GFF: Gated Fully Fusion for Semantic Segmentation

Xiangtai Li[1]*, Houlong Zhao[2], Lei Han[3], Yunhai Tong[1], Kuiyuan Yang[2]

[1] Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
[2] DeepMotion [3] Tencent AI lab

{lxtpku,yhtong}@pku.edu.cn, {houlongzhao,kuiyuanyang}@deepmotion.ai, leihan.cs@gmail.com

Abstract

Semantic segmentation generates comprehensive understanding of scenes at a semantic level through densely predicting the category for each pixel. High-level features from Deep Convolutional Neural Networks already demonstrate their effectiveness in semantic segmentation tasks, however the coarse resolution of high-level features often leads to inferior results for small/thin objects where detailed information is important but missing. It is natural to consider importing low level features to compensate the lost detailed information in high level representations. Unfortunately, simply combining multi-level features is less effective due to the semantic gap existing among them. In this paper, we propose a new architecture, named Gated Fully Fusion(GFF), to selectively fuse features from multiple levels using gates in a fully connected way. Specifically, features at each level are enhanced by higher-level features with stronger semantics and lower-level features with more details, and gates are used to control the propagation of useful information which significantly reduces the noises during fusion. We achieve the state of the art results on two challenging scene understanding datasets, i.e., 82.3% mIoU on Cityscapes test set and 45.3% mIoU on ADE20K validation set. Codes and the trained models will be made publicly available.

1. Introduction

Semantic segmentation densely predicts the semantic category for every pixel in an image, such comprehensive image understanding is valuable for many vision-based applications such as medical image analysis [28], remote sensing [15] and autonomous driving [36]. However, precisely predicting label for every pixel is challenging as illustrated in Fig. 1, since these pixels can be from tiny or large objects, far or near objects, and inside object or object boundary.

As being a semantic prediction problem, the basic task of semantic segmentation is to generate high-level representation for each pixel, i.e., a high-level and high-resolution feature map.Given the ability of ConvNets in learning high-level representation from data, semantic segmentation has made much progress by leveraging such high-level representation. However, high-level representation from ConvNets is generated along lowering the resolution, thus high-resolution and high-level feature maps are distributed in two ends in a ConvNet.

To get a feature map with both high-resolution and high-level, which is not readily available in a ConvNet, it is natural to consider fusing high-level feature maps from higher layers and high-resolution feature maps from lower layers. These feature maps are with different properties, that high-level feature map can correctly predict most of the pixels on large patterns in a coarse manner, which dominants the current semantic segmentation approaches, while low-level feature maps can only predict few pixels on small patterns.

Unfortunately, simply combining high-level feature maps and high-resolution feature maps will drown useful information in massive useless information, and cannot get a desired high-level and high-resolution feature map.

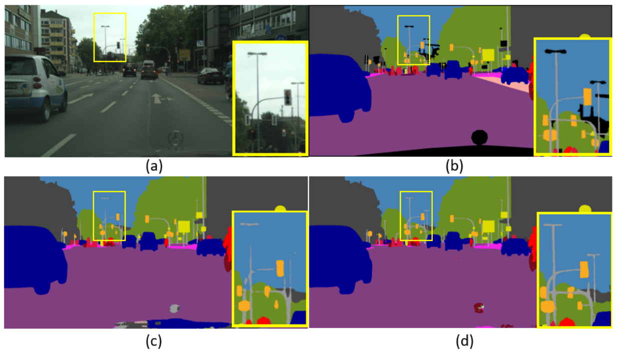

Figure 1. Illustration of challenges in semantic segmentation. (a), Input Image. (b), Ground Truth. (c), PSPNet result (d), Our result. Our method performs much better on small patterns such as distant poles and traffic lights.

*This work is done when Xiangtai Li is an intern at DeepMotion.

Therefore, an advanced fusion mechanism is required to collect information selectively from different feature maps. To achieve this, we propose Gated Fully Fusion (GFF) which uses a gate, a kind of operation commonly used in recurrent networks, at each pixel to measure the usefulness of each corresponding feature vector, and thus to control the propagation of the information through the gate. The principle of the gate at each layer is designed to either send out useful information to other layers or receive information from other layers when the information in the current layer is useless. Using gate to control information propagation, redundancies can also be effectively minimized in the network, allowing us to fuse multi-level feature maps in a fully-connected manner. Fig 1 compares the results of GFF and PSPNet [44], where GFF handles fine-level details such as poles in a much better way.

On the other hand, considering contextual information in large receptive field is also very important for semantic segmentation as proved by PSPNet [44], ASPP [3] and DenseASPP [37]. Therefore, we also model contextual information after GFF to achieve further performance improvement. Specifically, we propose a dense feature pyramid (DFP) module to encode the context information into each feature map. DFP reuses the contextual information for each feature level and aims to enhance the context modeling part while GFF operates on the backbone of network to capture more detailed information. Combining both components in a single end-to-end network, we achieve the state-of-the-art results on both Cityscapes and ADE20K datasets.

