Split-Merge Pooling

Omid Hosseini Jafari and Carsten Rother
Visual Learning Lab
Heidelberg University (HCI/IWR)
http://vislearn.de

Abstract

There are a variety of approaches to obtain a vast receptive field with convolutional neural networks (CNNs), such as pooling or striding convolutions. Most of these approaches were initially designed for image classification and later adapted to dense prediction tasks, such as semantic segmentation. However, the major drawback of this adaptation is the loss of spatial information. Even the popular dilated convolution approach, which in theory is able to operate with full spatial resolution, needs to subsample features for large image sizes in order to make the training and inference tractable. In this work, we introduce Split-Merge pooling to fully preserve the spatial information without any subsampling. By applying Split-Merge pooling to deep networks, we achieve, at the same time, a very large receptive field. We evaluate our approach for dense semantic segmentation of large image sizes taken from the Cityscapes and GTA-5 datasets. We demonstrate that by replacing max-pooling and striding convolutions with our split-merge pooling, we are able to improve the accuracy of different variations of ResNet significantly.

1. Introduction

Convolutional neural networks (CNNs) are the method of choice for image classification [14, 11, 18, 8, 4]. CNNs capture multi-scale contextual information in an image via subsampling the intermediate features through the network layers [14, 17]. This approach is successful due to the expansion of the receptive fields, which are large enough to capture the context of the image for classification.

In this work, we consider dense prediction tasks, which are popular in computer vision [23]. One desideratum of dense prediction tasks is to have pixel-accurate predictions, for example in semantic segmentation [8, 12, 13, 16, 7] or depth estimation [6, 5]. Even small inaccuracies, such as missing a small object lying on the street, may lead to an accident of an autonomous vehicle.

Most state-of-the-art approaches for dense prediction adapt existing CNNs which were originally designed for image classification. However, these adapted CNNs lose a vast amount of spatial information due to subsampling. As a result this reduces the prediction performance, mostly because of missing small objects or predicting coarse and inaccurate object boundaries. In order to mitigate this problem, other methods are proposed, such as progressive upsampling [6, 5, 12], deconvolution (or transpose convolutions) [13], skip connections [16, 7, 1], utilizing multiple scales of features [9, 24], and attention mechanism [25]. Another line of work preserves the resolution by replacing subsampling layers with dilated convolution layers [2, 21, 22] and widely used in other approaches [24, 25, 19, 20]. In theory, it is possible to design a dilated convolution network without using any subsampling operation to obtain an output with the same size as the input. Unfortunately, for large input sizes the training and inference of the resulting CNN is extremely slow or even sometimes intractable due to limited amount of memory on a GPU, hence subsampling is still needed in practice.

In this work, we propose novel pooling layers called Split-Merge Pooling (SMP). The split pooling layer splits (re-arranges) a feature tensor along its spatial dimension and treats the resulting splits as individual batches (see Fig. 1). The split pooling reduces the spatial size with a fixed scaling factor, related to the size of the non-overlapping pooling window. The merge layer acts as the reverse of split pooling operation, i.e., it receives the split batches and re-arranges the elements of the batches to their original locations. In our experiments, we replace all subsampling layers of standard ResNet networks by our split pooling. Finally, we merge the resulting split batches using merge pooling layers to output full resolution prediction. In this way, we do not lose any spatial information since it is preserved in the split batches. The batches are processed independently after the split, which has the major advantage that after the split each batch can be processed on a different GPU. This enables us to perform dense prediction tasks with very large networks on large images, while always preserving the spatial information. Furthermore, to

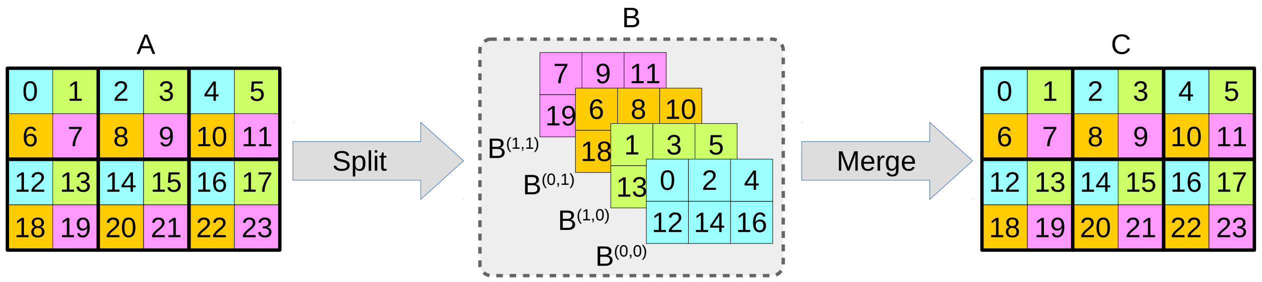

Figure 1. **Split and Merge Pooling.** The illustration of split and merge pooling layers with a window size of 2×2. The advantage of splitting the input into batches is to make it possible to process each part of the input (*i.e.* each split batch) independently. In this example, after splitting A into batches in B, we can send each batch $B^{(i,j)}$ to a different GPU and continue the forward pass from this point onward on multiple GPUs, or process one batch at a time on a single GPU. This enables us to execute dense prediction tasks with very deep networks and for large images, while always preserving the spatial information.

