Dense Dual-Path Network for Real-time Semantic Segmentation

Xinneng Yang[1], Yan Wu*[,1], Junqiao Zhao[1,2], and Feilin Liu[1]

[1] College of Electronics and Information Engineering, Tongji University, Shanghai, China yanwu@tongji.edu.cn
[2] Institute of Intelligent Vehicle, Tongji University, Shanghai, China

Abstract. Semantic segmentation has achieved remarkable results with high computational cost and a large number of parameters. However, real-world applications require efficient inference speed on embedded devices. Most previous works address the challenge by reducing depth, width and layer capacity of network, which leads to poor performance. In this paper, we introduce a novel Dense Dual-Path Network (DDPNet) for real-time semantic segmentation under resource constraints. We design a light-weight and powerful backbone with dense connectivity to facilitate feature reuse throughout the whole network and the proposed Dual-Path module (DPM) to sufficiently aggregate multi-scale contexts. Meanwhile, a simple and effective framework is built with a skip architecture utilizing the high-resolution feature maps to refine the segmentation output and an upsampling module leveraging context information from the feature maps to refine the heatmaps. The proposed DDPNet shows an obvious advantage in balancing accuracy and speed. Specifically, on Cityscapes test dataset, DDPNet achieves 75.3% mIoU with 52.6 FPS for an input of 1024×2048 resolution on a single GTX 1080Ti card. Compared with other state-of-the-art methods, DDPNet achieves a significant better accuracy with a comparable speed and fewer parameters.

1 Introduction

Semantic segmentation is a fundamental task in computer vision, the purpose of which is to assign semantic labels to each image pixel. It has many potential applications in the fields of autonomous driving, video surveillance, robot sensing and so on. Existing methods mainly focus on improving accuracy. However, these real-world applications require efficient inference speed on high-resolution images.

Previous works[1–6] have already obtained outstanding performances on various benchmarks[7–11]. By analyzing existing state-of-the-art semantic segmentation methods, we find the keys to achieving high accuracy. 1) The backbone of these methods has a powerful feature extraction capability, such as ResNet[12], ResNeXt[13], DenseNet[14], which is usually pre-trained on ImageNet[15]. These backbones have a strong generalization capability and thus are adapted to many computer vision tasks. 2) These methods aggregate multi-scale context information sufficiently. There are many objects in semantic segmentation that are

difficult to be distinguished only by their appearance, such as 'field' and 'grass', 'building' and 'wall'. Due to multiple scales, occlusion and illumination, some objects like 'car' and 'person' require multi-scale context information to be correctly identified. To address above issues, dilated convolution and pooling operation are often used to enlarge the receptive field. Even though the theoretical receptive field is large enough, it still can't fully exploit the capability of global context information. Therefore, some approaches[1, 4, 6, 16, 17] aggregate multi-scale contexts via fusing feature maps generated by parallel dilated convolutions and pooling operations to robustly segment objects at multiple scales. 3) These methods recover spatial information effectively. Downsampling enlarges the receptive field and decreases the size of feature maps. It enriches high-level features, but loses spatial details. However, detailed spatial information is essential for semantic segmentation. In order to preserve spatial information, most works[1, 3, 4, 6] remove the last two pooling operations and replace the subsequent convolutions with dilated convolutions to keep the receptive field unchanged at the expense of computational efficiency. Unlike them, [2, 5, 18–22] utilize upsampling methods and self-designed skip connection to refine the boundaries of objects and small objects.

Based on the above analysis, we summarize the keys to achieving high accuracy in semantic segmentation as follows:

- Backbone with a powerful feature extraction capability.
- Sufficient aggregation of context information.
- Effective recovery of spatial information.

Recently, real-time semantic segmentation methods[23–28] have shown promising results. [29, 30] reduce the input size to accelerate the model, while easily losing the spatial details around boundaries and small objects. [31, 32] remove some of downsampling operations to create an extremely small model. Nevertheless, the receptive field of these models is not sufficient to cover large objects, resulting in a poor performance. To achieve real-time speed, some works[23, 25, 31–33] design a specialized network for semantic segmentation as backbone. Differently, some works[24, 26–28, 34] adopt a light-weight classification network as backbone, such as MobileNets[35–37], ShuffleNets[38, 39], ResNet-18[12]. Convolution factorization refers to the decomposition of a large convolution operation into several smaller operations, such as factorized convolution and depth-wise separable convolution, which is widely adopted in these backbones to reduce the computational cost and the number of parameters. However, convolution factorization is not conducive to GPU parallel processing, which results in a much slower inference speed under the same computational budget. On the other hand, these backbones have a limited capability due to fewer convolution operations and feature maps. Recent works[27, 30, 40] propose a two-column network which consists of a deep network for encoding context information and a shallow network for capturing spatial information. However, the extra network on high-resolution images limits the inference speed, and the independence between networks limits the performance of the model.

