JEI Journal of Electronic Imaging

RESEARCH PAPER

Bilateral network with dual-guided attention for real-time semantic segmentation of road scene

Liang Liao,[a] Liang Wan,[a,*] Mingsheng Liu,[a] and Shusheng Li[b]

[a]Guizhou University, State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guiyang, China

[b]Northeastern University, School of Computer Science and Engineering, Shenyang, China

ABSTRACT. Deep learning methods have demonstrated excellent performance in semantic segmentation tasks across various scenarios. However, their performance remains inadequate for real-time applications, such as road scenes, prompting extensive research into real-time semantic segmentation. We proposed a network with a two-branch architecture that is a prevalent framework in this field. We first create an efficient backbone based on the U-like residual block we proposed to enhance multi-scale feature extraction capabilities. In addition, many two-branch networks have designed complex structures for fusing the feature maps of the two branches at the end of the networks. To simplify this process, our network designs a light-weight fusion mechanism that utilizes attention calculations to guide the fusing of features. We name this module as dual-guided attention. This fusion module operates in parallel, with each branch employing a fully connected layer with few neurons to determine the correlation of the flattened feature map, thereby executing the attention and feature fusion simultaneously. Extensive experiments on the Cityscapes and Cambridge-driving Labeled Video datasets show the effectiveness of our method.

 [DOI: 10.1117/1.JEI.33.6.063026]

Keywords: semantic segmentation; deep learning; feature extraction; attention mechanism

Paper 240548G received May 29, 2024; revised Oct. 21, 2024; accepted Oct. 28, 2024; published Nov. 21, 2024.

1 Introduction

Semantic segmentation, a pivotal task in computer vision, aims to divide images into semantically meaningful regions. This technique finds widespread application in diverse fields, including autonomous driving, computational photography, and human–machine interaction. Since the advent of the fully convolutional network (FCN)[1] for image segmentation, deep learning technologies have begun to outperform traditional methods that rely on handcrafted features, both in accuracy and efficiency. However, some applications require extremely fast inference speeds. Methods such as DeepLab,[2] SegNet,[3] and PSPNet[4] offer promising segmentation accuracy but tend to be memory-intensive and sacrifice time complexity with high-resolution inputs. Significantly, this limits their application in real-time scenarios such as autonomous driving, where a high resolution of input images is typically required.

In recent years, significant advancements have been made in the field of real-time semantic segmentation, as evidenced in Refs. 5 and 6, with the aim of enhancing model usability on devices with limited computing power. These methods generally fall into two categories.

*Address all correspondence to Liang Wan, lwan@gzu.edu.cn

1017-9909/2024/$28.00

The first is the single-branch encoder–decoder architecture,[7,8] which directly builds on the research lineage established by the FCN. The second category encompasses the two-[9] or multi-branch architectures,[5] specifically designed for real-time semantic segmentation. The primary difference between these categories lies in their approach to handling multi-scale semantic features. Encoder–decoder methods typically rely on layer-by-layer down-sampling and feature fusion operations within a single branch to extract semantic features. On the other hand, multi-branch methods take a different approach, focusing on the integration of low-level details and high-level semantics. They suggest that spatial details and contextual semantics can be extracted independently, offering a distinct perspective on the extraction of multi-scale semantic features. Remarkably, BiSeNetV2,[6] a hand-crafted two-branch network, has attained state-of-the-art performance, establishing the two-branch architecture as a benchmark paradigm for real-time semantic segmentation. This innovative architecture not only enhances the segmentation of boundaries and small objects, surpassing the outperforming of the single-branch encoder–decoder structure, but also facilitates faster inference speeds, making it a highly efficient solution.

However, the two-branch architecture is not flawless. Because the two-branch architecture is designed to independently extract spatial details and semantic information, each branch requires distinct channel numbers and down-sampling ratios. Due to the differences in feature channels and scales extracted by the two branches, an intricate fusion mechanism needs to be designed when fusing features at the end of the network. Unfortunately, many methods have overly complicated designs in the feature fusion. In addition, although well-known networks such as BiSeNetV2 achieve impressive speeds through their handcrafted light-weight backbone, this design choice poses challenges in optimizing the model. To mitigate convergence issues associated with relying on this handcrafted backbone, BiSeNetV2 incorporates numerous auxiliary losses, which in turn increases the complexity of the optimization process. Therefore, this paper intends to further develop the two-branch architecture by devising a simple and highly effective feature fusion method, thereby better leveraging the advantages of the two-branch architecture.

Based on the previous research of two-branch networks discussed in this paper, we introduce a lightweight attention mechanism into the feature fusion process at the end of network and design a parallel structure that employs this attention mechanism to guide feature fusion, named as dual-guided attention (DGA) module. In conjunction with this, we have developed a new two-branch network that leverages the DGA module to deliver enhanced performance. The network comprises a high-resolution branch for spatial information extraction and a low-resolution branch that utilizes the U-like residual block (URS) introduced in this paper to capture semantic details. The feature maps extracted by these parallel branches are simultaneously fed into the DGA module. Subsequently, we employ straightforward bilinear interpolation rather than any decoder structures to reconstruct the segmentation map. Since our network is designed based on a two-branch architecture, featuring a bilateral structure, and employs dual-guided attention for feature fusion, it can be termed the Bilateral Network with Dual-Guided Attention, abbreviated as BiDGANet. Comparative evaluations with other leading models on the Cityscapes[10] and CamVid[11] datasets demonstrate that our method achieves competitive results, thereby validating its effectiveness. We summarize the contributions of this paper as follows.

1. Based on residual connections and an asymmetric encoder–decoder architecture, we have designed the U-like residual block. We then stack these blocks to construct the low-resolution branch of our network, resulting in a backbone that is simple and has low parameters but has better multi-scale feature extraction capabilities. In addition, this branch can be pretrained independently on the target dataset, which substantially improves the generalization ability of the entire model.
2. We propose a lightweight attention mechanism called DGA that achieves feature fusion and attention computation simultaneously. This approach significantly reduces the overhead associated with feature fusion at the end of the two-branch architecture network.
3. Based on a more lightweight backbone design and the exceptional dual-guided attention module for feature fusion, we have designed BiDGANet, a real-time two-branch network that achieves competitive results on standard benchmarks.

2 Related Work

2.1 Single-Branch Encoder–Decoder Architecture

Certain networks employ the classic encoder–decoder architecture as their fundamental framework. These networks utilize layer-by-layer down-sampling and feature fusion operations to extract semantic features, effectively encoding both low-level details and high-level semantics simultaneously. ESPNet,[7] for instance, introduces parallel convolutions with varying dilation rates to broaden the receptive field, thereby enhancing the decoder's efficiency. EDANet[8] proposes the efficient dense module, which densely connects input images and output features within a larger block, enabling information sharing across a broader receptive field. DFANet[12] leverages deep multi-scale feature aggregation and lightweight depth-wise separable convolutions to effectively refine both high-level and low-level features. Despite achieving state-of-the-art performance, the majority of these single-branch methods encounter challenges in maintaining inference speeds when dealing with higher-resolution input images. The main issue is the substantial computation needed for deep channels of high-resolution feature maps, which slows down the overall processing speed. However, using networks with shallow channel numbers often fails to achieve satisfactory segmentation accuracy.

