ICNet for Real-Time Semantic Segmentation on High-Resolution Images

Hengshuang Zhao[1](✉), Xiaojuan Qi[1], Xiaoyong Shen[2], Jianping Shi[3], and Jiaya Jia[1,2]

[1] The Chinese University of Hong Kong, Shatin, Hong Kong
{hszhao,xjqi,leojia}@cse.cuhk.edu.hk
[2] Tencent Youtu Lab, Shenzhen, China
dylanshen@tencent.com
[3] SenseTime Research, Beijing, China
shijianping@sensetime.com

Abstract. We focus on the challenging task of real-time semantic segmentation in this paper. It finds many practical applications and yet is with fundamental difficulty of reducing a large portion of computation for pixel-wise label inference. We propose an image cascade network (ICNet) that incorporates multi-resolution branches under proper label guidance to address this challenge. We provide in-depth analysis of our framework and introduce the cascade feature fusion unit to quickly achieve high-quality segmentation. Our system yields real-time inference on a single GPU card with decent quality results evaluated on challenging datasets like Cityscapes, CamVid and COCO-Stuff.

Keywords: Real-time · High-resolution · Semantic segmentation

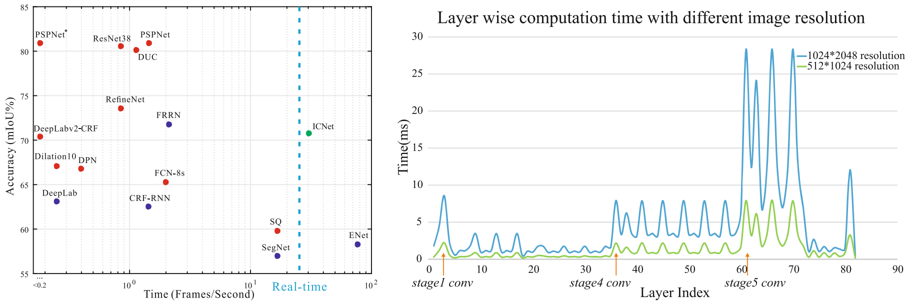
1 Introduction

Semantic image segmentation is a fundamental task in computer vision. It predicts dense labels for all pixels in the image, and is regarded as a very important task that can help deep understanding of scene, objects, and human. Development of recent deep *convolutional neural networks* (CNNs) makes remarkable progress on semantic segmentation [1–6]. The effectiveness of these networks largely depends on the sophisticated model design regarding depth and width, which has to involve many operations and parameters.

CNN-based semantic segmentation mainly exploits *fully convolutional networks* (FCNs). It is common wisdom now that increase of result accuracy almost means more operations, especially for pixel-level prediction tasks like semantic segmentation. To illustrate it, we show in Fig. 1(a) the accuracy and inference time of different frameworks on Cityscapes [7] dataset.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01219-9_25) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11207, pp. 418–434, 2018.
https://doi.org/10.1007/978-3-030-01219-9_25



(a) Inference speed and mIoU

(b) Time in each layer of PSPNet50

Fig. 1. (a): Inference speed and mIoU performance on Cityscapes [7] test set. Methods involved are PSPNet [5], ResNet38 [6], DUC [10], RefineNet [11], FRRN [12], DeepLabv2-CRF [13], Dilation10 [14], DPN [15], FCN-8s [1], DeepLab [2], CRF-RNN [16], SQ [9], ENet [8], SegNet [3], and our ICNet. (b): Time spent on PSPNet50 with dilation 8 for two input images. Roughly running time is proportional to the pixel number and kernel number. (Blue ones are tested with downsampled images. Inference speed is reported with single network forward while accuracy of several mIoU aimed approaches (like PSPNet*) may contain testing tricks like multi-scale and flipping, resulting much more time. See supplementary material for detailed information.) (Color figure online)

Status of Fast Semantic Segmentation. Contrary to the extraordinary development of high-quality semantic segmentation, research along the line to make semantic segmentation run *fast* while not sacrificing too much quality is left behind. We note actually this line of work is similarly important since it can inspire or enable many practical tasks in, for example, automatic driving, robotic interaction, online video processing, and even mobile computing where running time becomes a critical factor to evaluate system performance.

Our experiments show that high-accuracy methods of ResNet38 [6] and PSPNet [5] take around 1 second to predict a 1024 × 2048 high-resolution image on one Nvidia TitanX GPU card during testing. These methods fall into the area illustrated in Fig. 1(a) with high accuracy and low speed. Recent fast semantic segmentation methods of ENet [8] and SQ [9], contrarily, take quite different positions in the plot. The speed is much accelerated; but accuracy drops, where the final mIoUs are lower than 60%. These methods are located in the lower right phase in the figure.

Our Focus and Contributions. In this paper, we focus on building a practically fast semantic segmentation system with decent prediction accuracy. Our method is the first in its kind to locate in the top-right area shown in Fig. 1(a) and is one of the only two available real-time approaches. It achieves decent trade-off between efficiency and accuracy.