increase training speed, we introduced the Shrink and Expand pooling layers as a batch-subsampling of the split and merge pooling.
In summary, our contributions are as follows:

- We propose a novel pooling method called Split-Merge Pooling (SMP) which enables the unique mapping of each input element to one output element, without losing any spatial information.
- We propose a batch-subsampling variant of SMP, termed Shrink-Expand Pooling, to make the training efficient and tractable for very deep networks.
- To show the effectiveness of SMP on dense prediction tasks, we choose semantic segmentation and apply SMP to ResNet networks with varying depths. We chose ResNet as our baseline since it is used as the backbone in state-of-the-art approaches. For semantic segmentation of large images from Cityscapes [3] and GTA-5 [15] datasets, our SMP networks outperform the corresponding ResNet networks by a significant margin, with up to 6.8% in IoU score.
- We even observe that a SMP version of a shallow ResNet (ResNet18) outperforms the original ResNet101 by 2.8% in IoU score, although ResNet101 is 3.8 times deeper than ResNet18.

2. Related Work

Utilizing CNNs for image classification became very popular with the introduction of the Imagenet challenge [4] and after some popular network architectures such as Alexnet [11], VGG [18] and ResNet [8] emerged. Pooling layers in CNNs were originally introduced for image classification tasks on MNIST dataset [14]. The aim of pooling layers is to summarize the information over a spatial neighborhood.

Dense prediction tasks. Most of the dense prediction models are based on adapted versions of image classification networks. Eigen *et al.* [6, 5] adapt the Alexnet [11] for single image depth estimation. Since the output of such networks is very coarse, they upsample the output progressively and combine it with local features from the input image. Long *et al.* [12] introduce the first fully covolutional network (FCN) for semantic image segmentation by adapting VGG network [18] for this task. They map intermediate features with higher resolution to label space and combine them with the coarse prediction of the network in order to recover the missing spatial information. Noh *et al.* [13] introduce more parameters in a decoder consisting of transpose convolutions (Deconvolution layers) to upsample the coarse output of the encoder. However, it is still difficult to recover the missing information from the coarse output with this approach. Furthermore, other approaches [16, 7, 1] use skip connections between encoder and decoder to obtain more detailed information from lower level features.

Yu *et al.* introduced the concept of dilated convolutions [21] and later developed the dilated residual network (DRN) [22], which uses dilated convolutions to preserve the spatial information. However, they still use three subsampeling layers to reduce the spatial resolution to make training feasible, i.e. to fit the model into memory. Therefore, DRNs lose spatial information and need upsampling to obtain the full size output prediction. In contrast, our approach does not lose any spatial information. The batch-based design of our split pooling gives us the flexibility of distributed processing of batches on multiple GPUs or sequential processing of batches on one GPU during inference time (see Sec 3.1). Furthermore, for training, we learn our network weights using only one subsampled split batch to make the training faster and tractable (see Sec 3.2).

3. Method

The goal of this work is to design a set of pooling layers that preserve spatial information, in contrast to subsampling

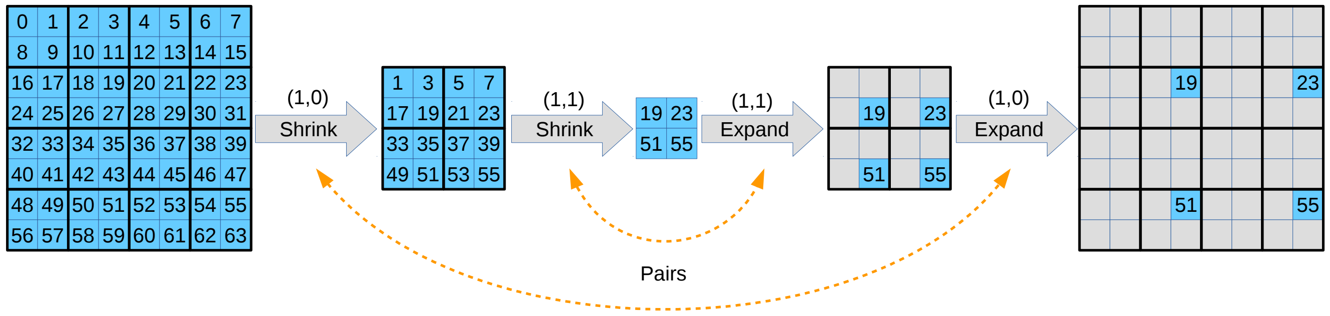

Figure 2. **Shrink and Expand Pooling example.** The illustration of two pairs of shrink and expand pooling layers with window size of 2×2 and sampling location of $(1, 0)$ and $(1, 1)$. During training the expand pooling layers back-propagate the error of sparse elements.

