Efficient Dense Modules of Asymmetric Convolution for Real-Time Semantic Segmentation

Shao-Yuan Lo[1] Hsueh-Ming Hang[1] Sheng-Wei Chan[2] Jing-Jhih Lin[2]
[1] National Chiao Tung University [2] Industrial Technology Research Institute
sylo95.eecs02@g2.nctu.edu.tw, hmhang@nctu.edu.tw, {ShengWeiChan, jeromelin}@itri.org.tw

Abstract

Real-time semantic segmentation plays an important role in practical applications such as self-driving and robots. Most semantic segmentation research work focuses on improving estimation accuracy with little consideration on the efficiency. Several previous studies that emphasize high-speed inference often cannot produce high-accuracy segmentation results. In this paper, we propose a novel convolutional network named Efficient Dense modules with Asymmetric convolution (EDANet), which employs an asymmetric convolution structure and incorporates the dilated convolution and the dense connectivity to achieve high efficiency at low computational cost and model size. EDANet is 2.7 times faster than the existing fast segmentation network ICNet, while it achieves a similar mIoU score without any additional context module, post-processing scheme, and pretrained model. We evaluate EDANet on Cityscapes and CamVid datasets and compare it with the other state-of-art systems. Our network can run with the high-resolution inputs at the speed of 108 FPS on a single GTX 1080Ti card.

1. Introduction

Semantic segmentation is an essential area in computer vision. It performs pixel-level label prediction for images. In recent years, the development of deep convolutional neural networks (CNNs) has made notable progress in providing accurate segmentation results [4, 18, 34]. The achievements of these networks mainly rely on their complicated model designs, which consist of considerable depth and width, and need a huge number of parameters and long inference time. However, recent interests in many real-world applications, such as autonomous driving, augmented reality, robotic interaction, and intelligent surveillance, has generated a high demand for the scene understanding systems that are able to operate in real-time. Thus, it is paramount to develop effective convolutional networks for real-time semantic segmentation.

Project page: https://github.com/shaoyuanlo/EDANet

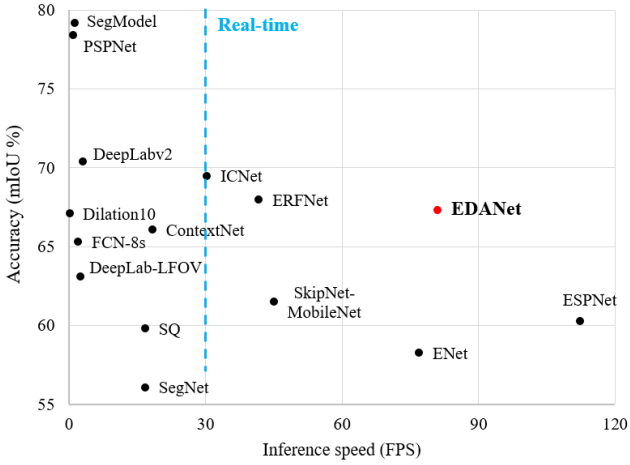

Figure 1: Inference speed and mIoU accuracy on Cityscapes test set [7]. The speeds are measured on one Titan X card. The networks included are SegNet [1], DeepLab [3, 4], FCN [18], ESPNet [19], ENet [21], ContextNet [22], ERFNet [23], SegModel [25], SkipNet-MobileNet [26], SQ [30], Dilation10 [32], PSPNet [33], ICNet [34], and our EDANet.

The challenge of designing neural networks by taking both efficiency and reliability into consideration can be seen in Figure 1. For example, most of the top performing methods, such as PSPNet [33] and SegModel [25], focus on improving accuracy at the expense of large increases in computational cost. Therefore, in Figure 1, these methods are located near the area of high accuracy (measured by the mean of intersection-over-union, mIoU) and low inference speed (frames per second, FPS). On the other hand, some approaches, such as ENet [21] and ESPNet [19], emphasize on the speed but their accuracy drops notably. They are located at the bottom right in Figure 1.

In this paper, we propose a new network architecture, Efficient Dense modules with Asymmetric convolution (EDANet) that simultaneously accomplishes high efficiency and accuracy. Our method is not only among the few systems whose inference speed exceeds 30 FPS (real-time) and are located on the upper right region of Figure 1.

One important feature of EDANet is asymmetric convolution. It decomposes a standard two-dimensional convolution into two one-dimensional convolutions. That is,

an original $n \times n$ convolution kernel is factorized into two convolution kernels, $n \times 1$ and $1 \times n$, respectively. This technique can dramatically reduce the number of parameters with little performance degradation. We take the essence of the densely connected structure [13] and modify it for real-time semantic segmentation. Although DenseNet was initially created for image classification challenges, our experiments show that its capability of gathering the features extracted from different layers and aggregating the multi-scale information is innately beneficial to segmentation tasks. This structure can also substantially reduce the number of parameters. The dilated convolution is also employed by EDANet. The idea is enlarging the receptive field of our network through this type of convolution in order to retain the feature map resolution and avoid losing spatial information. To achieve a good balance in efficiency and reliability, we do not add any extra decoder structure, context module, and post-processing scheme into our system. We further build several types of EDANet variants to evaluate the performance of different network design choices.

