

# Rethinking BiSeNet For Real-time Semantic Segmentation

Mingyuan Fan\*, Shenqi Lai\*, Junshi Huang†, Xiaoming Wei†, Zhenhua Chai,  
Junfeng Luo, Xiaolin Wei  
Meituan

{fanmingyuan, laishenqi, huangjunshi, weixiaoming, chaizhenhua,  
luojunfeng, weixiaolin02}@meituan.com

## Abstract

BiSeNet [28, 27] has been proved to be a popular two-stream network for real-time segmentation. However, its principle of adding an extra path to encode spatial information is time-consuming, and the backbones borrowed from pretrained tasks, e.g., image classification, may be inefficient for image segmentation due to the deficiency of task-specific design. To handle these problems, we propose a novel and efficient structure named Short-Term Dense Concatenate network (STDC network) by removing structure redundancy. Specifically, we gradually reduce the dimension of feature maps and use the aggregation of them for image representation, which forms the basic module of STDC network. In the decoder, we propose a Detail Aggregation module by integrating the learning of spatial information into low-level layers in single-stream manner. Finally, the low-level features and deep features are fused to predict the final segmentation results. Extensive experiments on Cityscapes and CamVid dataset demonstrate the effectiveness of our method by achieving promising trade-off between segmentation accuracy and inference speed. On Cityscapes, we achieve 71.9% mIoU on the test set with a speed of **250.4 FPS** on NVIDIA GTX 1080Ti, which is **45.2% faster** than the latest methods, and achieve **76.8% mIoU** with 97.0 FPS while inferring on higher resolution images. Code is available at <https://github.com/MichaelFan01/STDC-Seg>.

## 1. Introduction

Semantic segmentation is a classic and fundamental topic in computer vision, which aims to assign pixel-level labels in images. The prosperity of deep learning greatly promotes the performance of semantic segmentation by making various breakthroughs [18, 27, 22, 4], coming with fast-growing demands in many applications, e.g., autonomous driving, video surveillance, robot sensing, and so

\*Equal contribution.

†Co-corresponding author.

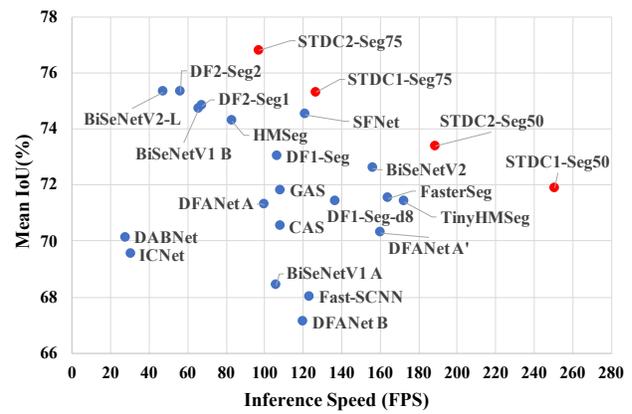


Figure 1. Speed-Accuracy performance comparison on the Cityscapes test set. Our methods are presented in red dots while other methods are presented in blue dots. Our approaches achieve state-of-the-art speed-accuracy trade-off.

on. These applications motivate researchers to explore effective and efficient segmentation networks, particularly for mobile field.

To fulfill those demands, many researchers propose to design low-latency, high-efficiency CNN models with satisfactory segmentation accuracy. These real-time semantic segmentation methods have achieved promising performance on various benchmarks. For real-time inference, some works, e.g., DFANet [18] and BiSeNetV1 [28] choose the lightweight backbones and investigate ways of feature fusion or aggregation modules to compensate for the drop of accuracy. However, these lightweight backbones borrowed from image classification task may not be perfect for image segmentation problem due to the deficiency of task-specific design. Besides the choice of lightweight backbones, restricting the input image size is another commonly used method to promote the inference speed. Smaller input resolution seems to be effective, but it can easily neglect the detailed appearance around boundaries and small objects. To tackle this problem, as shown in Figure 2(a), BiSeNet [28, 27] adopt multi-path framework to combine the low-level details and high-level semantics. However,

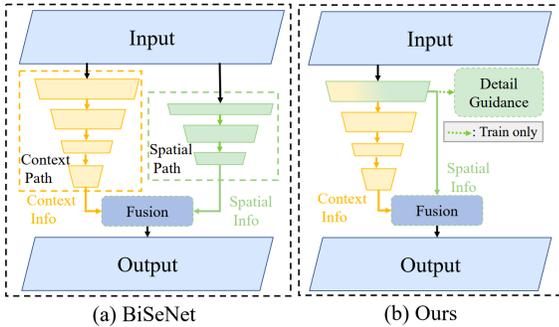


Figure 2. Illustration of architectures of BiSeNet [28] and our proposed approach. (a) presents Bilateral Segmentation Network (BiSeNet [28]), which use an extra *Spatial Path* to encode spatial information. (b) demonstrates our proposed method, which use a *Detail Guidance* module to encode spatial information in the low-level features without an extra time-consuming path.

adding an additional path to get low-level features is time-consuming, and the auxiliary path is always lack of low-level information guidance.

To this end, we propose a novel hand-craft network for the purpose of faster inference speed, explainable structure, and competitive performance to that of existing methods. First, we design a novel structure, called Short-Term Dense Concatenate module (STDC module), to get variant scalable receptive fields with a few parameters. Then, the STDC modules are seamlessly integrated into U-net architecture to form the STDC network, which greatly promote network performance in semantic segmentation task.

