LEDNET: A LIGHTWEIGHT ENCODER-DECODER NETWORK FOR REAL-TIME SEMANTIC SEGMENTATION

Yu Wang[1], *Quan Zhou*[1,2,*], *Jia Liu*[1], *Jian Xiong*[1], *Guangwei Gao*[3], *Xiaofu Wu*[1], *and Longin Jan Latecki*[4]

[1]National Engineering Research Center of Communications and Networking,
Nanjing University of Posts & Telecommunications, P.R. China.
[2]Key Lab. of Broadband Wireless Communications and Sensor Network Technology,
Nanjing University of Posts & Telecommunications, P.R. China.
[3]Institute of Advanced Technology, Nanjing University of Posts & Telecommunications, P.R. China.
[4]Department of Computer and Information Sciences, Temple University, Philadelphia, USA.

ABSTRACT

The extensive computational burden limits the usage of CNNs in mobile devices for dense estimation tasks. In this paper, we present a lightweight network to address this problem, namely *LEDNet*, which employs an *asymmetric* encoder-decoder architecture for the task of real-time semantic segmentation. More specifically, the encoder adopts a ResNet as backbone network, where two new operations, channel split and shuffle, are utilized in each residual block to greatly reduce computation cost while maintaining higher segmentation accuracy. On the other hand, an attention pyramid network (APN) is employed in the decoder to further lighten the entire network complexity. Our model has less than 1M parameters, and is able to run at over 71 FPS in a single GTX 1080Ti GPU. The comprehensive experiments demonstrate that our approach achieves state-of-the-art results in terms of speed and accuracy trade-off on CityScapes dataset.

Index Terms— CNN, Lightweight network, Encoder-decoder network, ResNet, Real-time semantic segmentation

1. INTRODUCTION

Recently, building deeper and larger convolutional neural networks (CNNs) is a primary trend for solving scene understanding tasks [1, 2, 3, 4, 5]. The most accurate CNNs usually have hundreds of convolutional layers and thousands of feature channels. In spite of achieving higher performance, these advances are at the sacrifice of running time and speed. Especially in the context of many real-world scenarios, such as augmented reality, robotics, and self-driving car to name of few, the computationally cheap networks with smaller size are often required to carry out online estimation in a timely fashion. Therefore, those accurate networks requiring enormous resources are not suitable for computationally limited mobile platforms (e.g., drones, robots, and smartphones), which have limited energy overhead, restrictive memory constraints, and reduced computational capabilities. Such kind of limitation is particularly prominent on the computationally heavy task of semantic segmentation [4, 5, 6, 7, 8], where the goal here is to assign a semantic category label for each image pixel.

In order to overcome this problem, many lightweight style networks have been designed to balance the segmentation accuracy and implementing efficiency, which are roughly divided into two categories: network compression [9, 10, 11, 12] and convolution factorization [13, 14, 15]. The first category prefers to reduce inference computation by compressing pre-trained networks, including hashing [9], pruning [10], and quantization [11, 12]. To further remove the redundancy, an alternative approach to lighten CNNs depends on sparse coding theory [16, 17]. On the contrary, motivated from the convolution factorization principle (CFP) that decomposes a standard convolution into group convolution and depthwise separable convolution [3, 15, 18], the second category focuses on directly training network with smaller size. For example, ENet [13] employs ResNet [2] as backbone to perform efficient inference. Zhao *et al.* [19] propose a cascade network that incorporates high-level label guidance to improve performance. In [7, 14, 20], a symmetrical encoder-decoder architecture is adopted, which greatly reduce the number of parameters while maintaining the accuracy. Although some work have conducted preliminary research on lightweight architecture networks, pursuing the best accuracy in very limited computational budgets is still an open research question for the task of real-time semantic segmentation.

In this paper, we aim at solving this trade-off as a whole, without sitting on only one of its sides. We introduce a novel lightweight network called *LEDNet*, adopting an *asymmetric* encoder-decoder architecture for real-time semantic segmentation. As shown in Figure 1, our LEDNet is composed of two parts: encoder and decoder network. Following CFP, the core

*Corresponding author: Quan Zhou, quan.zhou@njupt.edu.cn. This work is partly supported by NSFC (No. 61876093, 61701258, 61701252, 61671253), NSFJS (No. BK20181393, BK20170906), NSF (No. IIS-1302164), and Huawei Innovation Research Program (HIRP2018).