2.2 Two-Branch Architecture

Distinct from single-branch architecture methods, the two-branch architecture preserves high-resolution details captured early in the network by independently extracting features across different scales. BiSeNet[9] introduced a two-branch architecture consisting of a context path and a spatial path. The former, based on a compact pretrained backbone, aims to extract contextual information, whereas the latter focuses on spatial details using several convolutional layers. Subsequently, BiSeNetV2[6] further streamlined the network structure. It introduced bilateral guided aggregation to replace the feature fusion module in BiSeNet and designed an entirely handcrafted semantic branch. All these efforts enhanced the network's efficiency. However, certain researchers contend that a limitation of these approaches is the lack of effective feature interaction among the branches. Features are solely fused at the end of the network, leading to a meticulously designed fusion process. The complexity of this process may result in reduced accuracy and introduce additional computational overhead, thereby posing a new performance bottleneck. Therefore, Fast-SCNN[13] and DDRNet[14] adopted a shared backbone architecture. These networks initially follow a single branch and then split into two parallel deep branches with two different resolutions. This design shares some network parameters from the outset and allows for many interactions in the middle stages of the network, such as the bilateral fusion proposed by DDRNet. Despite acknowledging that the performance of two-branch networks is indeed constrained by the fusion of features from their respective branches, it is crucial to note that the primary advantage of this architecture lies in its simplicity, allowing for the substitution of various network components. However, enabling interactions in the middle stages of the network can inadvertently compromise its flexibility. Given this trade-off, this paper will persist in focusing on improving the fusion process at the end of the network, aiming to achieve performance parity with the aforementioned methods while preserving the network's inherent advantage.

2.3 Attention Mechanism

Attention mechanisms are employed to tackle local challenges in neural networks, bridging the gap between local information and global insights. DANet[15] introduced a dual attention mechanism, utilizing a position attention module to focus on spatial information and a channel attention module to correlate channel information. However, due to the high computational cost of the channel attention module, it is not suitable for real-time environments if the backbone's output features have deep channels. Consequently, CCNet[16] focuses solely on spatial information and introduces a criss-cross attention module to identify connections among pixels in the same row or column. This reduces computational demands, making it feasible to stack multiple attention modules. In addition, linear complexity attention mechanisms, such as external attention (EA),[17] achieve effects comparable to self-attention mechanisms[18] at a lower computational cost.

These efforts have inspired our design, prompting us to develop dual-guided attention for our two-branch architecture network.

3 Method

This section provides a detailed description of our work. In the previous version of our conference paper,[19] we have also adopted a similar design structure and further developed the dual-guided attention module based on it. This paper primarily focuses on reconstructing the low-resolution branch and reintroducing the dual-guided attention module. Figure 1 illustrates the construction of our network, which is primarily divided into three parts: a high-resolution branch enclosed within a pink dashed line, a low-resolution branch within a blue dashed line, and a dual-guided attention module within a yellow dashed line. The numbers inside the cubes represent the ratio of the feature map size to the input resolution. Specific ratios of channel numbers are shown in Table 1. The final stage of the low-resolution branch outputs from a context embedding block, represented by green cubes, with the bottom of the blue cubes gradually transitioning to green at this stage. Detailed descriptions of each component are as follows.

3.1 High-Resolution Branch

The high-resolution branch of our network specializes in capturing spatial details. It has wide channels and shallow layers, ensuring a rich flow of spatial information. Drawing inspiration from the detailed branch of BiSeNetV2, we have designed our high-resolution branch to suit our needs. This branch comprises three layers of 3×3 convolution, effectively broadening the channel capacity. Following each convolution layer, a max-pooling layer is appended for swift

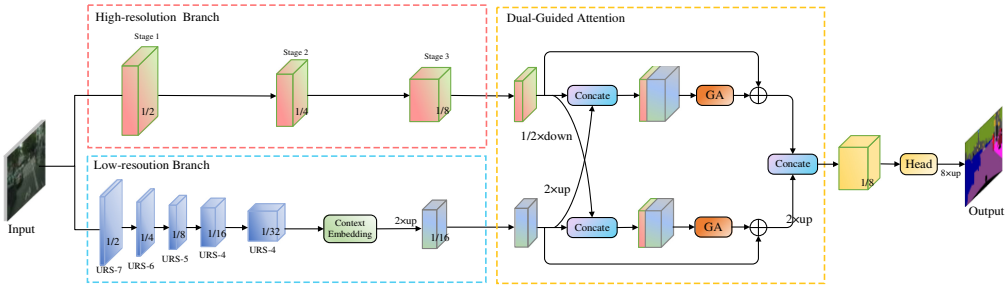

Fig. 1 Overall architecture of our proposed BiDGANet.

Table 1 Different versions of BiDGANet.

	High-resolution branch			Low-resolution branch				
Version	Stage 1	Stage 2	Stage 3	URS-7	URS-6	URS-5	URS-4	URS-4
Light	I: 3 O: 64	I: 64 O: 64	I: 64 O: 128	I: 3 M: 16 O: 32	I: 32 M: 16 O: 64	I: 64 M: 16 O: 64	I: 64 M: 16 O: 64	I: 64 M: 16 O: 64
Base				I: 3 M: 16 O: 32	I: 32 M: 16 O: 64	I: 64 M: 32 O: 128	I: 128 M: 64 O: 256	I: 256 M: 128 O: 256
Large				I: 3 M: 32 O: 64	I: 64 M: 32 O: 128	I: 128 M: 64 O: 256	I: 256 M: 128 O: 512	I: 512 M: 256 O: 512

All versions have the same scale of high-resolution branch. URS, U-like residual block; *I*, number of input channels (C_{in}); *M*, middle channels; *O*, output channels (C_{out}).

