \beta_2 \cdot \mathbf{D} \tag{6}$$

Despite achieving better performance (as shown in Sec. IV-C), the current SFF module uses fully connected layers for global interaction to predict channel-wise weights, as shown in Fig. 4 (b). However, inspired by the lateral inhibition in neurobiology [26], i.e., the activation and inhibition of a feature is correlated with its neighboring features, the fully connected way is redundant and introduces noise and unnecessary computation. That is, since our goal is only to obtain the weights of the corresponding channel features (i.e., two feature maps for the n-th channel in the low- and high-level features), performing local interaction between cross-level features can yield beneficial relationships. Based on this insight, we propose a local interaction strategy to further improve the SFF module. In the following, SFF adopts this strategy by default, if not mentioned.

As shown in Fig. 4 (c), SFF first performs feature entanglement ($\mathbf{F}_{et}$) on $\mathbf{g}$ to ensure that cross-level corresponding channel features are adjacent:

$$\mathbf{e} = \mathbf{F}_{et}(\mathbf{s}, \mathbf{d}) = \{\mathbf{s}_0, \mathbf{d}_0, \ldots, \mathbf{s}_i, \mathbf{d}_i, \ldots \mathbf{s}_{C''}, \mathbf{d}_{C''}\} \tag{7}$$

where $\mathbf{e} \in \mathbb{R}^{2 \times C''}$. Then, SFF adopts partially connected layer ($\mathbf{F}_{pc}$) and activation function to calculate the weight of $\mathbf{e}_i$ by only considering interaction between $\mathbf{e}_i$ and its k neighbors:

$$\S_i = 1 + tanh\left(\sum_{j=1}^{k} w^j \mathbf{e}_i^j \right), e_i^j \in \Omega_i^k \tag{8}$$

where $\S \in \mathbb{R}^{2 \times C''}$, Ω_i^k denotes the set of k adjacent channels of $\mathbf{e}_i$. Note that $\mathbf{F}_{pc}$ can be efficiently implemented by 1D convolution of kernel size k. Compared with $\mathbf{F}_{fc}$, our $\mathbf{F}_{pc}$ reduces the number of parameters from $\frac{4C''^2}{r}$ to k, greatly improving the efficiency. Finally, SFF uses feature disentanglement ($\mathbf{F}_{dt}$) operation to generate attention weights for high-level and low-level features:

$$\beta_1^i = \S_{2i}, \ \beta_2^i = \S_{2i+1}, i \in [0, C'') \tag{9}$$

With the above design, our SFF module uses the calculated cross-level weights (β_1, β_2) to selectively output features of different levels, which improves the robustness of the network for objects at different scales.

IV. Experiments

To evaluate the effectiveness of SANet, we conduct detailed comparison experiments on the Cityscapes and CamVid datasets. In this section, we first introduce the datasets and implementation details, then conduct thorough ablation and comparison experiments on the Cityscapes dataset. Finally, we report the results on the CamVid dataset.

A. Benchmarks and Evaluation Metrics

1) Cityscapes: Cityscapes [5] is a dataset for street segmentation. It contains 5000 fine pixel-level annotations with 2975/500/1525 images for training/validation/testing respectively, where each image is of 1024×2048 resolution. The annotations contain 30 semantic classes, 19 of which are generally used to evaluate algorithms.

2) CamVid: CamVid dataset [27] is another street scene parsing dataset from a driving perspective, which contains 701 labeled images in total. Following previous methods [7], [15], the dataset is split into 367 images for training, 101 for validation, and 233 for testing. The images have a resolution of 720×960 and 11 different classes.

3) Evaluation Metrics: We employ the standard mean pixel intersection-over-union (mIoU) to report the accuracy and Frames Per Second (FPS) measured on RTX 3090 to report inference speed. Further, we use the number of parameters (Params) and float-point operations (FLOPs) to evaluate the model complexity.

B. Implementation Details

1) Training: During the training stage, we use stochastic gradient descent (SGD) with momentum 0.9, weight decay $5e-4$ to train our model. The mini-batch size is set to 12 for Cityscapes and CamVid. As common setting [45], we employ the "ploy" learning rate policy in which the initial learning rate is set to $1e-2$ and decayed by $\left(1-\frac{iter}{total_{iter}}\right)^{0.9}$ after each iteration. For a fair comparison with previous works [21], [23], [24], we also employ Online Hard Example Mining (OHEM) [48] and auxiliary loss to help the network learning process, the auxiliary loss weight is set to 0.4. In addition, we train the network for 120K and 30K iterations for the Cityscapes and CamVid datasets, respectively. For data augmentation, we use random horizontal flipping, random scaling, random cropping, and random color jittering. The crop size is set to 1024×1024 and 720×960 for Cityscapes and CamVid, respectively. The scale ranges in [0.125, 2.0] and [0.5, 2.0] for training the Cityscapes and CamVid, respectively.

C. Ablation Study on the Cityscapes

In this section, we perform extensive experiments to prove the effectiveness of each component in SANet. In the following experiments, the network is trained on the Cityscapes training set and evaluated on the Cityscapes validation (Val) set. Besides, we also discuss the module design details.

1) Baseline: We use ResNet-18 pre-trained on the ImageNet dataset as the backbone network. Then, we directly upsample the output of the last stage in the backbone network to the original image size, like FCN-32s [28]. We evaluate its performance and adopt it as our baseline.

2) Ablation for SCE: To study the effectiveness of SCE, we add the SCE module before the upsampling phase of the Baseline. Results from different settings for SCE are shown in Table I. As we can see, the pooling layer's output size n affects the efficiency and accuracy of SCE, and the accuracy and computation load generally increase with output diversity.

TABLE I

ABLATION EXPERIMENT ON THE DESIGN DETAILS OF THE SCE MODULE. "n" COLUMN INDICATES THE OUTPUT HEIGHT/WIDTH OF THE ADAPTIVE POOLING LAYER. WE CALCULATE FLOPS OF DIFFERENT METHODS EXCLUDING THE BACKBONE NETWORK AND ALL OUTPUT LAYERS, WHERE THE INPUT SIZE IS $3 \times 1024 \times 2048$

Upsampling Method	n	Ar	mIoU(%)	FLOPs(M)
BI	$(1,2,3,6)$	w/o	76.58	293.17
BI	$(1,2,3,6)$	w/	77.07	293.18
NNI	$(1,2,3,6)$	w/o	76.79	**275.35**
NNI	$(1,2,3,6)$	w/	**77.32**	273.36
NNI	(1)	w/	75.41	272.46
	(6)		74.15	272.99
	$(1,2,3)$		76.57	274.56
	$(1,3,6)$		76.62	274.57
	$(1,2,6)$		76.93	274.57
	$(2,3,6)$		76.87	274.57
	$(1,2,3,6)$		**77.32**	**273.36**
	$(1,3,6,8)$		77.07	275.37

