16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

Uncertainty based model selection for fast semantic segmentation

Yu-Hui Huang
KU Leuven
yu-hui.huang@esat.kuleuven.be

Stamatios Georgoulis
ETHZ
georgous@vision.ee.ethz.ch

Marc Proesmans
KU Leuven
marc.proesmans@esat.kuleuven.be

Luc Van Gool
KU Leuven / ETHZ
vangool@vision.ee.ethz.ch

Abstract

Semantic segmentation approaches can largely be divided into two categories. One with accurate results but slow inference, and another one with real-time inference but sacrificing some performance for speed. In this paper, we try to exploit the benefits of both categories, i.e. accuracy and speed, through the use of model selection techniques. Using the uncertainty, calculated from the entropy map, as our selection criterion, we leverage the speed of the fast, but not so accurate, model for regions with high certainty, that comprise the majority of the input image, while for a few, carefully selected regions with low certainty we employ an accurate, yet expensive, model, to predict the semantic labels. Our experimental results show that our method greatly boosts the performance of the baseline model, while retaining reasonable inference speeds.

1 Introduction

Semantic segmentation is the task of assigning semantic labels from a predefined set of classes at every pixel of an input image. With the recent advent of Convolutional Neural Networks (CNNs) [1] this task has received renewed interest. The improved performance with CNN architectures has enabled applications in a broad number of fields, such as autonomous driving, augmented reality, human-computer interaction, and video surveillance. Apart from accurate segmentation results these applications typically require efficient inference speeds in order to be deployed in real-time systems. However, most state-of-the-art approaches in semantic segmentation usually excel on one of these aspects, i.e. performance or speed, sacrificing the other.

On the one hand, methods like PSPNet [2] and the DeepLab models [8, 7] employ very deep architectures trained on large datasets to achieve high performance. On top of that, they add spatial pyramid pooling modules to segment objects at different scales. On the other hand, real-time semantic segmentation models, like ENet [4] and ICNet [3], use different design choices to achieve efficient inference speeds. The input images are downsampled to reduce the computational complexity and the number of feature channels are decreased to boost the inference speed. However, these design choices compromise the segmentation results as they essentially sacrifice the accuracy for faster inference. If possible, one would like to have both of these aspects, i.e. accuracy and speed, in a single framework.

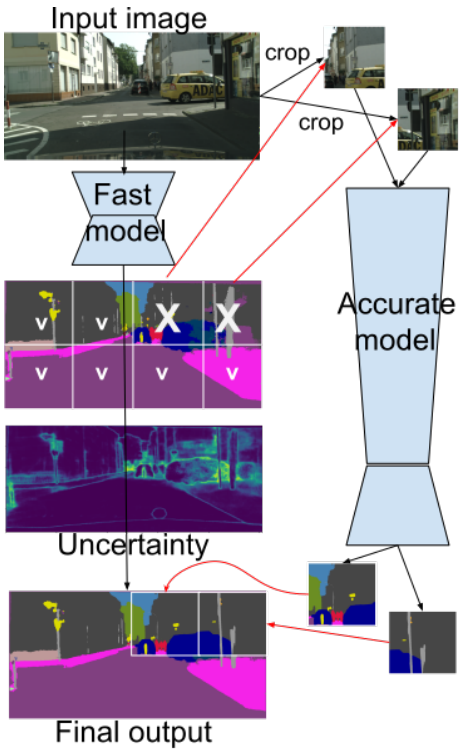

Figure 1. The pipeline of our model selection approach. The fast model on the left operates on the full image, and outputs an initial segmentation map and an uncertainty map indicating the image blocks with lower accuracy. The accurate network on the right processes the corresponding blocks, and the results are inserted into the initial response to create a final output.