Different from previous architectures, we make comprehensive consideration on the two factors of speed and accuracy that are seemingly contracting. We first make in-depth analysis of time budget in semantic segmentation frameworks and conduct extensive experiments to demonstrate insufficiency of intuitive speedup strategies. This motivates development of *image cascade network* (ICNet), a high efficiency segmentation system with decent quality. It exploits efficiency of processing low-resolution images and high inference quality of high-resolution ones. The idea is to let low-resolution images go through the full semantic perception network first for a coarse prediction map. Then cascade feature fusion unit and cascade label guidance strategy are proposed to integrate medium and high resolution features, which refine the coarse semantic map gradually. We make all our code and models publicly available[1]. Our main contributions and performance statistics are the following.

- We develop a novel and unique image cascade network for real-time semantic segmentation, it utilizes semantic information in low resolution along with details from high-resolution images efficiently.
- The developed cascade feature fusion unit together with cascade label guidance can recover and refine segmentation prediction progressively with a low computation cost.
- Our ICNet achieves 5× speedup of inference time, and reduces memory consumption by 5× times. It can run at high resolution 1024×2048 in speed of 30 fps while accomplishing high-quality results. It yields real-time inference on various datasets including Cityscapes [7], CamVid [17] and COCO-Stuff [18].

2 Related Work

Traditional semantic segmentation methods [19] adopt handcrafted feature to learn the representation. Recently, CNN based methods largely improve the performance.

High Quality Semantic Segmentation. FCN [1] is the pioneer work to replace the last fully-connected layers in classification with convolution layers. DeepLab [2,13] and [14] used dilated convolution to enlarge the receptive field for dense labeling. Encoder-decoder structures [3,4] can combine the high-level semantic information from later layers with the spatial information from earlier ones. Multi-scale feature ensembles are also used in [20–22]. In [2,15,16], conditional random fields (CRF) or Markov random fields (MRF) were used to model spatial relationship. Zhao *et al.* [5] used pyramid pooling to aggregate global and local context information. Wu *et al.* [6] adopted a wider network to boost performance. In [11], a multi-path refinement network combined multi-scale image features. These methods are effective, but preclude real-time inference.

High Efficiency Semantic Segmentation. In object detection, speed became one important factor in system design [23,24]. Recent Yolo [25,26] and SSD [27] are representative solutions. In contrast, high speed inference in semantic segmentation is under-explored. ENet [8] and [28] are lightweight networks. These methods greatly raise efficiency with notably sacrificed accuracy.

[1] https://github.com/hszhao/ICNet.

Video Semantic Segmentation. Videos contain redundant information in frames, which can be utilized to reduce computation. Recent Clockwork [29] reuses feature maps given stable video input. Deep feature flow [30] is based on a small-scale optical flow network to propagate features from key frames to others. FSO [31] performs structured prediction with dense CRF applied on optimized features to get temporal consistent predictions. NetWarp [32] utilizes optical flow of adjacent frames to warp internal features across time space in video sequences. We note when a good-accuracy fast image semantic-segmentation framework comes into existence, video segmentation will also be benefited.

3 Image Cascade Network

We start by analyzing computation time budget of different components on the high performance segmentation framework PSPNet [5] with experimental statistics. Then we introduce the *image cascade network* (ICNet) as illustrated in Fig. 2, along with the cascade feature fusion unit and cascade label guidance, for fast semantic segmentation.

3.1 Speed Analysis

In convolution, the transformation function $\varPhi$ is applied to input feature map $V \in \mathbb{R}^{c \times h \times w}$ to obtain the output map $U \in \mathbb{R}^{c' \times h' \times w'}$, where c, h and w denote features channel, height and width respectively. The transformation operation $\varPhi : V \rightarrow U$ is achieved by applying c' number of 3D kernels $K \in \mathbb{R}^{c \times k \times k}$ where $k \times k$ (e.g, 3×3) is kernel spatial size. Thus the total number of operations $O(\varPhi)$ in convolution layer is $c'ck^2h'w'$. The spatial size of the output map h' and w' are highly related to the input, controlled by parameter stride s as $h' = h/s, w' = w/s$, making

$$O(\varPhi) \approx c'ck^2hw/s^2. \tag{1}$$

The computation complexity is associated with feature map resolution (e.g., h, w, s), number of kernels and network width (e.g., c, c'). Figure 1(b) shows the time cost of two resolution images in PSPNet50. Blue curve corresponds to high-resolution input with size 1024×2048 and green curve is for image with resolution 512×1024. Computation increases squarely regarding image resolution. For either curve, feature maps in stage4 and stage5 are with the same spatial resolution, i.e., 1/8 of the original input; but the computation in stage5 is four times heavier than that in stage4. It is because convolutional layers in stage5 double the number of kernels c together with input channel c'.

