Complex & Intelligent Systems
https://doi.org/10.1007/s40747-023-01054-y

ORIGINAL ARTICLE

EBUNet: a fast and accurate semantic segmentation network with lightweight efficient bottleneck unit

Siyuan Shen[1] · Zhengjun Zhai[1] · Guanfeng Yu[2] · Youyu Yan[3] · Wei Dai[1]

Received: 21 December 2022 / Accepted: 20 March 2023

Abstract

It has been difficult to achieve a suitable balance between effectiveness and efficiency in lightweight semantic segmentation networks in recent years. The goal of this work is to present an efficient and reliable semantic segmentation method called EBUNet, which is aimed at achieving a favorable trade-off between inference speed and prediction accuracy. Initially, we develop an Efficient Bottleneck Unit (EBU) that employs depth-wise convolution and depth-wise dilated convolution to obtain adequate features with moderate computation costs. Then, we developed a novel Image Partition Attention Module (IPAM), which divides feature maps into subregions and generates attention weights based on them. As a third step, we developed a novel lightweight attention decoder with which to retrieve spatial information effectively. Extensive experiments show that our EBUNet achieves 73.4% mIou and 152 FPS on the Cityscapes dataset and 72.2% mIoU and 147 FPS on the Camvid dataset with only 1.57 M parameters. The results of the experiment confirm that the proposed model is capable of making a decent trade-off in terms of accuracy, inference, and model size. The source code of our EBUNet is available at (https://github.com/Skybird1101/EBUNet).

Keywords Lightweight semantic segmentation · Decoder · Bottleneck unit · Attention mechanism

Introduction

Semantic segmentation assigns a category to each pixel in the input image, which is a dense classification task in computer vision. As a result of dense segmentation predictions, it has a wide range of applications in the real world, including autonomous driving [1], virtual reality [2], scene understanding [3] and so on.

Deep learning technology has made significant progress in many fields, including fault diagnosis [4], automation control [5], and semantic segmentation. Using Convolution Neural Network (CNNs), some advanced semantic segmentation methods have achieved a significant progress in terms of accuracy, including PSPNet [6], RefineNet [7], and DeepLab [8] in recent years. However, they typically possess a complex structure and hundreds of convolution layers and feature channels, which consume a large amount of computing resource and limit the wide application in real world. Consequently, it remains a challenge to design a lightweight network that achieves real-time performance and a satisfactory accuracy.

At present, many lightweight semantic segmentation methods have been developed to achieve a good tradoff between inference speed and prediction accuracy, which can be broadly divided into two categories: 1. Mode compression: eliminate unnecessary calculations and reduce the amount of data that needs to be stored by simplifying models, including pruning networks [9], and knowledge distillation [10, 11]. 2.

✉ Youyu Yan
yanyouyu@mail.nwpu.edu.cn

Siyuan Shen
shensiyuan@mail.nwpu.edu.cn

Zhengjun Zhai
zhaizjun@nwpu.edu.cn

Guanfeng Yu
yuguanfeng@mail.nwpu.edu.cn

Wei Dai
daiwei1016@mail.nwpu.edu.cn

1 School of Computer Science, Northwestern Polytechnical University, No. 127 Youyi Xilu, Xi'an 710072, Shaanxi, China

2 AVIC Xi'an Aeronautics Computing Technique Research Institute, jinye 2rd road, Xi'an 710065, Shaanxi, China

3 School of Marine Science and Technology, Northwestern Polytechnical University, No. 127 Youyi Xilu, Xi'an 710072, Shaanxi, China

Published online: 17 April 2023

Convolution decomposition: build shallow networks from the perspective of reducing convolutional computational costs, such as depth-wise separable convolution and group convolution.

Based on the idea of convolution decomposition, MobileNet [12] adopted depth-wise convolution to construct the backbone and achieved a fast running speed compared to the traditional convolution. DABNet [13] introduced a depth-wise asymmetric bottleneck, which achieved a well-balanced performance in terms of running speed and segmentation accuracy.

Furthermore, multi-scale feature fusion is often employed in the design of lightweight semantic segmentation. For example, a MAD module was introduced in LMFFNet [14] to combine the different levels of features into one stage and generate more accurate attention maps. CFNet [15] implemented a channel attention mechanism and a cross-fusion module to enhance the fusion effect.

In this paper, we present a novel lightweight network called EBUNet, which employs an encoder–decoder architecture, to achieve real-time semantic segmentation. Our EBUNet is mainly composed of three modules: Efficient Bottleneck Unit (EBU), Image Partition Attention Mechanism (IPAM) and a Lightweight Attentional Decoder (LAD).

Using depth-wise separation and dilated convolution simultaneously, we devise a novel residual-like structure named EBU, which achieves high accuracy with low computation costs. The IPAM module is designed to enhance the feature. The LAD module is presented to recover the spatial information and generate the segmentation results. We compare our method with other semantic segmentation methods in terms of parameters and mIoU. The results can be seen in Fig. 1.

Our main contributions can be listed as follows:

- A novel Efficient Bottleneck Unit (EBU) module is proposed to gather detailed and semantic information. Based on the EBU module, we have implemented two EBU blocks as the main part of our EBUNet.
- We propose a novel Lightweight Attentional Decoder (LAD) for recover spatial information effectively. In the proposed LAD module, different attention mechanisms are employed in the proposed LAD module to refine different levels of feature maps.
- We present a novel Image Partition Attention Module (IPAM) to refine the feature maps from different EBU blocks. The proposed IPAM module partitions the feature maps into sub-regions and generates an attention map based on the partitioned sub-regions.
- Our EBUNet obtains competitive results, it achieves 73.4% and 72.2% mIoU on Cityscapes and Camvid test sets along with 152 and 147 FPS, respectively.

The structure of this paper is organized as follows: Section II introduces some previous work about residual structures, lightweight semantic methods, and feature fusion methods. Section III presents our method, including the EBU module, IPAM module, and LAD module. Section IV discusses the experiment details and results. Section V concludes the whole paper.

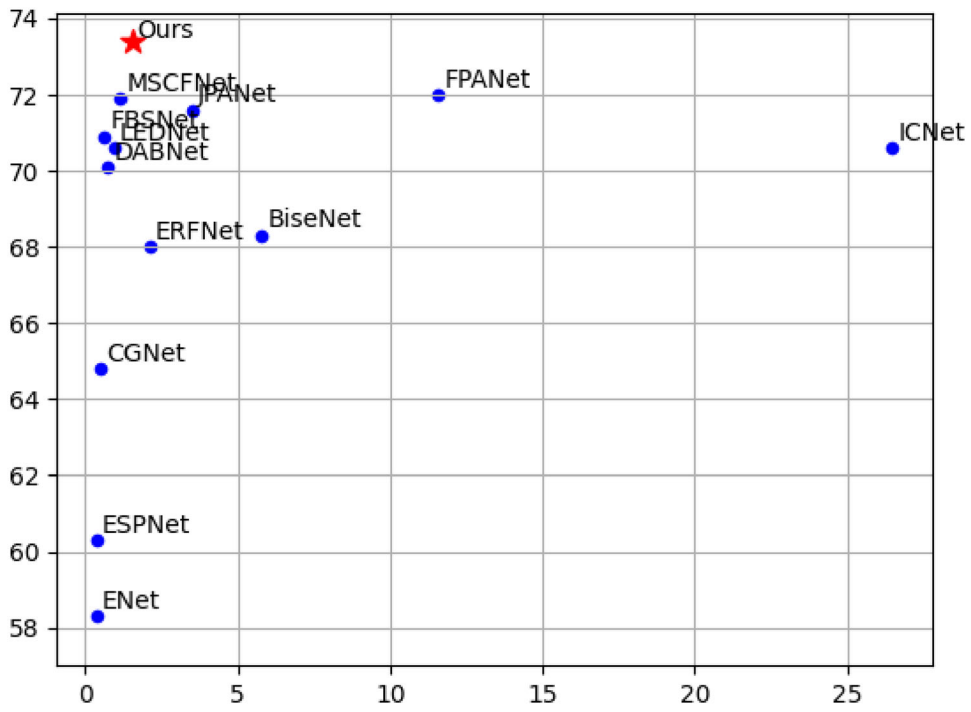

Fig. 1 Comparisons with other methods in terms of parameters and accuracy. Our EBUNet achieved a competitive result

Related works

In this section, we will review some related works, including residual structure, lightweight semantic networks, and multi-scale feature fusion methods.

Residual structure

Residual structure, which has been proven to be an effective way to overcome gradient explosion or vanishing problems. It was originally proposed in ResNet [16], and many residual-like structures have been proposed for various computer vision tasks since then. As an example, ShuffleNet [17, 18] developed lightweight backbone networks with depth-wise convolutions. LEDNet [19] introduced SS-nbt modules that combine factorized and dilated convolution for feature extraction. FBSNet [20] employed BRU modules to capture rich contextual information. DABNet [13] utilized dilated convolution in DAB modules to enlarge the receptive field, which helped to promote the detailed segmentation effect. MSCFNet [21] applied EAR modules to retrieve contextual and detailed information.

Lightweight semantic networks

In recent years, rapid-growing applications have required semantic segmentation approaches to run efficiently in real-world scenarios. The key concept of lightweight semantic segmentation is to maximize accuracy while minimizing feed-forward inference time.