To get the best of both worlds, in this paper we propose a model selection technique for the semantic

segmentation problem. The selection is done on the basis of per-pixel values of the uncertainty map (see Fig. 2) generated by a real-time segmentation model, like [4, 3]. This allows us to leverage the speed of the fast, but not so accurate, model for regions with high certainty, while for regions with low certainty we employ an accurate, yet expensive, model, like [2, 7], to predict the semantic labels. As will be shown in the following sections, our approach exploits the fact that regions with high prediction certainty, even for real-time segmentation models, comprise the majority of the input image, essentially allowing us to combine the fast and accurate parts of a real-time segmentation model with a few, carefully selected, low certainty parts where the expensive model is computed. These selected, low certainty parts are not only few, but also very low in size compared to the input image, greatly reducing the computational burden of the expensive model. Fig. 1 gives an overview of our model selection pipeline.

The input image first passes through a real-time segmentation network, like ENet [4] or ICNet [3], where both a segmentation and an uncertainty map are predicted. We propose to use the entropy map as a way to measure the prediction uncertainty. Consequently, we split the image into blocks of fixed size, e.g. 256×256 or 512×512, and for each block we compute the average uncertainty from the predicted uncertainty map. If the average uncertainty for a specific block is below a certain threshold we select the predicted segmentation map of the fast model that corresponds to that specific block. As such, this threshold serves as our model selection criterion. For those blocks with higher uncertainty w.r.t. the selected threshold, we crop the corresponding part of the input image and pass it through an expensive segmentation model, like PSPNet [2].

Our contributions can be summarized as follows:

- a model selection technique for combining the speed of fast segmentation models with the accuracy of high performing segmentation approaches.
- an adaptive selection procedure that can change the ratio of speed versus accuracy on-the-fly by changing the selected threshold value in a slider fashion.

2 Uncertainty estimation

In order to present our uncertainty model, let us consider a deep segmentation network F. Forward passing an image I through the network F, we obtain a logit vector y for every pixel of I. Then, a softmax operation is applied to normalize the logits, which after argmax serves as the classification result. As indicated in the literature, the certainty of the classifier can be defined by measuring the classification margin. For deep neural networks the softmax value is the most obvious choice. In particular, if the difference between the highest and the second highest value is large, we can claim the predicted class with higher certainty.

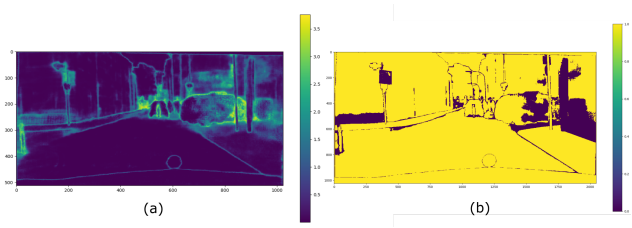

Figure 2. (a) An example of the entropy map estimated from ENet segmentation result. The higher value means a more uncertain prediction. (b) Corresponding error map of the left figure. Value 1 means correct and 0 means wrong.

Inspired by [5], we select the entropy map as our uncertainty measure. Given the prediction probability after softmax z, the entropy map is computed as follows:

$$H[z] = -\sum_{i=1}^{K} z_i * log_2(z_i) \quad (1)$$

Fig. 2 illustrates an example entropy map and error map estimated from ENet's segmentation result. We can easily observe that the entropy map has high correlation to the error map. In particular, in the plain area, such as road, where ENet is confident it has lower uncertainty value, while on the boundary of objects, where segmentation networks typically fail, it has higher intensity in the entropy map.

3 Model selection

Let us assume that we have two segmentation models: a fast, but not so accurate, model, such as [4, 3], referred to as *fast*, and an accurate, yet expensive, model, like [2, 7], referred to as *accurate*. Given an input image, we first pass it through the fast network, to get a segmentation map as well as an uncertainty map u_i, as explained in the previous section. The predicted uncertainty map is then used to perform model selection based on a threshold value θ that serves as our selection criterion. Below, we describe both a pixel-wise and a block-wise selection mechanism.

