International Journal of Computer Vision (2021) 129:3051–3068
https://doi.org/10.1007/s11263-021-01515-2

BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation

Changqian Yu[1,2] · Changxin Gao[1] · Jingbo Wang[3] · Gang Yu[4] · Chunhua Shen[2] · Nong Sang[1]

Received: 5 April 2020 / Accepted: 9 August 2021 / Published online: 3 September 2021

Abstract

Low-level details and high-level semantics are both essential to the semantic segmentation task. However, to speed up the model inference, current approaches almost always sacrifice the low-level details, leading to a considerable decrease in accuracy. We propose to treat these spatial details and categorical semantics separately to achieve high accuracy and high efficiency for real-time semantic segmentation. For this purpose, we propose an efficient and effective architecture with a good trade-off between speed and accuracy, termed Bilateral Segmentation Network (BiSeNet V2). This architecture involves the following: (i) A detail branch, with wide channels and shallow layers to capture low-level details and generate high-resolution feature representation; (ii) A semantics branch, with narrow channels and deep layers to obtain high-level semantic context. The detail branch has wide channel dimensions and shallow layers, while the semantics branch has narrow channel dimensions and deep layers. Due to the reduction in the channel capacity and the use of a fast-downsampling strategy, the semantics branch is lightweight and can be implemented by any efficient model. We design a guided aggregation layer to enhance mutual connections and fuse both types of feature representation. Moreover, a booster training strategy is designed to improve the segmentation performance without any extra inference cost. Extensive quantitative and qualitative evaluations demonstrate that the proposed architecture shows favorable performance compared to several *state-of-the-art* real-time semantic segmentation approaches. Specifically, for a 2048×1024 input, we achieve 72.6% Mean IoU on the Cityscapes test set with a speed of 156 FPS on one NVIDIA GeForce GTX 1080 Ti card, which is significantly faster than existing methods, yet we achieve better segmentation accuracy. The code and trained models are available online at https://git.io/BiSeNet.

Keywords Semantic segmentation · Real-time processing · Bilateral network · Deep learning

1 Introduction

Semantic segmentation is the task of assigning semantic labels to each pixel. It is a fundamental problem in computer vision that has extensive applications, including scene understanding (Zhou et al. 2019), autonomous driving (Cordts et al. 2016; Geiger et al. 2012), human-machine interaction and video surveillance, to name just a few. In recent years, with the advance of convolutional neural networks (Krizhevsky et al. 2012), a series of semantic segmentation methods (Zhao et al. 2017; Chen et al. 2017; Yu et al. 2018b; Chen et al. 2018; Zhang et al. 2018a) based on fully convolutional network (FCN) (Long et al. 2015) have constantly advanced the state-of-the-art performance.

The high accuracy of these methods depends on their backbone networks. Two architectures are mainly used as the backbone networks: (i) *Dilation backbone*, which removes the downsampling operations and upsamples the correspond-

Communicated by Laurent Najman.

✉ Changxin Gao
cgao@hust.edu.cn

Changqian Yu
changqian_yu@hust.edu.cn

Chunhua Shen
chunhua.shen@adelaide.edu.au

Nong Sang
nsang@hust.edu.cn

[1] National Key Laboratory of Science and Technology on Multispectral Information Processing, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China

[2] The University of Adelaide, Adelaide, Australia

[3] The Chinese University of Hong Kong, Shatin, China

[4] Tencent, Shanghai, China

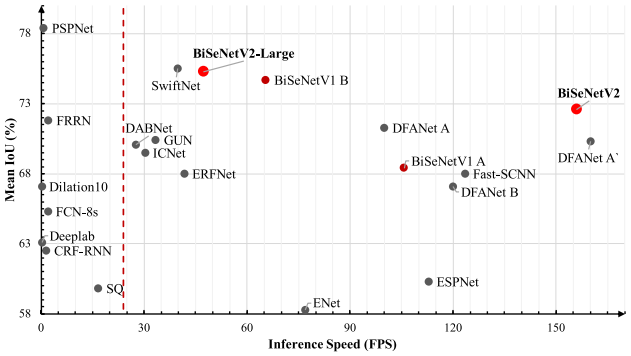
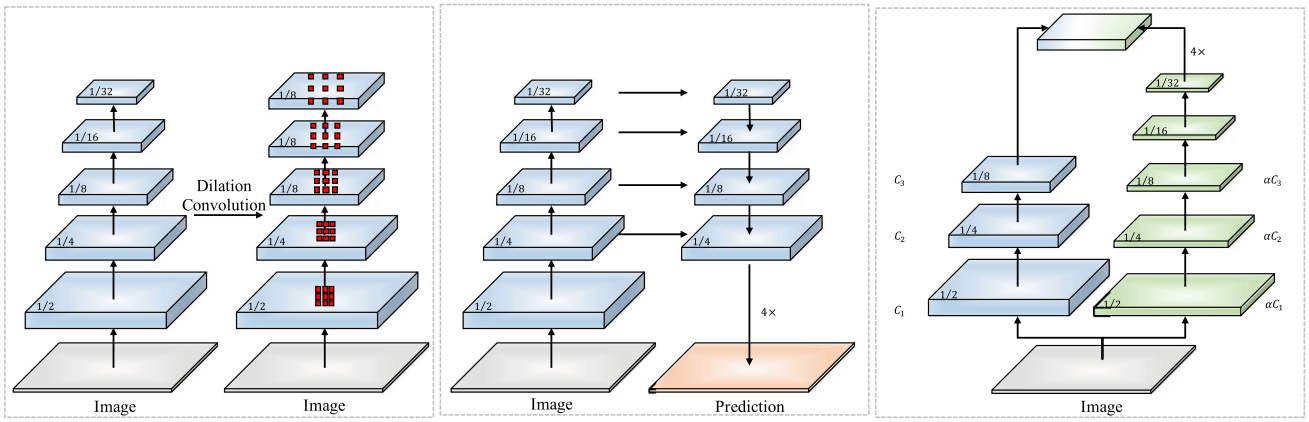


Fig. 1 *Speed-accuracy trade-off comparison on the Cityscapes test* set Red dots indicate our methods, while gray dots means other methods. The red line represents the real-time speed (Color figure online)

ing filter kernels to maintain high-resolution feature representation (Chen et al. 2015; Chen et al. 2018; Chen et al. 2018; Zhao et al. 2017, 2018b; Fu et al. 2019; Yu et al. 2020b), as shown in Fig. 2a, and (ii) *Encoder-decoder backbone*, with top-down and skip connections to recover the high-resolution feature representation in the decoder part (Lin et al. 2017; Peng et al. 2017; Yu et al. 2018b), as illustrated in Fig. 2b. However, both architectures are designed for general semantic segmentation tasks with little consideration of the inference speed and computational cost. In the dilation backbone, the dilation convolution is time-consuming and the removal of the down-sampling operation gives rise to high computational complexity and memory footprint. The numerous connections in the encoder-decoder architecture are unfavorable for obtaining low memory access cost (Ma et al. 2018). However, real-time semantic segmentation applications require efficient inference with high speed.

To meet this demand, based on both backbone networks, existing methods (Badrinarayanan et al. 2017; Paszke et al. 2016; Zhao et al. 2018a; Romera et al. 2018; Mazzini 2018) mainly employ two approaches to accelerate the model: (i) *Input Restricting*. Smaller input resolution results in lower computational cost with the same network architecture. To achieve real-time inference speed, many algorithms (Zhao et al. 2018a; Romera et al. 2018; Mazzini 2018; Romera et al. 2018) attempt to restrict the input size to reduce the overall computational complexity; (ii) *Channel Pruning*. This is a straight-forward acceleration method, particularly for the pruning of channels in the early stages to boost inference speed (Badrinarayanan et al. 2017; Paszke et al. 2016; Chollet 2017). Despite the improvements on the inference speed, both approaches sacrifice the low-level details and spatial capacity leading to a dramatic deterioration in accuracy. Therefore, the simultaneous achievement of high efficiency and high accuracy is challenging and of great importance for exploiting a specific architecture for the real-time semantic segmentation task.

Both the low-level details and high-level semantics are vital for the semantic segmentation task. In the architectures for general semantic segmentation task, the deep and wide networks encode both types of information simultaneously. However, we observe that both types of information inherently require different architectures, i.e., different width and depth. Low-level details, usually have high-resolution feature maps and exist in the low stages of the architec-



(a) Dilation Backbone **(b)** Encoder-Decoder Backbone **(c)** Bilateral Segmentation Backbone

Fig. 2 *Illustration of different backbone architectures.* **a** is the dilation backbone network, which removes the downsampling operations and upsamples the corresponding convolution filters. It has high computational complexity and memory footprint. **b** is the encoder-decoder backbone network, which adds extra top-down and lateral connections to recover the high-resolution feature map. These connections in the network are less favorable with respect to the memory access cost. To achieve high accuracy and high efficiency simultaneously, we design the **c** bilateral segmentation backbone network. This architecture has two pathways, a detail branch for spatial details and a semantics branch for categorical semantics. The detail branch has wide channels and shallow layers, while the semantics branch has narrow channels and deep layers, that can be made very lightweight by the factor (λ, e.g., $1/4$)

ture, and encode many diverse patterns, thus requiring wide channel dimension. High-level semantics, usually have low-resolution feature maps and exist in the high stages of the architecture, and need deep layers to abstract information. This motivates us to treat spatial details and categorical semantics separately to achieve the trade-off between the accuracy and inference speed for the real-time semantic segmentation task.