The main contributions of our work can be summarized as follows:

- Gated Fully Fusion is proposed to generate high-resolution and high-level feature map from multi-level feature maps.
- Detailed analysis with visualization of gates learned in different layers intuitively shows the information regulation mechanism in GFF.
- The proposed method is extensively verified on two standard semantic segmentation benchmarks including Cityscapes and ADE20K, and achieves new state-of-the-art performance. In particular, our model achieves 82.3% mIoU on Cityscapes test set trained only on the fine labeled data.

The rest of the paper is organized as follows: Section 2 provides an overview of the related works. Section 3 introduces the proposed method. Section 4 presents the experiments and analysis, and conclusion is made in Section 5.

2. Related Work

In this section, we review deep learning based methods for semantic segmentation from three categories, i.e., context modeling, multi-level feature fusion and gating mechanism.

Context modeling Though high-level feature maps in ConvNets have shown promising results on semantic segmentation [25], their receptive field sizes are still not large enough to capture contextual information for large objects and regions. Thus, context modeling becomes a practical direction in semantic segmentation. ParseNet [24] utilizes global pooling to encode contextual information, and PSPNet [44] uses spatial pyramid pooling to aggregate multi-scale contextual information. Deeplab series [2, 3, 4] develop atrous spatial pyramid pooling (ASPP) to capture multi-scale contextual information by dilated convolutional layers with different dilation rates. Instead of parallel aggregation as adopted in PSPNet and Deeplab, Yang et al. [37] and Bilinski et al. [1] follow the idea of the dense connection [13] to encode contextual information in a dense way. In [27], factorized large filters are directly used to increase the receptive field size for context modeling. In PSANet [45], contextual information is collected from all positions according to the similarities defined in a projected feature space. Similarly, OCNet [40] and DANet [9] use non-local operator [32] to aggregate information from the whole image based on similarities.

Multi-level feature fusion Despite a loss of contextual information, the top layer also lacks of fine detailed information. To address this issue, in FCN [25], predictions from middle layers are used to improve segmentation for detailed structures, while hypercolumns [10] directly combines features from multiple layers for prediction. The U-Net [28] adds skip connections between the encoder and decoder to reuse low level features, [43] improves U-Net by fusing high-level features into low-level features. Feature Pyramid Network (FPN) [23] uses the structure of U-Net with predictions made on each level of the feature pyramid. DeepLabV3+ [5] refines the decoder of its previous version by combing low-level features. In [21], Di et al. proposed to locally fuse every two adjacent feature maps in the feature pyramid into one feature map until only one feature map is left. These fusion methods operate locally in the feature pyramid without awareness of the usefulness of all feature maps to be fused, which limits the propagation of useful features.

Gating mechanism In deep neural networks, especially for recurrent networks, gates are commonly utilized to control the information propagation. For example, LSTM [12] and GRU [6] are two typical cases using different gates to handle long-term memory and dependencies. The highway network [30] uses gates to make training very deep network possible. To improve multi-task learning for scene parsing

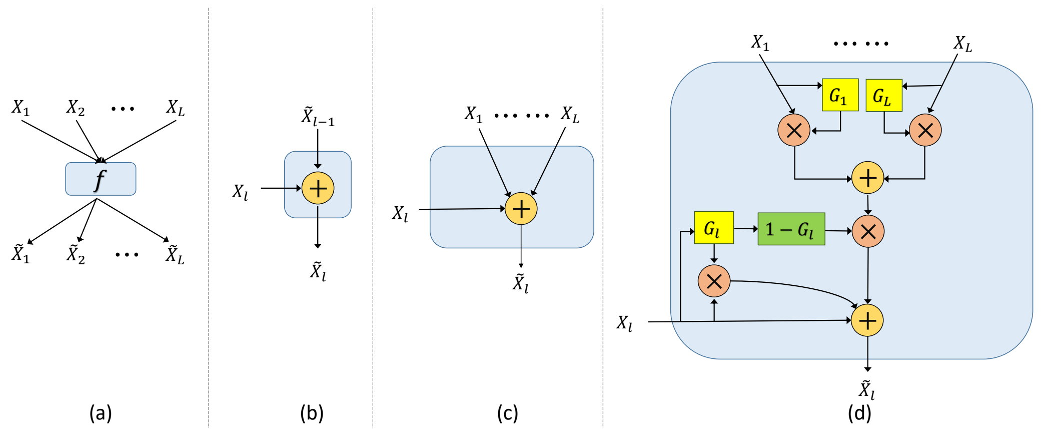

Figure 2. (a) shows a general fusion process f; (b) shows a fusing block in FPN; (c) shows a simple way of addition for fusion; and (d) is the proposed gated fully fusion, where G_l is the gate map generated from X_l, and features corresponding high gate values are allowed to send out and regions with low gate values are allowed to receive.

and depth estimation, PAD-Net [35] is proposed to use gates to fuse multi-modal features trained from multiple auxiliary tasks. DepthSeg[18] proposes depth-aware gating module which uses depth estimates to adaptively modify the pooling field size in high-level feature map.

Our method is related and inspired by the above methods, and differs from them that we propose to fuse multi-level feature maps simultaneously through gating mechanism, and the resulting method surpasses the state-of-the-art approaches.

3. Method

In this section, we first present the basic setting of multi-level feature fusion and three baseline fusion strategies. Then, we introduce the proposed multi-level fusion module (GFF) and the whole network with the context modeling module (DFP).