To strike a better balance between accuracy and efficiency, we follow the principle of simplicity in designing the model. A complicated and sophisticated work may improve accuracy, but in most cases it hurts efficiency significantly. A simple and clean framework can make it easier to re-implement and improve. Therefore, we propose a light-weight yet powerful backbone and a simple yet effective framework for fast and accurate semantic segmentation. The proposed Dense Dual-Path Network (DDPNet) achieves 75.3% mIoU with merely 2.53 M parameters on Cityscapes test dataset. It can run on high-resolution images (1024×2048) at 52.6 FPS on a single GTX 1080Ti card. DDPNet is superior to most of the state-of-the-art real-time semantic segmentation methods in accuracy and speed, and requires fewer parameters.

Our main contributions are summarized as follows:

- We design a light-weight and powerful backbone with dense connectivity to facilitate feature reuse throughout the whole network and the proposed Dual-Path module (DPM) to sufficiently aggregate multi-scale contexts.
- We propose a simple and effective framework with a skip architecture utilizing the high-resolution feature maps to refine the segmentation output and an upsampling module leveraging context information from the feature maps to refine the heatmaps.
- We conduct a series of experiments on two standard benchmarks, Cityscapes and CamVid, to investigate the effectiveness of each component of our proposed DDPNet and compare accuracy and efficiency with other state-of-the-art methods.

2 Related Work

Recently, FCN[20] based methods have greatly improved the performance of semantic segmentation. Most of them focus on encoding content information and recovering spatial information.

Real-time Segmentation: The goal of real-time semantic segmentation is to achieve the best trade off between accuracy and efficiency. In order to reach a real-time speed, SegNet[41] and U-Net[42] perform multiple downsampling operations to significantly reduce the feature map size. SegNet designs a symmetric encoder-decoder architecture to carefully recover feature maps. U-Net proposes a symmetric architecture with skip connection to enable precise localization. Differently, E-Net[32] constructs an extremely light-weight network with fewer downsampling operations to boost the inference speed. ERFNet[33] focuses on a better accuracy with a deeper network that uses residual connection and factorized convolution. ESPNet[31] proposes a light-weight network with efficient spatial pyramid module. ICNet[30] uses an image cascade network to capture objects of different sizes from multi-scale images. BiSeNet[27] designs a spatial path to preserve spatial information and a context path to obtain sufficient receptive field. Based on multi-scale feature propagation, DFANet[24] reduces the number of parameters and maintains high accuracy.

Context Information: Semantic segmentation needs to sufficiently obtain context information to correctly identify objects that are similar in appearance but belong to different categories and objects that are different in appearance but belong to the same category. Most works capture diverse context information by using different dilation convolutions to enlarge the receptive field. DeepLab[16] proposes an atrous spatial pyramid pooling module to aggregate multi-scale contexts. In the follow-up work, [17] further improves performance by extending the previously proposed atrous spatial pyramid pooling module with a global average pool, which is able to capture the global context of images. Similarly, PSPNet[6] proposes a pyramid pooling module which consists of different sizes of average pooling operations to aggregate multi-scale contexts. [43] designs a scale-adaptive convolution to acquire flexible-size receptive fields. PAN[44] combines attention mechanism and spatial pyramid to learn a better feature representation. DMNet[4] adaptively captures multi-scale contents with multiple dynamic convolutional modules arranged in parallel.

Spatial Information: Semantic segmentation requires spatial details to restore the boundaries of objects and small objects. The reason for the loss of spatial details is downsampling operations in the convolutional network. Downsampling is essential for convolutional networks because it can reduce the size of feature maps and enlarge the receptive field. Most works reduce the number of downsampling operations to preserve spatial details, which leads to slow inference speed. Differently, [21, 32, 33, 41] construct an encoder-decoder structure with unpooling and deconvolution to recover spatial details. However, this structure still can not effectively recover the loss of spatial details and have high computational complexity. Skip connection is first introduced in FCN[20], which combines semantic information from a deep layer with appearance information from a shallow layer to produce accurate and detailed segmentation result. Based on FCNs, RefineNet[19] presents a multi-path refinement network that refines high-level semantic features using long-range residual connection. BiseNet[27] and Fast-SCNN[40] explicitly acquire spatial information at a fast inference speed using a light-weight spatial path.

3 Dense Dual-Path Network

In this section, we introduce the backbone of DDPNet and the proposed framework for real-time semantic segmentation. Furthermore, we elaborate the design choices and motivations in detail.

3.1 Dense Connectivity

The backbone of DDPNet is a variant of densely connected convolutional network, which adopts dense connectivity to stack convolution operations. Dense connectivity is originally proposed by DenseNet[14], in which each layer has direct connections to all subsequent layers. Formally, the m^{th} layer takes feature

