Large Kernel Matters ——
Improve Semantic Segmentation by Global Convolutional Network

Chao Peng Xiangyu Zhang Gang Yu Guiming Luo Jian Sun
School of Software, Tsinghua University, {pengc14@mails.tsinghua.edu.cn, gluo@tsinghua.edu.cn}
Megvii Inc. (Face++), {zhangxiangyu, yugang, sunjian}@megvii.com

Abstract

One of recent trends [30, 31, 14] in network architecture design is stacking small filters (e.g., 1x1 or 3x3) in the entire network because the stacked small filters is more efficient than a large kernel, given the same computational complexity. However, in the field of semantic segmentation, where we need to perform dense per-pixel prediction, we find that the large kernel (and effective receptive field) plays an important role when we have to perform the classification and localization tasks simultaneously. Following our design principle, we propose a Global Convolutional Network to address both the classification and localization issues for the semantic segmentation. We also suggest a residual-based boundary refinement to further refine the object boundaries. Our approach achieves state-of-art performance on two public benchmarks and significantly outperforms previous results, ***82.2%*** *(vs 80.2%) on PASCAL VOC 2012 dataset and* ***76.9%*** *(vs 71.8%) on Cityscapes dataset.*

1. Introduction

Semantic segmentation can be considered as a per-pixel classification problem. There are two challenges in this task: 1) **classification**: an object associated to a specific semantic concept should be marked correctly; 2) **localization**: the classification label for a pixel must be aligned to the appropriate coordinates in output score map. A well-designed segmentation model should deal with the two issues simultaneously.

However, these two tasks are naturally contradictory. For the classification task, the models are required to be invariant to various transformations like translation and rotation. But for the localization task, models should be transformation-sensitive, i.e., precisely locate every pixel for each semantic category. The conventional semantic segmentation algorithms mainly target for the localization issue, as shown in Figure 1 B. But this might decrease the classification performance.

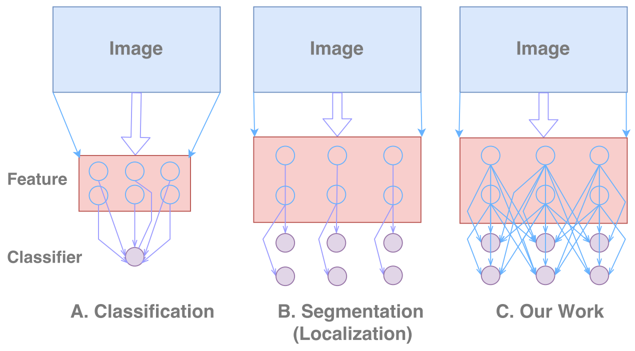

Figure 1. A: Classification network; B: Conventional segmentation network, mainly designed for localization; C: Our Global Convolutional Network.

In this paper, we propose an improved net architecture, called Global Convolutional Network (GCN), to deal with the above two challenges simultaneously. We follow two design principles: 1) from the localization view, the model structure should be fully convolutional to retain the localization performance and no fully-connected or global pooling layers should be used as these layers will discard the localization information; 2) from the classification view, large kernel size should be adopted in the network architecture to enable densely connections between feature maps and per-pixel classifiers, which enhances the capability to handle different transformations. These two principles lead to our GCN, as in Figure 2 A. The FCN [25]-like structure is employed as our basic framework and our GCN is used to generate semantic score maps. To make global convolution practical, we adopt symmetric, separable large filters to reduce the model parameters and computation cost. To further improve the localization ability near the object boundaries, we introduce *boundary refinement* block to model the boundary alignment as a residual structure, shown in Figure 2 C. Unlike the CRF-like post-process [6], our boundary refinement block is integrated into the network and trained end-to-end.

Our contributions are summarized as follows: 1) we propose Global Convolutional Network for semantic segmentation which explicitly address the “classification” and “localization” problems simultaneously; 2) a Boundary Refinement block is introduced which can further improve the localization performance near the object boundaries; 3) we achieve state-of-art results on two standard benchmarks, with **82.2%** on PASCAL VOC 2012 and **76.9%** on the Cityscapes.

2. Related Work

In this section we quickly review the literatures on semantic segmentation. One of the most popular CNN based work is the Fully Convolutional Network (FCN) [25]. By converting the fully-connected layers into convolutional layers and concatenating the intermediate score maps, FCN has outperformed a lot of traditional methods on semantic segmentation. Following the structure of FCN, there are several works trying to improve the semantic segmentation task based on the following three aspects.

Context Embedding in semantic segmentation is a hot topic. Among the first, Zoom-out [26] proposes a hand-crafted hierarchical context features, while ParseNet [23] adds a global pooling branch to extract context information. Further, Dilated-Net [36] appends several layers after the score map to embed the multi-scale context, and Deeplab-V2 [7] uses the *Atrous Spatial Pyramid Pooling*, which is a combination of convolutions, to embed the context directly from feature map.

Resolution Enlarging is another research direction in semantic segmentation. Initially, FCN [25] proposes the deconvolution (i.e. inverse of convolution) operation to increase the resolution of small score map. Further, Deconv-Net [27] and SegNet [3] introduce the *unpooling* operation (i.e. inverse of pooling) and a glass-like network to learn the upsampling process. More recently, LRR [12] argues that upsampling a feature map is better than score map. Instead of learning the upsampling process, Deeplab [24] and Dilated-Net [36] propose a special *dilated convolution* to directly increase the spatial size of small feature maps, resulting in a larger score map.