operations such as max-pooling and striding convolutions. Additionally, the new pooling layers have to be applicable to very deep networks. Losing spatial information caused by subsampling is a prevalent problem for dense prediction tasks, such as object detection, semantic segmentation, instance segmentation, or depth estimation. Small objects in the input image can get lost completely with subsampling. Moreover, recovering precise object boundaries in an image becomes challenging due to missing spatial information. These problems, caused by losing spatial information, obviously reduced prediction accuracy. However, simply storing all the information with the original spatial resolution is not a practical option since it causes storage issues, in particular for training on large image datasets.

Our idea is to preserve all the spatial information by splitting the feature tensor into multiple downsampled batches (see Fig. 1) instead of preserving it inside the original spatial resolution. In this way, we have both advantages of large receptive fields due to standard downsampling, as well as keeping all the available spatial information for obtaining precise dense predictions.

3.1. Split-Merge Pooling

The split pooling layer splits an input feature tensor along spatial dimensions and outputs each split as a batch. The number of output batches depends on the window size of split pooling, *e.g.* a split pooling with a window size of 2×2 splits the input into 4 batches (see Fig. 1). The purpose of the split pooling is to downsample the input while preserving the whole information. Hence, the pooling window covers each element of the input once, without any gap or overlap.

The merge pooling layer acts as the inverse of split pooling, i.e., it takes the split batches and merges them into one batch. For dense prediction tasks, we apply the same number of merge pooling layers on the final output of network as the number of split pooling layers in a network. The result will be a one-to-one mapping between the input and output pixels of the network.

Formally, given an input $A_{W \times H}$, a split pooling with window size of $w \times h$ splits A into $w * h$ batches

$$\mathcal{B} = \{\, B^{(k,l)} \mid 0 \le k < w \wedge 0 \le l < h \,\}.$$

The elements of A are assigned to batch $B^{(k,l)}$ as follows

$$b_{i,j}^{(k,l)} = a_{i*w+k,j*h+l} \tag{1}$$

for all $0 \le i < W/w$ and $0 \le j < H/h$. The spatial size of each batch will be $W/w \times H/h$. If the input to the split pooling consist of multiple batches, the split pooling splits each input batch separately and returns all resulting splits as a set of batches.

The merge pooling performs the inverse of Eq. 1, *i.e.*, if $\mathcal{B}$ is the input and A is the output, the elements of $\mathcal{B}$ are assigned to A as follows

$$a_{i*w+k,j*h+l} = b_{i,j}^{(k,l)} \tag{2}$$

for all $0 \le k < w$, $0 \le l < h$, $0 \le i < W/w$ and $0 \le j < H/h$.

Although preserving the spatial information is beneficial, it increases the number of elements to store during the forward pass.

The advantage of splitting the inputs into batches is that we can distribute the computation of batches on multiple GPUs or processing them sequentially on one GPU. This is ideal for inference. However, in contrast to inference, training phase additionally requires to compute and store the gradients. This makes the training of very deep networks intractable for large batches of inputs with high resolution images. We handle this issue by introducing Shrink-Expand pooling layers.

3.2. Shrink-Expand Pooling

The batch-based design of the split pooling layer makes the forward process of each part (split batch) of the input independent of the other parts (split batches). Furthermore, the one-to-one mapping between the elements of input and output gives a clear path between these elements in both forward and backward direction. Giving these two properties, it is possible to train a network using a subset of split batches produced by each split pooling layer. If we reduce the size of the batch subset to one, the space complexity will be the same as the space complexity of the max pooling and striding convolutions. Also, the training time complexity stays the same.

Giving this reasoning, we introduce the shrink and expand pooling layers, which are the batch-subsampled versions of the split and merge pooling layers. The shrink pooling samples one element at a fixed location within the pooling window, and the corresponding expand pooling uses the same fixed location to perform the reverse of the shrink pooling (see Fig. 2). Hence, the output of expand pooling is sparse. In other words, the shrink pooling is the same as split pooling except it samples only one of the split batches and returns it.

The sampling location (i, j) is set randomly in each forward pass during training to avoid overfitting to part of the training data. For each pair of shrink and expand pooling in the network, we sample only two numbers (i, j). Fig. 2 shows an example of using a sequence of shrink and expand pooling layers. During training, the expand layers only backpropagate the error of sparse valid elements.

