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

Hengshuang Zhao[1] Xiaojuan Qi[1] Xiaoyong Shen[1] Jianping Shi[2] Jiaya Jia[1]
[1]The Chinese University of Hong Kong [2]SenseTime Group Limited
{hszhao, xjqi, xyshen, leojia}@cse.cuhk.edu.hk, shijianping@sensetime.com

## Abstract

*We focus on the challenging task of realtime 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 compressed-PSPNet-based 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 to quickly achieve high-quality segmentation. Our system yields realtime inference on a single GPU card with decent quality results evaluated on challenging Cityscapes dataset.*

## 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 achieve deep understanding of scene, objects, and human. Development of recent deep *convolutional neural networks* (CNNs) makes remarkable progress on semantic segmentation [19, 2, 1, 20, 33, 30]. The effectiveness of these networks largely depend 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 the accuracy and inference time of different frameworks on Cityscapes [5] dataset.

**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 far 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.

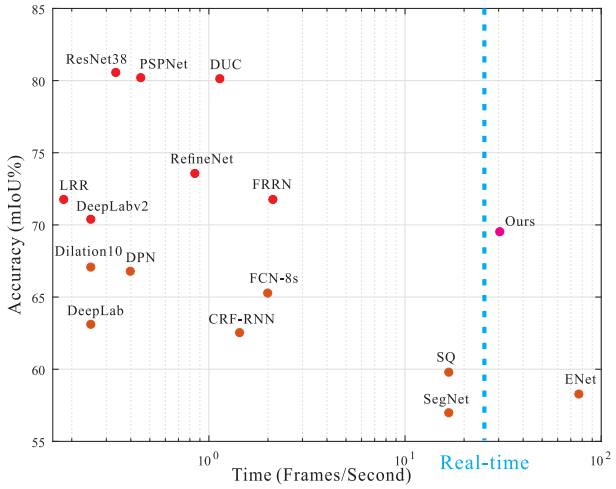



Figure 1. Inference speed and mIoU performance on Cityscapes [5] test set. Methods involved are ResNet38 [30], PSPNet [33], DUC [29], RefineNet [14], LRR [6], FRRN [22], DeepLabv2 [3], Dilation10 [32], DPN [18], FCN-8s [19], DeepLab [2], CRF-RNN [34], SQ [28], ENet [21], SegNet [1], and our ICNet.

Our experiments show that high-accuracy methods of PSPNet [33] and ResNet38 [30] take more than 1 second to predict a $1024 \times 2048$ high-resolution image on one Nvidia TitanX GPU card during testing. These methods fall into the area illustrated in Figure 1 with high accuracy and low speed. Recent fast semantic segmentation methods of SegNet [1], ENet [21], and SQ [28], contrarily, take quite different positions in the plot of Figure 1. The speed is much accelerated; but accuracy notably drops, where the final mIoUs are lower than 60%. These methods locate near the right bottom corner in Figure 1.

**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 Figure 1 and is between the only two available approaches that reach realtime semantic segmentation speed.

Different from previous strategies to only deal with accuracy or speed, we make comprehensive consideration on these two factors that are seemingly contracting. Our contribution is to utilize processing efficiency of low-resolution images and high inference quality of high-resolution ones and propose an image cascade framework to progressively refine segment prediction. The idea is to let low-resolution image go through the full semantic perception network first for a coarse prediction map. Then the proposed cascade fusion unit introduces middle- and high-resolution image feature and improves the coarse semantic map gradually.

This system reliably achieves high efficiency and decent quality in semantic segmentation. The code and trained models are publicly available [1]. Our main contributions and performance statistics are the following.

- We develop an image cascade network (ICNet), which utilizes semantic information in low resolution along with details from high-resolution images efficiently.
- The proposed ICNet achieves 5x+ speedup of inference, and reduces memory consumption by 5+ times.
- Our proposed fast semantic segmentation system can run at resolution $1024\times2048$ in speed of 30.3 fps while accomplishing high-quality results.

## 2. Related Work

Traditional semantic segmentation methods [16] adopt hand-craft feature to learn the representation. Recently, CNN based methods largely improve the performance.

**High Quality Semantic Segmentation** FCN [19] is the pioneer work to replace the last fully-connected layers in classification with convolution layers. DeepLab [2, 3] and [32] use dilated convolution to provide dense labeling and enlarge the receptive field. Encoder-decoder structures [20, 1] can restore the feature maps from higher layers with spatial information from lower layers. Multi-scale feature ensembles are also used in [4, 10, 31].