In summary, there are three main contributions in this study:

- We develop a novel network named EDANet, which incorporates asymmetric convolution with dilated convolution and dense connectivity. It can run on high-resolution images at 108 FPS on a single GPU and achieve 67.3% mIoU on the Cityscapes dataset [7].
- The proposed EDANet is nearly 3 times faster than ICNet [34] and attains comparable performance; it achieves this without any extra decoder structure, context module, post-processing scheme, and pretrained model.
- We design various types of EDANet variants to analyze the performance of different network architectures and analyze the reasons behind the results.

2. Related work

Originally, CNNs were created for image classification tasks [17], which predicts a single category for each input image. FCN [18] is a pioneering CNN in semantic segmentation. It adapts VGG16 [27] by replacing fully-connected layers by convolution layers to perform pixel-level label prediction. Since the development of FCN, the semantic segmentation research entered the era of CNN-based methods.

High accuracy networks. U-Net [24] develops an encoder-decoder architecture to collect spatial information from the shallower layers to enhance the features in the deeper layers. DeconvNet [20] proposes a decoder, which is symmetric to its encoder, to upsample the outputs of the encoder. These networks have a huge computational cost due to their heavy decoders. Dilation10 [32] creates a context module by stacking the dilated convolution layers with increasing dilation rates for aggregating multi-scale contextual information. DeepLab [4, 5, 6] introduces an atrous spatial pyramid pooling (ASPP) module employing multiple parallel filters with different dilation rates to exploit multi-scale representations. Both modules require enormous computation and inference time. As a result, although the aforementioned networks are accurate, they are not feasible for practical applications.

High inference speed networks. ENet [21] is one of the first networks aiming at semantic segmentation in real-time. It adapts the ResNet structure [11] but trims the number of convolution filters to reduce computation. ESPNet [19] designs an efficient spatial pyramid (ESP) module that uses point-wise convolution in front of the spatial pyramids to reduce computational cost. These two networks improve efficiency greatly but significantly sacrifice accuracy. Recent studies such as ICNet [34] and BiSeNet [31] make a better balance between speed and performance, but there is still a big room for further improvement.

Densely connected networks. DenseNet [13] achieves excellent performance on image classification challenges. It is based on the densely connected structure that each layer is directly connected to every other layer in a feed-forward manner. Some studies have extended DenseNet to do semantic segmentation tasks. FC-DenseNet [15] uses DenseNet as the encoder and adds a decoder structure based on the conventional skip connections [24] to build the fully convolutional DenseNet. SDN [10] takes DenseNet as their backbone model and combines it with the stacked deconvolutional architecture. These methods simply adopt DenseNet without extensive optimization, and the added complexity in their design further increases the computational cost of the entire networks.

In this paper, our EDANet adopts the asymmetric convolution structure for reducing the number of parameters and computational cost. We also adopt the idea of the dense connectivity in constructing our network. EDANet is able to achieve remarkable inference speed and retain the high accuracy at the same time.

3. Method

The architecture of the proposed EDANet is shown in Figure 2. It consists of three downsampling blocks, two EDA blocks, and a projection layer. The first and the second EDA block is composed of 5 and 8 densely connected EDA modules respectively. EDANet does not include any additional decoder, context module and post-processing scheme.

In this section, we first describe the core EDA module, and then elaborate on the other important network design choices.

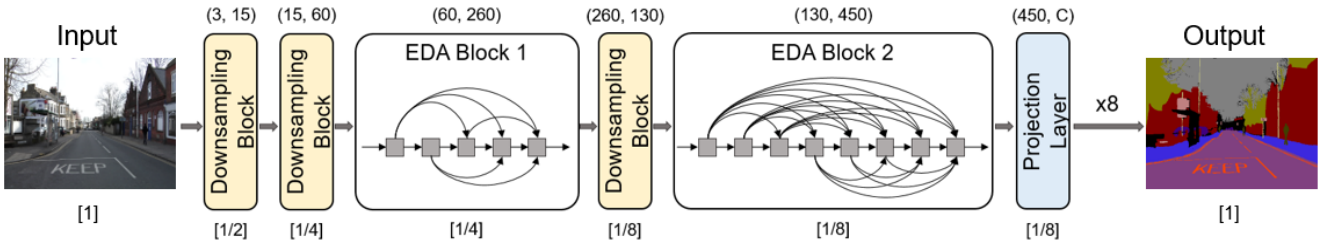

Figure 2: The proposed EDANet architecture. The numbers of input and output channels of each block are marked in parentheses. The numbers in brackets are output feature size ratios to the full-resolution input images. "C": the number of object classes.

3.1. EDA module

The EDA module is the core of the entire EDANet. Its structure is based on the dense module of asymmetric convolution, as shown in Figure 3. It consists of a point-wise convolution layer and two pairs of asymmetric convolution layers. The output of each EDA module is the concatenation of its input and the newly produced feature maps. Below we discuss each component in the proposed EDA module.

Point-wise convolution layer. The point-wise convolution layer is a 1×1 convolution at the beginning of each EDA module, which is used to reduce the number of input channels [11]. This design can dramatically decrease the number of parameters and computational complexity.

