Efficient Fast Semantic Segmentation Using Continuous Shuffle Dilated Convolutions

XUEGANG HU¹ AND HAIBO WANG²

¹School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
²Laboratory of Intelligent Analysis and Decision on Complex Systems, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Corresponding author: Haibo Wang (whbstudy@126.com)

This work was supported by the key project of the Natural Science Foundation of Chongqing, China, under Grant cstc2017jcyjBX0037.

ABSTRACT It is difficult for many semantic segmentation methods to perform useful inferences under extremely resource-constrained devices; therefore, an efficient fast semantic segmentation network (EFSNet) that employs a continuous shuffle dilated convolution (CSDC) and a up-sampling module is proposed in this paper. First, the number of parameters is reduced by group convolutions and the receptive field is enlarged by dilated convolution. Second, a up-sampling module is proposed for reducing noise and increasing inference speed. Finally, with the above CSDC module and up-sampling module, the EFSNet based on a fast semantic segmentation network is obtained. Our experiments on the CamVid and Cityscapes datasets show that the mean intersection over union (mIoU) values obtained by the proposed EFSNet with only 173 k parameters are 61.1 % and 61.9 %, respectively. The method is able to run in real-time at a speed of 332 frames per second (FPS) and 107 FPS on a single NVIDIA Titan Xp GPU. Compared with several existing methods, our network achieves efficient segmentation with low resource consumption.

INDEX TERMS Resource-constrained, semantic segmentation, continuous shuffle dilated convolution, real-time.

I. INTRODUCTION

With the rapid development of deep learning, convolutional neural networks (CNNs) have achieved satisfying results in different applications, e.g., classification [1], target detection and tracking [2], [3] and segmentation [4]. Semantic segmentation is used to extract rich contour feature information in an image by dividing each pixel into determinate categories, which has a wide range of applications in many aspects. In terms of image beautification, semantic segmentation is used as an important method to extract human external contours [5]. In addition, it is an important pre-processing step to replace some manual medical image segmentation to extract areas where lesions may occur [6]. It also plays an important role in road scene understanding, such as road edge segmentation [7]. Semantic segmentation is a big challenge in computer vision tasks because it requires precise classification of target pixels of different scales and categories.

Many semantic segmentation methods contain a large number of parameters, resulting in a low inference speed. The most classical method is the fully convolutional network (FCN) [4], which is the pioneer method used to perform end-to-end semantic segmentation. SegNet [8], a symmetric encoder-decoder structure, uses the previous layers of the image classification network VGGNet [9] as the encoder and adopts pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear up-sampling, but the network has many parameters. PSPNet [10] applies the pyramid pooling module to extract low-level features and contextual features with different scales and achieves excellent segmentation performance; However, the forward time is long because of the large number of convolution filters. DANet [11] proposed the self-attention mechanism in semantic segmentation, capturing rich contextual information relationships for better feature representations with intra-class compactness with a position attention module and a channel attention module; But a lot of matrix multiplication operations are required. Although the above methods have achieved competitive accuracy performance, there is no good trade-off between accuracy and speed. In some real-world applications, e.g., autonomous driving and intelligent surveillance, a light-weight network is still needed in embedded devices for fast image prediction.

Recently, some light-weight networks have made great contributions in reducing inference time and enhancing
segmentation accuracy performance. ENet [12] uses the bottleneck block [13] and asymmetric convolutions [14] to reduce the feature map dimensions and floating-point operations (FLOPs), achieving good semantic segmentation performance on embedded devices. CGNet [15] uses regular and dilated convolution to perform feature extraction in parallel, and then obtains the weight vector to guide the feature map fusion through the global average pooling layer and the fully-connected layer. It can learn local features and contextual information with low parameters. ESPNet [16] decomposes efficiently a standard convolutional kernel into a point-wise convolution and multiple dilated convolutions with a spatial pyramid structure, achieving large receptive fields and a good trade-off between speed and accuracy. Zhang et al. construct a fast semantic segmentation network named FSSNet [17] by multiple effective blocks to achieve high accuracy performance with only 0.2M parameters, but it did not have a good accuracy performance for similar inter-class objects. ShuffleNet [18], a light-weight image classification network, uses point-wise group convolution to reduce feature map dimensions first, and increase the dimensions later. In addition, to alleviate accuracy degradation caused by group convolution, it uses a channel shuffle operator to solve poor communication in different groups, achieving high accuracy classification with low computational complexity.

The light-weight semantic segmentation networks use a small number of parameters to achieve low-budget segmentation, but the resulting accuracy is relatively low. Therefore, methods to achieve high-precision semantic segmentation with a limited number of parameters are important and interesting. To further reduce parameters and improve accuracy performance, we propose the efficient fast semantic segmentation network named EFSNet, which adopts channel shuffle dilated convolution (CSDC) to perform fast and accurate segmentation. Unlike CDB [17], our proposed CSDC not only enlarges the receptive field but also reduces the number of FLOPs. It uses group convolution [18] to first reduce and then increase the number of feature map channels, uses dilated convolution [19] to enlarge the receptive field, and uses channel shuffle [18] to alleviate the loss of accuracy. Moreover, we construct a up-sampling module that combines feature map information of the same resolution in the encoder, which is able to generate prediction images with less noise while reducing a large number of element-wise additions. Finally, we use the continuous ShuffleNet Unit [18] to further reduce the number of FLOPs and improve segmentation accuracy performance.