4. Experiments

We evaluate the effectiveness of our approach for the semantic segmentation task. To examine the impact of our pooling layer, we modify the ResNet and then compare the performance of the modified version and the original one.

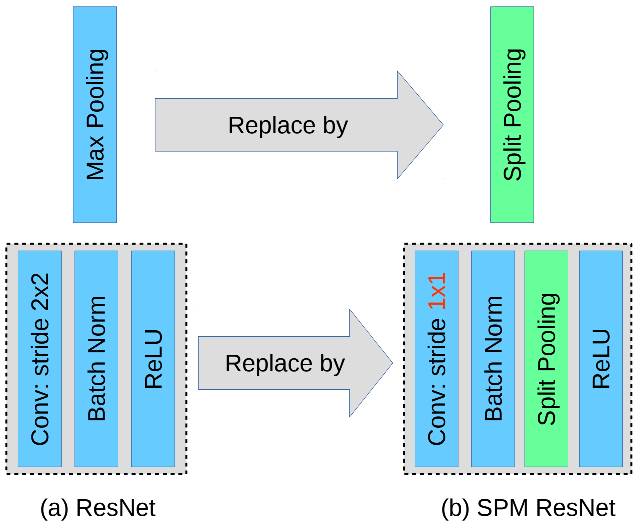

Figure 3. **Applying split pooling to ResNet.** For applying split pooling to ResNet, we replace the max pooling layer with split pooling (top). Furthermore, we add a split pooling layer after batch normalization layer of convolutional blocks with the stride of 2×2 and set the stride to 1×1 (bottom).

4.1. Experimental Setup

Baseline FCN32s. Our experimental models are based on FCN32s [12] with a variant of ResNet [8] as backbone. We adapt the ResNet for semantic segmentation task by removing the average pooling and fully connected layer and replacing them by a 1×1 convolutional layer which maps the output channels of last layer (layer4) to the number of semantic classes. We refer to this model as FCN32s and use it as baseline. FCN32s predictions are 32 times smaller than the input image to the network; thus, the coarse predictions are upsampled to full resolution using bilinear interpolation.

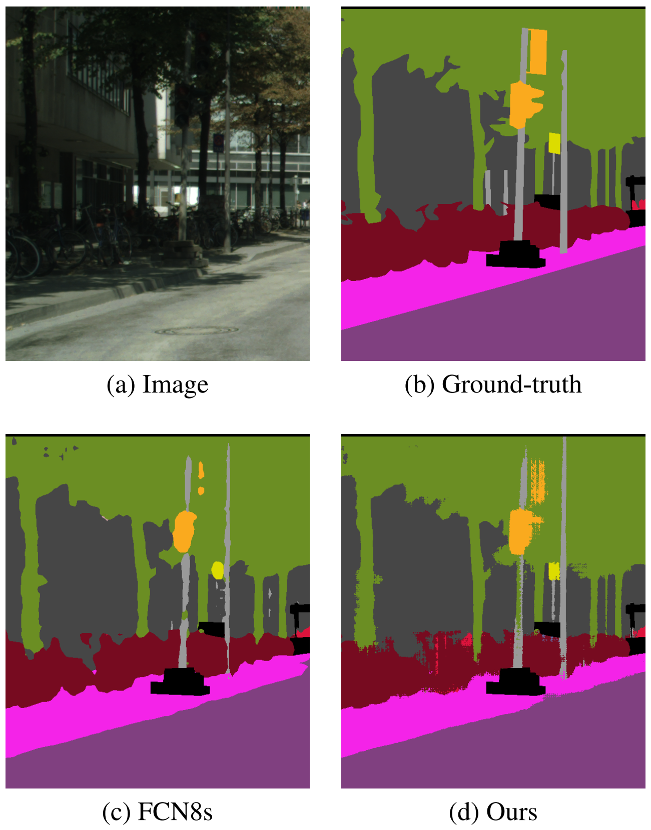

Figure 4. **An example of a slightly inaccurate annotation in Cityscapes dataset.** Although the trunk of the tree in the image (a) is visible through bicycles, it is labeled as bicycle in ground truth (b). Our method can obtain detailed boundaries (d) while FCN8s with max pooling cannot (c). Hence, to fully validate the full potential of our method we need a pixel-accurate ground-truth, such as GTA-5.

SMP-{18, 34, 101}. As illustrated in Fig. 3, in order to apply SMP to ResNet backbone of FCN32s, we simply replace the max pooling layer with a split pooling layer and add a split pooling layer after the batch normalization layer of convolutional blocks (Conv-BN-ReLU) with a stride of 2×2 and set their strides to 1×1. In our experiments, the window size of split pooling layers is 2×2. We call the resulting model, according to the type of backbone ResNet, SMP-18, SMP-34 and SMP-101. The output of SMP-X consist of 1024 batches, since we always have 5 SMP layers each giving 4 batches. We merge the output split batches by performing merge pooling five times. The final result has the same resolution as the input. During training phase, we replace split and merge pooling layer by shrink and expand layers.