A lot of attention has been paid to the design of lightweight semantic segmentation since ENet [22] was proposed. For example, Bisenet [23] introduced a dual-path structure. The context path was used for extracting contextual information, whereas the spatial path was used for extracting spatial information. A number of other approaches have been developed based on BiseNet, including STDCNet [24] and BiseNet-v2 [25], which produce more efficient and accurate results than the original BiseNet. Many lightweight networks have been developed in recent years to improve efficiency and effectiveness. For example, JPANet [26] presented a joint feature pyramid module for learning multi-stage features. FPANet [27] employed a feature pyramid fusion module to fuse features from different stages. RELAXNet [28] applied EBR and EABR modules to acquire context and detailed information.

Feature fusion

In different fields, feature fusion has different meanings. In signal processing, feature fusion is used to achieve high robustness by combining time and frequency information [29]. In deep learning technology, the feature fusion methods aim to fuse the feature maps from different stages.

There are two types of feature fusion methods that are commonly applied: channel-wise concatenation and element-wise addition. For example, in ContextNet [30] and Fast-SCNN [31], high-level feature maps are upsampled and then concatenated with low-level ones to achieve multi-scale feature fusion. The FBSNet [20] utilized element-wise addition for feature fusion, which combines features from the different branches. Since the attention mechanism was proposed, many attentional methods devoted to promoting the effect of multi-scale feature fusion. For example, The LMFFNet [14] introduced a FFM module for fusing different levels of feature maps, which employs an attention mechanism as well as depth-wise separable convolutions, ABCNet [32] utilized self-attention to fuse the feature maps from different stages.

Methodology

In this section, we fisrt examine the computation costs and parameters of the convolution operation. Then, we will introduce the components of our EBUNet, including EBU module, IPAM module, and LAD module. The architecture design of our EBUNet will be discussed at the end of this section.

Computation complexity analysis

The Convolutional Neural Networks (CNNs) are composed of convolutional layers and fully connected layers. In this section, we will discuss te computation complexity of the CNNs.

Before we start our discussion, we make some definitions to simplify our discussion. Defining a transformation function Φ to take C_{in} feature maps with a spatial size of $d \times d$ as inputs, and output C_{out} feature maps with the same size. C_{in} and C_{out} stand for the number of input and output channels, respectively. The convolutional kernel size is $k \times k$ and the stride is set to 1. Here, we use square feature maps and convolutional kernels for simplifying our discussion. We omit the bias and Batch normalization terms in the convolutional operation, which are often used in modern CNNs.

In this case, the number of parameters in the convolution is $k \times k \times C_{\text{in}} \times C_{\text{out}}$ and the computational complexity in terms of FLOPs is $k \times k \times C_{\text{in}} \times C_{\text{out}} \times d \times d$.

Based on the above conclusions, it is necessary to reduce the multiplication cost between $k \times k$ and $C_{\text{in}} \times C_{\text{out}}$, which is an effective way to cut down the size and the computation burden of convolutions. The depth-wise convolution applies this approach to explore compact models.

Compared to the standard convolution, the depth-wise separable convolution utilizes a single convolutional kernel

independently for each input feature map, thus generating the same number of output channels. Following that is a 1×1 convolution layer to merge the information of all output channels. The depth-wise separable decomposes the standard convolution into a depth-wise convolution and a point-wise convolution. By applying depth-wise separable convolution, the number of parameters becomes:

$$k \times k \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}} \tag{1}$$

and the computation complexity becomes:

$$k \times k \times C_{\text{in}} \times d \times d + C_{\text{in}} \times C_{\text{out}} \times d \times d \tag{2}$$

Based on the above equations, the amount of the parameters and computations are reduced by depth-wise separable convolution.

Efficient bottleneck unit

The EBU module is designed to extract semantic information more efficiently and effectively. Previous residual-like works, including bottleneck [22], SS-nbt [19], and EAR modules [21], have proven to be effective in the design of lightweight semantic segmentation.

As shown in Fig. 2d, we employ a 3×3 standard convolution to generate features and reduce the channels by half at the beginning of each EBU module. The output of the convolutional operation is then split into two branches, where each branch has 1/4 channels of the original input.

A convolutional kernel of 3×3 is used in the EBU module to preserve adequate spatial information for accurate segmentation. In order to improve computation efficiency, a 3×3 depth-wise convolution is employed in the left branch to acquire local information.

The right branch is developed to obtain adequate contextual information. For the purpose of enlarging the receptive field, we use a depth-wise dilated convolution without adding any additional parameters in the right branch.

For the sake of sharing information between two branches, we put the feature interaction operations through an element-wise addition between two branches. So as to the two branches can complement each other.

At the end of the EBU module, another 3×3 regular convolution is employed to integrate the multi-scale features and finally restore the number of channels as same as the number of input channels. The whole procedure can be expressed as follows:

$$F = f_{3\times3}(X) \tag{3}$$

$$F_1, F_2 = \text{Split}(X) \tag{4}$$

$$F_1 = f_{3\times3}^{\text{DW}}(F_1) \tag{5}$$

$$F_2 = f_{3\times3}^{\text{DDW}}(F_2) \tag{6}$$

$$F_2 = F_2 + F_1$$

$$F_1 = F_1 + F_2 \tag{7}$$

$$F_{\text{out}} = \text{Concat}(F1, F2) \tag{8}$$

$$F_{\text{out}} = X + F_{\text{out}}, \tag{9}$$

where, X is the input feature maps. F_1 and F_2 are the results of splitting operation. $f_{3\times3}$ represents standard 3×3 convolution. $f_{3\times3}^{\text{DW}}$ and $f_{3\times3}^{\text{DDW}}$ stand for depth-wise convolution and depth-wise dilated convolution. Concat means feature concatenation along with the channel dimension.

Image partition attention module

The Attention mechanism has been widely used in various segmentation methods, such as BiseNet [23], MSCFNet [21], DFANet [33], etc. We introduce an Image Partition Attention Module (IPAM) in this paper.

As Fig. 3 shows, the input features are partitioned into four regions through an average pooling operation. Then, global average pooling operations are applied to each partitioned sub-region in parallel.

Each partitioned sub-region S_i is then subjected to global average pooling simultaneously. The global average pooling operation is calculated as follows:

$$F_{\text{avg}} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_H(i, j). \tag{10}$$

Additionally, global average pooling is applied to the original input feature to acquire the global information. Following that, we use an element-wise addition to fuse the results from sub-region pooled features and global pooled features. This procedure is computed as follows:

$$T = \sum_i S_i + F_{\text{global}}, i \subset 1, 2, 3, 4, \tag{11}$$

where, S_i represents the pooled results of sub-region. F_{global} indicates the result from the pooled result from the original input.

The results from addition operation is then send into a projection layer by 1×1 convolution to generating attention weigh vector w.

To be specific, the addition results are compressed across the channel dimension, then the ReLU function is applied to introduced non-linearity. After that, the channel increasing layer is employed to recover the channel to the number of original input. A sigmoid function is used to generate the attention weight vector w. The operation of the projection layer can be expressed as follows:

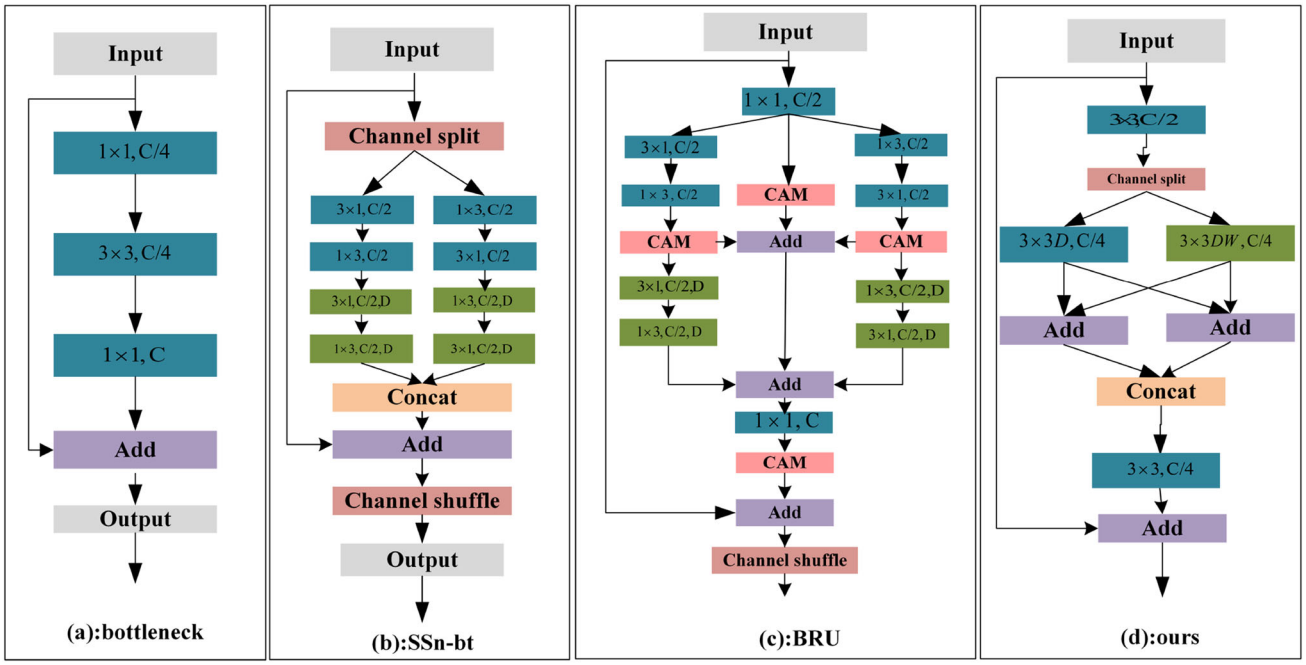

Fig. 2 Comparisons with several residual-like structures. **a** bottleneck [22] **b** SSn-bt [19] **c** BRU [20]. D means dilated convolution, DW stands for depth-wise convolution