3.2 Network Architecture

According to above time budget analysis, we adopt intuitive speedup strategies in experiments to be detailed in Sect. 5, including downsampling input, shrinking feature maps and conducting model compression. The corresponding results

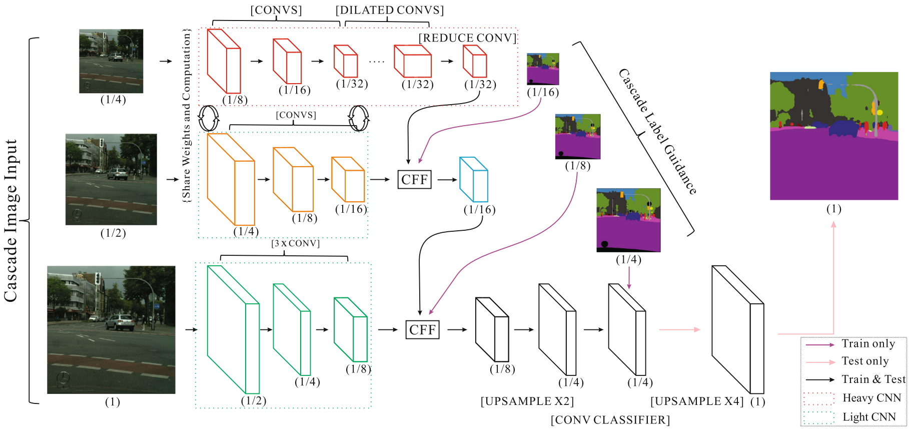


Fig. 2. Network architecture of ICNet. 'CFF' stands for cascade feature fusion detailed in Sect. 3.3. Numbers in parentheses are feature map size ratios to the full-resolution input. Operations are highlighted in brackets. The final ×4 upsampling in the bottom branch is only used during testing.

show that it is very difficult to keep a good balance between inference accuracy and speed. The intuitive strategies are effective to reduce running time, while they yield very coarse prediction maps. Directly feeding high-resolution images into a network is unbearable in computation.

Our proposed system *image cascade network* (ICNet) does not simply choose either way. Instead it takes cascade image inputs (i.e., low-, medium- and high resolution images), adopts cascade feature fusion unit (Sect. 3.3) and is trained with cascade label guidance (Sect. 3.4). The new architecture is illustrated in Fig. 2. The input image with full resolution (e.g., 1024×2048 in Cityscapes [7]) is downsampled by factors of 2 and 4, forming cascade input to medium- and high-resolution branches.

Segmenting the high-resolution input with classical frameworks like FCN directly is time consuming. To overcome this shortcoming, we get semantic extraction using low-resolution input as shown in top branch of Fig. 2. A 1/4 sized image is fed into PSPNet with downsampling rate 8, resulting in a 1/32-resolution feature map. To get high quality segmentation, medium and high resolution branches (middle and bottom parts in Fig. 2) help recover and refine the coarse prediction. Though some details are missing and blurry boundaries are generated in the top branch, it already harvests most semantic parts. Thus we can safely limit the number of parameters in both middle and bottom branches. Light weighted CNNs (green dotted box) are adopted in higher resolution branches; different-branch output feature maps are fused by cascade-feature-fusion unit (Sect. 3.3) and trained with cascade label guidance (Sect. 3.4).

Although the top branch is based on a full segmentation backbone, the input resolution is low, resulting in limited computation. Even for PSPNet with 50+ layers, inference time and memory are 18 ms and 0.6 GB for the large images in Cityscapes. Because weights and computation (in 17 layers) can be shared between low- and medium-branches, only 6ms is spent to construct the fusion map. Bottom branch has even less layers. Although the resolution is high, inference only takes 9 ms. Details of the architecture are presented in the supplementary file. With all these three branches, our ICNet becomes a very efficient and memory friendly architecture that can achieve good-quality segmentation.

3.3 Cascade Feature Fusion

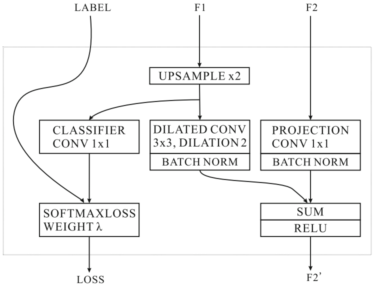


Fig. 3. Cascade feature fusion.

To combine cascade features from different-resolution inputs, we propose a cascade feature fusion (CFF) unit as shown in Fig. 3. The input to this unit contains three components: two feature maps F_1 and F_2 with sizes $C_1 \times H_1 \times W_1$ and $C_2 \times H_2 \times W_2$ respectively, and a ground-truth label with resolution $1 \times H_2 \times W_2$. F_2 is with doubled spatial size of F_1.

We first apply upsampling rate 2 on F_1 through bilinear interpolation, yielding the same spatial size as F_2. Then a dilated convolution layer with kernel size $C_3 \times 3 \times 3$ and dilation 2 is applied to refine the upsampled features. The resulting feature is with size $C_3 \times H_2 \times W_2$. This dilated convolution combines feature information from several originally neighboring pixels. Compared with deconvolution, upsampling followed by dilated convolution only needs small kernels, to harvest the same receptive field. To keep the same receptive field, deconvolution needs larger kernel sizes than upsampling with dilated convolution (i.e., 7×7 vs. 3×3), which causes more computation.

For feature F_2, a projection convolution with kernel size $C_3 \times 1 \times 1$ is utilized to project F_2 so that it has the same number of channels as the output of F_1. Then two batch normalization layers are used to normalize these two processed features as shown in Fig. 3. Followed by an element-wise 'sum' layer and a 'ReLU' layer, we obtain the fused feature F_2' as $C_3 \times H_2 \times W_2$. To enhance learning of F_1, we use an auxiliary label guidance on the upsampled feature of F_1.