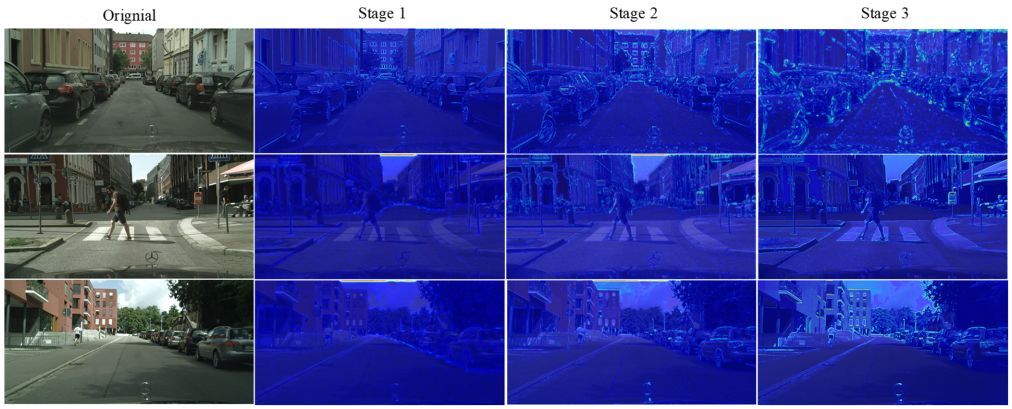

Fig. 2 Gradient-weighted class activation maps of the high-resolution branch.

down-sampling. After passing through three such convolution layers, the input image's scale will be down-sampled to 1/8 its original size. With a keen focus on local features and computational efficiency, this branch refrains from using dilated convolutions or residual connections. Consequently, the resulting output feature map not only preserves the edge details of objects within the image but also efficiently reduces resolution while expanding channels. Despite having somewhat constrained semantic extraction capabilities, the primary role of this branch is to efficiently convey spatial information to the low-resolution branch, facilitating its subsequent fusion. Therefore, the emphasis here is not on standalone feature extraction capabilities. Figure 2 shows the gradient-weighted class activation maps[20] of the high-resolution branch. These maps show that the focus of the high-resolution branch is mainly on the edge of the objects

3.2 Low-Resolution Branch

The low-resolution branch in the network is tasked with extracting semantic information and is critical to the overall model's performance. To enhance multi-scale feature extraction, we have designed a new backbone. The cornerstone of this branch is the URS, drawing inspiration from the residual blocks (RSs) in ResNet[21] and the encoder–decoder network U-Net.[22] We have combined the concepts of shortcut and skip connections with the asymmetric encoder–decoder architecture to create the URS block. Figure 3 presents a comparison between a common residual block in ResNet and our U-like residual block. As shown in the right half of Fig. 3, the height of the URS block is represented by L, which shows the URS module with the deepest number of layers used in low-resolution branches, that is, $L = 7$. Regardless of the size of the URS module, the resolution of its input and output feature maps is always the same. In the figure, H represents the height of input feature x, W represents the width, C_{in} represents the number of channels in the input feature map, M represents the number of channels in the intermediate layers, and C_{out} represents the number of channels in the final output of the entire module. For the illustrated RS module, the first convolution of the network is described as $F_1(x)$. As the 1×1 convolution acts as a linear transformation of input x across each channel, the output is described by $\text{conv}(x)$. Then, the RS module can be described as

$$f_{\text{RS}}(x) = F_2(F_1(x)) + \text{conv}(x). \tag{1}$$

The URS module depicted in the figure essentially combines an asymmetric U-like small network with a residual connection. The first convolutional layer of the network is also referred to as $F_1(x)$, and the component within the dotted box is labeled $U_7(x)$. The URS module can be described as follows:

$$f_{\text{URS}}(x) = U_7(F_1(x)) + F_1(x). \tag{2}$$

Generally, the key difference between the URS and RS models lies in replacing the feature extraction convolution layer $F_2(x)$ in RS with a U-like asymmetric encoder–decoder subnet, $U_L(x)$, and using the output $F_1(x)$ of the first convolutional layer to replace the original input feature x. This design allows the network to directly extract multi-scale features from a residual block. As most of the computation in this process occurs after down-sampling by max-pooling,

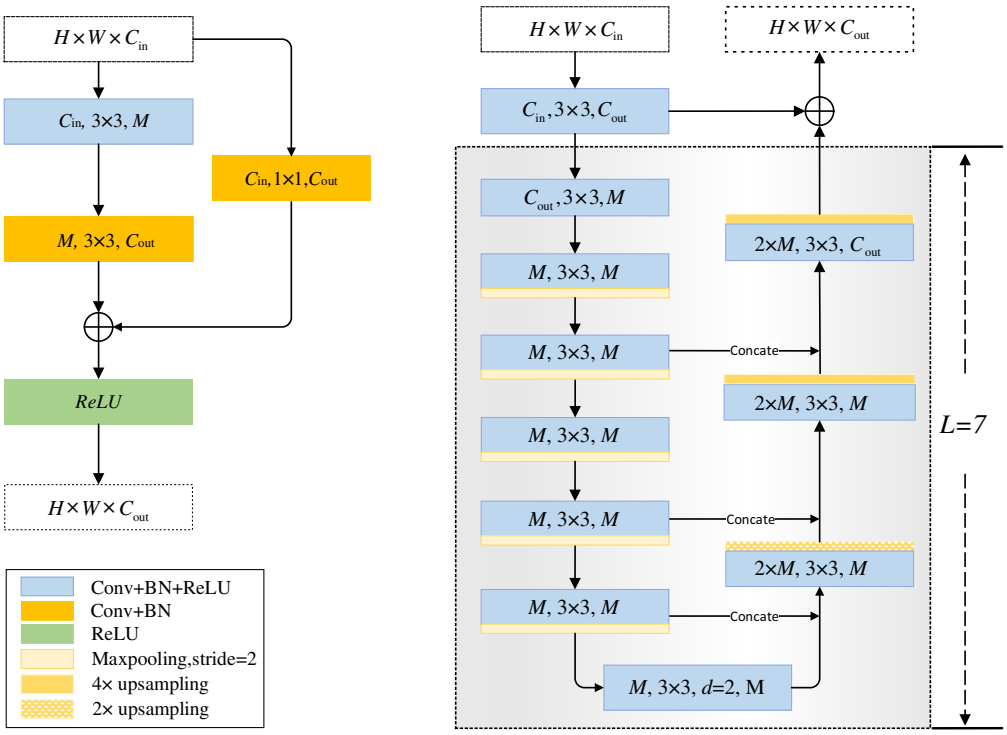

Fig. 3 Residual block and U-like residual block ($L = 7$).

and because the number of M and C_{out} channels is kept relatively low in subsequent network construction, the overall increase in the number of model parameters and computational load is not significant.

Based on the URS block, we have designed a new backbone shown in Fig. 4. The input image enters the network at its original size and passes through the first URS block with $L = 7$. The output from the first URS block then goes through a 2×2 max-pooling layer with a stride of 2, reducing the feature map to half the scale of the input before entering a second URS block with $L = 6$, which means the L of each URS block is one minus the previous block, and each URS block is followed by the same max-pooling layer. At the end of the backbone, the URS block with $L = 4$ is repeated twice, enabling the model to ultimately output feature maps with a resolution of 1/32 of the original image. At this point, we have concluded the stacking of our designed backbone, which is presented as the network architecture in Fig. 4. As it comprises 44 convolutional layers, we have named it URSNet-44. When URSNet-44 is integrated into the network as the low-resolution branch, we append a context embedding module[6] to incorporate global contextual information. Subsequently, the feature maps are up-sampled by two times. Finally, we will get the feature maps with 1/16 resolution as our low-resolution branch's output.