FCN8s. Furthermore, we compare our models to the FCN8s model with original ResNet backbone. The main difference between FCN32s and FCN8s is that FCN8s has two extra 1×1 convolution layers to map the features channel of layer2 and layer3 outputs of ResNet to the number of semantic classes. Then, the output of the network and these new convolutions are resized to the same size and are added together. The final output of FCN8s is 8 times smaller than the input image and should be upsampled to full resolution using bilinear interpolation.

4.2. Implementation Details

Data Augmentation. For all the models we used random crop of size 512×512 and horizontal flip data augmentation. We used a fixed seed for data augmentation for all the experiments.

Training. We initialize the ResNet backbones with pre-trained models from Imagenet. We optimize the parameters using Adam solver [10] with learning rate of $1e-5$ and weight decay of $5e-4$. We use the batch size of 10 for all the experiments.

	Backbone	Pooling	IoU	Size
FCN32s	ResNet34	MP	64.5	21.29M
FCN8s	ResNet34	MP	67.8	21.30M
SMP-34 (ours)	ResNet34	SMP	**68.8**	21.29M
FCN32s	ResNet101	MP	65.5	42.54M
FCN8s	ResNet101	MP	69.1	42.56M
SMP-101 (ours)	ResNet101	SMP	**69.2**	42.54M

Table 1. **Cityscapes quantitative results.** Please note that the architecture of SMP-X networks is the same as the FCN32s. The FCN8s architecture has two extra 1×1 convolutions.

4.3. Cityscapes

The Cityscapes dataset [3] consists of images with the size of 2048×1024. The images are annotated with 19 semantic classes. We evaluate the performance of our SMP-34 and SMP-101 on Cityascapes validation set and compare it with FCN32s and FCN8s with ResNet34 and ResNet101 backbone networks. The full size images are used for evaluation, *i.e.* without downsampling or cropping.

	Backbone	Pooling	IoU	Size
FCN32s	ResNet18	MP	71.1	11.186M
FCN8s	ResNet18	MP	74.2	11.193M
SMP-18 (ours)	ResNet18	SMP	**77.5**	11.186M
FCN32s	ResNet34	MP	73.1	21.29M
FCN8s	ResNet34	MP	76.7	21.30M
SMP-34 (ours)	ResNet34	SMP	**80.2**	21.29M
FCN32s	ResNet101	MP	74.6	42.54M
FCN8s	ResNet101	MP	76.7	42.56M
SMP-101 (ours)	ResNet101	SMP	**80.3**	42.54M

Table 2. **Quantitative results for GTA-5.** Please note that the architecture of SMP-X networks is the same as the FCN32s, while the FCN8s has two extra 1×1 convolutions.

	pole	traffic light	traffic sign	person	rider	motorcycle	bicycle	IoU
FCN32s-Res34	35.4	51.5	63.0	70.4	50.3	51.8	68.4	55.8
FCN8s-Res34	53.5	57.6	69.9	77.1	53.6	51.0	72.2	62.1
SMP-34(ours)	**60.5**	**63.7**	**74.5**	**78.8**	**54.1**	**52.9**	**73.9**	**65.5**
FCN32s-Res101	39.2	58.1	66.9	71.9	51.4	53.9	70.6	58.9
FCN8s-Res101	56.1	63.1	72.2	78.2	55.4	52.8	74.7	64.6
SMP-101(ours)	**63.3**	**68.7**	**75.8**	**79.9**	**56.1**	**52.9**	**76.6**	**67.6**

Table 3. **Performance on Cityscapes for small and thin objects.**

	pole	traffic light	traffic sign	person	rider	motorcycle	bicycle	IoU
FCN32s-Res18	44.6	45.3	60.7	69.3	64.5	59.8	38.5	54.7
FCN8s-Res18	54.0	52.7	63.8	74.4	**70.4**	66.5	40.3	60.3
SMP-18(ours)	**72.7**	**69.2**	**73.6**	**76.8**	67.8	**69.3**	**46.5**	**68.0**
FCN32s-Res34	46.0	47.7	63.0	70.0	67.9	63.9	49.5	58.3
FCN8s-Res34	55.6	57.2	68.3	76.3	**74.0**	65.8	51.3	64.1
SMP-34(ours)	**74.3**	**71.4**	**73.7**	**81.0**	70.5	**73.7**	**58.5**	**71.9**
FCN32s-Res101	47.3	50.3	68.7	70.3	63.9	66.2	58.1	60.7
FCN8s-Res101	57.1	59.0	70.8	75.5	67.1	**70.6**	62.4	66.1
SMP-101(ours)	**76.8**	**74.9**	**78.9**	**79.0**	**69.2**	69.6	**67.8**	**73.7**