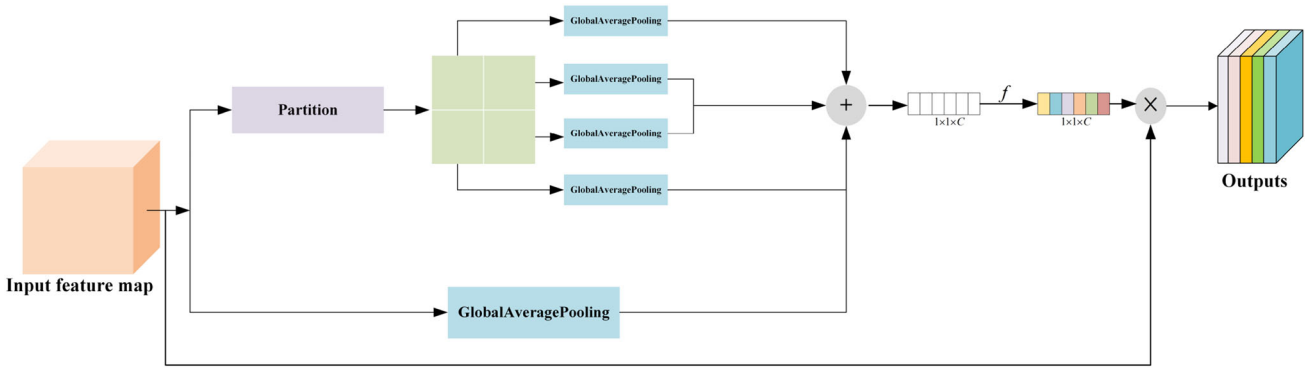

Fig. 3 Illustration of the proposed IPAM module

$$w = \sigma(\pi_2(\mathrm{ReLU}(\pi_1(T)))), \tag{12}$$

where, π_1 and π_2 represent the channel reduction and expansion function implemented by the two regular 1×1 convolution, respectively. ReLU indicates the Rectified Linear Unit function.

At the end of the IPAM module, we can obtain the final output F_{out} as follows:

$$F_{\mathrm{out}} = F_{\mathrm{in}} \otimes w, \tag{13}$$

where, F_{out} and F_{in} represents the input and output respectively, w means attention weights generated from IPAM module.

Lightweight attentional decoder

There are different roles assigned to encoders and decoders in encoder–decoder segmentation frameworks. The encoder is responsible for producing dense feature maps, whereas the decoder is responsible for upsampling the resolution of feature maps to match the original input size. It is possible to improve the accuracy of prediction with the use of well-designed decoders.

In our paper, we present a novel lightweight attentional decoder (LAD). It consists of two blocks and can fuse different-level features effectively. A channel attention module is proposed for the refinement of high-level feature maps, while a spatial attention module is proposed for the refine-

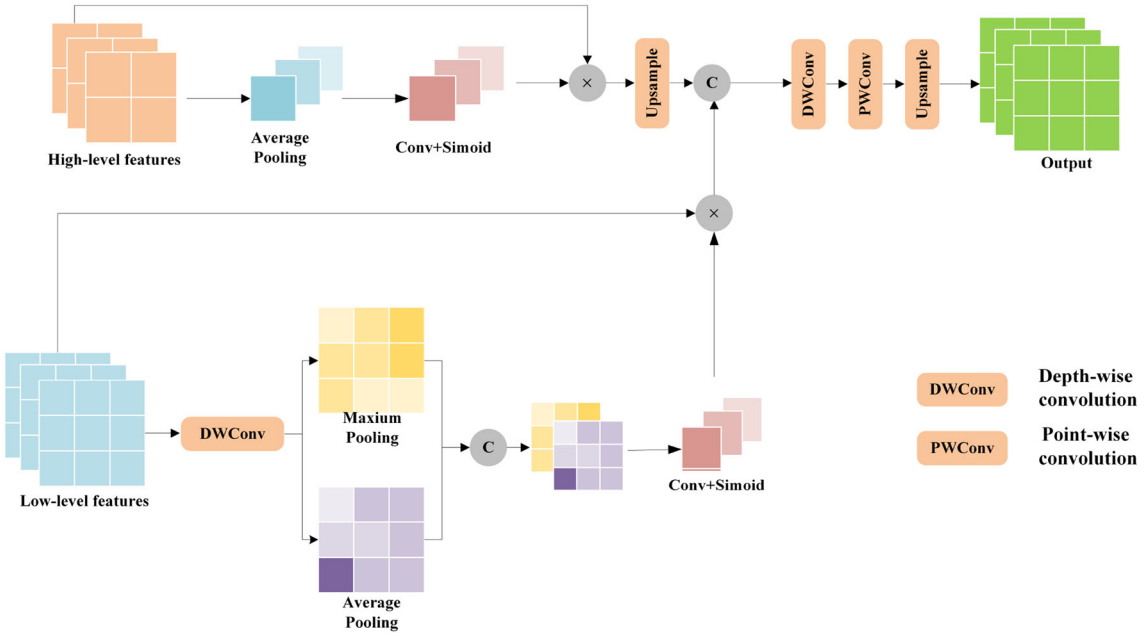

Fig. 4 Illustration of our LAD module

ment of low-level ones. The structure of our proposed LAD is shown in Fig. 4.

We present a spatial attention module (SAM) to make the low-level features pay more attention to informative features. Let X_L denote the input low-level feature maps, f_{conv} represents the regular convolution operation, f_{mean} and f_{max} are the mean operation and maximum in the channel dimension, respectively. The spatial attention map S is computed as follows:

$$S = \sigma(f_{\text{conv}} f_{\text{cat}}(f_{\text{mean}}, f_{\text{max}})), \tag{14}$$

where, $\sigma(.)$ represents the sigmoid function. After the transformation, the shape of low-level features changes from $C \times H \times W$ to $1 \times H \times W$. Finally, we element-wise multiply the input low-level feature X_L and the spatial weights map S to get our refined feature X_L^S:

$$X_L^S = X_L \otimes S, \tag{15}$$

where, $\otimes$ denotes the element-wise multiplication.

Our channel attention module (CAM) uses global average pooling to obtain global contextual informative and generates an channel attention map to refine the high-level features. Let $X_H(i, j)$ denotes values of X_H at pixel location (i, j). X_H represents input high-level feature maps. The global average pooling can be expressed as follows:

$$F_{\text{avg}} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_H(i, j). \tag{16}$$

Consequently, the shape of the high-level features changes from $C \times H \times W$ to $1 \times 1 \times C$. Following that, F_{avg} is fed into a convolution layer, and then passed through a sigmoid to generate channel attention map C:

$$C = \sigma(f_{\text{conv}}^{1\times1} F_{\text{avg}}). \tag{17}$$

The final weighted high-level feature are acquired by multiplying feature map and the attention map:

$$X_H^C = X_H \otimes C. \tag{18}$$

As a result of the abstracted spatial attention map produced from low-level features, we are able to identify the importance of each pixel, which focuses on locating objects and refining the corresponding shapes and boundaries with spatial details. On the other hand, the squeezed channel attention map generated from upsampled high-level features focuses on the global context to provide context information.

After that, the refined low-level features and high-level features are concatenated along with channel dimension. Finally, another upsampling operation is utilized to restore the feature map to its original size.

Architecture design of EBUNet

The overall architecture of the proposed EBUNet is shown in Fig. 5 and is listed in Table 1.

