Learning Fully Dense Neural Networks for Image Semantic Segmentation

Mingmin Zhen[1], Jinglu Wang[2], Lei Zhou[1], Tian Fang[3], Long Quan[1]
[1]Hong Kong University of Science and Technology, [2]Microsoft Research Asia, [3]Altizure.com
{mzhen, lzhouai, quan}@cse.ust.hk, Jinglu.Wang@microsoft.com, fangtian@altizure.com

Abstract

Semantic segmentation is pixel-wise classification which retains critical spatial information. The "feature map reuse" has been commonly adopted in CNN based approaches to take advantage of feature maps in the early layers for the later spatial reconstruction. Along this direction, we go a step further by proposing a fully dense neural network with an encoder-decoder structure that we abbreviate as FDNet. For each stage in the decoder module, feature maps of all the previous blocks are adaptively aggregated to feedforward as input. On the one hand, it reconstructs the spatial boundaries accurately. On the other hand, it learns more efficiently with the more efficient gradient backpropagation. In addition, we propose boundary-aware loss function to focus more attention on the pixels near the boundary, which boosts the "hard examples" labeling. We have demonstrated the best performance of the FDNet on the two benchmark datasets: PASCAL VOC 2012, NYUDv2 over previous works when not considering training on other datasets.

Introduction

Recent works on semantic segmentation are mostly based on fully convolutional network (FCN) (Long, Shelhamer, and Darrell 2015). Generally, a pretrained classification network (VGGNet (Simonyan and Zisserman 2015), ResNet (He et al. 2016) and DenseNet (Huang et al. 2017)) is used as an encoder to generate a series of feature maps with rich semantic information at the higher layers. In order to obtain the probability map with the same resolution as the input image size, the decoder is adopted to recover the spatial resolution from the output of the encoder (Fig. 1 Top). The encoder-decoder structure is widely used for semantic segmentation (Vijay, Alex, and Roberto 2017; Long, Shelhamer, and Darrell 2015; Noh, Hong, and Han 2015; Zhao et al. 2017) .

The key difficulties for the encoder-decoder structure are twofold. First, as multiple stages of spatial pooling and convolutional strides are used to reduce the final feature map size in the encoder module, much spatial information is lost. This is hard to recover in the decoder module and leads to poor semantic segmentation results, especially for boundary localization. Second, the encoder-decoder structure has much deeper depth than the original encoder network for image classification tasks (such as VGGNet (Simonyan and Zisserman 2015), ResNet (He et al. 2016) and DenseNet (Huang et al. 2017)). This results in the training optimization problem as introduced in (He et al. 2016; Huang et al. 2017) though it has been partially solved by using batch normalization (BN) (Ioffe and Szegedy 2015).

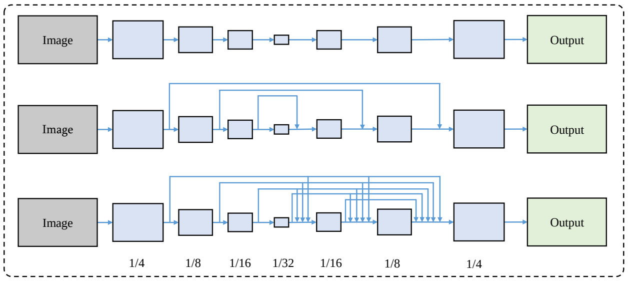

Figure 1: Different types of encoder-decoder structures for semantic segmentation. **Top**: basic encoder-decoder structure (e.g. DeconvNet (Noh, Hong, and Han 2015) and SegNet (Vijay, Alex, and Roberto 2017)) using multiple-stage decoder to predict masks, often results in very coarse pixel masks since spatial information is largely lost in the encoder module. **Middle**: Feature map reuse structures using previous feature maps of the encoder module achieves very good results in semantic segmentation tasks (Lin et al. 2017a; Islam et al. 2017; Ghiasi and Fowlkes 2016) and other related tasks (Pinheiro et al. 2016; Shen et al. 2017; Huang et al. 2018), but the potential of feature map reuse is not deeply released. **Bottom**: The proposed fully dense networks, using feature maps from all the previous blocks, are capable of capturing multi-scale information, of restoring the spatial information, and of benefitting the gradient backpropagation.

In order to address the spatial information loss problem, DeconvNet (Noh, Hong, and Han 2015) uses the unpooling layers to restore the spatial information by recording the locations of maximum activations during the pooling operation. However, this cannot completely solve the problem since only the location of maximum activations is restored. Another way to deal with this problem is to reuse the feature maps with rich spatial information of earlier layers. U-Net

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

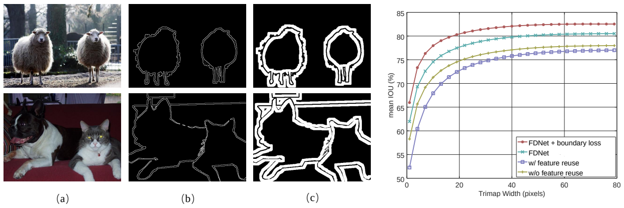

Figure 2: Left: (a) original images; (b) trimap example with 1 pixels; (c) trimap example with 10 pixels. Right: semantic segmentation result within a band around the object boundaries for different methods (mean IOU).