Boundary Alignment tries to refine the predictions near the object boundaries. Among the many methods, Conditional Random Field (CRF) is often employed here because of its good mathematical formation. Deeplab [6] directly employs *denseCRF* [18], which is a CRF-variant built on fully-connected graph, as a post-processing method after CNN. Then CRFAsRNN [37] models the denseCRF into a RNN-style operator and proposes an end-to-end pipeline, yet it involves too much CPU computation on Permutohedral Lattice [1]. DPN [24] makes a different approximation on denseCRF and put the whole pipeline completely on GPU. Furthermore, Adelaide [21] deeply incorporates CRF and CNN where hand-crafted potentials is replaced by convolutions and nonlinearities. Besides, there are also some alternatives to CRF. [4] presents a similar model to CRF, called *Bilateral Solver*, yet achieves 10x speed and comparable performance. [16] introduces the *bilateral filter* to learn the specific pairwise potentials within CNN.

In contrary to previous works, we argues that semantic segmentation is a classification task on large feature map and our Global Convolutional Network could simultaneously fulfill the demands of classification and localization.

3. Approach

In this section, we first propose a novel Global Convolutional Network (GCN) to address the contradictory aspects — classification and localization in semantic segmentation. Then using GCN we design a fully-convolutional framework for semantic segmentation task.

3.1. Global Convolutional Network

The task of semantic segmentation, or pixel-wise classification, requires to output a score map assigning each pixel from the input image with semantic label. As mentioned in Introduction section, this task implies two challenges: *classification* and *localization*. However, we find that the requirements of classification and localization problems are naturally contradictory: (1) For classification task, models are required invariant to transformation on the inputs — objects may be shifted, rotated or rescaled but the classification results are expected to be unchanged. (2) While for localization task, models should be transformation-sensitive because the localization results depend on the positions of inputs.

In deep learning, the differences between classification and localization lead to different styles of models. For classification, most modern frameworks such as AlexNet [20], VGG Net [30], GoogleNet [31, 32] or ResNet [14] employ the ”Cone-shaped” networks shown in Figure 1 A: features are extracted from a relatively small hidden layer, which is coarse on spatial dimensions, and classifiers are **densely connected** to entire feature map via fully-connected layer [20, 30] or global pooling layer [31, 32, 14], which makes features robust to locally disturbances and allows classifiers to handle different types of input transformations. For localization, in contrast, we need relatively large feature maps to encode more spatial information. That is why most semantic segmentation frameworks, such as FCN [25, 29], DeepLab [6, 7], Deconv-Net [27], adopt ”Barrel-shaped” networks shown in Figure 1 B. Techniques such as Deconvolution [25], Unpooling [27, 3] and Dilated-Convolution [6, 36] are used to generate high-resolution feature maps, then classifiers are connected **locally** to each

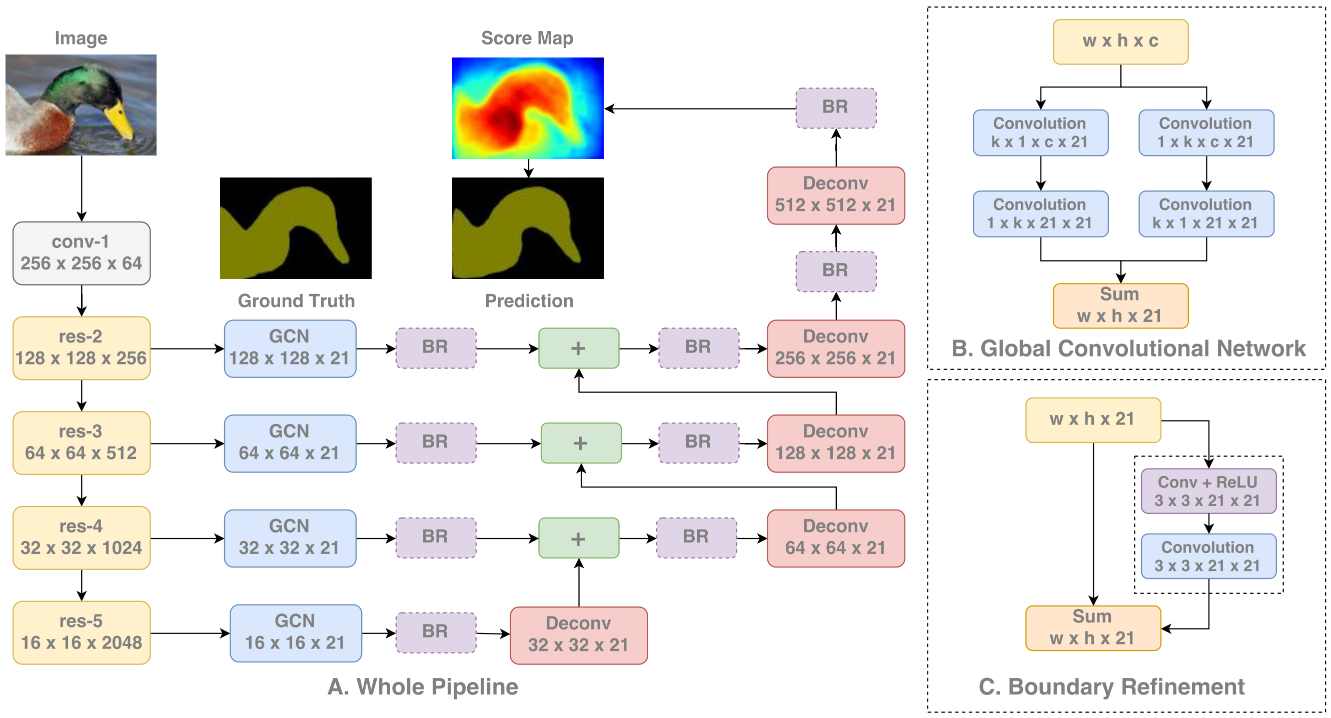

Figure 2. An overview of the whole pipeline in (A). The details of Global Convolutional Network (GCN) and Boundary Refinement (BR) block are illustrated in (B) and (C), respectively.

spatial location on the feature map to generate pixel-wise semantic labels.

