SIP (2020), vol. 9, e27, page 1 of 14 © The Author(s), 2020. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

doi:10.1017/ATSIP.2020.25

ORIGINAL PAPER

DSNet: an efficient CNN for road scene segmentation

PING-RONG CHEN,[1] HSUEH-MING HANG,[1] SHENG-WEI CHAN[2] AND JING-JHIH LIN[2]

Road scene understanding is a critical component in an autonomous driving system. Although the deep learning-based road scene segmentation can achieve very high accuracy, its complexity is also very high for developing real-time applications. It is challenging to design a neural net with high accuracy and low computational complexity. To address this issue, we investigate the advantages and disadvantages of several popular convolutional neural network (CNN) architectures in terms of speed, storage, and segmentation accuracy. We start from the fully convolutional network with VGG, and then we study ResNet and DenseNet. Through detailed experiments, we pick up the favorable components from the existing architectures and at the end, we construct a light-weight network architecture based on the DenseNet. Our proposed network, called DSNet, demonstrates a real-time testing (inferencing) ability (on the popular GPU platform) and it maintains an accuracy comparable with most previous systems. We test our system on several datasets including the challenging Cityscapes dataset (resolution of 1024 × 512) with an Mean Intersection over Union (mIoU) of about 69.1% and runtime of 0.0147 s/image on a single GTX 1080Ti. We also design a more accurate model but at the price of a slower speed, which has an mIoU of about 72.6% on the CamVid dataset.

Keywords: Semantic segmentation, Real-time CNN segmentation, CNN architecture, Road scene segmentation

Received 9 August 2020; Revised 27 October 2020

I. INTRODUCTION

With the fast development of automated driving systems, a stable and reliable surrounding scene analysis becomes essential for a safe driving environment.[1] The deep-learning based image semantic segmentation is one of the best solutions because it is sufficiently robust in analyzing the complicated environments. It partitions a captured image into several regions and recognizes the class (object) of every pixel, so it can be viewed as pixel-level classification. Different from image classification, the image semantic segmentation identifies the object classes in images and also finds the locations of objects in images. In addition, it provides precise object boundary information. Nevertheless, the high accuracy of image semantic segmentation is often at the high complexity of a convolutional neural network (CNN) model without consideration of inference time, resulting in a difficult implementation on several light devices. Therefore, a fast and efficient CNN model is very desirable and imperative for a practical semantic segmentation system.

Recently, an encoder–decoder architecture is popular for semantic segmentation. The encoder is usually a classification network, such as VGG [2], ResNet [3], and DenseNet [4]. It employs a series of down-sampling layers to condense the information. However, the down-sampling operation drastically reduces the detailed spatial information which is quite important for the image semantic segmentation task. To address this issue, some decoders are designed to recover the spatial resolution by using the up-sampling process. Deconvolution is commonly used to produce a learnable up-sampling process in many popular networks, such as DeconvNet [5] and fully convolutional network (FCN) [6]. Un-pooling used in SegNet [7] is another method to up-sample the feature maps by reusing the max-pooling indices produced by the encoder. On the other hand, some networks are constructed without a decoder network but retain the detailed spatial information from the encoder part, such as DeepLab v2 [8], DeepLab v3 [9], and PSPNet [10]. They remove some down-sampling layers and apply the dilated convolution, which can maintain the spatial resolution without sacrificing the receptive field. Although this method can improve accuracy, the enlarged feature maps often significantly slow down the processing, especially for a deep architecture together with large feature maps. Also, DeepLab v3+ [11] includes a decoder network to combine the multi-scale information to obtain better results. These previous works give us clues in constructing a fast network that is able to capture multi-scale information without using dilated convolution.

In the past a few years, several efficient semantic segmentation networks have been proposed, such as ENet [12] and ERFNet [13]. ENet is constructed based on the concept of

[1]National Chiao Tung University, Hsinchu, Taiwan
[2]Industrial Technology Research Institute, Hsinchu, Taiwan

Corresponding author:
Hsueh-Ming Hang
Email: hmhang@nctu.edu.tw

[1] A preliminary version of this work appears in [1].

SegNet, but it is much slander than the latter and thus offers a light and fast architecture. Moreover, ENet uses dilated convolution and stacked residual layers to deepen the network so that the accuracy can be maintained. ERFNet is a wider version of ENet and uses deconvolutional layers for the up-sampling process. Also, they adopt the factorized filter structure [14] to separate a two-dimensional filter into two one-dimensional filters in the convolutional layer and thus considerably reduce the number of parameters. Further, both of ENet and ERFNet use an early down-sampling scheme and an extremely narrow architecture compared to the schemes with a heavy encoder such as VGG16 and ResNet101 (top-ranked on ILSVRC [15]). In this work, we adopt a similar idea in constructing a narrow architecture, but DenseNet is selected as the backbone instead of ResNet because it combines the multi-scale information more frequently, which may be more appropriate for semantic segmentation purpose.

II. PROPOSED NETWORK OVERVIEW

In this paper, the target is to construct a fast network architecture without degrading its accuracy. Up to now, constructing a CNN is often empirical, and it is hard to predict in advance the results of a modified network. Our aim is to design a favorable architecture for real-time applications. We conducted a series of experiments to select the efficient structures together with various components as described in Section III. Here, we first give an overview of the proposed network, Dense Segmentation Network, DSNet, in brief. The entire architecture is shown in Figs 1 and 2. Figure 1 is the *fast* version, which has a smaller model size and a bit lower accuracy comparing to the *accurate* version in Fig. 2. But these two versions share a lot of the common architecture and components. Basically, the network architecture includes two parts, the encoder and the decoder. The details are described below.

A) Encoder – fast version

The encoder part is constructed based on the concept of DenseNet because it achieves high performance with narrow layers, resulting in less overall computational cost. The encoder consists of one initial block, four non-bottleneck units (without 1 × 1 convolution), and 26 bottleneck units (with 1 × 1 convolution). The early down-sampling operation (convolution with a stride of 2) is employed at the initial block to shrink the size of feature maps and to speed up the network. Meanwhile, the output channel of the initial block is set to 32, the growth rate is set to 32, which represents how many feature maps are generated in one dense unit, and the channels are compressed with a ratio of 0.5 in the transition layer before a pooling operation to reduce the complexity. The bottleneck units adopt a 1 × 1 convolution to reduce the number of channels. As the number of channels is quite small in Block 1 and Block 2, it is not necessary to decease the channel numbers there. Thus, the non-bottleneck units are adopted in Block 1 and Block 2 rather than the bottleneck units.

In the above components, we two modified two original dense units to make a trade-off between accuracy and speed. First, we modify the composition unit from BN-ReLU-Conv [16] to Conv-BN-ReLU. Even though the full pre-activation unit (BN-ReLU-Conv) is claimed to improve the results, it is not possible to merge BN layers into Conv layers during the testing phase (as discussed in [17]). Thus, to increase speed, we redesign the DenseNet architecture by using the Conv-BN-ReLU units to replace the BN-ReLU-Conv units.

Second, we slightly modify the dense unit by inserting another convolutional layer at the end of bottleneck and the non-bottleneck architectures, as shown in Fig. 3. By adopting this modification, the network can be deepened at a low computational cost because the preceding layer is sufficiently narrow. Also, the additional convolutional layer can enlarge the receptive field to capture large-scale objects and to produce a better result on the high-resolution dataset, such as Cityscapes dataset. The experimental justification on the modification is to be discussed in Section III.

B) Encoder – accurate version

In our experiments (Section III), we find that the convolutional layers operated on the large feature maps are important for an accurate pixel-level classification, especially for processing a low-resolution image. Thus, we propose another architecture (Fig. 2) to deal with the low-resolution images. This architecture removes the early down-sampling operation in the initial block in Fig. 1 so that the larger feature maps are fed to the rest of the network.

Meanwhile, in the decoder, one skip connection connected to Block 2 is removed and all the feature maps are up-sampled to the resolution of Block 3. Thus, the number of channels after the concatenated layer becomes 96 and the last deconvolutional layer up-samples the feature maps by a factor of 4. As shown in Fig. 2, this architecture is called *DSNet-accurate* (accurate dense segmentation network). In summary, we proposed two architectures to deal with different input resolutions. We name the architecture with the early down-sampling layer as *DSNet-fast*. And, the architecture without the early down-sampling operation is called as *DSNet-accurate*.