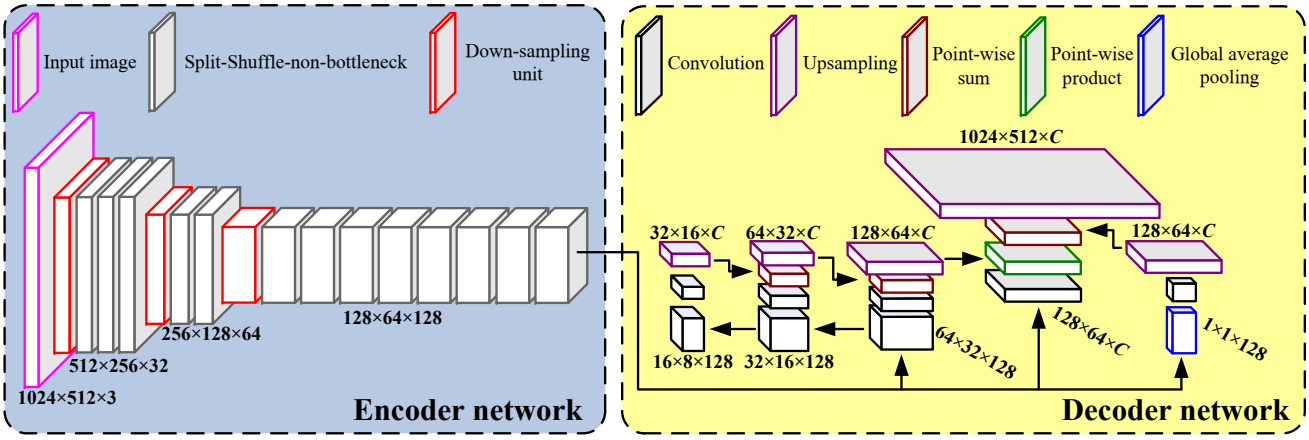

Fig. 1. Overall asymmetric architecture of the proposed LEDNet. The encoder employs a FCN-like network, while an APN is adopted in decoder. C denotes the number of classes. Please refer to text for more details. (Best viewed in color)

unit of encoder is a novel residual module that leverages skip connections and convolutions with channel split and shuffle. While the skip connections allow the convolutions to learn residual functions that facilitate training, the split and shuffle operations enhance the information exchange within the feature channels while maintaining similar computational costs compared to 1D factorized convolutions. In the decoder, instead of complicated dilated convolution [20], we design an attention pyramid network (APN) to extract dense features, where the attention mechanism is utilized to estimate semantic label for each pixel. Our contributions are three-folds: (1) The asymmetrical architecture of our LEDNet leads to the great reduction of network parameters, which accelerates the inference process; (2) The channel split and shuffle operations in our residual layer leverage network size and powerful feature representation. In addition, channel shuffle is also differentiable, which means it can be embedded into network structures for end-to-end training. (3) Attention mechanism of feature pyramid is employed to design APN in our decoder-end, further lightening the complexity of the whole network.

2. OUR APPROACH

2.1. Residual Module with Split and Shuffle Operations

We focus on solving the efficiency limitation that is essentially present in the residual block, which is used in recent accurate CNNs for image classification [2, 21, 22] and semantic segmentation [6, 13, 14]. The recent years have witnessed multiple successful instances of lightweight residual layer [13, 15], such as bottleneck (Figure 2 (a)), non-bottleneck-1D (Figure 2 (b)), and ShuffleNet module (Figure 2 (c)), where the pointwise convolution is widely used. However, the contrary opinion of [21] claims that pointwise convolution accounts for most of the computational complexity, which is especially disadvantageous for lightweight models.

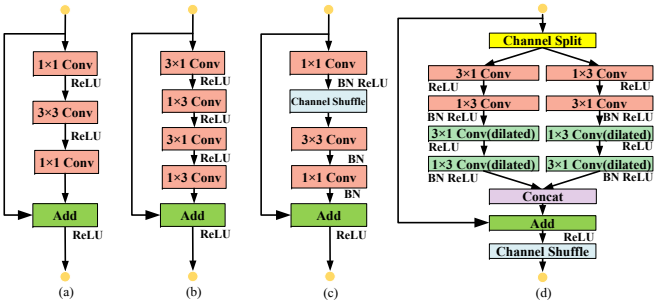

Fig. 2. Comparison of different residual layer modules. From left to right are (a) bottleneck [13, 15], (b) non-bottleneck-1D [14], (c) ShuffleNet [21], and (d) our SS-nbt module.