Asymmetric convolution. The asymmetric convolution is to factorize a standard two-dimensional convolution kernel into two one-dimension convolution kernels. In other words, an $n\times 1$ convolution followed by a $1\times n$ convolution can substitute for an $n\times n$ convolution [23, 29]. This mechanism can be expressed as:

$$\sum_{i=-M}^{M}\sum_{j=-N}^{N} W(i,j)I(x-i,y-j) = \sum_{i=-M}^{M} W_x(i)\left[\sum_{j=-N}^{N} W_y(j)I(x-i,y-j)\right] \quad (1)$$

where I is a 2D image, W is a 2D kernel, W_x is a 1D kernel along x-dimension, and W_y is a 1D kernel along y-dimension. When the kernel size is 3, the number of parameters and computational cost can be saved significantly by 33%, and the performance degradation is often very small.

Dilated convolution. The dilated convolution is a particular type of convolution, which inserts zeros between two consecutive kernel values along each dimension [3, 33] and can be defined as:

$$O(i,j) = \sum_{i=-M}^{M}\sum_{j=-N}^{N} W(i,j)I(x-i\cdot r, y-j\cdot r) \quad (2)$$

where I is an input image, O is a output image, W is a convolution kernel, and r is a dilation rate. This type of convolution is able to enlarge the effective receptive field of kernels without increasing the number of parameters. For instance, the effective size of an $n\times n$ convolution kernel with dilation rate r is equal to $[r(n\text{-}1)+1]\times[r(n\text{-}1)+1]$.

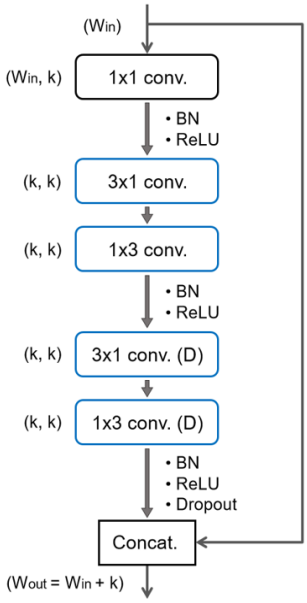

Figure 3: The proposed EDA module structure. "(D)": possible dilated convolution. "BN": batch normalization. The numbers of input and output channels of each layer are marked in parentheses. "k": growth rate, we set it to 40 in our EDANet.

For aggregating more contextual information to improve accuracy, we employ the dilated convolution at the second asymmetric convolution pair in the EDA modules to form the dilated EDA modules and thus is called "dilated asymmetric convolution". The last two EDA modules in EDA block 1 and all the eight EDA modules in EDA block 2 are the dilated EDA modules. The dilation rates in the system are 2, 2, 2, 2, 4, 4, 8, 8, 16, and 16, respectively. We choose this sequential placement for enlarging the receptive field in a gradual manner.

Dense connectivity. The dense connectivity was proprosed by DenseNet [13], in which each layer takes all preceding feature maps as its input. We adopt this strategy in the proposed EDANet, in which each module concatenates its input and the new learned features together to form the final output:

$$y_m = [H_m(y_{m-1}), y_{m-1}] \quad (3)$$

where m indicates the m^{th} module, H is the composite function of the module, and y is the final output, which is the concatenation of its input and the output of the composite function.

This densely connected structure can substantially increase processing efficiency because each module is only responsible for acquiring a few new features. Furthermore, it is well-known that the deeper layers have larger receptive fields [27]. For example, a stack of two 3×3 convolution layers has the same effective receptive field as a single 5×5 convolution layer, and three such layers have an effective receptive field of 7×7. Thus, the dense connectivity concatenating the features learned from each module that has a different receptive field individually, allows our network to naturally gather multi-scale information together. This enables our system to have a good semantic segmentation results at low computational cost.

3.2. Network design choices

In this subsection, we discuss the other crucial design choices on the downsampling, decoder, and composite function.

Downsampling. We adopt the ENet [21] initial block as our downsampling block. The structure is shown in Figure 4. ENet uses the initial block to do the first downsampling, but we apply it to all the downsampling layers, and further extend it to two modes. When the number of output channels W_{out} of a block is less than the number of input channels W_{in}, this block is simply a single 3×3 convolution layer with stride 2, and $W_{conv} = W_{out}$. Our third downsampling block that divides the network into two EDA blocks adopts this mode (see Figure 2). If $W_{out} > W_{in}$, a 2×2 max-pooling layer with stride 2 would be included, and then the concatenation of the features from the convolution and the max-pooling branches forms the final output. In this mode, $W_{conv} = W_{out} - W_{in}$. The first two downsampling blocks adopt this mode (see Figure 2). This two-branch design saves the computation of the convolution layers.

The downsampled feature maps enable the networks to have a larger receptive field for collecting more contextual information. However, reducing feature map resolution would lose spatial details, which is especially harmful to the pixel-wise segmentation. In order to address this problem, we find a balanced structure that contains only three downsampling operations in our network. The ratio of the feature size at the end of EDANet to the full-resolution input images is 1/8. Compared to other networks such as SegNet [1], whose ratio of feature map size to inputs is 1/32, EDANet can remain more spatial details during feature extraction. To compensate for the receptive field, we use the dilated convolution in many EDA modules to effectively achieve this goal.