We notice that current state-of-the-art semantic segmentation models [25, 6, 27] mainly follow the design principles for localization, however, which may be suboptimal for classification. As classifiers are connected locally rather than globally to the feature map, it is difficult for classifiers to handle different variations of transformations on the input. For example, consider the situations in Figure 3: a classifier is aligned to the center of an input object, so it is expected to give the semantic label for the object. At first, the *valid receptive filed* (VRF)[1] is large enough to hold the entire object. However, if the input object is resized to a large scale, then VRF can only cover a part of the object, which may be harmful for classification. It will be even worse if larger feature maps are used, because the gap between classification and localization becomes larger.

Based on above observation, we try to design a new architecture to overcome the drawbacks. First from the localization view, the structure must be fully-convolutional without any fully-connected layer or global pooling layer that used by many classification networks, since the latter will discard localization information. Second from the classification view, motivated by the densely-connected structure of classification models, the kernel size of the convolutional structure should be as large as possible. Specially, if the kernel size increases to the spatial size of feature map (named *global convolution*), the network will share the same benefit with pure classification models. Based on these two principles, we propose a novel *Global Convolutional Network* (GCN) in Figure 2 B. Instead of directly using larger kernel or global convolution, our GCN module employs a combination of $1 \times k \ + k \times 1$ and $k \times 1 \ + 1 \times k$ convolutions, which enables densely connections within a large $k \times k$ region in the feature map. Different from the separable kernels used by [32], we do not use any nonlinearity after convolution layers. Compared with the trivial $k \times k$ convolution, our GCN structure involves only $O(\frac{2}{k})$ computation cost and number of parameters, which is more practical for large kernel sizes.

3.2. Overall Framework

Our overall segmentation model are shown in Figure 2. We use pretrained ResNet [14] as the feature network and FCN4 [25, 35] as the segmentation framework. Multi-scale feature maps are extracted from different stages in the feature network. Global Convolutional Network structures are used to generate multi-scale semantic score maps for each

[1]Feature maps from modern networks such as GoolgeNet or ResNet usually have very large receptive field because of the deep architecture. However, studies [38] show that network tends to gather information mainly from a much smaller region in the receptive field, which is called *valid receptive field* (VRF) in this paper.

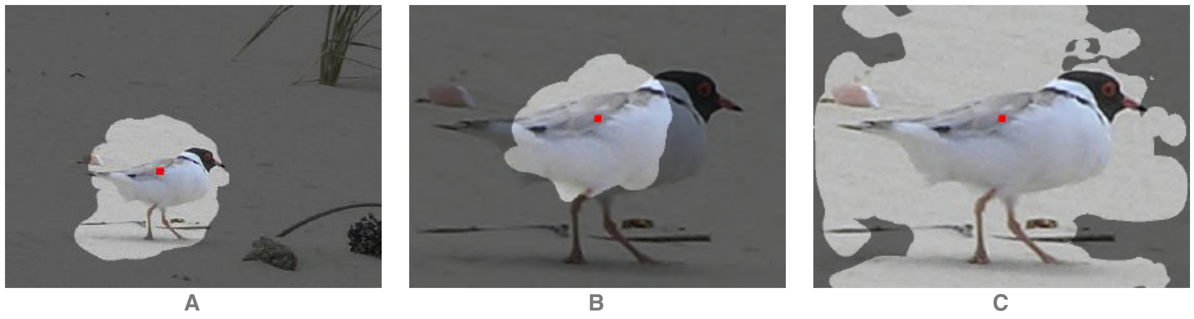

Figure 3. Visualization of *valid receptive field* (VRF) introduced by [38]. Regions on images show the VRF for the score map located at the center of the bird. For traditional segmentation model, even though the receptive field is as large as the input image, however, the VRF just covers the bird (A) and fails to hold the entire object if the input resized to a larger scale (B). As a comparison, our Global Convolution Network significantly enlarges the VRF (C).

class. Similar to [25, 35], score maps of lower resolution will be upsampled with a deconvolution layer, then added up with higher ones to generate new score maps. The final semantic score map will be generated after the last upsampling, which is used to output the prediction results.

In addition, we propose a Boundary Refinement (BR) block shown in Figure 2 C. Here, we models the boundary alignment as a residual structure. More specifically, we define $\tilde{S}$ as the refined score map: $\tilde{S} = S + \mathcal{R}(S)$, where S is the coarse score map and $\mathcal{R}(\cdot)$ is the residual branch. The details can be referred to Figure 2.