3.1 Pixel-wise selection

For pixel-wise selection, we simply select the accurate model for the higher uncertainty pixels, and the fast one for the rest. In practice, we binarize the entropy map based on the threshold value[1] and for a specific pixel coordinate (x,y) we select the result from

[1]Here, we assume that we have only two models to select from, and this is why we need only one binary selection vector. However, the same technique can be extended to N models in

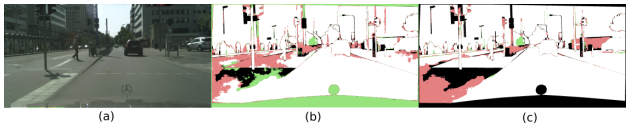

Figure 3. Example of pixel-wise oracle selection applied to the same image (a). (b) from ENet and PSPNet. (c) from ICNet and PSPNet. (Notation: white color: both models are correct, pink color: only the accurate model is correct, green color: only the fast model is correct, and black color: both models are wrong.)

the corresponding model i. Note that, pixel-wise selection requires the segmentation map of both models estimated in advance, which is computationally-wise expensive, but it serves as the theoretical maximum performance one would be able to obtain if the per-pixel selection criterion was possible. We refer to this in the remainder of this paper as the oracle selection. For more details and analysis regarding this we refer the reader to Sec. 4.2.

3.2 Block-wise selection

As explained, although pixel-wise selection gives us accurate results, it is not feasible for the real-time setting we are targeting in this paper, because we would still need to apply the inference to the full image for all models. Instead, we propose block-wise selection to get a better balance between accuracy and speed. Motivated by [6], we crop the image into blocks and for each block we decide which model to do inference from based on our selection criterion. We begin by applying average pooling to each block of the fast model's uncertainty map. For each block, the pooling size/stride equals to the size of the block itself. Based on the selected threshold's value we then determine which block will go through the accurate model. By using a high threshold, less blocks have to go through the accurate model which leads to a faster, but possibly less accurate, result. On the contrary using a low threshold leads to more accurate results with higher inference times as well, since more blocks have to go through the accurate model. In general, the selection criterion, i.e. threshold θ in our case, serves as a slider that determines the desired ratio of speed versus accuracy. We can choose different threshold depending on the application. Take autonomous driving for example, it is especially critical to have a fast but accurate model. With our model selection, we can get the balance between these two criteria.

a cascaded fashion, in which case we would need $N-1$ binary vectors.

4 Experiments

4.1 Settings

Models to select We perform our experiments on the following state-of-the-art models, ENet [4], ICNet [3] and PSPNet [2], where ENet and ICNet serve as fast models and PSPNet as accurate model. The PSPNet in our experiments is trained with Resnet-50 [10] as the backbone. In order to simulate the real-time application, we only do the single-scale inference without flipping during the inference time.

Dataset We select Cityscapes dataset [9] to evaluate our model. The segmentation models we use were trained on the train set only without coarse labels. There are in total 5,000 images with a resolution of 1024 by 2048 pixels. Among those images, 2,975 images are in the train set, 500 images are in the validation set and 1,525 images in the test set. The performance is evaluated in terms of mean Intersection-over-Union (IoU). The first rows of Table 1 show the mean IoU and inference time of each model, which is regarded as our baseline for model selection. The inference time is calculated image-wise on a single NVidia GeForce GTX 1080 Ti and Intel Core i7-6800K CPU. Among these three models, ENet is the fastest one (13ms) but with inferior accuracy (60.1%), while PSPNet is the most accurate one (77.6%) but with long inference time (1652ms).