To balance performance and efficiency given limited computational budgets, we introduce two simple operators, called channel split and shuffle, in residual layer. We refer to this proposed module as split-shuffle-non-bottleneck (SS-nbt), as depicted in Figure 2 (d). Motivated from [12, 18], a *split-transform-merge* strategy is employed in the designment of our SS-nbt, approaching the representational power of large and dense layers, but at a considerably lower computational complexity. At the beginning of each SS-nbt, the input is split into two lower-dimensional branches, where each one has half channels of the input. To avoid pointwise convolution, the transformation is performed using a set of specialized 1D filters (e.g., 1×3, 3×1), and the convolutional outputs of two branches are merged using concatenation so that the number of channels keeps the same. To facilitate training, the stacked output is added with input through the branch of identity mapping. The same channel shuffle operation [21] is finally used to enable information communication between two split branches. After the shuffle, the next SS-nbt unit begins. It is clear that our residual module is not only efficient,

Table 1. The architecture of LEDNet. "Size" denotes the dimension of output feature maps, C is the number of classes.

Stage	Type	Size
Encoder	**Downsampling Unit**	$512 \times 256 \times 32$
	$3\times$ **SS-nbt Unit**	$512 \times 256 \times 32$
	Downsampling Unit	$256 \times 128 \times 64$
	$2\times$ **SS-nbt Unit**	$256 \times 128 \times 64$
	Downsampling Unit	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 1$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 2$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 5$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 9$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 2$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 5$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 9$)	$128 \times 64 \times 128$
	SS-nbt Unit (dilated $r = 17$)	$128 \times 64 \times 128$
Decoder	**APN Module**	$128 \times 64 \times C$
	Upsampling Unit ($\times 8$)	$1024 \times 512 \times C$

but also accurate. Firstly, the high efficiency in each SS-nbt allows us to use more feature channels. Secondly, in each SS-nbt unit, the merged feature channels are randomly shuffled, and then join into next unit. This can be regarded as a kind of feature reuse, which to some extent enlarges network capacity without significantly increasing complexity.

2.2. LEDNet Architecture Designment

As shown in Table 1, our LEDNet follows an encoder-decoder architecture. Unlike [7], our approach employs an asymmetric sequential architecture, where a encoder produces downsampled feature maps, and a subsequent decoder adopts APN that upsamples the feature maps to match input resolution.

Besides SS-nbt unit, the encoder also includes downsampling unit, which is performed by stacking two parallel outputs of a single 3×3 convolution with stride 2 and a Max-pooling. Downsampling enables more deeper network to gather context, while at the same time helps to reduce computation. Note we postpone downsampling in encoder, in the similar spirit of [18]. Moreover, the usage of dilated convolutions [14, 23] allows our architecture to have large receptive field, leading to an improvement in accuracy. Compared to the use of larger kernel sizes, this technique has been proven more effective in terms of computational cost and parameters.

Inspired by attention mechanism [24], our decoder designs a APN to perform dense estimation using spatial-wise attention. To increase receptive field, the APN adopts a pyramid attention module, which integrates features from three different pyramid scales. As shown in Figure 1, we first utilize 3×3, 5×5, and 7×7 convolution with stride 2 to form multi-scale feature pyramid. Then the pyramid structure fuses information of different scales step-by-step, which can incorporate neighbor scales of context more precisely. Since high-level feature maps has small resolution, using large kernel size does not bring too much computation burden. Thereafter, a 1×1 convolution is applied to the output of encoder, then the convolutional feature maps are pixel-wisely multiplied by the pyramid attention features. To further enhance performance, a global average pooling branch is introduced to integrate global context prior attention. Finally, an upsampling unit is employed to match the resolution of input image. Benefiting from pyramid architecture, APN can capture multi-scale context cues, and produce pixel-level attention for convolutional features. Unlike DeepLab [5] and PSPNet [8] that stack multi-scale feature maps, our context information is pixel-wisely multiplied with original convolutional features, without introducing too much computational budgets.

Table 2. Comparison with the state-of-the-art approaches in terms of segmentation accuracy and implementing efficiency.

Method	Cla	Cat	Time(ms)	Speed(Fps)	Para(M)
SegNet[7]	57.0	79.1	67	15	29.5
ENet[13]	58.3	80.4	34	31	**0.36**
ESPNet[20]	60.3	82.2	**9**	**112**	0.40
CGNet[25]	64.8	85.7	20	50	0.50
ICNet [19]	69.5	86.4	33	30	7.80
Ours	**70.6**	**87.1**	14	71	0.94