In [2, 18, 34], conditional random fields (CRF) or Markov random fields (MRF) were used to model the spatial relationship. Zhao *et al*. [33] used spatial pyramid pooling to aggregate context information for prediction. Wu *et al*. [30] explored network width and adopted a wider network to boost performance. In [14], a multi-path refinement network combines multi-scale image features. These methods are effective to increase performance while cannot be directly applied in real-time systems.

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

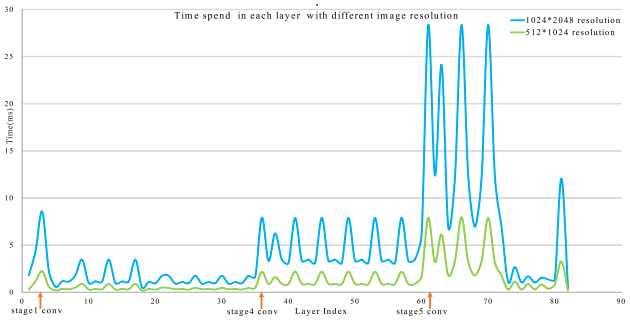



Figure 2. Time spent on PSPNet50 with dilation 8 for two input images. Blue and green curves are for images of resolutions $1024 \times 2048$ and $512 \times 1024$. Roughly running time is proportional to the pixel number.

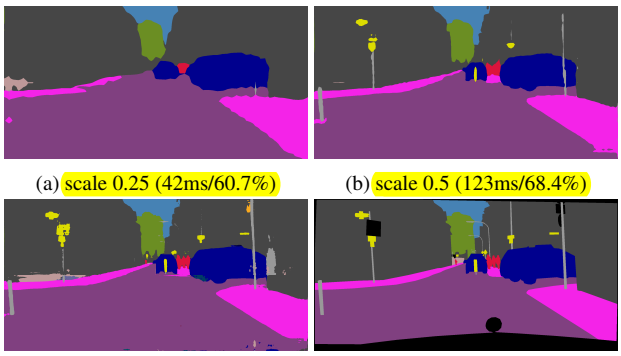
**Fast Semantic Segmentation** In object detection, speed became one important factor in system design [7, 25]. Recent Yolo [23, 24] and SSD [17] much improved it. In contrast, research towards high inference speed in semantic segmentation just commences. SegNet [1] abandons layers to reduce layer parameters and ENet [21] is a lightweight network. These methods greatly raise efficiency. But accuracy turns a concern.

**Video Segmentation Architectures** Videos contain redundant information in frames, which can be utilized to reduce computation. Recent Clockwork [27] reused feature maps given stable video input. Deep feature flow [35] was based on a small-scale optical flow network to propagate features from key frames to others. We note when a high accuracy fast image semantic segmentation framework comes into existence, video segmentation will also be benefitted.

Our work gives a hierarchical structure for fast semantic segmentation. It is by nature different from the method of [14] where the latter focuses on fusion of features from different layers in a single forward pass to increase performance. Our image cascade network, on the contrary, adopts cascade images as input to speed up inference and constructs a real-time segmentation system. More comparisons will be provided in following sections.

## 3. Speed Analysis

To understand our system, we start from analyzing computation of components on the high performance segmentation framework PSPNet [33]. Then we introduce intuitive strategies for speedup in high-resolution image semantic segmentation. Learning from their drawbacks, we will detail our image cascade framework and the cascade feature fusion unit.



(a) scale 0.25 (42ms/60.7%) (b) scale 0.5 (123ms/68.4%)

(c) scale 1 (446ms/71.7%) (d) ground truth

Figure 3. Prediction of PSPNet50 on validation set of Cityscapes. We scale the input images when fed into the network. Values in the brackets are the inference time and mIoU. (a) Taking the 0.25 scaled image as input. (b) Taking the 0.5 scaled image as input. (c) Original resolution image as input. (d) Ground truth.

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

Table 1. Total time spent on PSPNet50 when choosing downsampling factors 8, 16 and 32.

### 3.1. Time Budget

We first study influence of image resolution in semantic segmentation using PSPNet. Fig. 2 shows the time cost with two resolution images in optimized PSPNet50 (structure change detailed in Section 6). Blue line corresponds to the high resolution input with size $1024 \times 2048$ and the green line is for the image with resolution $512 \times 1024$. Computation increases in an order of squares of image resolution.