The main contributions in this paper are as follows:

1) We propose an CSDC module, which uses dilated convolution with different spatial dilation rates to efficiently enlarge the receptive field.

2) We construct an up-sampling module that improves inference speed and semantic segmentation accuracy.

3) Our network uses only 173 k parameters and achieves good performance on the Camvid and CityScapes test sets.

II. RELATED WORK

A. HIGH ACCURACY SEMANTIC SEGMENTATION

FCN [4], a pioneer work in image semantic segmentation, replaces the last fully-connected layers with convolutions. SegNet [8], an encoder–decoder architecture, applies max-pooling indices to an up-sampling low-resolution feature map to enhance non-linear transformations. DeepLab [20] uses dilated convolution to enlarge the receptive field for dense labeling, and employs dense conditional random fields for increasing segmentation accuracy. PSPNet [10] adopts multiple branch convolutions with different sizes to extract spatial features, then applies up-sampling to the same size to enrich the details. Zhao et al. [23] use a point-wise spatial attention network to self-adaptively learned contextual informations. DANet [11] proposes a dual attention module based on self-attention mechanism to learn local features and global dependencies. CCNet [24] proposes a criss-cross attention module to utilizes the contextual information of all the pixels on its criss-cross path. These methods are effective, but high FLOPs are required because of the model with many parameters and the lack of efficient techniques for group convolution [18], [21] and depth-wise separable convolution [22].

B. REAL-TIME SEMANTIC SEGMENTATION

Recently, some real-time semantic segmentation networks have received extensive attention because of high speed and accuracy. SQ [25] is based on the low-latency network SqueezeNet [26] with high classification accuracy, first reducing and then increasing the channels of the feature map by point-wise convolutions for real-time segmentation. ICNet [27] based on PSPNet [10], uses low resolution images to rapidly obtain semantic information and high resolution images to refine the details for high accuracy segmentation performance. BiseNet [28] adopts the attention mechanism to capture high level semantic features, and the feature fusion module fuses high-level semantic features and low-level textures to enrich details. ERFNet [29] employs cascaded factorization convolution to enhance non-linear capabilities, while using residual connections to enhance gradient propagation. Although these semantic segmentation networks have achieved high accuracy, they did not work well on mobile devices with low computation budgets, such as mobile phones.

C. LIGHT-WEIGHT NETWORK

Light-weight networks represent a good trade-off between accuracy and model parameters, enabling their deployment on mobile devices. The image classification network ShuffleNet [18], [21] employs the principle of convolution decomposition to reduce parameters by factorizing a standard convolution into point-wise and depth-wise convolutions. Specifically, channel shuffle is used after the point-wise convolution to alleviate the poor communication between different groups caused by group convolution. ENet [12] uses
small convolution kernels instead of standard convolutions for building light-weight semantic segmentation networks, making semantic segmentation based on embedded devices feasible. CGNet [15] applies the context information guidance module to learn local features and features of different stages, but the segmentation speed is slow for high resolution images. ESPNet [16] is an efficient semantic segmentation network with a large receptive field that uses point-wise convolution and a spatial pyramid of dilated convolutions to build efficient feature extraction modules. FSSNet [17] uses continuous factorized blocks to extract low-level features; additionally, a continuous dilated block is applied to enlarge the receptive field. However, FSSNet does not use group convolution; therefore, it still consumes a large number of FLOPs. Our proposed CSDC not only reduces computational complexity but also enlarge the receptive field.

III. OUR APPROACH

In this section, we first describe the EFSNet based on FSSNet [17] architecture. Then, we introduce our proposed CSDC module and up-sampling module in detail and the decoder network. Finally, we perform a series of experiments that show that our method can reduce computational cost and increase segmentation accuracy performance.

A. THE STRUCTURE OF EFSNet

We propose the structure for EFSNet as shown in Fig. 1 and Table 1. EFSNet is a light-weight encoder-decoder semantic segmentation network, which not only learns multi-scale context, but also reduces the computational cost.

The encoder network consists of four parts. First, the initial block [12] reduces the dimension of the high-resolution images for training and prediction. The down-sampling block [17] preserves the salient features and learns the down-sampling feature map by max-pooling operations and spatial convolutions, respectively. The factorization block [17] reduces the parameters by using asymmetric convolutions that decompose a spatial convolution into two smaller convolutions.

shuffle dilated convolution (SDC) and continuous shuffle dilated convolution (CSDC) as shown in Fig. 2. To learn multi-scale features and ease the gridding problem, we replace the $3 \times 3$ convolution filter in the ShuffleNet Unit [18] by dilated $3 \times 3$ convolutions with different dilation rates. Table 6 illustrates that the SDC module can obtain higher segmentation accuracy with fewer parameters compared to the ShuffleNet Unit.

We form the CSDC module by stacking multiple SDC modules with different dilation rates. The CSDC
module learns multi-scale contexts based on the principle of reduce-transform-expand, which not only enlarges the receptive field but also requires less calculational effort. The hyper-parameter $N$ is the number of CSDC modules stacked together to control the depth of the network. CSDC $\times N$ stands for the CSDC module is repeated $N$ times. Table 6 illustrates that an SDC module can build a deeper network with lower number of parameters to improve accuracy performance.