4.2 Error analysis

Before we present the practical results on model selection, we performed an error analysis of the different segmentation models with respect to one another. In particular, we compare the results of each segmentation model (M_i) with the ground-truth labels (L). We define the difference between these two as the error map (E),

$$E_{M_i}(i,j) = \begin{cases} 1, & \text{if } M_i(i,j) \neq L(i,j), \\ 0, & \text{if } M_i(i,j) = L(i,j). \end{cases}$$

Using the generated error maps, we apply model selection in an oracle way - we select all correct labels from the current model w.r.t. ground-truth labels, before moving to the next model - to analyze if those models are complementary to each other. The purpose of this experiment is to verify whether the different models do different mistakes, and as such there is merit in applying a model selection technique.

For pixel-wise selection, we start from the fast model (ENet or ICNet) and apply the selection. That is, from the error map of ENet (E_1) we select ENet's output only if the value of E_1 equals zero, otherwise we select the outputs from PSPNet. After evaluation on the validation set of Cityscapes, we obtain a mean IoU of 83.3% which serves as the upper bound of pixel-wise

selection from ENet and PSPNet. Similarly, we apply the oracle selection to ICNet and PSPNet and obtain a mean IoU of 83.1%.

To further analyze where the fast models perform better than the accurate model, we perform the same oracle selection in the opposite direction, i.e. we first select the result from the most accurate model (PSPNet), then from the erroneous regions of PSPNet we select the segmentation result from ICNet and so on. Fig. 3 illustrates the guidance for pixel-wise oracle selection. Note that, the green color appears at the object boundaries, e.g. *poles*, *pedestrian*, indicating that fast models can make correct predictions for regions where accurate models fail. Also, from Fig. 3(b) and (c), we can find that different fast model makes different mistakes.

Similarly, we perform block-wise model selection in an oracle way. We choose two sizes of blocks, that is 256×256 and 512×512 with ENet/ICNet and PSPNet to select from. The selection procedure is the same with one difference. For each block, we compare the segmentation result from each model with the ground-truth labels and select the model with the most correct predictions within the block. However, if the number of within-block correct predictions are the same among some models, then we give the faster model higher priority. The result can be found in the second big row of the Table 1. For the case of ICNet plus PSPNet in 256×256 block, the performance is 79.4% which is better than the models it chose from. The results coincide with the pixel-wise experiment which is a strongest indication that irrespective of the selection mechanism the segmentation models are complementary to each other.

4.3 Block-wise inference for PSPNet

As already explained, for block-wise inference we divide the image into rectangular blocks - we experimented with three different sizes of blocks - and evaluate the segmentation performance. The accurate network's input is a single block at a time - depending on the threshold a different number of blocks pass through - and the inference was done without flipping nor multi-scale sampling to achieve fair comparisons. To measure the accuracy, we reassemble the cropped segmentation result back to the original position and measure the performance as in Cityscapes benchmark. In order to obtain the best performance, we finetune the accurate model for the different crop sizes (eg. 256×256 and 512×512). We measure the inference time at all possible locations of the crops and take the mean value. Below, we report their average statistics over all blocks and images for PSPNet.

Table 2 shows the performance w.r.t. accuracy and speed on Cityscapes validation set. The baseline model was trained on 864×864 crops. We observe that the performance in mIoU drops as the crop size becomes smaller. In particular, it drops from 77.6% to 70.9% as the image size goes from 864×864 to 256×256 respectively. This is to be expected as the cropped image restricts the amount of context information available in the input image, which is crucial for segmentation networks. On the other hand, the smaller the cropped image, the faster the inference time is.

Table 1. Performance of the model selection on different block size and threshold with corresponding baselines. The first two big rows present the baseline models and oracle selections and the later two big rows show the results of different model selection. The third column (thres) is the threshold applied to uncertainty map estimated from the faster model (ENet or ICNet). The fifth column (avg. time) presents the average computation time from the whole validation set. And the last column denotes the number of blocks that go through the PSPNet in average per image. (*) denotes overlapping blocks.

