Dense In Dense: Training Segmentation from Scratch

Anonymous ACCV 2018 submission

Paper ID 771

Abstract. In recent years, training image segmentation network often needs fine-tuning the model which comes from the initial training upon large-scale classification datasets like ImageNet. Such fine-tuning methods are confronted with three problems: (1). domain gap. (2). mismatch between data size and model size. (3). poor controllability. A more practical solution is to train the segmentation model from scratch, which motivates our Dense In Dense(DID) network. In DID, we put forward an efficient architecture based on DenseNet to further accelerate the information flow inside and outside the dense block. Deep supervision also applies to a progressive upsampling rather than the traditional straightforward upsampling. Our DID Network performs favorably on Camvid dataset, Inria Aerial Image Labeling dataset , Cityscapes by training from scratch with less parameter burden.

1 Introduction

Convolutional Neural Networks(CNNs) are driving major advances in many computer vision tasks, such as image classification[1], object detection[2] and semantic image segmentation[3]. Recently, more and more network architectures[4, 5] are proposed to be applied in those basic tasks, which are trained on large scale standard benchmarks such as ImageNet[6]. For instance, Szegedy *et al.* [7] propose an "Inception" module which concatenates feature maps produced by various sized filters. He *et al.*[4] propose the residual learning blocks with skip connections, which makes it possible to train *very deep* networks with more than 100 layers. Huang *et al.*[5] propose the DenseNet with dense layer-wise connections. Reasonable model architectures and large-scale data simultaneously push forward the frontier results of basic computer vision tasks step by step.

In the field of image semantic segmentation, many works[8, 9] are pretrained on large scale datasets such as ImageNet to utilize the big data to obtain an initial feature extractor and then fine-tuned on their own specific dataset. The fine-tuning process which comes from transfer learning[10] owns several advantages. Firstly, it enables better learning of low-level features representation by the large-scale data and high-level features based on specific task. Secondly, fine-tuning costs much less training time compared with training classification task on ImageNet from scratch. In the end, less annotation data is needed for fine-tuning because some basic features can be shared from the source model.

On the other hand, there are also several serious limitations when fine-tuning is adopted in image segmentation task. (1) *domain gap*. There are one thousand data distributions in one thousand datasets, which varies in lighting, scene, distance, etc. Directly fine-tuning sometimes tends to cause a poor result due to the domain gap between the source dataset and target dataset. (2) *mismatch between data size and model size*. As *"no free lunch theorem"* indicates, model size should correspond with data size suitable. Too large data size would cause under-fitting, inversely overabundant model size would give rise to over-fitting. In fine-tuning process, the mismatch between source dataset and target dataset in data size often leads to inappropriate fitting. (3) *poor controllability*. The main network structure trained from the source dataset is often reserved when fine-tuning, which greatly limits the network design space.

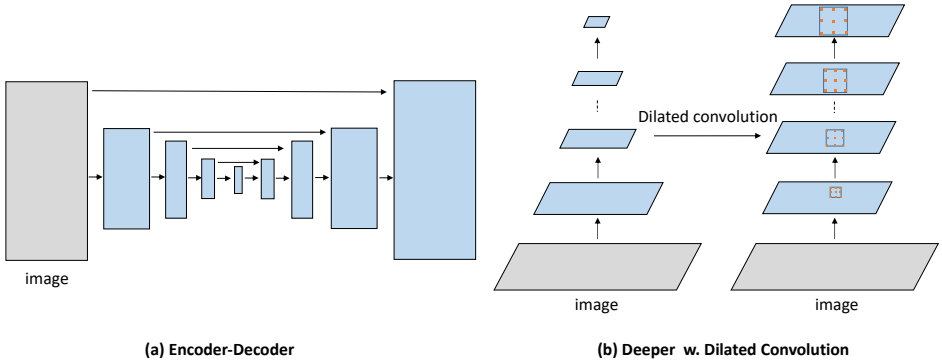

Fig. 1. Two typical network structures in image segmentation. (a). Encoder-Decoder with U-shape. (b). Deeper network enlarging the receptive field without reducing the image resolution.

We here restate some normal cases under which we have to train segmentation task from scratch. At first, most pre-trained models are trained on large-scale RGB image dataset such as ImageNet. It's very difficult to alter the model for the application in domains of depth images(with R-G-B-D four channels), multi-spectrum images in remote sensing field, medical images, etc. Secondly, to make the segmentation model efficiently work in the embedded system which is very resource-limited, it's unrealistic to use a model with large parameter number and heavy computing load. Training from scratch is an inevitable method in those embedded applications. Last but not least, data is precious in some fields such as medical diagnosis, which seriously need highly experienced physician to annotate medical data. Under these circumstances, utilizing large models such as resnet50 and resnet101 to pretrain on those small datasets will inevitably lead to over-fitting. The generalization ability will be seriously influenced.

Motivated by the above limitations of fine-tuning, we propose our DID Network architecture which can train image segmentation task from scratch and reach a better result than the fine-tuning. Xie *et al.*[11] proposed a holistically-nested structure for edge detection named HED, which also contains side-output layers from diverse scale context for explicit multi-scale fusion. Same with HED, the key point of our DID structure is to further enhance the information flow based on the DenseNet and simultaneously fuse the multi-scale context information to progressively refine the segmentation performance. Notably, as DenseNet only guarantees that the information flows sufficiently inside the dense block, our DID Network both accelerates the information flow **inside and outside** the dense block, which is where the name of Dense In Dense(DID) Network comes from.