TABLE II

COMPARISON OF DIFFERENT CONTEXT ENCODING METHODS. WE CALCULATE THE PARAMS AND FLOPS FOR VARIOUS MODULES W/ DRL AND THE ORIGINAL INPUT SIZE IS $3 \times 1024 \times 2048$

Method	Params(M)	FLOPs(G)	mIoU(%)
Baseline	-	-	72.69
DASPP [20]	0.76	1.57	76.39(3.70 ↑)
PPM [8]	0.20	0.28	76.45(3.76 ↑)
ASPP [9]	2.36	4.70	76.57(3.88 ↑)
DAPF [15]	4.98	2.43	76.75(4.06 ↑)
DAPPM [21]	1.07	1.72	77.30(4.61 ↑)
RCE	**0.13**	**0.14**	76.59(3.90 ↑)
SCE	0.20	0.27	**77.32(4.63 ↑)**

For example, setting $(1, 2, 3)$ achieves 76.57% mIoU, while $(1, 2, 6)$ and $(1, 2, 3, 6)$ achieves 76.93% and 77.32% mIoU, respectively. Nevertheless, unduly detailed division (e.g., from $(1, 2, 3, 6)$ to $(1, 3, 6, 8)$) will harm performance. This may be because the resulting output contains less context information, which will cause inconsistent results. Eventually, considering the compromise between efficiency and performance, we choose $(1, 2, 3, 6)$ as the default setting. Besides, we find that the choice of upsampling method and the aspect ratio (*AR*) of the pooling layer output will also impact the efficiency and performance of the SCE module. As in Table I, Nearest Neighbor Interpolation (NNI) performs favorably against Bilinear Interpolation (BI) (76.88% vs. 77.10%), which may be because NI can produce consistent contexts in the same spatial grid. Moreover, maintaining the aspect ratio (*MAR)* of input and output can also improve the accuracy (e.g., 76.79% vs. 77.32%). We claim that performance improvement mainly comes from the following two attributes: first, more detailed grid division can make the obtained context in the region more refined. Additionally, the middle-range context avoids introducing potential noise.

Further, we compare our SCE module with other context encoding modules, as in Table II. For a fair comparison,

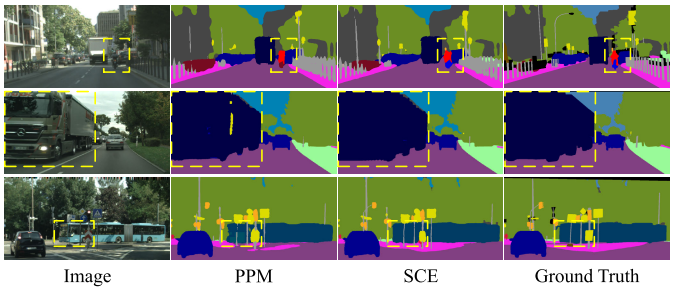


Fig. 5. Visualization results of the Selective Context Encoding module on the Cityscapes Val set, where notably improved regions are marked with yellow dashed boxes. Best viewed in color and zoom in.

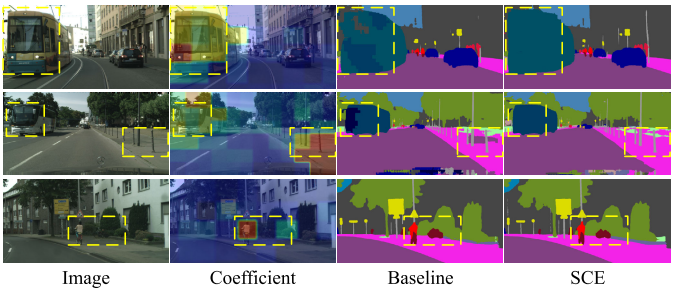


Fig. 6. Visualization of the pixel-wise context demand coefficients α, where the first two rows represent α_1 and the last row represents α_3. Note that since α is predicted from the pooling features, it exhibits a blocky distribution.

we reproduce all compared modules under the same settings. Note that we set the reduction ratio of the DRL to 2 for PPM [8] and ASPP [9], and to 4 for other modules, which means the latter have fewer feature maps in the input. Nevertheless, as in Table II, our SCE still performs better than all competitors, none of which consider the context demands of pixels. Concretely, SCE outperforms the PPM [8] and RCE by 0.89% and 0.73% mIoU, respectively. Besides, SCE outperforms DAPF [15] and DAPPM [21] by 0.57% and 0.02% mIoU, respectively, with less parameters and computation. The experimental results verify the effectiveness of our SCE module.

We crop some patches from certain images to analyze the effect of the SCE module on segmentation performance qualitatively. As illustrated in Fig. 5, since the contexts obtained by the pooling layer are easily biased towards salient objects or stuff, the prediction of inconspicuous objects by PPM is somewhat weakened or even disregarded (e.g., the "motorcycle" in the first row). In contrast, our SCE selectively constructs the contexts for each pixel by considering its context demands, avoiding the side effects of salient objects. Besides, our SCE can also improve semantic consistency within large objects (e.g., the "bus" in the third row). The visualizations demonstrate that selective context encoding can indeed improve segmentation performance.

We also visualize the key factor (Coefficient α) in the SCE module. As in Fig. 6, large objects (e.g., "train") generally require more global contexts (higher α_1) to eliminate inconsistent predictions. Meanwhile, visually-similar classes (e.g., "road" and "sidewalk") often suffer from semantic confusion, and this problem also requires global scene cues to remedy. In addition, some small objects (e.g., "rider") require more local (higher α_3) than global contexts to improve prediction results. The visualization results illustrate that the coefficient α can reasonably adjust the context demands of each pixel, thus improving the segmentation results.

We also perform visual analysis on high-level feature similarity of various methods to reveal the effectiveness of our SCE module. As in Fig. 7, our SCE module can produce purer features than other counterparts, which indicates that the features of a certain class in SCE module have little confusion with others. For example, the similarity for class "bus" and "wall" in the SCE module is more complete and clearer than other competitors, which allows it to reduce unexpected inconsistent predictions. In addition, the similarity between

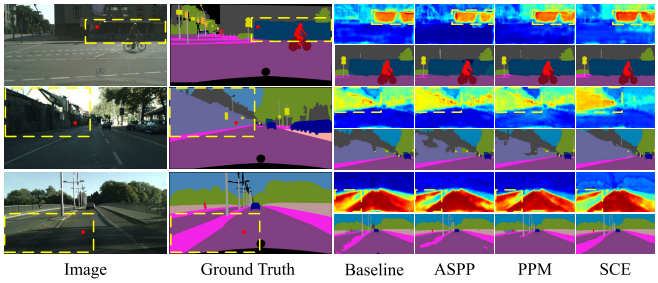


Fig. 7. Visualization results of high-level features and outputs of various methods. We randomly sample a point and compute its cosine similarity against the whole feature map. We visualize the semantic segmentation output for several challenging regions in the second row. Hot colors represent larger values and vice versa. Best viewed in color and zoom in.