Table 4. **Performance on GTA-5 for small and thin objects**

Table 1 summarizes the comparison of different methods with respect to their setups, performances and number of parameters (size). Each SMP model outperforms the corresponding FCN32s and FCN8s with the same ResNet backbone significantly. SMP-34 even outperforms FCN32s-ResNet101 by 3.3%, although the latter is almost 2 times larger. The performance of our approach on objects from small and thin classes is reported in Table 6. As we can see in the table, SMP models outperforms their corresponding original models on small and thin objects. Both SMP mod-

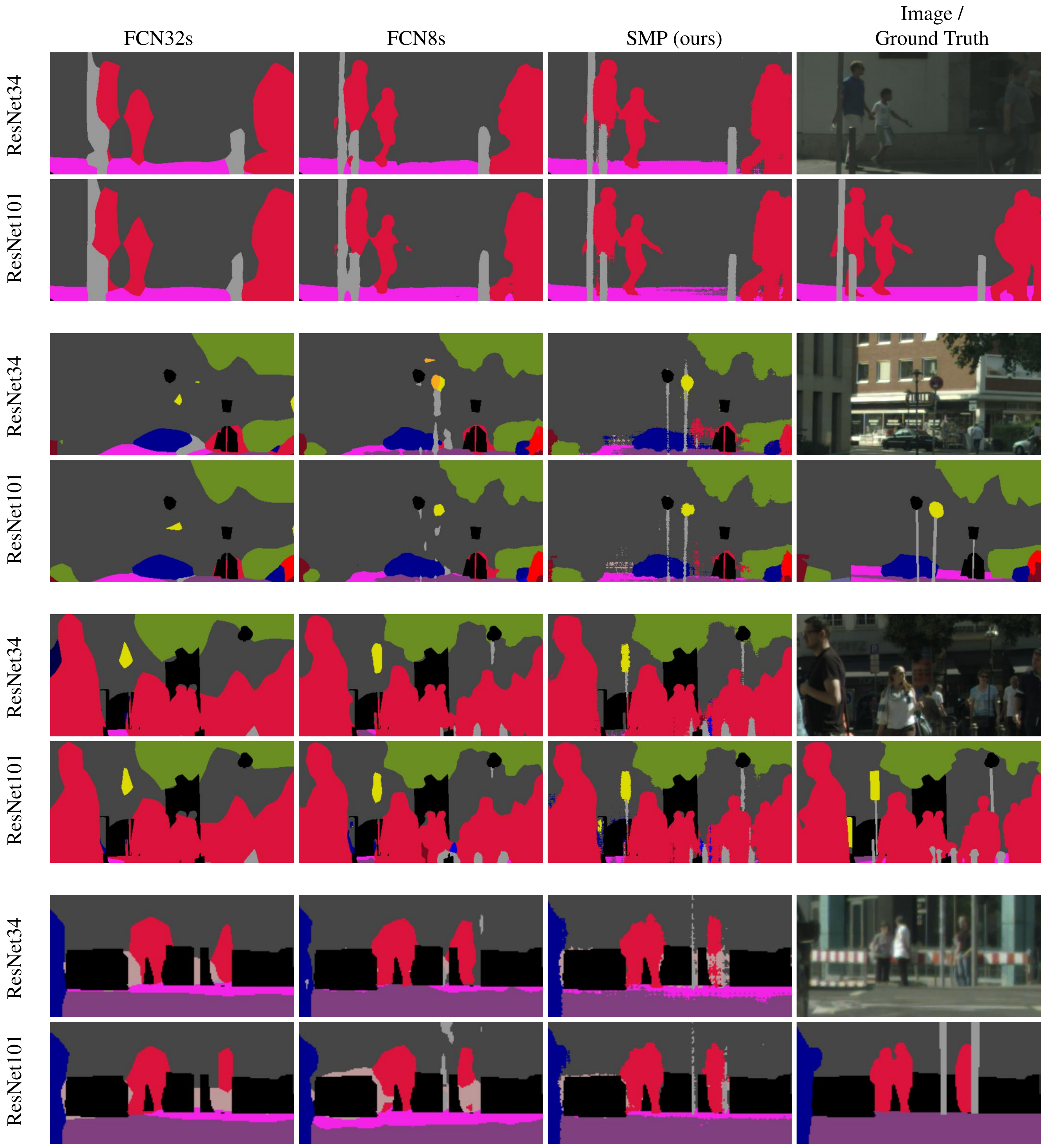

Figure 5. **Cityscapes qualitative results.** In each block, the top row is related to models with ResNet34 backbone and the bottom row to ResNet101. The last column of each block shows the input image (top) and the ground-truth (bottom). To enhance the visual comparison of the results, we have cropped the output labelling. Further results, also in full resolution, can be found in supplementary material.