Our main contributions are summarized as follows:

- (1) We put forward Dense In Dense(DID) Network, which not only explicitly fuses multi-scale context to realize progressive upsampling but also sufficiently facilitates the neural network information flow inside and outside the dense block.
- (2) We validate some factors to design efficient segmentation networks for training from scratch though step-by-step ablation studies.
- (3) Our DID Network performs favorably on Camvid dataset, Inria Aerial Image Labeling dataset , Cityscapes by training from scratch with less parameter burden.

2 Related Work

2.1 Image Segmentation

In the early period, the CNN is only employed in the classification task due to GPU memory limitation. Most of them[1, 7] are composed of convolution layers and fully connected layers. Fully Convolutional Network(FCN)[3] firstly applies CNN into image segmentation, which makes it a fundamental architecture for segmentation. FCN is the first one to build "fully convolutional" network that takes the input of arbitrary size and produces correspondingly-sized output with efficient inference and learning. A skip connection that combines semantic information between a deep, coarse layer and a shallow, fine layer is deployed to produce both accurate and detailed segmentation result. Many different structures are proposed based on FCN, most of which can be divided into (1). Encoder-Decoder. (2). Deeper with Dilated Convolution, as shown in Fig 1. Our DID network takes advantage of both structures to realize a progressive decoder based on a deep network with Dilated Convolution.

Encoder-Decoder: The architecture is composed of two parts: (a). encoder. The spatial dimension of feature maps is reduced gradually to a minimal size. Multiscale context can be extracted from this part. (b). decoder. The spatial dimension of feature maps is increased gradually to the original image size. In

addition, skip layers from encoder can be also utilized at multiscale. For instance, SegNet[12] reuses the pooling indices from the encoder and adds extra convolutional layers to recover the image size. Meanwhile skip connections are widely applied in the decoder parts in U-net[13] to further fuse the multi-context.

Deeper with Dilated Convolution: In Deeplab[8], Dilated Convolutions are introduced as an alternative to CNN pooling layers in deep part to capture larger context without reducing the image resolution. A module named atrous spatial pyramid pooling (ASPP) is also included in Deeplab where parallel Dilated Convolution layers with different rates capture multi-scale information. Following the same spirit, Yu *et al.*[14] propose to provide FCNs with a context module built as a stack of dilated convolutional layers to enlarge the field of view of the network. Recently, Pyramid Scene Parsing Net (PSPNet)[15] performs spatial pooling at several grid scales and demonstrates outstanding performance on several semantic segmentation benchmarks.

Post Processing: DenseCRF[16] is a typical post processing method in segmentation. Position, color based pairwise potential and segmentation probability based unary potential are incorporated in DenseCRF. In another aspect, RNN has been introduced to approximate mean-field iterations of CRF optimization, allowing for an end-to-end training of both the FCN and the RNN[17]. Recently, Spatial Propagation Networks(SPN)[18] is proposed to directly learn the affinity matrix of pixels in a purely data-driven manner.

Devils in Decoder: In the work of Deeplab[8], by application of Dilated Convolution, the features are bilinearly upsampled by a factor of 8, which can be considered as a naive decoder module. Because of the diversity of object scale, this naive decoder module may not successfully recover object semantic details. Meanwhile, great memory cost caused by larger deep feature map size will do harm to the architecture performance. To mitigate this problem, Dense Upsampling Convolution (DUC)[19] is designed to generate a pixel-level prediction, which is able to capture and decode more detailed information that is generally missing in bilinear upsampling. However, DUC misses the global spatial information which is critical for segmentation task. In our work, we progressively upsample the feature map based on cascaded multi-scale context rather than directly upsample to original image size. Our architecture significantly saves large memory cost and widely refines the result.

2.2 Training Neural Network from Scratch

About training neural network from scratch in classification, there exist some methods in semantic segmentation[20] and object detection[21]. DSOD[21] proposes a framework based on DenseNet that can learn object detectors from scratch, meanwhile deep supervision is also applied in their model. Jegou *et al.*[20] demonstrate that a well-designed network structure FC-DenseNet can outperform state-of-the-art solutions without using the pre-trained models. It extends DenseNet to fully convolutional networks by adding an upsampling path

to recover the full input resolution. Our DID architecture demonstrates better result than FC-DenseNet by refining semantic result step by step.

3 Method

Our DID architecture is composed of Dense Encoder and DID Decoder. The structure of Dense Encoder is based on DenseNet. DID Decoder takes the multi-scale context of Dense Encoder as input and progressively decodes the feature map to the original image size.

In this section, we introduce the detailed structure of DID network. As the baseline of our work, we review the DenseNet architecture in a glance. Then we introduce our DID network from different aspects.