(Ronneberger, Fischer, and Brox 2015) exploits previous feature maps in the decoder module by "skip connections" structure (See Fig. 1 Middle). Furthermore, RefineNet (Lin et al. 2017a) refines semantic feature maps from later layers with fine-grained feature maps from earlier layers. Similarly, G-FRNet (Islam et al. 2017) adopts multi-stage gate units to make use of previous feature maps progressively. The feature map reuse significantly improves the restoration of spatial information. Meanwhile, it helps to capture multi-scale information from the multi-scale feature maps of earlier layers in the encoder module. In addition, it also boosts information flow and gradient backpropagation as the path from the earlier layers to the loss layer is shortened.

However, the potential of feature map reuse is not completely revealed. In order to further improve the performance, we propose to reconstruct encoder-decoder neural network to form a fully dense neural network (See Fig. 1 Bottom). We refer to our neural network as **FDNet**. FDNet is a nearly symmetric encoder-decoder network and is easy to optimize. We choose DenseNet-264 (Huang et al. 2017) as the encoder, which achieves state-of-the-art results in the image classification tasks. The feature maps in the encoder module are beneficial to the decoder module. The decoder module is operated as an upsampling process to recover the spatial resolution, aiming for accurate boundary localization. The feature maps of different scale size (including feature maps in the decoder module) will be fully reused through adaptive aggregation structure, which will generate a fully dense connected structure.

In general, cross entropy loss function is used to propagate the loss in previous works (Liu, Rabinovich, and Berg 2016; Lin et al. 2017a). The weakness of this method is that it sees all pixels as the same. As shown in Fig. 2, labeling for the pixels near the boundary (band width < 40) is not very accurate. In other words, the pixels near the boundary are "hard examples", which need to be treated differently. Based on this observation, we propose a boundary-aware loss function, which pays more attention on the pixels near the boundary. Though attention based loss has been adopted in object detection task (Lin et al. 2017c), our boundary-aware loss comes from the prior that pixels near the boundary are "hard examples". This is very different from focal loss, which pays more attention to the pixels with higher loss. In order to further boost training optimization, we use multiple losses for the output feature maps of the decoder module. As a result, basically each layer of FDNet has direct access to the gradients from loss layers. This will be very helpful to gradient propagation (Huang et al. 2017).

Related work

Fully convolutional network (FCN) (Long, Shelhamer, and Darrell 2015) has improved the performance of semantic segmentation significantly. In the FCN architecture, a fully convolutional structure and bilinear interpolation are used to realize pixel-wise prediction, which results in coarse boundaries as large amounts of spatial information have been lost. Following the FCN method, many works (Vijay, Alex, and Roberto 2017; Lin et al. 2017a; Zhao et al. 2017) have tried to further improve the performance of semantic segmentation.

Encoder-decoder. The encoder-decoder structure with a multi-stage decoder gradually recovers sharp object boundaries. DeconvNet (Noh, Hong, and Han 2015) and SegNet (Vijay, Alex, and Roberto 2017) employ symmetric encoder-decoder structures to restore spatial resolution by using unpooling layers. RefineNet (Lin et al. 2017a) and G-FRNet (Islam et al. 2017) also adopt a multi-stage decoder with feature map reuse in each stage of the decoder module. In LRR (Ghiasi and Fowlkes 2016), a multiplicative gating method is used to refine the feature map of each stage and a Laplacian reconstruction pyramid is used to fuse predictions. Moreover, (Fu et al. 2017) stacks many encoder-decoder architectures to capture multi-scale information. Following these works, we also use an encoder-decoder structure to generate pixel-wise prediction label maps.

Feature map reuse. The feature maps in the higher layers tend to be invariant to translation and illumination. This invariance is crucial for specific tasks such as image classification, but is not ideal for semantic segmentation which requires precise spatial information, since important spatial relationships have been lost. Thus, the reuse of feature maps with rich spatial information of previous layers can boost the spatial structure reconstruction process. Furthermore, feature map reuse has also been used in object detection tasks (Shen et al. 2017; Lin et al. 2017b) and instance segmentation tasks (Pinheiro et al. 2016; He et al. 2017) to capture multi-scale information when considering the objects with different scales. In our architecture, we fully aggregate previous feature maps in the decoder module, which shows outstanding performances in the experiments.

Fully dense neural networks

In this section, we introduce the proposed fully dense neural network (FDNet), which is visualized in Fig. 3 comprehensively. We first introduce the whole architecture. Next, the adaptive aggregation structure for dense feature maps is presented in detail. At last, we show the boundary-aware loss function.