In details, as shown in Figure 3, we concatenate response maps from multiple continuous layers, each of which encodes input image/feature in different scales and respective fields, leading to multi-scale feature representation. To speed up, the filter size of layers is gradually reduced with negligible loss in segmentation performance. The details structure of STDC networks can be found in Table 2.

In the phase of decoding, as shown in Figure 2(b), instead of utilizing an extra time-consuming path, Detail Guidance are adopted to guide the low-level layers for the learning of spatial details. We first utilize Detail Aggregation module to generate detail ground-truth. Then, the binary cross-entropy loss and dice loss are employed to optimize the learning task of detail information, which is considered as one type of side-information learning. It should be noted that this side-information is not required in the inference time. Finally, the spatial details from low-level layers and semantic information from deep layers are fused to predict the semantic segmentation results. The whole architecture of our method is shown in Figure 4.

Our main contributions can be summarized as follows:

- We design a Short-Term Dense Concatenate module (STDC module) to extract deep features with scalable receptive field and multi-scale information. This module promotes the performance of our STDC network

with affordable computational cost.

- We propose the Detail Aggregation module to learn the decoder, leading to more precise preservation of spatial details in low-level layers without extra computation cost in the inference time.
- We conduct extensive experiments to present the effectiveness of our methods. The experiment results present that STDC networks achieve new state-of-the-art results on ImageNet, Cityscapes and CamVid. Specifically, our STDC1-Seg50 achieves 71.9% mIoU on the Cityscapes *test* set at a speed of **250.4 FPS** on one NVIDIA GTX 1080Ti card. Under the same experiment setting, our STDC2-Seg75 achieves **76.8% mIoU** at a speed of 97.0 FPS.

## 2. Related Work

### 2.1. Efficient Network Designs

Model design plays an important role in computer vision tasks. SqueezeNet [16] used the fire module and certain strategies to reduce the model parameters. MobileNet V1 [13] utilized depth-wise separable convolution to reduce the FLOPs in inference phase. ResNet [9] [10] adopted residual building layers to achieve outstanding performance. MobileNet V2 [25] and ShuffleNet [29] used group convolution to reduce computation cost while maintaining comparable accuracy. These works are particularly designed for the image classification tasks, and their extensions to semantic segmentation application should be carefully tuned.

### 2.2. Generic Semantic Segmentation

Traditional segmentation algorithms, *e.g.*, threshold selection, super-pixel, utilized the hand-crafted features to assign pixel-level labels in images. With the development of convolution neural network, methods [3, 1, 32, 14] based on FCN [23] achieved impressive performance on various benchmarks. The Deeplabv3 [3] adopted an atrous spatial pyramid pooling module to capture multi-scale context. The SegNet [1] utilized the encoder-decoder structure to recover the high-resolution feature maps. The PSPNet [32] devised a pyramid pooling to capture both local and global context information on the dilation backbone. Both dilation backbone and encoder-decoder structure can simultaneously learn the low-level details and high-level semantics. However, most approaches require large computation cost due to the high-resolution feature and the complicate network connections. In this paper, we propose an efficient and effective architecture which achieves good trade-off between speed and accuracy.

### 2.3. Real-time Semantic Segmentation

Recently, there are fast-growing practical applications for real-time semantic segmentation. In this circumstance,

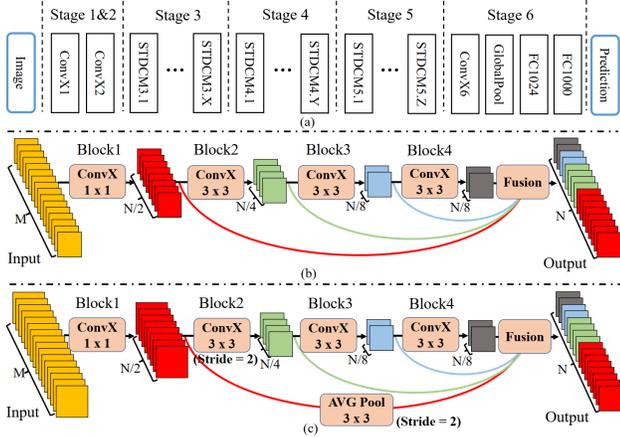


Figure 3. (a) General STDC network architecture.  $ConvX$  operation refers to the Conv-BN-ReLU. (b) Short-Term Dense Concatenate module (STDC module) used in our network.  $M$  denotes the dimension of input channels,  $N$  denotes the dimension of output channels. Each block is a  $ConvX$  operation with different kernel size. (c) STDC module with stride=2.

there are two mainstreams to devise efficient segmentation methods. (i) *lightweight backbone*. DFANet [18] adopted a lightweight backbone to reduce computation cost and devised a cross-level feature aggregation module to enhance performance. DFNet [21] utilized “Partial Order Pruning” algorithm to obtain a lightweight backbone and efficient decoder. (ii) *multi-branch architecture*. ICNet [31] devised the multi-scale image cascade to achieve good speed-accuracy trade-off. BiSeNetV1 [28] and BiSeNetV2 [27] proposed two-stream paths for low-level details and high-level context information, separately. In this paper, we propose an efficient lightweight backbone to provide scalable receptive field. Furthermore, we set a single path decoder which uses detail information guidance to learn the low-level details.

### 3. Proposed Method

BiSeNetV1 [28] utilizes lightweight backbones, *e.g.*, ResNet18 and spatial path as encoding networks to form two-stream segmentation architecture. However, the classification backbones and two-stream architecture may be inefficient due to the structure redundancy. In this section, we first introduce the details of our proposed STDC network. Then we present the whole architecture of our single-stream method with detail guidance.