3.1 Review of DenseNet

DenseNet[5] design a sophisticated connectivity pattern that iteratively concatenates all feature outputs in a feed-forward fashion. Thus, the output of the l^{th} layer is defined as

$$x_l = H_l(concat(x_{l-1}, x_{l-2}, ..., x_0)) \quad (1)$$

where "concat" represents the concatenation operation. In this case, H is defined as BN, followed by ReLU, a convolution, and Dropout[22]. Such connectivity pattern strongly encourages the reuse of features and makes all layers in the architecture receive direct supervision signal. The output dimension of each layer l has k feature maps where k, hereafter referred to *growth rate*, is typically set to a small value(*e.g.* k=12). Thus, the number of feature maps in DenseNet grows linearly with the depth.

Dense Block. *dense block* represents the concatenation of the new feature maps created at a given resolution. A *dense block* with 4 units is demonstrated in Fig 2.

Transition Layer. A *transition layer* is introduced to reduce the spatial dimensionality of the feature maps. Such transformation is composed of a 1×1 convolution followed by a 2×2 pooling operation.

Compression Rate. To further improve model compactness, we can reduce the number of feature-maps at transition layers. If a dense block contains m feature-maps, we let the following transition layer generate $\theta \times m$ output feature maps, where $0 \leq \theta \leq 1$ represents the compression factor.

Difference between our DID Network and DenseNet. Our Dense Encoder architecture is based on DenseNet while differs from DenseNet in the following aspects: (a). Dilated convolution is adopted in Dense Block (3) and (4) to further enlarge the receptive field without downsampling the feature map, whose effect will be validated in the later experiment. (b). Transition layer after dense block (3) doesn't include 2×2 pooling to reduce feature map size. (c). Our dense block number varies from the official implementation because we believe the block number should coordinate well with dataset scale. (d). Larger growth

rate(48) is adopted. (e). We set compression factor as 1 instead of 0.5 in official DenseNet. (f). Most importantly, our DID Decoder accelerates information flowing inside and outside dense block and progressively refines the semantic result though multi-context fusion.

Table 1. Dense Encoder in DID architecture for Camvid Dataset. We demonstrate it as image size is 312×321. dense{m}-k{n}-d{p} means m dense units, growth rate is n, max dilation is p. dense30-k48-d4 is chosen as our final model structure, dense30-k48-d2 and dense58-k24-d2 are network structures for later ablation study in Sec 4.1. B-R-C means "Batch Normalization-ReLU-Convolution" combination. The output of dense block(1), dense block(2), dense block(3) are blended by DID Decoder which is depicted in Fig 2.

Layers	Output Size	dense30-k48-d2	**dense30-k48-d4**	dense58-k24-d2
Convolution	161×161	3×3 conv, stride 2		
Convolution	161×161	3×3 conv, stride 1		
Convolution	161×161	3×3 conv, stride 1		
Pooling	81×81	3×3 max pool, stride 2		
Dense Block(1)	81×81	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 6$
Transition Layer(1)	81×81	1×1 B-R-C		
	41×41	2×2 average pool, stride 2		
Dense Block(2)	41×41	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 12$
Transition Layer(2)	41×41	1×1 B-R-C		
	21×21	2×2 average pool, stride 2		
Dense Block(3)	21×21	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$ with dilation=2	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$ with dilation=2	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 24$ with dilation=2
Transition Layer(3)	21×21	1×1 B-R-C		
Dense Block(4)	21×21	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$ with dilation=2	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 8$ with dilation=4	$\begin{bmatrix} 1 \times 1 \text{ B-R-C} \\ 3 \times 3 \text{ B-R-C} \end{bmatrix} \times 16$ with dilation=4

3.2 Dense In Dense Network

In this subsection, we elaborate the DID architecture. The intention of the detail design in DID network can be found in the later Sec 4.

Dense Encoder. We demonstrate our Dense Encoder structure elaborately shown in Table 1. Dense30-k48-d4 is chosen as our final Dense Encoder in DID network architecture based on the experiment in Sec 4. In Dense Encoder we define the structure of dense block and the transition layer. Following pre-activation[23], our dense block layers are composed of BN, followed by ReLU, a 3×3 Convolution and dropout with p=0.2. The growth rate of the layer is set to k=48. The transition layer is composed of BN, followed by ReLU, a 1×1 Convolution, dropout with p=0.2. Notably, In the deeper part, we do not down-sample the feature map in transition layer and apply Dilated Convolution in the