In additional to image resolution, width of a network or the number of kernels also effect the inference time. As shown in Fig. 2, although feature maps in stage4 and stage5 have the same spatial resolution, i.e., $1/8$ of the original input, computation in stage5 is four times more than stage4. It is because convolutional layers in stage5 double the number of kernels.

### 3.2. Intuitive Speedup

**Downsampling Input** Image resolution is the most critical factor that affect speed, since above analysis shows a half-resolution image only uses quarter time compared to the full-resolution one. 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 feed the resulting images into a FCN-based framework like PSPNet. After getting the prediction, we directly upsample it to the original size.

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

Table 2. Kernel keeping rates in model compression along with related mIoUs and inference time.

This approach empirically has several drawbacks as illustrated in Fig. 3. 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 lost and object boundaries are blurred. Worse, the running time is too long for a real-time system.

**Downsampling Feature** Besides directly downsampling the input image, another straightforward choice is to scale down the feature map by a large ratio in the inference process. FCN [19] 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 Table 1. A smaller feature map can yield faster inference at the cost of sacrificing prediction accuracy. The lost information is similarly details contained in low-level layers. Also, considering that even with the smallest resulting feature map using ratio 1:32, the system still takes about 131ms in inference. There is still a gap towards our goal of real-time segmentation.

**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 [11, 8, 9, 13] can update a complicated network to a lighter one under user-controlled accuracy reduction.

We test the recent effective model compression strategy presented in [13]. For each filter, we first calculate the $L1$ sum of its kernel weights. Then we sort these $L1$ sums in a descending order for keeping only the most significant ones. Disappointingly, this strategy also does not meet our requirement given the compressed models listed in Table 2. 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.

## 4. Our Image Cascade Network

Above analysis and experiments manifest the high difficulty to design a practical real-time segmentation network. Using low-resolution input images is effective to reduce

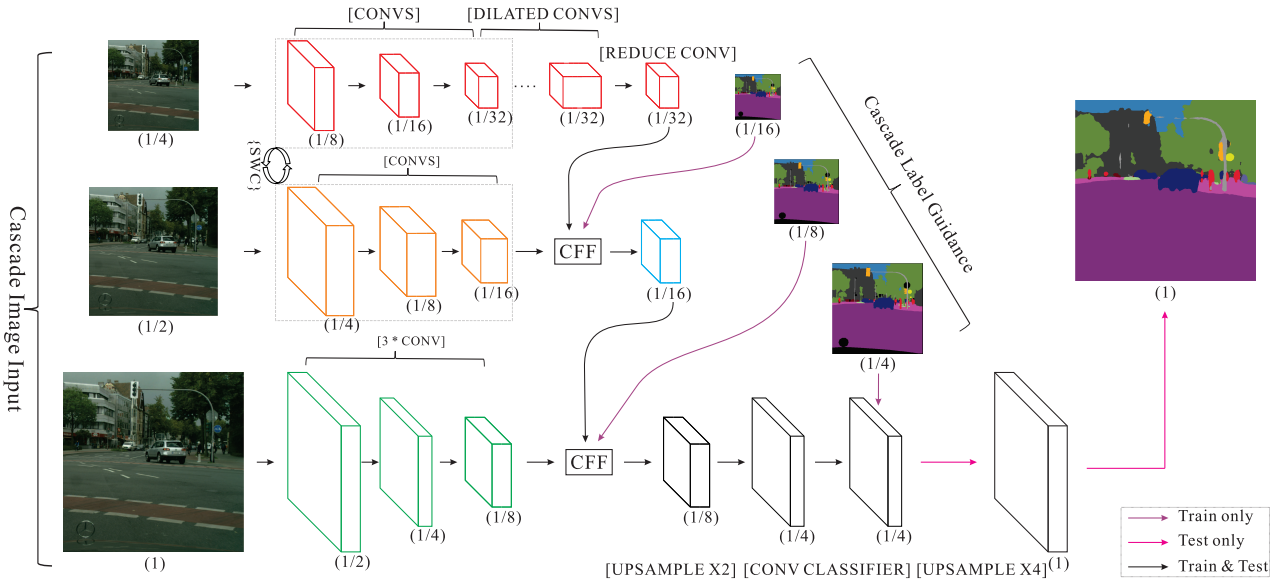



Figure 4. Our image cascade network (ICNet). Numbers in parentheses are feature map size ratios to the full-resolution input. Operations are highlighted in brackets. 'SWC' in the braces stands for sharing weights and computation of several convolution layers in top and middle branches. The final upsampling is only used during testing.

running time. But they would yield coarse prediction maps. Missing a lot of details and generating blurry boundaries are not what we want. Directly feeding high-resolution images into a network is also unbearable in computation.