#### 3.1. Design of Encoding Network

##### 3.1.1 Short-Term Dense Concatenate Module

The key component of our proposed network is the Short-Term Dense Concatenate module (STDC module). Figure 3(b) and (c) illustrate the layout of STDC module. Specifically, each module is separated into several blocks,

STDC module	Block1	Block2	Block3	Block4	Fusion
RF( $S=1$ )	$1 \times 1$	$3 \times 3$	$5 \times 5$	$7 \times 7$	$1 \times 1, 3 \times 3$ $5 \times 5, 7 \times 7$
RF( $S=2$ )	$1 \times 1$	$3 \times 3$	$7 \times 7$	$11 \times 11$	$3 \times 3$ $7 \times 7, 11 \times 11$

Table 1. Receptive Field of blocks in our STDC module.  $RF$  denotes Receptive Field,  $S$  means stride, Note that if stride=2, the  $1 \times 1$   $RF$  of Block1 is turned into  $3 \times 3$   $RF$  by Average Pool operation.

and we use  $ConvX_i$  to denote the operations of  $i$ -th block. Therefore, the output of  $i$ -th block is calculated as follows:

$$x_i = ConvX_i(x_{i-1}, k_i) \quad (1)$$

where  $x_{i-1}$  and  $x_i$  are the input and output of  $i$ -th block, separately.  $ConvX$  includes one convolutional layer, one batch normalization layer and ReLU activation layer, and  $k_i$  is the kernel size of convolutional layer.

In STDC module, the kernel size of first block is 1, and the rest of them are simply set as 3. Given the channel number of STDC module’s output  $N$ , the filter number of convolutional layer in  $i$ -th block is  $N/2^i$ , except the filters of last convolutional layer, whose number is the same to that of previous convolutional layer. In image classification tasks, its a common practice to using more channels in higher layers. But in semantic segmentation tasks, we focus on scalable receptive field and multi-scale informations. Low-level layers need enough channels to encode more fine-grained informations with small receptive field, while high-level layers with large receptive field focus more on high-level information induction, setting the same channel with low-level layers may cause information redundancy. Down-sample is only happened in Block2. To enrich the feature information, we concatenate  $x_1$  to  $x_n$  feature maps as the output of STDC module by skip-path. Before concatenation, the response maps of different blocks in STDC module is down-sampled to the same spatial size by average pooling operation with  $3 \times 3$  pooling size, as shown in Figure 3(c). In our setting, the final output of STDC module is:

$$x_{output} = F(x_1, x_2, \dots, x_n) \quad (2)$$

where  $x_{output}$  denotes the STDC module output,  $F$  is the fusion operation in our method, while  $x_1, x_2, \dots, x_n$  are feature maps from all  $n$  blocks. In the consideration of efficiency, we adopt concatenation as our fusion operation. In our method, we use the STDC module in 4 blocks.

Table 1 presents the receptive field of blocks in STDC module, and  $x_{output}$  thus gathers multi-scale information from all blocks, We claim that our STDC module has two advantages: (1) we elaborately tune the filter size of blocks by gradually decreasing in geometric progression manner, leading to significant reduction in computation complexity. (2) the final output of STDC module is concatenated from

Stages	Output size	KSize	S	STDC1		STDC2	
				R	C	R	C
Image	224×224				3		3
ConvX1	112×112	3×3	2	1	32	1	32
ConvX2	56×56	3×3	2	1	64	1	64
Stage3	28×28		2	1	256	1	256
	28×28		1	1		3	
Stage4	14×14		2	1	512	1	512
	14×14		1	1		4	
Stage5	7×7		2	1	1024	1	1024
	7×7		1	1		2	
ConvX6	7×7	1×1	1	1	1024	1	1024
GlobalPool	1×1	7×7					
FC1					1024		1024
FC2					1000		1000
FLOPs					813M		1446M
Params					8.44M		12.47M

Table 2. Detailed architecture of STDC networks. Note that *ConvX* shown in the table refers to the Conv-BN-ReLU. The basic module of Stage 3, 4 and 5 is STDC module. KSize mean kernel size. S, R, C denote stride, repeat times and output channels respectively.

all blocks, which preserves scalable respective fields and multi-scale information.

Given the input channel dimension  $M$  and output channel dimension  $N$ , the parameter number of STDC module is:

$$\begin{aligned}
S_{param} &= M \times 1 \times 1 \times \frac{N}{2^1} + \sum_{i=2}^{n-1} \frac{N}{2^{i-1}} \times 3 \times 3 \times \frac{N}{2^i} + \\
&\quad \frac{N}{2^{n-1}} \times 3 \times 3 \times \frac{N}{2^{n-1}} \\
&= \frac{NM}{2} + \frac{9N^2}{2^3} \times \sum_{i=0}^{n-3} \frac{1}{2^{2i}} + \frac{9N^2}{2^{2n-2}} \\
&= \frac{NM}{2} + \frac{3N^2}{2} \times \left(1 + \frac{1}{2^{2n-3}}\right) \quad (3)
\end{aligned}$$

As shown in Equation 3, the parameter number of STDC module is dominated by the predefined input and output channel dimension, while the number of blocks has slight impact on the parameter size. Particularly, if  $n$  reaches the maximum limit, the parameter number of STDC module almost keeps constant, which is only defined by  $M$  and  $N$ .

### 3.1.2 Network Architecture

We demonstrate our network architecture in Figure 3(a). It consists of 6 stages except input layer and prediction layer. Generally, Stage 1~5 down-sample the spatial resolution of the input with a stride of 2, respectively, and the Stage 6 outputs the prediction logits by one ConvX, one global average pooling layer and two fully connected layer.