3.4 Cascade Label Guidance

To enhance the learning procedure in each branch, we adopt a cascade label guidance strategy. It utilizes different-scale (e.g., 1/16, 1/8, and 1/4) ground-truth labels to guide the learning stage of low, medium and high resolution input. Given $\mathcal{T}$ branches (i.e., $\mathcal{T} = 3$) and $\mathcal{N}$ categories. In branch t, the predicted feature map $\mathcal{F}^t$ has spatial size $\mathcal{Y}_t \times \mathcal{X}_t$. The value at position (n, y, x) is $\mathcal{F}^t_{n,y,x}$. The corresponding ground truth label for 2D position (y, x) is $\hat{n}$. To train ICNet, we append weighted softmax cross entropy loss in each branch with related loss weight λ_t. Thus we minimize the loss function $\mathcal{L}$ defined as

$$\mathcal{L} = -\sum_{t=1}^{\mathcal{T}} \lambda_t \frac{1}{\mathcal{Y}_t \mathcal{X}_t} \sum_{y=1}^{\mathcal{Y}_t} \sum_{x=1}^{\mathcal{X}_t} \log \frac{e^{\mathcal{F}^t_{\hat{n},y,x}}}{\sum_{n=1}^{\mathcal{N}} e^{\mathcal{F}^t_{n,y,x}}}. \tag{2}$$

In the testing phase, the low and medium guidance operations are simply abandoned, where only high-resolution branch is retained. This strategy makes gradient optimization smoother for easy training. With more powerful learning ability in each branch, the final prediction map is not dominated by any single branch.

4 Structure Comparison and Analysis

Now we illustrate the difference of ICNet from existing cascade architectures for semantic segmentation. Typical structures in previous semantic segmentation systems are illustrated in Fig. 4. Our proposed ICNet (Fig. 4(d)) is by nature different from others. Previous frameworks are all with relatively intensive computation given the high-resolution input. While in our cascade structure, only the lowest-resolution input is fed into the heavy CNN with much reduced computation to get the coarse semantic prediction. The higher-res inputs are designed to recover and refine the prediction progressively regarding blurred boundaries and missing details. Thus they are processed by light-weighted CNNs. Newly introduced cascade-feature-fusion unit and cascade label guidance strategy integrate medium and high resolution features to refine the coarse semantic map gradually. In this special design, ICNet achieves high-efficiency inference with reasonable-quality segmentation results.

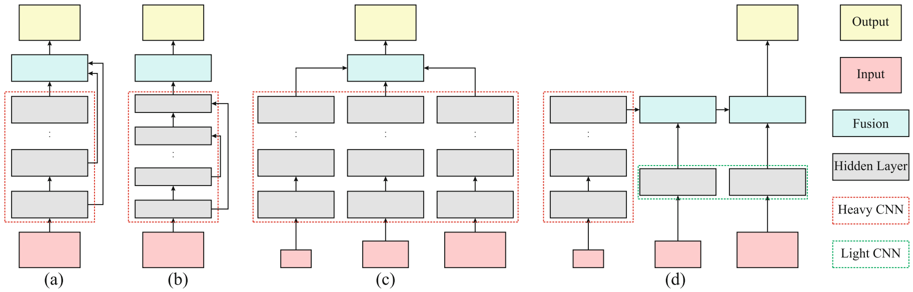


Fig. 4. Comparison of semantic segmentation frameworks. (a) Intermediate skip connection used by FCN [1] and Hypercolumns [21]. (b) Encoder-decoder structure incorporated in SegNet [3], DeconvNet [4], UNet [33], ENet [8], and step-wise reconstruction & refinement from LRR [34] and RefineNet [11]. (c) Multi-scale prediction ensemble adopted by DeepLab-MSC [2] and PSPNet-MSC [5]. (d) Our ICNet architecture.

5 Experimental Evaluation

Our method is effective for high resolution images. We evaluate the architecture on three challenging datasets, including urban-scene understanding dataset Cityscapes [7] with image resolution 1024×2048, CamVid [17] with image resolution 720×960 and stuff understanding dataset COCO-Stuff [18] with image resolution up to 640×640. There is a notable difference between COCO-Stuff and object/scene segmentation datasets of VOC2012 [35] and ADE20K [36]. In the latter two sets, most images are of low resolution (e.g., 300×500), which can already be processed quickly. While in COCO-Stuff, most images are larger, making it more difficult to achieve real-time performance.

In the following, we first show intuitive speedup strategies and their drawbacks, then reveal our improvement with quantitative and visual analysis.