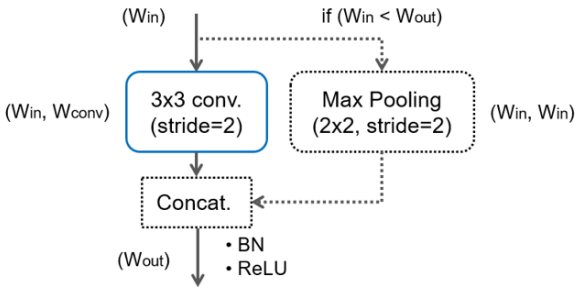

Figure 4: Downsampling block structure. "BN": batch normalization. The numbers of input and output channels of each layer are marked in parentheses.

Decoding. Many state-of-art systems use a decoder to upsample feature maps at the expense of huge computation [1, 20]. Even choosing a relatively small decoder would still increase the computational cost [23]. Since EDANet aims at fast semantic segmentation, we discard the decoder structure in our design. After EDA block 2, we add a 1×1 convolution layer as a projection layer to output C (the number of classes) feature maps, then use the bilinear interpolation to upsample the feature maps by a factor of 8 to the size of the full-resolution input images (see Figure 2). This strategy reduces the accuracy only slightly but saves a lot of computational cost.

Composite function. In our network, in order to accelerate the actual inference speed, we choose the traditional wisdom of post-activation composite function instead of pre-activation [11]. To be more specific, the sequence of three successive operations is a convolution, followed by batch normalization [14] and ReLU. We apply this structure to all the convolution layers, as shown in Figures 3 and 4, except for the last projection layer. The advantage is that each batch normalization layer can be merged with its preceding convolution layer during inference, which is able to decrease the inference time. Also, in the training phase, we place a dropout layer [28] between the last ReLU and the concatenation of each module as a regularization measure (see Figure 3). We set the dropout rate to 0.02 in our networks.

4. Experiments

We evaluate our method on two challenging datasets, Cityscapes [7] and CamVid [2]. In this section, we first describe the datasets and our training setup. Then, we conduct a series of experiments to examine the proposed network. Finally, we report the comparisons with the other state-of-art systems.

Cityscapes. The Cityscapes dataset is an urban street scene dataset that contains 19 object classes. It consists 5000 fine-annotated images at the high-resolution of 1024×2048, which are split into three sets: 2975 images for training, 500

images for validation, and 1525 images for testing. There is another set of 19,998 images with coarse annotation, but we only use the fine annotation set for all experiments. Our network is trained and tested on the downsampled 512×1024 inputs, but for evaluation, the output feature maps are upsampled by bilinear interpolation to the original dataset resolution.

CamVid. The CamVid dataset is another dataset for vehicle applications, which consists of 367 training and 233 testing images. It includes 11 classes and has resolution of 360×480.

Training. We train our networks by using the Adam optimization [16] with a weight decay of 0.0001 and a batch size of 10. We employ the poly learning rate policy, where the learning rate is multiplied by $(1 - iter/\max\_iter)^{power}$ with power 0.9 and initial learning rate 0.0005. Inspired by ENet [21], we use the class weighting scheme defined by $w_{class} = 1/log(p_{class} + k)$, where we set k to 1.12. We include data augmentation in training for both Cityscapes and CamVid by using random horizontal flip and the translation of 0~2 pixels on both axes. All the reported accuracy results are measured in the mIoU metric.

4.1. Ablation study

In this subsection, we perform a series of experiments to validate the potential of our network. All the following experiments are evaluated on the Cityscapes dataset.

Core module. The asymmetric convolution structure and the dense connection concept [13] are two key elements in the proposed EDA module. In order to further investigate potential improvements, we design two variants of our module for comparisons.

The first one is a "non-asymmetric" variant that replaces the two pairs of asymmetric convolution by two standard 3×3 convolution layers (see Figure 5a). The other one is a "non-dense" variant, which employs the conventional residual connection [11] instead of the dense connection, and removes the point-wise convolution layer (see Figure 5b). This variant is the same as the ERF module [23]. In order to make comparison at the same computational cost, we set its width W (the number of feature maps) to 40 in block 1 and 80 in block 2 (64 and 128 in ERFNet respectively). We use the same layer placement as EDANet to build two networks composed of the two module variants respectively, and they are called EDA-non-asym and EDA-non-dense.

As shown in Table 1a, EDA-non-asym obtains almost the same accuracy as EDANet but has 27% more computational cost. This result indicates the advantage of our asymmetric convolution design. On the other hand, EDA-non-dense performs 1.18% lower accuracy than EDANet. Apparently, the densely connectivity is effective.