3.1. Multi-level Feature Fusion

Given L feature maps $\{X_i \in \mathbb{R}^{H_i \times W_i \times C_i}\}_{i=1}^{L}$ extracted from some backbone networks such as ResNet [11], where feature maps are ordered by their depth in the network with increasing semantics but decreasing details, H_i, W_i and C_i are the height, width and number of channels of the ith feature map respectively, feature maps of higher levels are with lower resolution due to the downsampling operations, i.e., $H_{i+1} \leq H_i, W_{i+1} \leq W_i$. In semantic segmentation, the top feature map X_L with $1/8$ resolution of the raw input image is mostly used for its rich semantics. The major limitation of X_L is its low spatial resolution without detailed information, because the outputs need to be with the same resolution as the input image. In contrast, feature maps of low level from shallow layers are with high resolution, but with limited semantics. Intuitively, combining the complementary strengths of multiple level feature maps would achieve the goal of both high resolution and rich semantics, and this process can be abstracted as a fusion process f, i.e.,

$$\{X_1, X_2 \cdots X_L\} \xrightarrow{f} \{\tilde{X}_1, \tilde{X}_2 \cdots \tilde{X}_L\} \tag{1}$$

where $\tilde{X}_l$ is the fused feature map for the lth level. Fig 2 illustrates several fusion strategies to be discussed. To simplify the notations in following equations, bilinear sampling and 1×1 convolution are ignored which are used to reshape the feature maps at the right hand side to let the fused feature maps have the same size as those at the left hand side.

Concatenation is a straightforward operation to aggregate all the information in multiple feature maps, but it mixes the useful information with large amount of non-informative features. Addition is another simple way to combine feature maps by adding features at each position, while it suffers from the similar problem as concatenation. FPN [23] conducts the fusion process through a top-down pathway with lateral connections, where semantic features in higher levels are gradually fused into lower levels. The three fusion strategies can be formulated as,

$$\text{Concat: } \tilde{X}_l = \text{concat}(X_1, ..., X_L), \tag{2}$$

$$\text{Addition: } \tilde{X}_l = \sum_{i=1}^{L} X_i, \tag{3}$$

$$\text{FPN: } \tilde{X}_l = \tilde{X}_{l+1} + X_l, \text{ where } \tilde{X}_L = X_L. \tag{4}$$

The problem of these basic fusion strategies is that feature maps are fused together without measuring the usefulness of each feature vector, and massive useless features are mixed with useful feature during fusion.

3.2. Gated Fully Fusion

The basic task in multi-level feature fusion is to aggregate useful information together under interference of massive useless information. Gating is a mature mechanism to

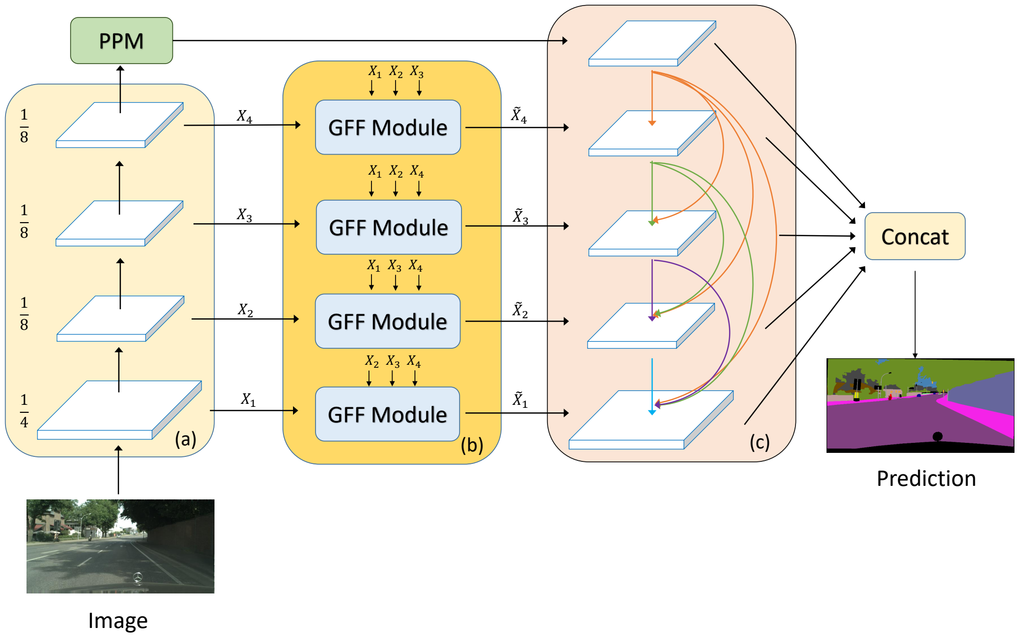

Figure 3. Illustration of the overall architecture. (a) Backbone Network(e.g. ResNet [11]) with pyramid pooling module (PPM) [44] on the top. The backbone provides a pyramid of features at different levels. (b), Feature pyramid through gated fully fusion (GFF) modules. The detail of GFF module is illustrated in Fig 2 . (c), Then the final features containing context information are obtained from a dense feature pyramid (DFP) module.