Encoder-decoder architecture

Our model (Fig. 3) is based on the deep encoder-decoder architecture (e.g. (Noh, Hong, and Han 2015; Vijay, Alex, and

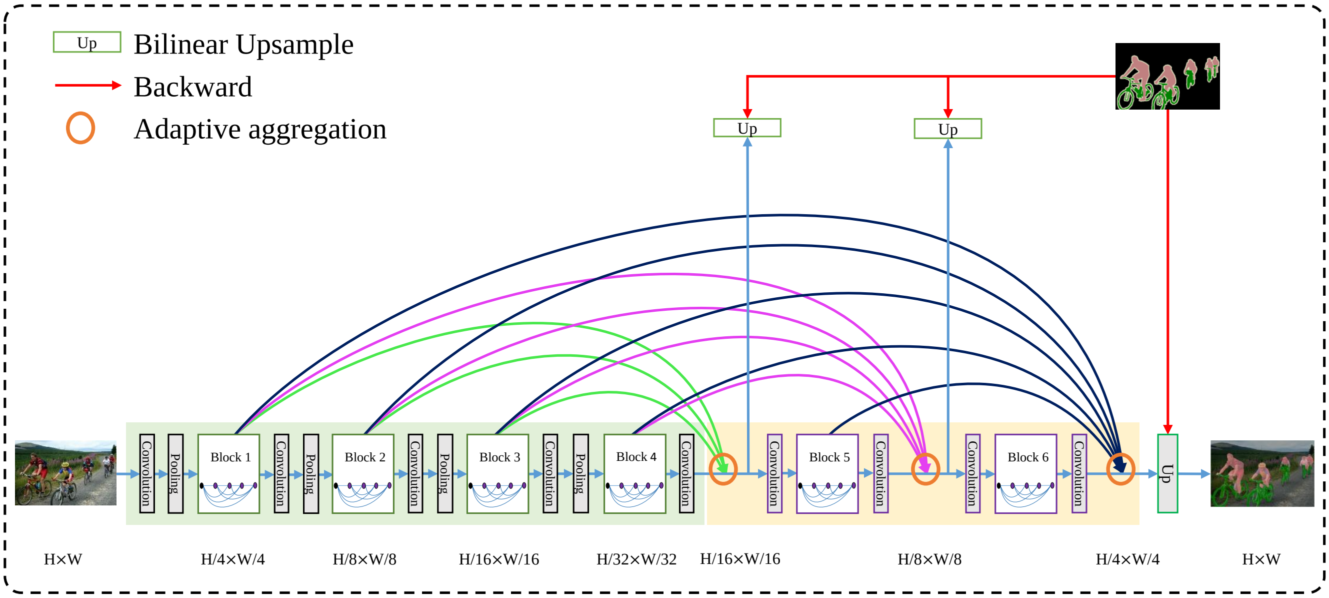

Figure 3: Overview of the proposed fully dense neural network (FDNet). The feature maps (output of dense block $1, 2, 3, 4$) of the encoder module and even the feature maps (output of dense block 5) of the decoder module are fully reused. The adaptive aggregation module combines feature maps from all the previous blocks to form new feature maps as the input of subsequent blocks. After an adaptive aggregation module or a dense block, a convolution layer is used to compress the feature maps. The aggregated feature maps are upsampled into size $H \times W \times C$ (C is number of classes for labels) and pixel-wise cross entropy loss is computed.

Roberto 2017)). The encoder module extracts features from an image and the decoder module produces semantic segmentation prediction.

Encoder. Our encoder network is based on the DenseNet-264 (Huang et al. 2017) while removing the softmax and fully connected layers of the original network (from the starting convolutional layer to dense block 4 in Fig. 3). The input of each convolutional layer within a dense block is the concatenation of all outputs of its previous layers at a given resolution. Given that x_l is the output of the ℓ^{th} layer in a dense block, x_ℓ can be computed as follows:

$$x_\ell = H_l([x_0, x_1, ..., x_{\ell-1}]) \tag{1}$$

where $[x_0, x_1, ..., x_{\ell-1}]$ denotes the concatenation operation of the feature maps $x_0, x_1, ..., x_{\ell-1}$, and x_0 is the input feature map of the dense block. Meanwhile, $H_\ell(\cdot)$ is defined as a composite function of operations: BN, ReLU, a 1×1 convolution operation followed by BN, ReLU, a 3×3 convolution operation. As a result, the output of a dense block includes feature maps from all the layers in this block. Each dense block is followed by a transition layer, which is to compress the number and the size of feature maps through 1×1 convolution and pooling layers. For an input image I, the encoder network produces 4 feature maps (B_1, B_2, B_3, B_4) with decreasing spatial resolution $(\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32})$. In order to reduce spatial information loss, we can remove the pooling layer before dense block 4 so that the output feature map of the last dense block (i.e. B_4) in the encoder module is $\frac{1}{16}$ of the size. Atrous convolution is also used to control the spatial density of computed feature responses in the last block as suggested in (Chen et al. 2017). For this architecture, we refer to it as FDNet-16s. The original architecture can be taken as FDNet-32s.

Decoder. As the encoder-decoder structure has much more layers than the original encoder network, how to boost gradient backpropagation and information flow becomes another problem we have to deal with. The decoder module progressively enlarges the feature maps while densely reusing previous feature maps by aggregating them into a new feature map. As the input feature map of each dense block has a direct connection to the output of the block, the inputs of previous blocks in the encoder module also connect to the new feature map directly. The new feature map is then upsampled to compute loss with the groundtruth, which leads to multiple losses computation. Thus, the inputs of all dense blocks in the FDNet have a direct connection to the loss layers. This will significantly boost the gradient backpropagation.