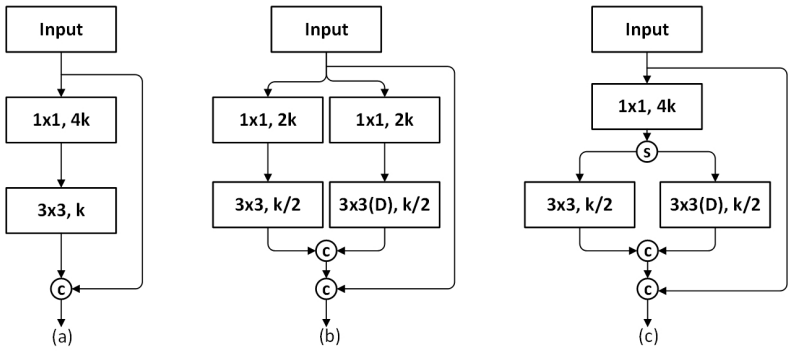

Fig. 1. Depiction of the dense layer originally proposed in DenseNet[14], and our proposed Dual-Path module (DPM). "k": the growth rate. "D": dilated convolution. "c": channel concatenate operation. "s": channel split operation. In the convolutional blocks, "$s1 \times s2, n_o$" indicates their kernel sizes (s_1, s_2) and the number of output feature maps (n_o). (a) Original dense layer. (b-c) Dual-Path Module. For brevity, we omit normalization and activation function.

maps from all preceding layers as input:

$$x_m = H_m([x_0, x_1, ..., x_{m-1}]) \tag{1}$$

where x_n, $n \in [1, 2, ..., m]$ refers to the output of n^{th} layer. x_0 refers to the input of dense block, bracket indicates concatenation operation, and H is a composite function of three consecutive operations: batch normalization[45], followed by a rectified linear unit[46] and a convolution operation.

Some works[47, 48] utilize dense connectivity in the network to boost the performance of semantic segmentation. However, they focus on accuracy rather than speed. Dense connectivity allows for the reuse of features throughout the network and significantly reduces the number of parameters, which contributes to the implementation of light-weight models. Therefore, we use dense connectivity to build a light-weight and powerful backbone for real-time semantic segmentation. Following PeleeNet[49], we use post-activation as composite function, which conducts convolution operation before batch normalization, for improving inference speed.

3.2 Dual-Path Module

Dual-Path module (DPM) is the basic unit of dense block. Motivated by the diversity of object scales in semantic segmentation, we propose a specific Dual-Path module composed of two bottleneck layers in parallel. One of the bottleneck layers uses a point-wise convolution to reduce the number of input feature maps, followed by a 3×3 convolution to produce $\frac{k}{2}$ new feature maps. k refers to the growth rate. The other bottleneck layer replaces the 3×3 convolution with

dilation convolutions to learn visual patterns for large objects. The structural details are shown in Fig. 1(b). We refer this structure as DPM-b. Notice that the point-wise convolutions in both branches have the same input. Therefore, we combine the two point-wise convolutions into one, and split the output of the convolution into two independent inputs for the two branches, as depicted in Fig. 1(c). We refer this structure as DPM-c. The two implementations are functionally identical but differ in efficiency. DPM-c is more efficient than DPM-b in GPU parallel processing. For this reason, we adopt the structure of DPM-c as our final implementation of DPM. Finally, the output of two branches is concatenated, followed by a dropout layer[50].

As can be seen from Fig. 1, Dual-Path module has a larger effective receptive field than the original dense layer. With an extra dilated branch in dense layer, Dual-Path module can extract features from different scales. Intuitively, more branches can aggregate multi-scale contexts more effectively, such as ASPP in [16]. However, the decomposition of a single convolution into multiple parallel convolutions is not conducive to the acceleration of the model. Due to the ability to effectively aggregate feature maps at different scales, Dual-Path module significantly improves the capacity of backbone.

3.3 Backbone Design

In this subsection, we discuss the main components and algorithms used to build the backbone of Dense Dual-Path Network. In this work, we aim to design an architecture that gets the best possible trade-off between accuracy and efficiency. Many approaches that focus on designing a light-weight architecture largely adopt depth-wise separable convolution and factorized convolution which lack efficient implementation. Instead of using depth-wise separable convolution or factorized convolution, Dense Dual-Path Network is build with traditional convolution.

Transition Layer. Transition layer is used to reduce the size of feature maps and compress the model. It is composed of a point-wise convolution followed by a 2×2 average pooling layer. In order to fully exploit dense connectivity, DDPNet keeps the number of input channels and output channels the same in all transition layers. This design facilitates feature reuse throughout the whole network.

Initial Block. Initial block is used to reduce the input size, which typically involves several downsampling operations. Due to direct operation on the original image, initial block is often computationally expensive. However, a well-designed initial block can effectively improve feature expression and preserve rich spatial information. The initial block of DPPNet is motivated by ENet[32] and PeleeNet[49], which preserves rich spatial information and takes full advantage of feature reuse. In our initial block, a 3×3 convolution with stride 2 is performed on the original image, followed by two branches. One of the branches is a 3×3 convolution with stride 2. The other branch is a 2×2 max pooling layer with stride 2. Finally, the output of two branches is concatenated.

Table 1. The backbone of our proposed DDPNet. "$DenseLayer\,(or DPM) \times n_d, k$" indicates the operation in dense block is the original dense layer ($DenseLayer$) or the proposed Dual-Path module (DPM), the number of layers (n_d) and the growth rate (k). Input size is ($1024 \times 2048 \times 3$).