C) Decoder

In our experiments (Section III), we also find that a decoder with a heavy structure does not seem to provide much accuracy improvement. Hence, simplifying the decoder is a feasible way to speed up the network. So, we reduce the number of channels to 32 by employing four convolutional layers after the Block 2, Block 3, Block 4, and Block 5 (DSNet-fast in Fig. 1). Furthermore, all the feature maps are up-sampled to the resolution of Block 2 to concatenate them together. In the end, a deconvolutional layer with a four up-sampling rate is used to recover the spatial

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

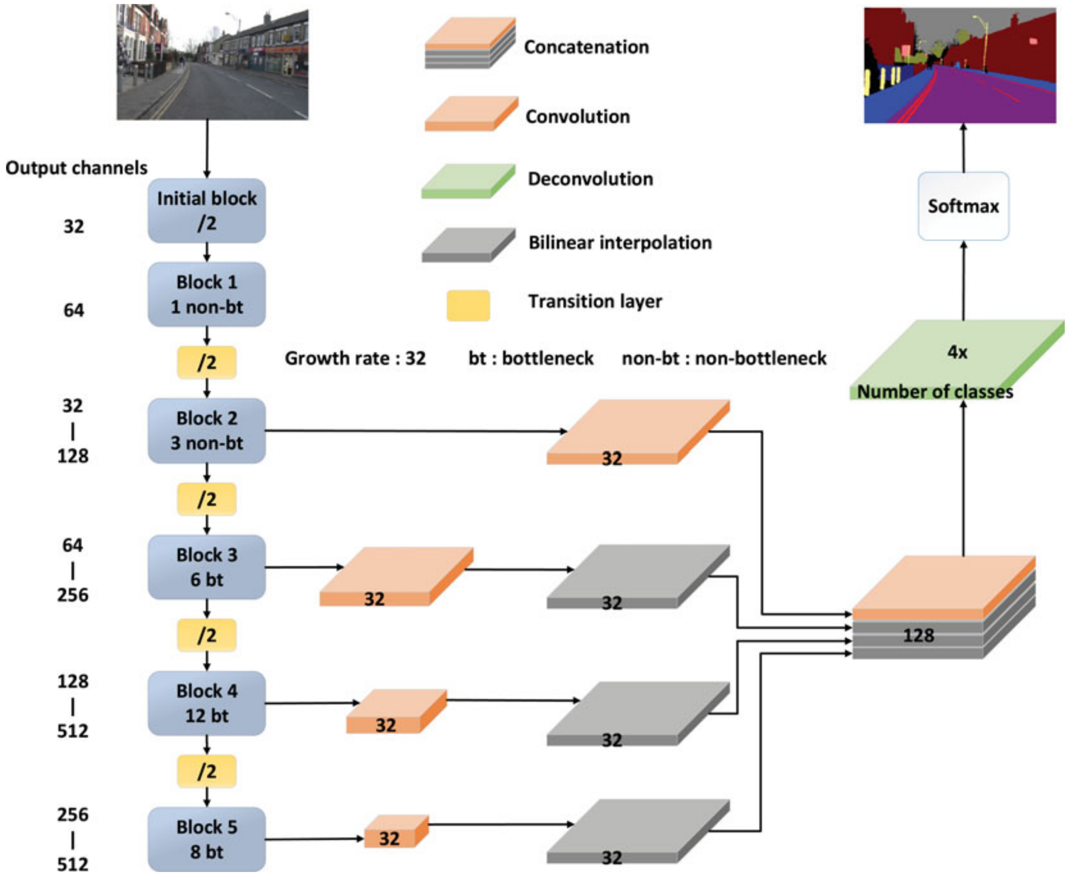

Fig. 1. The architecture of fast dense segmentation network (DSNet-fast). The encoder is constructed based on the deep dense unit described in Fig. 3.

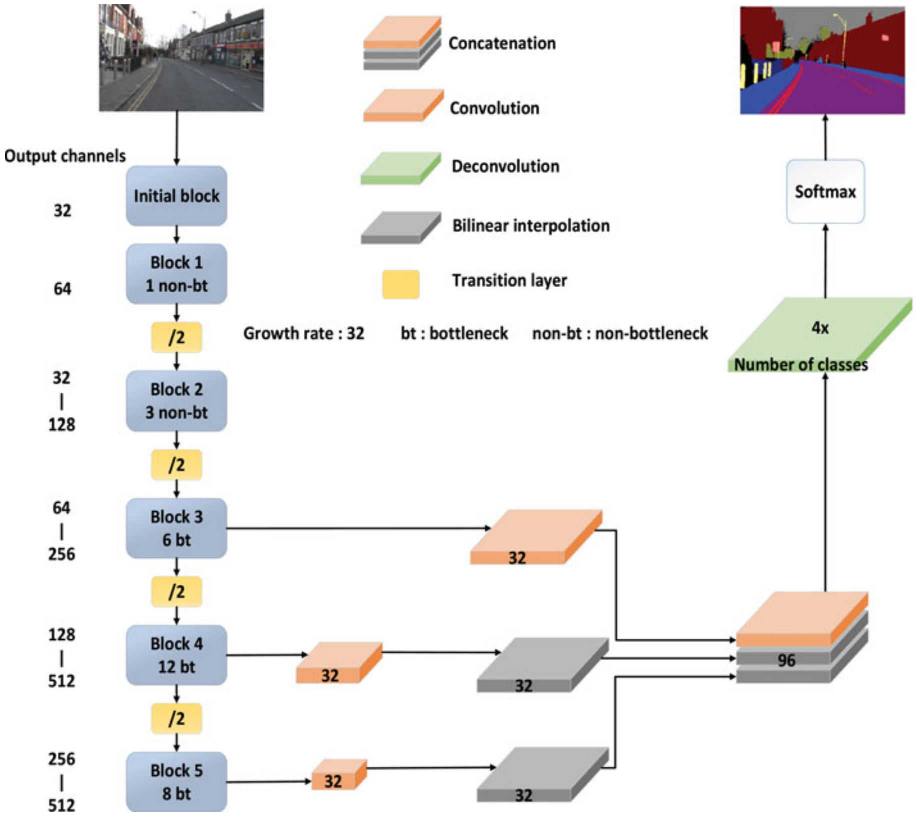

Fig. 2. The architecture of the proposed network, DSNet-accurate. It differs from Fig. 1 (DSNet-fast) mainly in that it removes the down-sampling operation in the initial block. And it removes the skip connection connected to Block 2, and thus has 96 channels at the concatenation layer rather than 128 channels in DSNet-fast.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

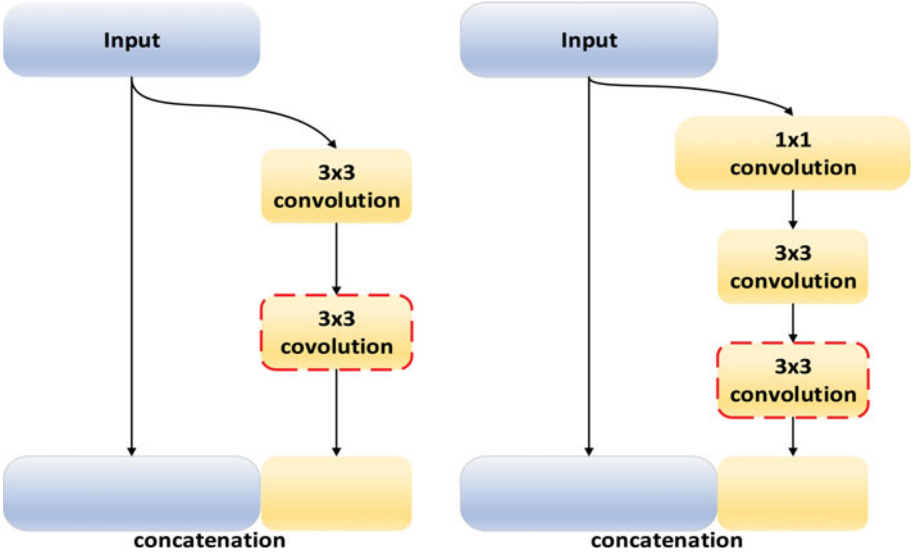

Fig. 3. The modified deep dense units by inserting another convolutional layer (red dotted block). (a) Non-bottleneck architecture. (b) Bottleneck architecture.

resolution for the final dense segmentation. By using this simple decoder, the computational complexity is considerably decreased and the accuracy can be maintained at the same time.