Initial Unit is employed at the beginning of EBUNet to adjust the resolution of the input images and eliminate the redundant information. Initial Unit is composed of three

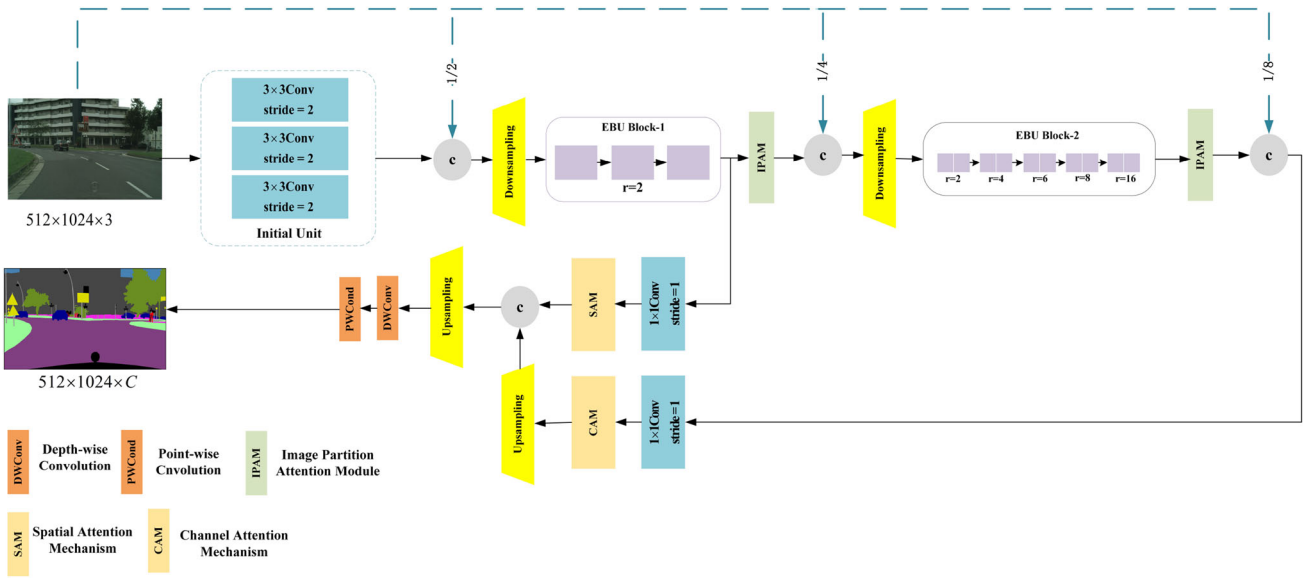

Fig. 5 Overall architecture of our EBUNet

Table 1 Overall architecture of EBUNet

Module	Operation	Channel number	Output Size
	3×3Conv	32	256×512
Initial unit	3×3Conv	32	256×512
	3×3Conv	32	256×512
Feature Fusion	Feature concatenation	32+3	256× 512
Downsampling	Downsampling	64	128× 256
ERU-Block 1	ERU module×3	64	128× 256
Feature fusion	Feature concatenation	128+3	128× 256
Downsampling	Downsampling	128	64× 128
ERU-Block 2	ERU module×10	128	64× 128
Feature Fusion	Feature concatenation	256+3	64× 128
LAD	LAD	Number_classes	512× 1024

consecutive standard convolutions. To be specific, the first convolution is used to reduce the image resolution by half. In the meanwhile, the channel number of the feature map is adjusted to 32. Afterwards, two 3 × 3 convolutions are utilized to obtain abundant contextual information.

Besides, downsampling operation is used to enlarge the receptive field. The downsampling operation is composed of two parallel branches: a standard 3 × 3 convolution with a stride of 2 and a 2 × 2 maximum pooling operation. Then the outputs of above the two parallel branches are concatenated along with the channel dimension.

After that, the feature map obtained by downsampling the output of the initial unit is input into the first EBU Block for dense feature extraction. The first EBU block contains three EBU modules with a dilated rate of 2. The input feature map of second EBU Block is 1/8 of the input, which contains 10 consecutive EBU modules with a gradually increasing dilated rates {2,2,4,4,6,6,8,8,16,16}. The IPAM is employed to refine the features from EBU block 1 and EBU block 2. Consequently, in the decoder phase, the LAD employs different kinds of attention mechanism for different-level feature maps and produces more accurate outputs.

Experiments

In this section, we first illustrate brief information about Cityscapes [34] and CamVid [35] datasets, following that, we introduce the training protocols for our experiments. Subsequently, ablation studies about several components of our EBUNet will be discussed. At the end of this section, we will discuss the performance of our method in the metric of prediction accuracy and running efficiency.

Datasets

We utilize Cityscapes and CamVid datasets in our training and testing experiments.

The Cityscapes dataset is a well-known dataset for semantic segmentation of urban scenes. There are 5000 fine-annotated images in the Cityscapes dataset: 2975 images for networks training, 500 images for networks validation, and 1525 images for networks testing. The original image resolution of Cityscapes is 2048×1024. For fair comparisons, we use the full resolution for performance evaluation in the validation and testing phases. In the training phase, the resolution is resized to 512×1024.

The CamVid dataset, derived from car-view videos, is another well-known urban scene dataset. The CamVid dataset consists of 701 images total: 367 images for the training phase, 101 images for the validation phase, and 233 images for the testing phase. The original resolution of CamVid dataset images is 720×960.

Training protocols

All the experiments are performed with one NVIDIA RTX 3090 GPU, CUDA 11.6, and cuDNN v8 on pytorch platform, Ubuntu 20.04 operating system with 32GB Memory.

We employ Mini-Batch Stochastic Gradient Descent [36] (SGD) in our optimization strategy, where we set the batch size to 8, the weight decay to 1×10^{-4}, the momentum to 0.9, and the initial learning rate to 4.5×10^{-2} in the training procedure of the Cityscapes dataset.

We train our EBUNet by using Adam optimizer when running experiments on the CamVid dataset. The initial learning is set to 1×10^{-3} and the weight decay is set to 2×10^{-4}.

Besides, polynomial policy is employed to adjust the learning rate in the training phase. The polynomial policy is expressed as the follow formula:

$$\mathrm{lr}_{\mathrm{cur}} = \mathrm{lr}_{\mathrm{init}} * \left(1 - \frac{\mathrm{cur\_epoch}}{\mathrm{max\_epoch}}\right)^{0.9}, \tag{19}$$

where, $\mathrm{lr}_{\mathrm{cur}}$ represents the learning rate in the current epoch, cur_epoch stands for the current epoch, max _epoch is the total epoch.

The max _epoch was set to 1000 during the training process for both the Cityscapes and CamVid datasets. During the training phase, data augmentation techniques, such as random scale, mean subtraction, and horizontal flipping are also applied. A variety of random parameters were set to transform training samples to different scales, including 0.75, 1.0, 1.25, 1.5, 1.75, and 2.0. We randomly cropped the training images and labels in the cityscapes dataset from the resolution of 2048×1024 to 512×1024.

Ablation studies

In this part, we design a series of ablation experiments to validate the effectiveness of some proposed components of our EBUNet. We conduct ablation studies on the EBU module and LAD module. Additionally, we investigate the influence of depth within the EBU block. We perform all the ablation experiments on the Camvid dataset.

Ablation on EBU module

The main part of our EBUNet is constructed using the EBU module. We devise two kinds of ablation study strategies to verify the effectiveness of our EBUNet. In the first step, we design a series of experiments to investigate the influence of different dilated rates. The second is that we compare our EBU module to some other residual structures, including DABNet [13] and ERFNet [37]. The ablation study results can be seen in Tables 2 and 3.

To study the effects of dilated rates, we devised five sequences with varying dilated rates and compared them with baseline. From Table 2, we can learn that when we set all the dialted rates in EBU modules to 2, the accuracy is 1.8% lower than the baseline. In addition, when we set a larger dialted rates sequence in EBU modules, the accuracy is 1.6% higher than R=2 but 0.2% lower than baseline.

Additionally, we designed experiments to test network performance using excessive dilated rates (32 and 48). As shown in Table 2, when the dilated rate was set to 32, the mIoU was decreased 69.8% and the FPS was also decreased from 147 to 140. Besides, both accuracy and speed decreased when all dilated rates were set to 48. We can concluded that the larger dilated rates would cause heavy computation cost. Furthermore, dilated convolution results are convolved from mutually independent subsets, which lose local information.

From Table 3, we can observe that when EBU modules are substituted with non-bottleneck, the forward inference is higher than EBU modules are used. However, the accuracy of our EBUNet is 1% higher than it. As a result, our EBU module strikes a good balance between accuracy and efficiency. Additionally, a visual comparison was also conducted and the results can be seen in Fig. 6.

Ablation on LAD module

LAD is used to recover the spatial information to the original input resolution. The ablation design of LAD is based on two strategies. In the first step, we compare our LAD with DABNet's decoder. We then discuss how our LAD is affected by the attention mechanism. Specifically, we performed ablation studies on the different attention mechanisms in our LAD. Results of ablation studies are presented in Tables 4 and 5.