### 4.1. Main structures and Branches

Our system does not simply choose either way. Instead it takes low-resolution input to only catch a level of semantic information. Our final image cascade network (ICNet) also properly incorporates high-resolution image features to increase result quality under very limited time budget.

**Low Resolution** The proposed framework is illustrated in Fig. 4. The input full-resolution image of scale 1 produces two lower-resolution images after downsampling with scales of $1/2$ and $1/4$. These cascade images are fed into our ICNet in three branches.

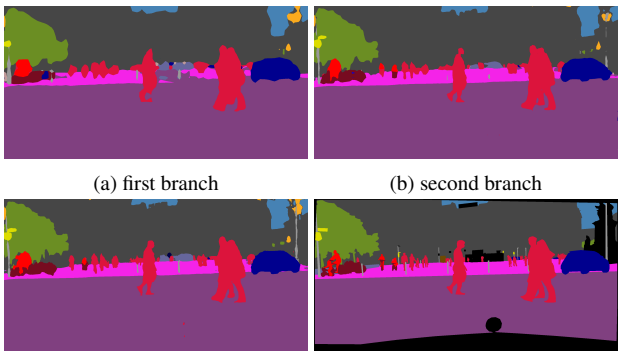
For the lowest resolution input, it goes through the top branch, which is an FCN-based PSPNet architecture. Since the input size is only $1/4$ of the original one, convolution layers correspondingly downsize the feature maps by a ratio of $1/8$ and yield $1/32$ of the original spatial size. Then several dilated convolution layers are used to enlarge the receptive fields without downsampling the spatial size, outputting feature maps with sizes $1/32$ of original ones.

**Median Resolution** For the $1/2$ size middle-resolution image, it is processed in the second branch. Going through several convolution layers with downsampling rate 8, the output feature maps are of size $1/16$ of the original ones. To fusion the $1/16$ size feature map with the $1/32$ size feature map in the top branch, we propose a cascade feature fusion (CFF) unit that will be discussed later in this paper. This fusion yields the combined feature map with resolution $1/16$ of the original one. It is noteworthy that the convolutional parameters can be shared between the $1/4$ and $1/2$ resolution inputs, thus saving computational and reducing parameter number.

**High Resolution** For the high-resolution image, similar to the operation in the second branch, it is processed by several convolutional layers with downsampling rate 8. A $1/8$ size feature map is resulted. Since the median-resolution image already restore most semantically meaningful details that are missing in the low-resolution one, we can safely limit the number of convolutional layers when processing the high-resolution input.

Here we use only three convolutional layers each with kernel size $3 \times 3$ and stride 2 to downsize the resolution to $1/8$ of the original input. Similarly with the fusion as described in branch two, we use the CFF unit to incorporate the output of previous CFF unit and current feature map in full resolution in branch three. Finally, we get the feature map size $1/8$ of the original one.

**Cascade Label Guidance** For the output feature map in each resolution, we first upsample it by a factor of 2 that will be detailed later when describing the CFF unit. To enhance the learning procedure, we adopt a cascade label guidance strategy. It uses $1/16$, $1/8$, and $1/4$ of ground truth labels to guide the learning stage of low, median and high resolution input. In the testing phase, the low and median guidance operations are simply abandoned, where only high-resolution



(a) first branch (b) second branch

(c) third branch (d) ground truth

Figure 5. Prediction of ICNet on each branch.

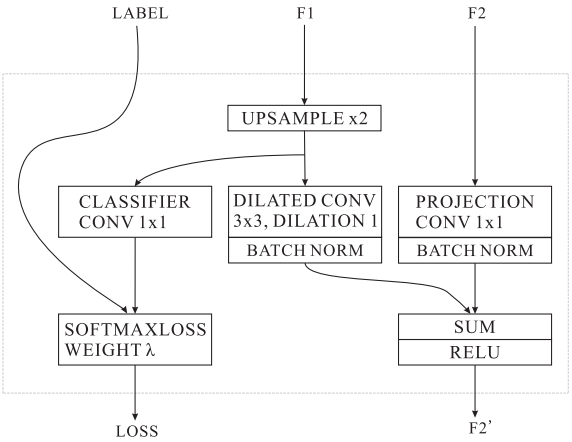



Figure 6. Cascade feature fusion unit. Given input two feature maps $F_1$ and $F_2$ where the spatial resolution of the latter is twice of the former one, the fused feature map $F_2'$ is of the same spatial size as $F_2$.

branch is retained. This cascade label guidance is surprisingly effective to cut down inference cost while not scarifying much accuracy of our final results.