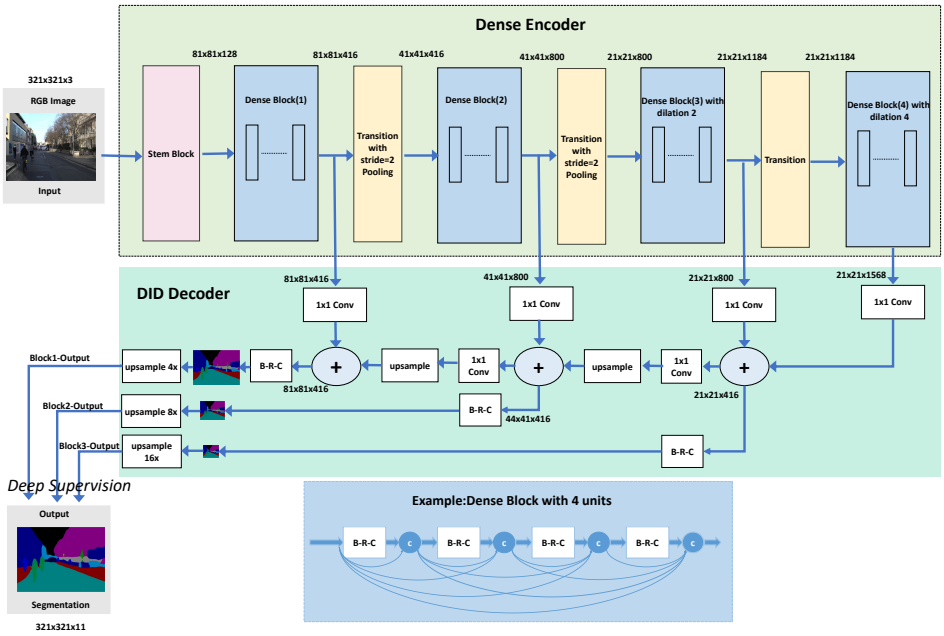

Fig. 2. Our network structure. Dense Encoder and DID Decoder make up our total architecture. Our Dense Encoder contains several combinations of dense block and transition block which is dense30-k48-d4 shown in Table 1 concretely. DID Decoder includes progressive refinement based on 1×1 convolution, bilinear upsampling. B-R-C means "Batch Normalization-ReLU-Convolution" combination. The feature map dimension is marked as $w \times h \times c$ after some operations. Our network structure finally outputs three probability map:block1-output, block2-output, block3-output(ranging from fine to coarse), only the finest output(block1-output) is used for our final inference. Best viewed in color.

deeper dense block. Inspired by DSOD[21], we define *stem block* as a stack of three 3×3 convolution layers followed by a 2×2 max pooling layer. The first convolution layer works with stride=2 and the other two with stride=1.

DID Decoder. Details of our DID decoder are demonstrated in Fig 2. All of the Convolutions in DID Decoder is 1×1 Convolution without BN and ReLU, except for the final convolution before upsampling. For 321×321 input images, four scales of feature maps are applied for segmentation. The largest feature maps are from the dense block (1) which owns the finest resolution(81×81) in order to handle small objects in an image. The remaining feature maps are from the output of dense block (2),(3) respectively. For short, we call these feature maps as block1-output, block2-output, block3-output accordingly.

Deep Supervision. Deep supervised learning has been successfully utilized in GoogLeNet[7], HED[24], etc. The core idea is to provide integrated objective function as direct supervision for the earlier hidden layers, rather than only for

the output layer. We adopt deep supervision in our DID Network based on the following points: (1). These "auxiliary" objective functions at multiple hidden layers can mitigate the "vanishing" gradients problem. By deep supervision, it enables the precious gradient to keep propagating in the shallow parts of DID Network. (2). This deep supervised pattern greatly facilitates the multi-context fusion. In contrast, DenseNet only guarantees that the information flows sufficiently in the dense block. our DID Network both accelerates the information flow **inside and outside** the dense block. Sufficient information flow will make the network easier to be trained from scratch.

Wide DenseNet. As the original implementation of DenseNet is not memory efficient, wide DenseNet is adopted in our DID Network for better memory utilization. We set growth rate as k=48. Interestingly, we also find that wide DenseNet leads to better accuracy than deep DenseNet on basis of nearly same parameter number. We conclude that wide structure is superior to a deep structure in the task of segmentation because far-ranging semantic features are essential for segmentation, which can only be extracted by wide DenseNet.

Loss Function As we adopt the deep supervision in our DID Network, three cross entropy losses will be generated as indicated in Fig 2. Our final loss is the average of three losses. Compared to previous fuse strategies[24], only the finest probability map(block1-output) will be used for inference. Our progressive refinement strategy is proved to be effective.

3.3 Training Setting

We implement our code based on the tensorflow framework[25]. All our models are trained from scratch with Adam[26] solver on NVidia TitanX GPU. We initialize our models using Xavier[27] and train with an initial learning rate of 1e-3 and divide learning rate by 10 until we reach 2/3 of the total training process. All models are trained on data augmented with random crops and vertical flips. Following the method in Deeplabv3[28], we fine-tune on the trainval set for another 100 epochs with a smaller base learning rate=1e-4 after training. We regularize our models with a weight decay of 1e-4 and a dropout rate of 0.2. For Batch Normalization, we employ current batch statistics at training and use the moving average statistics of Batch Normalization during validation and test time.

4 Experimental Results

To better evaluate our model in the domain that owns different distribution from ImageNet, we conduct experiments on the Camvid Dataset and Inria Aerial Image labeling dataset[29]. The performance is measured by standard mean Intersection-over-Union(IoU).

Camvid is a dataset of fully segmented videos for urban scene understanding. We use the split and image resolution from [30], which consists of 367 frames for

training, 101 frames for validation and 233 frames for testing. Each frame has a size of 360×480 and its pixels are labeled with 11 semantic classes.

Inria Aerial Image labeling dataset[29] owns images with coverage of 810 km^2 (405 km^2 for training and 405 km^2 for testing). Its ground truth contains two semantic classes: building and not building. The images cover dissimilar urban settlements, ranging from densely populated areas (e.g, San Franciscos financial district) to alpine towns (e.g,. Lienz in Austrian Tyrol). Instead of splitting adjacent portions of the same images into the training and test subsets, different cities are included in each of the subsets. The original dataset image size is 5000×5000, we crop it into 473×473 randomly.