Type	**Operator**	**Output Shape**
Initial Block		256 x 512 x 64
Dense Block	Dense Layer x 2, k=32	256 x 512 x 128
Transition Layer	1 x 1 conv, stride 1 2 x 2 average pool, stride 2	128 x 256 x 128
Dense Block	Dense Layer x 4, k=32	128 x 256 x 256
Transition Layer	1 x 1 conv, stride 1 2 x 2 average pool, stride 2	64 x 128 x 256
Dense Block	DPM x 8, k=32	64 x 128 x 512
Transition Layer	1 x 1 conv, stride 1 2 x 2 average pool, stride 2	32 x 64 x 512
Dense Block	DPM x 8, k=32	32 x 64 x 768
Transition Layer	1 x 1 conv, stride 1	32 x 64 x 768

Architecture Detail. The backbone of our proposed DDPNet is shown in Table 1. The entire architecture consists of an initial block and four dense blocks followed by a transition layer. To maintain a better balance between accuracy and computational cost, DDPNet first reduces spatial resolution twice in the initial block and performs downsampling operation in each transition layer (except for the last one). Except for the last block, DDPNet doubles the number of feature maps in each dense block.

In DenseNet[14], each layer produces k new feature maps, where k refers to as the growth rate. The growth rate is usually a small constant due to feature reuse. With a fixed number of output channels and a fixed growth rate, we can get the number of layers in a certain block. For example, the n^{th} block has $\frac{(n_{out}-n_{in})}{k}$ layers, where n_{out} is the number of channels at the end of the block, and n_{in} is the number of channels in the input layer. As mentioned in [14, 51], convolution layers in a deeper block tend to rely more on high-level features than on low-level features. Based on this observation, DDPNet produces more new feature maps in deeper blocks, which means that more layers are needed in deeper blocks.

Since the feature maps from a higher layer contain more semantic information and less spatial information, we adopt the original dense layer in the first two blocks and replace the original dense layer with Dual-Path module in the last two blocks. We explore further in the experiment section. A larger growth rate requires fewer layers to generate new feature maps, which can boost the

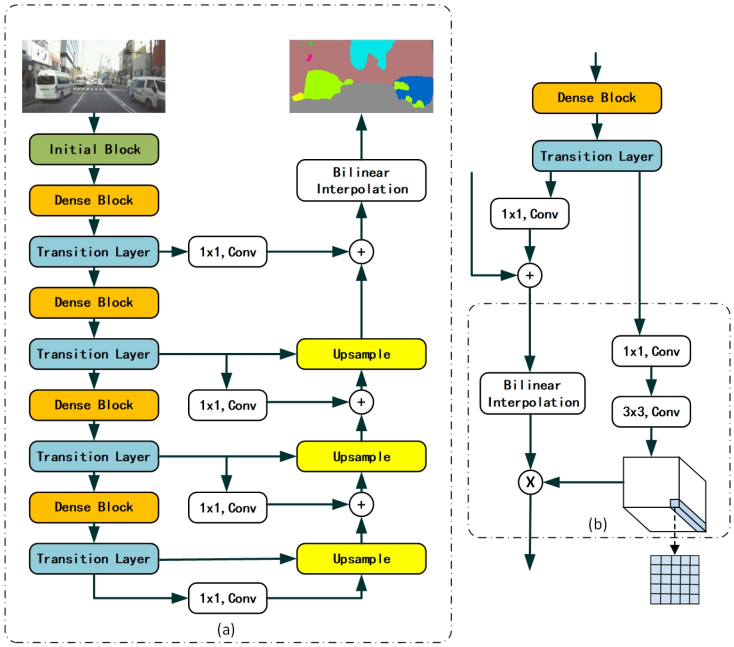

Fig. 2. (a) Framework of the proposed DDPNet. (b) Architecture details of the usampling module. Note that the feature maps used to generate heatmaps are the output of the point-wise convolution in transition layer before downsampling. For brevity, we refer to these feature maps as the output of transition layer in this diagram. "+": element-wise addition. "×": weighted sum operation.

inference speed. However, a smaller growth rate forms a denser connections, which improves the quality of feature maps. To strike a better balance between accuracy and efficiency, we set the growth rate to 32 in DDPNet.

3.4 Framework for Real-time Semantic Segmentation

In this subsection, we propose a simple and effective framework for real-time semantic segmentation. Fig. 2(a) shows the overall framework. The proposed backbone and the framework are adopted to construct the DDPNet. The backbone can be other classification networks, such as ResNet[13]. Most of the frameworks used for real-time semantic segmentation adopt an encoder-decoder architecture. The encoder is used to provide information for the final classification and reduce the computational cost by downsampling. The main purpose of the decoder is to recover the spatial details lost during the downsampling process.