This cascade label guidance strategy grants each branch auxiliary learning promote, making gradient optimization smoother for easy training iteration. With more powerful learning ability in each branch, the final prediction map is far from being dominated by any single branch. Meanwhile, abandon of guidance parts during testing is a type of gain in terms of efficiency.

### 4.2. Branch Analysis

With these gradual feature fusion steps and cascade label guidance structure, we produce reasonably accurate prediction results. The output map of the second stage is much finer than the first branch and the output of the third branch is undoubted the best, as shown in Figure 5. It manifests that our different-resolution information is properly made use of in this framework.

Meantime, in our ICNet, the deepest network structure with the most layers are only applied to the low resolution data, which can efficiently extract most semantic information. Therefore even with more than 50 layers, the inference operation and memory consumption are not large as 18ms and 0.6GB. Our experimental analysis in following sections will illustrate more statistics.

There are 17 convolutional layers in branch two and only 3 in branch three. With these small numbers of convolutional layers to process median and high resolution input data, the total number of inference operations from branches two and three are well constrained. Meantime, convolutional layers in branch two share weights and computation with branch one. Thus only 6ms more is spent to construct the fusion feature map using two branches. Branch three has even less layers. Although the resolution is high, the inference time in branch three is just 9ms. With all these three branches, our proposed ICNet becomes a very efficient and memory friendly architecture that can achieve high prediction in semantic segmentation.

### 4.3. Difference from Other Cascade Structures

Several other segmentation frameworks also incorporate the multi-scale structure. FCN [19] and DeepLab-MSC [2] sum the multi-resolution prediction maps to generate the final score map. SegNet [1] and UNet [26] adopt skip connections during deconvolution to exploit middle-level features. LRR [6] applied a pyramid reconstruction architecture, where the feature maps are reconstructed from bottom up. RefineNet [14] fuses multi-path feature maps from different layers using long-range residual connections.

We note these methods focus on fusing features from different layers from a single-scale input or multi-scale ones [3]. All input data goes through the whole network. They all face the same problem of expensive computation given high-resolution input. Different by nature from all these methods, our ICNet uses the low-resolution input to go through the main semantic segmentation branch and adopts the high-resolution information to help refinement. The resulting feature maps are much reduced while still maintaining necessary details.

## 5. Cascade Feature Fusion and Final Model

To combine cascade features from different-resolution images, we finally propose a cascade feature fusion (CFF) unit as shown in Fig. 6. The input to this unit contains three components: the two feature maps $F_1$ and $F_2$ of resolution $H_1 \times W_1 \times C_1$ and $H_2 \times W_2 \times C_2$ and a ground truth label in resolution $H_2 \times W_2 \times 1$. $F_2$ is with doubled size in spatial dimension of $F_1$. Upsampling is applied to make $F_1$ the same size as $F_2$. Then a dilated convolution layer with kernel size $3 \times 3$ and dilation 1 is applied to refine upsampled features.

This dilated convolution can combine feature information from several originally neighboring pixels while direct upsampling makes each pixel depends on just one location. Compared with deconvolution operation on the original feature, dilated convolution only need small kernels. In our implementation, for feature $F_2$, a projection convolutional layer with kernel size $1 \times 1$ is utilized to project it with the same size as the output of feature $F_1$.

Then two batch normalization layers are used to normalize these two features. Followed by an element-wise layer with 'SUM' operation and a 'ReLU' layer, we get the fused feature $F_2'$, which has the same resolution as $F_2$. To enhance learning of $F_1$, we use an auxiliary label guidance to the upsampled $F_1$. The auxiliary loss weight is set to 0.4 as in [33].

### 5.1. The Loss Function

To train ICNet, we append softmax cross entropy loss in each branch denoted as $L_1$, $L_2$ and $L_3$ with corresponding weights $\lambda_1$, $\lambda_2$, and $\lambda_3$. The total loss is $L$. The overall loss function is

$$L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3. \quad (1)$$

The framework is trained to minimize the above loss function. All the losses we adopted are the cross-entropy loss on the corresponding downsampled score maps.

### 5.2. Final Model Compression

Our image cascade network tackles the computation complexity problem from the system design perspective. As discussed in Section 3.2, another orthogonal strategy is to reduce kernels in each layer. Directly training a simplified network, such as ENet [21], faces the problem of under-fitting, thus possibly capping the segmentation accuracy in the structure level. We choose the scheme of compressing models. Note current model compression methods [11, 8, 9, 13] mainly aim at image classification. To quickly reduce structure complexity while not greatly affecting accuracy, we instead adopt a simple but effective process given below.