Following the DenseNet structure, we also use dense block at each stage of the same size after a compression layer with convolution operation, which is to change the number of feature maps from adaptive aggregation structure. The compression layer is composed of BN, ReLU and 1×1 convolution operation. In the two compression layers after adaptive aggregation, their filter numbers are set to 1024 and 768. In the two compression layers after block 5 and block 6, the filter numbers are set to 768 and 512. For block 5 and block 6, there are 2 convolutional layers in each of them.

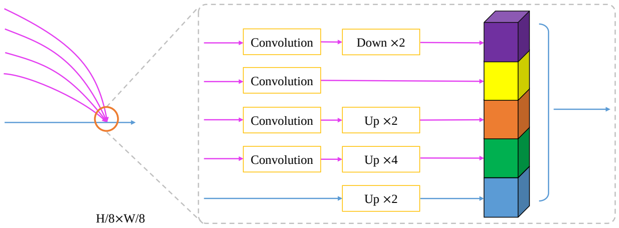

Figure 4: An example of an adaptive aggregation structure for dense feature maps. For all the input feature maps (not including direct connected input feature map, i.e. blue line), a compression layer with BN, ReLU and 1 × 1 convolution is applied to adjust the number of feature maps. Then an upsampling or downsampling layer is first operated so that all the feature maps are consistent in size with the output feature map. They are then concatenated to form a new feature map with $\frac{1}{8}$ of the size of the input image.

Adaptive aggregation of dense feature maps

In previous works, e.g. U-Net (Ronneberger, Fischer, and Brox 2015) for semantic segmentation, FPN (Lin et al. 2017b) for object detection and SharpMask (Pinheiro et al. 2016) for instance segmentation, feature maps are reused directly in the corresponding decoder module by concatenating the feature maps or adding them. Furthermore, RefineNet (Lin et al. 2017a), LRR (Ghiasi and Fowlkes 2016) and G-FRNet (Islam et al. 2017) refine the feature maps progressively stage by stage. Instead of just using previous feature maps as before, we introduce an adaptive aggregation structure to make better use of feature maps from previous blocks. As shown in Fig. 4, the feature maps from previous blocks are densely concatenated together by using the adaptive aggregation structure.

The adaptive aggregation structure takes all the feature maps from previous blocks ($B_1, B_2, ...$) as input. The feature maps from the lower layers (e.g. B_1, B_2) are of high resolution with coarse semantic information, whereas feature maps from the higher layers (e.g. B_3, B_4) are of low resolution with rich semantic information. The adaptive aggregation structure combines all previous feature maps to generate rich contextual information and also spatial information. For incoming feature maps, the scale sizes may be different. As shown in Fig. 4, the output feature map is $\frac{1}{8}$ of the size of the input image. To reduce memory consumption, we firstly use the convolutional layer to compress the incoming feature maps except for the direct connected feature map (which has been compressed). The compression layer is also composed of BN, ReLU and a 1 × 1 convolution operation. In order to make all feature maps consistent in size, we use the convolutional layer to downsample and the deconvolutional layer to upsample the feature maps. Intuitively, we directly concatenate the feature map if it is equivalent to the size of the output feature map. The convolutional layers are all composed of BN, ReLU and a 3 × 3 convolution operation with different strides. The deconvolutional layers are all composed of BN, ReLU and a 4 × 4 deconvolutional operation with different strides. At last, all the resultant feature maps $D_1^i, D_2^i, ..., D_M^i$ (M input feature maps) are concatenated into a new feature map F^i for the i^{th} stage, which is then fed to latter loss computation operation or dense block. The formulation for obtaining the i^{th} dense feature map from the previous feature maps can be written as follows:

$$\begin{aligned} D_1^i &= T_1^i(B_1), D_2^i = T_2^i(B_2), ..., D_M^i = T_M^i(B_M) \\ F^i &= [D_1^i, D_2^i, ..., D_M^i] \end{aligned} \quad (2)$$

where $T(\cdot)$ denotes the transformation operation (downsample or upsample). If B_j is of the same size as the output feature map, no operation is performaned on B_j. In addition, $[\cdots]$ stands for the concatenation operation.

In the adaptive aggregation structures for the three stages of the decoder module, the filter numbers in the compression layer for the reused feature map are set to 384, 256 and 128 respectively. The upsampling and downsampling layers will not change the dimension of feature maps.

Boundary-aware loss

In previous works, cross entropy loss function is often used in pipeline, which treat all pixels equally. As shown in Fig. 2, we can see that the pixels surrounding the boundary are "hard examples", which lead to bad prediction. Based on this observation, we construct a boundary-aware loss function, which guides the network to pay more attention on the pixels near the boundary. The loss function is

$$loss(L, L^{gt}) = -\frac{1}{N}\sum_{j=1}^{K}\sum_{I_i \in S_j}\sum_{c=1}^{C}\alpha_j L_{i,c}^{gt} w(L_{i,c}) log L_{i,c} \quad (3)$$

where L is the result of $softmax$ operation on the output feature map and L^{gt} is the groundtruth. The I_i is the i-th pixel in the image I and C is number of categories. We split all the N pixels of image I into several sets S_j based on the distance between the pixels and the boundary so that $I = \{S_1, S_2, ..., S_K\}$. We apply image dilation operation on the boundary with varying kernel size, which refers to as band width shown in Fig. 2, to obtain different set of pixels surrounding the boundary. α_j is the balancing weight and $w(L_{i,c})$ is an attention weight function. Motivated by (Lin et al. 2017c), we test two attention weight functions ($poly$ and exp): $w(L_{i,c}) = (1 - L_{i,c})^\lambda$ and $w(L_{i,c}) = e^{-\lambda(1-L_{i,c})}$. The λ is used to control attention weight. The ablation experiment results are shown in Table 2.