measure the usefulness of each feature vector in a feature map and aggregates information accordingly. In this paper, Gated Fully Fusion (GFF) is designed based on the simple addition-based fusion by controlling information flow with gates. Specifically, each level l is associated with a gate map $G_l \in [0,1]^{H_l \times W_l}$. With these gate maps, the addition-based fusion is formally defined as

$$\tilde{X}_l = (1 + G_l) \cdot X_l + (1 - G_l) \cdot \sum_{i=1, i \neq l}^{L} G_i \cdot X_i, \quad (5)$$

where $\cdot$ denotes element-wise multiplication broadcasting in the channel dimension, each gate map $G_l = \text{sigmoid}(\mathrm{w}_i * X_i)$ is estimated by a convolutional layer parameterized with $\mathrm{w}_i \in \mathbb{R}^{1\times1\times C_i}$. A feature vector at (x,y) from level $i, i \neq l$ can be fused to l only when $G_i(x,y)$ is large **and** $G_l(x,y)$ is small, i.e., information is sent when level i has the useful information that level l is missing. Besides that useful information can be regulated to the right place through gates, useless information can also be effectively suppressed on both the sender and receiver sides, and information redundancy can be avoided because the information is only received when the current position has useless features.

3.3. Dense Feature Pyramid

Context modeling aims to encode more global information, and it is orthogonal to the proposed GFF. Therefore, we further design a module to encode more contextual information from outputs of both PSPNet [44] and GFF. Considering dense connections can strengthen feature propagation [13, 37], we also densely connect the feature maps in a top-down manner starting from feature map outputted from the PSPNet, and high-level feature maps are reused multiple times to add more contextual information to low levels, which was found important in our experiments for correctly segmenting large objects. Since the feature pyramid is in a densely connected manner, we denote this module as Dense Feature Pyramid (DFP). Both GFF and DFP can be plugged into any existing FCNs for end-to-end training with only slightly extra computation cost.

3.4. Network Architecture

Our network architecture is designed based on previous state-of-the-art network PSPNet [44] with ResNet [11] as backbone for basic feature extraction, the last two stages in ResNet are modified with dilated convolution to make both strides to 1 and keep spatial information. Fig 3 shows the overall framework including both GFF and DFP. PSPNet forms the bottom-up pathway with backbone network and pyramid pooling module (PPM), where PPM is at the top to encode contextual information. Feature maps from last residual blocks in each stage of backbone are used as the input for GFF module, and all feature maps are reduced to 256 channels with 1×1 convolutional layers. The output feature maps from GFF are further fused with two 3×3 convolutional layers in each level before feeding into the DFP module. All convolutional layers are followed by batch normalization [14] and ReLU activation function. After DFP, all feature maps are concatenated for final semantic segmentation. Comparing with the basic PSPNet, the proposed method only slightly increases the number of parameters and computations. The entire network is trained in an end-to-end manner driving by cross-entropy loss defined on the segmentation benchmarks. To facilitate the training process, an auxiliary loss together with the main loss are used to help optimization following [19, 44], where the main loss is defined on the final output of the network and the auxiliary loss is defined on the output feature map at stage3 of ResNet with weight of 0.4 [44].

4. Experiment

In this section, we evaluate and analyze the proposed method on Cityscapes [7] and ADE20K [46].

4.1. Implementation Details

Our implementation is based on PyTorch [26], and uses ResNet50 and ResNet101 pre-trained from ImageNet [29] as backbones.

Training settings: the weight decay is set to 1e-4. Standard SGD is used for optimization, and "poly" learning rate scheduling policy is used to adjust learning rate, where initial learning rate is set to 1e-3 and decayed by $(1-\frac{iter}{max\_iter})^{power}$ with $power = 0.9$. Synchronized batch normalization [41] is used for better mean/variance estimation due to the limited number of images that can be hosted in each GPU. For Cityscapes, crop size of 864×864 is used, 100K training iterations with mini-batch size of 8 is carried for training. For ADE20K, crop size of 512×512 is used (images with side smaller than the crop size are padded with zeros), 150K training iterations are used with mini-batch size of 16. As a common practice to avoid overfitting, data augmentation including random horizontal flipping, random cropping, random color jittering within the range of $[-10, 10]$, and random scaling in the range of $[0.75, 2]$ are used during training.

4.2. Cityscapes Dataset

Cityscapes is a large-scale dataset for semantic urban scene understanding. It contains 5000 fine pixel-level annotated images, which is divided into 2975, 500, and 1525 images for training, validation and testing respectively, where labels of training and validation are publicly released and labels of testing set are hold for online evaluation. It also provides 20000 coarsely annotated images. 30 classes are annotated and 19 of them are used for pixel-level semantic labeling task. Images are in high resolution with the size of 1024×2048. The evaluation metric for this dataset is mean Intersection over Union (mIoU).

Method	mIoU(%)
PSPNet(Baseline)	78.6
PSPNet + Concat	78.8 (0.2↑)
PSPNet + Addition	78.7 (0.1↑)
PSPNet + FPN	79.3 (0.7↑)
PSPNet + Gated FPN	79.4 (0.8 ↑)
PSPNet + GFF	80.4 (1.8↑)

Table 1. Comparison experiments on Cityscapes validation set, where PSPNet serves as the baseline method.