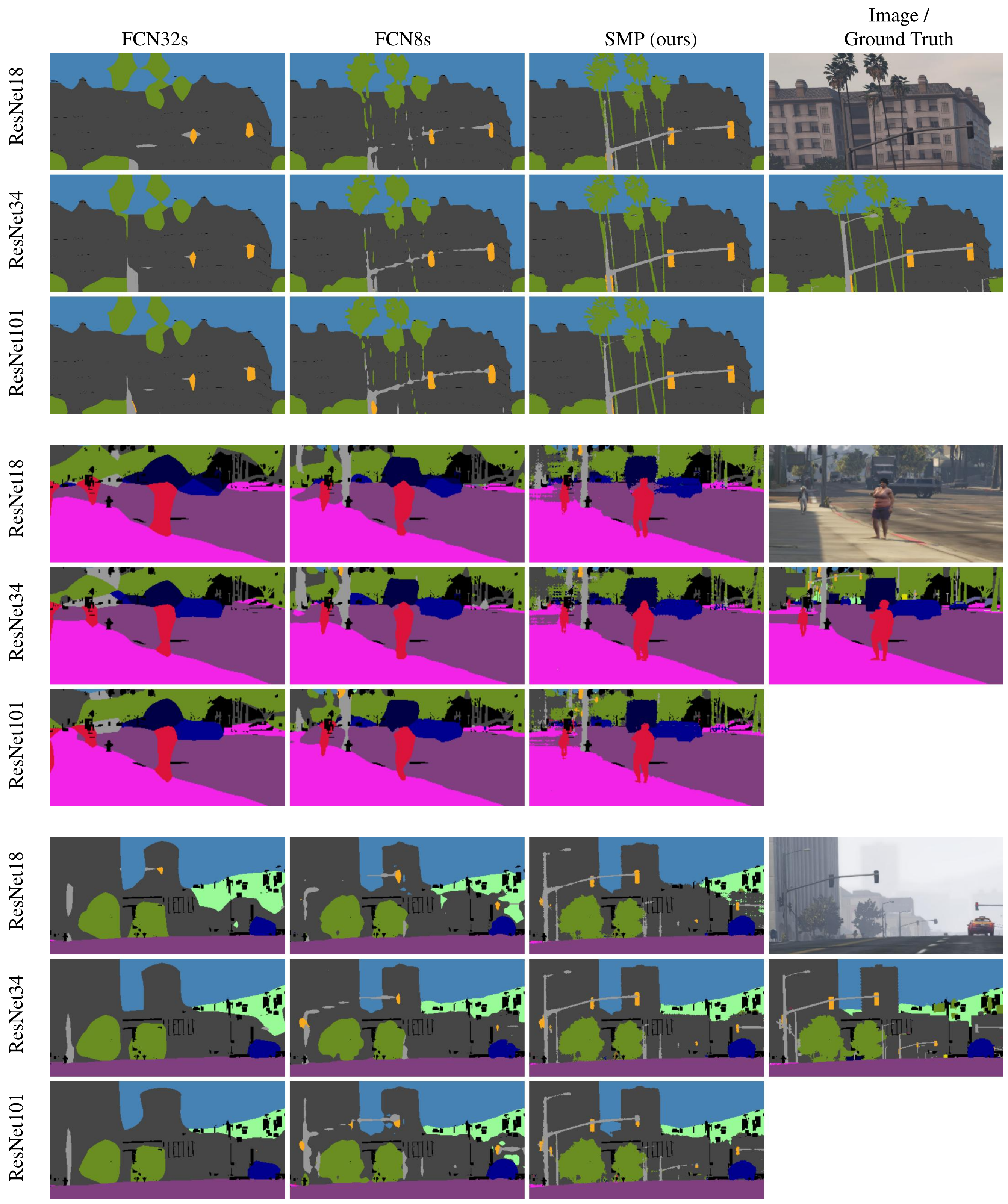

Figure 6. **GTA-5 qualitative results.** In each block, the top row is related to ResNet18 backbone, middle row to ResNet34, and bottom row to ResNet101. The last column of each block shows the input image (top) and the ground-truth (bottom). To enhance the visual comparison of the results, we have cropped the output labelling. Further results, also in full resolution, can be found in supplementary material.

els outperform their corresponding FCN32s models for *pole* by 24%, for *traffic light* by more than 10%, for *traffic sign* by more than 9%, and for *person* by 8%. As it is shown in Fig. 5, the improvement of these classes can be noticed visually as well. The performance for the remaining objects is mostly better, or sometimes slightly worse.