TABLE III

ABLATION STUDY OF DIFFERENT SETTINGS FOR SFF. WE ATTACH TWO SFF MODULES W/O DRL TO THE BASELINE TO EVALUATE PERFORMANCE. NOTE PARAMS ONLY REPRESENT THE NUMBER OF PARAMETERS FOR CALCULATING THE WEIGHTS

GeM	k	F_{act}	Params	mIoU(%)
$p=1$	5	$1+tanh$	10	77.69
$p=3$	3	$Sigmoid$	**6**	77.54
	5		10	78.07
	7		14	78.27
	9		18	78.05
	fc		4096	77.52
	3	$1+tanh$	**6**	77.99
	5		10	78.17
	7		14	**78.37**
	9		18	78.13
	fc		4096	77.65
$p=5$	5	$1+tanh$	10	78.00
$p=\infty$	5	$1+tanh$	10	77.44

"road" and "sidewalk" in the third image is highlighted in the Baseline, ASPP, and PPM and suppressed in SCE, as shown in the yellow box. Consequently, SCE has reduced confusion errors between these two.

3) Effectiveness of the SFF Module: We further explore the capability of our Selective Feature Fusion module. Results from different settings are shown in Table III, where p denotes hyper-parameter in GeM, k denotes kernel size of F_{pc}, fc denotes global interaction with F_{fc}. As we can see, $p=3$ (GeM) substantially outperforms $p=1$ (GAP) and $p=\infty$ (GMP) by 0.48% and 0.73% mIoU, respectively. This is mainly because GeM can highlight salient information and

TABLE IV

PERFORMANCE COMPARISON OF DIFFERENT FEATURE FUSION METHODS. WE ONLY CALCULATE THE PARAMS AND FLOPS FOR VARIOUS MODULES AND THE ORIGINAL INPUT SIZE IS $3 \times 1024 \times 2048$

Method	Params(M)	FLOPs(G)	mIoU(%)
Baseline	-	-	72.69
FF [9]	1.77	21.77	74.59(1.90 ↑)
MRF [47]	**1.18**	**9.68**	74.74(2.05 ↑)
FFM [23]	1.53	13.59	76.65(3.96 ↑)
iAFF [50]	1.38	13.46	76.99(4.30 ↑)
ACM [49]	1.25	9.78	77.44(4.75 ↑)
SKNet [25]	1.23	9.76	77.59(4.90 ↑)
FSFM [30]	1.62	9.99	78.15(5.46 ↑)
SFF($k = fc$)	1.19	9.69	77.65(4.96 ↑)
SFF($k = 5$)	1.18	9.69	**78.17(5.48 ↑)**

suppress irrelevant information, while GAP and GMP cannot. Besides, the $1+tanh$ activation function always achieves better performance, probably because it can model more diversified relationships between multi-level features. Finally, F_{pc} consistently outperforms F_{fc} by a large margin, and only contains extremely few parameters (e.g., 6 parameters), which verifies the effectiveness of local interaction in cross-level feature fusion. In particular, different k brings different performance gains, e.g., $k = 5$ outperforms $k = 3$ by 0.18% mIoU for $1 + tanh$ activation.

In addition, we also compare our SFF module with other feature fusion methods. For a fair comparison, we append all compared modules to the backbone network Stage4 Stage4. The results are presented in Table IV, where FF [9] and MRF [47] indicate feature fusion using concatenation and summation, respectively. Besides, since ACM [49] and SKNet [25] are not designed for multi-level feature fusion, we adapt them to this task. Following the original setting, we set reduction ratio r to 4, 2, and 2 for FFM [23], SKNet [25], and FSFM [30] respectively, while other methods do not involve dimension reduction. We can see that SFF ($k = fc$) greatly improves the performance from 72.69% to 77.65%, which also outperforms FF [9], MRF [47], FFM [23] and iAFF [50] significantly. In addition, SFF ($k = 5$) achieve the best performance with similar or fewer computation compared with SKNet [25], ACM [49], and FSFM [30]. The experimental results prove the superiority of our SFF module, which can be attributed to our more robust selective feature fusion strategy.

Similarly, we have observed the visual improvement brought by the SFF module. As shown in Fig. 8, MRF using a simple feature fusion strategy cannot deal with large objects with large internal changes. By contrast, our SFF can reduce the ambiguity within large objects (e.g., the "truck" in the first and second rows). Moreover, MRF also struggles to classify some small objects. For example, the "pole" in the third row is lost in the MRF prediction, while the SFF module can predict it well. The improvement of segmentation results shows the effectiveness of our selective feature fusion mechanism.

We further visualize some feature maps to analyze the effects of the $1 + tanh$ function. As in Fig. 9, the low-level feature maps contain sharper details (e.g., edges of the object)

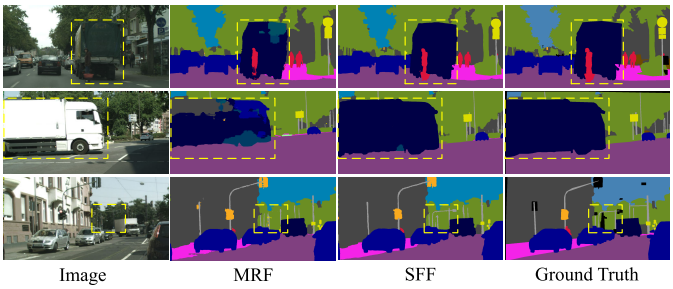


Fig. 8. Visualization results of the Selective Feature Fusion module on the Cityscapes Val set, where notably improved regions are marked with yellow dashed boxes. Best viewed in color and zoom in.

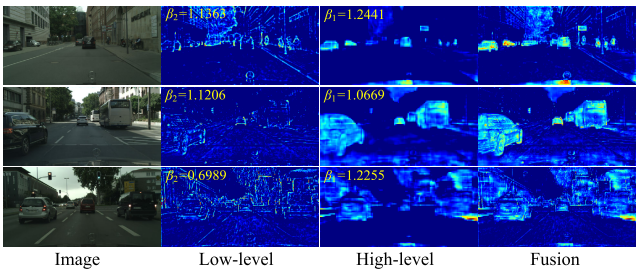


Fig. 9. Visualizations of the different feature maps in the network. We visualize the features of some channels. Best viewed in color and zoom in.