Since the main use of the decoder network is to restore the feature map resolution, a decoder network structure with a low number of parameters can also obtain a decent segmentation probability map. The decoder consists of the up-sampling module, the ShuffleNet Unit [18] and the deconvolution [30]. In the up-sampling module, previous down-sampled feature map and the output of the previous layer are up-sampled twice, and then concatenated to refine the details. Then, the ShuffleNet Unit is used to further improve the prediction accuracy, as shown in Table 2. In the last layer, a transposed convolution with a $3 \times 3$ kernel size and a stride of 2 is performed to generate prediction maps of $C$ classes. When the decoder network performs up-sampling and convolution operations, it combines high-level semantic information with low-level details, thus effectively improving the accuracy of the segmentation.

EFSNet$^1$ and EFSNet$^2$ are obtained by using different dimensionality reduction factors in EFSNet. In EFSNet$^1$ and EFSNet$^2$, except for the initial block, up-sampling block and deconvolution layer, the number of all bottleneck channels is set to 1/4 and 1/2 of the input channel, respectively. The experiments described in IV show that our proposed EFSNet peforms better with respect to parameter size and mIoU when compared to similar methods.

### B. CONTINUOUS SHUFFLE DILATED CONVOLUTION

Shuffle dilated convolution was proposed based on the ShuffleNet Unit [18] and dilated convolution [19], as shown in Fig. 2 (a). Unlike the ShuffleNet unit, SDC uses dilated convolutions with different dilation rates of $r = 2^k, k \in [0, K - 1]$, which enlarge the receptive field without additional parameters. The left branch takes the output of previous layer as the input feature map. In the right branch, a $1 \times 1$ group convolution is used to reduce and increase channels. However, group convolution may cause the problem that outputs from a certain group only relate to the inputs within the group. Channel shuffle [18] is used to alleviate this drawback. Spatial convolution is carried out for each feature map by $3 \times 3$ dilated convolutions.

By stacking four SDC layers with rising expansion rates, the CSDC module is obtained, as illustrated in Fig. 2 (b). Compare to CDB [17], we use a different dilation rates strategy and more convolution layers to build a deeper and wider network. Experiments in Section IV show that our CSDC module achieves higher accuracy with fewer parameters.

### C. UP-SAMPLING MODULE

The feature fusion up-sampling module is divided into two parts, using both the shallow and high-level features of the encoder, as shown in Fig. 3. In the left branch, a $1 \times 1$ convolution is used to reduce the number of channels of the high-level feature map, and then the bilinear interpolation algorithm is used to double-sample the feature map. In the right branch, low-level output feature maps from the down-sampling block reduce the number of input channels using a $1 \times 1$ convolution, and then a $2 \times 2$ deconvolution is

---

**TABLE 2. Ablation study of ours on CamVid dataset. □ denotes CSDC module, △ denotes ShuffleNet Unit, □ denotes Bottleneck Block.**

| Model | Encoder | Decoder | Sky | Building | Pole | Road | Pavement | Tree | Sign | Fence | Car | Pedestrian | Bicyclist | Mean IoU | Global Acc |
|-------|---------|---------|-----|----------|------|------|----------|------|------|-------|-----|-------------|-----------|----------|-------------|
| A     | □       | □       | 95.2 | 80.2     | 57.2 | 94.3 | 94.1     | 85.6 | 63.2 | 39.2  | 84.7| 82.7        | 64.0      | 60.3     | 84.9        |
| B     | △       | □       | 94.6 | 77.9     | 58.9 | 95.4 | 88.4     | 87.9 | 60.1 | 35.1  | 85.2| 78.7        | 62.5      | 58.4     | 83.7        |
| C     | □       | △       | 94.8 | 79.6     | 58.3 | 94.8 | 93.2     | 88.8 | 59.1 | 43.9  | 87.2| 82.1        | 58.1      | 60.7     | 84.8        |
| D     | △       | △       | 94.7 | 79.4     | 56.7 | 93.8 | 90.6     | 86.6 | 52.6 | 43.1  | 85.5| 78.3        | 58.8      | 57.4     | 83.8        |

---

**FIGURE 2. The structure of SDC and CSDC. (a) SDC. (b) CSDC.**

**FIGURE 3. The structure of up-sampling module.**
used to double the up-sampling and gradually increase feature map resolution. Finally, the concatenation operation is used to merge features of two branches; this helps to reduce noise in predicted images. Compared with an up-sampling block [17], our module has fewer parameters. The 1 $\times$ 1 convolutions in the first and second up-sampling modules reduces the number of input channels to 1/4 and 1/8, respectively. The module uses a concatenation operation to fusion features instead of element-wise summation, maintaining similar accuracy and low inference time.

D. DECODER NETWORK

The decoder network consists of an up-sampling block, the ShuffleNet Unit [18] and a transposed convolution [30], as shown in Fig. 1. The main function of the decoder network is to restore the feature map resolution and to output semantic segmentation probability maps. Therefore, an asymmetric decoder can be assigned to reduce parameters and inference time. The decoder network is light-weight, and the number of parameters is less than 20 k when 1 $\times$ 1 convolution is used to decrease and increase the number of channels times.