5.1 Implementation Details

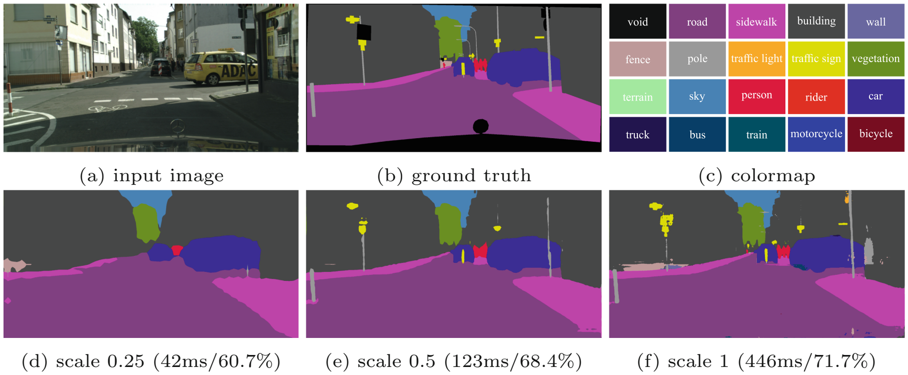
We conduct experiments based on platform Caffe [37]. All experiments are on a workstation with Maxwell TitanX GPU cards under CUDA 7.5 and CUDNN V5. Our testing uses only one card. To measure the forward inference time, we use the time measure tool 'Caffe time' and set the repeating iteration number to 100 to eliminate accidental errors during testing. All the parameters in batch normalization layers are merged into the neighboring front convolution layers.

For the training hyper-parameters, the mini-batch size is set to 16. The base learning rate is 0.01 and the 'poly' learning rate policy is adopted with power 0.9, together with the maximum iteration number set to 30K for Cityscapes, 10K for CamVid and 30K for COCO-Stuff. Momentum is 0.9 and weight decay is 0.0001. Data augmentation contains random mirror and rand resizing between 0.5 and 2. The auxiliary loss weights are empirically set to 0.4 for λ_1 and λ_2, 1 for λ_3 in Eq. 2, as adopted in [5]. For evaluation, both *mean of class-wise intersection over union* (mIoU) and *network forward time* (Time) are used.

5.2 Cityscapes

We first apply our framework to the recent urban scene understanding dataset Cityscapes [7]. This dataset contains high-resolution 1024×2048 images, which make it a big challenge for fast semantic segmentation. It contains 5,000 finely annotated images split into training, validation and testing sets with 2,975, 500, and 1,525 images respectively. The dense annotation contains 30 common classes of road, person, car, etc. 19 of them are used in training and testing.

Intuitive Speedup. According to the time complexity shown in Eq. (1), we do intuitive speedup in three aspects, namely downsampling input, downsampling feature, and model compression.



(a) input image (b) ground truth (c) colormap

(d) scale 0.25 (42ms/60.7%) (e) scale 0.5 (123ms/68.4%) (f) scale 1 (446ms/71.7%)

Fig. 5. Downsampling input: prediction of PSPNet50 on the validation set of Cityscapes. Values in the parentheses are the inference time and mIoU.

Table 1. Left: Downsampling feature with factors 8, 16 and 32. Right: model compression with kernel keeping rates 1, 0.5 and 0.25.

Downsample Size	8	16	32
mIoU (%)	71.7	70.2	67.1
Time (ms)	446	177	131

Kernel Keeping Rates	1	0.5	0.25
mIoU (%)	71.7	67.9	59.4
Time (ms)	446	170	72

Downsampling Input. Image resolution is the most critical factor that affects running speed as analyzed in Sect. 3.1. A simple approach is to use the small-resolution image as input. We test downsampling the image with ratios 1/2 and 1/4, and feeding the resulting images into PSPNet50. We directly upsample prediction results to the original size. This approach empirically has several drawbacks as illustrated in Fig. 5. With scaling ratio 0.25, although the inference time is reduced by a large margin, the prediction map is very coarse, missing many small but important details compared to the higher resolution prediction. With scaling ratio 0.5, the prediction recovers more information compared to the 0.25 case. Unfortunately, the person and traffic light far from the camera are still missing and object boundaries are blurred. To make things worse, the running time is still too long for a real-time system.

Downsampling Feature. Besides directly downsampling the input image, another simple choice is to scale down the feature map by a large ratio in the inference process. FCN [1] downsampled it for 32 times and DeepLab [2] did that for 8 times. We test PSPNet50 with downsampling ratios of 1:8, 1:16 and 1:32 and show results in the left of Table 1. A smaller feature map can yield faster inference at the cost of sacrificing prediction accuracy. The lost information is mostly detail contained in low-level layers. Also, even with the smallest resulting feature map under ratio 1:32, the system still takes 131ms in inference.

Table 2. Performance of ICNet with different branches on validation set of Citysapes. The baseline method is PSPNet50 compressed to a half. 'sub4', 'sub24' and 'sub124' represent predictions in low-, medium-, and high-resolution branches respectively.

Items	Baseline	sub4	sub24	sub124
mIoU (%)	67.9	59.6	66.5	**67.7**
Time (ms)	170	18	25	33
Frame (fps)	5.9	55.6	40	**30.3**
Speedup	1×	9.4×	6.8×	**5.2**×
Memory (GB)	9.2	0.6	1.1	1.6
Memory save	1×	15.3×	8.4×	**5.8**×

Table 3. Effectiveness of cascade feature fusion unit (CFF) and cascade label guidance (CLG). 'DC3', 'DC5' and 'DC7' denote replacing 'bilinear upsampling + dilated convolution' with deconvolution operation with kernels 3×3, 5×5 and 7×7 respectively.