To achieve this goal, we propose a two-pathway architecture, which we call the **bi**lateral **se**gmentation **net**work (BiSeNet V2) for real-time semantic segmentation. One pathway is designed to capture the spatial details with wide channel dimensions and shallow layers, i.e., larger branch width and smaller branch depth, and is called the *detail branch*. In contrast, the other pathway is introduced to extract the categorical semantics with narrow channel dimensions and deep layers, i.e., smaller branch width and larger branch depth, and is named the *semantics branch*. The semantics branch simply requires a large receptive field to capture semantic context, while the detailed information can be supplied by the detail branch. Therefore, the semantics branch can be made very lightweight with fewer channel dimension and a fast-downsampling strategy. Both types of features are merged to construct a stronger and more comprehensive feature representation. This conceptual design leads to an efficient and effective architecture for real-time semantic segmentation, as illustrated in Fig. 2c.

Both branches employ different depths to encode the information, leading to the semantic gap between both features. In this study, we design a *guided aggregation layer* to merge the two types of features effectively. BiSeNetV1 (Yu et al. 2018a) adopts lateral connections to further enhance the features of the semantics branch iteratively. However, lateral connections are unfavorable for obtaining low memory access cost. Thus, following the philosophy of deep supervision (Xie and Tu 2015), we present a *booster* training strategy with a series of auxiliary prediction heads. It enhances the feature representation capability stage-by-stage in the training phase, and is discarded in the inference phase without increasing any inference complexity. Extensive quantitative and qualitative evaluations demonstrate that the proposed architecture shows favorable performance compared to *state-of-the-art* real-time semantic segmentation methods, as shown in Fig. 1.

The main contributions of this work are summarized as follows:

- We propose an efficient and effective two-pathway architecture, termed the bilateral segmentation network, for real-time semantic segmentation, which treats the spatial details and categorical semantics separately.
- For the semantics branch, we design a new lightweight structure based on depthwise convolutions to enhance the receptive field and capture rich contextual information.
- A booster training strategy is introduced to further improve the segmentation performance without increasing the inference cost.
- Our architecture achieves impressive results on the Cityscapes (Cordts et al. 2016), CamVid (Brostow et al. 2008a), COCO-Stuff (Caesar et al. 2018), and ADE20K (Zhou et al. 2019) benchmarks. Specifically, we obtain the results of 72.6% mean IoU on the Cityscapes *test* set with the speed of 156 FPS on one NVIDIA GeForce GTX 1080Ti card.

A preliminary version of this work was published in (Yu et al. 2018a). We have extended our conference version as follows. (i) We simplify the original structure to present an efficient and effective architecture for real-time semantic segmentation. We remove the time-consuming cross-layer connections in the original version to obtain a more clear and simpler architecture. (ii) We re-design the overall architecture with more compact network structures and well-designed components. Specifically, we deepen the detail branch to encode more details. We design lightweight components based on the depthwise convolutions for the semantics branch. Meanwhile, we propose an efficient aggregation layer to enhance the mutual connections between both paths. (iii) We conduct comprehensive ablation experiments to elaborate on the effectiveness and efficiency of the proposed method. (iv) We have significantly improved the accuracy and speed of the method in our previous work; i.e., for a 2048×1024 input, achieving 72.6% Mean IoU on the Cityscapes *test* set with a speed of 156 FPS on one NVIDIA GeForce GTX 1080Ti card.

2 Related Work

Recent years have witnessed significant advances in image semantic segmentation. In this section, our discussion mainly focuses on three groups of methods most relevant to our work, namely, generic semantic segmentation methods, real-time semantic segmentation methods, and lightweight architectures.

2.1 Generic Semantic Segmentation

Traditional segmentation methods based on the threshold selection (Otsu 1979), region growing (Vincent and Soille 1991), super-pixel (Ren and Malik 2003; Achanta et al. 2012; Van den Bergh et al. 2012) and graph (Boykov and Jolly 2001; Rother et al. 2004) algorithms adopt hand-crafted features to solve this problem. Recently, a new generation of algorithms based on FCN (Long et al. 2015; Shelhamer et al. 2017) have continuously improved the *state-of-the-art* performance on different benchmarks. Various methods are based on two

types of backbone network: the (i) *Dilation backbone network*, and the (ii) *Encoder-decoder backbone network*.

On the one hand, the dilation backbone removes the downsampling operations and upsamples the convolution filter to preserve high-resolution feature representations. Due to the simplicity of the dilation convolution, various methods (Chen et al. 2018; Chen et al. 2018; Zhao et al. 2017; Wang et al. 2018a; Zhang et al. 2018a; Yu et al. 2020b) develop different novel and effective components on it. Deeplabv3 (Chen et al. 2017) devises an atrous spatial pyramid pooling to capture multi-scale context, while PSPNet (Zhao et al. 2017) adopts a pyramid pooling module on the dilation backbone. Meanwhile, some methods introduce the attention mechanisms, e.g., self-attention (Yuan and Wang, 2018; Fu et al., 2019; Yu et al., 2020b), spatial attention (Zhao et al. 2018b) and channel attention (Zhang et al. 2018a), to capture long-range context based on the dilation backbone.

On the other hand, the encoder-decoder backbone network adds extra top-down and lateral connections to recover the high-resolution feature maps in the decoder part. FCN and Hypercolumns (Hariharan et al. 2015) adopt the skip connection to integrate the low-level features. Meanwhile, U-net (Ronneberger et al. 2015), SegNet with saved pooling indices (Badrinarayanan et al. 2017), RefineNet with multi-path refinement (Lin et al. 2017), LRR with stepwise reconstruction (Ghiasi and Fowlkes 2016), GCN with "large kernel" convolution (Peng et al. 2017) and DFN with channel attention module (Yu et al. 2018b) incorporate this backbone network to recover the detailed information. HRNet (Wang et al. 2020) and Lite-HRNet (Yu et al. 2021) adopts multi-branches to maintain high resolution.

Both types of backbone network encode the low-level details and high-level semantics simultaneously with the wide and deep networks. Although both types of backbone networks achieve state-of-the-art performance, most methods show a slow inference speed. In this study, we propose a novel and efficient architecture to treat the spatial details and categorical semantics separately to achieve a favorable trade-off between segmentation accuracy and inference speed.

2.2 Real-time Semantic Segmentation

Real-time semantic segmentation algorithms attract increasing attention when increasingly demanding practical applications require fast interaction and response. SegNet (Badrinarayanan et al. 2017) uses a compact network structure and the skip connection to achieve high speed. E-Net (Paszke et al. 2016) devises a lightweight network and delivers a very high inference speed. DLC (Li et al. 2017) employs a cascade network structure to reduce the computation in "easy regions". ERFNet (Romera et al. 2018) adopts the residual connection and factorized convolutions to retain efficiency and accuracy. Meanwhile, ESPNet (Mehta et al. 2018, 2019) devises an efficient spatial pyramid dilated convolution for real-time semantic segmentation. DFANet (Li et al. 2019b) reuses the feature to enhance the feature representation and reduces the complexity.

Although these methods can achieve a real-time inference speed, they often considerably sacrifice accuracy for efficiency, largely due to the loss of the low-level details. In this work, we take both of the low-level details and high-level semantics into consideration to achieve high accuracy and high efficiency.

Two/multi-branch architectures. This work is also related to other two/multi-branch architectures in the real-time semantic segmentation task. ICNet (Zhao et al. 2018a) is the first multi-branch architecture for the real-time semantic segmentation. It employs tree branches with different depths to process different resolution inputs. Low- and medium-resolution branches share the weights partially. GUN (Mazzini 2018) and ContextNet (Poudel et al. 2018) share the similar structure of the ICNet. They have only two branches for two resolution inputs. Fast-SCNN (Poudel et al. 2019) follows the architecture philosophy of BiSeNetV1 (Yu et al. 2018a). It differs from BiSeNetV1 in that it learns to down-sample the input, and then processes the information with two branches.

Different from these works, our study observes that (i) rich image information is essential for the segmentation accuracy, and (ii) spatial details and categorical semantics require different encoding structures. Thus, the proposed architecture maintains rich image information, and employs different structures for the spatial details and categorical semantics, respectively.

2.3 Lightweight Architecture

Following the pioneering work of group/depthwise convolution and separable convolution, lightweight architecture design has achieved rapid development, including Xception (Chollet 2017), MobileNet (Howard et al. 2017; Sandler et al. 2018), and ShuffleNet (Zhang et al. 2018b; Ma et al. 2018), to name a few. These methods achieve a good trade-off between speed and accuracy for the classification task. In this study, we design a lightweight network given the computation complexity, memory access cost and real inference speed for the real-time semantic segmentation.