TABLE V

ABLATION STUDY ON THE CITYSCAPES VAL SET. FLOPS ARE ESTIMATED FOR INPUT OF $3 \times 640 \times 360$

SCE	SFF	Params (M)	FLOPs(G)	mIoU(%)
		11.79	10.37	72.69
√		11.54	10.42(0.05 ↑)	77.32(4.63 ↑)
	√	12.53	11.80(1.43 ↑)	78.17(5.48 ↑)
√	√	12.28	11.73(1.36 ↑)	**79.09(6.40 ↑)**

TABLE VI

EFFECTS OF STANDARD BELLS AND WHISTLES, INCLUDING DEEP SUPERVISION, OHEM AND TRAINING AT A CROP SIZE OF 1024×1024

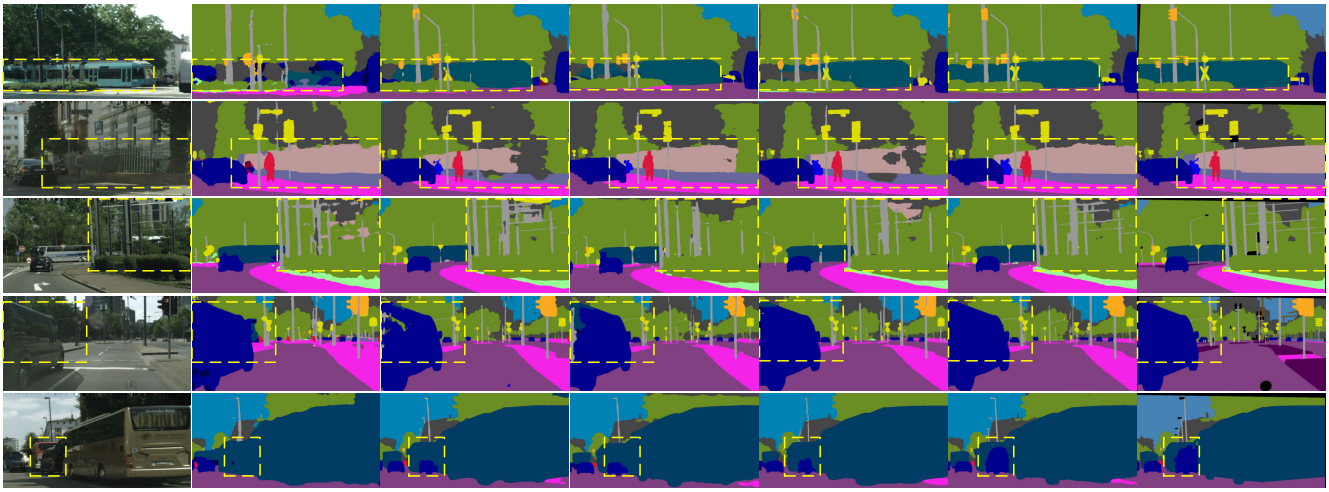
Method	DS	OHEM	1024×1024	mIoU(%)
SANet				77.23
SANet	√			77.81
SANet	√	√		78.19
SANet	√		√	78.35
SANet	√	√	√	**79.09**

with high frequency while the high-level feature maps contain smooth representation inside the object (e.g., body of the object) with low frequency. Our SFF module adaptively selects the desirable information while squeezing the irrelevant information containing different levels of features by producing gated vectors. Specifically, in the first image, the values of β_2 and β_1 are 1.1363 and 1.2441 respectively, thus the two-level features are enhanced simultaneously (means cooperation) to obtain the output, which is more structured than the input, especially for the edges. In the last image, the values of β_2 and β_1 are 0.6989 and 1.2255 respectively. Hence, the output is obtained by fusing the weakened low-level features

TABLE VII

CLASS-WISE EVALUATION RESULTS ON THE CITYSCAPES VALIDATION SET. LIST OF CLASSES (FROM LEFT TO RIGHT): ROAD, SIDE-WALK, BUILDING, WALL, FENCE, POLE, TRAFFIC LIGHT, TRAFFIC SIGN, VEGETATION, TERRAIN, SKY, PERSON, RIDER, CAR, TRUCK, BUS, TRAIN, MOTORCYCLE, AND BICYCLE. (THE BEST RESULT IN BOLD)

Method	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Per	Rid	Car	Tru	Bus	Tra	Mot	Bic	**mIoU**
Baseline	97.8	82.9	91.6	45.9	57.0	61.6	65.9	74.2	91.8	61.1	93.9	78.7	56.7	93.9	61.5	77.7	57.9	57.3	73.7	72.69
PPM [8]	98.0	84.4	92.3	57.3	58.1	63.1	67.2	75.7	92.1	63.9	94.3	79.5	59.5	94.7	81.0	86.2	72.3	59.1	73.8	76.45
ASPP [9]	**98.3**	85.6	92.2	51.5	59.2	64.7	67.9	77.3	92.2	62.6	94.4	80.1	59.5	94.9	79.3	85.5	74.9	59.8	74.9	76.57
DASPP [20]	**98.3**	85.8	92.1	47.0	59.0	64.5	67.7	76.5	92.0	61.5	94.5	80.4	61.3	94.8	78.9	86.3	74.4	61.3	75.2	76.39
DAPF [15]	**98.3**	85.4	92.2	51.0	59.3	64.1	67.5	76.4	92.1	62.7	94.3	80.4	61.2	94.8	80.9	85.7	75.7	61.2	75.1	76.75
DAPPM [21]	98.2	85.2	92.3	54.8	59.3	64.2	67.5	76.4	92.2	65.8	94.4	80.4	60.6	94.9	**82.7**	87.4	76.5	60.8	75.0	77.30
RCE	98.0	84.2	92.2	53.1	59.7	63.3	66.5	75.6	92.2	63.1	94.2	79.4	57.3	94.7	82.0	86.7	77.6	60.7	74.8	76.59
SCE	98.1	84.3	92.4	55.4	59.5	63.8	67.8	76.1	92.2	63.7	94.4	80.2	59.0	94.9	81.9	87.7	80.1	62.2	75.1	77.32
FF [9]	97.8	83.4	91.9	44.1	56.0	64.7	70.4	78.6	92.2	60.6	94.6	81.7	61.1	94.4	65.8	84.5	60.5	58.4	76.4	74.59
MRF [47]	97.9	83.4	91.9	48.5	55.6	65.6	69.6	78.0	92.1	61.4	94.9	81.6	61.1	94.4	68.3	81.3	60.5	59.3	75.8	74.74
FFM [23]	97.9	83.6	92.3	53.2	57.4	64.2	69.3	77.7	92.4	64.1	95.0	81.1	60.6	94.9	75.0	84.9	76.4	60.4	75.9	76.65
iAFF [50]	**98.3**	85.6	92.1	48.6	57.3	**66.6**	71.7	78.4	92.6	62.7	95.0	82.1	60.0	95.1	79.1	84.2	75.7	61.1	76.9	76.99
ACM [49]	98.2	85.2	92.6	52.8	59.2	66.3	70.9	79.6	92.6	63.1	95.1	82.0	60.6	95.1	79.8	84.9	75.9	60.4	76.9	77.44
SKNet [25]	**98.3**	85.6	92.3	47.0	59.7	66.2	71.2	79.7	92.5	64.0	95.0	82.4	61.6	95.2	81.5	87.2	79.5	58.5	76.7	77.59
FSFM [30]	**98.3**	**86.0**	92.5	49.6	60.4	**66.6**	70.8	79.9	92.5	64.4	95.0	82.2	**62.7**	95.2	81.3	87.6	82.7	60.2	76.8	78.15
SFF($k = fc$)	98.2	85.0	92.3	50.7	59.4	66.5	70.5	79.0	92.5	65.1	94.9	82.0	60.5	95.1	77.7	86.8	82.7	60.1	76.6	77.65
SFF($k = 5$)	98.3	85.9	92.6	55.9	58.8	66.0	70.8	79.0	92.6	63.3	95.0	82.2	61.2	95.2	82.5	88.5	79.0	61.7	76.7	78.17
SANet($k = 5$)	98.2	85.4	**92.9**	**60.8**	61.2	66.4	**72.2**	**79.3**	**92.7**	65.2	**95.2**	**82.4**	61.1	**95.3**	82.9	88.9	83.2	**62.3**	**77.0**	**79.09**
SANet($k = 7$)	**98.3**	85.5	92.8	58.9	**61.3**	66.2	71.9	**79.3**	**92.7**	**65.4**	95.1	**82.4**	61.7	**95.3**	**83.0**	**89.0**	**84.0**	61.9	77.0	79.02