Skip Architecture. DDPNet employs an asymmetric sequential architecture, where an extreme light-weight decoder is adopted to upsample the feature maps to match the input resolution. The decoder of DDPNet is a skip architecture, which utilizes the high-resolution feature maps to refine the segmentation

output. In U-Net[42], skip connection is performed on the feature space, which is very computational expensive because it is directly affected by the number of channels. To accelerate the inference speed, DDPNet adopts skip connection on heatmaps in the label space, similar to FCN[20]. The output of each transition layer is followed by a point-wise convolution to map from feature space to label space, resulting in a heatmap corresponding to each class. Note that the feature maps used to generate heatmaps are the output of the point-wise convolution in transition layer before downsampling. For convenience, we refer to these feature maps as the output of transition layer in the following paper. A low-resolution heatmap is upsampled by an upsampling module, and then element-wise addition is performed between this heatmap and a high-resolution heatmap. After three skip connections, a bilinear interpolation with stride 4 is adopted to produce the final output. Fig. 2(a) shows the overall framework of DDPNet.

Upsampling Module. The commonly used upsampling methods in semantic segmentation are bilinear interpolation and deconvolution. Bilinear interpolation is computationally cheap, but only leverages the distance information between pixels. Deconvolution uses fixed kernels to perform upsampling, which has been proved to be limited and inefficient. The upsampling module of DDPNet is modified from [22] to save a lot of computation by performing upsampling on label space instead of upsampling on feature space. More specifically, DDPNet uses a bilinear interpolation to upsample the heatmap and refine the upsampled heatmap with a dynamic filtering layer. Dynamic filtering layer is originally proposed in dynamic filter networks[52]. In dynamic filter networks, a dynamic filter module consists of a filter generating path that produces filters based on one input, and a dynamic filtering layer that applies the generated filters to another input. In the upsampling module of DDPNet, the filter generating path takes the output of transition layer as input, then compresses the input feature channel with a point-wise convolution. Finally, a 3×3 convolution followed by a pixel shuffle operation and a softmax is used to generate the filters. In dynamic filtering layer, the generated filters are applied to refine the upsampled heatmap, which is a weighted sum operation. Formally, the upsampling module can be written as

$$\mathbf{Y} = U(\mathbf{X_1}) \otimes \delta(S(N(F^1(w_{3\times3}, \sigma(N(F^2(w_{1\times1}, \mathbf{X_2}))))))) \tag{2}$$

where Y and X are the output and input of the upsampling module respectively. X_1 and X_2 refer to the input of dynamic filtering layer and filter generating path. U denotes bilinear interpolation, $\otimes$ indicates weighted sum operation, F and N represent convolution operation and batch normalization, S refers to pixel shuffle operation, σ and δ indicate ReLU and softmax activation. $w_{n\times n}$ is convolutional parameter and n represents kernel size. Fig. 2(b) shows the detailed architecture of the upsampling module.

4 Experiment

In this section, we evaluate the proposed DDPNet on Cityscapes[8] and CamVid [7] benchmarks. We first introduce the datasets and the implementation protocol. Then, we conduct a series of experiments on Cityscapes validation set to investigate the effectiveness of each component of our proposed method. Finally, we report the accuracy and efficiency results compared with other state-of-the-art methods. All accuracy results are reported using the commonly used mean Intersection-over-Union (IoU) metric. Runtime evaluations are performed on a single 1080Ti card. To eliminate the error fluctuation, we report an average of 5000 frames for the frames per second (FPS) measurement.

Cityscapes. The Cityscapes dataset is a large urban street scene dataset which is collected from 50 different cities. It contains 5000 fine annotated images, which are split into three sets: 2975 images for training, 500 images for validation, and 1525 images for testing. For fair a comparison, the 1525 test images is offered without ground-truth. There is another set of 19,998 images with coarse annotation, but we only use the fine annotated images for all experiments. All images have a resolution of 1024×2048, in which each pixel is annotated to pre-defined 19 classes.

CamVid. The CamVid dataset is another street scene dataset which is extracted from video sequences. It contains 701 images, which are split into three sets: 367 images for training, 101 images for validation, and 233 images for testing. All images have a resolution of 720×960 and 11 semantic categories. Following the general settings, we reduce the image resolution to 360×480 and use the training set and the validation set to train our model, then test it on the test set.

4.1 Implementation Protocol

We conduct all experiments using PyTorch with CUDA 10.0 and cuDNN backends on a single 1080Ti card. We use Adam optimizer [53] with batch size 8 and weight decay $2e^{-4}$ in training. For learning rate schedule, we use the learning rate warmup strategy suggested by [54] and the cosine learning rate decay policy. The learning rate l_i is computed as:

$$l_i = \frac{1}{2} \times \left(1 + \cos \frac{i \times \pi}{T}\right) \times l_{base} \tag{3}$$

where i refers to current epoch, the maximum number of epochs T is set to 350 for training on Cityscapes and 700 for training on CamVid, the initial learning rate l_{base} is $5e^{-4}$. As for data augmentation, we employ mean subtraction, random horizontal flip and random scale on the input images during training. A random parameter between [0.75, 2] is used to transform the images to different scales. Finally, we randomly crop the images into fixed size for training.

Table 2. Ablation studies for dense connectivity and DPM. All models are trained from scratch on Cityscapes training set and evaluated on Cityscapes validation set. The numbers in brackets denote the performance improvement over the baseline. The bold entries are the final settings for DDPNet.