The Stage 1&2 are usually regarded as low-level layers for appearance feature extraction. In pursuit of efficiency,

we only use one convolutional block in each of Stage 1&2, which is proved to be sufficient according to our experiences. The number of STDC module in Stage 3, 4, 5 is carefully tuned in our network. Within those stages, the first STDC module in each stage down-samples the spatial resolution with a stride of 2. The following STDC modules in each stage keep the spatial resolution unchanged.

We denote the output channel number of stage as  $N_l$ , where  $l$  is the index of stage. In practice, we empirically set  $N_6$  as 1024, and carefully tune the channel number of rest stages, until reaching a good trade-off between accuracy and efficiency. Since our network mainly consists of Short-Term Dense Concatenate modules, we call our network STDC network. Table 2 shows the detailed structure of our STDC networks.

## 3.2. Design of Decoder

### 3.2.1 Segmentation Architecture

We use the pretrained STDC networks as the backbone of our encoder and adopt the context path of BiSeNet [28] to encode the context information. As shown in Figure 4(a), we use the Stage 3, 4, 5 to produce the feature maps with down-sample ratio 1/8, 1/16, 1/32, respectively. Then we use global average pooling to provide global context information with large receptive field. The U-shape structure are deployed to up-sample the features stem from global feature, and combine each of them with the counterparts from last two stages (Stage 4&5) in our encoding phase. Following BiSeNet [28], we use *Attention Refine module* to refine the combination features of every two stages. For the final semantic segmentation prediction, we adopt *Feature Fusion module* in BiSeNet [28] to fuse the 1/8 down-sampled feature from Stage 3 in the encoder and the counterpart from the decoder. We claim that the features of these two stages are in different levels of feature representation. The feature from the encoding backbone preserves rich detail information, while the feature from the decoder contains context information due to the input from global pooling layer. Specifically, the *Seg Head* includes a  $3 \times 3$  Conv-BN-ReLU operator followed with a  $1 \times 1$  convolution to get the output dimension  $N$ , which is set as the number of classes. We adopt cross-entry loss with Online Hard Example Mining to optimize the semantic segmentation learning task.

### 3.2.2 Detail Guidance of Low-level Features

We visualize the features of BiSeNet’s spatial path in Figure 5(b). Compared with the backbone’s low-level layers(Stage 3) of same downsample ratio, spatial path can encode more spatial detail, *e.g.*, boundary, corners. Based on this observation, we propose a Detail Guidance module to guide the low-level layers to learn the spatial information in single-stream manner. We model the detail prediction

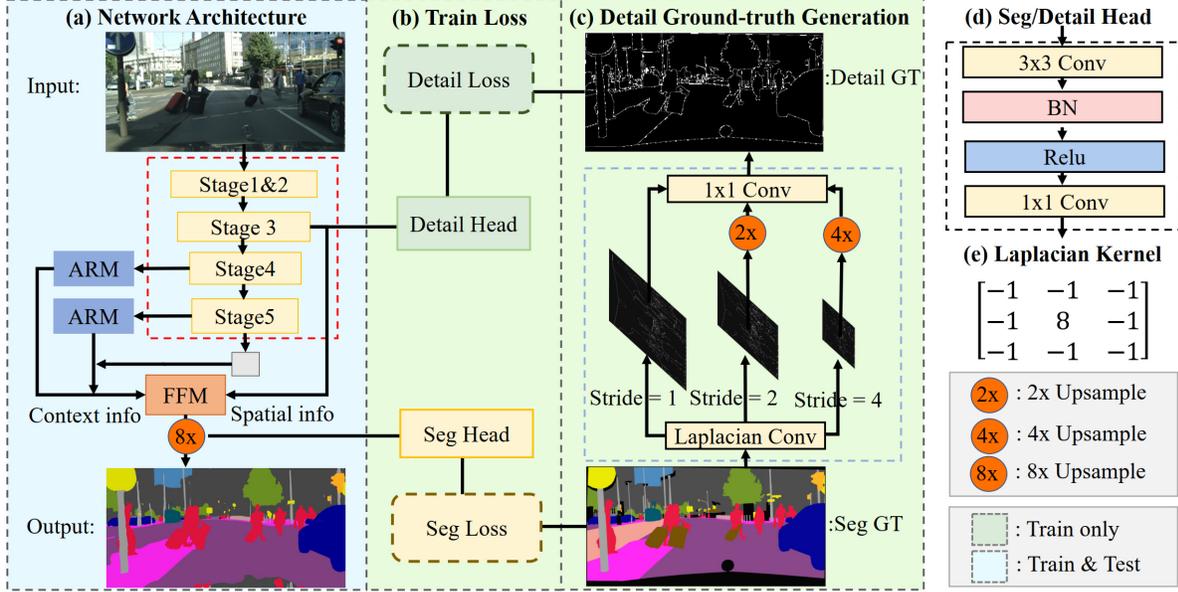


Figure 4. Overview of the STDC Segmentation network. *ARM* denotes *Attention Refine module*, and *FFM* denotes *Feature Fusion Module* in [28]. The operation in the dashed red box is our STDC network. The operation in the dashed blue box is *Detail Aggregation Module*.