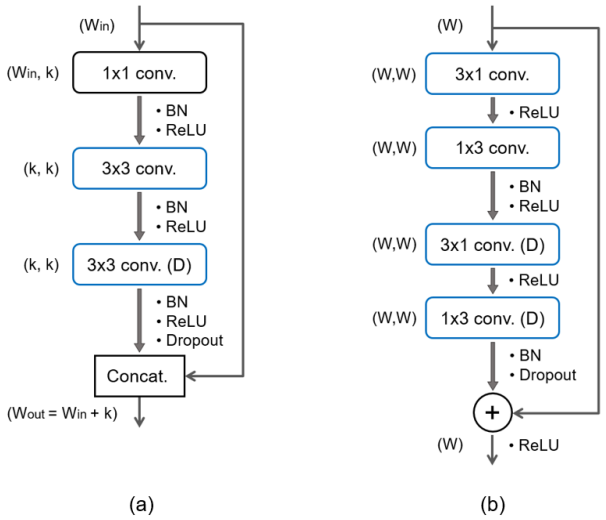

Figure 5: (a) "non-asymmetric" module variant. (b) "non-dense" module variant. "(D)": possible dilated convolution. "BN": batch normalization. The numbers of input and output channels of each layer are marked in parentheses. "k": growth rate, we set it to 40 in EDA-non-aysm.

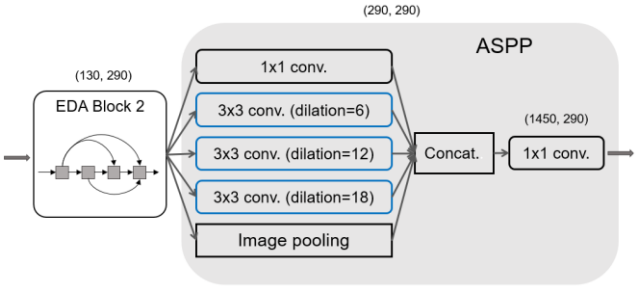

Figure 6: A part of EDANet-ASPP structure. The image pooling is a global average pooling followed by a 1×1 convolution and bilinear interpolation. The numbers of input and output channels of each layer are marked in parentheses.

Table 1: Ablation study results.

Method	mIoU (%)	Params	Multi-Adds
EDANet	65.10	0.68M	8.97B
(a) Core module.			
EDA-non-asym	65.11	0.81M	11.41B
EDA-non-dense	63.92	0.73M	8.87B
(b) Extra context module			
EDA-shallow	58.09	0.55M	7.77B
EDA-ASPP	60.64	3.41M	41.42B
(c) Decoder			
EDA-ERFdec	65.56	0.78M	12.95B
(d) Downsampling block			
EDA-DenseDown	61.63	0.42M	8.51B

Table 2: Evaluation results on the Cityscapes test set. The methods whose speed is faster than 15 FPS are included. "†": GTX 1080Ti, with 3,584 CUDA cores and 11,340 GFLOPS. "††": Titan XP, with 3,854 CUDA cores and 12,150 GFLOPS.

Method	Pretrained	mIoU (%)	Speed (FPS)		Parameters
			Titan X	Other GPUs	
SegNet [1]	ImageNet [8]	56.1	16.7	-	29.5M
ENet [21]	No	58.3	76.9	-	0.36M
SQ [30]	ImageNet	59.8	16.7	-	-
ESPNet [19]	No	60.3	112.9	-	0.36M
SkipNet-MobileNet [26]	ImageNet	61.5	45.0	-	-
ContextNet [22]	No	66.1	18.3	-	0.85M
ERFNet [23]	No	68.0	41.7	-	2.1M
BiSeNet [31]	ImageNet	68.4	-	105.8††	5.8M
ICNet [34]	ImageNet	69.5	30.3	-	-
EDANet (ours)	No	67.3	81.3	108.7†	0.68M

Table 3: Evaluation results on the CamVid test set.

Method	mIoU (%)	Class acc. (%)	Global acc. (%)	Parameters
ENet [21]	51.3	68.3	-	0.36M
ESPNet [19]	55.6	68.3	-	0.36M
SegNet [1]	55.6	65.2	88.5	29.5M
FCN-8s [18]	57.0	-	88.0	134.5M
FC-DenseNet56 [15]	58.9	-	88.9	1.5M
DeepLab-LFOV [3]	61.6	-	-	37.3M
Dilation8 [32]	65.3	-	79.0	140.8M
BiSeNet [31]	65.6	-	-	5.8M
ICNet [34]	67.1	-	-	-
EDANet (ours)	66.4	76.7	90.8	0.68M

Extra context module. The dense connectivity allows EDANet to concatenate multi-scale features and go deeper simultaneously. We compare the ability of our EDA block and the atrous spatial pyramid pooling (ASPP) context module proposed in DeepLab [5] to extract multi-scale representations. We construct the EDA-shallow that contains only four EDA modules in its EDA block 2 as a baseline. Then, we replace the last four EDA modules in EDANet with the ASPP as EDA-ASPP (see Figure 6).

Table 1b shows the results. EDANet attains 7.01% higher accuracy than EDA-shallow, while EDA-ASPP only improves 2.55%. Moreover, EDA-ASPP has 5 times more parameters and 4.6 times more computational cost than EDANet due to its 5-branch structure. Therefore, we observe that a block of only four connected EDA modules is able to outperform a heavy ASPP context module because of its deeper structure and the excellent capability of aggregating multi-scale information.

Decoder. After going through the trade-off analysis between efficiency and accuracy, we do not include the decoder structure in our network design. In our investigation, we build a network called EDA-ERFdec that adds the ERFNet decoder [23] for comparison. The decoder consists of two blocks of a deconvolution layer with stride 2 followed by two ERF modules (see Figure 5b), plus the last deconvolution layer with stride 2 for final output.