4. Experiment

We evaluate our approach on the standard benchmark PASCAL VOC 2012 [11, 10] and Cityscapes [8]. PASCAL VOC 2012 has 1464 images for training, 1449 images for validation and 1456 images for testing, which belongs to 20 object classes along with one background class. We also use the *Semantic Boundaries Dataset* [13] as auxiliary dataset, resulting in 10,582 images for training. We choose the state-of-the-art network ResNet 152 [14] (pretrained on ImageNet [28]) as our base model for fine tuning. During the training time, we use standard SGD [20] with batch size 1, momentum 0.99 and weight decay 0.0005 . Data augmentations like mean subtraction and horizontal flip are also applied in training. The performance is measured by standard mean intersection-over-union (IoU). All the experiments are running with Caffe [17] tool.

In the next subsections, first we will perform a series of ablation experiments to evaluate the effectiveness of our approach. Then we will report the full results on PASCAL VOC 2012 and Cityscapes.

4.1. Ablation Experiments

In this subsection, we will make apple-to-apple comparisons to evaluate our approaches proposed in Section 3. As mentioned above, we use PASCAL VOC 2012 validation set for the evaluation. For all succeeding experiments, we pad each input image into 512×512 so that the top-most feature map is 16×16.

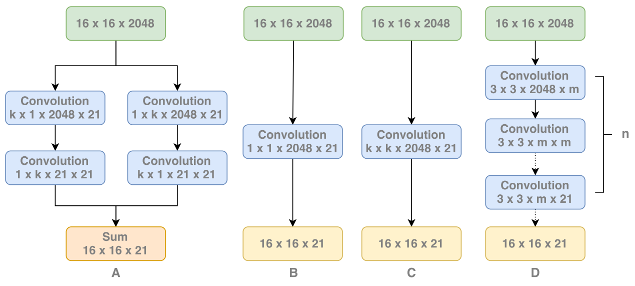
16 x 16 x 2048
Convolution k x 1 x 2048 x 21
Convolution 1 x k x 2048 x 21
Convolution 1 x k x 21 x 21
Convolution k x 1 x 21 x 21
Sum 16 x 16 x 21
A
16 x 16 x 2048
Convolution 1 x 1 x 2048 x 21
16 x 16 x 21
B
16 x 16 x 2048
Convolution k x k x 2048 x 21
16 x 16 x 21
C
16 x 16 x 2048
Convolution 3 x 3 x 2048 x m
Convolution 3 x 3 x m x m
Convolution 3 x 3 x m x 21
n
16 x 16 x 21
D

Figure 4. (A) Global Convolutional Network. (B) 1×1 convolution baseline. (C) $k \times k$ convolution. (D) stack of 3×3 convolutions.

4.1.1 Global Convolutional Network — Large Kernel Matters

In Section 3.1 we propose Global Convolutional Network (GCN) to enable densely connections between classifiers and features. The key idea of GCN is to use large kernels, whose size is controlled by the parameter k (see Figure 2 B). To verify this intuition, we enumerate different k and test the performance respectively. The overall network architecture is shown as in Figure 2 A except that Boundary Refinement block is not applied. For better comparison, a naive baseline is added just to replace GCN with a simple 1×1 convolution (shown in Figure 4 B). The results are presented in Table 1.

We try different kernel sizes ranging from 3 to 15. Note that only odd size are used just to avoid alignment error. In the case $k = 15$, which roughly equals to the feature map size (16×16), the structure becomes "really global convolu-

k	base	3	5	7	9	11	13	15
Score	69.0	70.1	71.1	72.8	73.4	73.7	74.0	74.5

Table 1. Experimental results on different k settings of Global Convolutional Network. The score is evaluated by standard mean IoU(%) on PASCAL VOC 2012 validation set.

tional". From the results, we can find that the performance consistently increases with the kernel size k. Especially, the "global convolutional" version ($k = 15$) surpasses the smallest one by a significant margin 5.5%. Results show that large kernel brings great benefit in our GCN structure, which is consistent with our analysis in Section 3.1.

Further Discussion: In the experiments in Table 1, since there are other differences between baseline and different versions of GCN, it seems not so confirmed to attribute the improvements to large kernels or GCN. For example, one may argue that the extra parameters brought by larger k lead to the performance gain. Or someone may think to use another simple structure instead of GCN to achieve large equivalent kernel size. So we will give more evidences for better understanding.

(1) *Are more parameters helpful?* In GCN, the number of parameters increases linearity with kernel size k, so one natural hypothesis is that the improvements in Table 1 are mainly brought by the increased number of parameters. To address this, we compare our GCN with the trivial large kernel design with a trivial $k \times k$ convolution shown in Figure 4 C. Results are shown in Table 2. From the results we can see that for any given kernel size, the trivial convolution design contains more parameters than GCN. However, the latter is consistently better than the former in performance respectively. It is also clear that for trivial convolution version,

k	3	5	7	9
Score (GCN)	70.1	71.1	72.8	73.4
Score (Conv)	69.8	70.4	69.6	68.8
# of Params (GCN)	260K	434K	608K	782K
# of Params (Conv)	387K	1075K	2107K	3484K

Table 2. Comparison experiments between Global Convolutional Network and the trivial implementation. The score is measured under standard mean IoU(%), and the 3rd and 4th rows show number of parameters of GCN and trivial Convolution after res-5.

larger kernel will result in better performance if $k \leq 5$, yet for $k \geq 7$ the performance drops. One hypothesis is that too many parameters make the training suffer from overfit, which weakens the benefits from larger kernels. However, in training we find trivial large kernels in fact make the network difficult to converge, while our GCN structure will not suffer from this drawback. Thus the actual reason still needs further study.