4.1 Ablation study on Camvid Dataset

We mainly investigate the basic improvement in cascade dilation, stem, compression rate, wide DenseNet, DID structure, deep supervision, etc. The overall result is shown in Table 2.

Table 2. Various design result on Camvid Dataset. Please refer to Table 3 for more details.

	DID							
baseline	✓	✓	✓	✓	✓	✓	✓	✓
cascade dilation?		✓	✓	✓	✓	✓	✓	✓
stem?			✓	✓	✓	✓	✓	✓
need compression?				✓	✓	✓	✓	✓
wider or deeper?					✓	✓	✓	✓
dense in dense?						✓	✓	✓
deep supervision?							✓	✓
more iteration?								✓
Camvid Test mIoU	54.9	56	58.8	60	61.3	64.5	65.7	68.3

Cascade Dilation. Dilated Convolution plays an important role in segmentation task. Without Dilated Convolution, the final output feature map size will be only 1/32 of the original image size, which greatly does harm to the performance. Therefore, we adopt Dilated Convolution in transition layer (3),(4) and regard it as our baseline. In Table 3 (row 4,5,6), we mainly compare the different dilation rate(2, 4, 8) in transition block (4). We find that dilation rate = 4 performs 1.1% mIoU better than dilation rate=2, while dilation rate =8 nearly shows the same mIoU with dilation rate =4. Therefore, we use the dilation rate =4 dense block for our later experiment.

Stem Block in DenseNet. As DSOD[21] states, stem block is important in detection whose architecture is based on DenseNet. Motivated by this, we set stem block as a stack of three 3×3 convolution layers followed by a 2×2 max pooling layer. Our result (row 6, 7 in Table 2) could obtain 3.2% extra

gain in our segmentation task. We conclude that 7×7 convolution is difficult for Camvid such small size dataset to train a good result. Instead, a stack of 3×3 convolution is much more convenient for training especially when the dataset scale is not so huge as ImageNet.

Compression Rate. We compare two compression rates (0.5, 1) in the transition layer of DenseNet. Results are shown in Table 3(row 7 and 8). Compression rate =1 means that there is no feature map number reduction in the transition layer, while compression rate=0.5 means half of the feature map number is reduced. Results show that compression rate =1 yields 1.2% higher mIoU than compression rate=0.5 with extra 2.5M parameters. On the other hand, the wide DenseNet could also mitigate the memory efficiency problem in DenseNet so that we can fit larger batch size into the GPU memory. In the remaining ablation study, we mainly adopt the wide DenseNet architecture.

Wider or Deeper. We mainly increase the growth rate while reducing the depth on the basis of keeping nearly same parameter numbers. As shown in Table 3(row 3, 8), we mainly compare dense58k24 and dense30k48. Dense30k36 achieves slightly 0.1% better mIoU than dense58k24 with 1.9M less parameters. Furthermore, when we make the DenseNet wider(growth rate as 48), wider DenseNet brings us extra 1.3% mIoU(row 8, 9). This result makes us firmly believe that wide structure is more important than deep structure in the task of segmentation, for it owns a better utilization of the total parameter space.

Table 3. Ablation Study Result on Camvid Dataset. did3 means DID Network with 3×3 convolution, did1 means DID Network with 1×1 convolution, dense30k48.did1.ds means DID Network with 1×1 convolution and deep supervision.

row	pre-train	input size	backbone network	stem	dilation	compression	# parameters(M)	iterations	Test mIoU
1	✓	321 × 321	ResNet50	✗	2	–	23.7	9.9K	61
2	✗	321 × 321	ResNet50	✗	2	–	23.7	44.8K	59.8
3	✗	321 × 321	dense58k24	✓	4	1	7.6	73.6K	59.9
4	✗	321 × 321	dense30k36	✗	2	0.5	2.9	32K	54.9
5	✗	321 × 321	dense30k36	✗	4	0.5	2.9	32K	56
6	✗	321 × 321	dense30k36	✗	8	0.5	2.9	32K	55.6
7	✗	321 × 321	dense30k36	✓	4	0.5	3.2	38.4K	58.8
8	✗	321 × 321	dense30k36	✓	4	1	5.7	44K	60
9	✗	321 × 321	dense30k48	✓	4	1	9.6	60.8K	61.3
10	✗	353 × 353	dense30k48	✓	4	1	9.6	67.2K	61.3
11	✗	385 × 385	dense30k48	✓	4	1	9.6	83.2K	59.2
12	✗	321 × 321	dense30k48	✓	4	1	9.6	171K	63.4
13	✗	321 × 321	dense30k48.did3	✓	4	1	14.4	67.2K	63.7
14	✗	321 × 321	dense30k48.did1	✓	4	1	11.6	67.2K	64.5
15	✗	321 × 321	dense30k48.did1.ds	✓	4	1	11.6	67.2K	65.7
16	✗	321 × 321	dense30k48.did1.ds	✓	4	1	11.6	182K	68.3

Dense In Dense(DID). Our DID structure mainly fuses the larger scale context with refined context recursively. Multi-scale context feature from differ-

ent dense blocks is smoothened by 1×1 convolutions. Neighbor context features are fused by adding operation. As indicated in Table 3(row 9, 13), our DID structure(row 13) outperforms 2.4% mIoU than naive wide DenseNet baseline(row 9).