3 Bilateral Segmentation Network

The concept of our BiSeNet is general and can be implemented with different convolutional models (He et al. 2016; Huang et al. 2017; Chollet 2017; Iandola et al. 2016; Howard et al. 2017; Sandler et al. 2018; Zhang et al. 2018b; Ma et al. 2018) and any specific designs. There are mainly three key

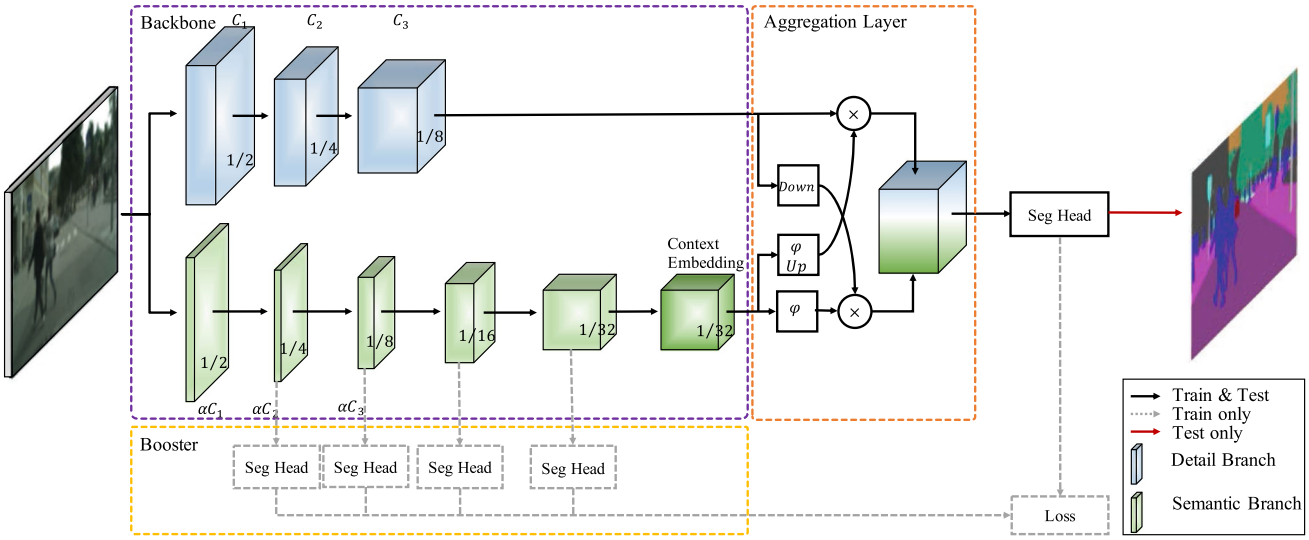


Fig. 3 *Overview of the bilateral segmentation network.* The network has three main components: the two-pathway backbone in the purple dashed box, the aggregation layer in the orange dashed box, and the booster component in the yellow dashed box. The two-pathway backbone contains a detail branch (the blue cubes) and a semantics branch (the green cubes). The three stages in the detail branch have C_1, C_2, C_3 channels, respectively. The channels of the corresponding stages in the semantics branch can be made lightweight by the factor $\lambda(\lambda < 1)$. The last stage of the semantics branch is the output of the context embedding block. Meanwhile, the numbers in the cubes show the ratios of the feature map to the resolution of the input. In the aggregation layer component, we adopt the bilateral aggregation layer. *Down* indicates the downsampling operation, Up represents the upsampling operation, φ is the sigmoid function, and $\otimes$ means an elementwise product. Moreover, in the booster component, we design some auxiliary segmentation heads to improve the segmentation performance without any extra inference cost (Color figure online)

concepts: **(i)** The detail branch has *wide* channel dimensions and *shallow* layers with *small* receptive field for the spatial details; **(ii)** The semantics branch has *narrow* channel dimensions and *deep* layers with *large* receptive field for the categorical semantics. **(iii)** An efficient aggregation layer is designed to fuse both types of representation.

In this section, we demonstrate the concept and the corresponding instantiations of the overall architecture and some other specific designs, as shown in Fig. 3.

3.1 Detail Branch

Concept. The detail branch is responsible for the low-level spatial details. The detail information has rich patterns, requiring a high channel capacity, i.e., wide channel dimensions. Since the low-level detail information usually exists in the low stages of the architecture, we can design a shallow structure with fewer branch layers, having a *small* stride for this branch. Overall, the key concept of the detail branch is to use *wide* channel dimensions and *shallow* layers, i.e., larger branch width and smaller branch depth, for the spatial details. Due to the high-resolution feature maps and wide channel dimensions of this branch, with consideration of the memory access cost, it is better to follow the architecture of VGGNet (Simonyan and Zisserman 2015) without any residual connections.

Instantiation. As shown in Table 1, the detail branch has tree stages, each layer of which is a convolution layer followed by batch normalization (Ioffe and Szegedy 2015) and activation function (Glorot et al. 2011). The first layer of each stage has a stride $s = 2$, while the other layers in the same stage have the same number of filters and output feature map size. Therefore, this branch extracts the output feature maps that are $1/8$ of the original input. This detail branch encodes rich spatial details due to the high channel capacity.

3.2 Semantics Branch

Concept. In parallel to the detail branch, the semantics branch is designed to capture high-level semantics. This branch has *narrow* channel dimensions, since the spatial details can be provided by the detail branch. High-level semantics require contextual dependencies and *large* receptive field. Therefore, (i) We adopt a fast-downsampling strategy to promote the level of the feature representation and enlarge the receptive field quickly; (ii) We employ the global average pooling (Liu et al. 2016) to embed the global contextual response. Due to the fast-downsampling strategy and narrow channel dimension, the semantics branch is lightweight and can be implemented by any efficient model(e.g., (Chollet 2017; Iandola et al. 2016; Howard et al. 2017; Sandler et al. 2018; Zhang et al. 2018b; Ma et al. 2018)).

Table 1 *Instantiation of the detail branch and the semantics branch.* Each stage S contains one or more operations *opr* (e.g., *Conv2d*, *Stem*, *GE*, *CE*). Each operation has a kernels size k, stride s and output channels c, repeated r times. The expansion factor e is applied to expand the channel number of the operation. Here the channel ratio is $\lambda = 1/4$. The green colors mark *fewer* channels of the semantics branch in the corresponding stage of the detail branch. Notation: *Conv2d* means the convolutional layer, followed by one batch normalization layer and relu activation function. *Stem* indicates the stem block. *GE* represents the gather-and-expansion layer. *CE* is the context embedding block

Stage	Output Size	Detail Branch						Semantics Branch						
		opr	*k*	*c*	*s*	*r*	FLOPs	*opr*	*k*	*c*	*e*	*s*	*r*	FLOPs
Input	512×1024													
S_1	256×512	Conv2d	3	64	2	1	5.1G	Stem	3	16	–	4	1	145.2M
	256×512	Conv2d	3	64	1	1								
S_2	128×256	Conv2d	3	64	2	1	3.6G							
	128×256	Conv2d	3	64	1	2								
S_3	64×128	Conv2d	3	128	2	1	3.0G	GE	3	32	6	2	1	1.0G
	64×128	Conv2d	3	128	1	2		GE	3	32	6	1	1	
S_4	32×64							GE	3	64	6	2	1	1.0G
	32×64							GE	3	64	6	1	1	
S_5	16×32							GE	3	128	6	2	1	2.1G
	16×32							GE	3	128	6	1	3	
	16×32							CE	3	128	-	1	1	

Instantiation. In consideration of the large receptive field and efficient computation simultaneously, we design the semantics branch, inspired by the designs from the lightweight model, e.g., Xception (Chollet 2017), MobileNet (Howard et al. 2017; Sandler et al. 2018; Howard et al. 2019), ShuffleNet (Zhang et al. 2018b; Ma et al. 2018). The semantics branch has a ratio of $\lambda(\lambda < 1)$ channels of the detail branch to keep the narrow channel dimensions. Some of the key features of the semantics branch are as follows.

Stem Block. Inspired by (Szegedy et al. 2017; Shen et al. 2017; Wang et al. 2018b), we adopt the stem block as the first stage of the semantics branch, as illustrated in Fig. 4. It uses two different downsampling manners to shrink the feature representation. The output feature of both branches are concatenated as the output. This structure has efficient computation cost and effective feature expression ability.

Context Embedding Block. As discussed in Sect. 3.2, the semantics branch requires large receptive field to capture high-level semantics. Inspired from (Yu et al. 2018b; Liu et al. 2016; Zhao et al. 2017; Chen et al. 2017), we design the context embedding block. This block uses the global average pooling and residual connection (He et al. 2016) to embed the global contextual information efficiently, as shown in Fig. 4.

Gather-and-Expansion Layer. Taking advantage of the benefit of depthwise convolution, we propose the gather-and-expansion Layer, as illustrated in Fig. 5. The gather-and-expansion Layer consists of: (i) A 3×3 convolution to efficiently aggregate feature responses and expand to a higher-dimensional space; (ii) A 3×3 depthwise convolution performed independently over each individual output channel of the expansion layer; (iii) A 1×1 convolution as the projection layer to project the output of depthwise convolution into a low channel capacity space. Recent lightweight works (Tan et al. 2019; Howard et al. 2019) adopt 5×5 separable convolution heavily to enlarge the receptive field.

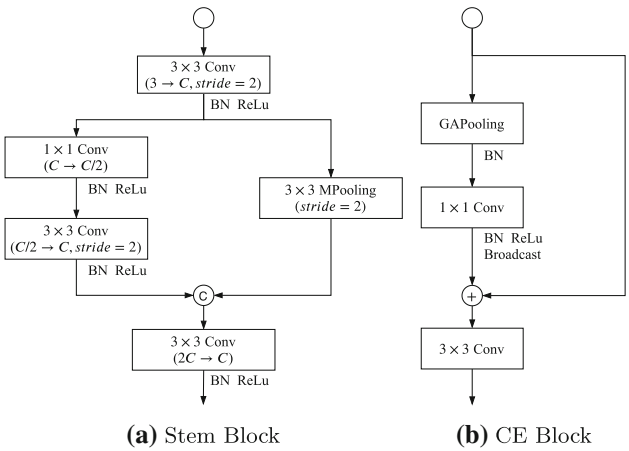


Fig. 4 *Illustration of the stem block and context embedding block.* **a** is the stem block, which adopts a fast-downsampling strategy. This block has two branches with different manners to downsample the feature representation. Then, both feature responses of the two branches are concatenated as the output. **b** is the context embedding block. As demonstrated in Sect. 3.2, the semantics branch requires large receptive field. Therefore, we design a context embedding block with the global average pooling to embed the global contextual information. Notation: *Conv* is convolutional operation. *BN* is the batch normalization. *ReLu* is the ReLu activation function. *Mpooling* is the max pooling. *GPooling* is the global average pooling. and C means concatenation. Meanwhile, 1×1, 3×3 denote the kernel size, and $H \times W \times C$ means the tensor shape (height, width, channel dimension)