In the decoder network, we use two continuous ShuffleNet Units instead of the Bottleneck Block, as shown in Fig. 4 (a) and Fig. 4 (b). The ShuffleNet Unit reduces the number of FLOPs and keeps accuracy comparable with Bottleneck Block. The Bottleneck Block was proposed by ResNet [13] to build a thin and deep network, that can be divided into two branches. In the right branch of the Bottleneck Block, the 1 $\times$ 1 convolution is used to first reduce and then increase the feature map channels, and features are extracted using a 3 $\times$ 3 convolution, while maintaining a low parameters count. Element-wise summation is used to merge the two branch outputs, thus alleviating the problem of vanish gradients. The ShuffleNet Unit performs better than the Bottleneck Block. In Fig. 4 (b), a 1 $\times$ 1 group convolution is used to reduce the number of parameters and the computing resource consumption, the channel shuffle increases the information flow between channel groups, and a 3 $\times$ 3 depth-wise convolution performs convolutions on each channel of the input feature map, which greatly reduces the number of parameters compared to a standard convolution. The comparison experiment of the Bottleneck and the ShuffleNet Unit is shown in Table 2. The last layer in the network, a transposed convolution layer with 3 $\times$ 3 kernel size and a stride of 2, generates a segmentation probability map with $C$ channels.

IV. EXPERIMENTS

In this section, we evaluate our EFSNet on the common road scene understanding datasets CamVid [31] and Cityscapes [32]. First, We introduce the experimental details and the evaluation metrics. Then, we perform a series of ablation studies on our CSDC module to validate its effectiveness. Finally, we report our competitive evaluation results and compare them with other state-of-the-art methods on the CamVid and Cityscapes test set.

V. CamVid DATASET

The CamVid dataset is a video set that contains five videos for urban scene understanding. There are 701 labeled images in the dataset. Similar to SegNet [8], we separated the images into 367 training images, 233 testing images, and 101 validation images for fair comparison. There are 11 different classes, such as building, tree, sky, road, bicyclist, etc., ignoring the 12th class that contains unlabeled data. Following SegNet [8], we down-sample each image from 960 $\times$ 720 to 480 $\times$ 360 before training and testing. In the training, we randomly crop the training images to 256 $\times$ 256, and set the batch size as 10.

A. CITYSCAPES DATASET

The Cityscapes dataset is a large-scale urban scene perception dataset that includes 5,000 high-quality finely annotated images and 20,000 coarsely annotated images collected from 50 cities. The entire dataset has a total of 19 small semantic categories, such as car, person and rider. We only use the fine-annotated images training model for verifying the effectiveness of the proposed method. The fine-annotated images contain 2,975 training, 500 validation and 1,525 testing images. During the training, we first down-sample the original image from 2048 $\times$ 1024 to 1024 $\times$ 512, and then random crop it to 720 $\times$ 360 for rapid training. We set the batch size to 16 to acquire better test results.

B. IMPLEMENTATION DETAILS

The proposed method is implemented using PyTorch 1.0.1 [33] open-source deep learning library, a computer with an Intel i5-6500 3.2 GHz, 16GB RAM, and a GeForce GTX 1080Ti GPU card to accelerate training. The version of CUDA is 9.0. We train the networks with the Adam optimization algorithm [34] with an initial learning rate of 0.0005. The ‘poly’ learning rate policy is adopted with power 0.9, and the momentum and weight decay are set to 0.9 and 0.0004, respectively. The learning rate $lr$ is defined as follows:

$$lr = lr_{init} \times \left(1 - \frac{\text{epoch}}{\text{epoch}_{\text{max}}}\right).$$

(1)
TABLE 3. Comparison to published entries for CamVid test set.

| Model          | Prew | Parameters | lps | Sky | Building | Pole | Road | Pavement | Tree | Sign | Fence | Car | Pedestrian | Cyclist | Mean IoU | Global Acc | Class Avg |
|----------------|------|------------|-----|-----|----------|------|------|----------|------|------|-------|-----|-------------|---------|----------|-------------|-----------|
| FCN-8s [4]     | ✓    | 134 M      | 44  | 88.7| 77.8     | 19.9 | 91.2 | 72.7     | 71.0 | 32.7 | 24.4   | 76.1| 41.7        | 31.0    | 57.0     | 88.0        | 57.0      |
| SegNet [8]     | ✓    | 29.5 M     | 53  | 92.4| 88.8     | 27.5 | 97.2 | 84.4     | 87.3 | 20.5 | 49.3   | 82.1| 57.1        | 30.7    | 55.6     | 88.6        | 65.2      |
| ENet [12]      | ×    | 0.37 M     | 153 | 95.1| 74.7     | 35.4 | 95.1 | 86.7     | 77.8 | 51.0 | 51.7   | 82.4| 67.2        | 34.1    | 51.3     | n/a         | 68.3      |
| ESPNet [16]    | ×    | 0.36 M     | n/a | n/a | n/a      | n/a | n/a | n/a      | n/a | n/a | n/a    | n/a | n/a         | n/a     | 55.6     | n/a         | 68.3      |
| FSSNet [17]    | ×    | 0.2 M      | 179 | 94.3| 84.6     | 53.6 | 95.2 | 92.9     | 86.0 | 57.9 | 43.4   | 84.6| 80.9        | 67.4    | 58.6     | 85.5        | 76.4      |
| EFSNet1 (ours) | ×    | 185 k      | 294 | n/a | n/a      | n/a | n/a | n/a      | n/a | n/a | n/a    | n/a | n/a         | n/a     | 56.1     | 87.1        | 70.9      |
| EFSNet2 (ours) | ×    | 173 k      | 352 | 95.4| 84.1     | 54.2 | 93.1 | 93.9     | 86.0 | 57.9 | 38.7   | 86.1| 83.9        | 56.2    | 61.1     | 85.6        | 73.6      |