(2) *GCN vs. Stack of small convolutions.* Instead of GCN, another trivial approach to form a large kernel is to use stack of small kernel convolutions(for example, stack of 3×3 kernels in Figure 4 D), , which is very common in modern CNN architectures such as VGG-net [30]. For example, we can use two 3×3 convolutions to approximate a 5×5 kernel. In Table 3, we compare GCN with convolutional stacks under different equivalent kernel sizes. Different from [30], we do not apply nonlinearity within convolutional stacks so as to keep consistent with GCN structure. Results shows that GCN still outperforms trivial convolution stacks for any large kernel sizes.

k	3	5	7	9	11
Score (GCN)	70.1	71.1	72.8	73.4	73.7
Score (Stack)	69.8	71.8	71.3	69.5	67.5

Table 3. Comparison Experiments between Global Convolutional Network and the equivalent stack of small kernel convolutions. The score is measured under standard mean IoU(%). GCN is still better with large kernels ($k > 7$).

For large kernel size (e.g. $k = 7$) 3×3 convolutional stack will bring much more parameters than GCN, which may have side effects on the results. So we try to reduce the number of intermediate feature maps for convolutional stack and make further comparison. Results are listed in Table 4. It is clear that its performance suffers from degradation with fewer parameters. In conclusion, GCN is a better structure compared with trivial convolutional stacks.

m (Stack)	2048	1024	210	2048 (GCN)
Score	71.3	70.4	68.8	72.8
# of Params	75885K	28505K	4307K	608K

Table 4. Experimental results on the channels of stacking of small kernel convolutions. The score is measured under standard mean IoU. GCN outperforms the convolutional stack design with less parameters.

(3) *How GCN contributes to the segmentation results?* In Section 3.1, we claim that GCN improves the classification capability of segmentation model by introducing densely connections to the feature map, which is helpful to handle large variations of transformations. Based on this, we can infer that pixels lying in the center of large objects may benefit more from GCN because it is very close to "pure" classification problem. As for the boundary pixels of objects, however, the performance is mainly affected by the localization ability.

To verify our inference, we divide the segmentation score map into two parts: a) boundary region, whose pixels locate close to objects' boundary (distance ≤ 7), and b) internal region as other pixels. We evaluate our segmentation model (GCN with $k = 15$) in both regions. Results are shown in Table 5. We find that our GCN model mainly improves the accuracy in internal region while the effect in boundary region is minor, which strongly supports our argument. Furthermore, in Table 5 we also evaluate the boundary refinement (BF) block referred in Section 3.2. In contrary to GCN structure, BF mainly improves the accuracy in boundary region, which also confirms its effectiveness.

Model	Boundary (acc.)	Internal (acc.)	Overall (IoU)
Baseline	71.3	93.9	69.0
GCN	71.5	95.0	74.5
GCN + BR	73.4	95.1	74.7

Table 5. Experimental results on *Residual Boundary Alignment*. The Boundary and Internal columns are measured by the per-pixel accuracy while the 3rd column is measured by standard mean IoU.

4.1.2 Global Convolutional Network for Pretrained Model

In the above subsection our segmentation models are finetuned from ResNet-152 network. Since large kernel plays a critical role in segmentation tasks, it is nature to apply the idea of GCN also on the pretrained model. Thus we propose a new ResNet-GCN structure, as shown in Figure 5. We remove the first two layers in the original bottleneck structure used by ResNet, and replace them with a GCN module. In order to keep consistent with the original, we also apply Batch Normalization [15] and ReLU after each of the convolution layers.

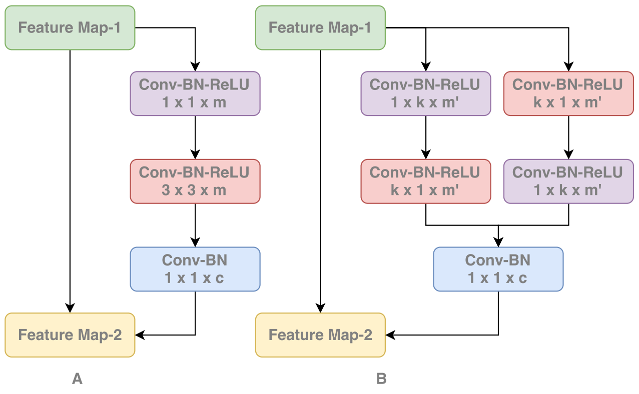

Figure 5. A: the bottleneck module in original ResNet. B: our *Global Convolutional Network* in ResNet-GCN.

We compare our ResNet-GCN structure with the original ResNet model. For fair comparison, sizes for ResNet-GCN are carefully selected so that both network have similar computation cost and number of parameters. More details are provided in the appendix. We first pretrain ResNet-GCN on ImageNet 2015 [28] and fine tune on PASCAL VOC 2012 segmentation dataset. Results are shown in Table 6. Note that we take ResNet50 model (with or without GCN) for comparison because the training of large ResNet152 is very costly. From the results we can see that our GCN-based ResNet is slightly poorer than original ResNet as an ImageNet classification model. However, after finetuning on segmentation dataset ResNet-GCN model outperforms original ResNet significantly by 5.5%. With the application of GCN and boundary refinement, the gain of GCN-based pretrained model becomes minor but still prevails. We can safely conclude that GCN mainly helps to improve segmentation performance, no matter in pretrained model or segmentation-specific structures.

Pretrained Model	ResNet50	ResNet50-GCN
ImageNet cls err (%)	7.7	7.9
Seg. Score (Baseline)	65.7	71.2
Seg. Score (GCN + BR)	72.3	**72.5**

Table 6. Experimental results on ResNet50 and ResNet50-GCN. Top-5 error of 224×224 center-crop on 256×256 image is used in ImageNet classification error. The segmentation score is measured under standard mean IoU.