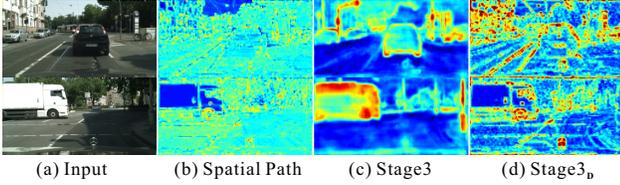


Figure 5. Visual explanations for features in the spatial path and Stage 3 without and with *Detail Guidance*. The column with subscript **D** denotes results with *Detail Guidance*. The visualization shows that spatial path can encode more spatial detail, *e.g.*, boundary, corners, than backbone’s low-level layers, while our *Detail Guidance* module can do the same thing without extra computation cost.

as a binary segmentation task. We first generate the detail map ground-truth from the segmentation ground-truth by Laplacian operator as shown in Figure 4 (c). As illustrated in Figure 4(a), we insert the *Detail Head* in Stage 3 to generate the detail feature map. Then we use the detail ground-truth as the guidance of detail feature map to guide the low-level layers to learn the feature of spatial details. As shown in Figure 5(d), the feature map with detail guidance can encode more spatial details than aforementioned result presented in Figure 5(c). Finally, the learned detail features are fused with the context features from the deep block of the decoder for segmentation prediction.

**Detail Ground-truth Generation:** We generate the binary detail ground-truth from the semantic segmentation ground-truth by our *Detail Aggregation module*, as shown in dashed blue box of Figure 4(c). This operation can be carried out by 2-D convolution kernel named Laplacian kernel and a trainable  $1 \times 1$  convolution. We use the Laplacian operator shown in Figure 4(e) to produce soft thin detail feature

maps with different strides to obtain multi-scale detail informations. Then we upsample the detail feature maps to the original size and fuse it with a trainable  $1 \times 1$  convolution for dynamic re-weighting. Finally, we adopt a threshold 0.1 to convert the predicted details to the final binary detail ground-truth with boundary and corner informations.

**Detail Loss:** Since the number of detail pixels is much less than the non-detail pixels, detail prediction is a class-imbalance problem. Because weighted cross-entropy always leads to coarse results, following [7], we adopt binary cross-entropy and dice loss to jointly optimize the detail learning. Dice loss measures the overlap between predict maps and ground-truth. Also, it is insensitive to the number of foreground/background pixels, which means it can alleviating the class-imbalance problem. So for the predicted detail map with the height  $H$  and the width  $W$ , the detail loss  $L_{detail}$  is formulated as follows:

$$L_{detail}(p_d, g_d) = L_{dice}(p_d, g_d) + L_{bce}(p_d, g_d) \quad (4)$$

where  $p_d \in \mathbb{R}^{H \times W}$  denotes the predicted detail and  $g_d \in \mathbb{R}^{H \times W}$  denotes the corresponding detail ground-truth.  $L_{bce}$  denotes the binary cross-entropy loss while  $L_{dice}$  denotes the dice loss, which is given as follows:

$$L_{dice}(p_d, g_d) = 1 - \frac{2 \sum_i^{H \times W} p_d^i g_d^i + \epsilon}{\sum_i^{H \times W} (p_d^i)^2 + \sum_i^{H \times W} (g_d^i)^2 + \epsilon} \quad (5)$$

where  $i$  denotes the  $i$ -th pixel and  $\epsilon$  is a Laplace smoothing item to avoid zero division. In this paper we set  $\epsilon = 1$ .

As shown in Figure 4(b), we use a *Detail Head* to produce the detail map, which guide the shallow layer to encode spatial information. *Detail Head* includes a  $3 \times 3$

Conv-BN-ReLU operator followed with a  $1 \times 1$  convolution to get the output detail map. In the experiment, the *Detail Head* is proved to be effective to enhance the feature representation. Note that this branch is discarded in the inference phase. Therefore, this side-information can easily boost the accuracy of segmentation task without any cost in inference.

## 4. Experimental Results

We implement our method on three datasets: ImageNet [6], Cityscapes [5] and CamVid [2] to evaluate the effectiveness of our proposed backbone and segmentation network, respectively. We first introduce the datasets and implementation details. Then, we report our accuracy and speed results on different benchmarks compared with other algorithms. Finally, we discuss the impact of components in our proposed approach.

### 4.1. Benchmarks and Evaluation Metrics

**ImageNet.** The ILSVRC [6] 2012 is the most popular image classification dataset. It contains 1.2 million images for training, and 50,000 for validation with 1,000 categories. It is also widely used for training a pretrained model for downstream tasks, like object detection or semantic segmentation.

**Cityscapes.** Cityscapes [5] is a semantic scene parsing dataset, which is taken from a car perspective. It contains 5,000 fine annotated images and split into training, validation and test sets, with 2,975, 500 and 1,525 images respectively. The annotation includes 30 classes, 19 of which are used for semantic segmentation task. The images have a high resolution of  $2,048 \times 1,024$ , thus it is challenging for the real-time semantic segmentation. For fair comparison, we only use the fine annotated images in our experiments.

**CamVid.** Cambridge-driving Labeled Video Database (Camvid) [2] is a road scene dataset, which is taken from a driving automobile perspective. This dataset contains 701 annotated images extracted from the video sequence, in which 367 for training, 101 for validation and 233 for testing. The images have a resolution of  $960 \times 720$  and 32 semantic categories, in which the subset of 11 classes are used for segmentation experiments.

**Evaluation Metrics.** For classification evaluation, we use evaluate top-1 accuracy as the evaluation metrics following [9]. For segmentation evaluation, we adopt mean of class-wise intersection over union (mIoU) and Frames Per Second (FPS) as the evaluation metrics.