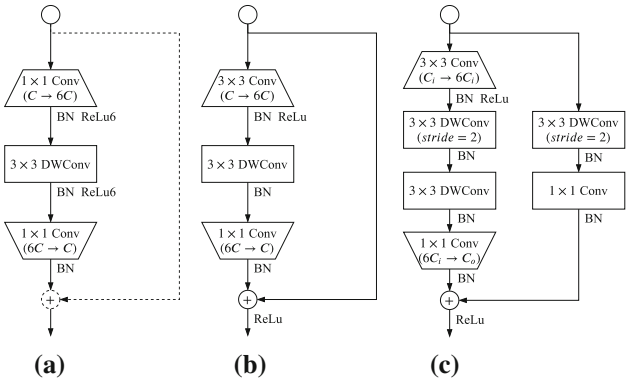


Fig. 5 *Illustration of the inverted bottleneck and gather-and-expansion layer*. **a** is the mobile inverted bottleneck Conv proposed in MobileNetv2. The dashed shortcut path and summation circle do not exist with the $stride = 2$. **bc** are the proposed gather-and-expansion layer. The bottleneck structure adopts: (i) A 3×3 convolution to gather local feature response and expand to higher-dimensional space;(ii) A 3×3 depthwise convolution performed independently over each individual output channel of the expansion layer; (iii) A 1×1 convolution as the projection layer to project the output of depthwise convolution into a low channel capacity space. When the $stride = 2$, we adopt two $kernel\_size = 3$ depthwise convolutions on the main path and a 3×3 separable convolution as the shortcut. Notation: *Conv* is the convolutional operation. *BN* is the batch normalization. *ReLu* is the ReLu activation function. Meanwhile, $1 \times 1, 3 \times 3$ denote the kernel size, and $H \times W \times C$ represents the tensor shape (height, width, channel dimension)

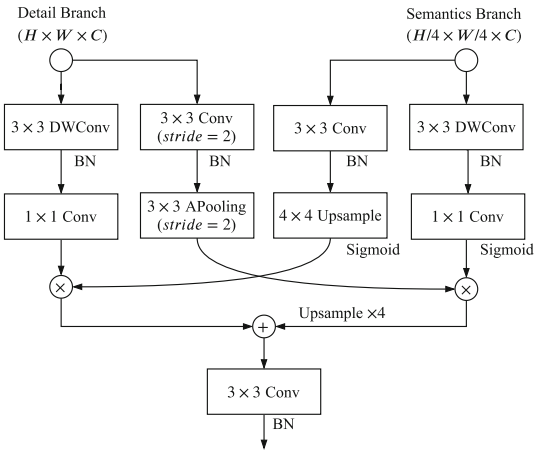


Fig. 6 *Detailed design of the bilateral guided aggregation layer*. Notation: *Conv* is the convolutional operation. *DWConv* is the depthwise convolution. *APooling* is the average pooling. *BN* denotes the batch normalization. *Upsample* means bilinear interpolation. *Sigmoid* is the sigmoid activation function. *Sum* means summation. Meanwhile, $1 \times 1, 3 \times 3$ denote the kernel size, $H \times W \times C$ represents the tensor shape (height, width, channel dimension), and $\otimes$ represents the elementwise product

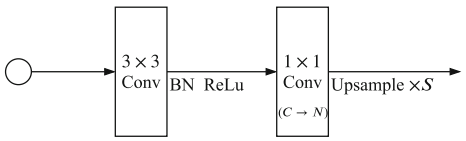


Fig. 7 *Detailed design of the segmentation head in Booster*. Notation: *Conv* is the convolutional operation. *BN* denotes the batch normalization. *Upsample* means bilinear interpolation. Meanwhile, $1 \times 1, 3 \times 3$ denote the kernel size, $H \times W \times C$ means the tensor shape (height, width, channel dimension), C represents the channel dimension, S denotes the scale ratio of upsampling, and N is the final output dimension

In this layer, when $stride = 2$, we adopt two 3×3 depthwise convolution to replace the 5×5 depthwise convolution enlarging the receptive field with fewer FLOPs.

In contrast to the inverted bottleneck in MobileNetv2, the GE layer has one more 3×3 convolution. This is because the 3×3 convolution is specially optimized in the CUDNN library (Chetlur et al. 2014). As also observed in ShuffleNetV2 (Ma et al. 2018), the 3×3 convolution is not much slower than the 1×1 convolution. The 3×3 convolution has larger receptive field, beneficial to the semantics branch. Thus, it is a good choice for the trade-off between segmentation accuracy and efficiency.

3.3 Bilateral Guided Aggregation

Concept. Both branches have different depths and widths, leading to a semantic gap between both feature representations. The downsampling strategies of both branches are also different, thus giving rise to the resolution gap. Thus, an aggregation layer is required to compensate the semantic and resolution gaps to merge both types of feature representation.

Instantiation. Simple combination, i.e., elementwise summation and concatenation, ignores the diversity of both types of information, leading to worse performance and difficult optimization.

Based on the observations, we propose the bilateral guided aggregation layer to fuse the complementary information from both branches, as illustrated in Fig. 6. This layer employs the contextual information of the semantics branch to guide the feature response of the detail branch. Using different scale guidance, we can capture different scale feature representation, inherently encoding the multi-scale information. Meanwhile, this guidance manner enables efficient communication between both branches compared to the simple combination.

3.4 Booster Training Strategy

Stagewisely enhancing the semantics is beneficial to the semantics branch. However, lateral connections, used by BiSeNetV1, are unfavorable for obtaining low memory access cost. Following the deep supervision (Xie and Tu 2015), we propose a booster training strategy. As the name implies, it is similar to the rocket booster. It enhances the feature representation in the training phase and is discarded

in the inference phase. It adds constraints for each stage to enhance the semantic information.

Therefore, it does not increase any computational complexity in the inference phase. As illustrated in Fig. 3, we can insert the auxiliary segmentation head into different positions of the semantics branch. In Sect. 4.1, we analyze the effect of insertion at different positions. Figure 7 illustrates the details of the segmentation head. We can adjust the computational complexity of auxiliary segmentation head and main segmentation head by controlling the channel dimension C_t.

4 Experimental Results

In this section, we first introduce the datasets and the implementation details. Next, we investigate the effects of each component of our proposed approach on the Cityscapes validation set. Finally, we report our final accuracy and speed results on different benchmarks compared with other algorithms.

Datasets. Cityscapes (Cordts et al. 2016) focuses on semantic understanding of urban street scenes from the perspective of a car. The dataset is split into training, validation and test sets of 2975, 500 and 1525 images respectively. In our experiments, we only use the fine annotated images to validate the effectiveness of our proposed method. The annotation includes 30 classes, 19 of which are used for the semantic segmentation task. This dataset is challenging for the real-time semantic segmentation because of its high resolution of 2048 × 1024.

Cambridge-driving Labeled Video Database (CamVid) (Brostow et al. 2008a) is a road scene dataset from the perspective of a driving automobile. It contains 701 images with 960 × 720 resolution extracted from the video sequences. Following the pioneering work of Brostow et al. (2008b); Sturgess et al. (2009); Badrinarayanan et al. (2017), the images are split into 367 for training, 101 for validation and 233 for testing. We use the subset of 11 classes of the provided 32 candidate categories for fair comparison with other methods. The pixels that do not belong to one of these classes are ignored.

COCO-Stuff (Caesar et al. 2018) augments 10K complex images of the popular COCO (Lin et al. 2014) dataset with dense stuff annotations. This is also a challenging dataset for the real-time semantic segmentation because it has more complex categories, including 91 thing and 91 stuff classes for evaluation. For a fair comparison, we follow the split in (Caesar et al. 2018) and use 9K images for training and 1K images for testing.

ADE20K (Zhou et al. 2019) is a scene understanding dataset, containing 20K training images and 2K validation images, with up to 150 category labels. Due to numerous categories and challenging scenes, this dataset is quite challenging for the real-time semantic segmentation methods.

Training. Our models are trained from scratch using the initialization manner proposed in (He et al. 2015). We use the stochastic gradient descent (SGD) algorithm with 0.9 momentum to train our model. For all datasets, we adopt 16 batch size. For the Cityscapes and CamVid datasets, the weight decay is 0.0005 weight decay while the weight decay is 0.0001 for the COCO-Stuff and ADE20K datasets. We note that the weight decay regularization is only employed on the parameters of the convolution layers. The initial rate is set to $5e^{-2}$ with a "poly" learning rate strategy in which the initial rate is multiplied by $(1 - \frac{iter}{iters_{max}})^{power}$ each iteration with power 0.9. Moreover, we train the model for 150K, 10K, 20K, 150K iterations for the Cityscapes, CamVid, COCO-Stuff, and ADE20K datasets, respectively.

For the augmentation, we randomly horizontally flip, randomly scale, and randomly crop the input images to a fixed size for training. The random scales contain {0.75, 1, 1.25, 1.5, 1.75, 2.0}. The cropped resolutions are 2048 × 1024 for Cityscapes, 960 × 720 for CamVid, 640 × 640 for COCO-Stuff, 480 × 480 for ADE20K, respectively. Moreover, the augmented input of Cityscapes will be resized into 1024 × 512 resolution to train our model.

Inference. We do not adopt any evaluation tricks, e.g., sliding-window evaluation and multi-scale testing that can improve accuracy but are time-consuming. For Cityscapes, with the input of 2048 × 1024 resolution, we first resize it to 1024 × 512 resolution for inference and then resize the prediction to the original size of the input. We measure the inference time with only one GPU card and repeat 5000 iterations to eliminate the error fluctuation. We note that the time of resizing is included in the inference time measurement. In other words, when measuring the inference time, the practical input size is 2048 × 1024. Meanwhile, we adopt the standard metric of the mean intersection of union (mIoU) for the Cityscapes and CamVid datasets, while the mIoU and pixel accuracy (pixAcc) are used as metrics for the COCO-Stuff and ADE20K datasets.