4.2. PASCAL VOC 2012

In this section we discuss our practice on PASCAL VOC 2012 dataset. Following [6, 37, 24, 7], we employ the Microsoft COCO dataset [22] to pre-train our model. COCO has 80 classes and here we only retain the images including the same 20 classes in PASCAL VOC 2012. The training phase is split into three stages: (1) In *Stage-1*, we mix up all the images from COCO, SBD and standard PASCAL VOC 2012, resulting in 109,892 images for training. (2) During the *Stage-2*, we use the SBD and standard PASCAL VOC 2012 images, the same as Section 4.1. (3) For *Stage-3*, we only use the standard PASCAL VOC 2012 dataset. The input image is padded to 640×640 in Stage-1 and 512×512 for Stage-2 and Stage-3. The evaluation on validation set is shown in Table 7.

Phase	Baseline	GCN	GCN + BR
Stage-1(%)	69.6	74.1	75.0
Stage-2(%)	72.4	77.6	78.6
Stage-3(%)	74.0	78.7	80.3
Stage-3-MS(%)			80.4
Stage-3-MS-CRF(%)			**81.0**

Table 7. Experimental results on PASCAL VOC 2012 validation set. The results are evaluated by standard mean IoU.

Our GCN + BR model clearly prevails, meanwhile the post-processing multi-scale and denseCRF [18] also bring benefits. Some visual comparisons are given in Figure 6. We also submit our best model to the on-line evaluation server, obtaining **82.2%** on PASCAL VOC 2012 test set,

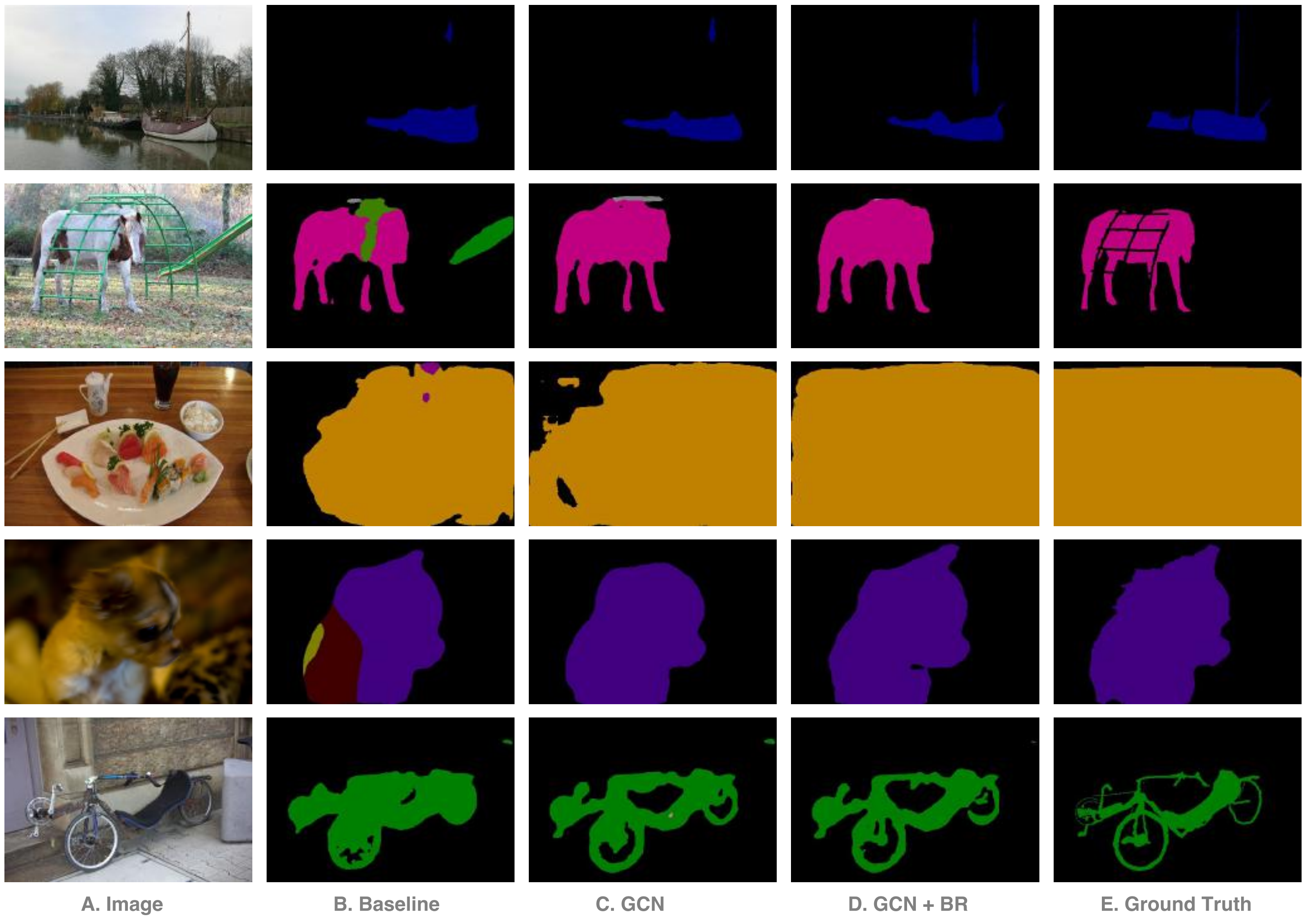

Figure 6. Examples of semantic segmentation results on PASCAL VOC 2012. For every row we list input image (A), 1×1 convolution baseline (B), Global Convolutional Network (GCN) (C), Global Convolutional Network plus Boundary Refinement (GCN + BR) (D), and Ground truth (E).

as shown in Table 8. Our work has outperformed all the previous **state-of-the-arts**.