DID with 1×1 Convolution. Even though DID structure discussed above can bring large gain for the segmentation task, it brings extra 4.8M parameter overload. After analysis, we find the parameter burden is mainly caused by the convolution operation after the add operation in DID structure. After replacing the 3×3 convolution into 1×1 convolution, interestingly, we achieve 0.7% better mIoU than the 3×3 convolution with nearly less 2.8M parameters(row 13, 14).

Deep Supervision. Inspired by the wide application of deep supervision in dense pixel tasks[15], we creatively append deep supervision in every level of our DID structure output, which is depicted in Fig 2. Three cross entropy losses are finally averaged for back propagation. 1.2% extra mIoU gain is obtained after applying the deep supervision as shown in Table 3 (row 14, 15). We also try to blend the three probability maps in inference, the result is worse than using the finest probability map, which inversely demonstrates our refinement strategy really works.

Image Size and Iteration Number. Image size is an easily missed factor which greatly dominates the performance. We compare four image sizes: 321×321, 353×353, 385×385, 360×480. We try our best to make the image input with spatial dimensions that fits 16n+1,$n \in \{0, 1, 2, ...\}$ to facilitate alignment of the features to image. 321×321 and 353×353 demonstrates nearly the same better result that other image size choice. In order to make the batch size larger, we choose 321×321 as our final image size. As row 9, 12 in Table 3 indicates, larger iteration number leads a 2.1% better mIoU than the basic dense30k48. In order to save time, we conduct main ablation studies in small iteration number, after fixing the final architecture, training with a relative larger iteration number will be conducted. In the end, based on our best model structure dense30k48.did1.ds, we retrain it with a larger iteration number and obtain 68.3% mIoU(row 16).

Comparing with traditional fine-tuning method. By the analysis of disadvantage in fine-tuning in Sec 1, mismatch problem between data size and model size obviously exists on our Camvid Dataset. Our DID Network demonstrates nearly 4% mIoU better than the fine-tuning method with 12M fewer parameters, the result of which is shown in Table 3(row 1, 16).

Final Result on Camvid Dataset The overall design result is shown in Table 2. With larger iteration numbers, our DID Network could achieve the final 68.3% mIoU. Compared with the recent work FC-DenseNet[20], our result surpasses about 1.4% mIoU with slightly more parameter burden. On the other hand, Some approaches such as [31] utilize the temporal information, while our method only consider single-frame information, we don't compare with those methods for fairness.

It's worth noting that our DID Network training from scratch also performs favorably against some methods based on pretraining, such as deeplabv1-res50. To prove the progressive refinement, we try to calculate the mIoU that comes from the probability maps of Block3-output, Block2-output, Block1-output.The

mIoU results are 67%, 67.9%, 68.3% accordingly. This observation shows that fine feature map will obtain a better result than a coarse result. Therefore it quantitatively proves our progressive refinement strategy in DID.

Some visualization results on Camvid Dataset are shown in Fig 3. We demonstrate the Block3-Output, Block2-Output, Block1-Output which accordingly correspond to the probability map from coarsest to finest. Block3-Output(finest result) will be adopted in our final inference, the detailed structure can be found in Fig 2. From the visualization result, we can easily observe that our progressive refinement mainly occurs in semantic boundary(tree, building boundary) or tiny objects(pole). These phenomena qualitatively prove that our step-by-step refinement based on DID Network functions well as our assumption.

Table 4. Test Result on Camvid Dataset

Model	Pretrained	#params(M)	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Pavement	Cyclist	mIoU	Acc
SegNet[30]	✓	29.5	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4	62.5
DeconvNet[32]	✓	252	n/a											48.9	85.9
Visin et al.[33]	✓	32.3	n/a											58.8	88.7
FCN8s[3]	✗	134.5	77.8	71.0	88.7	76.1	32.7	91.2	41.7	24.4	19.9	72.7	31.0	57.0	88.0
deeplabv1-res50[8]	✓	24	80.9	75.1	89.8	81.6	36.5	94	45.5	34.6	7.8	78.3	46.9	61.0	90.35
Dilation8[14]	✓	140.8	82.6	76.2	89.0	**84.0**	**46.9**	92.2	56.3	35.8	23.4	75.3	**55.5**	48.9	85.9
FC-DenseNet103 (k=16)[20]	✗	9.4	83.0	77.3	93.0	77.3	43.9	94.5	59.6	37.1	**37.8**	82.2	50.5	66.9	91.5
our DID Network	✗	11.2	**85.1**	**79.4**	**93.8**	83.5	46.4	**94.9**	**62.5**	**37.8**	33.5	**84.8**	49.8	**68.3**	**92.6**

4.2 Result on Inria Aerial Image labeling dataset

In Aerial Image Labeling dataset(IAIL), we crop the large image evenly into 500×500, then randomly crop out 473×473 as the input of neural network. 20% of the images are used as validation data. The total number of training iterations is 200K.