We compress our model in a progressive way. Taking compression rate $1/2$ as an example, instead of removing a half of kernels directly, we first choose to keep $3/4$ of the kernels and initialize this compressed model for following fine tuning. After it is done, we remove more kernels and repeat this process until the goal of compression is achieved.

To choose the portion of kernels to keep, we follow the strategy of [13]. For each filter, we calculate the $L1$ sum of its kernel weights. Then we sort these sums in a descending order for ranking. Finally, we remove those least important kernels which have smaller weights. Progressive compression does not drastically update all parameters and thus is practically important to safely reduce network scales.

## 6. Experimental Evaluation

We conduct experiments based on learning platform Caffe [12]. All our experiments are performed on a workstation with 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 number to 100 to eliminate accidental errors during testing.

Our network structure is modified from PSPNet. We changed the concatenation operations in the pyramid pooling module to summation, thus reducing feature length from 4096 to 2048. We changed the kernel size in the convolution layer after pyramid pooling from original $3 \times 3$ to $1 \times 1$. It does not much affect final accuracy but saves computation a lot. To train the 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 30k. Momentum is 0.9 and weight decy is 0.0001. Data augmentation contains random mirror and rand resizing is between 0.5 and 2.

**Dataset and Evaluation Metrics** We apply our system to the recent urban scene understanding data, i.e., Cityscapes [5]. This dataset contains high-resolution images up to $1024 \times 2048$, which is a big challenge for fast semantic segmentation. The data set 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 class labels of road, person, car, etc. 19 of them are used in training and testing. For evaluation, both *mean of class-wsie intersection over union* (mIoU) and network forward time are used.

### 6.1. Model Compression

We first evaluate how model compression is useful in reducing computation in semantic segmentation. We take PSPNet50 as the example. When directly compressing PSPNet50 following our previously described procedure, 170ms inference time is yielded with mIoU reducing to 67.9% as shown in Table 3. They indicate that only model compression has almost no chance to achieve realtime performance under the condition of keeping decent segmentation quality. In what follows, we take the model-compressed PSPNet50, which is reasonably accelerated, as our baseline system for comparison.

Based on the compressed PSPNet50, our ICNet yields mIoU 67.7% with inference at 30.3 fps. It is 5.2 times accelerated.

### 6.2. Ablation Study for Image Cascade Framework

To show the effectiveness of the proposed cascade framework, we divide it into three stages and experiment with three settings denoted as 'sub4', 'sub24' and 'sub124'.

| Items | Baseline | ICNet |
|---|---|---|
| mIoU (%) | 67.9 | **67.7** |
| Time (ms) | 170 | 33 |
| Frame (fps) | 5.9 | **30.3** |
| Speedup | 1× | **5.2**× |
| Memory (G) | 9.2 | 1.6 |
| Memory Save | 1× | **5.8**× |

Table 3. Performance of baseline and ICNet on validation set of Citysapes. The baseline method is the structure-optimized PSP-Net50 with compress operation by half.

| 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 4. Our ICNet performance on the validation set of Cityscapes with different settings.

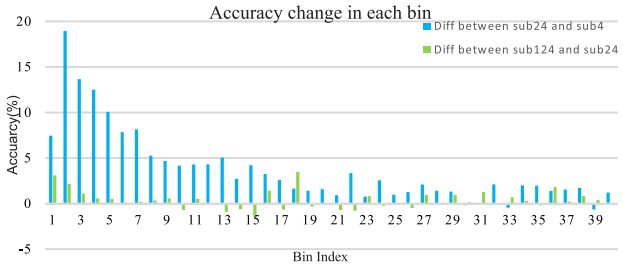



Figure 7. Quantitative analysis of accuarcy change in connected components.

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 4. With just the low-resolution input branch, although running time is short, the result quality drops to 59.6%. By 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 result quality is already close to the powerful PSP-Net50, and yet the method is 5.2+ times faster. The memory consumption is significantly reduced.