Strong Baseline We choose PSPNet [44] as our baseline model which achieved state-of-the-art performance for semantic segmentation. We re-implement PSPNet on Cityscapes and achieve similar performance with mIoU of 78.6% on validation set. All results are reported by using sliding window crop prediction.

Ablation Study on Feature Fusion Methods First, we compare several methods introduced in Section 3. To speed up the training process, we use weights from the trained PSPNet to initialize the parameters in each fusion method. We use train-fine data (2975 images) for training and report performance on validation set. For fair comparison with concatenation and addition, we also reduce the channel dimension of feature maps to be fused to 256 and use two 3×3 convolutional layers to refine the fused feature map. As for FPN, we implement the original FPN for semantic segmentation following [17] and we add it to PSPNet. Note that FPN based on PSPNet fuses 5 feature maps, where one is context feature map from pyramid pooling module and others are from the backbone.

All the results are shown in Table 1. As expected, concatenation and addition only slightly improve the baseline, and FPN achieves the best performance among the three base fusion methods, while the proposed GFF obtains even much more improvement with mIoU of 80.4%. Since GFF is a gated version of addition-based fusion, the results demonstrate the effectiveness of the used gating mechanism. For further comparison, we also add the proposed gating mechanism into top-down pathway of FPN and obtains slightly improvement, which is reasonable since most high-level features are useful for low levels. This demonstrates the advantage of fully fusing multi-level feature maps, and the importance of gating mechanism especially during fusing low-level features to high levels. Fig 4 shows results after using GFF, where the accuracies of predictions for both distant objects and object boundaries are significantly improved.

Ablation Study for Improvement Strategies We per-

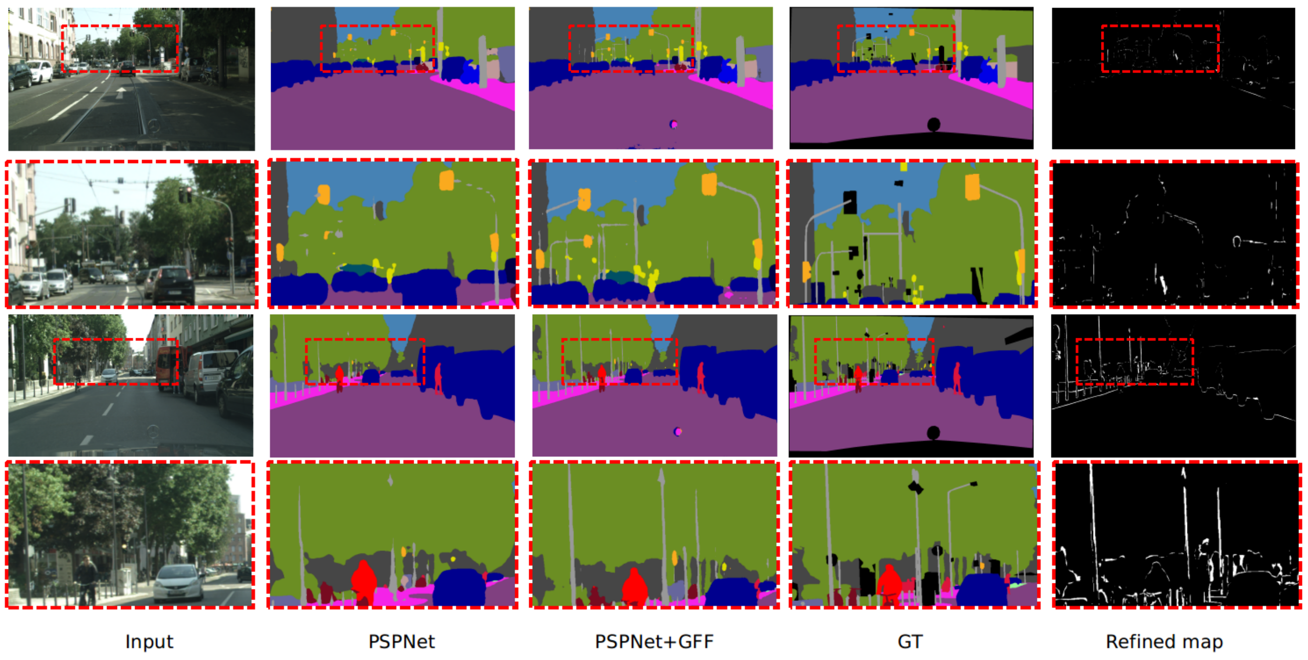

Figure 4. Visualization of segmentation results on two images using GFF and PSPNet. First column shows two input images with zoomed in regions marked with red dash rectangles. The second column shows results of PSPNet, and the third column shows results of using GFF. Fourth column lists the ground truth. Last column shows the refine parts by GFF. It shows that GFF can well handle distant missing objects like poles, traffic lights and object boundaries. Best view in color.