(a) Input (b) ENet (c) BiSeNet2 (d) BiSeNetV2 (e) STDC2-Seg75 (f) SANet (g) Ground Truth

Fig. 10. Qualitative visual comparison against different methods on the Cityscapes Val set, where notably improved regions are marked with yellow dashed boxes. Best viewed in color and zoom in.

and the enhanced high-level features (means competition). As in Fig. 9, the output eliminates excessive detail information in low-level features and retains a suitable semantic structure.

4) In-Depth Analysis: We finally exhibit the influence of the SCE and SFF modules. As in Table V, they have improved the accuracy remarkably. Specifically, SCE and SFF can bring 4.63% and 5.48% mIoU improvement, respectively, which reveals that they are beneficial to semantic segmentation. Further, based on these two modules, SANet achieves 6.40% performance improvements with only a little extra computation cost. Note that the SANet is rather less computational than the SFF due to the DRL.

5) Ablative Studies of Standard Bells and Whistles: We analyze the effects of commonly used training tricks, which are also adopted by some recent real-time semantic segmentation methods, such as SFNet, DDRNet, etc. As shown in Table VI, the accuracy is raised from 77.23% to 79.09% with deep supervision (DS), OHEM, and training at a larger crop size (the default is 768 × 768).

Furthermore, we give class-wise comparison results of different methods. As in Table VII, SCE achieves significant improvements on large objects, surpassing the Baseline by 20.4% and 22.2% mIoU on "truck" and "train", respectively. In particular, compared to PPM [8], SCE performs better on small objects such as "motorcycle" or "bicycle". Meanwhile, SFF also brings huge performance gains. Quantitatively, SFF outperforms the Baseline by 4.4%, 4.9%, and 10.8% mIoU on "pole", "traffic light" and "bus", respectively. In particular,

TABLE VIII

CLASS-WISE EVALUATION RESULTS ON THE CITYSCAPES TEST SET. LIST OF CLASSES (FROM LEFT TO RIGHT): ROAD, SIDE-WALK, BUILDING, WALL, FENCE, POLE, TRAFFIC LIGHT, TRAFFIC SIGN, VEGETATION, TERRAIN, SKY, PERSON, RIDER, CAR, TRUCK, BUS, TRAIN, MOTORCYCLE, AND BICYCLE. METHODS WITH THE ‡ SYMBOL ARE REPRESENTED AS ACCURACY-ORIENTED METHODS. (THE BEST RESULT IN BOLD)

Method	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Per	Rid	Car	Tru	Bus	Tra	Mot	Bic	**mIoU**
ENet [12]	96.3	74.2	85.0	32.2	33.2	43.5	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.3
CGNet [52]	95.5	78.7	88.1	40.0	43.0	54.1	59.8	63.9	89.6	67.6	92.9	74.9	54.9	90.2	44.1	59.5	25.2	47.3	60.2	64.8
LEDNet [16]	97.1	78.6	90.4	46.5	48.1	60.9	60.4	71.1	91.2	60.0	93.2	74.3	51.8	92.3	61.0	72.4	51.0	43.3	70.2	69.2
ERFNet [13]	97.9	82.1	90.7	45.2	50.4	59.0	62.6	68.4	91.9	69.4	94.2	78.5	59.8	93.4	52.3	60.8	53.7	49.9	64.2	69.7
PCNet [22]	98.3	84.4	91.4	48.4	52.6	57.1	63.8	69.7	92.3	70.0	94.6	80.6	61.5	94.5	61.2	73.9	63.2	57.3	69.3	72.9
Reference [20]	98.2	84.0	91.6	50.7	49.5	60.9	69.0	73.6	92.6	70.3	94.4	83.0	65.7	94.9	62.0	70.9	53.3	62.5	71.8	73.6
RELAXNet [53]	**98.9**	84.9	92.2	57.2	54.8	64.3	70.6	74.0	93.0	**71.8**	94.8	83.7	64.4	95.1	58.6	72.7	58.2	59.9	71.8	74.8
DMA-Net [7]	98.5	**85.5**	92.2	53.3	55.3	62.5	70.9	74.9	93.0	71.2	95.0	84.0	66.6	95.6	68.2	**82.8**	76.6	**64.5**	73.2	77.0
SANet-50	98.2	83.3	90.9	**54.9**	51.2	55.5	61.6	67.0	91.7	69.5	94.1	78.6	61.8	94.0	67.8	77.6	62.0	56.2	65.9	72.7
SANet-75	98.4	84.9	92.2	52.9	56.3	63.2	69.5	73.7	92.9	70.5	94.8	83.4	65.9	95.4	71.3	82.6	74.2	61.2	72.0	76.6
SANet-100	98.5	**85.5**	**92.8**	54.6	**58.0**	**65.8**	**72.9**	**76.4**	**93.4**	71.2	**95.3**	**85.2**	**68.6**	**95.7**	**69.5**	82.6	**81.1**	63.3	**74.3**	**78.1**