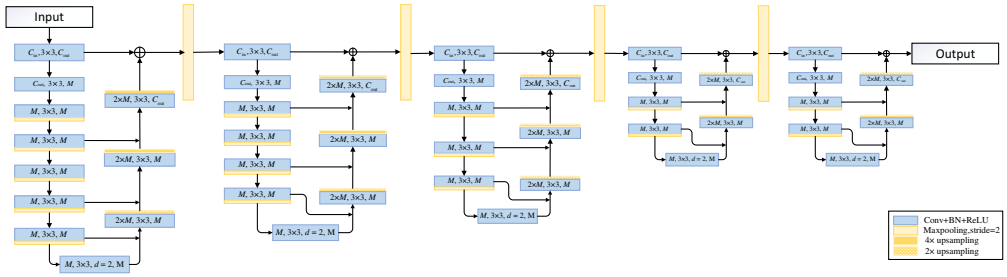

Fig. 4 By changing the number of L and stacking URS blocks, the detailed design of URSNet-44 is shown in this figure.

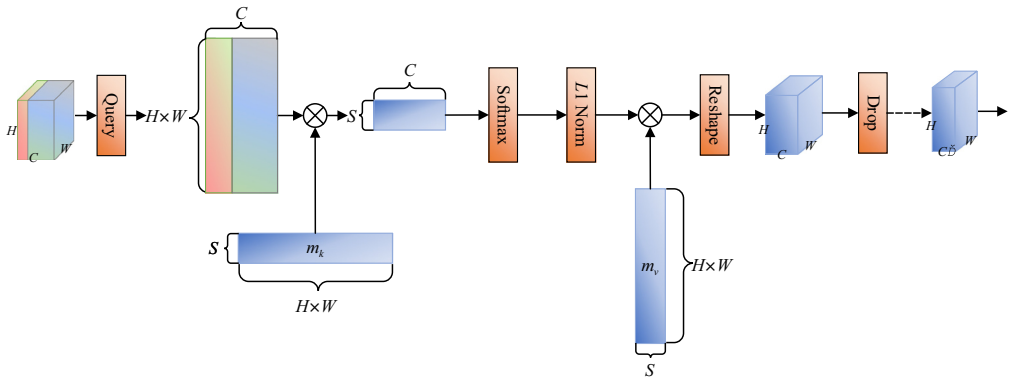

Fig. 5 The guided attention module. The dashed line on the far right of the figure indicates that the drop operation will output feature maps with different numbers of channels based on the parameters set. C' represents the variable number of channels.

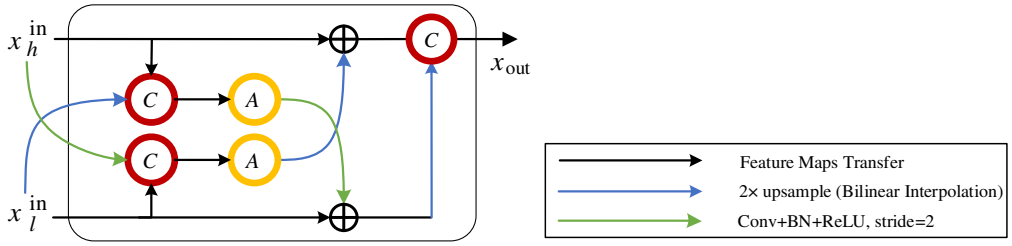

Fig. 6 Dual-guided attention module. *C* stands for concatenate operation. *A* represents the GA module.

3.3 Dual-Guided Attention

Inspired by EA,[17] we acknowledge that a linear complexity attention mechanism can be tailored for real-time semantic segmentation, ensuring no compromise in performance. As illustrated in Fig. 5, we have modified the original EA and named it guided attention (GA). Building upon GA, we introduce the dual-guided attention module, as illustrated by Fig. 6. This module begins using a parallel structure to down-sample the output feature map from the high-resolution branch (X_h^{in} in Fig. 6) to half its size via a 3×3 convolutional layer, and simultaneously, it up-samples the feature map from the low-resolution branch (X_l^{in} in Fig. 6) to twice its size through bilinear interpolation. Following this, both feature maps are concatenated with the other branch's output. After concatenation, we obtain two feature maps of different resolutions but with the same channel number, ready for two separate guided attention computations. Although we will delve into the technical specifics of GA later, it is essential to note that the output of GA maintains the same number of channels as the previous feature map (X_l^{in} and X_h^{in}), primarily achieved through a dropout layer in the final stage of GA. Consequently, we directly add the output of GA to the output of each branch at the corresponding resolution to form a residual structure. Finally, we double the scale of the lower-resolution feature map and concatenate it with the higher-resolution feature map, using the resulting combination as the final feature map (X_{out} in Fig. 6). The segmentation head receives the final feature map as input and performs prediction, outputting a segmentation image.

The specifics of guided attention are illustrated in Fig. 5. For the given input feature map $F \in \mathbb{R}^{N \times d}$, N is the number of pixels in images and d stands for the number of feature dimensions, and in the figure, it is $H \times W$. Drawing inspiration from the design of EA, we introduce two separate linear layers as external learnable components. They are described as M_k, $M_v \in \mathbb{R}^{N \times S}$, where S is the number of nodes in the fully connected layer. In this article, we adopt $S = 64$ as a hyperparameter. M_k and M_v serve as the key and value in the attention calculation. It is straightforward to infer that the input feature map F serves as the query. We briefly describe the calculation of GA as

$$A = \text{Norm}(F_{\text{in}} \otimes M_k^T) \tag{3}$$

$$F_{\text{mid}} = A \otimes M_v^T \tag{4}$$

$$F_{\text{out}} = \text{Drop}(F_{\text{mid}}). \tag{5}$$

The $\otimes$ in Eqs. (3) and (4) is the inner product between the input feature map F_{in} and the transpose of external learnable units M. To overcome the issue of attention map A being overly sensitive to the scale of the input features, we ensure that our GA is solely influenced by the number of dimensions present in the data from the feature map. To achieve this, we adopt the double-normalization technique proposed in Ref. 23, which involves normalizing both columns and rows independently. This double-normalization corresponds to the "Norm" operation in Eq. (1) and is described as follows:

$$\tilde{A} = F_{\text{in}} \otimes M_k^T, \tag{6}$$

$$\hat{a}_{i,j} = \frac{\exp(\tilde{a}_{i,j})}{\sum_k \exp(\tilde{a}_{k,j})}, \tag{7}$$

$$a_{i,j} = \frac{\hat{a}_{i,j}}{\sum_k \hat{a}_{i,k}}, \tag{8}$$

Matrix $\tilde{A}$ is the output of Eq. (6); it is the original output of the first step calculation, and the size is (k, k). The following equation indicates that we perform a softmax calculation on each row of the matrix $\tilde{A}$, and the output of each element is $\hat{a}_{i,j}$. After that, we do an $L1$ normalization at each column of the matrix. Compared with the self-attention mechanism, the attention mechanism we designed successfully reduces a significant number of learnable parameters, lowering the computational complexity from $O(N^2)$ to $O(NSd)$.