Results are summarized in Table 5. Some of the methods are from the official benchmark who didn't reveal their detail method. Notably, our DID Network achieves 74.95% mIoU on the test set, which exceeds the ResNet101-pretrained result with a large margin.

Fig 4 shows some qualitative segmentation examples on IAIL Dataset with our DID Network. We can get the following finding: when the buildings in the IAIL Dataset are closely adjacent to each other, our DID network is able to gradually separate each single building through our step-by-step refinement. Our refining upsampling strategy obviously benefits from the DID Decoder to perform favorably against other methods.

Table 5. Test result on Inria Aerial Image Labeling Dataset

method	Bellingham		Bloomington		Innsbruck		San Francisco		East Tyrol		Overall	
	IoU	Acc	IoU	Acc	IoU	Acc	IoU	Acc	IoU	Acc	IoU	Acc
Inria1	52.91	95.14	46.08	94.95	58.12	95.16	57.84	86.05	59.03	96.40	55.82	93.54
DukeAMLL	66.90	96.69	58.48	96.15	69.92	96.37	75.54	91.87	72.34	97.42	70.91	95.70
NUS	**70.74**	97.00	66.06	96.74	73.17	96.75	73.57	91.19	76.06	97.81	72.45	95.90
ENPC Singh2	64.28	96.00	65.84	96.52	**77.11**	**97.31**	75.86	**92.01**	**78.68**	**98.12**	73.30	95.99
Deeplabv1-ResNet101 pretrained	68.62	96.81	66.06	96.82	76.43	97.24	74.68	91.58	77.92	98.05	73.41	96.10
Our DID Network	70.73	**97.10**	**69.75**	**97.15**	76.58	97.26	**76.10**	91.80	78.63	98.11	**74.95**	**96.29**

4.3 Result on Cityscapes Dataset

We also conducted experiment about DID Network in Cityscapes Dataset in Table 6. Compared with other methods training segmentation from scratch, our method obtains a better result 73.7 mIoU with much less parameter burden. Noticeably, the FC-DenseNet[20] only experiment with Camvid Dataset except Cityscapes Dataset, our method reaches state of the art in Cityscapes compared with other methods of the same type. Therefore, our method has more generality in dataset than FC-DenseNet[20].

Table 6. mIoU scores from the Cityscapes test set

Model	Coarse	Pretrained	#params(M)	Road	Sidewalk	Building	Wall	Fence	Pole	Traf. Light	Traf. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Class IoU	Category IoU
ENet[34]	✗	✗	0.36	96.3	74.2	75.0	32.2	33.2	43.4	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.28	80.39
Adelaide[35]	✗	✓	–	98.0	82.6	90.6	44.0	50.7	51.1	65.0	71.7	92.0	72.0	94.1	81.5	61.1	94.3	61.1	65.1	53.8	61.6	70.6	71.6	–
ERFNet[36]	✗	✓	–	97.9	82.1	90.7	45.2	50.4	59.0	62.6	68.4	91.9	69.4	94.2	78.5	59.8	93.4	52.3	60.8	53.7	49.9	64.2	69.7	87.3
FRRN[37]	✗	✗	17.7	98.2	83.3	91.6	45.8	51.1	62.2	69.4	72.4	92.6	70.0	94.9	81.6	62.7	94.6	49.1	67.1	55.3	53.5	69.5	71.8	–
LRR[38]	✓	✓	–	97.9	81.5	91.4	50.5	52.7	59.4	66.8	72.7	92.5	70.1	95.0	81.3	60.1	94.3	51.2	67.7	54.6	55.6	69.6	71.8	–
our DID Network	✗	✗	11.2	98.4	84.6	92.1	48.0	54.1	63.9	70.4	74.9	93.1	68.4	95.6	84.2	66.4	95.2	47.6	62.7	68.4	60.6	72.0	**73.7**	**89.66**

5 Conclusion

In this paper, we have presented Dense In Dense(DID) Network, a simple yet efficient framework for training segmentation from scratch. DID not only can realize progressive upsampling for better segmentation refinement, but also greatly facilitate the gradient flow inside and outside the dense block. Our experimental results show the salient effects of parameters and less redundancy of features. DID has great potential on the scenario with diverse domain such as medical segmentation, aerial image segmentation, remote sensing image segmentation, etc. Our future work is trying to apply DID in those fields.