TABLE IX

ACCURACY AND SPEED COMPARISON WITH SOTA ON THE CITYSCAPES DATASET. METHOD DENOTED WITH † USES TENSORRT ACCELERATION STRATEGY TO MEASURE THE SPEED. METHOD DENOTED WITH ‡ USES MULTI-SCALE TESTING TO EVALUATE PERFORMANCE. METHOD DENOTED WITH * INDICATES THAT ITS SPEED IS REPORTED IN PIDNET [54]

Method	Backbone	Input Size	GPU	FPS	mIoU(val)	mIoU(test)
Fast-scnn [32]	Custom	1024 × 2048	TitanXp	123.5	68.8	68.0
FarSee-Net [55]	ResNet-18	512 × 1024	TitanX M	68.5	70.3	70.2
FRNet [14]	Custom	512 × 1024	RTX 2080Ti	157.0	70.1	70.4
DFANet A [19]	Xception A	1024 × 1024	TitanX	100	71.9	71.3
MFNet [4]	Custom	512 × 1024	TitanXp	116.0	71.6	72.1
MsNet-ANAS [3]	Custom	768 × 1536	Tesla V100	119.9	-	72.9
Reference [20]	MobileNetV2	448 × 896	TitanX	51.0	74.4	73.6
LAANet [17]	Custom	448 × 896	GTX 1080Ti	95.8	72.7	73.6
CSRNet-light [46]	ResNet-18	1024 × 2048	GTX 1080Ti	56.0	76.1	74.0
BiSeNet2 [23]	ResNet-18	768 × 1536	GTX 1080Ti	65.5	74.8	74.7
BiSeNetV2† [24]	Custom	512 × 1024	GTX 1080Ti	156	73.4	72.6
BiSeNetV2-L† [24]	Custom	512 × 1024	GTX 1080Ti	47.3	75.8	75.3
HyperSeg-M* [56]	EfficientNet-B1	512 × 1024	RTX 3090	59.1	76.2	75.8
SwiftNetRN-18 [18]	ResNet-18	1024 × 2048	GTX 1080Ti	41.0	75.5	75.4
HoloSeg [6]	ResNet-18	512 × 1024	GTX 2080Ti	118.0	76.6	76.2
RTFormer-Slim [57]	Custom	1024 × 2048	RTX 2080Ti	110.0	76.3	-
NDNet-18 [58]	Custom	512 × 1024	RTX 2080Ti	109.3	-	76.5
STDC1-Seg75* [51]	STDC1	768 × 1536	RTX 3090	74.8	74.5	75.3
STDC2-Seg75* [51]	STDC2	768 × 1536	RTX 3090	58.2	77.0	76.8
DMA-Net [7]	ResNet-18	1024 × 2048	GTX 1080Ti	46.7	77.4	77.0
FSFNet [30]	ResNet-18	1024 × 2048	GTX 1080Ti	39.0	-	77.1
MSFNet [59]	ResNet-18	1024 × 2048	GTX 2080Ti	41.0	77.2	77.1
DDRNet-23-Slim* [21]	DDRNet-23-Slim	1024 × 2048	RTX 3090	108.1	77.8	77.4
FASSD-Net [15]	HarDNet	1024 × 2048	GTX 1080Ti	41.1	78.8	77.5
PIDNet-S-Simple [54]	DDRNet	1024 × 2048	RTX 3090	100.8	78.8	**78.2**
SeaFormer [60]	SeaFormer-B	1024 × 2048	RTX 3090	48.5	77.7	77.5
Trans4Trans‡ [39]	PVTv2-B1	1024 × 2048	-	-	78.2	-
SegFormer [40]	MiT-B0	1024 × 2048	GTX 1080Ti	15.2	-	76.2
KD [36]	MobilenetV2	1025 × 2049	-	26.3	71.0	72.7
SSTKD [38]	PSPNet18	1024 × 2048	-	-	75.2	74.4
StructKD [35]	ResNet-18	512 × 1024	-	-	73.1	75.3
TransKD-Base [37]	SegFormer-B0	512 × 1024	GTX 1080Ti	47.6	75.7	-
SANet-50	ResNet-18	512 × 1024	RTX 3090	**309.7**	73.7	72.7
SANet-75	ResNet-18	768 × 1536	RTX 3090	167.3	77.6	76.6
SANet-100	ResNet-18	1024 × 2048	RTX 3090	109.0	**79.1**	78.1

compared to MRF [47], SFF solves inconsistent predictions inside large objects (e.g., "truck" or "train") to some extent. Naturally, with two proposed modules, SANet achieves varying degrees of improvement on various scale objects. Note that when building SANet, the highly-abstracted semantic information from SCE is fed into SFF, using a larger kernel

TABLE X

SPEED (FPS) EVALUATION ON DIFFERENT PLATFORMS, WHERE TRT32 AND TRT16 INDICATE USING TENSORRT OPTIMIZATION AND QUANTIZING TO FP32 AND FP16, RESPECTIVELY

Methods	Geforce RTX 3090			Jetson AGX Xavier		
	Native	TRT32	TRT16	Native	TRT32	TRT16
SANet-50	309.7	416.4	1123.6	34.3	46.1	116.6
SANet-75	167.3	223.4	641.8	16.4	21.8	65.8
SANet-100	109.0	143.8	384.5	9.8	12.1	40.7

(e.g., $k = 7$) will instead introduce redundant information. Hence, we set k to 5 by default.

6) Visualization Analysis: To further illustrate the superiority of our method, we present some visualization results of SANet and several fashionable real-time methods, i.e., ENet [12], BiSeNet2 [23], BiSeNetV2 [24] and STDC2-Seg75 [51]. As in Fig. 10, our SANet produces higher-quality segmentation results on both large and small objects. Specifically, for the first two images, SANet generates more consistent predictions inside large objects (e.g., "train" and "fence") with a large internal variation. In contrast, other counterparts cannot handle this situation due to their limited receptive fields. For the third and fourth images, ENet, and STDC2-Seg75 that lose considerable spatial information perform poorly on small objects, while our SANet has classified correctly with accurate boundaries. Finally, our method is more robust when dealing with occluded objects, such as the "car" in the last images. That is, our SANet performs favorably against the other methods.

D. Comparison With SOTA Methods

1) Results on CityScapes Dataset: We compare our SANet with SOTA real-time methods on the Cityscapes test set. Following methods [29], [51], [61], we train SANet with only fine annotated training set and validation set data and then submit the test results to the official server to obtain the performance evaluation results. Moreover, we use 50, 75 and 100 after the method name (i.e., SANet) to indicate the input sizes of 512×1024, 768×1536 and 1024×2048, respectively. For example, with 768×1536 input size, we call the method SANet-75. We first present the detailed class-wise results on the Cityscapes test dataset to analyze the effectiveness of our method. As shown in Table VIII, our SANet-100 achieves the best IoU by a large margin on most classes, where 14 of the 19 classes have obtained the best scores. Moreover, our SANet has maintained a faster speed than other competing methods, proving our method's effectiveness and efficiency.