We initialize the network parameters using He initialization and choose PReLU [35] as our activation function. Batch Normalization [36] is used to accelerate deep network training. The probability of dropout is set to 0.2. Our experiments are carried out on the CamVid and Cityscapes datasets. The cross-entropy loss function is used as the loss function. Because of the imbalance of the training data, different weights are assigned to each class in the training, forcing the network to learn more difficult classes, such as person and rider. Following ENet [12], the class weighting scheme is defined as follows:

$$w_{\text{class}} = \frac{1}{\ln(c + p_{\text{class}})}.$$  \hspace{1cm} (2)

During training, regular data enhancements are performed on the CamVid and Cityscapes datasets, including random horizontal flipping and image cropping, thus improving the generalization performance for the proposed EFSNet. For the low resolution CamVid dataset, the dilation rates of 1, 2, 5, and 9 are adopted. The EFSNet can achieve end-to-end training without pre-training from scratch.

C. EVALUATION METRICS

We evaluated our model using standard strategies to measure network performance, e.g., segmentation accuracy, computational cost, and parameters of the network. These metrics are widely used in real-time semantic segmentation.

For the CamVid dataset, we evaluated our results using global accuracy, mean intersection over union (mIoU), class average accuracy, and per-class accuracy. These metrics are defined as follows:

$$\text{Global Accuracy} = \frac{\sum_i n_{ij}}{\sum_i t_i},$$  \hspace{1cm} (3)

$$\text{IoU} = \frac{1}{n_{\text{cls}}} \sum_{i=1}^{n_{\text{cls}}} \frac{n_{ij}}{t_i + \sum_j n_{ji} - n_{ij}},$$  \hspace{1cm} (4)

where $n_{ij}$ is the number of pixels of class $i$ predicted to belong to class $j$, and there are $n_{\text{cls}}$ different classes, $t_i = \sum_j n_{ji}$ is the total number of pixels of class $i$.

For the Cityscapes dataset, we compare our results using class-wise IoU, category-wise IoU, and instance-level intersection over union (iIoU) with existing state-of-the-art approaches. Class IoU and Category IoU represent the mean performance IoU of nineteen classes and eight categories. Unlike the CamVid dataset, we add an additional instance-level intersection-over-union (iIoU) to evaluate the segmentation accuracy of individual instances in the labeling. The iIoU is formulated as follows:

$$\text{iIoU} = \frac{iTP + FP + iFN}{iTP},$$  \hspace{1cm} (5)

where $FP$ is the number of false positive pixels and $iTP$ and $iFN$ denote weighted counts of true positive and false negative pixels, respectively. In contrast to the standard IoU measure, iIoU is computed by weighting the contribution of each pixel by the ratio of the class average instance size to the size of the respective ground truth labeling.

We use both FLOPs and frames per second (FPS) for evaluating the network computational complexity. FLOPs is an indirect metric [18] that is widely used to evaluate the computational complexity of algorithms. However, depending on the actual operating platform, the operating speeds with similar FLOPs are also very different.

D. RESULT ON CamVid DATASET

In the testing, we compare the global accuracy, the per-class accuracy and the mIoU with existing state-of-the-art approaches. Furthermore, we measure the inference speed of our EFSNet by averaging 100 iterations for 480 × 360 inputs on a NVIDIA Titan Xp GPU. Our proposed EFSNet has only 93 k parameters achieves 60.7 % mIoU on the CamVid test set and processes the inputs at a rate of 376 FPS. In addition, with our EFSNet2, which has 173 k parameters, we obtain 61.1 % mIoU on the test set with 332 FPS.

The experimental results are shown in Table 3, where n/a indicates that the semantic segmentation algorithm did not publish the corresponding results. Table 3 demonstrates that our EFSNet1 achieves the best available trade-off in terms of accuracy and efficiency. Among all the approaches, our EFSNet1 yields 60.7 % mIoU and 84.8 % global pixel accuracy, where 4 out of the 11 categories obtain best scores. Regarding to the efficiency, our method is nearly 7 times faster and uses 317 fewer parameters than SegNet [8]. Furthermore, compare to FSSNet [17], an another fast semantic
segmentation network, our method is 2.1% more accurate, 2.1 times faster and has 2 times fewer parameters.

Our EFSNet\(^2\) with 173 k parameters achieves 61.1% mIoU, and the visual result is better than that achieved with FSSNet. Fig. 5 shows some visual examples of our method and FSSNet on the Camvid dataset. Fig. 7 show our EFSNet produces more accurate results with fewer network parameters than other methods. The figure demonstrate that our method performs better on pole, sign, car, and pedestrian; i.e., the method correctly classifies objects at different scales. Additionally, our network is capable of differentiating larger classes such as building and road, nearly as well as SegNet.

**E. RESULT ON CITYSCAPES DATASET**

During the testing, we up-sampled the output feature map using nearest interpolation to 2048 × 1024 for matching the
official image resolution requirements. Moreover, we measured the inference speed of our method by averaging 100 iterations for 1024 x 512 inputs on a single NVIDIA Titan Xp GPU. In Table 4, we report the quantitative results compared with several baselines. Experimental results show that our EFSNet\(^1\) obtains 57.75% mIoU with only 93 k parameters on the Cityscapes test set and processes the inputs at a rate of 128 FPS.