Method	mean-IoU(%)
FCN-8s-heavy [29]	67.2
TTI_zoomout_v2 [26]	69.6
MSRA_BoxSup [9]	71.0
DeepLab-MSc-CRF-LargeFOV [6]	71.6
Oxford_TVG_CRF_RNN_COCO [37]	74.7
CUHK_DPN_COCO [24]	77.5
Oxford_TVG_HO_CRF [2]	77.9
CASIA_IVA_OASeg [33]	78.3
Adelaide_VeryDeep_FCN_VOC [34]	79.1
LRR_4x_ResNet_COCO [12]	79.3
Deeplabv2-CRF [7]	79.7
CentraleSupelec Deep G-CRF[5]	80.2
Our approach	**82.2**

Table 8. Experimental results on PASCAL VOC 2012 test set.

4.3. Cityscapes

Cityscapes [8] is a dataset collected for semantic segmentation on urban street scenes. It contains 24998 images from 50 cities with different conditions, which belongs to 30 classes without background class. For some reasons, only 19 out of 30 classes are evaluated on leaderboard. The images are split into two set according to their labeling quality. 5,000 of them are fine annotated while the other 19,998 are coarse annotated. The 5,000 fine annotated images are further grouped into 2975 training images, 500 validation images and 1525 testing images.

The images in Cityscapes have a fixed size of 1024×2048, which is too large to our network architecture. Therefore we randomly crop the images into 800×800 during training phase. We also increase k of GCN from 15 to 25 as the final feature map is 25×25. The training phase is split into two stages: (1) In *Stage-1*, we mix up the coarse annotated images and the training set, resulting in 22,973 images. (2) For *Stage-2*, we only finetune the network on training set. During the evaluation phase, we split the images into four 1024×1024 crops and fuse their score maps. The results are given in Table 9.

Phase	GCN + BR
Stage-1(%)	73.0
Stage-2(%)	76.9
Stage-2-MS(%)	77.2
Stage-2-MS-CRF(%)	**77.4**

Table 9. Experimental results on Cityscapes validation set. The standard mean IoU is used here.

We submit our best model to the on-line evaluation server, obtaining **76.9%** on Cityscapes test set as shown in Table 10. Once again, we outperforms all the previous publications and reaches the new **state-of-art**.

Method	mean-IoU(%)
FCN 8s [29]	65.3
DPN [24]	59.1
CRFasRNN [37]	62.5
Scale invariant CNN + CRF [19]	66.3
Dilation10 [36]	67.1
DeepLabv2-CRF [7]	70.4
Adelaide_context [21]	71.6
LRR-4x [12]	71.8
Our approach	**76.9**

Table 10. Experimental results on Cityscapes test set.

5. Conclusion

According to our analysis on classification and segmentation, we find that large kernels is crucial to relieve the contradiction between classification and localization. Following the principle of large-size kernels, we propose the Global Convolutional Network. The ablation experiments show that our proposed structures meet a good trade-off between valid receptive field and the number of parameters, while achieves good performance. To further refine the object boundaries, we present a novel Boundary Refinement block. Qualitatively, our Global Convolutional Network mainly improve the internal regions while Boundary Refinement increase performance near boundaries. Our best model achieves state-of-the-art on two public benchmarks: PASCAL VOC 2012 (82.2%) and Cityscapes (76.9%).

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010. 2

[2] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr. Higher order conditional random fields in deep neural networks. In *European Conference on Computer Vision*, pages 524–540. Springer, 2016. 7

[3] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015. 2

[4] J. T. Barron and B. Poole. The fast bilateral solver. *ECCV*, 2016. 2

[5] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. *arXiv preprint arXiv:1603.08358*, 2016. 7

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1, 2, 3, 6, 7

[7] 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. *arXiv preprint arXiv:1606.00915*, 2016. 2, 6, 7, 8

[8] 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. *arXiv preprint arXiv:1604.01685*, 2016. 4, 7

[9] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643, 2015. 7

[10] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 4

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[12] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, pages 519–534. Springer, 2016. 2, 7, 8

[13] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998. IEEE, 2011. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 3, 4

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 448–456, 2015. 6

[16] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral

neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 4

[18] V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2011. 2, 6

[19] I. Krešo, D. Čaušević, J. Krapac, and S. Šegvić. Convolutional scale invariance for semantic segmentation. In *German Conference on Pattern Recognition*, pages 64–75. Springer, 2016. 8

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2, 4

[21] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 8

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 6

[23] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 2

[24] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1385, 2015. 2, 6, 7, 8

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1, 2, 3, 4

[26] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015. 2, 7

[27] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015. 2, 3

[28] 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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4, 6

[29] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. 2016. 2, 7, 8

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 5

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1, 2

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015. 2, 3

[33] Y. Wang, J. Liu, Y. Li, J. Yan, and H. Lu. Objectness-aware semantic segmentation. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 307–311. ACM, 2016. 7

[34] Z. Wu, C. Shen, and A. v. d. Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *arXiv preprint arXiv:1604.04339*, 2016. 7