At the end of each GA module, a drop operation is performed. This operation effectively reduces the number of channels in the output feature map without the need for a 1×1 convolution to compress the channels, thus lowering the computational load. In addition, the dropout layer introduces nonlinearities and is a standard component in fully connected layers. The drop operation mainly relies on the features of the programming framework to randomly set a certain number of channels to zero on the three-dimensional feature map and then delete the zeroed channels, resulting in the required number of channels (C' in Fig. 5). After this drop operation, the number of channels in the feature map output by GA aligns with the number of channels before entering the DGA module, facilitating the final residual calculation within the DGA.

With the fusion of the feature maps from the two branches in the DGA, the computation across the entire network is completed. Finally, these feature maps are fed into a segmentation head, and the ultimate segmentation result is obtained through bilinear interpolation.

4 Experiments

4.1 Datasets and Evaluation Metrics

Cityscapes[10] is designed for the semantic understanding of urban street scenes from a driving perspective. The dataset provider collected a vast amount of driving video data from 27 cities in Germany and selected 5000 images for meticulous annotation. The dataset provider divided the dataset based on the following principles: in large, medium, and small cities; in the geographic west, center, and east of the country; in the geographic north, center, and south of the country; and at the beginning, middle, and end of the year. The data are split at the city level, i.e., a city is completely within a single split. The dataset divided accordingly contains 2975 training images, 500 validation images, and 1525 test images. All images are uniformly sized with a resolution of 2048×1024 pix. In our experiments, we adhere to the standard of 19 semantic categories instead of the original 30. We exclusively use the finely annotated images to train and validate our network.

The CamVid database[11] is a road scene dataset for semantic segmentation. It consists of 701 densely annotated frames, each with a resolution of 960×720 pix. These images were carefully selected by the dataset provider from a 55-min daytime video and a 31-min dusk video, and all were meticulously annotated. They encompass a total of 11 categories, such as pedestrians, bicycles, roads, and buildings. The images are not continuous and are divided in a ratio of

2:1 between daytime and dusk images. According to this division, the original dataset provided by the dataset provider includes 367 training images, 101 validation images, and 233 test images. Due to the small number of images in this dataset, we merged the training and validation images as training data in our experiments, using the test images as validation data, ultimately achieving a nearly 7:3 training-to-validation ratio.

All the methods tested in the experiment, including the model proposed in this paper, we use mean of class-wise intersection over union (mIoU) and frames per second (FPS) as the evaluation metrics. However, for some comparative methods whose FPS cannot be accurately measured using our scripts, we assess their performance based on the average inference delay for a single image, with milliseconds as the unit of measurement.

Where *TPc* represents the number of pixels correctly predicted to belong to category *c*, *FPc* denotes the number of pixels incorrectly predicted as class *c*, and *FNc* refers to the number of pixels that truly belong to class *c* but were not predicted as such. Ideally, if all pixels are predicted correctly, the Intersection over Union (IoU) for an individual category would be 1, and the mIoU for the entire model would also be calculated as 1. Finally, *n* represents the total number of categories, and thus, mIoU assesses the overall performance of the model's semantic segmentation capabilities

$$\mathrm{mIoU} = \frac{1}{n}\sum_{1}^{c}\frac{TP_c}{TP_c + FP_c + FN_c}. \tag{9}$$

For FPS, n represents the total number of samples included in the validation, b denotes the batch size used during validation, t indicates the total time required for all samples to complete validation, and t_1 refers to the time taken to validate the first batch

$$\mathrm{FPS} = \frac{n-b}{t-t_1}. \tag{10}$$

4.2 Implementation Details

4.2.1 *Training settings*

As our model does not utilize pretrained weights, we must train it from scratch on the training sets of both the Cityscapes and CamVid datasets. We chose the stochastic gradient descent (SGD) algorithm for optimization, setting the momentum at 0.9. In addition, we implement a warm-up strategy and the "polynomial decay" learning rate scheduler; we set the warm-up iterations to 1/100 of the total number of iterations. For the Cityscapes dataset, we randomly crop the input images to a resolution of 512×512 pix to expand the training set and allow for a batch size of 32 on a single Nvidia Quadro A6000 graphics processing unit (GPU), which has 48 gigabytes of video memory. The initial learning rate is set at 0.01 with a weight decay of 1×10^{-4} in the optimizer. The training will run for 1000 epochs to ensure adequate learning. For the CamVid dataset, we adjusted the cropped image size to 672×672 pix and reduced the batch size to 16. We set the initial learning rate at 0.005 and applied a weight decay of 5×10^{-5}. Furthermore, considering the reduced data compared with Cityscapes, we decreased the number of epochs to 400.

4.2.2 *Inference settings*

For a fair comparison, all inference experiments were conducted using CUDA 12.0, Pytorch1.13.1, and CUDNN 8.7, without TensorRT acceleration, on a single Nvidia RTX 3070 GPU. The batch size for inference was set to 1 for all experiments. Our method processes images at the full resolution of both datasets, without resizing. As mentioned previously, some comparative methods resize images, generate predictions, and thereby affect the accuracy of our FPS calculations. For these methods, we present the average inference delay for a single image, including the resizing steps, and calculate an approximate FPS excluding the resizing steps. We also report the average inference delay for a single image using our method.

Table 2 Ablation study on BiDGANet-Large evaluated on the Cityscapes validation set.

High-resolution branch	Low-resolution branch	Components				mIoU (%)
		Concate	EA	DGA	OHEM	
✓	—	—	—	—	—	43.8
—	✓	—	—	—	—	65.1
✓	✓	✓	—	—	—	69.7
✓	✓	✓	✓	—	—	73.6
✓	✓	✓	—	✓	—	77.2
✓	✓	✓	—	✓	✓	78.3

4.3 Ablation Experiments on Cityscapes

In subsequent experiments, we adopt the large version of our network to firmly establish the effectiveness of each component within our methodology. This version has the highest number of parameters, which more clearly demonstrates improvements. Table 1 presents three distinct versions of our network, with a notable commonality: the number of channels remains consistent in the high-resolution branch across all versions. However, the primary differentiating factor lies in the varying number of channels within the low-resolution branch. Next, we explored the ablation results of various parts of the network.

a. Two branches: We first study the effects of the two individual branches. As shown in the first three rows of Table 2, the high-resolution branch alone achieves an mIoU of 43.8%, whereas the low-resolution branch alone attains an mIoU of 65.1%. However, with just one concate operation, the final result of the two branches can achieve 69.7% mIoU.
b. EA: By utilizing both branches and directly incorporating an EA module as proposed in paper,[17] we observe a significant improvement, demonstrating the module's effectiveness.
c. DGA: Based on our DGA module, we design the network shown in Fig. 1. Within our DGA module, low-level spatial details and high-level semantics interact and fuse with each other effectively. Compared with just using a single EA, the integration of our DGA module brings about a nearly 4% improvement. Finally, we use online hard example mining (OHEM)[24] to replace the standard cross-entropy loss, resulting in a slight additional improvement.