In order to further boost the gradient backpropagation and information flow, we compute multiple losses for different aggregated feature map F^i motivated by (Zhao et al. 2017; Islam et al. 2017; Fu et al. 2017). Specifically, F^i is fed to upsample module to obtain a feature map L^i with channel C, where C is number of classes in prediction labels. Then the feature map L^i is upsampled by using bilinear interpolation method directly to produce feature map $H \times W \times C$, which is used to compute pixel-wise loss with groundtruth. In terms of formula, the final loss L_{final} is computed as fol-

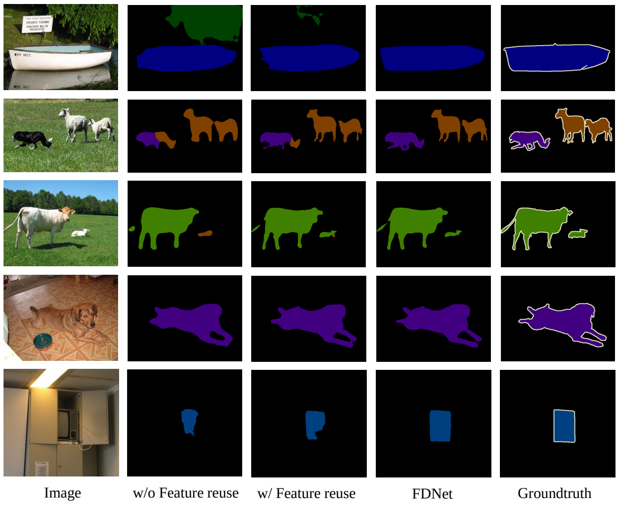

Figure 5: The effect of employing the proposed fully dense feature map reuse structure compared with other frameworks. Our proposed FDNet shows better results (Column **4**), especially on the **boundary localization**, compared with the results (Column **3**) of encoder-decoder structure with feature reuse method (Fig. 1 **Middle**) and the results (Column **2**) of encoder-decoder structure without feature reuse method (Fig. 1 **Top**).

lows:

$$L^i = softmax(U_i(F^i))$$
$$L_{final} = \sum_i loss(L^i, L^{gt}) \tag{4}$$

where $U_i(\cdot)$ denotes a upsample module with bilinear interpolation operation.

In the encoder module, the output feature map of each module is the concatenation of all the feature maps within this block, including the input. And the aggregated feature map is feature maps from all the previous blocks. Thus, each feature map in the encoder has much shorter path to loss compared with previous encoder-decoder structure (Lin et al. 2017a; Islam et al. 2017). The gradient backpropagation and information flowing is much more efficient. This will further boost our network optimization.

Implementation details

Training: The proposed FDNet is implemented with PyTorch on a single NVIDIA GTX 1080Ti. The weights of DenseNet-264 are directly employed in the encoder module of FDNet. In the training step, we adopt data augmentation similar to (Chen et al. 2016a). Random crops of 512×512 and horizontal flip is applied. We train the dataset with 30K iterations. We optimize the network by using the "poly" learning rate policy where the initial learning rate is multiplied by $(1 - \frac{iter}{max\_iter})^{power}$ with $power = 0.9$. The initial learning rate is set to 0.00025. We set momentum to 0.9 and weight decay to 0.0005.

Inference: In the inference step, we pad images with mean value before feeding full images into the network. We apply multi-scale inference, which is commonly used in semantic segmentation methods (Lin et al. 2017a; Fu et al. 2017). For multi-scale inference, we average the predictions on the same image across different scales for the final prediction. We set the scales ranging from 0.6 to 1.4. Horizontal flipping is also adopted in the inference. In the ablation experiments, we just use the single scale (i.e. scale = 1.0) and horizontal flipping method to do inference. In addition, we use the last-stage feature map of the decoder module to generate final prediction label map.

Table 1: The mean IoU scores (%) for encoder-decoder with different feature map reuse methods on PASCAL VOC 2012 validation dataset.