DC3	DC5	DC7	CFF	CLG	mIoU (%)	Time (ms)
✓				✓	66.7	31
	✓			✓	66.7	34
		✓		✓	68.0	38
			✓	✓	**67.7**	**33**
			✓		66.8	33

Model Compression. Apart from the above two strategies, another natural way to reduce network complexity is to trim kernels in each layer. Compressing models becomes an active research topic in recent years due to the high demand. The solutions [38–41] can make a complicated network reduce to a lighter one under user-controlled accuracy reduction. We adopt recent effective classification model compression strategy presented in [41] on our segmentation models. For each filter, we first calculate the sum of kernel ℓ_1-norm. Then we sort these sum results in a descending order and keep only the most significant ones. Disappointingly, this strategy also does not meet our requirement given the compressed models listed in the right of Table 1. Even by keeping only a quarter of kernels, the inference time is still too long. Meanwhile the corresponding mIoU is intolerably low – it already cannot produce reasonable segmentation for many applications.

Cascade Branches. We do ablation study on cascade branches, the results are shown in Table 2. Our baseline is the half-compressed PSPNet50, 170 ms inference time is yielded with mIoU reducing to 67.9%. They indicate that model compression has almost no chance to achieve real-time performance under the condition of keeping decent segmentation quality. Based on this baseline, we test our ICNet on different branches. To show the effectiveness of the proposed

Table 4. Predicted mIoU and inference time on Cityscapes test set with image resolution 1024 × 2048. 'DR' stands for image downsampling ratio during testing (e.g, DR = 4 represents testing at resolution 256 × 512). Methods trained using both fine and coarse data are marked with '†'.

Method	DR	mIoU (%)	Time (ms)	Frame (fps)
SegNet [3]	4	57.0	60	16.7
ENet [8]	2	58.3	13	76.9
SQ [9]	No	59.8	60	16.7
CRF-RNN [16]	2	62.5	700	1.4
DeepLab [2]	2	63.1	4000	0.25
FCN-8S [1]	No	65.3	500	2
Dilation10 [14]	No	67.1	4000	0.25
FRRN [12]	2	71.8	469	2.1
PSPNet[a] [5]	No	81.2	1288	0.78
ICNet	No	**69.5**	**33**	**30.3**
ICNet†	No	**70.6**	**33**	**30.3**

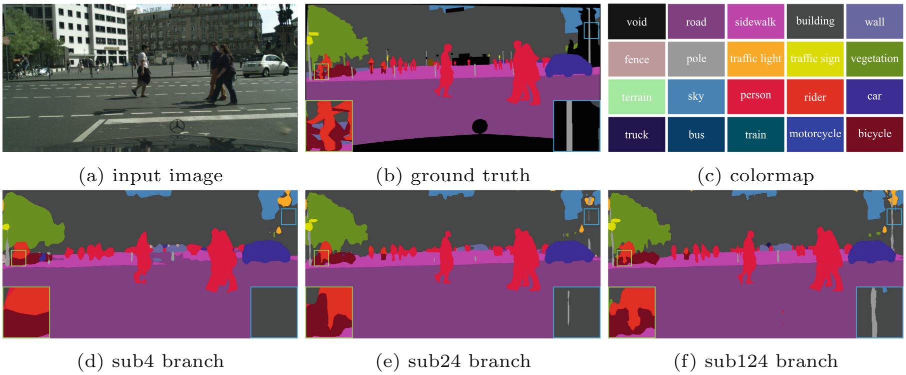
[a] Single network forward costs 1288 ms (with TitanX Maxwell, 680 ms for Pascal) while mIoU aimed testing for boosting performance (81.2% mIoU) costs 51.0 s

cascade framework, we denote the outputs of low-, medium- and high-resolution branches as 'sub4', 'sub24' and 'sub124', where the numbers stand for the information used. The setting 'sub4' only uses the top branch with the low-resolution input. 'sub24' and 'sub124' respectively contain top two and all three branches.

We test these three settings on the validation set of Cityscapes and list the results in Table 2. With just the low-resolution input branch, although running time is short, the result quality drops to 59.6%. Using two and three branches, we increase mIoU to 66.5% and 67.7% respectively. The running time only increases by 7ms and 8ms. Note our segmentation quality nearly stays the same as the baseline, and yet is 5.2× times faster. The memory consumption is significantly reduced by 5.8×.

Cascade Structure. We also do ablation study on cascade feature fusion unit and cascade label guidance. The results are shown in Table 3. Compared to the deconvolution layer with 3 × 3 and 5 × 5 kernels, with similar inference efficiency, cascade feature fusion unit gets higher mIoU performance. Compared to deconvolution layer with a larger kernel with size 7 × 7, the mIoU performance is close, while cascade feature fusion unit yields faster processing speed. Without the cascade label guidance, the performance drops a lot as shown in the last row.