3. EXPERIMENTS

3.1. Implementation Details

We select widely-used CityScapes dataset [26] to evaluate our LEDNet, which includes 19 object categories and one additional background. Beside the images with fine pixel-level annotations that contain 2,975 training, 500 validation and 1,525 testing images, we also use the 20K coarsely annotated images for training. We adopt mean intersection-over-union (mIOU) averaged across all classes and categories to evaluate segmentation accuracy, while running time, speed (FPS), and model size (number of parameters) to measure implementing efficiency. To show the advantages of LEDNet, we selected 6 state-of-the-art lightweight networks as baselines, including SegNet [7], ENet [13], ERFNet [14], ICNet [19], CGNet [25], and ESPNet [20]. For fair comparison, all the methods are conducted on the same hardware platform of Dell workstation with a single GTX 1080Ti GPU. We favor a large minibatch size (set as 5) to make full use of the GPU memory, where the initial learning rate is 5×10^{-4} and the 'poly' learning rate policy is adopted with power 0.9, together with momentum and weight decay are set to 0.9 and 10^{-4}, respectively.

Table 3. Individual category results on the CityScapes test set in terms of class and category mIOU scores. Methods trained using both fine and coarse data are marked with superscript '†'.

Method	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter	Sky	Ped	Rid	Car	Tru	Bus	Tra	Mot	Bic	Cla	Cat
SegNet [7]	96.4	73.2	84.0	28.4	29.0	35.7	39.8	45.1	87.0	63.8	91.8	62.8	42.8	89.3	38.1	43.1	44.1	35.8	51.9	57.0	79.1
ENet [13]	96.3	74.2	75.0	32.2	33.2	43.4	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.3	80.4
ESPNet [20]	97.0	77.5	76.2	35.0	36.1	45.0	35.6	46.3	90.8	63.2	92.6	67.0	40.9	92.3	38.1	52.5	50.1	41.8	57.2	60.3	82.2
CGNet [25]	95.5	78.7	88.1	40.0	43.0	54.1	59.8	63.9	89.6	67.6	92.9	74.9	54.9	90.2	44.1	59.5	25.2	47.3	60.2	64.8	85.7
ERFNet [14]	97.2	**80.0**	89.5	41.6	45.3	56.4	60.5	64.6	91.4	**68.7**	94.2	76.1	**56.4**	92.4	45.7	60.6	27.0	48.7	61.8	66.3	85.2
ICNet [19]	97.1	79.2	89.7	43.2	48.9	61.5	60.4	63.4	91.5	68.3	93.5	74.6	56.1	**92.6**	51.3	**72.7**	51.3	**53.6**	70.5	69.5	86.4
Ours	97.1	78.6	90.4	46.5	48.1	60.9	60.4	71.1	91.2	60.0	93.2	74.3	51.8	92.3	61.0	72.4	51.0	43.3	70.2	69.2	86.8
Ours†	**98.1**	79.5	**91.6**	**47.7**	**49.9**	**62.8**	**61.3**	**72.8**	**92.6**	61.2	**94.9**	**76.2**	53.7	90.9	**64.4**	64.0	**52.7**	44.4	**71.6**	**70.6**	**87.1**