Setup. We conduct experiments based on PyTorch 1.0. The models are converted to ONNX format and optimized with TensorRT v5.1.5[1] The inference time compared with other methods is obtained by running on one NVIDIA GeForce GTX 1080Ti with the CUDA 9.0, CUDNN 7.0.

We also report the inference time on the NVIDIA Jetson TX2 with the TensorRT optimization.

[1] For running on GTX 1080Ti, we use FP32 data precision to compare with other methods.

Table 2 *Ablations on Cityscapes*. We validate the effectiveness of each component step by step. We show segmentation accuracy (mIoU%), and computational complexity measured in GFLOPs with the input of spatial size 2048 × 1024. Notation: *Detail* is the detail branch. *Semantics* is the semantics branch. *BGA* represents the bilateral guided aggregation layer. *Booster* means the booster training strategy. *OHEM* is the online hard example mining

Detail	Semantics	Aggregation			Booster	OHEM	mIoU	GFLOPs
		Sum	Concate	BGA				
✓							62.35	15.26
	✓						64.68	7.63
✓	✓	✓					68.60	20.77
✓	✓		✓				68.93	21.98
✓	✓			✓			69.67	21.15
✓	✓			✓	✓		73.19	21.15
✓	✓			✓	✓	✓	**73.36**	21.15

(a) Stage_1 **(b)** Stage_2 **(c)** Stage_3

Fig. 8 *Examples showing visual explanations for the different stages of the detail branch.* Following the Grad-CAM (Selvaraju et al. 2017), we visualize the Grad-CAMs of the detail branch. The visualization shows that the detail branch can focus on the spatial details, e.g., boundary, gradually

4.1 Ablative Evaluation on Cityscapes

This section introduces the ablation experiments to validate the effectiveness of each component in our method. In the following experiments, we train our models on the Cityscapes (Cordts et al. 2016) training set and evaluate on the Cityscapes validation set.

Individual pathways. We first explore the effect of individual pathways specifically. The first two rows in Table 2 illustrate the segmentation accuracy and computational complexity of using one pathway only. The detail branch lacks sufficient high-level semantics, while the semantics branch suffers from a lack of low-level spatial details, leading to unsatisfactory results. Figure 8 shows the gradual attention on the spatial details of the detail branch. The second group in Table 2 shows that the different combinations of both branches are better in all cases than the models with only one pathway. Both branches can provide complementary representations to achieve better segmentation performance. The semantics branch and the detail branch alone only achieve 64.68% and 62.35% mIoU, respectively. However, with the simple summation, the semantics branch shows an improvement of more than 6% compared to the detail branch, while the detail branch acquires 4% gain for the semantics branch. This observation shows that both representations are complementary and essential for the real-time semantic segmentation task.

Aggregation methods. We also investigate the aggregation methods of two branches, as illustrated in Table 2. For an effective and efficient aggregation, we design the bilateral

Table 3 Ablations on the semantics branch design on Cityscapes. We conduct experiments investigating the channel capacity, the block design, and the expansion ratio of the semantics branch. Notation: *GLayer* indicates the gather layer, the first 3×3 convolution in the GE layer. *DDWConv* is double depthwise convolution layer

(a) *Channel capacity ratio* [a]

	mIoU	GFLOPs
Detail-only	62.35	15.26
$\lambda = 1/2$	69.66	25.84
1/4	**69.67**	21.15
1/8	69.26	19.93
1/16	68.27	19.61

(b) *Block Analysis* [b]

GLayer	DDWConv	Context	mIoU	GFLOPs
✓	✓	✓	**69.67**	21.15
✓	✓		69.01	21.07
✓		✓	68.98	21.15
	✓	✓	66.62	15.78

(c) *Expansion ratio* [c]

	mIoU	GFLOPs
Detail-only	62.35	15.26
$\epsilon = 1$	67.48	17.78
2	68.41	18.45
4	68.78	19.8
6	**69.67**	21.15
8	68.99	22.49

[a] Varying values of λ can control the channel capacity of the first two stages in the semantics branch. The channel dimensions of the last two stages are still 64 and 128. Here, we choose $\lambda = 1/4$

[b] We specifically design the GE Layer and adopt double depth-wise convolutions when $stride = 2$. The second row means we use one 5×5 depth-wise convolution instead of two 3×3 depth-wise convolution. The third row represents we replace the first 3×3 convolution layer of GE Layer with the 1×1 convolution

[c] Varying values of ϵ can affect the representative ability of the semantics branch. We choose the $\epsilon = 6$ to make the trade-off between accuracy and complexity

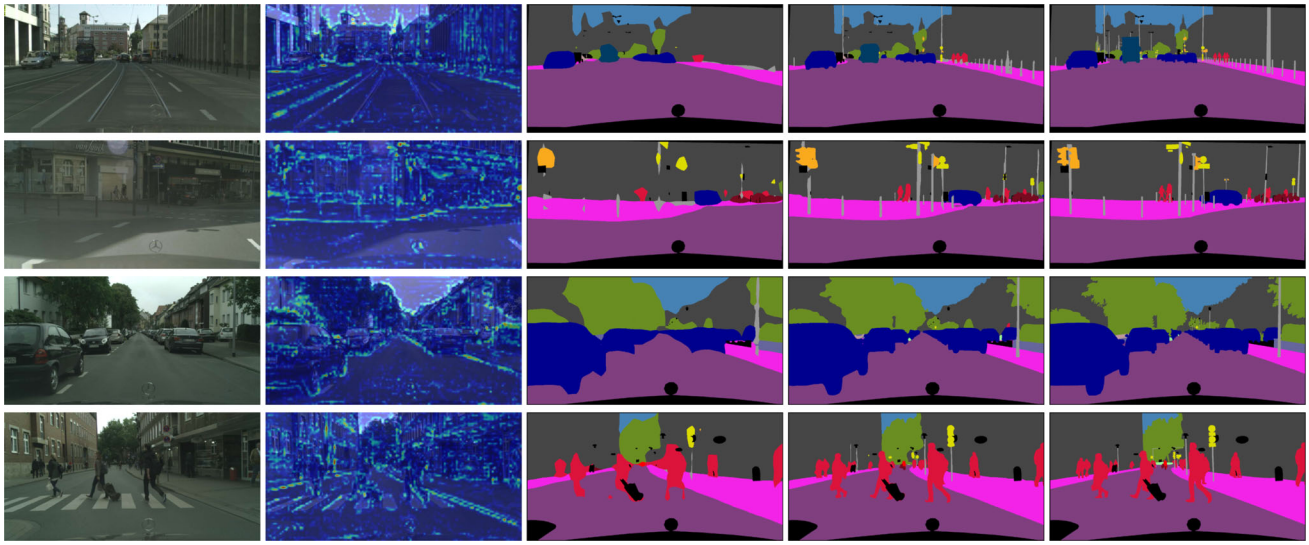
guided aggregation layer that adopts the high-level semantics as the guidance to aggregate the multi-scale low-level details. We also show two variants without the bilateral guided aggregation layer as the naive aggregation baseline: *summation* and *concatenation* of the outputs of both branches. For fair comparison, the inputs of the *summation* and *concatenation* go through one separable layer, respectively. Figure 9 demonstrates the visualization outputs of the detail branch, the semantics branch and the aggregation of both branches. This illustrates that the detail branch can provide sufficient spatial details, while the semantics branch captures the semantic context.

Table 3 illustrates a series of analysis experiments on the semantics branch design.

Channel capacity of the semantics branch. As discussed in Sect. 3.2, the semantics branch is responsible for the high-level semantics and does not consider the spatial details. Therefore, the semantics branch can be made very lightweight with narrow channel dimension that is modified using the channel capacity ratio of λ. Table 3a presents the results of the detailed comparison experiments for varying λ.

Different λ results in different extents of improvement to the Detail-only baseline. Even for $\lambda = 1/16$, the first layer of the semantics branch has only 4 channel dimensions, leading to an improvement of 6% (62.35% $\rightarrow$ 68.27%) relative to the baseline. Here, we employ $\lambda = 1/4$ as our default.

Block design of the semantics branch. Following the pioneer work (Sandler et al. 2018; Howard et al. 2019), we design a gather-and-expansion layer, as discussed in Sect. 3.2 and illustrated in Fig. 5. The main improvements are two-fold: (i) We adopt one 3×3 convolution as the gather layer instead of one pointwise convolution in the inverted bottleneck of MobileNetV2 (Sandler et al. 2018); (ii) When $stride = 2$, we employ two 3×3 depthwise convolutions to substitute a 5×5 depthwise convolution.



(a) Input **(b)** Detail Branch **(c)** Semantics Branch **(d)** Prediction **(e)** Groundtruth

Fig. 9 *Visual improvem ent of the bilateral guided aggregation layer on the Cityscapes val* **set**

Table 4 *Booster position*. We can add the auxiliary segmentation head to different positions as the booster of the semantics branch. Here, stage_s represents that the auxiliary segmentation head is added after s stage. stage_{5_4} and stage_{5_5} means the position before and after the context embedding block respectively. OHEM represents the online bootstrapping strategy

Stage_2	Stage_3	Stage_4	Stage_{5_4}	Stage_{5_5}	OHEM	mIoU 69.67
✓	✓	✓	✓	✓		73.04
✓	✓	✓	✓			73.19
✓	✓	✓				71.62
	✓	✓	✓	✓		72.84
		✓	✓	✓		72.68
			✓	✓		72.03
✓	✓	✓	✓		✓	**73.36**

Table 3b shows the improvement of our block design. The gather-and-expansion Layer can enlarge the receptive field to efficiently capture high-level semantics.