Encoder stride	w/o feature reuse	w/ feature reuse	dense feature reuse
32	77.2	78.5	78.9
16	78.2	79.1	79.4

Table 2: The mean IoU scores (%) for boundary-aware loss on PASCAL VOC 2012 validation dataset. The *poly* and *exp* represent different weighting methods.

loss		**mIoU**
CE		79.4
b-aware(*poly*)	$kernel = (10, 20, 30, 40), \lambda = 0$	
	$\alpha = (5, 4, 3, 2, 1)$	79.5
	$\alpha = (8, 6, 4, 2, 1)$	**80.3**
b-aware(*poly*)	$\alpha = (8, 6, 4, 2, 1), \lambda = 0$	
	$kernel = (5, 10, 15, 20)$	79.6
b-aware(*poly*)	$\alpha = (8, 6, 4, 2, 1),$	
	$kernel = (10, 20, 30, 40)$	
	$\lambda = 1$	80.0
	$\lambda = 2$	79.6
	$\lambda = 5$	77.7
b-aware(*exp*)	$\alpha = (8, 6, 4, 2, 1),$	
	$kernel = (10, 20, 30, 40)$	
	$\lambda = 0.25$	80.7
	$\lambda = 0.5$	80.3
	$\lambda = 0.75$	**80.9**
	$\lambda = 1$	80.6
	$\lambda = 2$	79.2

Experiments

In this section, we describe configurations of experimental datasets and show ablation experiments on PASCAL VOC 2012. At last, we report the results on two benchmark datasets: PASCAL VOC 2012 and NYUDv2.

Datasets description

To show the effectiveness of our approach, we conduct comprehensive experiments on PASCAL VOC 2012 dataset (Everingham et al. 2010) and NYUDv2 dataset (Silberman et al. 2012).

Table 3: GPU memory, number of parameters and some results on VOC 2012 test dataset are reported.

Methods	RefineNet-152	FDNet	SDN$_{M2}$
GPU Memory (MB)	4253	**2907**	-
Parameters (M)	109.2	**113.1**	161.7
mIOU	83.4	**84.2**	83.5

Table 4: Comparison of different mothods on PASCAL VOC 2012 validation dataset with mean IoU score (%). *FDNet-16s-MS* denotes the evaluation on multiple scales. *FDNet-16s-finetuning-MS* denotes fine-tuning on standard training data (1464 images) of PASCAL VOC 2012 dataset after training on the *trainaug* dataset.

Method	mIoU
Deeplab-MSc-CRF-LargeFOV	68.7
DeconvNet	67.1
DeepLabv2	77.7
G-FRNet	77.8
DeepLabv3	79.8
SDN	80.7
DeepLabv3+	81.4
FDNet-16s	80.9
FDNet-16s-MS	82.1
FDNet-16s-finetuning-MS	**84.1**

PASCAL VOC 2012: The dataset has 1,464 images for training, 1,449 images for validation and 1,456 images for testing, which involves 20 foreground object classes and one background class. Meanwhile, we augment the training set with extra labeled PASCAL VOC images provided by Semantic Boundaries Dataset (Hariharan et al. 2011), resulting in 10,582 images as *trainaug* dataset for training.

NYUDv2: The NYUDv2 dataset (Silberman et al. 2012) consists of 1449 RGB-D images showing indoor scenes. We use the segmentation labels provided in (Gupta, Arbelaez, and Malik 2013), in which all labels are mapped to 40 classes. We use the standard training/test split with 795 and 654 images, respectively. Only RGB images are used in our experiments.

Moreover, we perform a series of ablation evaluations on PASCAL VOC 2012 dataset with mean IoU score reported. We use the *trainaug* and validation dataset of PASCAL VOC 2012 for training and inference, respectively.

Feature map reuse

To verify the power of dense feature maps reuse, we compare our method with other two baseline frameworks. In this experiment, cross entropy loss is used. One is encoder-decoder structure without feature map reuse (Fig. 1 Top) and the other is encoder-decoder structure with naive feature map reuse (Fig. 1 Middle). We also compare the three frameworks on different encoder strides (the ratio of input image resolution to smallest output feature map of encoder, i.e. 16 and 32).

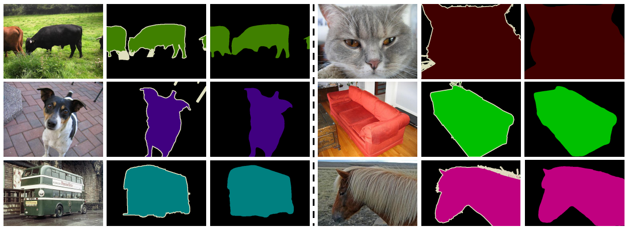

Figure 6: Some visual results on PASACAL VOC 2012 dataset. Three columns of each group are image, groundtruth and prediction label map.

The results are shown in Table 1. It is observed that the performance increases when feature maps are reused. Specifically, the performance for encoder-decoder (encoder stride = 32) without feature map reuse is only 77.2%. After the naive feature map reuse, the performance can increase to 78.5%. Furthermore, our fully dense feature map reuse can further improve the performance to 78.9%. In addition, when we adopt the stride 16 for the encoder module, the performance is much better than the original encoder with stride 32 on the three frameworks. This is because the spatial information loss is reduced by the encoder with smaller stride. We speculate that encoder with stride 8 can have better result similar to (Chen et al. 2017; 2018). Because of memory limitation, we only test on the encoder with stride 16 and 32.

We also show some predicted semantic label maps for different feature map reuse methods in Fig. 5. For the encoder-decoder structure without feature map reuse, the result is poor, especially for boundary localization. Though the naive feature map reuse method improves the segmentation result partially, it is still hard to obtain accurate pixel-wise prediction. The fully dense feature map reuse method shows very excellent results on the boundary localization.