We further compare SANet with SOTA methods in Table IX. Overall, our SANet achieves superior accuracy and inference efficiency trade-off among all previous methods on the Cityscapes dataset. Specifically, our SANet-100 outperforms STDC2-Seg75 [51] and DDRNet-23-Slim [21] in terms of speed and accuracy. SANet-100 also provides comparable performance to previous SOTA method, PIDNet, without bells and whistles. In particular, our SANet-100 achieves impressive results with a lightweight backbone (ResNet-18) and outperforms BiSeNet2 [23] by a large margin (3.4%) with the same backbone. Meanwhile, it also surpasses HyperSeg-M [56], which adopts a more powerful backbone (EfficientNet-B1), i.e., 78.1% vs 75.9% mIoU and 109.0 vs 59.1 FPS. Morover, our method still achieves higher accuracy than recent methods (i.e., RTFormer [57], NDNet [58]) at a similar speed. Besides, our method also has certain advantages over Transformer-based methods, e.g., Trans4Trans [39], SegFormer [40], and SeaFormer [60].

TABLE XI

ACCURACY AND SPEED COMPARISON WITH SOTA ON THE CAMVID TEST DATASET. METHOD DENOTED WITH † USES TENSORRT ACCELERATION STRATEGY TO MEASURE THE INFERENCE SPEED

Method	Input Size	FPS	mIoU(%)
DFANet A [19]	720×960	120	64.7
CGNet [52]	720×960	-	65.6
FRNet [14]	360×480	183.0	67.4
BiSeNet2 [23]	720×960	116.3	68.7
PCNet [22]	360×480	62.1	67.0
LAANet [17]	360×480	112.5	67.7
FASSD-Net [15]	720×960	80.0	69.3
RELAXNet [53]	360×480	79.0	71.2
MFNet [4]	512×512	145.0	71.5
BiSeNetV2† [24]	720×960	124.5	72.4
BiSeNetV2-L† [24]	720×960	32.7	73.2
DMA-Net [7]	720×960	119.8	73.6
SwiftNetRN-18 pyr [18]	720×960	-	73.7
STDC1-Seg† [51]	720×960	197.6	73.0
STDC2-Seg† [51]	720×960	152.2	73.9
MsNet-ANAS [3]	512×512	184.2	74.1
HoloSeg [6]	720×960	105.3	74.1
DDRNet-23-Slim [21]	720×960	230.0	74.4
CSRNet-heavy [46]	720×960	46.4	74.5
FSFNet [30]	720×960	91.0	75.1
MSFNet [59]	768×1024	91.0	75.4
SANet	720×960	**250.4**	**77.2**

Knowledge distillation (KD) can improve the segmentation accuracy without increasing any other computing costs by transferring knowledge from a cumbersome network to a lightweight network. Hence, there are some methods to utilize KD to obtain efficient segmentation models. For example, through well-designed knowledge distillation methods, KD [36] improves the performance of the student network from 70.2% to 72.7%, SSTKD [38] improves the performance from 67.7% to 74.4%. Compared with KD-based methods, our SANet also achieves a promising performance (78.1%) without any other knowledge transfer. However, knowledge distillation still has excellent potential to improve the performance of lightweight networks.

We use TensorRT for SANet and evaluate on RTX 3090 and Jetson AGX Xavier. SANet-100 achieves 78.9% and 79.1% mIoU on Cityscpaes validation using half-precision and full-precision formats, respectively. Table X reports the inference speed for different resolutions using 16-bit and 32-bit floating point numbers. As in Table X, with TensorRT optimization and quantization to FP16, our method can achieve real-time inference on embedded devices under different resolutions. For example, for SANet-100, the speed has increased from 9.8 FPS to 40.7 FPS, but the accuracy decreases slightly (79.1% vs. 78.9%). This speed/accuracy trade-off is acceptable to obtain some speed gain.

2) Results on CamVid Dataset: To prove the generalization ability of our method, we also conduct experiments on the CamVid dataset, and Table XI presents the testing results. We find that SANet obtains 77.2% mIoU with 250.4 FPS on the CamVid test dataset for a 720×960 input, which also achieves the SOTA trade-off between speed and segmentation performance. We, therefore, conclude that SANet is an advanced real-time general semantic segmentation model that performs competitively across a wide range of datasets.

V. Conclusion

In this paper, we focus on extracting scale-aware features and propose a novel Scale-Aware Network (SANet) for real-time semantic segmentation of street scenes. SANet contains two core lightweight components: Selective Context Encoding (SCE) module and Selective Feature Fusion (SFF) module. Specifically, we introduce the SCE module to customize contextual features for each pixel and the SFF module to fuse low-level and high-level features effectively and efficiently. Two modules are tightly coupled to obtain the scale-aware features, where each feature map contains scale-specific information, avoiding information redundancy and confusion caused by inflexible context encoding and simple feature fusion strategy, thus dramatically improving performance. Extensive experiments show that our SANet achieves SOTA trade-offs on the Cityscapes and CamVid datasets regarding inference speed and segmentation accuracy.

References

[1] X. Zhang, Y. Ling, Y. Yang, C. Chu, and Z. Zhou, "Center-point-pair detection and context-aware re-identification for end-to-end multi-object tracking," *Neurocomputing*, vol. 524, pp. 17–30, Mar. 2023.

[2] V.-C. Miclea and S. Nedevschi, "Real-time semantic segmentation-based stereo reconstruction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1514–1524, Apr. 2020.

[3] B. Xie et al., "Multi-scale fusion with matching attention model: A novel decoding network cooperated with NAS for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 12622–12632, Aug. 2022.

[4] M. Lu, Z. Chen, C. Liu, S. Ma, L. Cai, and H. Qin, "MFNet: Multi-feature fusion network for real-time semantic segmentation in road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20991–21003, Nov. 2022.

[5] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[6] S. Li, Q. Yan, C. Liu, M. Liu, and Q. Chen, "HoloSeg: An efficient holographic segmentation network for real-time scene parsing," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 2395–2402.

[7] X. Weng et al., "Deep multi-branch aggregation network for real-time semantic segmentation in street scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17224–17240, Oct. 2022.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[10] X. Li, H. Zhao, L. Han, Y. Tong, S. Tan, and K. Yang, "Gated fully fusion for semantic segmentation," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2020, vol. 34, no. 7, pp. 11418–11425.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[12] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.

[13] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.

[14] M. Lu, Z. Chen, Q. M. J. Wu, N. Wang, X. Rong, and X. Yan, "FRNet: Factorized and regular blocks network for semantic segmentation in road scene," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 3522–3530, Apr. 2022.