Expansion ratio of GE layer. The first 3×3 convolution layer in the GE layer is also an expansion layer that can project the input to a high-dimensional space. It has an advantage with regard to memory access cost (Sandler et al. 2018; Howard et al. 2019). The expansion ratio of ϵ can control the output dimension of this layer. Table 3c investigates the effect of varying ϵ. It is surprising to observe that even with $\epsilon = 1$, the semantics branch can also improve the baseline by 4% (62.35% $\rightarrow$ 67.48%) mean IoU, validating that the lightweight semantics branch is efficient and effective.

Booster training strategy. We propose a booster training strategy to further improve segmentation accuracy, as discussed in Sect. 3.4. We insert the segmentation head illustrated in Fig. 7 into different positions of the semantics branch in the training phase that are discarded in the inference phase. Therefore, they do not increase the computational complexity in the inference phase, which is similar to the booster of the rocket. Table 4 shows the effect of insertion of the segmentation head at different positions. As we can see, the booster training strategy can obviously improve segmentation accuracy. We choose the configuration of the third row of Table 4 that further improves the mean IoU by over 3% (69.67% $\rightarrow$ 73.19%), without sacrificing the inference speed. Based on this configuration, we adopt the online bootstrapping strategy (Wu et al. 2016) to improve the performance further.

4.2 Generalization Capability

In this section, we mainly explore the generalization capability of our proposed architecture. First, we investigate the performance of a wider model and deeper model in Table 5. Next, we replace the semantics branch with some other general lightweight models to explore the compatibility with the results shown in Table 6.

Generalization to large models. Although our architecture is designed mainly for the lightweight task, e.g., real-time semantic segmentation, BiSeNet V2 can also be generalized to large models. We mainly enlarge the architecture in two aspects: (i) Wider models, controlled by the width multiplier α; (ii) Deeper models, controlled by the depth multiplier d. Table 5 shows the segmentation accuracy and computational complexity of wider models with the different width multiplier α and different depth multiplier d. Based on the experimental results, we choose $\alpha = 2.0$ and $d = 3.0$ to

Table 5 *Generalization to large models*. We enlarge our models from two aspects: (i) Wider models; (ii) Deeper models

(a) *Wider models* [a]

Wider	mIoU	GFLOPs
$\alpha = 1.0$	73.36	21.15
1.25	73.61	34.98
1.50	74.67	49.46
1.75	74.04	66.45
2.0	**75.11**	85.94

(b) *Deeper models* [b]

Deeper	mIoU	GFLOPs
$d = 1.0$	73.36	21.15
2.0	74.10	25.26
3.0	**74.28**	29.38
4.0	74.02	33.5

[a] Varying values of α can control the channel capacity of our architecture

[b] Varying values of d represents the layer number of the model

Table 6 *Compatibility with other models*. We employ different lightweight models as the semantics branch to explore the compatibility of our architecture

Semantics Branch	Pretrained	mIoU	GFLOPs
ShuffleNetV2 1.5×	ImageNet	74.07	128.71
MobileNetV2	ImageNet	72.95	129.45
ResNet-18	ImageNet	75.22	143.34
Ours($\alpha = 1.0, d = 1.0$)	no	73.36	21.15
Ours($\alpha = 2.0, d = 3.0$)	no	**75.80**	118.51

build our large architecture, termed BiSeNetV2-Large; the architecture achieves 75.8% mIoU and GFLOPs.

Compatibility with other models. BiSeNetV2 is a generic architecture with two branches. In this work, we design some specific blocks for the semantics branch. The semantics branch can be any lightweight convolutional models (He et al. 2016; Howard et al. 2017). Therefore, to explore the compatibility of our architecture, we conduct a series of experiments with different general lightweight models. Table 6 shows the results of the combination with different models.

4.3 Performance Evaluation

In this section, we compare our best model (BiSeNetV2 and BiSeNetV2-Large) with other *state-of-the-art* methods on four benchmark datasets: Cityscapes, CamVid, COCO-Stuff, and ADE20K. Finally, we report the runtime efficiency on the NVIDIA Jeston TX2.

Cityscapes. We present the segmentation accuracy and inference speed of the proposed BiSeNetV2 on the Cityscapes *test* set. We use the training set and validation set with 2048×1024 input to train our models that is resized into 1024×512 resolution at first in the models. Then, the models are evaluated on the test set. The measurement of the inference time is conducted on one NVIDIA GeForce 1080Ti card. Table 7 reports the comparison results of our method and state-of-the-art methods. The first group is non-real-time methods, containing CRF-RNN (Zheng et al. 2015), Deeplab-CRF (Chen et al. 2015), FCN-8S (Long et al. 2015), Dilation10 (Yu and Koltun 2016), LRR (Ghiasi and Fowlkes 2016), Deeplabv2-CRF (Chen et al. 2018), FRRN (Pohlen et al. 2017), RefineNet (Lin et al. 2017), DUC (Wang et al. 2018a), PSPNet (Zhao et al. 2017). The real-time semantic segmentation algorithms are listed in the second group, including ENet (Paszke et al. 2016), SQ (Treml et al. 2016), ESPNet (Mehta et al. 2018), ESPNetV2 (Mehta et al. 2019), ERFNet (Romera et al. 2018), Fast-SCNN (Poudel et al. 2019), ICNet (Zhao et al. 2018a), DABNet (Li et al. 2019a), DFANet (Li et al. 2019b), GUN (Mazzini 2018), SwiftNet (Orsic et al. 2019), SwiftNet-pyr (Orsic and Segvic 2021), and BiSeNetV1 (Yu et al. 2018a). The third group is our methods with different levels of complexities.

As shown in Table 7, our method achieves 72.6% mean IoU with 156 FPS. BiSeNetV2 achieves better segmentation results and higher inference speed than most other real-time methods. Compared to DFANet, BiSeNetV2 achieves higher segmentation results with comparable inference speed. Compared to SwiftNet and SwiftNet-pyr, BiSeNetV2 obtains a lower result but far higher inference speed. The pre-trained models adopted by these methods can improve the performance dramatically. Our models are all trained from scratch. With scaling up to larger model, BiSeNetV2-Large achieves 75.3% mean IoU with 47.3 FPS. These results are even better than those of some non-real-time algorithms in the first group of Table 7. We note that many non-real-time methods may adopt some evaluation tricks, e.g., multi-scale testing and multi-crop evaluation, that dramatically improve the accuracy but are time-consuming. Therefore, we do not adopt this strategy due to the need to consider the inference speed. For better visualization, we illustrate the trade-off between performance and speed in Fig. 1. To highlight the effectiveness of our method, we also present some visual examples of BiSeNetV2 on Cityscapes in Fig. 11.

CamVid. Table 8 shows the results for statistical accuracy metrics and speed on the CamVid dataset. In the inference phase, we use the training dataset and validation dataset to train our model with 960×720 resolution input. Our models are compared to some non-real-time algorithms, namely, SegNet (Badrinarayanan et al. 2017), Deeplab (Chen et al. 2015), RTA (Huang et al. 2018), Dilate8 (Yu and Koltun 2016), PSPNet (Zhao et al. 2017), VideoGCRF (Chandra et al. 2018), and DenseDecoder (Bilinski and Prisacariu

Table 7 *Comparison with state-of-the-art on Cityscapes*. We train and evaluate our models with 2048 × 1024 resolution input that is resized into 1024 × 512 in the model. The inference time is measured on one NVIDIA GeForce 1080Ti card. Notation: γ is the downsampling ratio corresponding to the original 2048 × 1024 resolution. *backbone* indicates the backbone models pre-trained on the ImageNet dataset. † represents that the models are trained from scratch. "–" represents that the methods do not report the corresponding result. The DFANet A and DFANet B adopt the 1024 × 1024 input size and use the optimized depthwise convolutions to increase speed

Method	Ref.	γ	Backbone	mIoU *val*	mIoU *Test*	FPS
Large models						
CRF-RNN*	ICCV2015	0.5	VGG16	–	62.5	1.4
DeepLab*	ICLR2015	0.5	VGG16	–	63.1	0.25
FCN-8S*	CVPR2015	1.0	VGG16	–	65.3	2
Dilation10	ICLR2016	1.0	VGG16	68.7	67.1	0.25
LRR	ECCV2016	1.0	VGG16	70.0	69.7	–
Deeplabv2	ICLR2016	1.0	ResNet101	71.4	70.4	–
FRRN	CVPR2017	0.5	no	–	71.8	2.1
RefineNet	CVPR2017	1.0	ResNet101	–	73.6	–
DUC	WACV2018	1.0	ResNet101	76.7	76.1	–
PSPNet	CVPR2017	1.0	ResNet101	–	**78.4**	0.78
Small models						
ENet	arXiv2016	0.5	no	–	58.3	76.9
SQ	NIPSW2016	1.0	SqueezeNet	–	59.8	16.7
ESPNet	ECCV2018	0.5	ESPNet	–	60.3	112.9
ESPNetV2	CVPR2019	0.5	ESPNetV2	66.4	66.2	–
ERFNet	TITS2018	0.5	no	70.0	68.0	41.7
Fast-SCNN	BMVC2019	1.0	no	68.6	68.0	123.5
ICNet	ECCV2018	1.0	PSPNet50	–	69.5	30.3
DABNet	BMVC2019	1.0	no	–	70.1	27.7
DFANet B	CVPR2019	0.5*	Xception B	–	67.1	120
DFANet A′	CVPR2019	0.5	Xception A	–	70.3	**160**
DFANet A	CVPR2019	0.5*	Xception A	–	71.3	100
GUN	BMVC2018	0.5	DRN-D-22	69.6	70.4	33.3
SwiftNet	CVPR2019	1.0	ResNet18	75.4	75.5	39.9
SwiftNet pyr	PR2021	1.0	ResNet18†	72.2	–	34.0
SwiftNet pyr	PR2021	1.0	ResNet18	–	75.9	34.0
BiSeNetV1	ECCV2018	0.75	Xception39	69.0	68.4	105.8
BiSeNetV1	ECCV2018	0.75	ResNet18	74.8	74.7	65.5
BiSeNetV2		0.5	no	73.4	72.6	156
BiSeNetV2-L		0.5	no	**75.8**	75.3	47.3

2018), and real-time algorithms, namely, ENet (Paszke et al. 2016), ICNet (Zhao et al. 2018a), DABNet (Li et al. 2019a), DFANet (Li et al. 2019b), SwiftNet (Orsic et al. 2019), SwiftNet-pyr (Orsic and Segvic 2021), BiSeNetV1 (Yu et al. 2018a).