III. STRUCTURE/COMPONENT SELECTION

As mentioned earlier, a series of experiments are carried out to investigate the pros and cons of various structures and components. The results give us the clues in selecting appropriate architecture and components for designing DSNet. We now explain the ablation study in detail in this section. DSNet was hinted by the architecture of FCN. FCN is an encoder–decoder network and its decoder combines the multi-scale information in order to produce an accurate prediction. We modify the backbone network of FCN using ResNet [3] and DenseNet [4]. Extensive experiments were conducted to compare various combinations before we settled on a modified version based on DenseNet. We divide the experiments into two parts, encoder experiments and decoder experiments. Before describing them, we first describe the dataset used to benchmark the performance and the parameter setting in training the networks.

A) Dataset

We use two popular road-scene datasets to evaluate all the networks in this paper. The first one is CamVid [18], which consists of 367 training images, 101 validation images, and 233 testing images. The resolution of all images is 480 × 360 and there are in total 11 classes in the dataset. The CamVid dataset can show the importance of detailed information because the image resolution is relatively low.

The second dataset is Cityscapes [19], which is a larger dataset for semantic understanding of urban street scenes. All images are at 2048 × 1024 resolution and there are 19 classes for training. Two kinds of annotations are provided, fine-annotation and coarse-annotation. In this paper, we use only the fine-annotated dataset to train and evaluate the networks. It is composed of 2950 training images, 500 validation images, and 1525 test images. For speed consideration, the images are down-sampled by a factor of 2 (horizontally and vertically) in some experiments. The Cityscapes dataset can show the influence of receptive field on the networks when the input images are of high resolutions.

B) Implementation details

All the experiments are conducted on the Pytorch framework [20] with a single Maxwell Titan X GPU. The optimizer is stochastic gradient descend, with a weight decay of 0.0005, a momentum of 0.9, a batch size of 4, and a base learning rate of 0.05. Inspired by [21, 22], the learning rate is adjusted after every iteration according to the following equation:

$$lr = lr_{base} \times \left(1 - \frac{iteration}{total\ iterations}\right)^{power} \quad (1)$$

with a power of 0.9. Also, a total of 13 800 iterations (150 epochs) is set in the training on the CamVid dataset, and 74 400 iterations (100 epochs) is set in the training on the Cityscapes dataset. In addition, we adopt a class balancing strategy to compensate small-size classes. Inspired by [5, 11], the class weightings are calculated by:

$$w_c = \frac{1}{log(p_c + k)} \quad (2)$$

where k is a constant set to 1.1 and p_c represents the probability of the presence of class c in pixel level; then, these class weightings are divided by their maximum value to normalize their values into [0, 1], so that the other hyper-parameters can be fixed without affecting the convergence in training.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

Table 1. Results of FCN-VGG16 and FCN-ResNet50 on CamVid test set and Cityscapes validation set.

Method	Dataset	Image resolution	mIoU (%)	Frame rate (fps)
FCN-VGG16	CamVid	480 × 360	67.5	39.8
	Cityscapes	2048 × 1024	65.2	4.1
FCN-ResNet50	CamVid	480 × 360	65.0	38.1
	Cityscapes	2048 × 1024	67.4	5.1

C) Encoder structure – ResNet and variations

In the previous studies, many semantic segmentation researchers adopted and modified the neural networks originally designed for image classification. As said earlier that we also started from the FCN structure which adopts VGG16 as the backbone (we name it, *FCN-VGG16*) and replace its network by the recent high-performance networks. Essentially, there are three network architectures that we have investigated: VGG16 [2], ResNet50 [3], and DenseNet [4].

First, it has been shown that the residual block proposed by He *et al.* [3] can significantly improve the classification accuracy if the depth of the CNN gets deeper. Hence, we modify the encoder of FCN by replacing VGG16 with ResNet50 to construct the *FCN-ResNet50* network. The modified network structure is shown in Fig. 4, and is called *FCN-ResNet*. As the resolution of feature maps at the end of FCN-ResNet is twice smaller than FCN-VGG, we slightly revise the decoder of FCN-ResNet. Different from the decoder in FCN-VGG, another convolutional layer and another de-convolutional layer are added into the decoder of FCN-ResNet so that the last de-convolutional layer can up-sample the feature maps by a factor of 4, which is identical to FCN-VGG. In addition, the number of channels in ResNet increases considerably compared to VGG.

However, according to Table 1, we find that the performance varies on different datasets. The mean Intersection over Union (mIoU) is a commonly used metric for evaluating the segmentation performance [5–7]. The performance of FCN-ResNet50 is worse than that of FCN-VGG16 on the CamVid dataset, but the accuracy of FCN-ResNet50 is significantly improved on the Cityscapes dataset. We suppose that this is due to different input resolutions. Thus, we investigate the impact of the input resolution by using the following experiments.

When viewing the segmentation maps in Fig. 5, we find that the truck object estimated by FCN-VGG16 is fragmented and incomplete. In contrast, FCN-ResNet50 is able to capture the truck more accurately. One explanation is that the receptive field of ResNet50 is larger than that of VGG16; that is, the ResNet50 is able to recognize large size objects and thus results in a more accurate estimation on the high-resolution images.

Next, we study the performance on the low-resolution images. We investigate the problem based on the structure of VGG16 and ResNet50. In the first few layers, the structure of ResNet50 has an early down-sampling layer, and then followed by a down-sampling layer (max-pooling layer). Heuristically, these consecutive down-sampling operations may reduce the feature maps to a rather small size. Thus, it is difficult to retain the detailed spatial information and thus leads to a poor segmentation result on small objects. In order to verify this speculation, we replace the first four convolutional layers and two max-pooling layers of VGG16 by one early down-sampling layer followed by a down-sampling layer and call it FCN-VGG-ED as shown in Fig. 6.

Table 2. Results of FCN-VGG16 and FCN-VGG-ED on CamVid test set (training from scratch).

Method	mIoU (%)	Frame rate (fps)	Model size (MB)
FCN-VGG16	56.9	39.8	72.6
FCN-VGG-ED	52.2	60.6	71.1

As there is no pre-trained model for the modified network, FCN-VGG16 and FCN-VGG-ED (early down-sampling) in Table 2 are trained with the random initialization (training from scratch). The results show that the FCN-VGG-ED performance degrades considerably and we thus confirm that the early down-sampling layer is one of the reasons causing FCN-ResNet50 performance degradation on low resolution images. On the other hand, the number of parameters may be another reason causing the degradation in FCN-VGG-ED. In summary, we have the following observations based on the results of encoder experiments:

- In order to retain the detailed spatial information, the convolution operation performed on the large feature maps is important for semantic segmentation.
- In order to capture the long-range information and large-scale objects, a deep architecture is needed for high-resolution images.
- The early down-sampling layer speeds up the operation significantly but it degrades the small-size image results.

In brief, we need a deep architecture to process a high-resolution image. Also, we hope the network is capable of adding extra convolutional layers for large-size feature maps but we also want to have a low computational cost. Thus, the trade-off between using the early down-sampling layer and the extra convolutional layers is a critical issue in designing a fast neural network. The above observations give us clues in designing the encoder part in Fig. 1. We next look into the DenseNet [4]. Due to the low complexity in every dense unit, we can insert additional convolutional layer to process the large feature maps (e.g. Block 1 in Fig. 1) and thus more detailed spatial information can be retained.