	Models_block size	thres	mIoU	avg. time	blocks
baseline	ENet(E)[4]	/	60.1%	13ms	/
	ICNet(I) [3]	/	67.3%	33ms	/
	PSPNet(P) [2]	/	77.6%	1652ms	/
oracle	E+P_256×256	/	78.8%	1652ms	30.8/32
	E+P_512×512	/	78.4%	1652ms	7.8/8
	I+P_256×256	/	79.4%	1652ms	27.8/32
	I+P_512×512	/	78.9%	1652ms	7.0/8
ENet+PSP	E+P_256×256	0.25	71.6%	471ms	16.0/32
	E+P_256×256	0.5	70.1%	288ms	9.6/32
	E+P_512×512	0.25	73.7%	366ms	4.8/8
	E+P_512×512	0.5	70.6%	210ms	2.7/8
	E+P_480×480(*)	0.25	76.2%	842ms	12.3/18
	E+P_480×480(*)	0.5	73.7%	459ms	6.6/18
ICNet+PSP	I+P_256×256	0.25	71.8%	339ms	10.7/32
	I+P_256×256	0.5	71.5%	169ms	4.7/32
	I+P_512×512	0.25	74.2%	268ms	3.2/8
	I+P_512×512	0.5	70.3%	89ms	0.76/8
	I+P_480×480(*)	0.25	76.4%	571ms	8.0/18
	I+P_480×480(*)	0.5	71.5%	149ms	1.7/18

4.4 Model selection

In this section, we apply our full pipeline, i.e. block-wise model, in the two-model selection setting. In particular, we test the performance of our full pipeline when using one fast and one accurate model as baselines. Again, as fast model we choose either ENet or ICNet, and for the accurate model we adopt PSPNet. The selection criterion is determined by the uncertainty of the fast model. In particular, we compute the entropy map from the segmentation result of the fast model after softmax. For block size, we choose either 256×256 or 512×512, since block sizes larger than 512×512 take too much time for inference on PSPNet. To see the effect of overlapping blocks, we also conduct the experiments on 480×480 blocks with two-third stride. For each experiment, we set a threshold value θ for the selection.

The third/forth big row of Table 1 presents the results of model selection between ENet/ICNet and PSPNet. Based on the uncertainty estimated from the ENet/ICNet segmentation result, we compute the average pooling as mentioned in Sec. 3.2. The blocks with uncertainty value greater than the defined threshold will pass through PSPNet. As we can observe from the table, the combination of ENet and PSPNet takes longer inference time compared to ICNet and PSPNet. For the same model and the same threshold, inference on larger blocks takes less time than on smaller blocks and it also gives better accuracy.

Compared with the baseline models in the first rows of Table 1, the most accurate one from our models achieves 76.4% for ICNet+PSPNet with the average inference time of 571ms which is about one third of the inference time from the baseline PSPNet model. Fig. 4 illustrates our selection results from ICNet and PSPNet on the Cityscapes validation set. It can be seen that our model improves the *train*, *sidewalk* and *pole* class compared to baseline ICNet. Compared with PSPNet, our result removes some misclassified *pole* class.

To further analyze the relationship between the threshold value θ and the total inference time, we estimate the average inference time with θ equals to 0.25 to 0.5. The result can be found in Fig. 6. From the scatter plot, there is a clear trend for each combination. As we increase the threshold, the inference time decreases linearly. Fig. 5 presents a similar scatter plot showing mean IoU versus inference time for different model combinations. From the plot, we can observe that our model selection results are located in between two baseline models. Depending on the application and the given target inference time, we can select the model with the best accuracy. This is very practical for autonomous driving scenarios where inference time is crucial. For example, if speed is preferred we can apply the model selection from 512×512 blocks with smaller threshold. In the case where the inference time is less important, we can select the overlapping blocks to boost the accuracy. The latter can be automatically selected based on the frame rates.

Table 3 illustrates class-wise quantitative results on the Cityscapes validation set. For E+P setting, our model has similar performance to the accurate model in 6 out of 19 classes. For I+P setting, ours outperforms PSPNet in *truck* class by five percentage points and have similar performance in another 5 out of 19 classes.