Table 2 The experiment results about the dilated rates

Method	FPS (↑)	mIoU (%) ↑
EBUNet (R = 2)	144.84	70.4
EBUNet (R = {3,3,5,5,7,7,9,9,11,11})	145.06	71.9
EBUNet (R = {4,4,6,6,8,8,16,16,32,32})	143.75	72.0
EBUNet-baseline (R = {2,2,4,4,6,6,8,8,16,16})	146.55	72.2
EBUNet (R = {32,32,32,32,32,32,32,32,32,32})	140.91	71.3
EBUNet (R = {48,48,48,48,48,48,48,48,48,48})	141.53	70.0

Table 3 Experiment results on residual structures

Method	mIoU (%) ↑	Parameters (M) ↓	FPS ↑	FLOPS (G) ↓
EBUNet-baseline	72.26	1.57	152	24.13
EBUNet-non-bottleneck	71.2	2.40	160	34
EBUNet-DABModule	71.4	1.18	150	18.91

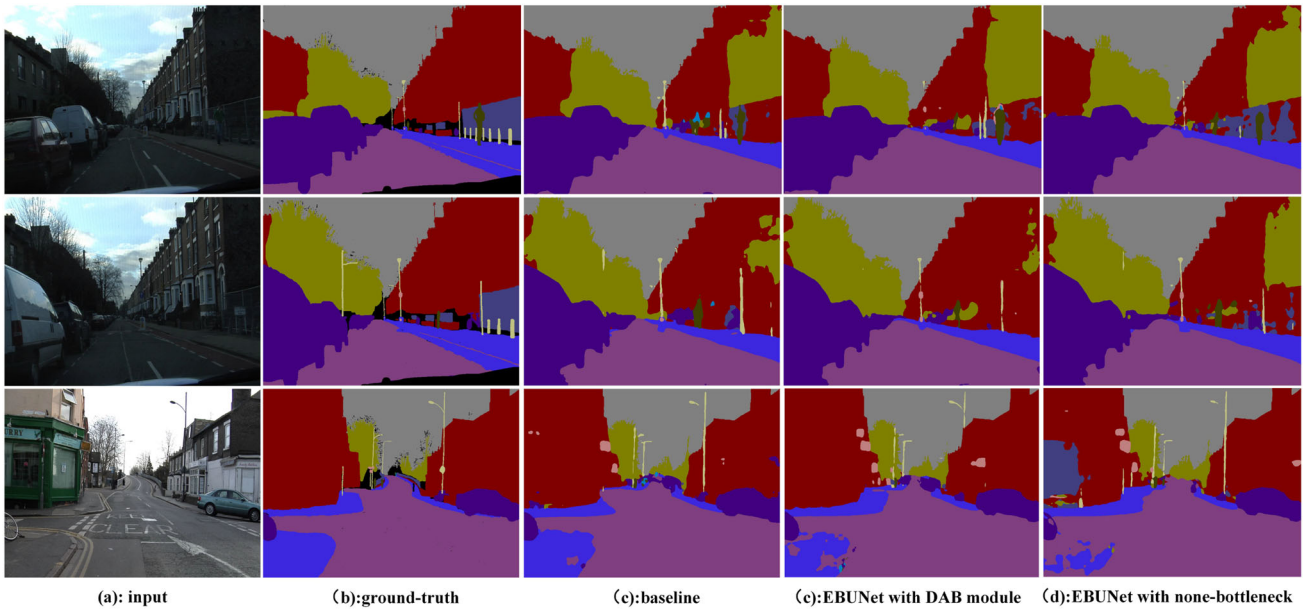

Fig. 6 Visual results about ablation study on EBU module. From the left column to the right column is: input, ground-truth, baseline, EBUNet with DABmodule, and EBUNet with non-bottleneck

Table 4 Experiment results on LAD

Method	Decoders		Parameters (M)↓	FLOPS (G) ↓	mIoU (%) ↑
	LAD	DABNet			
EBUNet	✓		1.572983	24.13	72.26
EBUNet		✓	1.566011	23.94	70.93

Table 5 Experiment results on the attention mechanism used in LAD

Method	LAD		Parameters (M)↓	Flops (G)↓	mIoU (%)↑
	SAM	CAM			
EBUNet	✓	✓	1.572983	24.13	72.26
EBUNet	✓		1.572980	24.13	71.95
EBUNet		✓	1.572965	24.13	71.76
EBUNet			1.572962	24.13	71.25

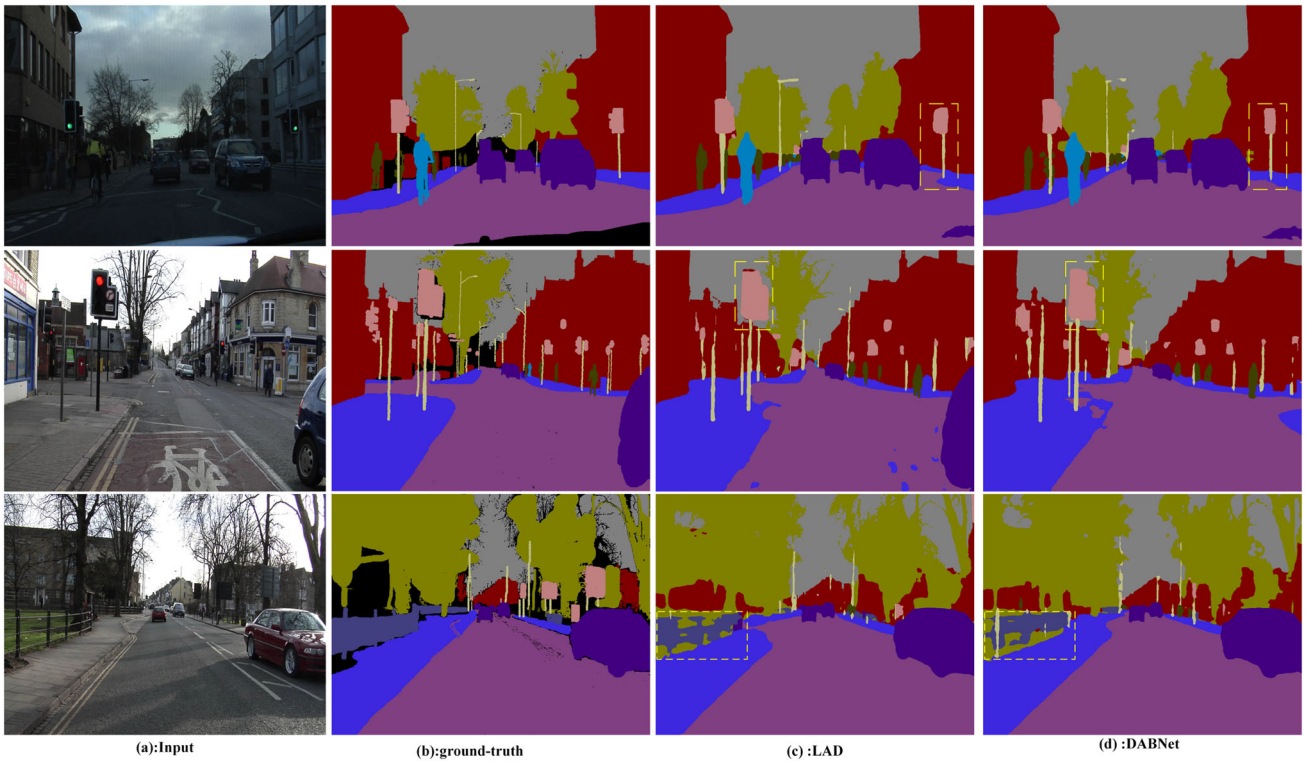

Fig. 7 Visual results about ablation study on EBU module. From the left column to right column is: input, ground-truth, baseline, and decoder in DABNet

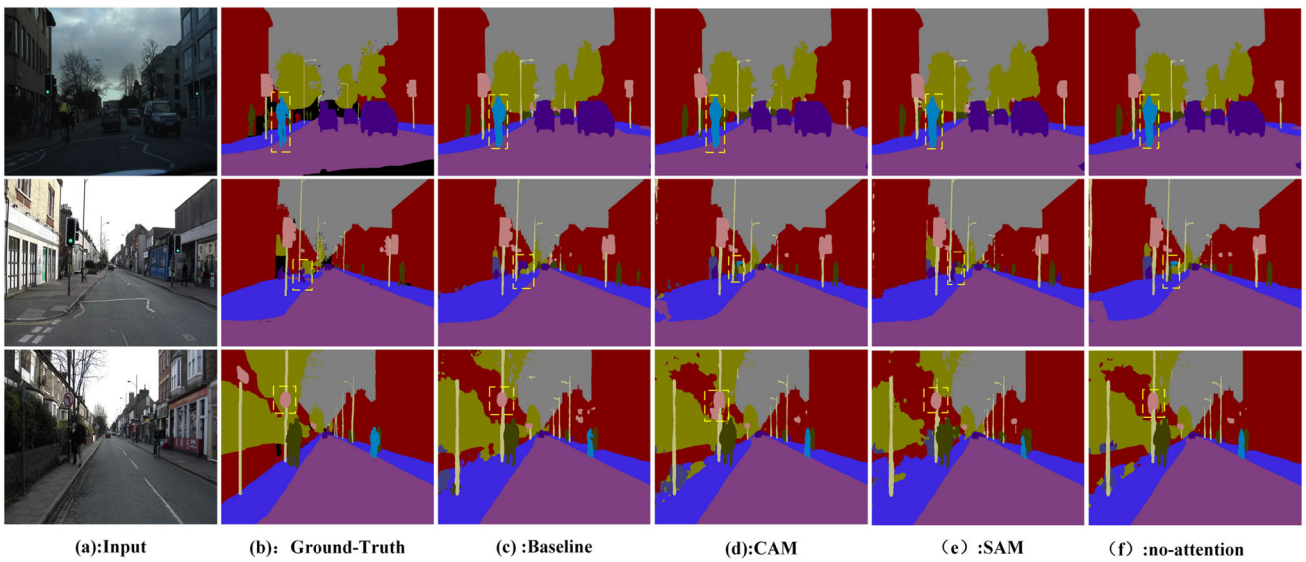

Fig. 8 Visual comparisons about LAD. From the left-most to right-most are **a** input **b** ground-truth **c** baseline of LAD **d** only use CAM in LAD **e** only use SAM in LAD **f** no attention used in LAD

As shown in 4, the accuracy of LAD increased by 1.33% when compared to the DABNet decoder, but there was only an increase in parameters of 0.01 M, meaning the cost is negligible.