BiSeNetV2 achieves higher segmentation accuracy with much higher inference speed than most methods. Compared to DFANet, BiSeNetV2 is slower but obtains far higher segmentation results. Moreover, we investigate the effect of the pre-training datasets on CamVid. The last two rows of Table 8 show that pre-training on Cityscapes can greatly improve the mean IoU by over 6% on the CamVid test set.

COCO-Stuff. We also report our accuracy and speed results on the COCO-Stuff validation dataset in Table 9. In the inference phase, we pad the input into 640 × 640 resolution. For fair comparison (Long et al. 2015; Chen et al. 2018; Zhao et al. 2017, 2018a), we do not adopt any time-consuming testing tricks, such as multi-scale and flipping testing. Even for the more complex categories in this dataset, compared to the pioneering work, our BiSeNetV2 still shows greater efficiency and achieve comparable accuracy.

ADE20K. Table 10 reports the results of our method and other state-of-the-art methods, including CPNet (Yu et al. 2020b), RGNet (Yu et al. 2020a), PSANet (Zhao et al. 2018b), PSPNet (Zhao et al. 2017), Dilated FCN (Chen et al. 2018), and Swift-pyr (Orsic and Segvic 2021). BiSeNetV2 achieves 29.2% mIoU. With scaling up to larger model, BiSeNetV2-Large obtains 32.5% mIoU. These results validate the generalization ability and model capacity of our methods. Compared to large models, our methods achieves

Table 8 *Comparison with state-of-the-art on CamVid*. With 960 × 720 input, we evaluate the segmentation accuracy and corresponding inference speed. Notation: *backbone* means that the backbone models pre-trained on the additional datasets, e.g., the ImageNet dataset and the Cityscapes dataset. ∗ indicates that the models are pre-trained on Cityscapes. † represents that the models are trained from scratch

Method	Ref.	Backbone	mIoU	FPS
Large models				
SegNet	TPAMI2017	VGG16	60.1	4.6
DPN	ICCV2015	VGG16	60.1	1.2
Deeplab	ICLR2015	VGG16	61.6	4.9
RTA	ECCV2018	VGG16	62.5	0.2
Dilation8	ICLR2016	VGG16	65.3	4.4
PSPNet	CVPR2017	ResNet50	69.1	5.4
DenseDecoder	CVPR2018	ResNeXt101	70.9	–
VideoGCRF*	CVPR2018	ResNet101	75.2	–
Small models				
ENet	arXiv2016	no	51.3	61.2
DFANet B	CVPR2019	Xception B	59.3	160
DFANet A	CVPR2019	Xception A	64.7	120
ICNet	ECCV2018	PSPNet50	67.1	27.8
SwiftNet	CVPR2019	ResNet18†	63.33	–
SwiftNet	CVPR2019	ResNet18	72.58	–
SwiftNet pyr	PR2021	ResNet18†	65.7	–
SwiftNet pyr	PR2021	ResNet18	73.7	–
BiSeNetV1	ECCV2018	Xception 39	65.6	**175**
BiSeNetV1	ECCV2018	ResNet18	68.7	116.3
BiSeNetV2		no	72.4	124.5
BiSeNetV2-L		no	73.2	32.7
BiSeNetV2*		no	76.7	124.5
BiSeNetV2-L*		no	**78.5**	32.7

Table 9 *Comparison with state-of-the-art on COCO-Stuff*. Our models are trained and evaluated with the input of 640 × 640 resolution. Notation: *backbone* indicates the backbone models pre-trained on the ImageNet dataset

Method	Ref.	Backbone	mIoU	pixAcc	FPS
Large models					
FCN-16s	CVPR2015	VGG16	22.7	52.0	5.9
Deeplab	ICLR2015	VGG16	26.9	57.8	8.1
FCN-8S	CVPR2015	VGG16	27.2	60.4	–
PSPNet50	CVPR2017	ResNet50	**32.6**	–	6.6
Small models					
ICNet	ECCV2018	PSPNet50	29.1	–	35.7
BiSeNetV2		no	25.2	60.5	**87.9**
BiSeNetV2-L		no	28.7	**63.5**	42.5

lower results with far faster inference speed and lower complexity. Compared to SwiftNet-pyr, BiSeNetV2-Large achieves comparable results. We note that SwiftNet-pyr adopts the pre-trained models, which is of importance particularly in the ADE20K dataset because it contains many categories.

Table 10 Comparison with state-of-the-art on the ADE20K validation set. Our models are trained with the input of 480 × 480 resolution. All of the methods report the single-scale results without any testing tricks. Notation: *backbone* indicates the backbone models pre-trained on ImageNet dataset. *RT* means real-time inference speed

Method	Ref.	Backbone	RT	mIoU	pixAcc
Large models					
CPNet	CVPR2020	ResNet50		43.9	80.8
RGNet	ECCV2020	ResNet50		43.1	80.5
PSANet	ECCV2018	ResNet50		41.9	80.2
PSPNet	CVPR2017	ResNet50		41.5	79.6
Dilated FCN	CVPR2015	ResNet50		36.1	77.5
Small models					
SwiftNet pyr	PR2021	ResNet18	✓	35.0	–
BiSeNetV2		no	✓	29.2	74.8
BiSeNetV2-L		no	✓	32.5	75.9

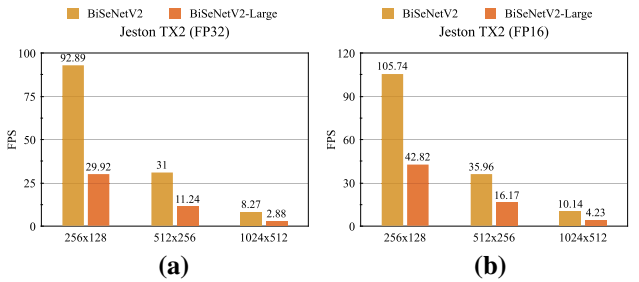


Fig. 10 *Inference speed on NVIDIA Jetson TX2 with different input resolutions*. Both models use TensorRT to optimize with FP32 (**a**) and FP16 precision (**b**)

Runtime efficiency on Jetson TX2. We employ our methods on the NVIDIA Jetson TX2 with PyTorch and TensorRT frameworks to measure the inference time. The technical specifications of TX2 are shown below: AI performance: 1.33 TFLOPs; Memory: 8 GB 128-bit LPDDR4, 59.7GB/s. Fig. 10 shows the inference speeds of BiSeNetV2 and BiSeNetV2-Large with 32-bit (FP32) and 16-bit (FP16) floating-point precision. Due to the limited resources, we did not conduct the experiments on 1024 × 2048 resolution.

5 Concluding Remarks

We observe that the semantic segmentation task requires both low-level details and high-level semantics. We propose a new architecture to treat the spatial details and categorical semantics separately, that is termed the bilateral segmen-