5 Conclusion

In this paper, we presented a model selection approach to find the balance between the speed of existing fast – but less accurate – networks, and the accuracy of the larger – yet slower – networks. The selection criterion is based on an uncertainty map that assigns the parts of the image to be processed by the faster or accurate networks. As future work, we plan to include more models and apply them in a cascaded way to further improve the segmentation accuracy vs. speed performance.

Table 2. Performance of PSPNet evaluated on non-overlapping/overlapping(*) sub-regions on Cityscapes validation set. The first and second column shows the block size and the corresponding performance in mIoU. The third and fourth columns present the average inference time with/without I/O in millisecond. (*) means with overlapping of two third of the previous block.

Block size	mIoU	avg. inf. time with I/O	without I/O
256×256	70.9%	26ms	24ms
480×480(*)	76.3%	62ms	55ms
512×512	74.6%	65ms	58ms
864×864(*)	77.6%	167ms	147ms
1024×1024	77.3%	223ms	196ms

Acknowledgments This work was supported by Toyota Motor Europe.

References

[1] A. Krizhevsky et al.: "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012

[2] H. Zhao et al.: "Pyramid Scene Parsing Network," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017

[3] H. Zhao et al.: "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," *The European Conference on Computer Vision*, 2018

[4] A. Paszke et al.: "ENet: A Deep Neural Network Architecture for Real-Time Semantic," *CoRR*, abs/1606.02147, 2016

[5] H. Wang et al.: "Gated convolutional neural network for semantic segmentation in high-resolution images," *Remote Sensing*, vol.9, no.5, pp.446, 2017

[6] Z. Wu et al.: "Real-time Semantic Image Segmentation via Spatial Sparsity," *CoRR*, abs/1712.00213, 2017

[7] L.-C. Chen et al.: "Rethinking atrous convolution for semantic image segmentation," *CoRR*, abs/1706.05587, 2017

[8] L.-C. Chen et al.: "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol.40, no.4, pp.834, 2018

[9] M. Cordts et al.: "The Cityscapes Dataset for Semantic Urban Scene Understanding," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016

[10] K. He et al.: "Deep Residual Learning for Image Recog-

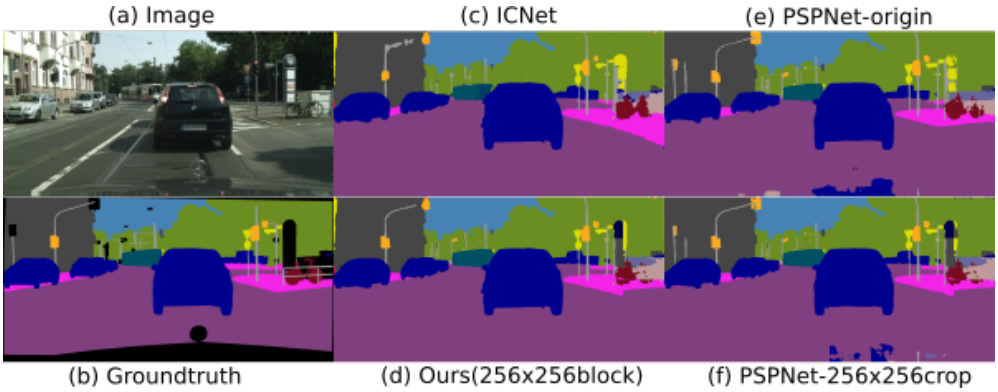

Figure 4. Visualization results on the Cityscapes validation set. (a) input image, (b) ground truth label, (c) segmentation result from ICNet, (d) our model selection result with block size of 256×256. (e) result from PSPNet baseline. (f) result from PSPNet 256×256 non-overlapping crop inference.