### 4.2. Implementation Details

**Image Classification.** We use mini-batch stochastic gradient descent (SGD) with batch size 64, momentum 0.9 and weight decay  $1e^{-4}$  to train the model. Three training methods from [11] are adopted, including learning rate warmup,

Backbone	Resolution	mIoU(%)	FPS
GhostNet [8]	$512 \times 1024$	67.8	135.0
MobileNetV3 [12]	$512 \times 1024$	70.1	148.3
EfficientNet-B0 [26]	$512 \times 1024$	72.2	99.9
STDC2	$512 \times 1024$	<b>74.2</b>	<b>188.6</b>
GhostNet [8]	$768 \times 1536$	71.3	60.9
MobileNetV3 [12]	$768 \times 1536$	73.0	70.4
EfficientNet-B0 [26]	$768 \times 1536$	73.9	45.9
STDC2	$768 \times 1536$	<b>77.0</b>	<b>97.0</b>

Table 3. Lightweight backbone comparison on Cityscapes *val* set. All experiments utilize the same decoder and same experiment settings.

cosine learning rate policy and label smoothing. The total epochs is 300 with warmup strategy at the first 5 epochs, within which the learning rate starts from 0.001 to 0.1. The dropout before classification block is set to 0.2. We do not use other special data augmentations, and all of them are the same as [9].

**Semantic Segmentation.** We use mini-batch stochastic gradient descent (SGD) with momentum 0.9, weight decay  $5e^{-4}$ . The batch size is set as 48, 24 for the Cityscapes, CamVid dataset respectively. As common configuration, we utilize "poly" learning rate policy in which the initial rate is multiplied by  $(1 - \frac{iter}{max.iter})^{power}$ . The power is set to 0.9 and the initial learning rate is set as 0.01. Besides, we train the model for 60,000, 10,000 iterations for the Cityscapes, CamVid dataset respectively, in which we adopt warmup strategy at the first 1000, 200 iterations.

Data augmentation contains color jittering, random horizontal flip, random crop and random resize. The scale ranges in [0.125, 1.5] and cropped resolution is  $1024 \times 512$  for training Cityscapes. For training CamVid, the scale ranges in [0.5, 2.5] and cropped resolution is  $960 \times 720$ .

In all experiments, we conduct our experiments base on pytorch-1.1 on a docker. We perform all experiments under CUDA 10.0, CUDNN 7.6.4 and TensorRT 5.0.1.5 on NVIDIA GTX 1080Ti GPU with batch size 1 for benchmarking the computing power of our method.

### 4.3. Ablation Study

This section introduces the ablation experiments to validate the effectiveness of each component in our method.

**Effectiveness of STDC Module.** We adjust the block number of STDC module in STDC2 and present the result in Figure 7. According to our Equation 3, as the group number increases, the FLOPs decrease obviously. And the best performance is in 4 blocks. The benefits of more blocks become very small and a deeper network is bad for the parallel calculation and FPS. Hence, in this paper, we set the block number in STDC1 and STDC2 to 4.

**Effectiveness of Our backbone.** To verify the effectiveness of our backbone designed for real-time segmentation, we adopt the latest lightweight backbones which has com-

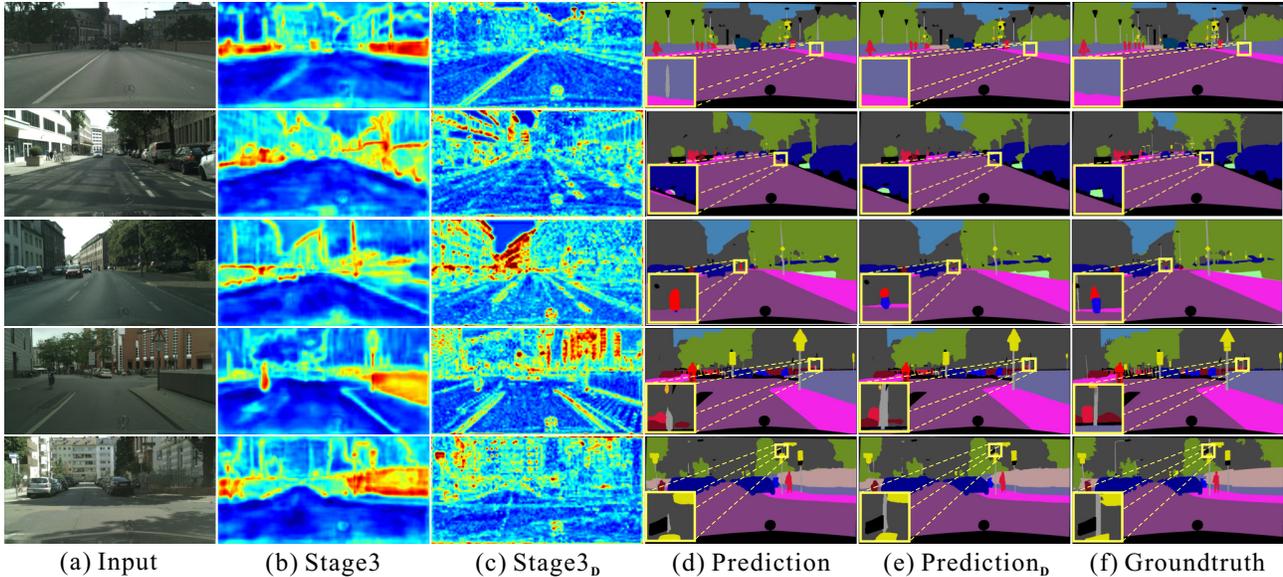


Figure 6. Visual comparison of our *Detail Guidance* on Cityscapes *val* set. The column with subscript **D** denotes results with *Detail Guidance*. The first row (a) shows the input images. (b) and (c) illustrate the heatmap of Stage 3 without and with *Detail Guidance*. (d) and (e) demonstrate the predictions without and with *Detail Guidance*. (f) is the ground-truth of input images.