[15] L. Rosas-Arias, G. Benitez-Garcia, J. Portillo-Portillo, J. Olivares-Mercado, G. Sanchez-Perez, and K. Yanai, "FASSD-Net: Fast and accurate real-time semantic segmentation for embedded systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14349–14360, Sep. 2022.

[16] Y. Wang et al., "LEDNet: A lightweight encoder–decoder network for real-time semantic segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1860–1864.

[17] X. Zhang, B. Du, Z. Wu, and T. Wan, "LAANet: Lightweight attention-guided asymmetric network for real-time semantic segmentation," *Neural Comput. Appl.*, vol. 34, no. 5, pp. 3573–3587, Mar. 2022.

[18] M. Oršić and S. Šegvić, "Efficient semantic segmentation with pyramidal fusion," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107611.

[19] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9514–9523.

[20] G. Dong, Y. Yan, C. Shen, and H. Wang, "Real-time high-performance semantic image segmentation of urban street scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3258–3274, Jun. 2021.

[21] H. Pan, Y. Hong, W. Sun, and Y. Jia, "Deep dual-resolution networks for real-time and accurate semantic segmentation of traffic scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3448–3460, Mar. 2023.

[22] Q. Lv, X. Sun, C. Chen, J. Dong, and H. Zhou, "Parallel complement network for real-time semantic segmentation of road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 4432–4444, May 2022.

[23] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 325–341.

[24] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation," *Int. J. Comput. Vis.*, vol. 129, no. 11, pp. 3051–3068, Nov. 2021.

[25] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 510–519.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[27] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2008, pp. 44–57.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[29] J. Fu et al., "Dual attention network for scene segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3141–3149.

[30] Y. Pei, B. Sun, and S. Li, "Multifeature selective fusion network for real-time driving scene parsing," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[32] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast semantic segmentation network," in *Proc. Brit. Mach. Vis. Conf.*, 2019, pp. 1–12.

[33] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[34] X. Wang, S. Lai, Z. Chai, X. Zhang, and X. Qian, "SPGNet: Serial and parallel group network," *IEEE Trans. Multimedia*, vol. 24, pp. 2804–2814, 2022.

[35] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured knowledge distillation for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2599–2608.

[36] T. He, C. Shen, Z. Tian, D. Gong, C. Sun, and Y. Yan, "Knowledge adaptation for efficient semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 578–587.
[37] R. Liu et al., "TransKD: Transformer knowledge distillation for efficient semantic segmentation," 2022, *arXiv:2202.13393*.
[38] D. Ji, H. Wang, M. Tao, J. Huang, X. Hua, and H. Lu, "Structural and statistical texture knowledge distillation for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 16855–16864.
[39] J. Zhang, K. Yang, A. Constantinescu, K. Peng, K. Müller, and R. Stiefelhagen, "Trans4Trans: Efficient transformer for transparent object and semantic scene segmentation in real-world navigation assistance," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19173–19186, Oct. 2022.
[40] E. Xie et al., "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Adv. Neural Inf. Process. Sys. (NIPS)*, vol. 34, Dec. 2021, pp. 12077–12090.
[41] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for semantic segmentation in street scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3684–3692.
[42] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 603–612.
[43] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.
[44] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[45] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.
[46] J. Xiong, L.-M. Po, W.-Y. Yu, C. Zhou, P. Xian, and W. Ou, "CSRNet: Cascaded selective resolution network for real-time semantic segmentation," *Expert Syst. Appl.*, vol. 211, Jan. 2023, Art. no. 118537.
[47] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5168–5177.
[48] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 761–769.
[49] X. Hu, K. Yang, L. Fei, and K. Wang, "ACNET: Attention based network to exploit complementary features for RGBD semantic segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1440–1444.
[50] Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional feature fusion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3559–3568.
[51] M. Fan et al., "Rethinking BiSeNet for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9711–9720.
[52] T. Wu, S. Tang, R. Zhang, J. Cao, and Y. Zhang, "CGNet: A light-weight context guided network for semantic segmentation," *IEEE Trans. Image Process.*, vol. 30, pp. 1169–1179, 2021.
[53] J. Liu, X. Xu, Y. Shi, C. Deng, and M. Shi, "RELAXNet: Residual efficient learning and attention expected fusion network for real-time semantic segmentation," *Neurocomputing*, vol. 474, pp. 115–127, Feb. 2022.
[54] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A real-time semantic segmentation network inspired by PID controllers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 19529–19539.
[55] Z. Zhang and K. Zhang, "FarSee-Net: Real-time semantic segmentation by efficient multi-scale context aggregation and feature space super-resolution," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8411–8417.
[56] Y. Nirkin, L. Wolf, and T. Hassner, "HyperSeg: Patch-wise hypernetwork for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4060–4069.
[57] J. Wang, C. Gou, Q. Wu, H. Feng, J. Han, E. Ding, and J. Wang, "RTFormer: Efficient design for real-time semantic segmentation with transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1–14.
[58] S. Li, Q. Yan, X. Zhou, D. Wang, C. Liu, and Q. Chen, "NDNet: Spacewise multiscale representation learning via neighbor decoupling for real-time driving scene parsing," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–15, 2022.
[59] H. Si, Z. Zhang, F. Lv, G. Yu, and F. Lu, "Real-time semantic segmentation via multiply spatial fusion network," in *Proc. Brit. Mach. Vis. Conf.*, 2020, pp. 1–12.
[60] Q. Wan, Z. Huang, J. Lu, G. Yu, and L. Zhang, "SeaFormer: Squeeze-enhanced axial transformer for mobile semantic segmentation," 2023, *arXiv:2301.13156*.
[61] P. Lin, P. Sun, G. Cheng, S. Xie, X. Li, and J. Shi, "Graph-guided architecture search for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4202–4211.

Kaige Li received the M.S. degree from the Ocean University of China, Qingdao, China, in 2020. He is currently pursuing the Ph.D. degree in computer science and technology with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China.

His current research interests include deep learning, image semantic segmentation, computer vision, and smart city.

Qichuan Geng received the B.S. degree in automation science and the Ph.D. degree in computer applied technology from Beihang University, Beijing, China, in 2012 and 2021, respectively. He is currently a Lecturer and a Master's Instructor with the Information Engineering College, Capital Normal University. His main research interests include computer vision, semantic segmentation, and scene geometry recovery.

Zhong Zhou received the B.S. degree in material physics from Nanjing University in 1999 and the Ph.D. degree in computer science and engineering from Beihang University, Beijing, China, in 2005. He is currently a Professor and a Ph.D. Advisor with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His main research interests include virtual reality, augmented reality, computer vision, and artificial intelligence.