Furthermore, to demonstrate the advantages of our URSNet-44, we conducted an ablation experiment using BiDGANet-Light. With all other network components remaining unchanged, we replaced URSNet-44 with ResNet-34, ResNet-50, and the "semantic branch" from BiSeNetV2 then performed experiments on the Cityscapes dataset. The input image size is 2048×1024 pix. The results are shown in Table 3. The experimental results reveal that the

Table 3 Ablation study on the effect of replacing the backbone of BiDGANet-Light evaluated on the Cityscapes validation set. FLOPS in the following table stands for Floating Point Operations Per Second.

Low-resolution branch	mIoU (%)	Params (M)	FLOPs (G)
ResNet-34	71.8	23.67	143
ResNet-50	71.4	26.26	163
Semantic branch (from BiSeNetV2)	72.7	4.26	99
URSNet-44-Light (ours)	72.1	3.03	103

URSNet-44, constructed using URS blocks, has fewer parameters. This suggests that although further reducing the number of parameters, it does not bring about any additional performance degradation. In addition, it not only maintains a comparable number of parameters to handcrafted backbones but also has a modular stacked structure similar to established backbones such as ResNet, allowing for easy configuration of various network scales and standalone pretraining.

4.4 Experiments on Cityscapes

We compared our network with previous real-time methods using the experimental configuration described above. Table 4 displays each method's model information, including input resolution size, mIoU, GPU device, and frames per second. For some methods, we adhered to their original training settings and evaluated their performance on our device, marking these with an asterisk. All other data were obtained directly from the original papers describing these methods. The experimental evaluation demonstrates that our method achieves competitive results. Specifically, the light version excels in terms of speed, the basic version offers a favorable balance between speed and accuracy, and the large version attains accuracy levels that are on par with state-of-the-art methods.

Figure 7 shows the visualization of segmentation results on the Cityscapes validation set. All images are presented in their original resolution of 2048×1024 pix. We use dashed boxes to highlight the vital contrasting sections. The black dashed boxes show the difference between the two version networks for some object segmentation details. Increasing the number of parameters in the low-resolution branch makes the segmentation result of some objects clearer. The segmentation accuracy of the network can be improved, but the speed will slow down. The white dashed boxes show where our network is better than our baseline BiSeNetV2. For certain categories that are particularly challenging to distinguish, the BiDGANet-Large demonstrates fewer misclassifications. Furthermore, our approach performs better in segmenting smaller objects, such as traffic lights, traffic signs, and poles. Further data explanation is provided in Table 5.

Table 4 Accuracy and speed comparison on the Cityscapes validation set. Bold represent the methods we proposed in this paper.

Method	Resolution (pix)	GPU	mIoU (%)	Frames (FPS)	Time (ms)
DFANet A[12]	1024 × 1024	Titan X	71.3	100	—
DFANet B[12]	1024 × 1024	Titan X	67.1	120	—
PP-LiteSeg-B_1[25]	1024 × 512	GTX 1080ti	73.9	196	—
PP-LiteSeg-B_2[25]	1536 × 768	GTX 1080ti	77.5	102	—
PIDNet-L (sota)[26]	2048 × 1024	RTX 3090	80.6	31	—
RTFormer (sota)[27]	2048 × 1024	RTX 2080ti	79.3	39	—
ICNet*[5]	2048 × 1024	RTX 3070	68.9	58	20
$STDC_2$-Seg50*[28]	1024 × 512	RTX 3070	73.5	71	11
$STDC_2$-Seg75*[28]	1536 × 768	RTX 3070	76.3	53	16
DDRNet-23-silm*[14]	2048 × 1024	RTX 3070	77.2	78	12
DDRNet-23*[14]	2048 × 1024	RTX 3070	78.8	46	22
BiSeNet V2*[6]	2048 × 1024	RTX 3070	73.1	143	7
BiSeNet V2-L*[6]	2048 × 1024	RTX 3070	75.6	41	24
BiDGANet-Light (ours)	2048 × 1024	RTX 3070	72.1	169	6
BiDGANet-Base (ours)	2048 × 1024	RTX 3070	75.9	80	12
BiDGANet-Large (ours)	2048 × 1024	RTX 3070	78.3	56	18

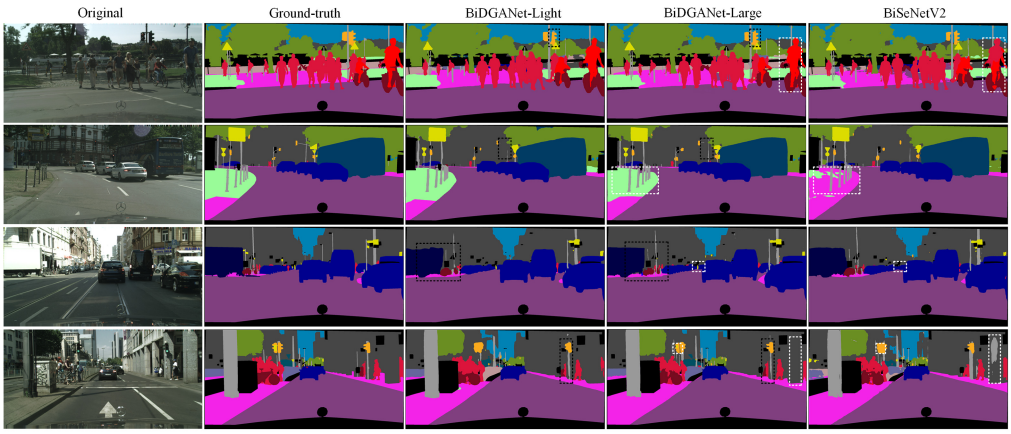

Fig. 7 Visualization of the segmentation results on the Cityscapes validation set.

Table 5 IoU of each category on the Cityscapes validation set.

Category	BiSeNetV2 (%)	BiSeNetV2-L (%)	BiDGANet-Light (%)	BiDGANet-Large (%)
Road	86.8%	90.4%	85.4%	92.4%↑
Sidewalk	72.6%	73.4%	73.6%	79.7%↑
Building	79.7%	83.5%	82.3%	90.5%↑
Wall	51.6%	53.2%	49.5%	51.4%
Fence	58.6%	59.7%	57.3%	60.9%
Pole	64.1%	69.5%	67.6%	73.9%
Traffic light	65.4%	66.2%	68.1%	72.4%
Traffic sign	64.6%	64.8%	70.3%	72.9%
Vegetation	79.3%	81.1%	75.7%	80.9%
Terrain	65.3%	70.6%	61.4%	71.8%↑
Sky	82.4%	85.1%	81.5%	90.4%↑
Person	76.2%	78.5%	76.5%	79.3%
Rider	74.4%	76.7%	71.7%	77.5%↑
Car	84.3%	86.1%	81.4%	86.8%↑
Truck	78.5%	81.3%	76.2%	83.2%↑
Bus	82.9%	83.6%	81.5%	86.1%↑
Train	65.7%	68.2%	64.3%	69.3%↑
Motorcycle	76.2%	78.8%	72.1%	79.8%↑
Bicycle	80.2%	85.9%	74.7%	88.4%↑
Params	**5.7M**	**49.5M**	**4.9M**	**55.7M**
mIoU	**73.1%**	**75.6%**	**72.1%**	**78.3%**