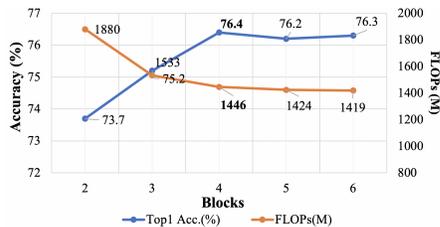


Figure 7. Comparisons with different block number of STDC2 on ImageNet.

Method	SP	DG			mIoU(%)	FPS
		4x	2x	1x		
BiSeNetV1 [28]	✓				69.0	105.8
STDC2-50	✓				73.7	171.6
STDC2-50					73.0	188.6
STDC2-50		✓			73.4	188.6
STDC2-50			✓		73.6	188.6
STDC2-50				✓	73.8	188.6
STDC2-50			✓	✓	73.9	188.6
STDC2-50		✓	✓	✓	<b>74.2</b>	188.6

Table 4. Detail information comparison on Cityscapes *val* set. *SP* means method with *Spatial Path* and *DG* indicates *Detail Guidance*, in which *1x*, *2x*, *4x* denotes detail features with different down-sample strides in *Detail Aggregation* module.

comparable classification performance compared with STDC2 to formulate a semantic segmentation network with our decoder. As shown in Table 3, our STDC2 yields the best speed-accuracy trade-off compared with other lightweight backbones.

**Effectiveness of Detail Guidance.** We first visualize the heatmap of the feature map of Stage 3 as shown in Figure 6. The features of Stage 3 with detail guidance encode

more spatial information comparing to that of Stage 3 without detail guidance. Hence the final prediction of small objects and boundaries are more precise. We show some quantitative results in Table 4. To verify the effectiveness of our Detail Guidance, we show the comparison of different detail guidance strategies of STDC2-Seg on Cityscapes *val* dataset. To further demonstrate the capability of Detail Guidance, we first use the Spatial Path in BiSeNetV1 [28] to encode the spatial information, then use the features generated from the Spatial Path to replace the features from Stage 3<sub>D</sub>. The setting of experiment with Spatial Path are exactly the same with other experiments. As shown in Table 4, Detail Guidance in STDC2-Seg can improve the mIoU without harming the inference speed. Adding Spatial Path to encode spatial information can also improve the performance on accuracy, but it increases the computation cost at the same time. Also we find our Detail Aggregation module encode the abundant detail information and yield the highest mIoU with aggregation of *1x*, *2x*, *4x* detail features.

#### 4.4. Compare with State-of-the-arts

In this part, we compare our methods with other existing state-of-the-art methods on three benchmarks, ImageNet, Cityscapes and CamVid.

**Results on ImageNet.** As shown in Table 5, our STDC networks achieve higher speed and accuracy compared with other lightweight backbones. Compared with the lightweight backbones used in real-time segmentation, e.g., DF1Net, the top-1 classification accuracy of our STDC1 network is 4.1% higher than baseline on the ImageNet *validation* set. Compared with popular lightweight networks,

Model	Top1 Acc.	Params	FLOPs	FPS
ResNet-18 [9]	69.0%	11.2M	1800M	1058.7
ResNet-50 [9]	75.3%	23.5M	3800M	378.7
DF1 [21]	69.8%	8.0M	746M	1281.3
DF2 [21]	73.9%	17.5M	1770M	713.2
DenseNet121 [15]	75.0%	9.9M	2882M	363.6
DenseNet161 [15]	76.2%	28.6M	7818M	255.0
GhostNet(x1.0) [8]	73.9%	5.2M	141M	699.1
GhostNet(x1.3) [8]	75.7%	7.3M	226M	566.2
MobileNetV2 [25]	72.0%	3.4M	300M	998.8
MobileNetV3 [12]	75.2%	5.4M	219M	661.2
EfficientNet-B0 [26]	76.3%	5.3M	390M	443.0
STDC1	73.9%	8.4M	813M	<b>1289.0</b>
STDC2	<b>76.4%</b>	12.5M	1446M	813.6

Table 5. Comparisons with other popular networks on ImageNet Classification.

Model	Resolution	Backbone	mIoU(%)		FPS
			val	test	
ENet [24]	512 × 1024	no	-	58.3	76.9
ICNet [31]	1024 × 2048	PSPNet50	-	69.5	30.3
DABNet [17]	1024 × 2048	no	-	70.1	27.7
DFANet B [18]	1024 × 1024	Xception B	-	67.1	120
DFANet A' [18]	512 × 1024	Xception A	-	70.3	160
DFANet A [18]	1024 × 1024	Xception A	-	71.3	100
BiSeNetV1 [28]	768 × 1536	Xception39	69.0	68.4	105.8
BiSeNetV1 [28]	768 × 1536	ResNet18	74.8	74.7	65.5
CAS [30]	768 × 1536	no	-	70.5	108.0
GAS [22]	769 × 1537	no	-	71.8	108.4
DF1-Seg-d8 [21]	1024 × 2048	DF1	72.4	71.4	136.9
DF1-Seg[21]	1024 × 2048	DF1	74.1	73.0	106.4
DF2-Seg1[21]	1024 × 2048	DF2	75.9	74.8	67.2
DF2-Seg2[21]	1024 × 2048	DF2	76.9	75.3	56.3
SFNet [20]	1024 × 2048	DF1	-	74.5	121
HMSeg [19]	768 × 1536	no	-	74.3	83.2
TinyHMSeg [19]	768 × 1536	no	-	71.4	172.4
BiSeNetV2 [27]	512 × 1024	no	73.4	72.6	156
BiSeNetV2-L [27]	512 × 1024	no	75.8	75.3	47.3
FasterSeg [4]	1024 × 2048	no	73.1	71.5	163.9
STDC1-Seg50	512 × 1024	STDC1	72.2	71.9	<b>250.4</b>
STDC2-Seg50	512 × 1024	STDC2	74.2	73.4	188.6
STDC1-Seg75	768 × 1536	STDC1	74.5	75.3	126.7
STDC2-Seg75	768 × 1536	STDC2	<b>77.0</b>	<b>76.8</b>	97.0

Table 6. Comparisons with other state-of-the-art methods on Cityscapes. *no* indicates the method do not have a backbone.