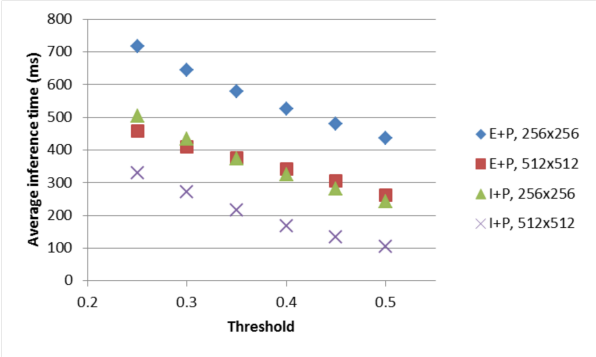

Figure 6. Scatter plot of the threshold (θ) applied on top of the entropy map estimated from both ENet and ICNet vs. mean IoU.

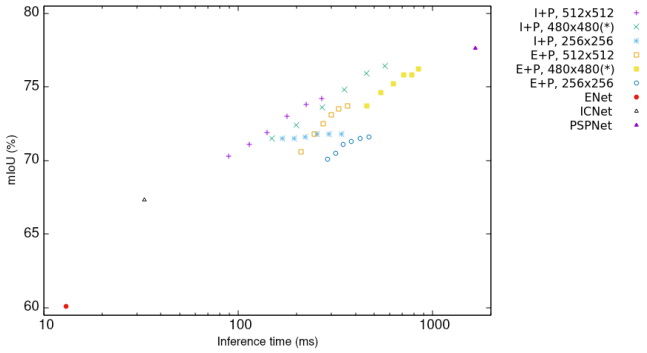

Figure 5. Scatter plot of the mIoU(%) vs. average inference time in millisecond.

nition," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016

Table 3. Quantitative results on the Cityscapes validation set.

Method	road	sidewalk	building	wall	fence	pole	tr.light	tr.sign	vege.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
ENet [4]	94.4	71.2	85.2	46.1	44.2	45.1	43.2	53.7	87.8	52.4	89.6	61.1	42.2	87.8	42.7	60.5	46.0	29.0	59.5	60.1
ICNet [3]	97.4	79.5	89.5	49.2	52.3	46.4	48.3	61.1	90.4	58.6	93.5	70.0	43.5	91.4	64.4	75.3	58.7	43.8	65.3	67.3
PSPNet [2]	98.1	84.9	92.4	58.7	59.2	64.3	70.5	78.3	92.5	65.8	94.3	82.0	62.7	94.9	64.5	86.8	79.8	67.7	77.4	77.6
(E+P)																				
$oracle_{block512}$	98.2	85.8	92.6	62.3	62.0	64.1	70.4	78.2	92.7	63.1	94.4	81.7	62.9	95.1	70.2	87.3	79.8	67.6	77.1	78.4
$oracle_{block256}$	98.3	86.3	92.9	63.3	62.7	64.5	70.3	78.4	92.8	69.0	94.5	82.0	64.3	95.1	71.1	87.2	79.4	67.5	77.2	78.8
Ours	97.5	83.5	91.9	52.2	58.8	63.0	70.3	77.9	92.3	62.7	94.2	81.1	59.8	94.1	61.4	86.2	76.2	67.0	77.3	76.2
(I+P)																				
$oracle_{block512}$	98.4	86.7	92.8	62.6	63.2	64.1	70.4	77.8	92.8	63.3	94.8	81.6	63.0	94.9	74.8	89.0	79.8	67.9	77.1	78.9
$oracle_{block256}$	98.5	87.1	93.0	64.9	64.5	64.4	70.5	78.3	92.9	69.0	95.0	82.0	63.6	95.1	76.6	88.6	79.9	67.9	77.3	79.4
Ours	97.7	82.7	91.7	53.3	59.2	61.5	69.2	75.9	92.3	63.1	94.2	80.3	59.8	94.1	69.8	86.7	76.7	66.8	76.6	76.4