Model	mIoU(%)	Time(ms)	FPS	Para	FLOPs	Memory
Baseline	72.8	9.5	105.7	2.7M	13.7G	333MB
ResNet[12]	70.9	9.8	101.6	11.2M	41.7G	287MB
PeleeNet[49]	72.5	11.7	85.3	2.1M	12.3G	406MB
+ $DPM \times 1$	74.6 (+1.8)	10.3	97.4	2.6M	13.5G	333MB
+ DPM × 2	75.5 (+2.7)	11.2	89.4	2.4M	12.8G	333MB
+ $DPM \times 4$	74.7 (+1.9)	11.7	85.5	2.4M	11.5G	333MB

4.2 Ablation Study

In this subsection, we perform a series of experiments to evaluate the effectiveness of each component in our proposed DDPNet. All ablation studies are trained on Cityscapes training set and evaluated on Cityscapes validation set. We reduce the image size to 768×1536 to accelerate the training process and evaluate the accuracy and efficiency of our models. For a fair comparison, we use the same training settings for all models.

Ablation Study for Dense Connectivity and DPM. We first explore the effects of dense connectivity and Dual-Path module specifically. A densely connected convolutional network without compression in transition layer is built as our backbone. The growth rate is set to 32. The proposed initial block and a skip architecture are adopted to construct a baseline for semantic segmentation. We replace the backbone with the frequently used ResNet-18[12] to make a comparison between two different connection mechanisms. As suggested by [54], we adopt a modified ResNet, which replace the 1×1 convolution with stride 2 in downsampling block with a 2×2 average pooling layer with stride 2 followed by a 1×1 convolution with stride 1. As can be seen from Table 2, our baseline is far more accurate (72.8% VS 70.9%) and efficient than ResNet. Due to feature reuse, our baseline has $4\times$ fewer parameters and $3\times$ fewer FLOPs than ResNet, which results in a much lower power consumption. To better understand the DPM, we replace the original dense layer with the proposed DPM stage by stage. Equipping the DPM in the last stage alone can boost the baseline by 1.8% (72.8% $\rightarrow$ 74.6%) in accuracy with a slight drop in inference speed. We continue to adopt the DPM in the third stage, which brings in 2.7% (72.8% $\rightarrow$ 75.5%) improvement to the baseline. However, adopting more DPMs at early stages does not yield further benefits. It verifies that feature maps from an early stage contain more spatial details, which is essential for further feature representation. Therefore, we adopt the original dense layer in the first two blocks and replace the original dense layer with the proposed DPM in the last two blocks as the backbone of DDPNet. Here, we adopt an increasing dilation rate in dense blocks to fully explore the ability to aggregate feature maps at different scales. For

Table 3. Ablation studies for skip architecture and upsampling module. All models are trained from scratch on Cityscapes training set and evaluated on Cityscapes validation set. The numbers in brackets denote the performance improvement or degradation over the baseline. The bold entries are the final settings for DDPNet.

Model	mIoU(%)	Time(ms)	FPS	Para	FLOPs	Memory
Baseline	69.7	10.7	93.4	2.4M	12.5G	326MB
+ Skip Architecture	75.5 (+5.8)	11.2 (+0.5)	89.4	2.4M	12.8G	333MB
+ Upsampling Module	76.2 (+6.5)	11.7 (+1.0)	85.4	2.5M	13.2G	412MB
+ U-Net Architecture	75.2 (+5.5)	11.7 (+1.0)	85.6	2.5M	13.7G	356MB
+ CARAFE[22]	76.2 (+6.5)	14.8 (+4.1)	67.4	2.5M	14.0G	601MB

example, the dilation rate sequence of the third dense block is set to {2, 4, 8, 16, 2, 4, 8, 16}. We also try to set all dilation rates to a fixed number, or increase them gradually in different ways, which all lead to a slight decrease in accuracy. We compare DDPNet with PeleeNet[49], which utilizes a different way to aggregate multi-scale representations. Table 2 shows that DDPNet is superior to PeleeNet (75.5% VS 72.5%).

Ablation Study for Skip Architecture and Upsampling Module. Here, we demonstrate the effectiveness of our proposed framework for real-time semantic segmentation. The proposed backbone is adopted as the encoder of baseline and the output heatmap is directly upsampled to the original image size, which leads to poor segmentation of boundaries and small objects. We compare two different decoder structures, skip architecture and U-Net architecture, which perform skip connection on label space and feature space respectively. As can be seen from Table 3, skip architecture is more efficient than U-Net architecture and has a comparable accuracy (75.5% VS 75.2%). By gradually restoring spatial information, skip architecture significantly improves the baseline by 5.8% (69.7% $\rightarrow$ 75.5%) in accuracy. Furthermore, we adopt the proposed upsampling module that leverages context information from feature maps to refine the heatmaps. DDPNet with the proposed framework boosts the baseline by 6.5% (69.7% $\rightarrow$ 76.2%) in accuracy with a negligible drop in inference speed (1.0 ms). We compare another upsampling method CARAFE[22], which performs upsampling on feature space. We directly adopt CARAFE in U-Net architecture and compare it with DDPNet. Table 3 shows that the proposed framework has a comparable accuracy, but 4 times faster (1.0 ms VS 4.1 ms). Some visual results and analyses are provided in the supplementary file.