As Table 1c shows, EDA-ERFdec obtains 0.46% better accuracy at the expense of 44% more computational cost. Obviously, when we focus on efficiency, adding the decoder does not seem benefit.

Downsampling block. We choose the initial block of ENet [21] as the foundation of our downsampling block. Then, we extend it to the two-mode configuration as described earlier. On the other hand, DenseNet [13] uses a 7×7 convolution layer with stride 2 followed by a 3×3 max-pooling with stride 2 for early downsampling, and creates the transition layers that consist of a 1×1 convolution layer followed by a 2×2 average-pooling with stride 2 for the other downsampling operations. In order to compare the downsampling approach of ours and the one proposed by

DenseNet, we construct EDA-DenseDown by replacing our first two downsampling blocks and the third downsampling block with the early downsampling layers and the transition layer in DenseNet, respectively.

As shown in Table 1d, EDANet attains significantly 3.47% higher accuracy than EDA-DenseDown with only a little more computational cost.

4.2. Evaluation on Cityscapes

We finally train our EDANet in two stages. In the first stage, we train it by the annotations downsampled to 1/8 to the input image size. In the second stage, we train it again by the annotations of the same size as the inputs. In the evaluations, we do not adopt any testing tricks such as multi-crop and multi-scale testing. Table 2 reports our results and the comparisons with the other state-of-art networks in terms of mIoU and inference efficiency on the Cityscapes test set. EDANet achieves 67.3% mIoU, which is better than most of the existing methods that can run at 30 FPS or higher, such as ENet [21] and ESPNet [19], and even outperforms many approaches with lower speed such as Dilation10 [32] and FCN [18]. EDANet attains 108.7 FPS and 81.3 FPS on a single GTX 1080Ti and Titan X GPU, respectively, and thus it is one of the fastest networks now. Some visual results are shown in Figure 7.

4.3. Evaluation on CamVid

We also evaluate our network on the CamVid dataset [2]. As reported in Table 3, our EDANet again achieves outstanding performance in efficiency and accuracy. It is able to process a 360×480 CamVid image at the speed of 163 FPS by using one GTX 1080Ti card. The visual results are shown in Figure 8.

5. Conclusion

In this paper, we have proposed a real-time semantic segmentation network, EDANet, based on the efficient dense modules with asymmetric convolution. The experimental results demonstrate its capability of producing pretty accurate segmentation results with a rather small computational cost comparing to the other state-of-art systems. Going through an extensive investigation, we finally design a well-balanced network architecture for semantic segmentation, which leads to a good trade-off between reliability and efficiency for scene understanding.

Acknowledgements

We would like to thank Ping-Rong Chen for his helpful discussions during the course of this project and Shang-Wei Hung for his drawing for this paper. This work was supported in part by the Mechanical and Mechatronics Systems Research Lab., ITRI, under Grant 3000547822.

Figure 7: Sample results of EDANet on Cityscapes validation set. From left to right: (a) Input, (b) Ground truth, (c) EDANet.

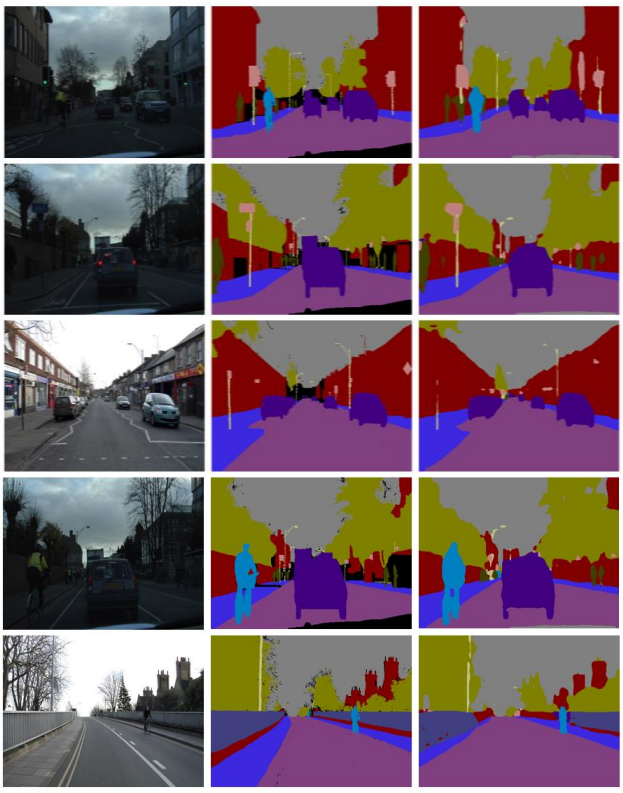

Figure 8: Sample results of EDANet on CamVid test set. From left to right: (a) Input, (b) Ground truth, (c) EDANet.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *TPAMI*, 2017.

[2] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008.

[3] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[4] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic imae segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *TPAMI*, 2017.

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[7] 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.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. In *IJCV*, 2010.

[10] J. Fu, J. Liu, Y. Wang, and H. Lu. Stacked deconvolutional network for semantic segmentation. *arXiv preprint arXiv:1708.04943*, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[13] H. Gao, L. Zhuang, and Q. W. Kilian. Densely connected convolutional networks. In *CVPR*, 2017.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[15] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *CVPRW*, 2017.