We can see that our EFSNet\(^2\) outperformed FSSNet [17] in terms of network parameters, class IoU (mIoU), class iIoU, category IoU and inference speed with 173 k network parameters. Specifically, our EFSNet\(^2\) is 3.11% more accurate than FSSNet. Our EFSNet\(^2\) achieves 61.91 % mIoU, and the visualization is better than FSSNet. Fig. 6 shows some visual examples of our method and FSSNet on the Cityscapes dataset. Some visual comparison of our proposed EFSNet and other methods as shown in Fig. 8. The data shows that our results are more accurate than those of FSSNet [17]. For example, the top right picture in Fig. 6 shows an incorrect prediction in building class and the lower right picture in Fig. 6 misrepresents a car in the center of the picture as a person, while our method produces accurate results. In addition, EFSNet\(^2\) achieved the highest iIoU, indicating that our method can achieve more accurate segmentation for small categories such as rider and fence.

To investigate the computational complexity, we measured the FLOPs and the FPS of the state-of-the-art networks on an NVIDIA Titan Xp GPU, as shown in Table 5. Our EFSNet\(^1\) requires a computational budget of approximately 130M FLOPs, which is 2.46 times less than that of ENet [12]; additionally, it is faster.

### Table 4. Comparison to published entries for Cityscapes test set.

| Method | Frame work | Parameters | Fr | Rp | Rd | Sd | Building | Wall | Fence | Park | Train | Lane | Traffic Light | Veh | Sign | Person | Rider | Bike | Skis | Bus | Truck | Reserve | Car | Pedestrian | Rider | Veh \(\times\) | Cat. acc. |
|--------|------------|------------|----|----|----|----|----------|------|-------|------|-------|------|-----------|------|------|--------|-------|------|------|------|-------|---------|------|-----------|-------|----------|--------|
| SegNet [8] | ✔️ | 24.5 M | 11 | 74.96 | 71.20 | 83.89 | 74.43 | 65.23 | 55.10 | 57.72 | 54.10 | 57.14 | 65.74 | 63.71 | 61.76 | 83.61 | 82.71 | 82.31 | 80.61 | 78.53 | 75.78 | 75.34 | 60.45 | 61.94 | 79.15 | 81.36 |
| EFSNet\(^1\) | ✔️ | 67.35 M | 12 | 85.93 | 76.52 | 77.09 | 83.79 | 76.12 | 80.22 | 77.19 | 83.21 | 84.10 | 80.09 | 76.13 | 73.74 | 76.94 | 71.80 | 56.08 | 79.13 | 77.28 | 58.64 | 53.52 | 51.92 | 50.16 | 58.55 | 53.42 | 61.78 |
| EFSNet\(^2\) | ✔️ | 27.35 M | 12 | 86.68 | 77.02 | 86.82 | 80.72 | 84.41 | 79.02 | 84.11 | 79.33 | 75.64 | 85.38 | 83.83 | 82.38 | 80.41 | 83.74 | 83.29 | 79.32 | 80.70 | 79.07 | 78.45 | 73.42 | 71.23 | 74.20 | 73.25 | 71.10 |

### Table 5. FLOPs and inference speed (FPS) comparison with other semantic segmentation networks for the Cityscapes dataset.

| Network | FLOPs | Speed(FPS) |
|---------|-------|------------|
| PCN-6x [4] | 62.7 B | 12 |
| SegNet [8] | 31.2 B | 19 |
| ENet [12] | 362 M | 91 |
| ESPNet [16] | 431 M | 124 |
| ESCNet\(^1\) [37] | 265 M | 165 |
| ESCNet\(^2\) [37] | 431 M | 146 |
| EFSNet\(^1\) (ours) | 130 M | 128 |
| EFSNet\(^2\) (ours) | 268 M | 107 |
In addition, Model C achieves the highest mIoU of 60.7%. Compared with the ShuffleNet Unit, the CSDC module enlarges the receptive field and improves the segmentation accuracy, indicating the effectiveness of the CSDC module proposed in this paper. Model C has 10 categories with higher pixel accuracy than Model D, and mIoU and global pixel accuracy are 3.3% and 1.0% higher, respectively. Moreover, we find that class accuracy with significant improvement, such as sky, pole, sign, fence and car, are comparable to FSSNet. This shows that the CSDC module is at work. However, some classes, such as building, road, and bicyclist, have a lower pixel accuracy, which indicates a slight decrease using the CSDC module.

Table 6. Analysis of the effectiveness of the CSDC module. Parameters represents the number of the module parameters and the parameters of the other layers in the segmentation network. FLOPs represents the floating-point operations of the module. Here, b, w and c represent the input height, width and number of input channels, respectively; m denotes the bottleneck channels; and g is the number of groups in point-wise convolutions.

| Model     | Parameters | FLOPs |
|-----------|------------|-------|
| FSSNet [17] | -          | h(2c m + 9m^2) |
| Others    | 100 k      | -     |
| EFSNet^1 (ours) | CSDC       | 39 k  | h(2c m + 9m) |
| Others    | 54 k       | -     |
| EFSNet^2 (ours) | CSDC       | 76 k  | h(2c m + 9m) |
| Others    | 97 k       | -     |

2) COMPARISON OF CDB AND CSDC
To evaluate the effectiveness of our CSDC module, we conduct a comparative analysis of the CDB [17] and the CSDC module, as shown in Table 6. For comparison, we separately counted the parameters and FLOPs of the two modules, and the parameters of the remaining layers. Table 6 shows our CSDC module has fewer parameters and uses less FLOPs. Although CDB can effectively enlarge the receptive field without extra parameters, it still accounts for half of the total parameters because of the many convolution filters. Compared with CDB, the number of parameters for our CSDC is reduced by approximately 2.7 times, and the FLOPs can be reduced. Moreover, CSDC increases the depth of the network and has better segmentation performance compared to CDB with 6 convolutional layers.