*e.g.* EfficientNet-B0, the FPS of STDC2 network is 83.7% higher than that of baseline with competitive classification result.

**Results on Cityscapes.** As shown in Table 6, we present the segmentation accuracy and inference speed of our proposed method on Cityscapes *validation* and *test* set. Following the previous methods [27, 22], we use the training set and validation set to train our models before submitting to Cityscapes online server. At test phase, we first resize the image into the fixed size 512 × 1024 or 768 × 1536 to inference, then we up-sample the results to 1024 × 2048. Overall, our methods get the best speed-accuracy trade-off

Model	Resolution	Backbone	mIoU(%)	FPS
ENet [24]	720 × 960	no	51.3	61.2
ICNet [31]	720 × 960	PSPNet50	67.1	34.5
BiSeNetV1 [28]	720 × 960	Xception39	65.6	175
BiSeNetV1 [28]	720 × 960	ResNet18	68.7	116.3
CAS [30]	720 × 960	no	71.2	169
GAS [22]	720 × 960	no	72.8	153.1
BiSeNetV2 [27]	720 × 960	no	72.4	124.5
BiSeNetV2-L [27]	720 × 960	no	73.2	32.7
STDC1-Seg	720 × 960	STDC1	73.0	<b>197.6</b>
STDC2-Seg	720 × 960	STDC2	<b>73.9</b>	152.2

Table 7. Comparisons with other state-of-the-art methods on CamVid. *no* indicates the method do not have a backbone.

among all methods. We use 50 and 75 after the method name to represent the input size 512 × 1024 and 768 × 1536 respectively. For example, with the STDC1 backbone and 512 × 1024 input size, we name the method STDC1-Seg50. As shown in Table 6, our STDC1-Seg50 achieves a significantly faster speed than baselines, *i.e.*, **250.4 FPS**, and still has 71.9% mIoU on *test* set, which is over **45.2%** faster than the runner-up. Our STDC2-Seg50 using 512 × 1024 input size achieves 73.4% mIoU with 188.6 FPS, which is the state-of-the-art trade-off between performance and speed. For 768 × 1536 input size, our STDC2-Seg75 achieves the best mIoU 77.0% in *validation* set and **76.8%** on *test* set at 97.0 FPS.

**Results on CamVid.** We also evaluate our method on CamVid dataset. Table 7 shows the comparison results with other methods. With the input size 720 × 960, STDC1-Seg achieves 73.0% mIoU with 197.6 FPS which is the state-of-the-art trade-off between performance and speed. This further demonstrates the superior capability of our method.

## 5. Conclusions

In this paper, we revisit the classical segmentation architecture BiSeNet [28, 27] for structure optimization. Generally, the classification backbone and extra spatial path of BiSeNet greatly hinder the inference efficiency. Therefore, we propose a novel Short-Term Dense Concatenate Module to extract deep features with scalable receptive field and multi-scale information. Based on this module, STDC networks are designed and achieve competitive accuracy with high FPS in image classification. Using STDC networks as backbone, our detail-guided STDC-Seg achieves state-of-the-art speed-accuracy trade-off in real-time semantic segmentation. Extensive experiments and visualization results indicates the effectiveness of our proposed STDC-Seg networks. In future, we extend our method by following directions: (i) the backbone will be validated in more tasks, *e.g.*, object detection. (ii) we will explore deeper on the utilization of spatial boundary in semantic segmentation tasks.

**Acknowledgements** This research is supported by Beijing Science and Technology Project. (No.Z181100008918018).

## References

- [1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. [2](#)
- [2] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 44–57, 2008. [6](#)
- [3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. [2](#)
- [4] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*, 2020. [1, 8](#)
- [5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. [6](#)
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. [6](#)
- [7] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 562–578, 2018. [5](#)
- [8] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. [6, 8](#)
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. [2, 6, 8](#)
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. [2](#)
- [11] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 558–567, 2019. [6](#)
- [12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. [6, 8](#)
- [13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. [2](#)
- [14] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8827, 2020. [2](#)
- [15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. [8](#)
- [16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv preprint arXiv:1602.07360*, 2016. [2](#)
- [17] Gen Li and Joongkyu Kim. Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In *British Machine Vision Conference*, 2019. [8](#)
- [18] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9522–9531, 2019. [1, 3, 8](#)
- [19] Peike Li, Xuanyi Dong, Xin Yu, and Yi Yang. a. when humans meet machines: Towards efficient segmentation networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2020. [8](#)
- [20] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. *arXiv preprint arXiv:2002.10120*, 2020. [8](#)
- [21] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 9145–9153, 2019. [3, 8](#)
- [22] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2020. [1, 8](#)
- [23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. [2](#)
- [24] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. [8](#)
- [25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. [2, 8](#)

- [26] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. [6](#), [8](#)
- [27] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv preprint arXiv:2004.02147*, 2020. [1](#), [3](#), [8](#)
- [28] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. [1](#), [2](#), [3](#), [4](#), [5](#), [7](#), [8](#)
- [29] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. [2](#)
- [30] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11641–11650, 2019. [8](#)
- [31] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnets for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018. [3](#), [8](#)
- [32] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. [2](#)