D) Encoder structure – DenseNet and variations

Figure 7 shows the architecture of the network using DenseNet, named *FCN-DenseNet*, which adopts the original residual units (without the dotted block in Fig. 2). We also contruct *FCN-DenseNet-D* (with the dotted block in Fig. 2), which adopts deeper residual units. The encoder consists of one initial block, four non-bottleneck units,

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

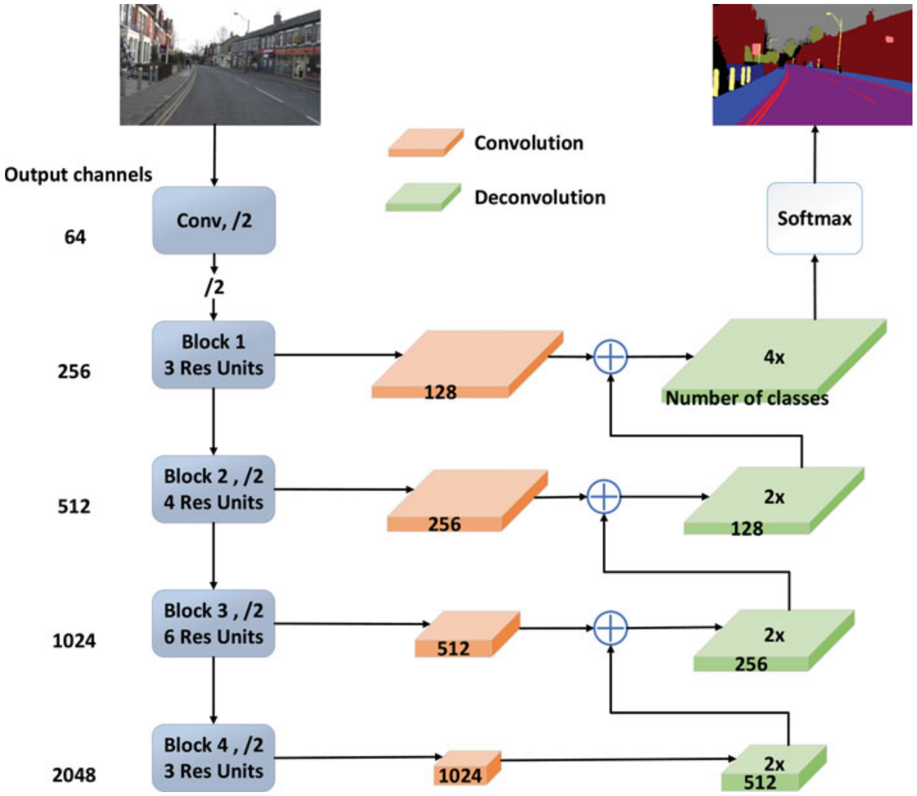

Fig. 4. The architecture of FCN-ResNet. The backbone of encoder is ResNet, and a FCN-based decoder is attached to fuse feature maps by summation.

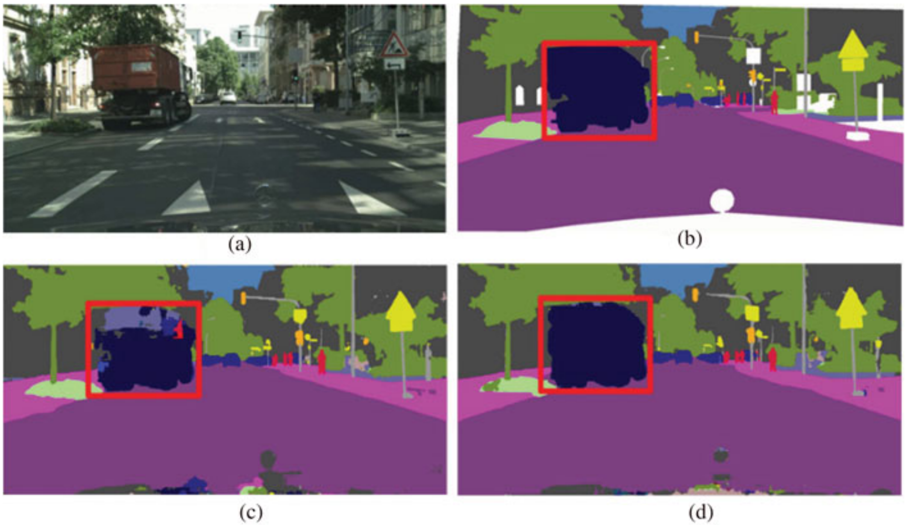

Fig. 5. An output sample that shows the importance of receptive field. (a) Input image, (b) Ground truth, (c) FCN-VGG16 output, (d) FCN-ResNet50 output.

and 26 bottleneck units. The early down-sampling operation is employed into the initial block to speed up the network. Different from the ResNet down-sampling immediately behind the early down-sampling layer, we insert another block (Block 1) to retain the detailed information. Meanwhile, the output channel number of the initial block is set to 32, and the growth rate is set to 32, which represents how many feature maps are generated in one residual unit. The channels are compressed with a ratio of 0.5 in the transition layer to reduce the complexity, and the dropout strategy is adopted at the end of every residual layer with 0.1 drop ratio. Incidentally, the bottleneck units adopt a 1 × 1 convolution to reduce the number of channels, but the number of channels is still few in Block 2 and Block 3, so we use non-bottleneck units in Block 2 and Block 3 rather than bottleneck units.

Table 3 shows the results of FCN-DenseNet and FCN-DenseNet-D and we used the Cityscapes dataset at 1024 × 512 resolution to compare the performance with FCN-ResNet and FCN-VGG. Here, although FCN-DenseNet and FCN-DenseNet-D do not have pre-trained models on the ImageNet dataset, the accuracy of FCN-DenseNet-D is very close to that of FCN-VGG. Also, the speed and the model size of FCN-DenseNet-D perform much better than

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

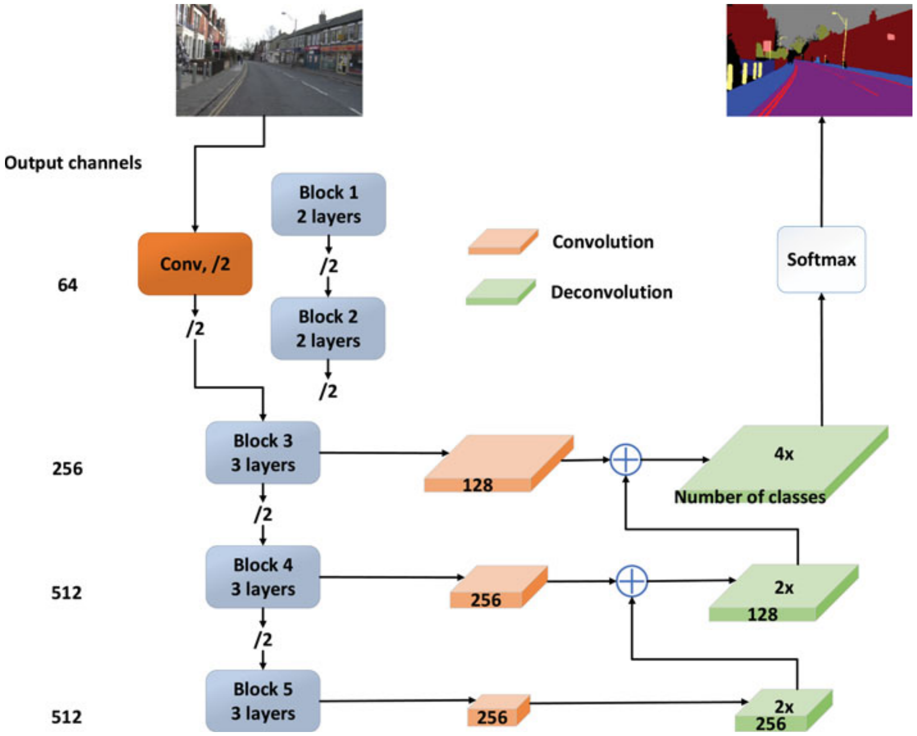

Fig. 6. The architecture of FCN-VGG-ED (ED: early down-sampling). Similar to Fig. 4, the decoder is a FCN-based structure, but the backbone of the encoder is VGG.