Boundary-aware loss

In order to demonstrate the effect of proposed boundary-aware loss method, we take FDNet-16s as baseline to test the performance of different parameters. We mainly use the $kernel = (10, 20, 30, 40)$ and $kernel = (5, 10, 15, 20)$ by splitting the pixels into $K = 5$ sets (the remaining pixels are referred to as S_5). For *poly* weight method, the boundary-aware loss method (b-aware) degrades into cross entropy method (CE) when $\alpha = (1, 1, 1, 1, 1)$ and $\lambda = 0$. As shown in Table 2, the simply weighting on the pixels surrounding the boundary shows better performance compared with general cross entropy method, which enhances the performance by 0.9%. By fixing the α and $kernel$, we try different parameter λ in Table 2. Comparing the $poly$ and exp methods, we can observe that exp brings obvious improvement by 1.5%. On the contrary, the $poly$ methods lead to worse effect compared with baseline method (80.0 vs 80.3). In addition, the network cannot converge for $\lambda < 1$. We also compare the labeling accuracy for the pixels near the boundary. As shown in Fig. 2, the FDNet with boundary-aware loss shows obvious better performance for the pixels surrounding the boundary.

Table 5: Quantitative results (%) in terms of mean IoU on PASCAL VOC 2012 test set. Only VOC data is used as training data and denseCRF (Krähenbühl and Koltun 2011) is not included.

Method	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mIoU**
DeconvNet (Noh, Hong, and Han 2015)	89.9	39.3	79.7	63.9	68.2	87.4	81.2	86.1	28.5	77.0	62.0	79.0	80.3	83.6	80.2	58.8	83.4	54.3	80.7	65.0	72.5
Attention (Chen et al. 2016b)	86.0	38.8	78.2	63.1	70.2	89.6	84.1	82.9	29.4	75.2	58.7	79.3	78.4	83.9	80.3	53.5	82.6	51.5	79.2	64.2	71.5
Deeplabv2 (Chen et al. 2016a)	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
GCRF (Vemulapalli et al. 2016)	85.2	43.9	83.3	65.2	68.3	89.0	82.7	85.3	31.1	79.5	63.3	80.5	79.3	85.5	81.0	60.5	85.5	52.0	77.3	65.1	73.2
Adelaide (Lin et al. 2016)	90.6	37.6	80.0	67.8	74.4	92.0	85.2	86.2	39.1	81.2	58.9	83.8	83.9	84.3	84.8	62.1	83.2	58.2	80.8	72.3	75.3
LRR (Ghiasi and Fowlkes 2016)	91.8	41.0	83.0	62.3	74.3	93.0	86.8	88.7	36.6	81.8	63.4	84.7	85.9	85.1	83.1	62.0	84.6	55.6	84.9	70.0	75.9
G-FRNet (Islam et al. 2017)	91.4	44.6	91.4	69.2	78.2	95.4	88.9	93.3	37.0	89.7	61.4	90.0	91.4	87.9	87.2	63.8	89.4	59.9	87.0	74.1	79.3
PSPNet (Zhao et al. 2017)	91.8	71.9	**94.7**	71.2	75.8	95.2	89.9	95.9	39.3	90.7	71.7	90.5	94.5	88.8	89.6	**72.8**	89.6	64.0	85.1	76.3	82.6
SDN (Fu et al. 2017)	**96.2**	73.9	94.0	74.1	76.1	**96.7**	89.9	**96.2**	**44.1**	92.6	**72.3**	91.2	94.1	89.2	**89.7**	71.2	93.0	59.0	**88.4**	76.5	83.5
FDNet	95.5	**79.9**	88.6	**76.1**	**79.5**	**96.7**	**91.4**	95.6	40.1	**93.0**	71.5	**93.4**	**95.7**	**91.1**	89.2	69.4	**93.3**	**68.0**	88.3	**76.8**	**84.2**

Memory analysis

For semantic segmentation task, memory consumption and parameter number are both important issues. The proposed FDNet uses fully dense connected structure with nearly the same number of parameters compared with RefineNet (Lin et al. 2017a). As shown in Table 3, the FDNet consumes much less GPU memory (training process) compared with RefineNet. In addition, the memory consumption of FDNet can be reduced by using sharing memory efficiently based on (Geoff Pleiss* 2017). Compared with SDN (Fu et al. 2017), there are much less parameters for FDNet but the performance is much better.

PASCAL VOC 2012

We evaluate the performance on PASCAL VOC 2012 dataset following previous works (Lin et al. 2017a; Zhao et al. 2017). As FDNet-16s shows better performance (Table 1), we only report the performance of FDNet-16s in following experiments. We adopt boundary-aware method in the training step. As shown in Table 4, FDNet-16s achieves very comparable result with 82.1% mean IoU accuracy compared with previous works ((Chen et al. 2017; Islam et al. 2017; Fu et al. 2017)) when evaluated on multiple scales. Moreover, after fine-tuning the model on the standard training data (1464 images) of PASCAL VOC 2012 dataset, we achieve much better result with 84.1% mean IoU accuracy, which is the best result currently if not considering pretraining on other dataset (such as MS-COCO (Lin et al. 2014)). Some visual results with image, groundtruth and prediction label map are shown in Fig. 6.