**Visual Comparison** Several examples are shown in Fig. 8. Among the seven rows, the top two show input images and their segmentation ground truth labels, the following three prediction maps are from settings 'sub4', 'sub24' and 'sub124' respectively. The second-last-row binary maps show missing information using setting 'sub4' that can be recovered by 'sub24'. The final binary maps contain missing details when applying 'sub24' that however can be recovered by 'sub124'.

| Method | Sub | mIoU (%) | Time (ms) | Frame (fps) |
|---|---|---|---|---|
| SegNet [1] | 4 | 57.0 | 60 | 16.7 |
| ENet [21] | 2 | 58.3 | 13 | 76.9 |
| SQ [28] | no | 59.8 | 60 | 16.7 |
| CRF-RNN [34] | 2 | 62.5 | 700 | 1.4 |
| DeepLab [2] | 2 | 63.1 | 400 | 0.25 |
| FCN-8S [19] | no | 65.3 | 500 | 2 |
| Adelaide [15] | no | 66.4 | 35000 | 0.03 |
| Dilation10 [32] | no | 67.1 | 4000 | 0.25 |
| ICNet | no | 69.5 | 33 | 30.3 |

Table 5. Final mIoUs on Cityscapes test set and inference time.

Intriguingly, the output of 'sub4' can already capture most of semantically meaningful objects. But the prediction is really coarse due to the low-resolution input. It misses a few small-size important regions, such as poles and traffic signs. Using the 'sub24' setting, a few of these regions are re-estimated correctly. It is noticeable that objects far from the camera including those persons are still lost with many blurry object boundaries. Our final system 'sub124' yields the best quality results remedying most of these problems.

**Quantitative Analysis** To further understand efficiency gain in each branch, we quantitatively analyze the predicted label maps based on connected components. Here we use ground truth label maps. For each connected region $R_i$, we get the number of pixels it contains, denotes as $N_i$. Then we count the number of pixels correctly predicted in the corresponding map as $n_i$. The predicted region accuracy $p_i$ in $R_i$ is thus $n_i/N_i$. According to the region size $N_i$, we project these regions onto a histogram $H$ with interval $K$ and average all related region accuracy $p_i$ as the value of current bin.

In experiments, we set bin size of the histogram as 40 and interval $K$ as 3,000. It thus covers region size $N_i$ between 1 to 120k. We ignore regions with size exceeding 120k. Fig. 7 shows the accuracy change in each bin. The blue histogram stands for the difference between stage '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'.