Methods Comparison. We finally list mIoU performance and inference time of our proposed ICNet on the test set of Cityscapes. It is trained on training and



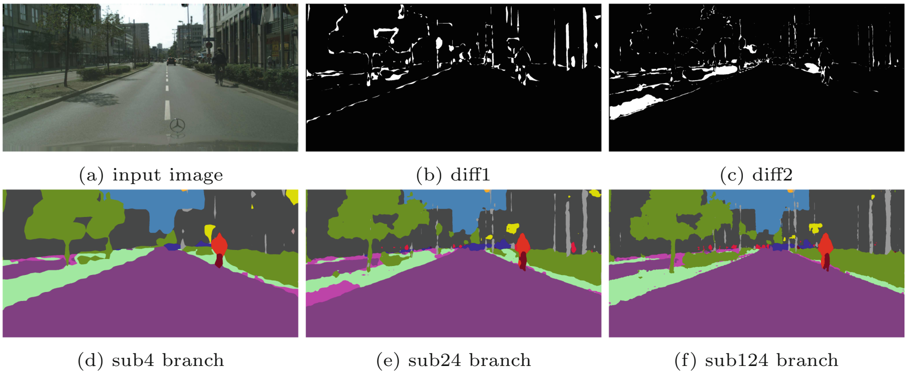
(a) input image (b) ground truth (c) colormap

(d) sub4 branch (e) sub24 branch (f) sub124 branch

Fig. 6. Visual prediction improvement of ICNet in each branch on Cityscapes dataset.

validation sets of Cityscapes for 90K iterations. Results are included in Table 4. The reported mIoUs and running time of other methods are shown in the official Cityscapes leadboard. For fairness, we do not include methods without reporting running time. Many of these methods may have adopted time-consuming multi-scale testing for the best result quality.

Our ICNet yields mIoU 69.5%. It is even quantitatively better than several methods that do not care about speed. It is about 10 points higher than ENet [8] and SQ [9]. Training with both fine and coarse data boosts mIoU performance to 70.6%. ICNet is a 30 fps method on 1024 × 2048 resolution images using only one TitanX GPU card. Video example can be accessed through link[2].



(a) input image (b) diff1 (c) diff2

(d) sub4 branch (e) sub24 branch (f) sub124 branch

Fig. 7. Visual prediction improvement of ICNet. White regions in 'diff1' and 'diff2' denote prediction difference between 'sub24' and 'sub4', and between 'sub124' and 'sub24' respectively. (Color figure online)

[2] https://youtu.be/qWl9idsCuLQ.

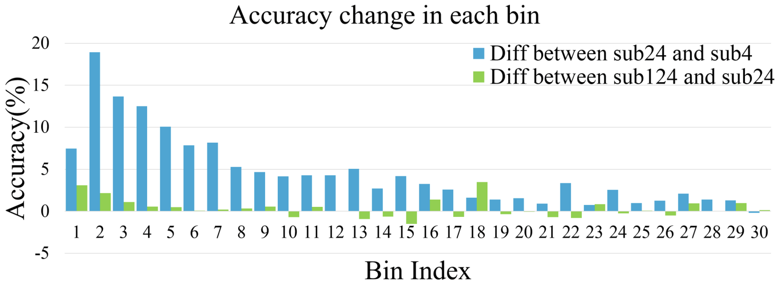


Fig. 8. Quantitative analysis of accuracy change in connected components.

Table 5. Results on CamVid test set with time reported on resolution 720 × 960.

Method	mIoU (%)	Time (ms)	Frame fps
SegNet [3]	46.4	217	4.6
DPN [15]	60.1	830	1.2
DeepLab [2]	61.6	203	4.9
Dilation8 [14]	65.3	227	4.4
PSPNet50 [5]	69.1	185	5.4
ICNet	**67.1**	**36**	**27.8**

Table 6. Results on COCO-Stuff test set with time reported on resolution 640 × 640.

Method	mIoU (%)	Time (ms)	Frame fps
FCN [1]	22.7	169	5.9
DeepLab [2]	26.9	124	8.1
PSPNet50 [5]	32.6	151	6.6
ICNet	**29.1**	**28**	**35.7**

Visual Improvement Figures 6 and 7 show the visual results of ICNet on Cityscapes. With proposed gradual feature fusion steps and cascade label guidance structure, we produce decent prediction results. Intriguingly, output of the 'sub4' branch can already capture most of semantically meaningful objects. But the prediction is coarse due to the low-resolution input. It misses a few small-size important regions, such as poles and traffic signs.

With the help of medium-resolution information, many of these regions are re-estimated and recovered as shown in the 'sub24' branch. It is noticeable that objects far from the camera, such as a few persons, are still missing with blurry object boundaries. The 'sub124' branch with full-resolution input helps refine these details – the output of this branch is undoubted the best. It manifests that our different-resolution information is properly made use of in this framework.

Quantitative Analysis. To further understand accuracy gain in each branch, we quantitatively analyze the predicted label maps based on connected components. For each connected region R_i, we calculate the number of pixels it contains, denoted as S_i. Then we count the number of pixels correctly predicted in the corresponding map as s_i. The predicted region accuracy p_i in R_i is thus s_i/S_i. According to the region size S_i, we project these regions onto a histogram $\mathcal{H}$ with interval $\mathcal{K}$ and average all related region accuracy p_i as the value of current bin.

In experiments, we set bin size of the histogram as 30 and interval $\mathcal{K}$ as 3,000. It thus covers region size S_i between 1 to 90K. We ignore regions with size exceeding 90K. Figure 8 shows the accuracy change in each bin. The blue histogram stands for the difference between 'sub24' and 'sub4' while the green histogram shows the difference between 'sub124' and 'sub24'. For both histograms, the large difference is mainly on the front bins with small region sizes. This manifests that small region objects like traffic light and pole can be well improved in our framework. The front changes are large positives, proving that 'sub24' can restore much information on small objects on top of 'sub4'. 'sub124' is also very useful compared to 'sub24'.