4.3 Accuracy and Efficiency Analysis

In this subsection, we compare the proposed DDPNet with other existing state-of-the-art real-time segmentation models on Cityscapes dataset. For a fair comparison, we measure the mIoU without any evaluation tricks like multi-crop, multi-scale, etc.

Table 4. Accuracy and efficiency results on Cityscapes test dataset. "-" indicates that the corresponding result is not provided by the method. "†" indicates that the model is evaluated on Cityscapes validation set.

Method	Pretrain	Input Size	#Params	FLOPs	GPU	FPS	mIoU(%)
SegNet[41]	ImageNet	640 x 360	29.5M	286G	TitanX M	14.6	56.1
ENet[32]	No	512 x 1024	0.4M	4.4G	TitanX M	76.9	58.3
ESPNet[31]	No	512 x 1024	0.4M	4.7G	TitanX	112	60.3
ERFNet[33]	No	512 x 1024	2.1M	-	TitanX M	41.7	68.0
ICNet[30]	ImageNet	1024 x 2048	26.5M	29.8G	TitanX M	30.3	69.5
BiSeNet1[27]	ImageNet	768 x 1536	5.8M	14.8G	TitanX	72.3	68.4
BiSeNet2[27]	ImageNet	768 x 1536	49.0M	55.3G	TitanX	45.7	74.7
DABNet[23]	No	1024 x 2048	0.8M	-	1080Ti	27.7	70.1
DFANet[24]	ImageNet	1024 x 1024	7.8M	3.4G	TitanX	100	71.3
SwiftNet†[26]	No	1024 x 2048	11.8M	104G	1080Ti	39.9	70.4
SwiftNet[26]	ImageNet	1024 x 2048	11.8M	104G	1080Ti	39.9	75.5
ShelfNet[28]	ImageNet	1024 x 2048	-	-	1080Ti	36.9	74.8
DDPNet†	No	768 x 1536	2.52M	13.2G	1080Ti	85.4	76.2
DDPNet†	No	1024 x 2048	2.52M	23.5G	1080Ti	52.6	77.2
DDPNet	No	768 x 1536	2.52M	13.2G	1080Ti	85.4	74.0
DDPNet	No	1024 x 2048	2.52M	23.5G	1080Ti	52.6	75.3

The comparison of the accuracy (class mIoU) and efficiency (FLOPs, FPS) is shown in Table 4. Our DDPNet outperforms most existing real-time semantic segmentation methods in accuracy, and maintains a superior inference speed. Specifically, DDPNet achieves 75.3% mIoU with 52.6 FPS for an input of 1024×2048 resolution and 74.0% mIoU with 85.4 FPS for an input of 768×1536 resolution on Cityscapes test set. ShelfNet[28] and SwiftNet[26] are recent published state-of-the-art models. Compared to ShelfNet, DDPNet is faster (52.6 FPS VS 36.9 FPS) and more accurate (75.3% VS 74.8%). SwiftNet has a slightly better accuracy than DDPNet (75.5% VS 75.3%). However, DDPNet is trained with only Cityscapes fine-annotated images, without using any extra data. For a fair comparison, we compare the results of SwiftNet and DDPNet trained from scratch and evaluated on Cityscapes validation set. As can be seen from Table 4, DDPNet achieves significant better results in accuracy (77.2% VS 70.4%) and inference speed (52.6 FPS VS 39.9 FPS). Note that DDPNet has fewer parameters and smaller FLOPs than most methods. The reason is that most real-time semantic segmentation methods adopt ResNet-18 as their backbone, while DDPNet designs a light-weight and powerful backbone with fewer parameters.

4.4 Result on Other Dataset

To verify the generality of our proposed DDPNet, we conduct experiments on CamVid dataset. We modify the DDPNet to better fit the image resolution by

Table 5. Accuracy results on CamVid test dataset.

Method	Pretrain	#Params	mIoU (%)
SegNet[41]	ImageNet	29.5M	55.6
ENet[32]	No	0.4M	51.3
ICNet[30]	ImageNet	26.5M	67.1
BiSeNet1[27]	ImageNet	5.8M	65.6
BiSeNet2[27]	ImageNet	49.0M	68.7
DABNet[23]	No	0.8M	66.4
FPENet[25]	No	0.4M	65.4
DFANet[24]	ImageNet	7.8M	64.7
SwiftNet[26]	No	11.8M	63.3
SwiftNet[26]	ImageNet	11.8M	72.6
FC-HarDNet[55]	No	1.4M	62.9
DDPNet	No	1.1M	67.3
DDPNet	Cityscapes	1.1M	73.8

replacing the initial block with a 3×3 convolution and removing the last dense block. As can be seen from Table 5, DDPNet achieves impressive results with only 1.1 M parameters. Besides, we investigate the effect of the pre-training datasets on CamVid. The last two rows of Table 5 show that pre-training on Cityscapes can significantly improve the accuracy over 6.5% (67.3% $\rightarrow$ 73.8%).