The reason for the above difference is the group convolution and depth-wise convolution [18]. Group convolution first divides the inputs into several groups and then performs convolution, reducing the number of parameters and the FLOPs. Depth-wise convolution performs convolution on each channel of the inputs to reduce the number of parameters and the FLOPs. However, several group convolutional layers, simply stacked together, causes a drop in accuracy because the outputs only come from the feature map in a certain group [18]. A channel shuffle operation is adopted to solve this problem. To increase the receptive field, we use dilated convolutions with different dilation rates.

3) ENCODER NETWORK WITH CSDC MODULE
To rapidly validate the effectiveness of our CSDC module, we evaluated the segmentation accuracy of the encoder using the Cityscapes validation set, as shown in Table 7. By adding a point-wise convolution layer to the last layer of the encoder, and then training 500 epochs using the default parameters, we obtain the output feature map for C classes. We measured the inference speed by averaging 100 iterations for 1024 × 512 inputs on a single NVIDIA Titan Xp GPU. We also adopt various experiments with different groups (G), kernel size (K) and staking times (N) to obtain a good trade-off between accuracy performance and time consumption. For a fair comparison, the light-weight segmentation network ESPNet-C [16] and the encoder of the ESCNet-C [37] are used as baselines to compare parameters, mIoU and inference speed obtained using the Cityscapes validation set.

a: GROUPS (G)
The input is divided into G groups; the number of parameters can be reduced by increasing G. Inputs with G = 2 achieve better validation results compared to inputs with G = 4. Although the case with G = 2 uses 76.8 k parameters, it can obtain higher mIoU performance and similar inference speed. To make a good trade-off in accuracy and parameters, we fixed G = 2 to validate the effectiveness of other hyper-parameters.

b: KERNEL SIZE (K)
The K represents the kernel size of the depth-wise convolutions, which enables us to increase the receptive field of the convolution kernel by increasing K. To evaluate the effect of the kernel size K, we change the value of K from 3 to 5 and do not use dilated convolutions. Table 7 show that increasing the convolution kernel improves the segmentation accuracy, but increases the number of parameters and the inference time. We set K = 5 and do not use dilated convolution. Even though the kernel size is set as 5, the accuracy is 3.0% lower and the inference speed is lower than that of the module using a dilated convolution with kernel size of 3. In addition, the segmentation accuracy of the encoder with a 3 × 3 dilated convolution is improved by 6.9%.
c: DEPTH FACTOR (N)
The $N$ represents the depth of the CSDC module. For example, given the depth factor $N$, the CSDC module will be repeatedly stacked 2 times in the encoder. In other words, model capacity can be increased by adjusting the depth factor $N$ to increase the network depth. The encoder with $N = 1$ has fewer parameters than the encoder with $N = 2$; however, the accuracy is greatly reduced because of the limited model capacity.

Unlike ShuffleNet Unit, the CSDC module uses a dilated convolution with different dilation rates, thus lessening the gridding problem and enlarging the receptive field. Multiple CSDC modules in the encoder are used to increase the depth and receptive field of the network and obtain preferable inference speed. The data from the above ablation experiments show that the encoder network using the CSDC module achieves the highest accuracy at 57.1% on the Cityscapes validation set. More specifically, EFSNet\textsuperscript{2} achieves 54.3%
module is replaced with an up-sampling module in FSSNet. An asterisk indicates that the corresponding up-sampling module is replaced with an up-sampling module in FSSNet. A denotes the use of the up-sampling block in FSSNet in our network. The results show that the up-sampling module of FSSNet slightly increases the number of parameters and the forward propagation time. The up-sampling block of FSSNet is used in EFSNet to achieve similar accuracy, but it increases the number of parameters and the inference time. In addition, the use of an up-sampling block of FSSNet in EFSNet results in a drop in accuracy. The data show that the use of the up-sampling block in FSSNet in our network will reduce the mIoU.

### TABLE 8. Results on the Cityscapes validation set. We quantify the performance of the up-sampling block using the forward inference speed (FPS), the class average accuracy (C), and the mean intersection over union (mIoU). An asterisk indicates that the corresponding up-sampling module is replaced with an up-sampling module in FSSNet. A denotes an up-sampling block of FSSNet in our network and O denotes our up-sampling module.

| Network   | Up-sampling | Parameters | FPS (FPS) | G (%)  | C (%)  | mIoU (%) |
|-----------|-------------|------------|-----------|--------|--------|----------|
| EFSNet    | O           | 104 k      | 108       | 77.5   | 57.7   |
| EFSNet*   | O           | 186 k      | 96        | 76.8   | 60.3   |
| EFSNet†   | (ours)      | 93 k       | 128       | 75.9   | 57.9   |
| EFSNet‡   | (ours)      | 173 k      | 107       | 77.6   | 61.5   |

### TABLE 9. Results on the Camvid test set. We quantify the performance of the up-sampling block using the forward inference speed (FPS), the global accuracy (G), the class average accuracy (C), and the mean intersection over union (mIoU). An asterisk indicates that the corresponding up-sampling module is replaced with an up-sampling module in FSSNet. A denotes the up-sampling block of FSSNet in our network and O denotes our up-sampling module.