5.3 CamVid

CamVid [17] dataset contains images extracted from high resolution video sequences with resolution up to 720 × 960. For easy comparison with prior work, we adopt the split of Sturgess et al. [42], which partitions the dataset into 367, 100, and 233 images for training, validation and testing respectively. 11 semantic classes are used for evaluation.

The testing results are listed in Table 5, our base-model is no compressed PSPNet50. ICNet gets much faster inference speed than other methods on this high resolution, reaching the real-time speed of 27.8 fps, 5.7 times faster than the second one and 5.1 times faster compared to the basic model. Apart from high efficiency, it also accomplishes high quality segmentation. Visual results are provided in the supplementary material.

5.4 COCO-Stuff

COCO-Stuff [18] is a recently labeled dataset based on MS-COCO [43] for stuff segmentation in context. We evaluate ICNet following the split in [18] that 9 K images are used for training and another 1 K for testing. This dataset is much more complex for multiple categories – up to 182 classes are used for evaluation, including 91 thing and 91 stuff classes.

Table 6 shows the testing results. ICNet still performs satisfyingly regarding common thing and stuff understanding. It is more efficient and accurate than modern segmentation frameworks, such as FCN and DeepLab. Compared to our baseline model, it achieves 5.4 times speedup. Visual predictions are provided in the supplementary material.

6 Conclusion

We have proposed a real-time semantic segmentation system ICNet. It incorporates effective strategies to accelerate network inference speed without sacrificing much performance. The major contributions include the new framework for saving operations in multiple resolutions and the powerful fusion unit.

We believe the optimal balance of speed and accuracy makes our system important since it can benefit many other tasks that require fast scene and object segmentation. It greatly enhances the practicality of semantic segmentation in other disciplines.

References

1. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
2. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: ICLR (2015)
3. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: a deep convolutional encoder-decoder architecture for image segmentation. arXiv:1511.00561 (2015)
4. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV (2015)
5. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
6. Wu, Z., Shen, C., van den Hengel, A.: Wider or deeper: revisiting the ResNet model for visual recognition. arXiv:1611.10080 (2016)
7. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
8. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: a deep neural network architecture for real-time semantic segmentation. arXiv:1606.02147 (2016)
9. Treml, M., et al.: Speeding up semantic segmentation for autonomous driving. In: NIPS Workshop (2016)
10. Wang, P., et al.: Understanding convolution for semantic segmentation. arXiv:1702.08502 (2017)
11. Lin, G., Milan, A., Shen, C., Reid, I.D.: RefineNet: multi-path refinement networks for high-resolution semantic segmentation. In: CVPR (2017)
12. Pohlen, T., Hermans, A., Mathias, M., Leibe, B.: Full-resolution residual networks for semantic segmentation in street scenes. In: CVPR (2017)
13. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv:1606.00915 (2016)
14. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
15. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: ICCV (2015)
16. Zheng, S., et al.: Conditional random fields as recurrent neural networks. In: ICCV (2015)
17. Brostow, G.J., Fauqueur, J., Cipolla, R.: Semantic object classes in video: a high-definition ground truth database. Pattern Recognit. Lett. **30**, 88–97 (2009)
18. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: thing and stuff classes in context. arXiv:1612.03716 (2016)
19. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. TPAMI **33**, 2368–2382 (2011)
20. Chen, L., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: scale-aware semantic image segmentation. In: CVPR (2016)

21. Hariharan, B., Arbeláez, P.A., Girshick, R.B., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: CVPR (2015)
22. Xia, F., Wang, P., Chen, L.-C., Yuille, A.L.: Zoom better to see clearer: human and object parsing with hierarchical auto-zoom net. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 648–663. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_39
23. Girshick, R.: Fast R-CNN. In: ICCV (2015)
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS. (2015)
25. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
26. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: CVPR (2017)
27. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
28. Romera, E., Alvarez, J.M., Bergasa, L.M., Arroyo, R.: Efficient ConvNet for real-time semantic segmentation. In: Intelligent Vehicles Symposium (IV) (2017)
29. Shelhamer, E., Rakelly, K., Hoffman, J., Darrell, T.: Clockwork convnets for video semantic segmentation. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 852–868. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_69
30. Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition. In: CVPR (2017)
31. Kundu, A., Vineet, V., Koltun, V.: Feature space optimization for semantic video segmentation. In: CVPR (2016)
32. Gadde, R., Jampani, V., Gehler, P.V.: Semantic video CNNs through representation warping. In: ICCV (2017)
33. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
34. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 519–534. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_32
35. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes VOC challenge. IJCV **88**, 303–338 (2010)
36. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ADE20K dataset. arXiv:1608.05442 (2016)
37. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding. In: ACM MM (2014)
38. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1mb model size. arXiv:1602.07360 (2016)
39. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding. In: ICLR (2016)
40. Han, S., et al.: DSD: regularizing deep neural networks with dense-sparse-dense training flow. In: ICLR (2017)
41. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: ICLR (2017)

42. Sturgess, P., Alahari, K., Ladicky, L., Torr, P.H.: Combining appearance and structure from motion features for road scene understanding. In: BMVC (2009)
43. Lin, T.-Y.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48