[35] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015. 3, 4

[36] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2, 8

[37] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015. 2, 6, 7, 8

[38] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 3, 4

Appendix.A ResNet50 and ResNet50-GCN

Component	Output Size	ResNet50	ResNet50-GCN
conv1	112×112	7×7, 64, stride 2	
res-2	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
res-3	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
res-4	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} (1 \times 5, 85) & (5 \times 1, 85) \\ (5 \times 1, 85) & (1 \times 5, 85) \\ \multicolumn{2}{c}{1 \times 1, 1024} \end{bmatrix} \times 6$
res-5	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} (1 \times 7, 128) & (7 \times 1, 128) \\ (7 \times 1, 128) & (1 \times 7, 128) \\ \multicolumn{2}{c}{1 \times 1, 2048} \end{bmatrix} \times 3$
ImageNet Classifier	1×1	global average pool, 1000-d fc, softmax	
MFlops (Conv)		3700	3700

Table 11. Architectures for ResNet50 and ResNet50-GCN, discussed in Section 4.1.2. The bottleneck and GCN blocks are shown in brackets (referred to Figure 5). Downsampling is performed between every components with stride 2 convolution. Output Size (2nd column) is measured with standard ImangeNet 224×224 images. The computational complexity of convolutions is shown in last row.

Appendix.B Examples of semantic segmentation results on Cityscapes.

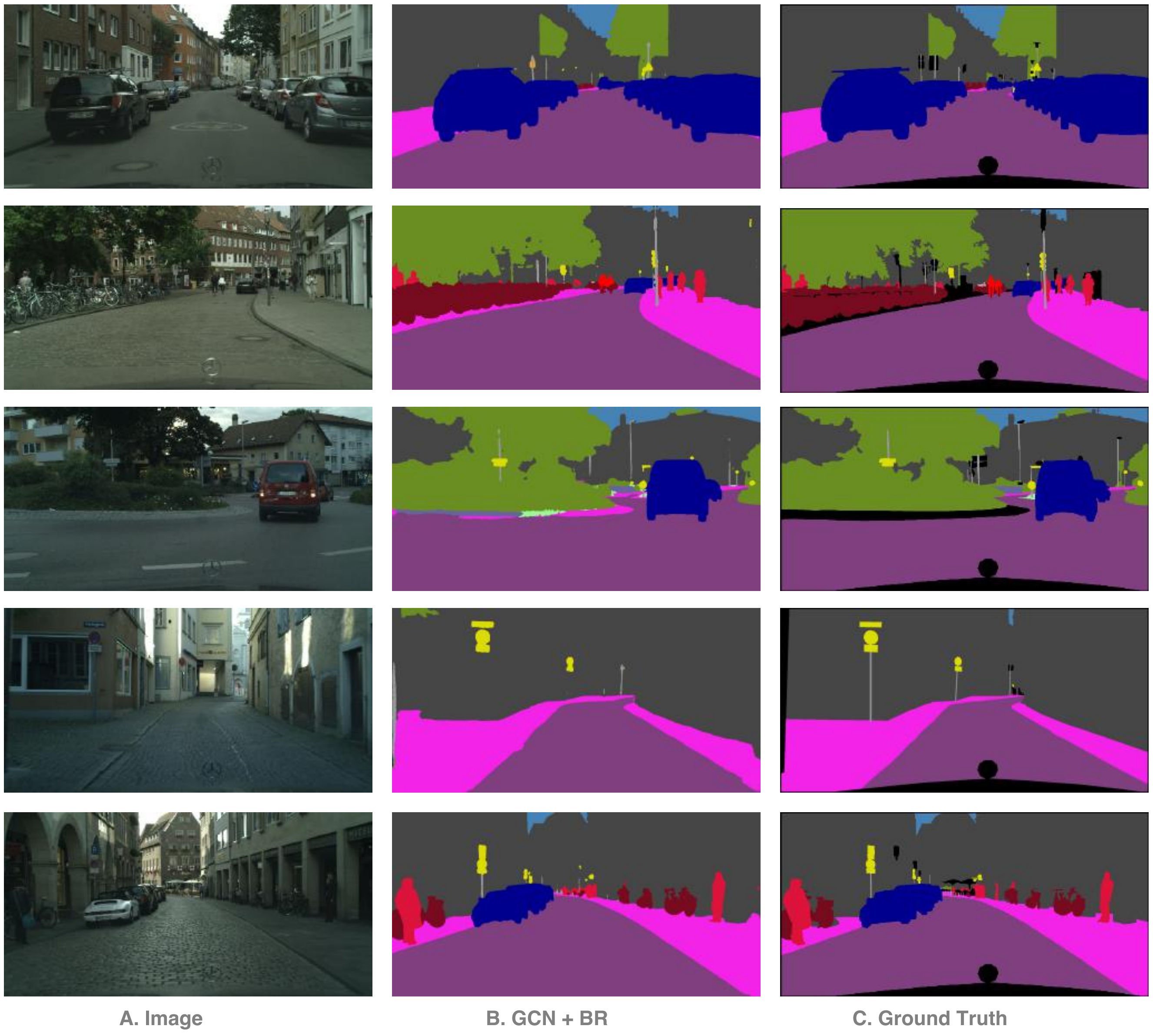

Figure 7. Examples of semantic segmentation results on Cityscapes. For every row we list input Image (A), Global Convolutional Network plus Boundary Refinement (GCN + BR) (B) and Ground Truth (C).