A visual comparison of the ablation results for the LAD module is also performed. The visual results can be seen in Figs. 7 and 8. The difference is highlighted by the yellow dashed line.

From Table 5, LAD achieves the highest mIoU when both SAM and CAM are used to refine different-level feature maps. The accuracy performance (mIoU) of LAD is 0.5% lower when only CAM is used to refine high-level features.

When SAM is only used to refine low-level features, mIoU is also 0.3% lower than the baseline of LAD. SAM and CAM canceling in the LAD leads to a 1% reduction in accuracy. As a result, we can conclude that the attention mechanism has the potential to effectively improve segmentation accuracy while consuming negligible computation resources.

Ablation on the depth of EBU Block

There are two parameters M and N that indicate the number of EBU modules contained within EBU block 1 and EBU block 2. In order to investigate the model performance in terms of segmentation accuracy (mIoU) and feed forward speed (FPS), we devised a series of experiments using different values for M and N. The experiment settings and results are listed in Table 6.

According to Table 6, accuracy tends to get better as the depth inside the EBU blocks increases. However, accuracy can only be slightly improved if we increase the depth inside the EBU blocks. Even when we continue to deepen the depth of the EBU blocks, performance drops.

In general, increasing the depth of the network at the beginning will improve network performance to a certain degree, with a moderate increase in computational cost. However, when we make the network deeper, the accuracy and efficiency of the network fall instead.

Comparisons with other works

We compare the performance of our EBUNet with some other state-of-art semantic segmentation methods on the Cityscapes and CamVid datasets in this subsection. Similar to other lightweight semantic segmentation models, we perform down-sampling operations on the input images on Cityscapes. The resolution is decreased to 512×1024 (for Cityscapes). For the CamVid dataset, we use origin resolution 720×960 to perform our experiments. In addition, the speed of our EBUNet is measured on three different GPUs: RTX3090, RTX2080Ti, and TiTan XP.

Table 6 The ablation results on the influence of the depth

M	N	Parameters (M)	FPS	Miou (%)
3	1	0.49	198	62.91
3	6	1.09	153	71.61
3	8	1.33	135	71.27
3	10	1.57	147	72.26
3	12	1.81	120	71.84
4	8	1.36	129	71.71
4	12	1.84	111	71.89
8	12	1.96	90	72.15

Comparisons on cityscapes

A number of semantic segmentation methods are selected in recent years to demonstrate the effectiveness of our EBUNet, including ENet [22], ESPNet [38], CGNet [39], ContextNet [30], EDANet [40], ERFNet [37], Fast-SCNN [31], BiseNetV1 [23], ICNet [41], DABNet [13], LedNet [19], FBSNet [20], JPANet [26], MSCFNet [21], EDGENet [42], and FPANet [27]. Moreover, we also choose non-real-time semantic segmentation, including SegNet [43], DeepLabV2 [8], and RefineNet [7] to demonstrate the advance of our EBUNet.

As a means of providing a comprehensive comparison, we have counted the input size, the parameters, the computational complexity (FLOPS), the forward inference speed (FPS), the GPU platform, and the accuracy (mIoU) for each model. The quantitative result is shown in Table 7. Our EBUNet achieves 73.4% mIoU at a speed of 152 FPS on a single RTX3090 GPU card. A speed evaluation of EBUNet on both Titan XP and RTX2080Ti was also conducted and reported in Table 7.

As shown in Table 7, the performance of our EBUNet can even outperform certain non-real-time approaches. It is worth noting that the speed of our EBUNet is 98 frames per second, which is much faster than the speed of DeepLabV2 [8] with RTX 2080Ti. Moreover, the accuracy of EBUNet is 3% higher than that of DeepLabV2. When compared to RefineNet, although the proposed EBUNet achieves a slightly accuracy lower (0.3%) than it. However, our EBUNet produces a much smaller amount of parameters than RefineNet, approximately 20× fewer parameters than RefineNet.

It is found that the parameter of our EBUNet is in the same order of magnitude when compared to the lightweight and real-time semantic segmentation methods, but EBUNet achieves a certain improvement in mIoU. Compared with ESNet [46], the mIoU increased 2.7%, and the EBUNet has fewer parameters, which is more lightweight than ESNet. Compared to the MSCFNet [21], the parameter of our EBUNet only increased 0.42 M, while the mIoU increased 1.5%. Meanwhile, the FPS of EBUNet on Titan XP is 63, which is faster than MSCFNet. In comparison to the AGLNet, our parameters increased by 0.45 M, but the mIoU has increased by 2.1%. Moreover, we are able to reach 98 FPS on the same GPU with RTX2080Ti, which is faster than AGLNet (46 FPS faster). When compared to FPANet [27], our EBUNet achieves the same accuracy performance, but with faster speed. Moreover, the number of parameters in our EBUNet is only 1/10 of FANet's parameter.

The speed of our method has decreased somewhat to some extent when compared to fast semantic segmentation methods on RTX3090, including ContexNet [30], EDANet [45],

Table 7 Comparison with state-of-art semantic segmentation methods on Cityscapes test set

Method	Input size	Parameters (M)	FLOPS (G)	FPS	Platform	mIoU (%)
SegNet [43]	360× 640	29.5	286	53	RTX3090	56.1
ENet [22]	512×1024	**0.36**	4.4	100.2	RTX3090	58.3
ESPNet [38]	512×1024	**0.36**	**3.5**	**320**	RTX3090	60.3
CGNet [39]	1024×2048	0.49	28	110	RTX3090	64.8
ContextNet [30]	1024×2048	0.85	7.2	121	RTX3090	66.1
SQNet [44]	1024×2048	16.3	576.4	6.1	RTX3090	59.8
EDANet [45]	512×1024	0.68	9	189	RTX3090	67.3
ERFNet [37]	512×1024	2.1	26.9	62.7	RTX3090	68.0
Fast-SCNN [31]	1024×2048	1.1	7	–	RTX3090	68.0
BiseNetV1 [44]	768×1536	5.8	14.8	105	TitanXP	68.4
DABNet [13]	512×1024	0.76	27.7	191	RTX3090	70.1
DeeLabV2 [8]	512×1024	4	457	1	RTX2080Ti	70.4
ICNet [41]	1024×2048	26.5	28.3	15.4	RTX3090	70.6
LedNet [19]	512×1024	0.94	–	87	RTX3090	70.6
ESNet [46]	512×1024	1.66	24.4	53	RTX3090	70.7
FBSNet [20]	512×1024	0.62	9.7	24	RTX3090	70.9
EDGENet [42]	512×1024	–	–	36	Titan XP	71.0
AGLNet [47]	512×1024	1.12	13.88	52	TitanXP	71.3
JPANet [26]	512×1024	3.49	10.9	110	GTX1080Ti	71.6
FRRN [48]	512×1024	24.8	–	2.1	RTX2080Ti	71.8
MSCFNet [21]	512× 1024	1.15	–	50	TitanXP	71.9
FPANet [27]	512×1024	15.45	–	63	RTX2080Ti	73.4
RefineNet [7]	512×1024	118.1	428.3	9	RTX2080Ti	73.6
Ours	512×1024	1.57	24.13	152	RTX 3090	73.4
				98	RTX2080Ti	
				63	TitanXP	

The best performance are highlighted in bold

and DABNet [13], but the accuracy has improved significantly, which are 7.3%, 6.1%, and 3.3%, respectively.

Additionally, we compare the performance of the different semantic classes in the cityscapes test set. The comparison results are shown in Table 8. We can learn from Table 8 that our EBUNet is able to achieve state-of-the-art results in 12 out of 19 semantic classes without requiring any pre-training. In addition, EBUNet achieves significant improvements in the three categories of trucks, sidewalks, and riders, which are 7%, 1.8%, and 1% higher than the second place, respectively. A visual comparison is also presented on the Cityscapes validation set, which can be seen in Fig. 9.

According to the discussion above, our EBUNet achieves a good balance between segmentation accuracy and running efficiency on Cityscapes dataset.

Comparisons on CamVid

We also evaluate the performance of the proposed EBUNet on CamVid to further investigate its robustness, Table 9 reports the performance of our EBUNet and other methods (Fig. 10).

We can learn from Table 9 that our EBUNet achieves outstanding results. It achieves 72.2 mIoU at a speed of 147 FPS. A number of segmentation methods are selected and compared on a comprehensive basis: pre-training, FPS, parameters, and accuracy (mIoU). As shown in Table 9, among these methods, the proposed EBUNet achieves the best performance in terms of speed and accuracy. The EBUNet parameter has only increased 0.42 M compared to the AGLNet, but the accuracy has increased 2.8%. EBUNet achieves fast speed (27FPS faster) and higher accuracy (3.1 mIoU higher) in comparison to LMFFNet.

Conclusion

In this paper, we proposed an EBUNet for fast and accurate semantic segmentation tasks. Our EBUNet consists of three main components: EBU blocks, IPAM, and LAD. The EBU module adopted depth-wise convolution and depth-wise dilated convolution simultaneously to acquire much useful contextual information with a lower computation cost.