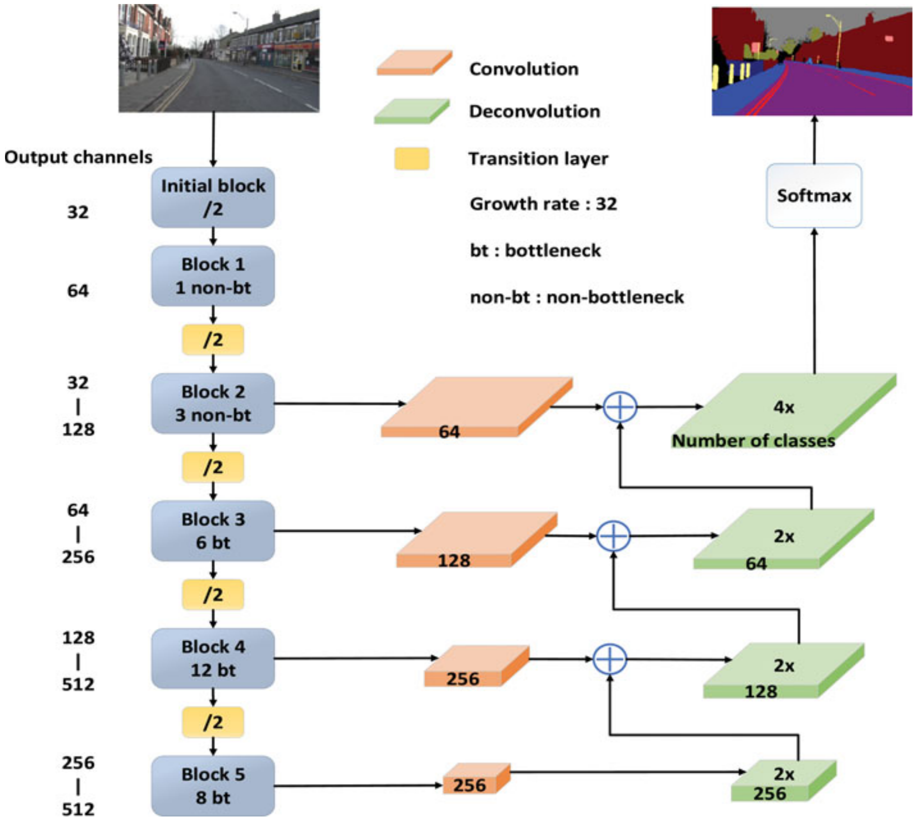

Fig. 7. The architecture of FCN-DenseNet and FCN-DenseNet-D ("D" means the deep dense unit described in Fig. 3). Similar to Figs 4 and 6, a FCN-base decoder is employed. The encoder is a customized structure based on the idea of DenseNet.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

Table 3. Results of FCN-DenseNet and FCN-DenseNet-D on Cityscape validation set.

Model	Pre-trained	Global Acc. (%)	Class Acc. (%)	mIoU (%)	Frame rate (fps)	Model size (MB)
FCN-VGG	✓	94.4	75.7	66.8	16.6	76.5
FCN-ResNet	✓	93.4	69.4	60.6	17.0	239.3
FCN-DenseNet		94.0	75.9	65.7	50.6	26.8
FCN-DenseNet-D		94.4	77.6	67.2	45.1	27.9

FCN-VGG and FCN-ResNet, which indicates that FCN-DenseNet-D is a potential architecture for real-life applications. So, we like to improve accuracy and speed based on the DenseNet-based architecture to construct a more efficient network.

Pre-training the networks on a large dataset is a common way to improve the performance because it can set up a good initialization and strengthen the feature representations. Hence, we pre-train the encoders of FCN-DenseNet and FCN-DenseNet-D on the ImageNet dataset [15] consisting of 1 281 167 training images and 50 000 validation images with 1000 object categories. Before training on the ImageNet, we modify the encoders by adding one global average pooling layer and one fully connected layer, so that the encoders can be converted to the networks for image classification purpose. Then, we discard the attached pooling layer and the fully connected layer, and attach the decoder network to reconstruct FCN-DenseNet and FCN-DenseNet-D. Then, we retrain these two encoders with the pre-trained models on the Cityscapes dataset at 1024 × 512 resolution. With the help of the pre-trained model, the performance can be improved 2–3% mIoU. Furthermore, although FCN-DenseNet-D is slightly slower than FCN-DensNet, it attains a higher accuracy and runs pretty fast (45 fps in Table 3). Hence, it (with pre-trained model) is chosen as the baseline in the later experiments.

E) Decoder experiments

As said, we discard the attached pooling layer and the fully connected layer of the pre-trained encoder, and connect it to the FCN-based decoder. We explore two fusion methods at decoder, summation, and concatenation, which have been used in ResNet and DenseNet, respectively. Often, the concatenation fusion has better performance; however, concatenating the feature maps directly increases the number of channels (called *wide decoder*) and results in a high computational cost in the following layers, as shown in Fig. 8.

Therefore, we reduce the output channels to half at every convolutional layer at the decoder to reduce the complexity (called *narrow decoder*, Fig. 8). According to the results in Table 4, there is no obvious difference among the accuracy of these three architectures. But the model with narrow decoder speeds up the network using fewer parameters. Hence, simplifying the decoder seems to be a way to construct an efficient network.

For the purpose of designing a light decoder, we conduct four variations of the decoder, as Fig. 9 shows. All of them have the identical encoder, which is similar to the front-end module in Dilation 8 [23], or, essentially it is a modified SegNet. The 14th convolutional layer is inserted to adjust the number of channels for the decoder. In total, four decoders are designed and tested. Model-1 adopts the SegNet-like decoder but replaces the un-pooling layers by the bilinear interpolation layers. Model-2 discards the decoder network and up-samples the feature maps directly to the input resolution without additional convolutional layers. It can be viewed as an architecture without the decoder. Model-3 recovers the spatial resolution gradually but it removes the last two convolutional layers as compared to Model-1. Model-4 up-samples the feature maps directly to the input resolution and uses two convolutional layers to recover the detailed information.

Table 4. Fusion methods in the decoder (Cityscapes validation set at 1024 × 512 resolution).

Decoder	mIoU (%)	Frame rate (fps)	Model size (MB)
Summation	68.8	45.1	27.9
Concat-wide	68.7	41.4	30.9
Concat-narrow	68.7	50.6	18.0

Table 5. Results of four decoders on CamVid test set.

Method	mIoU (%)	Frame rate (fps)	Model size (MB)
Model-1	64.1	24.3	70.6
Model-2	63.0	34.5	59.1
Model-3	63.9	27.0	70.2
Model-4	64.2	30.3	60.3

Table 5 shows the results of four architectures in Fig. 9, when the CamVid dataset is tested. The results of these four models are very close, which confirm that the decoder plays a lesser role in improving the overall performance. Thus, we can simplify the decoder to speed up the network. Additionally, the results show that a good decoder can slightly improve the accuracy; therefore, we design the network with a moderate decoder for accuracy consideration. Moreover, among Model-1, Model-3, and Model-4, Model-4 is fast and using fewer parameters, and thus it is preferred for constructing an efficient network. Thus, Model-4 becomes our choice for the decoder.

In summary, the decoder experiments give us the following observations:

- Using a narrow decoder is able to speed up the network and it provides similar accuracy compared to a wide decoder.
- Up-sampling the feature maps to a large size and/or using additional convolutional layers to recover the information can produce more accurate results.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