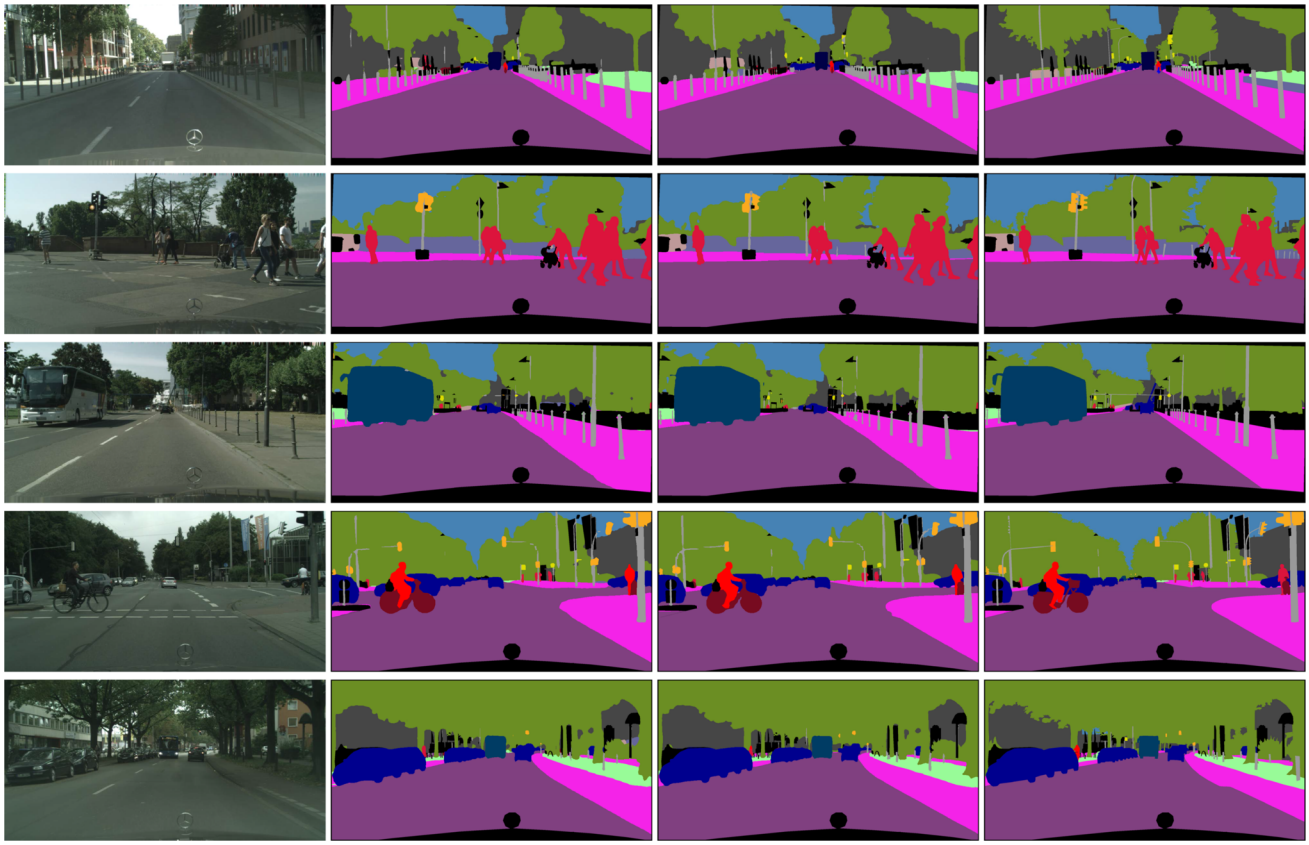


Fig. 11 *Visualization examples on the Cityscapes val* set produced from BiSeNetV2 and BiSeNetV2-Large. The first row shows that our architecture can focus on the details, e.g., fence. The bus in the third row demonstrates the architecture can capture a large object. The bus in the last row illustrates that the architecture can encode the spatial context to analyze it

tation network (BiSeNetV2). The BiSeNetV2 framework is a generic architecture that can be implemented by most convolutional models. Our implementations of BiSeNetV2 achieve a good trade-off between segmentation accuracy and inference speed. We hope that this generic architecture BiSeNetV2 will stimulate further research in semantic segmentation.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 61433007 and 61876210).

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(11), 2274–2282.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(12), 2481–2495.

Bilinski, P., & Prisacariu, V. (2018). Dense decoder shortcut connections for single-pass semantic segmentation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 6596–6605.

Boykov, Y. Y., Jolly, M. P. (2001) Interactive graph cuts for optimal boundary and region segmentation of objects in nd images. In: Proc. IEEE International Conference on Computer Vision, vol 1, pp. 105–112.

Brostow, G. J., Shotton, J., Fauqueur, J., & Cipolla, R. (2008a). Segmentation and recognition using structure from motion point clouds. In: Proc. European Conference on Computer Vision, pp. 44–57.

Brostow, G. J., Shotton, J., Fauqueur, J., & Cipolla, R. (2008b). Segmentation and recognition using structure from motion point clouds. In: Proc. European Conference on Computer Vision.

Caesar, H., Uijlings, J., & Ferrari, V. (2018). Coco-stuff: Thing and stuff classes in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1209–1218.

Chandra, S., Couprie, C., & Kokkinos, I. (2018). Deep spatio-temporal random fields for efficient video segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8915–8924.

Chen LC, Papandreou G, Schroff F, Adam H (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2015). Semantic image segmentation with deep convolutional nets and fully connected crfs. In: Proceedings of the International Conference on Learning Representations.

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision, pp. 801–818.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). *cudnn: Efficient primitives for deep learning*. arXiv preprint arXiv:1410.0759.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258.

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.

Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., & Lu, H. (2019). Dual attention network for scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3146–3154.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The kitti vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 3354–3361.

Ghiasi, G., & Fowlkes, C. C. (2016). LLaplacian pyramid reconstruction and refinement for semantic segmentation. In: Proceedings of the European Conference on Computer Vision, Springer, pp. 519–534.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Glorot, X., Bordes, A., Bengio, Y. (2011) Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp. 315–323.

Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 447–456.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034.

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.

Howard, A. G., 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 preprint arXiv:1704.04861.

Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1314–1324.

Huang, P. Y., Hsu, W. T., Chiu, C. Y., Wu, T. F., & Sun, M. (2018). Efficient uncertainty estimation for semantic segmentation in videos. In: Proceedings of the European Conference on Computer Vision, pp. 520–535.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., Keutzer, K. (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. arXiv preprint arXiv:1602.07360.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning, pp. 448–456.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105.

Li, X., Liu, Z., Luo, P., Change Loy, C., & Tang, X. (2017). Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3193–3202.

Li, H., Xiong, P., Fan, H., & Sun, J. (2019b). Dfanet: Deep feature aggregation for real-time semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9522–9531.

Li, G., Yun, I., Kim, J., & Kim, J. (2019a). Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In: Proceedings of the British Machine Vision Conference.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision, Springer, pp. 740–755.

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.

Liu, W., Rabinovich, A., Berg, A. C. (2016) Parsenet: Looking wider to see better. arXiv preprint arXiv:1506.04579.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440.

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, pp. 116–131.

Mazzini, D. (2018). Guided upsampling network for real-time semantic segmentation. In: Proceedings of the British Machine Vision Conference.

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, pp. 552–568.

Mehta, S., Rastegari, M., Shapiro, L. G., & Hajishirzi, H. (2019). spnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9190–9200.

Orsic, M., & Segvic, S. (2021). Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition, 110,* 107611.

Orsic, M., Kreso, I., Bevandic, P., & Segvic, S. (2019). In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 12607–12616.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics, 9*(1), 62–66.

Paszke A, Chaurasia A, Kim S, Culurciello E (2016) Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147

Peng, C., Zhang, X., Yu, G., Luo, G., & Sun, J. (2017). Large kernel matters–improve semantic segmentation by global convolutional

network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353–4361.
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.
Poudel, R. P., Bonde, U., Liwicki, S., & Zach, C. (2018). Contextnet: Exploring context and detail for semantic segmentation in real-time. In: Proceedings of the British Machine Vision Conference.
Poudel, R. P., Liwicki, S., & Cipolla, R. (2019). Fast-scnn: Fast semantic segmentation network. In: Proc. British Machine Vision Conference.
Ren, X., & Malik, J. (2003). Learning a classification model for segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, p. 10–17.
Romera, E., Alvarez, J. M., Bergasa, L. M., & Arroyo, R. (2018). Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems, 19*(1), 263–272.
Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp. 234–241.
Rother, C., Kolmogorov, V., & Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG), 23,* 309–314.
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520.
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626.
Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(4), 640–651.
Shen, Z., Liu, Z., Li, J., Jiang, Y. G., Chen, Y., & Xue, X. (2017). Dsod: Learning deeply supervised object detectors from scratch. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1919–1927.
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations.
Sturgess, P., Alahari, K., Ladicky, L., & Torr, P. H. S. (2009). Combining appearance and structure from motion features for road scene understanding. In: Proceedings of the British Machine Vision Conference.
Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 31.
Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820–2828.
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. *Proceeding of the Neural Information Processing Systems Workshops*.
Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., & Van Gool, L. (2012). Seeds: Superpixels extracted via energy-driven sampling. In: Proceedings of the European Conference on Computer Vision, pp. 13–26.
Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(6), 583–598.
Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., & Cottrell, G. (2018a). Understanding convolution for semantic segmentation. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, IEEE, pp. 1451–1460.
Wang, R. J., Li, X., & Ling, C. X. (2018b). Pelee: A real-time object detection system on mobile devices. *Proc* (pp. 1967–1976). Advances in Neural Information Processing Systems: Curran Associates Inc.
Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine* Intelligence. https://doi.org/10.1109/TPAMI.2020.2983686.
Wu, Z., Shen, C., Hengel Avd (2016) High-performance semantic segmentation using very deep fully convolutional networks. arXiv preprint arXiv:1604.04339
Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1395–1403.
Yu, F., & Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In: Proceedings of the International Conference on Learning Representations.
Yu, C., Liu, Y., Gao, C., Shen, C., & Sang, N. (2020a). Representative graph neural network. In: Proceedings of the European Conference on Computer Vision, Springer, pp. 379–396.
Yu, C., Wang, J., Gao, C., Yu, G., Shen, C., & Sang, N. (2020b). Context prior for scene segmentation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 12416–12425.
Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018a). Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European Conference on Computer Vision, pp. 325–341.
Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018b). Learning a discriminative feature network for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1857–1866.
Yu, C., Xiao, B., Gao, C., Yuan, L., Zhang, L., Sang, N., & Wang, J. (2021). Lite-hrnet: A lightweight high-resolution network. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 10440–10450.
Yuan, Y., Huang, L., Guo, J., Zhang, C., Chen, X., & Wang, J. (2021). Ocnet: Object context network for scene parsing. *International Journal of Computer Vision,* 129, 2375–2398.
Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., & Agrawal, A. (2018a). Context encoding for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7151–7160.
Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018b). 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.
Zhao, H., Qi, X., Shen, X., Shi, J., & Jia, J. (2018a). Icnet for real-time semantic segmentation on high-resolution images. In: Proceedings European Conference on Computer Vision, pp. 405–420.
Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 2881–2890.
Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C. C., Lin, D., & Jia, J. (2018b). PSANet: Point-wise spatial attention network for scene parsing. In: Proceedings of the European Conference on Computer Vision, pp. 267–283.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1529–1537.

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., et al. (2019). Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, *127*(3), 302–321.