5 Conclusion

In this paper, we propose a novel Dense Dual-Path Network (DDPNet) for real-time semantic segmentation on high-resolution images. The proposed DDPNet achieves a significant better accuracy with a comparable speed and fewer parameters than most real-time semantic segmentation methods.

6 Acknowledgement

This work is supported by the National Key Research and Development Program of China (No.2018YFB0105103, No.2017YFA0603104, No.2018YFB0505400), the National Natural Science Foundation of China (No.U19A2069, No.U1764261, No.41801335, No.41871370), the Shanghai Science and Technology Development Foundation (No.17DZ1100202, No.16DZ1100701) and the Fundamental Research Funds for the Central Universities (No.22120180095).

References

1. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). (2018) 801–818

2. Cheng, B., Chen, L.C., Wei, Y., Zhu, Y., Huang, Z., Xiong, J., Huang, T.S., Hwu, W.M., Shi, H.: Spgnet: Semantic prediction guidance for scene parsing. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 5218–5228
3. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3146–3154
4. He, J., Deng, Z., Qiao, Y.: Dynamic multi-scale filters for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 3562–3572
5. Tian, Z., He, T., Shen, C., Yan, Y.: Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3126–3135
6. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 2881–2890
7. Brostow, G.J., Fauqueur, J., Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. Pattern Recognition Letters **30** (2009) 88–97
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 3213–3223
9. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88** (2010) 303–338
10. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755
11. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 633–641
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
13. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1492–1500
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 4700–4708
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee (2009) 248–255
16. 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. IEEE transactions on pattern analysis and machine intelligence **40** (2017) 834–848
17. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)

18. Fu, J., Liu, J., Wang, Y., Li, Y., Bao, Y., Tang, J., Lu, H.: Adaptive context network for scene parsing. In: Proceedings of the IEEE international conference on computer vision. (2019) 6748–6757
19. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1925–1934
20. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3431–3440
21. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE international conference on computer vision. (2015) 1520–1528
22. Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D.: Carafe: Content-aware reassembly of features. arXiv preprint arXiv:1905.02188 (2019)
23. Li, G., Yun, I., Kim, J., Kim, J.: Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. arXiv preprint arXiv:1907.11357 (2019)
24. Li, H., Xiong, P., Fan, H., Sun, J.: Dfanet: Deep feature aggregation for real-time semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 9522–9531
25. Liu, M., Yin, H.: Feature pyramid encoding network for real-time semantic segmentation. arXiv preprint arXiv:1909.08599 (2019)
26. Orsic, M., Kreso, I., Bevandic, P., Segvic, S.: In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 12607–12616
27. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 325–341
28. Zhuang, J., Yang, J., Gu, L., Dvornek, N.: Shelfnet for fast semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. (2019) 0–0
29. Wu, Z., Shen, C., Hengel, A.v.d.: Real-time semantic image segmentation via spatial sparsity. arXiv preprint arXiv:1712.00213 (2017)
30. Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: Icnet for real-time semantic segmentation on high-resolution images. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 405–420
31. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L., Hajishirzi, H.: Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 552–568
32. 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)
33. Romera, E., Alvarez, J.M., Bergasa, L.M., Arroyo, R.: Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems **19** (2017) 263–272
34. Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H.: A comparative study of real-time semantic segmentation for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2018) 587–597

35. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. arXiv preprint arXiv:1905.02244 (2019)
36. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
37. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 4510–4520
38. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 116–131
39. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 6848–6856
40. Poudel, R.P., Liwicki, S., Cipolla, R.: Fast-scnn: fast semantic segmentation network. arXiv preprint arXiv:1902.04502 (2019)
41. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39** (2017) 2481–2495
42. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2015) 234–241
43. Zhang, R., Tang, S., Zhang, Y., Li, J., Yan, S.: Scale-adaptive convolutions for scene parsing. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 2031–2039
44. Li, H., Xiong, P., An, J., Wang, L.: Pyramid attention network for semantic segmentation. arXiv preprint arXiv:1805.10180 (2018)
45. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
46. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. (2011) 315–323
47. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. (2017) 11–19
48. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: Denseaspp for semantic segmentation in street scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3684–3692
49. Wang, R.J., Li, X., Ling, C.X.: Pelee: A real-time object detection system on mobile devices. In: Advances in Neural Information Processing Systems. (2018) 1963–1972
50. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15** (2014) 1929–1958
51. Huang, G., Liu, S., Van der Maaten, L., Weinberger, K.Q.: Condensenet: An efficient densenet using learned group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2752–2761
52. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in Neural Information Processing Systems. (2016) 667–675

53. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
54. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 558–567
55. Chao, P., Kao, C.Y., Ruan, Y.S., Huang, C.H., Lin, Y.L.: Hardnet: A low memory traffic network. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 3552–3561