Table 8 The individual class accuracy performance on cityscapes test set

Methods	Roa	Sky	Car	Veg	Bui	Sid	Ped	Bus	Tsi	Bic
SegNet	96.4	91.8	89.3	87.0	84.0	73.2	62.8	43.1	45.1	51.9
Enet	96.3	90.6	90.6	88.6	75.0	74.2	65.5	50.5	44.0	55.4
ESPNet	97.0	92.6	92.3	90.8	76.2	77.5	67.0	52.5	46.3	57.2
CGNet	95.5	92.9	90.2	89.6	88.1	78.7	74.9	59.5	63.9	60.2
EDANet	97.8	93.6	92.4	91.4	895.5	80.6	75.7	58.7	65.0	64.0
ERFNet	97.7	94.2	92.8	91.4	89.8	81.0	76.8	60.1	65.3	61.7
ICNet	97.1	93.5	92.6	91.5	89.7	79.2	74.6	72.7	63.4	70.5
FSBNet	98.0	94.4	93.9	92.7	91.5	83.2	82.5	56.0	71.5	70.1
MSCFNet	97.7	94.3	94.1	92.3	91.0	82.8	82.7	66.1	71.4	70.2
Ours	**98.2**	**94.6**	**94.7**	92.5	**92.4**	**84.6**	82.3	71.6	**72.1**	**70.9**

Methods	Ter	TLi	Rid	Pol	Tra	Mot	Wal	Fen	Tru	mIoU (%)
SegNet	63.8	39.8	42.8	35.7	44.1	35.8	28.4	29.0	38.1	57.0
Enet	61.4	34.1	38.4	43.4	48.1	38.8	32.2	33.2	36.9	58.3
ESPNet	63.2	35.6	40.9	45.0	50.1	41.8	35.0	36.1	38.1	60.3
CGNet	67.6	59.8	54.9	54.1	25.2	47.3	40.0	43.0	44.1	64.8
EDANet	68.7	59.8	54.3	52.3	56.0	50.4	42.0	46.0	40.9	67.3
ERFNet	68.2	59.8	57.1	56.3	51.8	47.3	42.5	48.0	50.8	68.0
ICNet	68.3	60.4	56.1	61.5	51.3	53.6	43.2	48.9	51.3	69.5
FSBNet	70.5	67.6	63.8	62.5	37.6	56.2	50.9	53.5	50.5	70.9
MSCFNet	70.2	67.1	62.7	61.2	51.9	57.6	49.0	52.5	50.9	71.9
Ours	**70.9**	67.2	**64.8**	62.1	**56.4**	56.9	49.0	**54.9**	**57.9**	**73.4**

The best accuracy performance are highlighted in bold

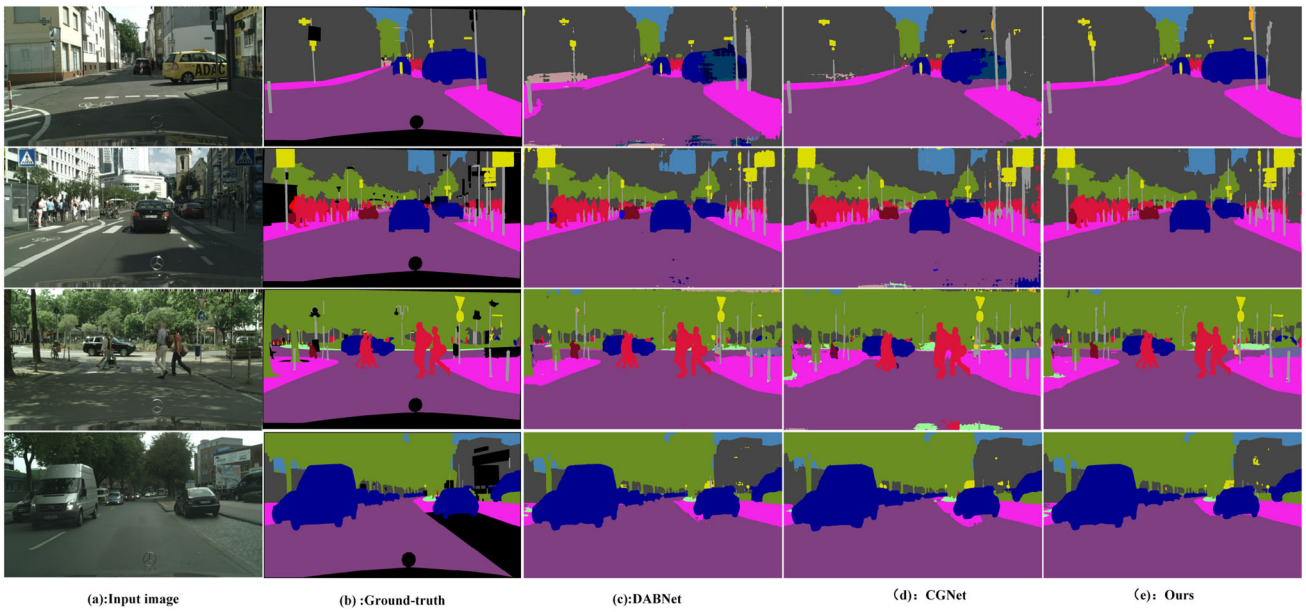

Fig. 9 Visual results on cityscapes. From lest column to right column is: input, ground-truth, DABNet, CGNet and our EBUNet

Table 9 Comparisons results on Camvid dataset with state-of-art works

Method	Pretrained	FPS	Platform	Parameter (M)	mIoU (%)
ENet	No	60	RTX3090	0.36	51.3
SegNet	ImageNet	68	RTX3090	29.5	55.6
DFANet	ImageNet	120	RTX3090	7.8	64.7
LMFFNet	No	120	RTX3090	1.4	69.1
AGLNet	No	90.1	RTX3090	1.1	69.4
Ours	No	**147**	RTX3090	1.57	**72.2**

The best performance in each dimension are highlighted in bold

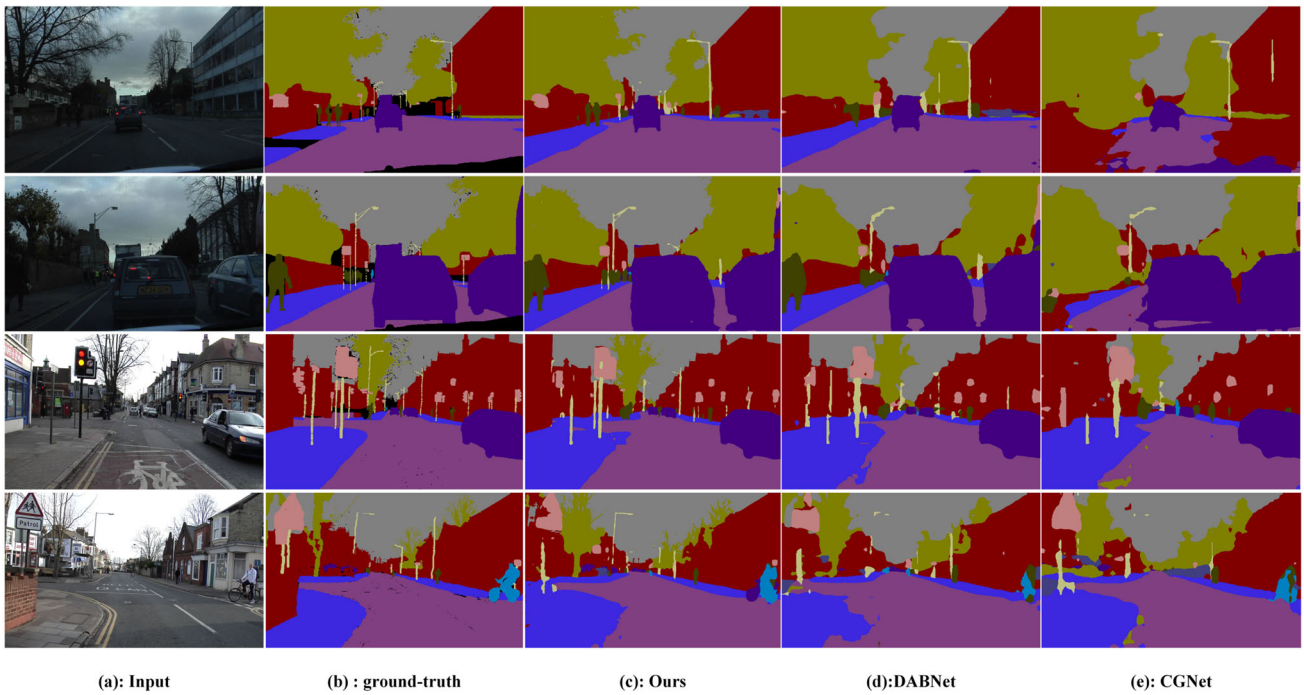

Fig. 10 Visual results with several lightweight semantic segmentation works on Camvid test set. From left-most to right-most are: input, ground-truth, our EBUNet, DABNet [13] and CGNet [39]

The IPAM module was mainly used for refining the feature maps, which promotes segmentation accuracy with negligible parameters and costs. The LAD module was employed to recover the spatial information to the origin resolution. We designed a series of ablation studies to demonstrate the effectiveness of the EBU module and LAD in our EBUNet. We also make extensive comprehensive comparisons with others methods on both Cityscapes and CamVid datasets. To be specific, our EBUNet achieved a 73.4% and 72.26% mIoU on the above datasets. The FPS on the two datasets is 152 and 147 on a NVIDIA RTX 3090 platform, which is a competitive result. In conclusion, our EBUNet strikes a better trade-off between segmentation accuracy and efficiency.