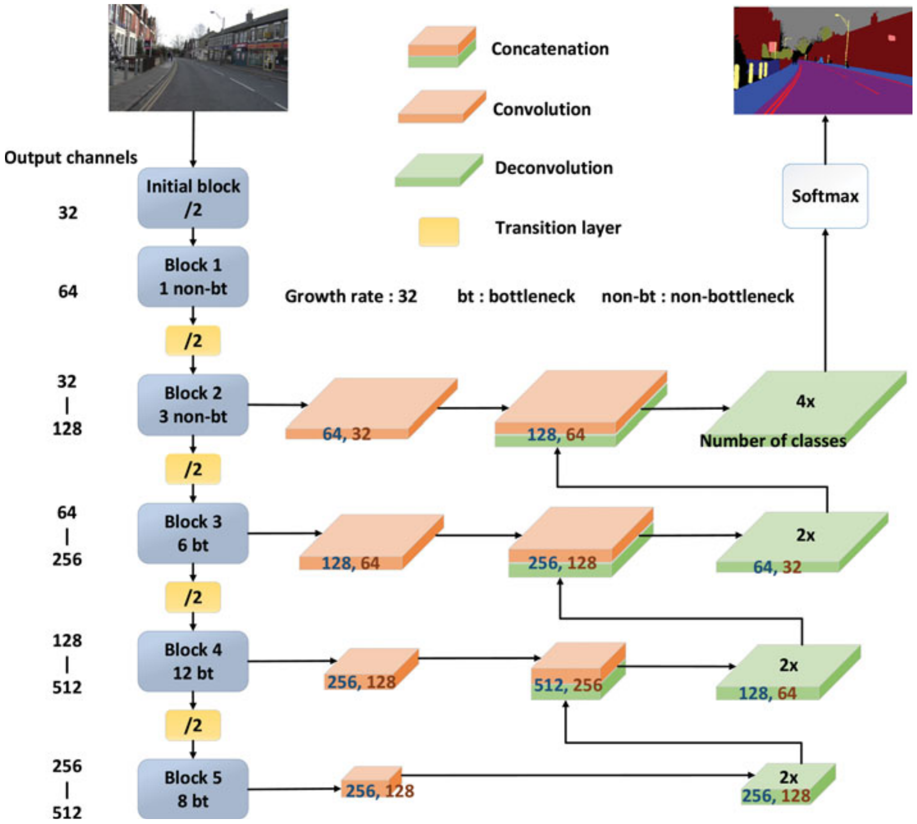

Fig. 8. The architecture of FCN-DenseNet-D with the *wide* decoder and the *narrow* decoder. Differing from Fig. 7, the concatenation operator is employed instead of summation in the decoder. There are two channel numbers in the skip connections and decoder layers. The left number (blue) is for the wide decoder and the right one (red) is for the narrow decoder.

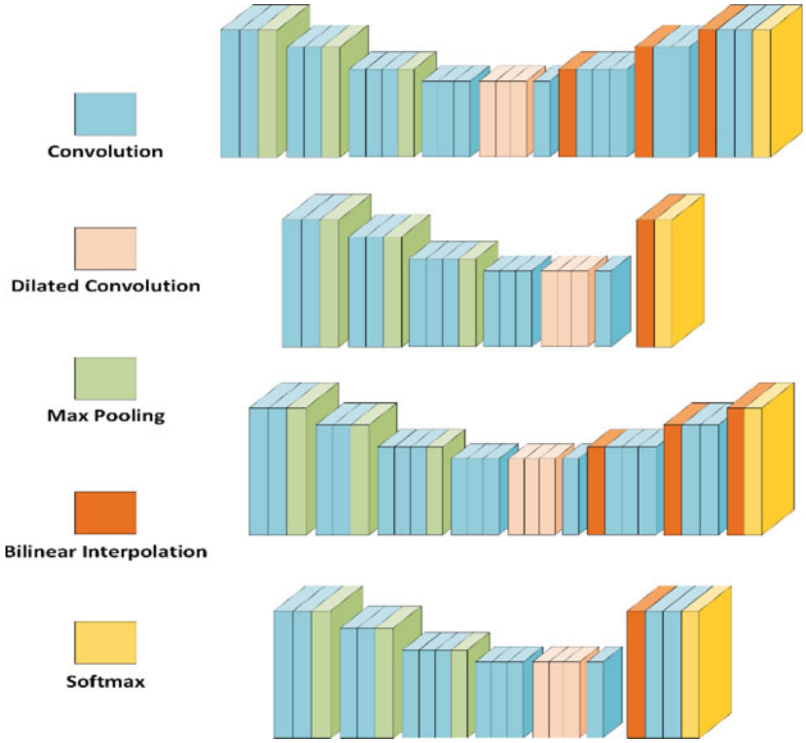

Fig. 9. Variations of Decoder. From top to bottom: (a) Model-1, (b) Model-2, (c) Model-3, (d) Model-4.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

Therefore, in our final design, we remove the decoder of FCN-DenseNet and use the narrow convolutional layers followed by the bilinear interpolation layers to produce the large-size feature maps. A concatenated layer is employed to combine the feature maps, and a deconvolutional layer is used to recover the detailed information, to fuse the feature maps and to determine the final estimation. This architecture is our proposed DSNet.

IV. PERFORMANCE OF DSNET

As mentioned above, two networks are proposed in this paper, DSNet-fast (Fig. 1) and DSNet-accurate (Fig. 2). These two architectures can be used to process different resolution images. In this section, we report the experimental results of the proposed network on CamVid and Cityscapes datasets. Also, we compare them with the other state-of-the-art networks to examine the effectiveness of the proposed method.

A) Results on CamVid

In this subsection, the CamVid dataset is used to evaluate the performance of DSNet. In addition to DSNet-fast, the DSNet-accurate is employed to process the small resolution images. Here, both DSNet-fast and DSNet-accurate adopt the pre-trained encoder on ImageNet. After pre-training, the data augmentation strategy (horizontal flip and pixel translation) is employed to produce the robust prediction. Also, we find that decaying the learning rate by equation (1) can slightly improve the accuracy.

The results are shown in Table 6. DSNet-accurate sacrifices the inference speed but its accuracy is higher than DSNet-fast for about 4% mIoU. In addition, Table 7 shows that the number of parameters in DSNet-accurate is less than DSNet-fast due to the elimination of a convolution layer in the decoder. Here, we already know that the parameters used in the decoder show less effect on accuracy improvement. Also, the number of the parameters used in first few layers is identical in both DSNet-fast and DSNet-accurate, which indicates the size of feature maps is supposed to be more important than using extra parameters. Thus, the results in Table 6 are consistent with our observation in Section III that the feature map size has a significant influence on the accuracy in semantic segmentation.

On the other hand, compared to the other state-of-the-art methods, DSNet-accurate shows an outstanding performance in processing the low-resolution images (480 × 360). Furthermore, according to the experimental results in Fig. 10, we find that DSNet-accurate is indeed capable of retaining more detailed information and capturing the small objects compared to DSNet-fast. Table 7 lists the frame rate of DSNet-fast and DSNet-accurate. Both of them process a 480 × 360 RGB image for more than 58 frames/s, which demonstrates the real-time testing (inferencing) ability. Thus, if the computing device is sufficiently powerful or the input size is small, the DSNet-accurate is also an appropriate architecture to balance speed and accuracy for the real-time applications.

Table 6. Comparison of DSNet and other schemes on CamVid test set.

Method	mIoU (%)	Global Acc. (%)
DeepLab-LFOV [8]	61.6	–
Bayesian SegNet [14]	63.1	86.9
Dilation8 [23]	65.3	79.0
EDANet [24]	66.4	90.8
FC-DenseNet103 [25]	66.9	91.5
ICNet [26]	67.1	–
G-FRNet [27]	68.0	–
DCDN [28]	68.4	91.4
SDN [29]	71.8	92.7
DSNet-fast	68.6	91.7
DSNet-accurate	72.6	92.7

Table 7. The speed of DSNet running on 480 × 360 resolution with 11 categories (CamVid dataset).

Method	Frame rate (fps)	Model size (MB)
DSNet-fast	81.9	11.9
DSNet-accurate	58.2	11.6

B) Results on Cityscapes

We also tested our systems on the Cityscapes dataset. For the speed consideration, only DSNet-fast is benchmarked on this high-resolution dataset. At the training step, we resize the image and its ground truth (map) to 1024 × 512 in order to speed up the network training. During inference, the input image is at 1024 × 512 resolution, but the output segmentation maps are up-sampled to the full resolution (2048 × 1024) corresponding to their ground truths for evaluation. In addition to decaying the learning rate by (1), we find that adjusting the weight decay from 0.0005 to 0.0001 and adopting the dropout strategy [30] at the end of every dense unit with a drop rate of 0.1 can further improve the accuracy for DSNet-fast. Also, the pre-trained encoder and the data augmentation strategy are used in training to strengthen the feature representation. The best model of DSNet-fast can achieve 71.5% mIoU on the validation set. Some output samples are displayed in Fig. 11.