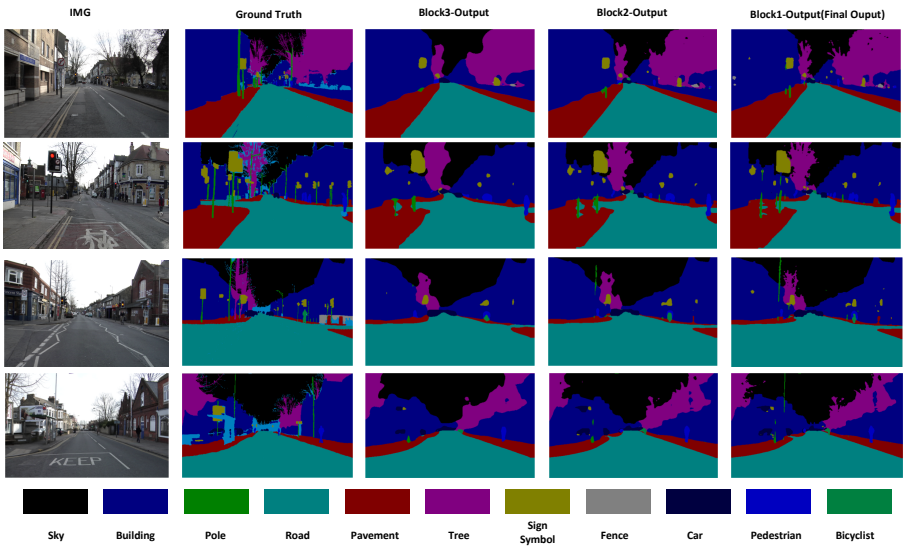

Fig. 3. Camvid Dataset result. Block3-Output, Block2-Output, Block1-Output accordingly correspond to the probability map from coarsest to finest, Block3-Output(finest probability map) will be adopted for our inference. Best viewed in color.

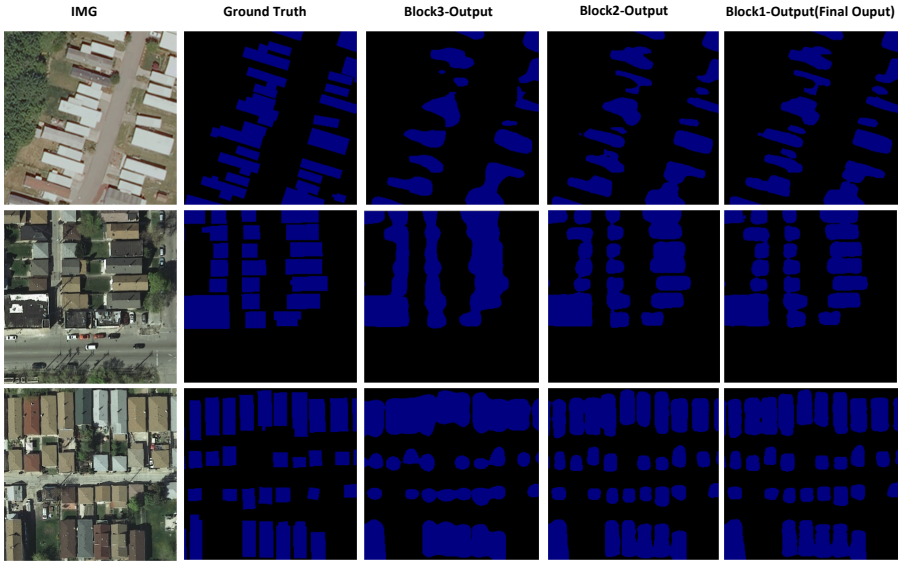

Fig. 4. Inria Aerial Image labeling Dataset result. Blue means buildings, black means non-buildings. Block3-Output, Block2-Output, Block1-Output accordingly correspond to the probability map from coarsest to finest, Block3-Output(finest probability map) will be adopted for our inference. Best viewed in color.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
2. Girshick, R.: Fast R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV). (2015)
3. 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
4. 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
5. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Volume 1. (2017) 3
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 248–255
7. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions, Cvpr (2015)
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. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
10. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 1717–1724
11. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE international conference on computer vision. (2015) 1395–1403
12. 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
13. 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
14. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
15. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2017) 2881–2890
16. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: Advances in neural information processing systems. (2011) 109–117
17. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1529–1537

18. Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.H., Kautz, J.: Learning affinity via spatial propagation networks. In: Advances in Neural Information Processing Systems. (2017) 1519–1529
19. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. arXiv preprint arXiv:1702.08502 (2017)
20. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on, IEEE (2017) 1175–1183
21. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: Dsod: Learning deeply supervised object detectors from scratch. In: The IEEE International Conference on Computer Vision (ICCV). Volume 3. (2017) 7
22. 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
23. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, Springer (2016) 630–645
24. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE international conference on computer vision. (2015) 1395–1403
25. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI. Volume 16. (2016) 265–283
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
27. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010) 249–256
28. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
29. Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P.: Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE (2017)
30. 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
31. Gadde, R., Jampani, V., Gehler, P.V.: Semantic video cnns through representation warping. CoRR, abs/1708.03088 **8** (2017) 9
32. 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
33. Visin, F., Romero, A., Cho, K., Matteucci, M., Ciccone, M., Kastner, K., Bengio, Y., Courville, A.: Reseg: A recurrent neural network-based model for semantic segmentation. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on, IEEE (2016) 426–433
34. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. (2016)
35. Lin, G., Shen, C., Van Den Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. (2016) 3194–3203

36. 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** (2018) 263–272
37. Pohlen, T., Hermans, A., Mathias, M., Leibe, B.: Full-resolution residual networks for semantic segmentation in street scenes. (2017)
38. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation, Springer (2016) 519–534
39. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
40. Han, S., Pool, J., Narang, S., Mao, H., Tang, S., Elsen, E., Catanzaro, B., Tran, J., Dally, W.J.: Dsd: Regularizing deep neural networks with dense-sparse-dense training flow. arXiv preprint arXiv:1607.04381 (2016)
41. Zink, J.M.: Automated dataset generation for image recognition using the example of taxonomy. arXiv preprint arXiv:1802.02207 (2018)