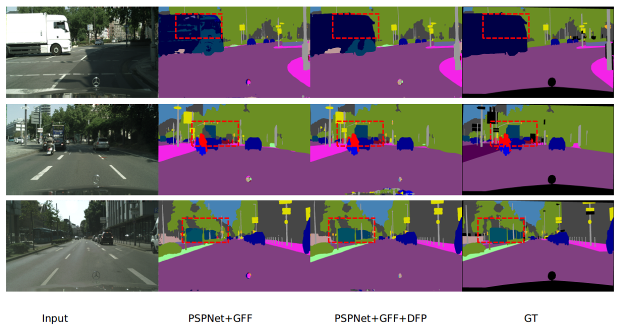

Figure 5. DFP enhances segmentation results on large scale objects and generates more consistent results. Best view in color and zoom in.

Method	mIoU(%)
PSPNet(Baseline)	78.6
PSPNet + GFF	80.4 (1.8↑)
PSPNet + GFF + DFP	81.2 (2.6↑)
PSPNet + GFF + DFP + MS	81.8 (3.2↑)

Table 2. Comparison experiments on Cityscapes validation set, where PSPNet serves as the baseline method.

Method	mIoU(%)	FLOPS(G)	Params(M)
PSPNet(Baseline)	78.6	580.1	65.6
PSPNet + GFF	80.4	600.1	69.7
PSPNet + GFF + DFP	81.2	625.5	70.5

Table 3. Computational cost comparison, where PSPNet serves as the baseline with image of size 512×512 as input.

form two strategies to further boost the performance of our model:, (1) DFP: Dense Feature Pyramid is used after the output of GFF module; and (2) MS: multi-scale inference is adopted, where the final segmentation map is averaged from segmentation probability maps with scales $\{0.75, 1, 1.25, 1.5, 1.75\}$ for evaluation. Experimental results are shown in Table 2, and DFP further improves the performance by 0.8% mIoU. Fig. 5 shows several visual comparisons, where DFP generates more consistent segmentation inside large objects and demonstrates the effectiveness in using contextual information for resolving local ambiguities.

With multi-scale inference, our model achieves 81.8% mIoU, which significantly outperforms previous state-of-the-art model DeepLabv3+ (79.55% on Cityscapes validation set) by 2.25%.

In Table 3, we also study the computational cost of using our modules, where our method spends 7.7% more computational cost and 6.3% more parameters comparing with the baseline PSPNet.

Comparison to the State-of-the-Art As a common practice toward best performance, we average the predic-

Method	Backbone	mIoU(%)
PSPNet [44]†	ResNet101	78.4
PSANet [45]†	ResNet101	78.6
Ours †	ResNet101	**80.9**
RefineNet [22]‡	ResNet101	73.6
SAC [42]‡	ResNet101	78.1
AAF [16]‡	ResNet101	79.1
BiSeNet [38]‡	ResNet101	78.9
PSANet [45]‡	ResNet101	80.1
DFN [39]‡	ResNet101	79.3
DepthSeg [18]‡	ResNet101	78.2
DenseASPP [37] ‡	DenseNet161	80.6
DANet [9] ‡	ResNet101	81.5
Ours‡	ResNet101	**82.3**

Table 4. State-of-the-art comparison experiments on Cityscapes test set. †means only using the train-fine dataset. ‡means both the train-fine and val-fine data are used.

Method	Backbone	mIoU(%)	Pixel Acc.(%))
PSPNet [44]	ResNet50	42.78	80.76
PSANet [45]	ResNet50	**42.97**	**80.92**
EncNet [41]	ResNet50	41.11	79.73
GCUNet [20]	ResNet50	42.60	79.51
Ours	ResNet50	**42.92**	**81.03**
RefineNet [22]	ResNet101	40.20	-
PSPNet [44]	ResNet101	43.29	81.39
PSANet [45]	ResNet101	43.77	81.51
UperNet [34]	ResNet101	41.22	79.98
EncNet [41]	ResNet101	44.65	81.69
GCUNet [20]	ResNet101	44.81	81.19
Ours	ResNet101	**45.33**	**82.01**

Table 5. State-of-the-art comparison experiments on ADE20K validation set. Our models achieve top performance measued by both mIoU and pixel accuracy.

tions of multi-scaled, left-right flipped and overlapping-tiled images for inference. For fair comparison, all methods are only trained using fine annotated dataset and evaluated on test set by the evaluation server. Table 4 summarizes the comparisons, our method achieves 80.9% mIoU by only use train-fine dataset and outperforms PSANet [45] by 2.3%. By fine-tuning the model on both train-fine and val-fine datasets, our method achieves best mIoU of 82.3%. Detailed per-category comparisons are reported in Table 6, where our method achieves the highest IoU on 15 out of 19 categories, and large improvements are from small/thin categories such as pole, street light/sign, person and rider. More detailed analysis will be given by gate visualization.

4.3. ADE20K Dataset

ADE20K is a challenging scene parsing dataset annotated with 150 classes, and it contains 20K/2K images for training and validation. Images in this dataset are from more different scenes with more small scale objects, and are with varied sizes including max side larger than 2000 and min side smaller than 100. Follow the standard protocol, both mIoU and pixel accuracy evaluated on validation set are used as the performance metrics. Both ResNet50 and ResNet101 are used as backbones.