Meanwhile, we present the IoU of each category on the Cityscapes dataset in Table 5. The data in bold indicate that BiDGANet has made improvements in both the number of parameters and segmentation accuracy compared to the baseline. The black arrows (↑) indicate the categories where the large version of BiDGANet shows a significant improvement over the light version

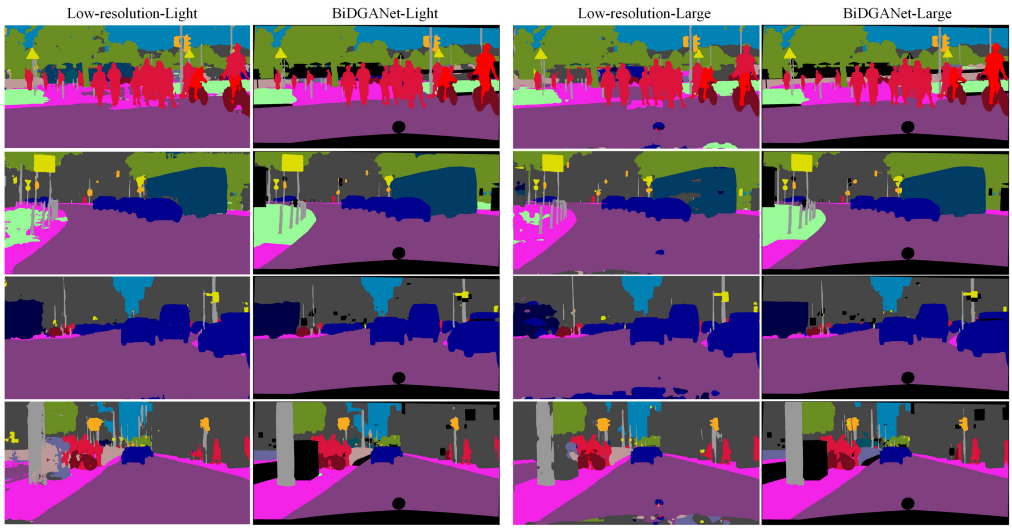

Fig. 8 Visual comparison of low-resolution branch output with full model output on the Cityscapes validation set.

after increasing the number of parameters. Overall, BiDGANet achieves improvements in most of the categories.

In addition, Fig. 8 illustrates the segmentation results directly from the feature map output of the low-resolution branch of the BiDGANet network before DGA processing, alongside a comparison with the output of the full model. It is evident that the inclusion of high-resolution branches and DGA modules significantly reduces misclassifications and enhances the contour distinction among various categories, resulting in clearer and more distinct segmentations with minimal overlap among similar categories.

4.5 Experiments on CamVid

We also evaluated our method on the CamVid dataset, using the original resolution of 960×720 pix for all methods tested; hence, the resolutions are not specified in Table 6.

Table 6 Accuracy and speed comparison on the CamVid test set. Bold represents the method we proposed in this paper.

Method	GPU	mIoU (%)	Frames (FPS)
ENet[29]	—	51.3	—
ICNet[5]	Titan X	67.1	35
DFANet A[12]	Titan X	64.7	120
BiSeNet V1[9]	Titan X	65.6	175
BiSeNet V1-L[9]	Titan X	68.7	117
STDC1-Seg[28]	GTX 1080ti	73	197
STDC2-Seg[28]	GTX 1080ti	73.9	152
S^2-FPN18[30]	GTX 1080ti	69.5	107
S^2-FPN34[30]	GTX 1080ti	71.1	124
PP-LiteSeg-T[25]	RTX 2080ti	73.3	222
PP-LiteSeg-B[25]	RTX 2080ti	75.0	154
BiSeNet V2*[6]	RTX 3070	72.4	33
BiSeNet V2-L*[6]	RTX 3070	73.3	110
BiDGANet-Base (ours)	RTX 3070	75.5	121

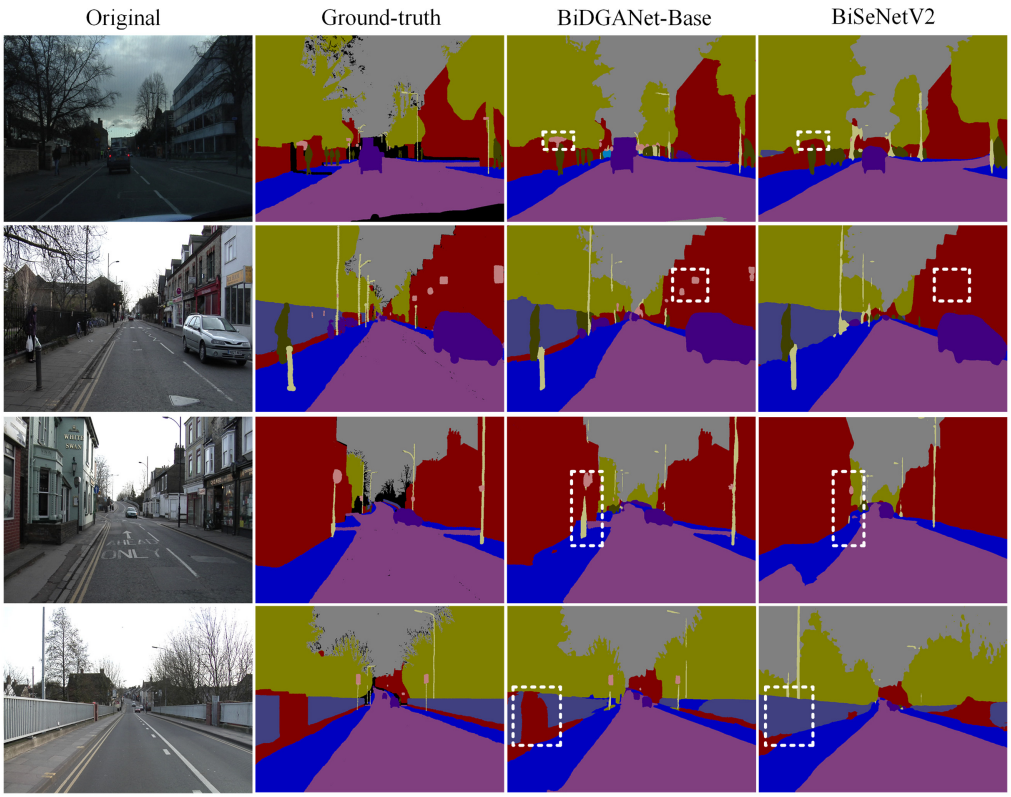

Fig. 9 Visualization of segmentation results on the CamVid test set.

Similar to Table 4, the asterisk indicates that we replicated the results on our device and in our environment setting. The results on the CamVid dataset demonstrate that our method also achieves excellent performance, indicating that it can adapt to various image qualities and has strong generalization capabilities.