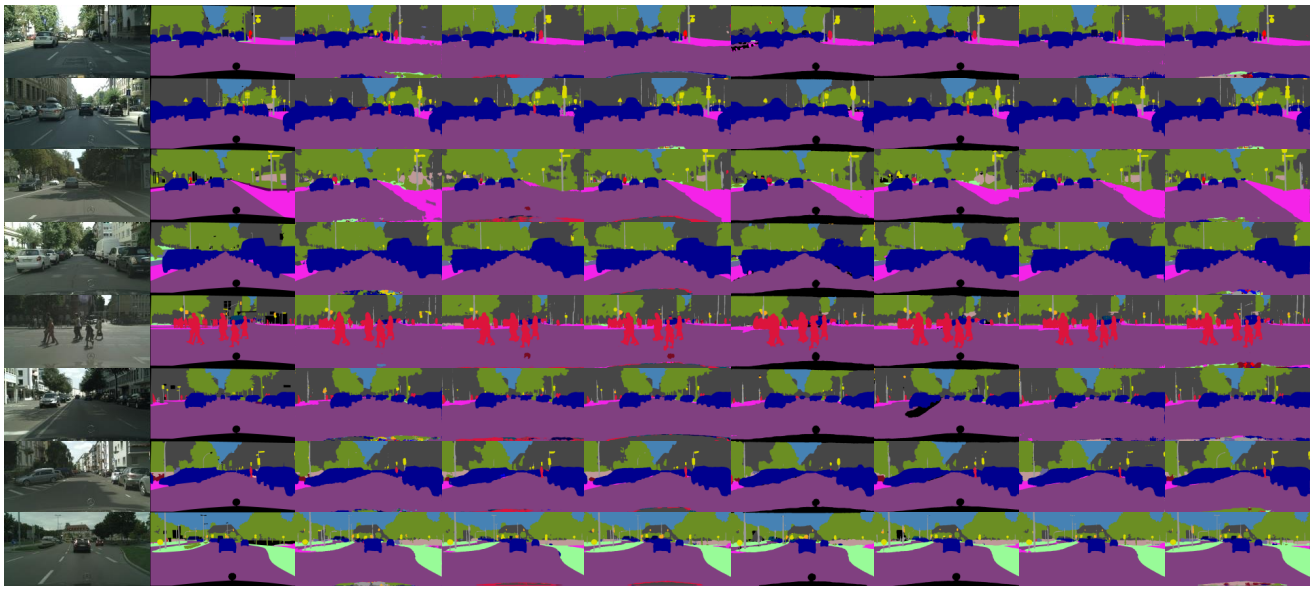

Fig. 3. The visual comparison on CityScapes val dataset. From left to right are input images, ground truth, segmentation outputs from SegNet [7], ENet [13], ERFNet [14], ESPNet [20], ICNet [19], CGNet [25], and our LEDNet. (Best viewed in color)

3.2. Evaluation Results

Table 2 and Table 3 report comparison results, demonstrating that LEDNet achieves the best available trade-off in terms of accuracy and efficiency. Among all the approaches, our LEDNet yields 70.6% class mIOU and 87.1% category mIOU, respectively, where 13 out of the 19 object categories obtains best scores. Regarding to the efficiency, LEDNet is nearly $5\times$ faster and $30\times$ smaller than SegNet [7]. Although ENet [13], another efficient network, is $1.5\times$ efficient, and has $3\times$ less parameters, but delivers poor segmentation accuracy of 10% drop than our LEDNet. Figure 3 shows some visual examples of segmentation outputs on the CityScapes validation set. It is demonstrated that, compared with baselines, our LEDNet not only correctly classifies object with different scales, but also produces consistent qualitative results for all classes.

4. CONCLUSION AND FUTURE WORK

This paper has described a LEDNet model, which designs an asymmetric encoder-decoder architecture for real-time semantic segmentation. The encoder adopts channel split and shuffle operations in residual layer, enhancing information communication in the manner of feature reuse. On the other hand, the decoder employs a APN, where the spatial pyramid structure is beneficial to enlarge receptive fields without introducing significant computational budgets. The entire network is trained end-to-end. The experimental results show our LEDNet achieves best trade-off on CityScapes dataset in terms of segmentation accuracy and implementing efficiency. The future work includes decomposing standard convolution in APN into 1D convolution, resulting in further lightweight network while still remaining segmentation accuracy.

5. REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014, pp. 580–587.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[4] L. Jonathan, S. Evan, and D. Trevor, "Fully convolutional networks for semantic segmentation," *IEEE TPAMI*, vol. 39, no. 4, pp. 640–651, 2017.

[5] 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 TPAMI*, vol. 40, no. 4, pp. 834–848, 2018.

[6] L. Guosheng, M. Anton, S. Chunhua, and I. Reid, "Refinenet: multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, 2017, pp. 5168–5177.

[7] B. Vijay, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE TPAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Y. Jia, "Pyramid scene parsing network," in *CVPR*, 2016, pp. 6230–6239.

[9] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *ICML*, 2015.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[11] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *CVPR*, 2016, pp. 5168–5177.

[12] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.

[13] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," in *arXiv preprint arXiv:1606.02147*, 2016.

[14] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE TITS*, vol. 19, no. 1, pp. 263–272, 2018.

[15] 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," in *arXiv preprint arXiv:1704.04861*, 2017.

[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *NIPS*, 2016, pp. 2074–2082.

[17] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *CVPR*, 2015, pp. 806–814.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016, pp. 2818–2826.

[19] H. S. Zhao, X. J. Qi, X. Y. Shen, J. P. Shi, and J. Y. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *arXiv preprint arXiv:1704.08545v2*, 2018.

[20] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *arXiv preprint arXiv:1803.06815v3*, 2018.

[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, 2018, pp. 6848–6856.

[22] X. Xie, R. Girshick, P. Dollar, Z. W. Tu, and K. M. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017, pp. 5987–5995.

[23] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *arXiv preprint arXiv:1709.01507*, 2017.

[25] T. Y. Wu, S. Tang, R. Zhang, and Y. D. Zhang, "Cgnet: A light-weight context guided network for semantic segmentation," in *arXiv preprint arXiv:1811.08201v1*, 2018.

[26] 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 *CVPR*, 2016, pp. 3213–3223.