[16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[19] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018.

[20] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[21] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv: 1606.02417*, 2016.

[22] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach. ContextNet: Exploring context and detail for semantic segmentation in real-time. In *BMVC*, 2018.

[23] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Efficient convnet for real-time semantic segmentation. In *IEEE Intelligent Vehicles Symposium*, 2017.

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[25] F. Shen, R. Gan, S. Yan, and G. Zeng. Semantic segmentation via structured patch prediction, context crf and guidance crf. In *CVPR*, 2017.

[26] M. Siam; M. Gamal, M. Abdel-Razek, S. Yogamani, and M. Jagersand. Rtseg: Real-time semantic segmentation comparative study. In *ICIP*, 2018.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*, 2014.

[29] C. Szegedy. V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[30] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler, and S. Hochreiter. Speeding up semantic segmentation for autonomous driving. In *NIPSW*, 2016.

[31] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, 2018.

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

[33] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[34] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

Appendix

A.1. Network details

In this appendix, we provide detailed descriptions for the network architectures of the proposed EDANet and all of the variants mentioned in the ablation study section. Tables 4, 5, 6, 7, 8, 9, and 10 correspond to EDANet, EDA-non-asym, EDA-non-dense, EDA-shallow, EDA-ASPP, EDA-ERFdec, and EDA-DenseDown, respectively. In all the following tables, the input sizes are 512×1024. The structures of EDA module, downsampling block, EDA-non-asymmetric module, EDA-non-dense module, and ASPP are shown in Figures 3, 4, 5a, 5b, and 6, respectively.

Table 4: Layer disposal of the proposed EDANet.

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		60	128×256
EDA module 1-1		40	100	128×256
EDA module 1-2		40	140	128×256
EDA module 1-3		40	180	128×256
EDA module 1-4	dilation 2	40	220	128×256
EDA module 1-5	dilation 2	40	260	128×256
Downsampleing block 3	$W_{in} > W_{out}$		130	64×128
EDA module 2-1	dilation 2	40	170	64×128
EDA module 2-2	dilation 2	40	210	64×128
EDA module 2-3	dilation 4	40	250	64×128
EDA module 2-4	dilation 4	40	290	64×128
EDA module 2-5	dilation 8	40	330	64×128
EDA module 2-6	dilation 8	40	370	64×128
EDA module 2-7	dilation 16	40	410	64×128
EDA module 2-8	dilation 16	40	450	64×128
Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 5: Layer disposal of EDA-non-asym.

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		60	128×256
EDA-non-asymmetric module 1-1		40	100	128×256
EDA-non-asymmetric module 1-2		40	140	128×256
EDA-non-asymmetric module 1-3		40	180	128×256
EDA-non-asymmetric module 1-4	dilation 2	40	220	128×256
EDA-non-asymmetric module 1-5	dilation 2	40	260	128×256
Downsampleing block 3	$W_{in} > W_{out}$		130	64×128
EDA-non-asymmetric module 2-1	dilation 2	40	170	64×128
EDA-non-asymmetric module 2-2	dilation 2	40	210	64×128
EDA-non-asymmetric module 2-3	dilation 4	40	250	64×128
EDA-non-asymmetric module 2-4	dilation 4	40	290	64×128
EDA-non-asymmetric module 2-5	dilation 8	40	330	64×128
EDA-non-asymmetric module 2-6	dilation 8	40	370	64×128
EDA-non-asymmetric module 2-7	dilation 16	40	410	64×128
EDA-non-asymmetric module 2-8	dilation 16	40	450	64×128
Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 6: Layer disposal of EDA-non-dense. This dilation rate placement is consistent with ERFNet [23].

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		40	128×256
EDA-non-dense module 1-1			40	128×256
EDA-non-dense module 1-2			40	128×256
EDA-non-dense module 1-3			40	128×256
EDA-non-dense module 1-4			40	128×256
EDA-non-dense module 1-5			40	128×256
Downsampleing block 3	$W_{in} < W_{out}$		80	64×128
EDA-non-dense module 2-1	dilation 2		80	64×128
EDA-non-dense module 2-2	dilation 4		80	64×128
EDA-non-dense module 2-3	dilation 8		80	64×128
EDA-non-dense module 2-4	dilation 16		80	64×128
EDA-non-dense module 2-5	dilation 2		80	64×128
EDA-non-dense module 2-6	dilation 4		80	64×128
EDA-non-dense module 2-7	dilation 8		80	64×128
EDA-non-dense module 2-8	dilation 16		80	64×128
Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 7: Layer disposal of EDA-shallow.

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		60	128×256
EDA module 1-1		40	100	128×256
EDA module 1-2		40	140	128×256
EDA module 1-3		40	180	128×256
EDA module 1-4	dilation 2	40	220	128×256
EDA module 1-5	dilation 2	40	260	128×256
Downsampleing block 3	$W_{in} > W_{out}$		130	64×128
EDA module 2-1	dilation 2	40	170	64×128
EDA module 2-2	dilation 2	40	210	64×128
EDA module 2-3	dilation 4	40	250	64×128
EDA module 2-4	dilation 4	40	290	64×128
Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 8: Layer disposal of EDA-ASPP. The ASPP structure is consistent with DeepLabv3 [5].