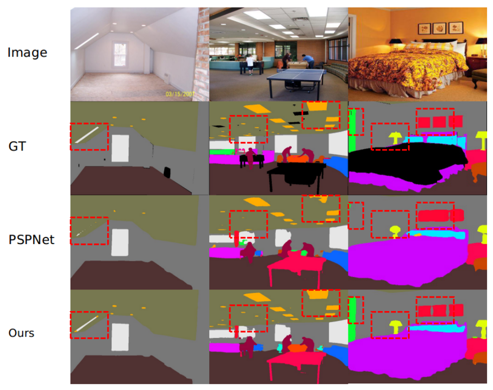

Figure 6. Visualization results on ADE20K validation dataset (ResNet101 as backbone). Comparing with PSPNet, our method captures more detailed information, and finds missing small objects (e.g., lights in first two examples) and generates "smoother" on object boundaries (e.g., figures on the wall in last example). Best view in color.

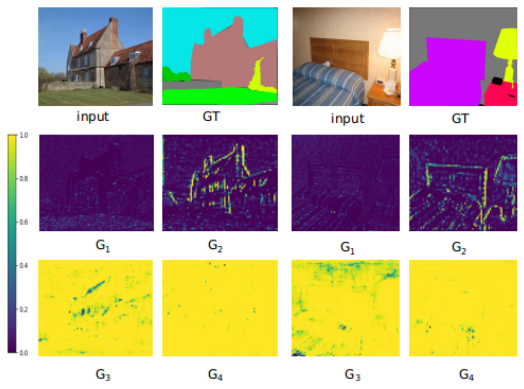

Figure 7. Visualization of learned gate maps on ADE20K dataset. G_i represents gate map of the ith layer. Best view in color and zoom in for detailed information.

As performance comparison listed in Table 5, with ResNet50, all methods achieve similar performance, and our method achieves comparable mIoU comparing with PSANet but with slightly better pixel accuracy. With stronger backbone ResNet101, our method outperforms state-of-the-art methods with considerable margin in terms of both mIoU and pixel accuracy. Several visual comparison results are shown in Fig 6, where our method performs much better at details and object boundaries.

4.4. Visualization of Gates

In this section, we visualize what gates have learned and analyze how gates control the information propagation. Fig 7 shows the gates learned from ADE20K and Fig 8(a) shows the gates learned from Cityscapes respectively. For each input image, we show the learned gate map of each

Method	road	swalk	build	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
DUC-HDC [31]	98.5	85.5	92.8	58.6	55.5	65.0	73.5	77.8	93.2	72.0	95.2	84.8	68.5	95.4	70.9	78.7	68.7	65.9	73.8	77.6
ResNet38 [33]	98.5	85.7	93.0	55.5	59.1	67.1	74.8	78.7	93.7	72.6	95.5	86.6	69.2	95.7	64.5	78.8	74.1	69.0	76.7	78.4
PSPNet [44]	98.6	86.2	92.9	50.8	58.8	64.0	75.6	79.0	93.4	72.3	95.4	86.5	71.3	95.9	68.2	79.5	73.8	69.5	77.2	78.4
AAF [16]	98.5	85.6	93.0	53.8	58.9	65.9	75.0	78.4	93.7	72.4	95.6	86.4	70.5	95.9	73.9	82.7	76.9	68.7	76.4	79.1
SegModel [8]	98.6	86.4	92.8	52.4	59.7	59.6	72.5	78.3	93.3	72.8	95.5	85.4	70.1	95.6	75.4	84.1	75.1	68.7	75.0	78.5
DFN [39]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	79.3
BiSeNet [38]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	78.9
PSANet [45]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	80.1
DenseASPP [37]	98.7	87.1	93.4	**60.7**	62.7	65.6	74.6	78.5	93.6	72.5	95.4	86.2	71.9	96.0	78.0	90.3	80.7	69.7	76.8	80.6
DANet [9]	98.6	87.1	93.5	56.1	63.3	69.7	77.3	81.3	93.9	**72.9**	95.7	87.3	72.9	96.2	76.8	89.4	**86.5**	**72.2**	78.2	81.5
Ours	**98.7**	**87.2**	**93.9**	59.6	**64.3**	**71.5**	**78.3**	**82.2**	**94.0**	72.6	**95.9**	**88.2**	**73.9**	**96.5**	**79.8**	**92.2**	84.7	71.5	**78.8**	**82.3**

Table 6. Per-category results on Cityscapes test set. Note that all the models are trained with only fine annotated data. Our method outperforms existing approaches on 15 out of 19 categories, and achieves 82.3% mIoU.