Furthermore, we compare DSNet-fast with the other state-of-the-art networks on the Cityscapes test set by submitting the test results to the online benchmark server. At the end, DSNet-fast achieves 69.1% mIoU, as Table 8 shows. For the top-ranked methods, the architectures are more complex than DSNet-fast. Also, a large amount of data is included in their training procedure, resulting in the better generalization and higher performance. Although the accuracy of DSNet is still lower than some high accuracy networks, DSNet is rather fast and accurate in competing with the efficient networks. It only takes 18.9 ms per image on a Titan X and 14.7 ms per image on a 1080Ti, for 1024 × 512 resolution inputs. The name of our method on the leaderboard is NCTU-ITRI.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

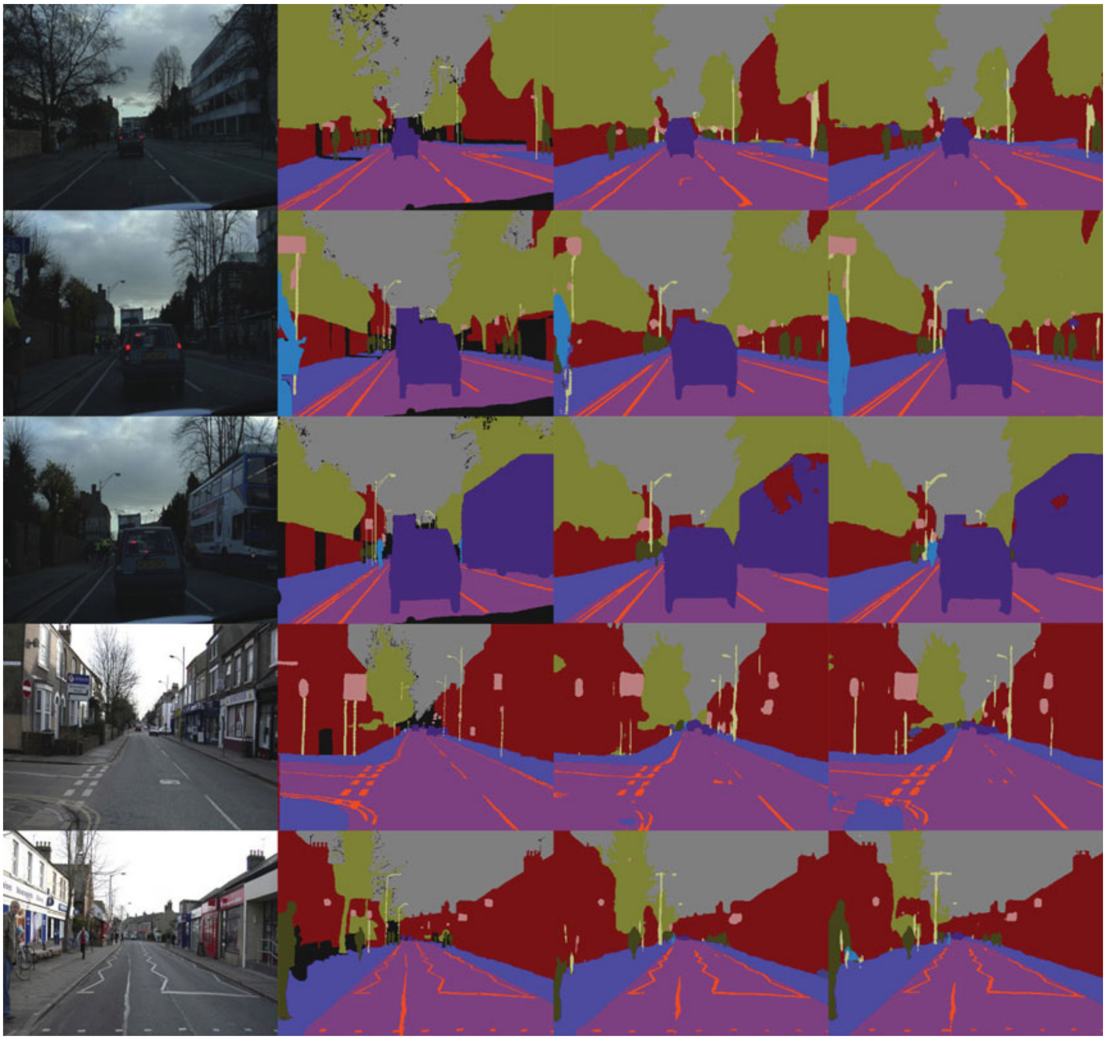

Fig. 10. The results of DSNet on CamVid test set. From left to right: (a) Input image, (b) Ground truth, (c) DSNet-fast output, (d) DSNet-accurate output.

Table 8. The results of DSNet-fast and other methods on Cityscapes test set. The results of other methods are listed according to the online leaderboard and their reference papers (Cityscape webpage).

High-speed network						
					Runtime (s)	
Method	Cityscapes data	Input downscaling factor	Additional data	mIoU (%)	Titan X	1080 Ti
SegNet [7]	Fine	4	ImageNet	56.1	0.0600	–
ENet [12]	Fine	2	–	58.3	0.0130	–
SQ [31]	Fine	no	ImageNet	59.8	0.0600	–
ESPNet [21]	Fine	2	–	60.3	0.0089	–
ESPNetv2 [32]	Fine	2	ImageNet	62.1	0.0120	–
ContextNet [22]	Fine	no	–	66.1	0.0238	–
EDANet [24]	Fine	2	–	66.3	0.0123	0.0092
EDANet [24]	Fine, Validation	2	–	67.3	0.0123	0.0092
ERFNet [13]	Fine	2	–	68.0	0.0240	
ICNet [26]	Fine, Validation	no	ImageNet	69.5	0.0330	–
ERFNet [13]	Fine	2	ImageNet	69.7	0.0240	–
DSNet-fast	Fine	2	ImageNet	69.1	0.0189	0.0147

V. CONCLUSION

In general, a deep learning model usually has high performance (accuracy) but often has a low inference speed. This makes the deep learning based methods difficult to apply into a real-world application. To solve the problem, we modify the network architecture based on the FCN and DenseNet.

In order to find an efficient trade-off between accuracy and speed, we conduct a series of experiments. We explore