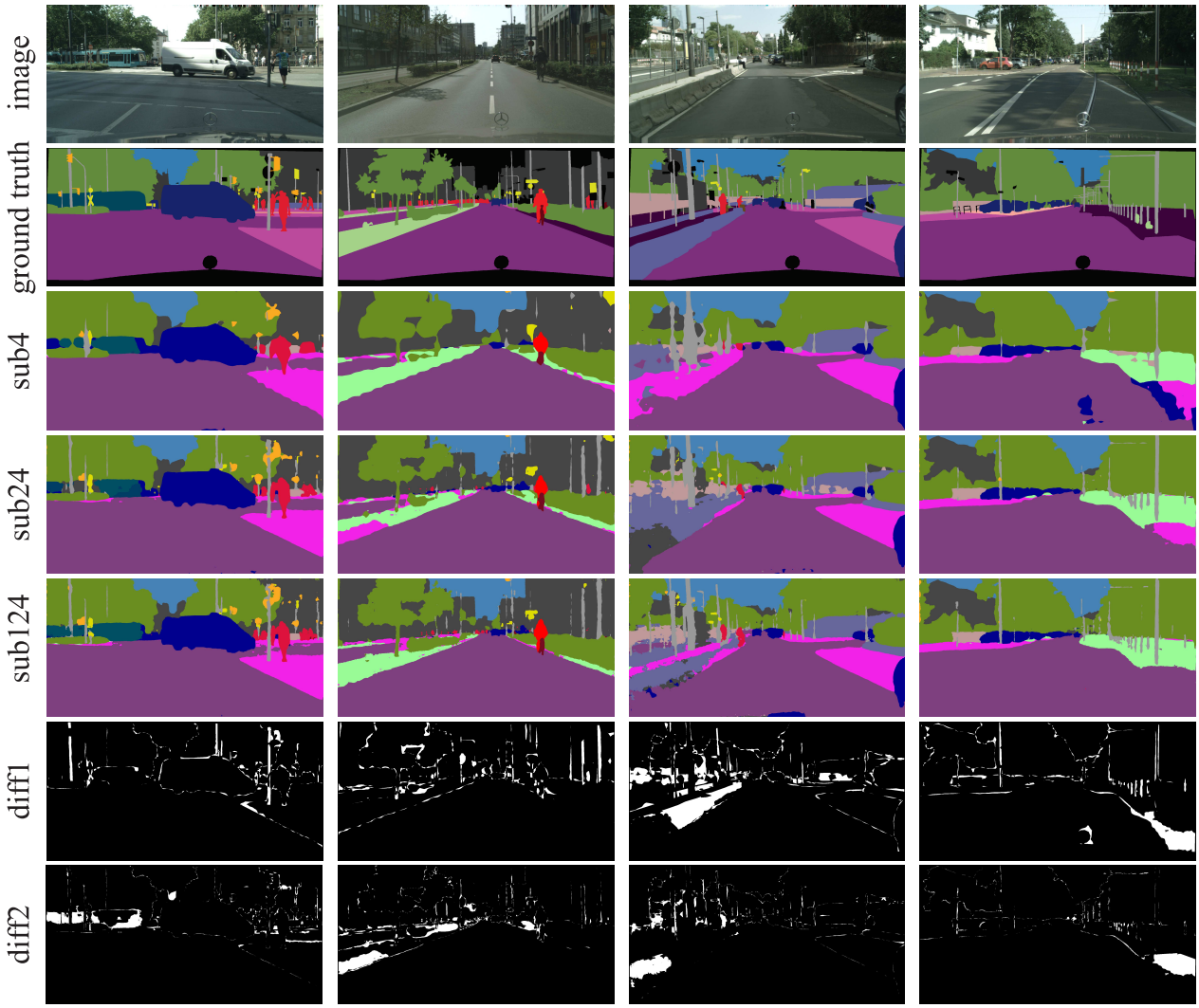



Figure 8. Four frame Examples. Top two rows: input frames and ground truth labels. Middle three rows: Results from 'sub4', 'sub24' and 'sub124'. Bottom two rows: 'diff1' and 'diff2' denote difference maps between 'sub4' and 'sub24', and between 'sub24' and 'sub124'.

### 6.3. Final Results and Comparison

We finally tabulate mIoU performance and inference time of our proposed ICNet on test set of Cityscapes together with comparison with other methods. It is trained on training and validation sets of Cityscapes for 90K iterations. Results are listed in Table 5. The reported mIoUs and running time of other methods are those shown in the official Cityscapes leadboard. For fairness, we do not include methods without reporting running time. Many of these methods may have adopted very time-consuming multi-scale testing for high result quality.

Compared with other methods that report time, our ICNet yields mIoU 69.5%. It is even quantitatively better than most methods that do not care about speed. It is about 10 points higher than the fast-speed SegNet [1], ENet [21] and SQ [28]. It a 30 fps method on $1024 \times 2048$ resolution images using only one TitanX GPU card. Video example can be accessed through link [2].

## 7. Conclusion

We have proposed a realtime semantic segmentation system ICNet. It incorporates effective strategies to simplify network structures without significantly reducing 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.

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

## References


[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*, 2015. 1, 2, 5, 7, 8

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv:1412.7062*, 2014. 1, 2, 3, 5, 7

[3] L. 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:1606.00915*, 2016. 1, 2, 5

[4] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 2

[5] 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. 1, 6

[6] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016. 1, 5

[7] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[8] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv:1510.00149*, 2015. 3, 6

[9] S. Han, J. Pool, S. Narang, H. Mao, S. Tang, E. Elsen, B. Catanzaro, J. Tran, and W. J. Dally. DSD: regularizing deep neural networks with dense-sparse-dense training flow. *arXiv:1607.04381*, 2016. 3, 6

[10] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2

[11] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *arXiv:1602.07360*, 2016. 3, 6

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 6

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv:1608.08710*, 2016. 3, 6

[14] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 1, 2, 5

[15] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 7

[16] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *TPAMI*, 2011. 2

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[18] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015. 1, 2

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 3, 5, 7

[20] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1, 2

[21] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016. 1, 2, 6, 7, 8

[22] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017. 1

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[24] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *arXiv:1612.08242*, 2016. 2

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5

[27] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *ECCVW*, 2016. 2

[28] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler1, and S. Hochreiter. Speeding up semantic segmentation for autonomous driving. *NIPSW*, 2016. 1, 7, 8

[29] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell. Understanding convolution for semantic segmentation. *arXiv:1702.08502*, 2017. 1

[30] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv:1611.10080*, 2016. 1, 2

[31] F. Xia, P. Wang, L. Chen, and A. L. Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *ECCV*, 2016. 2

[32] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv:1511.07122*, 2015. 1, 2, 7

[33] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 2, 6

[34] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 1, 2, 7

[35] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *CVPR*, 2017. 2