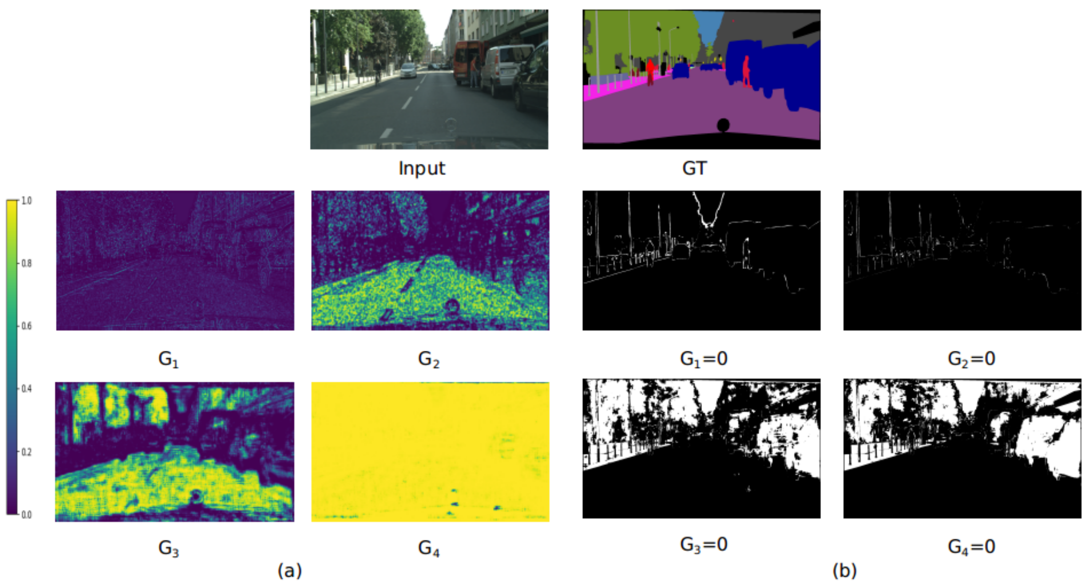

Figure 8. (a) Visualization of learned gate maps on Cityscapes dataset, where G_i represents the gate map of the ith layer. (b) Wrongly classified pixels are highlighted after setting G_i to 0 comparing with using original gate values. Best view in color and zoom in for detailed information.

level. As expected, we find that the higher-level features (e.g., G_3, G_4) are more useful for large structures with explicit semantics, while the lower-level features (e.g., G_1 and G_2) are mainly useful for local details and boundaries.

Functionally, we find that the higher level features always spread information to other layers and only receive sparse feature signals. For example, the gate from stage 4 (in G_4 of Fig 8) shows that almost all pixels are of high-confidence. Higher-level features cover large receptive field with fewer details, and they can provide a ground scope of the main semantics.

In contrast, the lower level layers prefer to receive information while only spread a few sparse signals. This verifies that lower level representations generally vary frequently along the spatial dimension and they require additional features as semantic supplement, while a benefit is that lower features can provide precise information for details and object boundaries (G_2 in Fig 7 and G_1 in Fig 8(a)).

To further verify the effectiveness of the learned gates, we set the value of each gate G_i to zero and compare the segmentation results with learned gate values. Fig 8 (b) shows the comparison results, where wrongly predicted pixels after setting G_i to zero are highlighted. Information through G_1 and G_2 is mainly help for object boundaries, while information through G_3 and G_4 is mainly help for large patterns such as cars. Additional visualization examples for the gates can be found in the supplementary materials.

5. Conclusion

In this work, we propose Gated Fully Fusion (GFF) to fully fuse multi-level feature maps controlled by learned gate maps. The novel module bridges the gap between high resolution with low semantics and low resolution with high semantics. We explore the proposed GFF for the task of semantic segmentation and achieve new state-of-the-art results on Cityscapes and ADE20K datasets. In particular, we find that the missing low-level features can be fused into each feature level in the pyramid, which indicates that our module can well handle small and thin objects in the scene. In our future work, we will verify the effectiveness of GFF in object detection tasks where fine details are also important.

References

[1] P. Bilinski and V. Prisacariu. Dense decoder shortcut connections for single-pass semantic segmentation. In *CVPR*, 2018.

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

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 2018.

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

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

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[8] S. Y. Falong Shen, Gan Rui and G. Zeng. Semantic segmentation via structured patch prediction, context crf and guidance crf. In *CVPR*, 2017.

[9] J. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu. Dual attention network for scene segmentation. *arXiv preprint arXiv:1809.02983*, 2018.

[10] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

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

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

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

[15] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *CVPRW*, 2016.

[16] T.-W. Ke, J.-J. Hwang, Z. Liu, and S. X. Yu. Adaptive affinity fields for semantic segmentation. In *ECCV*, 2018.

[17] A. Kirillov, K. He, R. B. Girshick, and P. Dollr. A unified architecture for instance and semantic segmentation, 2017.

[18] S. Kong and C. C. Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *CVPR*, 2018.

[19] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, 2015.

[20] Y. Li and A. Gupta. Beyond grids: Learning graph representations for visual recognition. In *NeurIPS*. 2018.

[21] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang. Multi-scale context intertwining for semantic segmentation. In *ECCV*, 2018.

[22] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[23] T.-Y. Lin, P. Dollr, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[24] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

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

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[27] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters improve semantic segmentation by global convolutional network. In *CVPR*, 2017.

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

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[30] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NIPS*, 2015.

[31] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018.

[32] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, June 2018.

[33] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016.

[34] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[35] D. Xu, W. Ouyang, X. Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, 2018.

[36] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *CVPR*, 2017.

[37] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang. Denseaspp for semantic segmentation in street scenes. In *CVPR*, 2018.

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

[39] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Learning a discriminative feature network for semantic segmentation. In *CVPR*, 2018.

[40] Y. Yuan and J. Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.

[41] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[42] R. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan. Scale-adaptive convolutions for scene parsing. In *ICCV*, 2017.

[43] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018.

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

[45] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, and J. Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018.

[46] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *arXiv preprint arXiv:1608.05442*, 2016.