However, our EBUNet has the following problems when applied in practical applications: First of all, the memories of the real applications are usually limited. Although we have adopted some lightweight techniques to reduce the computational complexity, our EBUNet still needs amount of memory resources, which is a huge burden for the equipment of the real applications. Besides, the storage space limited is another problem when deploys the EBUNet in real applications. Even we have adopted depth-wise separable convolution to reduce the amount of parameter and computational complexity in our EBUNet, it also requires about 6MB space to storage, which is consumption for the storage resources. Therefore, we dedicate to explore a novel architecture for semantic segmentation to gain a better tradeoff among running efficiency, segmentation accuracy, and memory consumption in the future.

Author Contributions Introduction: SS and ZZ; methodology: SS, YY, and GY; software: SS and WD; validation: SS; writing—original draft preparation: SS and YY; writing—review and editing: SS, YY, and GY. All authors have read and agreed to the published version of the manuscript.

Funding Funding was provided by Scientific Research Plan Projects of Shaanxi Education Department (Grant no. 21JK0684).

Data availability statement Two public datasets are used in this research. Researchers can get the datasets from the following websites: [Cityscapes]: https://www.cityscapes-dataset.com/ [Camvid]:http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/.

References

1. Lianos K-N, Schonberger JL, Pollefeys M, Sattler T (2018) Vso: visual semantic odometry. In: Proceedings of the European conference on computer vision (ECCV), pp 234–250
2. Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Garcia-Rodriguez J (2017) A review on deep learning techniques applied to semantic segmentation. arXiv:1704.06857
3. Ess A, Müller T, Grabner H, Van Gool L (2009) Segmentation-based urban traffic scene understanding. In: BMVC, vol 1. Citeseer, p 2
4. Tao H, Cheng L, Qiu J, Stojanovic V (2022) Few shot cross equipment fault diagnosis method based on parameter optimization and feature mertic. Meas Sci Technol 33:115005
5. Djordjevic V, Stojanovic V, Tao H, Song X, He S, Gao W (2022) Data-driven control of hydraulic servo actuator based on adaptive dynamic programming. Discrete Contin Dyn Syst Ser S 15
6. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2881–2890
7. Lin G, Milan A, Shen C, Reid I (2017) Refinenet: multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1925–1934
8. Chen L-C, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2017) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 40:834–848
9. Xia M, Zhong Z, Chen D (2022) Structured pruning learns compact and accurate models. arXiv:2204.00408
10. Zhao B, Cui Q, Song R, Qiu Y, Liang J (2022) Decoupled knowledge distillation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11953–11962
11. Hou Y, Zhu X, Ma Y, Loy CC, Li Y (2022) Point-to-voxel knowledge distillation for lidar semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8479–8488
12. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
13. Li G, Yun I, Kim J, Kim J (2019) Dabnet: depth-wise asymmetric bottleneck for real-time semantic segmentation. arXiv:1907.11357
14. Shi Min, Shen Jialin, Yi Qingming, Weng Jian, Huang Zunkai, Luo Aiwen, Zhou Yicong (2022) LMFFNet: a well-balanced lightweight network for fast and accurate semantic segmentation. IEEE Trans Neural Netw Learn Syst 1–15. https://doi.org/10.1109/TNNLS.2022.3176493
15. Peng C, Zhang K, Ma Y, Ma J (2021) Cross fusion net: a fast semantic segmentation network for small-scale semantic information capturing in aerial scenes. IEEE Trans Geosci Remote Sens 60:1–13
16. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
17. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6848–6856
18. Ma N, Zhang X, Zheng H-T, Sun J (2018) Shufflenet v2: practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV), pp 116–131
19. Wang Y, Zhou Q, Liu J, Xiong J, Gao G, Wu X, Latecki LJ (2019) Lednet: a lightweight encoder-decoder network for real-time semantic segmentation. In: 2019 IEEE international conference on image processing (ICIP), IEEE, pp 1860–1864
20. Gao G, Xu G, Li J, Yu Y, Lu H, Yang J (2022) Fbsnet: a fast bilateral symmetrical network for real-time semantic segmentation. IEEE Trans Multim
21. Gao G, Xu G, Yu Y, Xie J, Yang J, Yue D (2021) Mscfnet: a lightweight network with multi-scale context fusion for real-time semantic segmentation. IEEE Trans Intell Transp Syst
22. Paszke A, Chaurasia A, Kim S, Culurciello E (2016) Enet: a deep neural network architecture for real-time semantic segmentation. arXiv:1606.02147
23. Yu C, Wang J, Peng C, Gao C, Yu G, Sang N (2018) Bisenet: bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 325–341
24. Fan M, Lai S, Huang J, Wei X, Chai Z, Luo J, Wei X (2021) Rethinking bisenet for real-time semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9716–9725
25. Yu C, Gao C, Wang J, Yu G, Shen C, Sang N (2021) Bisenet v2: bilateral network with guided aggregation for real-time semantic segmentation. Int J Comput Vis 129:3051–3068
26. Hu X, Jing L, Sehar U (2022) Joint pyramid attention network for real-time semantic segmentation of urban scenes. Appl Intell 52:580–594
27. Wu Y, Jiang J, Huang Z, Tian Y (2022) Fpanet: feature pyramid aggregation network for real-time semantic segmentation. Appl Intell 52:3319–3336
28. Liu J, Xu X, Shi Y, Deng C, Shi M (2022) Relaxnet: residual efficient learning and attention expected fusion network for real-time semantic segmentation. Neurocomputing 474:115–127
29. Tao H, Qiu J, Chen Y, Stojanovic V, Cheng L (2023) Unsupervised cross-domain rolling bearing fault diagnosis based on time-frequency information fusion. J Frankl Inst 360:1454–1477
30. Poudel RP, Bonde U, Liwicki S, Zach C (2018) Contextnet: exploring context and detail for semantic segmentation in real-time. arXiv:1805.04554
31. Poudel RP, Liwicki S, Cipolla R (2019) Fast-scnn: fast semantic segmentation network. arXiv:1902.04502
32. Li R, Zheng S, Zhang C, Duan C, Wang L, Atkinson PM (2021) Abcnet: attentive bilateral contextual network for efficient semantic segmentation of fine-resolution remotely sensed imagery. ISPRS J Photogramm Remote Sens 181:84–98

33. Li H, Xiong P, Fan H, Sun J (2019) Dfanet: deep feature aggregation for real-time semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9522–9531
34. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3213–3223
35. Brostow GJ, Shotton J, Fauqueur J, Cipolla R (2008) Segmentation and recognition using structure from motion point clouds. In: European conference on computer vision. Springer, pp 44–57
36. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. Springer, pp 177–186
37. Romera E, Alvarez JM, Bergasa LM, Arroyo R (2017) Erfnet: efficient residual factorized convnet for real-time semantic segmentation. IEEE Trans Intell Transport Syst 19:263–272
38. Mehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H (2018) Espnet: efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 552–568
39. Wu T, Tang S, Zhang R, Cao J, Zhang Y (2020) Cgnet: a light-weight context guided network for semantic segmentation. IEEE Trans Image Process 30:1169–1179
40. Yang C, Gao F (2019) Eda-net: dense aggregation of deep and shallow information achieves quantitative photoacoustic blood oxygenation imaging deep in human breast. In: International conference on medical image computing and computer-assisted intervention. Springer, pp 246–254
41. Zhao H, Qi X, Shen X, Shi J, Jia J (2018) Icnet for real-time semantic segmentation on high-resolution images. In: Proceedings of the European conference on computer vision (ECCV), pp 405–420
42. Han H-Y, Chen Y-C, Hsiao P-Y, Fu L-C (2020) Using channel-wise attention for deep cnn based real-time semantic segmentation with class-aware edge information. IEEE Trans Intell Transp Syst 22:1041–1051
43. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39:2481–2495
44. Treml M, Arjona-Medina J, Unterthiner T, Durgesh R, Friedmann F, Schuberth P, Mayr A, Heusel M, Hofmarcher M, Widrich M et al (2016) Speeding up semantic segmentation for autonomous driving
45. Lo S-Y, Hang H-M, Chan S-W, Lin J-J (2019) Efficient dense modules of asymmetric convolution for real-time semantic segmentation. In: Proceedings of the ACM multimedia Asia, pp 1–6
46. Lyu H, Fu H, Hu X, Liu L (2019) Esnet: edge-based segmentation network for real-time semantic segmentation in traffic scenes. In: 2019 IEEE international conference on image processing (ICIP), IEEE, pp 1855–1859
47. Zhou Quan, Wang Yu, Fan Yawen, Wu Xiaofu, Zhang Suofei, Kang Bin, Latecki Longin Jan (2020) AGLNet: towards real-time semantic segmentation of self-driving images via attention-guided lightweight network. Appl Soft Comput 96:106682. https://doi.org/10.1016/j.asoc.2020.106682
48. Pohlen T, Hermans A, Mathias M, Leibe B (2017) Full-resolution residual networks for semantic segmentation in street scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4151–4160