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		60	128×256
EDA module 1-1		40	100	128×256
EDA module 1-2		40	140	128×256
EDA module 1-3		40	180	128×256
EDA module 1-4	dilation 2	40	220	128×256
EDA module 1-5	dilation 2	40	260	128×256
Downsampleing block 3	$W_{in} > W_{out}$		130	64×128
EDA module 2-1	dilation 2	40	170	64×128
EDA module 2-2	dilation 2	40	210	64×128
EDA module 2-3	dilation 4	40	250	64×128
EDA module 2-4	dilation 4	40	290	64×128

ASPP						
Branch	(1)	(2)	(3)	(4)	Branch	(5)
Convolution	1×1	3×3	3×3	3×3	Average-pooling	64×128
Dilation rate	-	6	12	18	# Output channels	290
# Output channels	290	290	290	290	Output size	1×1
Output size	64×128	64×128	64×128	64×128	Convolution	1×1
					# Output channels	290
					Output size	1×1
					Interpolation	-
					# Output channels	290
					Output size	64×128
Concatenation	-					
# Output channels	1450					
Output size	64×128					
Convolution	1×1					
# Output channels	290					
Output size	64×128					

Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 9: Layer disposal of EDA-ERFdec. The decoder structure is consistent with ERFNet.

Name	Mode	Growth rate	# Output channels	Output size
Downsampleing block 1	$W_{in} < W_{out}$		15	256×512
Downsampleing block 2	$W_{in} < W_{out}$		60	128×256
EDA module 1-1		40	100	128×256
EDA module 1-2		40	140	128×256
EDA module 1-3		40	180	128×256
EDA module 1-4	dilation 2	40	220	128×256
EDA module 1-5	dilation 2	40	260	128×256
Downsampleing block 3	$W_{in} > W_{out}$		130	64×128
EDA module 2-1	dilation 2	40	170	64×128
EDA module 2-2	dilation 2	40	210	64×128
EDA module 2-3	dilation 4	40	250	64×128
EDA module 2-4	dilation 4	40	290	64×128
EDA module 2-5	dilation 8	40	330	64×128
EDA module 2-6	dilation 8	40	370	64×128
EDA module 2-7	dilation 16	40	410	64×128
EDA module 2-8	dilation 16	40	450	64×128
Deconvolution 1	2×2, stride 2		64	128×256
EDA-non-asymmetric module d1-1			64	128×256
EDA-non-asymmetric module d1-2			64	128×256
Deconvolution 2	2×2, stride 2		16	256×512
EDA-non-asymmetric module d2-1			16	256×512
EDA-non-asymmetric module d2-2			16	256×512
Deconvolution 2	2×2, stride 2		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

Table 10: Layer disposal of EDA-DenseDown. The dowsampling layers are consistent with DenseNet [13].

Name	Mode	Growth rate	# Output channels	Output size
Convolution	7×7, stride 2		60	256×512
Max-pooling	3×3, stride 2		60	128×256
EDA module 1-1		40	100	128×256
EDA module 1-2		40	140	128×256
EDA module 1-3		40	180	128×256
EDA module 1-4	dilation 2	40	220	128×256
EDA module 1-5	dilation 2	40	260	128×256
Convolution	1×1		130	128×256
Average-pooling	2×2, stride 2		130	64×128
EDA module 2-1	dilation 2	40	170	64×128
EDA module 2-2	dilation 2	40	210	64×128
EDA module 2-3	dilation 4	40	250	64×128
EDA module 2-4	dilation 4	40	290	64×128
EDA module 2-5	dilation 8	40	330	64×128
EDA module 2-6	dilation 8	40	370	64×128
EDA module 2-7	dilation 16	40	410	64×128
EDA module 2-8	dilation 16	40	450	64×128
Projection layer	1×1 conv.		# Classes	64×128
Bilinear interpolation	×8		# Classes	512×1024
Bilinear interpolation (inference only)	×2		# Classes	1024×2048

A.2. Results on the Cityscapes and the CamVid datasets

In this section, we provide additional segmentation results of the proposed EDANet on Cityscapes [7] and CamVid dataset [2]. Tables 11 and 12 list the IoU scores for each class on the two datasets respectively.

Table 11: IoU scores on Cityscapes test set.

Class	IoU
Road	97.8
Sidewalk	80.6
Building	89.5
Wall	42.0
Fence	46.0
Pole	52.3
Traffic light	59.8
Traffic sign	65.0
Vegetation	91.4
Terrain	68.7
Sky	93.6
Person	75.7
Rider	54.3
Car	92.4
Truck	40.9
Bus	58.7
Train	56.0
Motorcycle	50.4
bicycle	64.0
Metric	Value
mIoU classes	67.3
mIoU categories	85.8

Table 12: IoU scores on CamVid test set.

Class	IoU
Sky	90.8
Building	82.5
Pole	28.5
Road	93.3
Pavement	78.3
Tree	75.0
Sign symbol	43.7
Fence	44.4
Vehicle	81.0
Pedestrian	54.6
Bike	57.9
Metric	Value
mIoU	66.4
Class average acc.	76.7
Global acc.	90.8