Table 5 shows quantitative results of our method on the test dataset, where we only report the results using PASCAL VOC dataset. We achieve the best result with 84.2% on test data without pretraining on other datasets, which is the highest score when considering training on PASCAL VOC 2012 dataset. Though latest work DeepLabv3+ (Chen et al. 2018) achieve mean IoU score of 89.0% on test data of PASCAL VOC 2012, the result is relying on pretraining on much larger dataset MS-COCO (Lin et al. 2014) or JFT (Chollet 2017). In fact, FDNet-16s shows very comparable result compared with DeepLabv3+ on validation dataset (Table 4).

NYUDv2 Dataset

We conduct experiments on NYUDv2 dataset to compare FDNet-16s with previous works. We follow the training setup in PASCAL VOC 2012. Multi-scale inference is also adopted. The results are reported on Table 6. Similar to (Lin et al. 2017a), pixel accuracy, mean accuracy and mean IoU are used to evaluate all the methods. Some works make use of both depth image and RGB image as input and obtain very better result. For example, RDF (Park, Hong, and Lee 2017) achieves 50.1% (mean IoU) by using depth information. For a fair comparison, we only report the results training on only RGB images. As is shown, FDNet-16s outperforms previous work in terms of all metrics. In particular, our result is better than RefineNet (Lin et al. 2017a) by 0.9% in terms of mean IoU accuracy.

Table 6: Quantitative results (%) on NYUDv2 dataset (40 classes). The model is only trained on the provided training image dataset.

Method	pixel acc.	mean acc.	**mIoU**
FCN-32s	60.0	42.2	29.2
SegNet	66.1	36.0	23.6
Bayesian SegNet	68.0	45.8	32.4
FCN-HHA	65.4	46.1	34.0
Piecewise	70.0	53.6	40.6
RefineNet	73.6	58.9	46.5
FDNet-16s	**73.9**	**60.3**	**47.4**

Conclusion

In this paper, we have presented the fully dense neural network (FDNet) with encoder-decoder structure for semantic segmentation. For each layer of the FDNet in the decoder module, feature maps of almost all the previous layers are aggregated as the input. Furthermore, we propose boundary-aware loss function by paying more attention to the pixels surrounding the boundary. The proposed FDNet is very advantageous to semantic segmentation. On the one hand, the class boundaries as the spatial information are well reconstructed by using Encoder-Decoder structure with boundary-aware loss function. On the other hand, the FDNet learns more efficiently with the more efficient gradient backpropagation, much similar to the arguments already demonstrated in ResNet and DenseNet. The experiments show that our model outperforms previous works on two public benchmarks when any training on other datasets is not considered.

References

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2016a. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*.

Chen, L.-C.; Yang, Y.; Wang, J.; Xu, W.; and Yuille, A. L. 2016b. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*.

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

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

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. *CVPR*.

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

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

Geoff Pleiss*, Danlu Chen*, G. H. T. L. L. v. d. M. K. Q. W. 2017. Memory-efficient implementation of densenets. In *Technical report*.

Ghiasi, G., and Fowlkes, C. C. 2016. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*.

Gupta, S.; Arbelaez, P.; and Malik, J. 2013. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*.

Hariharan, B.; Arbeláez, P.; Bourdev, L.; Maji, S.; and Malik, J. 2011. Semantic contours from inverse detectors. In *ICCV*.

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

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*.

Huang, G.; Liu, Z.; v. Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2018. Multi-scale dense convolutional networks for efficient prediction. *ICLR*.

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

Islam, M. A.; Rochan, M.; Bruce, N. D.; and Wang, Y. 2017. Gated feedback refinement network for dense image labeling. In *CVPR*.

Krähenbühl, P., and Koltun, V. 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Lin, G.; Shen, C.; Van Den Hengel, A.; and Reid, I. 2016. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*.

Lin, G.; Milan, A.; Shen, C.; and Reid, I. 2017a. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. *CVPR*.

Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017b. Feature pyramid networks for object detection. In *CVPR*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017c. Focal loss for dense object detection. *ICCV*.

Liu, W.; Rabinovich, A.; and Berg, A. C. 2016. Parsenet: Looking wider to see better. *ICLR*.

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

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

Park, S.-J.; Hong, K.-S.; and Lee, S. 2017. Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. In *ICCV*.

Pinheiro, P. O.; Lin, T.-Y.; Collobert, R.; and Dollár, P. 2016. Learning to refine object segments. In *ECCV*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCI*.

Shen, Z.; Liu, Z.; Li, J.; Jiang, Y.-G.; Chen, Y.; and Xue, X. 2017. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor segmentation and support inference from rgbd images. In *ECCV*.

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

Vemulapalli, R.; Tuzel, O.; Liu, M.-Y.; and Chellapa, R. 2016. Gaussian conditional random field network for semantic segmentation. In *CVPR*.

Vijay, B.; Alex, K.; and Roberto, C. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*.

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