| Network   | Up-sampling | Parameters | FPS | G (%)  | C (%)  | mIoU (%) |
|-----------|-------------|------------|-----|--------|--------|----------|
| EFSNet    | O           | 104 k      | 344 | 84.1   | 54.5   | 58.2     |
| EFSNet*   | O           | 186 k      | 303 | 84.9   | 76.0   | 60.2     |
| EFSNet†   | (ours)      | 93 k       | 376 | 84.8   | 76.4   | 60.7     |
| EFSNet‡   | (ours)      | 173 k      | 352 | 85.6   | 76.6   | 61.1     |

Some visual segmentation images are shown in Fig. 9 and Fig. 10. As can be seen from the second row of pictures in Figure 9, our network cannot segment trucks and cars well due to the use of the up-sampling module of FSSNet. Fig. 10 shows that after using the up-sampling module of FSSNet, similar objects such as car and building cannot be segmented well in low-contrast pictures. At the same time, the prediction images have some noise.

### VI. CONCLUSION

In this paper, we propose the light-weight neural network EFSNet with a low number of parameters to perform efficient semantic segmentation. The method is evaluated on the CamVid and Cityscapes datasets, which demonstrates the method performs well in segmentation accuracy, especially in terms of calculation cost and inference time. Some ablation studies have shown that the CSDC module can enlarge the receptive field without additional parameters and improve segmentation accuracy. In addition, the up-sampling module can reduce the noise of segmentation and increase the forward propagation speed. Our method achieved a good trade-off between accuracy and efficiency, indicating that it can be applied when computing resources are limited.

### REFERENCES

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

[2] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[3] J. Zhang, X. Jin, J. Sun, J. Wang, and A. K. Sangaiah, “Spatial and semantic convolutional features for robust visual object tracking,” *Multimedia Tools Appl.*, vol. 2018, pp. 1–21, Aug. 2018.

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

[5] X. Shen, A. Hertzmann, J. Jia, S. Paris, B. Price, E. Shechtman, and I. Sachs, “Automatic portrait segmentation for image stylization,” *Comput. Graph. Forum*, vol. 35, no. 2, pp. 93–102, May 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention—MICCAI*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[7] B. Wu, A. Wan, X. Yue, and K. Keutzer, “SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud,” in *Proc. IEEE Int. Conf. Robot. Autom.* (ICRA), May 2018, pp. 1887–1893.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.

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

[11] J. Fu, J. Liu, H. Tian, Y. Li, Y. Yao, Z. Fang, and H. Lu, “Dual attention network for scene segmentation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3146–3154.

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

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–777.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2017, pp. 2818–2826.

[15] T. Wu, S. Tang, R. Zhang, and Y. Zhang, “CGNet: A light-weight context guided network for semantic segmentation,” 2018, arXiv:1811.08201.

[16] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, “ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 552–568.

[17] X. Zhang, Z. Chen, Q. M. J. Wu, L. Cai, D. Lu, and X. Li, “Fast semantic segmentation for scene perception.” *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1183–1192, Feb. 2019.

[18] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.
[19] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” 2015, arXiv:1511.07122. [Online]. Available: http://arxiv.org/abs/1511.07122

[20] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 834–848, Apr. 2018.

[21] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet v2: Practical guidelines for efficient CNN architecture design,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 116–131.

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861

[23] H. Zhao, X. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, “PSANet: Point-wise spatial attention network for scene parsing,” in Proc. Eur. Conf. Comput. Vis., Sep. 2018, pp. 270–286.

[24] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, “CCNet: Criss-cross attention for semantic segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1–10.

[25] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schubert, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler, and S. Hochreiter, “Speeding up semantic segmentation for autonomous driving,” in Proc. MLITS NIPS Workshop, 2016, p. 7.

[26] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size,” 2016, arXiv:1602.07360. [Online]. Available: http://arxiv.org/abs/1602.07360

[27] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “ICNet for real-time semantic segmentation on high-resolution images,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 405–420.

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

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

[30] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1520–1528.

[31] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” Pattern Recognit. Lett., vol. 30, no. 2, pp. 88–97, Jan. 2009.

[32] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 3213–3223.

[33] A. Paszke, S. Gross, S. Chintala, and G. Chanan. (Apr. 31, 2018). PyTorch: Tensors and Dynamic Neural Networks in Python With Strong GPU Acceleration. [Online]. Available: http://pytorch.org/

[34] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent. (ICLR), May 2015, pp. 1–15. [Online]. Available: http://arxiv.org/abs/1412.6980

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1026–1034.

[36] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, arXiv:1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167

[37] J. Kim and Y. S. Heo, “Efficient semantic segmentation using spatio-channel dilated convolutions,” IEEE Access, vol. 7, pp. 154239–154252, 2019.

XUEGANG HU received the B.S. degree in applied mathematics from China West Normal University, China, in 1988, and the M.S. and Ph.D. degrees in applied mathematics from Sichuan University, China, in 1995 and 2006, respectively. He is currently a Professor with the Chongqing University of Posts and Telecommunications, China. His research interests include digital image processing and analysis and partial differential equations and their applications.

HAIBO WANG was born in Nanchong, Sichuan, China, in 1995. He received the B.S. degree in electronic information engineering from Panzhihua University, Panzhihua, China, in 2018. He is currently pursuing the M.S. degree in electronics and communication engineering with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include computer vision, deep learning, and semantic segmentation.