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

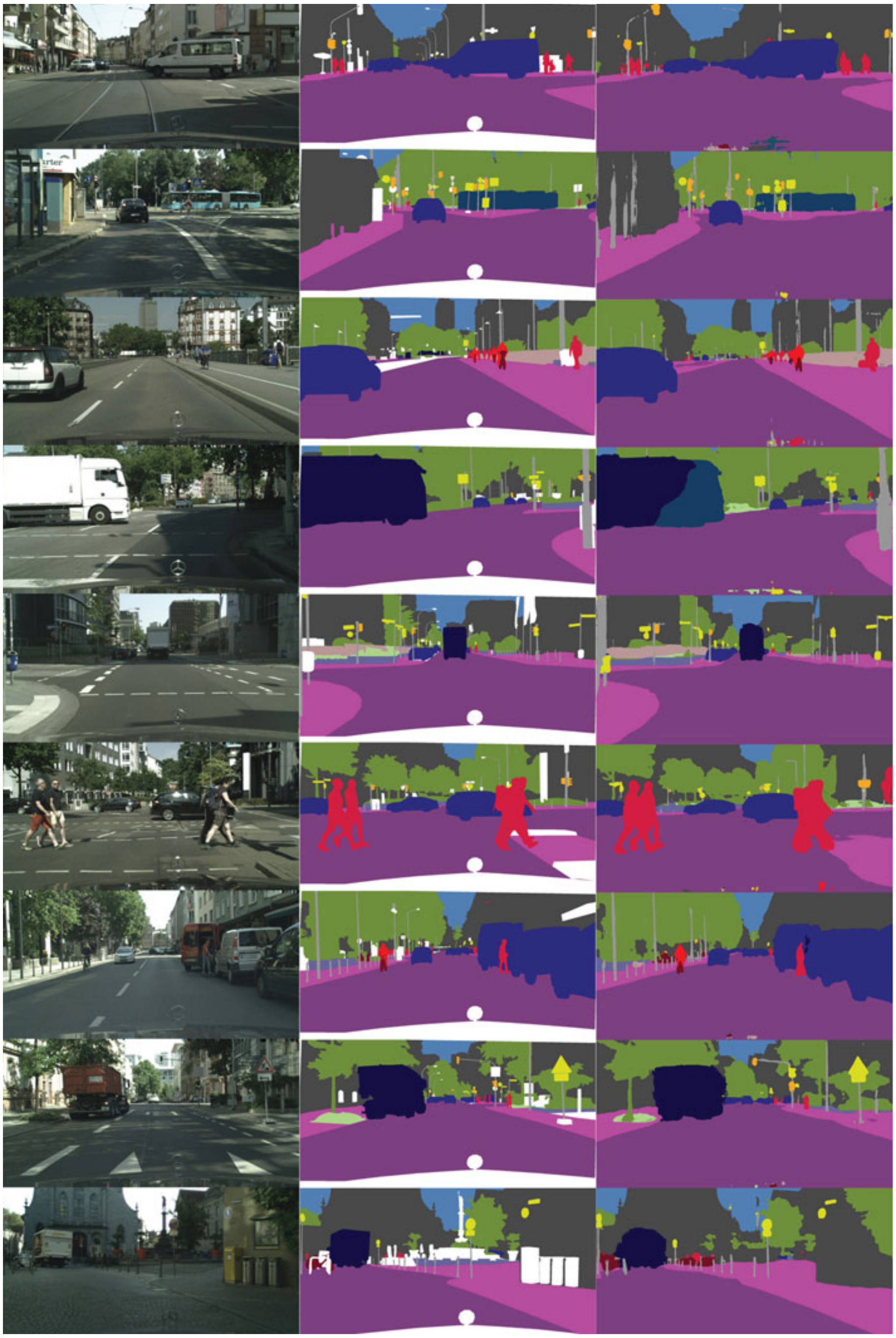

Fig. 11. Results of DSNet on Cityscapes validation set. From left to right: (a) Input image, (b) Ground truth, (c) DSNet-fast output.

a number of the encoder variations, examine the impact of input resolution, and the structure and the depth of a neural net. Next, we look into the fusion methods in the decoder, and ways to simplify the decoder. Finally, we propose an architecture that is able to process 1024 × 512 resolution images at 68 fps on a single 1080 Ti GPU card. In addition, our proposed architecture shows the good results on the two challenging road-scene datasets, CamVid and Cityscapes. This demonstrates that the proposed architecture is able to achieve a high speed and rather high accuracy processing.

ACKNOWLEDGEMENT

The authors thank Shao-Yuan Lo for his helpful discussions during the course of this project.

FINANCIAL SUPPORT

This work was supported in part by the Mechanical and Mechatronics Systems Research Lab., ITRI, Taiwan under Grant 3000547822.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

REFERENCES

[1] Chen, P.-R.; Hang, H.-M.; Chan, S.-W.; Lin, J.-J.: DSNet: an efficient CNN for road scene segmentation, in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conf. (APSIPA ASC)*, Lanzhou, China, 2019.

[2] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.

[3] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep residual learning for image recognition, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, 770–778.

[4] Huang, G.; Liu, Z.; Weinberger, K.Q.; van der Maaten, L.: Densely connected convolutional networks, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, 2017, 2261–2269.

[5] Noh, H.; Hong, S.; Han, B.: Learning deconvolution network for semantic segmentation, in *IEEE Int. Conf. Computer Vision (ICCV)*, Santiago, Chile, 2015, 1520–1528.

[6] Long, J.; Shelhamer, E.; Darrell, T.: Fully convolutional networks for semantic segmentation, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, 2015, 3431–3440.

[7] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, **39** (2017), 2481–2495.

[8] Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, arXiv preprint, arXiv:1606.00915, 2016.

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

[10] Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J.: Pyramid scene parsing network, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017, 2881–2890.

[11] Chen, L.-C.; Papandreou, G.; Schroff, F.; Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation, arXiv preprint, arXiv: 1802.02611, 2018.

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

[13] Romera, E.; Alvarez, J.M.; Bergasa, L.M.; Arroyo, R.: Erfnet: efficient residual factorized convnet for real-time semantic segmentation. *IEEE Trans. Intell. Transp. Syst.*, **19** (2018), 263–272.

[14] Kendall, A.; Badrinarayanan, V.; Cipolla, R.: Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, arXiv preprint, arXiv:1511.02680, 2015.

[15] Russakovsky, O., *et al.*: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, **115** (2015), 211–252.

[16] He, K.; Zhang, X.; Ren, S.; Sun, J.: Identity mappings in deep residual networks, in *European Conf. Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016, 630–645.

[17] Ioffe, S.; Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint, arXiv:1502.03167, 2015.

[18] Brostow, G.J.; Fauqueur, J.; Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.*, **30** (2009), 88–97.

[19] Cordts, M., *et al.*: The cityscapes dataset for semantic urban scene understanding, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, 3213–3223.

[20] Paszke, A., *et al.*: Automatic differentiation in PyTorch, in *Conf. Neural Information Processing Systems (NIPS)*, Long Beach, USA, 2017.

[21] Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H.: ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, arXiv preprint, arXiv:1803.06815, 2018.

[22] Poudel, R.P.; Bonde, U.; Liwicki, S.; Zach, C.: Contextnet: Exploring context and detail for semantic segmentation in real-time, arXiv preprint, arXiv:1805.04554, 2018.

[23] Yu, F.; Koltun, V.: Multi-scale context aggregation by dilated convolutions, arXiv preprint, arXiv:1511.07122, 2015.

[24] Lo, S.-Y.; Hang, H.-M.; Chan, S.-W.; Lin, J.-J.: Efficient dense modules of asymmetric convolution for real-time semantic segmentation, arXiv preprint, arXiv:1809.06323, 2018.

[25] Jégou, S.; Drozdzal, M.; Vazquez, D.; Romero, A.; Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, USA, 2017, 1175–1183.

[26] Zhao, H.; Qi, X.; Shen, X.; Shi, J.; Jia, J.: Icnet for real-time semantic segmentation on high-resolution images, in *European Conf. Computer Vision (ECCV)*, Munich, Germany, 2018, 405–420.

[27] Islam, M.A.; Rochan, M.; Bruce, N.D.; Wang, Y.: Gated feedback refinement network for dense image labeling, in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017, 4877–4885.

[28] Fu, J.; Liu, J.; Wang, Y.; Lu, H.: Densely connected deconvolutional network for semantic segmentation, in *IEEE Int. Conf. Image Processing* (ICIP), Beijing, China, 2017, 3085–3089.

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

[30] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15** (2014), 1929–1958.

[31] Treml, M., *et al.*: Speeding up semantic segmentation for autonomous driving, in *Conf. Neural Information Processing System (NIPS)*, Barcelona, Spain, 2016.

[32] Mehta, S.; Rastegari, M.; Shapiro, L.; Hajishirzi, H.: ESPNetv2: A lightweight, power efficient, and general purpose convolutional neural network, arXiv preprint, arXiv:1811.11431, 2018.

Ping-Rong Chen received his master degree in 2018 from the Electronics Engineering Department of National Chiao Tung University in Taiwan. He is currently serving as an engineer in Himax Technologies, Inc. His main work is to develop the lightweight and powerful deep learning algorithm, which can be deployed on the ultra-low power platform.

Hsueh-Ming Hang received the B.S. and M.S. degrees from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1978 and 1980, respectively, and Ph.D. from Rensselaer Polytechnic Institute, Troy, NY, in 1984. From 1984 to 1991, he was with AT&T Bell Laboratories, Holmdel, NJ, and then he joined NCTU in 1991. He served as the Dean of the ECE College at NCTU (2014–2017). He has been actively involved in the international MPEG standards and his current research interests include spherical image/video processing, and deep-learning based image/video processing. He has published over 200 technical papers. Dr. Hang was an associate editor of the

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press

IEEE Transactions on Image Processing and Transactions on Circuits and Systems for Video Technology. He was a Board Member of the Asia-Pacific Signal and Information Processing Association (APSIPA) (2013–2018). He is a recipient of the IEEE Third Millennium Medal and is a Fellow of IEEE and IET.

Sheng-Wei Chan received his master degree in 2013 from the Institute of Electrical and Control Engineering of National Chiao Tung University in Taiwan. He is currently serving as a project leader and engineer in the Mechanical and Mechatronics Systems Research Labs (MMSL) of Industrial Technology Research Institute (ITRI). His main work is to develop the real-time visual perception modules of self-driving vehicles which already validation on various vehicles. And one of the vehicle obtained the first self-driving test license in Taiwan.

Jing-Jhih Lin received his master degree in 2016 from the Power Mechanical Department of National Tsing Hua University in Taiwan. He is currently serving as an engineer at Industrial Technology Research Institute. His main work is to develop safety decision making for autonomous vehicles.

https://doi.org/10.1017/ATSIP.2020.25 Published online by Cambridge University Press