Figure 9 below presents the visualization of BiDGANet's segmentation results on the CamVid dataset. From left to right, each column displays the input image, the label, the output from BiDGANet-Base, and the output from BiSeNetV2. All images retain the original resolution of 960×720. The white dashed box shows that our method performs better than the baseline on some small object segmentation. The experiments indicate that the BiDGANet network is not sensitive to image size and continues to perform well on datasets with relatively lower image quality, demonstrating robust generalization capabilities.

5 Conclusions

As the research in the field of semantic segmentation becomes more and more focused on the real-time performance of networks, researchers have carried out a lot of work on the design of model architecture, among which the two-branch structure is a very popular architecture. However, many network models based on the two-branch architecture face the performance bottleneck of feature fusion at the end of networks. In this paper, to address the challenge of terminal feature fusion in two-branch networks commonly used in real-time semantic segmentation for road scenes, we propose a new two-branch network utilizing the URS module and the dual-guided attention mechanism. By stacking URS blocks, we have designed a lightweight multi-scale feature extraction backbone and established the low-resolution branch based on this. In addition, the dual-guided attention module proposed in this paper enables the BiDGANet network to efficiently integrate the two branches. The network achieved competitive results in two mainstream benchmark tests. The BiDGANet-Large achieves 78.3% mIoU and 56 FPS on the Cityscapes dataset. The BiDGANet-Base achieves 75.5% mIoU and 121 FPS on the CamVid dataset. In conclusion, this work largely meets its expectations. Furthermore, due to the simple and flexible network architecture, BiDGANet can adapt its parameter scales to

different scenarios. Future research will focus on employing neural architecture search techniques for deeper structural optimization and developing this model as a baseline for future work.

Disclosures

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Code and Data Availability

Data are contained within the article. Our codes and modules have been open-sourced at https://github.com/LikeLidoA/BiDGANet/tree/main/My.

References

1. L. Shelhamer, E. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(4), 640–651 (2017).
2. L.-C. Chen et al., "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(4), 834–848 (2018).
3. V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: a deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(12), 2481–2495 (2017).
4. H. Zhao et al., "Pyramid scene parsing network," in *Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recognit.*, Honolulu, HI, USA, 21-26 July 2017, pp. 6230–6239 (2017).
5. H. Zhao et al., "ICNet for real-time semantic segmentation on high-resolution images," *Lect. Notes Comput. Sci.* **11207**, 418–434 (2018).
6. C. Yu et al., "BiSeNetv2: bilateral network with guided aggregation for real-time semantic segmentation," *Int. J. Comput. Vis.* **129**(11), 3051–3068 (2021).
7. S. Mehta et al., "ESPNet: efficient spatial pyramid of dilated convolutions for semantic segmentation," *Lect. Notes Comput. Sci.* **11214**, 561–580 (2018).
8. S.-Y. Lo et al., "Efficient dense modules of asymmetric convolution for real-time semantic segmentation," in *1st ACM Int. Conf. Multimedia in Asia*, 15–18 December 2019, Beijing, China, pp. 1–6 (2019).
9. C. Yu et al., "BiSeNet: bilateral segmentation network for real-time semantic segmentation," *Lect. Notes Comput. Sci.* **11217**, 334–349 (2018).
10. M. Cordts et al., "The Cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, 26 June–1 July 2016, Las Vegas, NV, USA, pp. 3213–3223 (2016).
11. G. J. Brostow et al., "Segmentation and recognition using structure from motion point clouds," *Lect. Notes Comput. Sci.* **5302**, 44–57 (2008).
12. H. Li et al., "DFANet: deep feature aggregation for real-time semantic segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, 16-20 June 2019, Long Beach, CA, USA, pp. 9522–9531 (2019).
13. R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-sCNN: fast semantic segmentation network," arXiv: 1902.04502 (2019).
14. Y. Hong et al., "Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes," arXiv:2101.06085 (2021).
15. J. Fu et al., "Dual attention network for scene segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, 16–20 June 2019, Long Beach, CA, USA, pp. 3141–3149 (2019).
16. Z. Huang et al., "CCNet: criss-cross attention for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(6), 6896–6908 (2023).
17. M.-H. Guo et al., "Beyond self-attention: external attention using two linear layers for visual tasks," *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(5), 5436–5447 (2023).
18. A. Vaswani et al., "Attention is all you need," in *Annu. Conf. Neural Inf. Process. Syst.*, 4–7 December 2017, Long Beach, CA, USA, pp. 5999–6009 (2017).
19. L. Liang et al., "Bilateral network with residual U-blocks and dual-guided attention for real-time semantic segmentation," arXiv:2310.20305 (2023).
20. R. R. Selvaraju et al., "Grad-CAM: visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.* **128**(2), 336–359 (2020).
21. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, 27–30 June 2016, Las Vegas, NV, USA, pp. 770–778 (2016).
22. O. Ronneberger, P. Fischer, and T. Brox, "UNet: convolutional networks for biomedical image segmentation," *Lect. Notes Comput. Sci.* **9351**, 234–241 (2015).

23. M.-H. Guo et al., "PCT: point cloud transformer," *Comput. Visual Media* **7**(2), 187–199 (2021).
24. A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, 26 June–1 July 2016, Las Vegas, NV, USA, pp. 761–769 (2016).
25. J. Peng et al., "PP-LiteSeg: a superior real-time semantic segmentation model," arXiv:2204.02681 (2022).
26. 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. and Pattern Recognit.*, 17–24 June 2023, Vancouver, BC, Canada, pp. 19529–19539 (2023).
27. J. Wang et al., "RTFormer: efficient design for real-time semantic segmentation with transformer," in *Annu. Conf. Neural Inf. Process. Syst.*, 28 November–9 December 2022, New Orleans, LA, USA (2022).
28. M. Fan et al., "Rethinking BiSeNet for real-time semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. and Pattern Recognit.*, Virtual, Online, 19-25 June 2021, pp. 9716–9725 (2021).
29. A. Paszke et al., "ENet: a deep neural network architecture for real-time semantic segmentation," arXiv: 1606.02147 (2016).
30. M. A. M. Elhassan et al., "S^2-FPN: scale-ware strip attention guided feature pyramid network for real-time semantic segmentation," arXiv:2206.07298 (2022).

Liang Liao received his BS degree in computer science and technology from Guizhou University, China, in 2021. He is currently pursuing an MS degree in computer science and technology at the School of Computer Science and Technology, Guizhou University. His research interests include deep learning, computer vision, image processing, and semantic segmentation.

Liang Wan received his BS degree from Guizhou University, China, in 1995 and his PhD from Guizhou University, China, in 2009. From 2011 to 2013, he did a research project at the post-doctoral workstation of the School of Information of Renmin University of China. Currently, he is a professor at the School of Computer Science and Technology, Guizhou University, China. His research interests include deep learning and information security.

Biographies of the other authors are not available.