Qualitative results are shown in Fig. 5. For the sake of improved visibility we have cropped the results. The full size output images can be found in supplementary material.

4.4. GTA-5

Cityscapes is one of the most accurately annotated semantic segmentation datasets, however it is still not pixel-accurate (see Fig. 4). Obtaining pixel-accurate annotations from real data is extremely challenging and expensive. Therefore, for analysing the full potential of our method, we evaluate our method on GTA-5 [15], which is a synthetic dataset with the same semantic classes as Cityscapes. Since GTA-5 is a synthetic, the annotations are pixel-accurate and ideal for our purpose. The GTA-5 dataset [15] consist of 24,999 realistic synthetic images with pixel-accurate semantic annotations. We randomly select 500 images as validation set, which we did not use for training.

Table 1 summarizes the comparison of different methods with respect to their setups, performances and number of parameters (size). As we can see, due to the pixel-accurate annotations of GTA-5 dataset, the improvement of our proposed models, over their baselines, is more significant compared to Cityscapes. Each SMP model outperforms the corresponding FCN32s and FCN8s with the same ResNet backbone significantly. Particularly, our SMP-18 even outperforms its FCN32s-ResNet101 and FCN8s-ResNet101 counterparts, although it has 4 times fewer parameters. The performance of our approach on objects from small and thin classes is reported in Table 7. As we can see, similarly to Cityscapes, SMP models outperforms their corresponding original models on small and thin objects. Compared to FCN32s models, our corresponding SMP models improve the categories *pole* by more than 28%, *traffic light* by more than 23%, *traffic sign* by more than 10%, and *person* by more than 7%. The improvement over these classes is also visually significant (see Fig. 6).

4.5. Run-time Analysis

For analyzing the time complexity of the Split-Merge pooling, we designed small networks to just focus on the proposed pooling layers instead of analyzing the time complexity of them on a particular task with specific network architecture. As it is shown in Fig. 7, we consider three networks with (a) max pooling, (b) dilated convolution, and (c) Split pooling. We choose to compare our proposed pooling setup (c) with dilated convolutions (b) due to the success of the dilated convolution in dense prediction tasks. Almost all state-of-the-art approaches in dense prediction tasks (such as semantic segmentation, depth estimation, optical flow estimation) are using dilated convolutions in their architectures to achieve a detailed output.

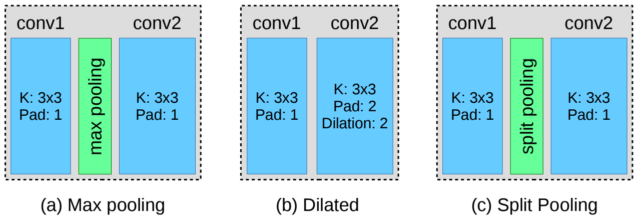

Figure 7. Architectures used for runtime analysis. *conv2* in (c) is identical to *conv2* in (a) while *conv2* in (c) is dilated convolution with padding 2.

In Table 5, we show the Giga floating-point operation (GFLOP) of each component of each setup for an input tensor of size $1 \times 3 \times 256 \times 256$. As we can see, the dilated convolution setup and split pooling setup have the same GFLOPs which means dilated convolution layers can be replaced with our proposed pooling layers in an arbitrary architecture without changing the complexity of the network. However, our split pooling layer has two advantages:

1. faster training time using shrink-expand layers
2. faster inference time by parallelizing the forward-computation of split layer output batches (in this example setup computation of *conv2*, see Table 5)

	batches	conv1	pooling	batches	conv2	total
(a) Max Pooling	1	0.23	0	1	2.42	2.65
(b) Dilated Conv.	1	0.23	-	1	9.68	9.92
(c) Split Pooling	1	0.23	0	4	9.68	9.92

Table 5. GFLOPs of the models calculated on the input size of $1 \times 3 \times 256 \times 256$. Note that the number of batches are increased after split pooling.

5. Conclusion

We proposed a novel pooling method SMP with the goal of preserving the spatial information throughout the entire network. SMP can be used instead of any subsampling operations in a network architecture. We show that by replacing subsampling operations with SMP in ResNet, we achieved two important properties for any dense prediction task at the same time: i) the network has a large receptive field, ii) the network provides a unique mapping from input pixels to output pixels. Furthermore, the computation of a network with SMP can be distributed to multiple GPUs due to batch-based design of SMP. We show experimentally that the resulting network outperforms the original one significantly.

Acknowledgements This work is funded by the DFG grant COVMAP: Intelligente Karten mittels gemeinsamer GPS- und Videodatenanalyse (RO 4804/2-1).

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *TPAMI*, 2017. 1, 2

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2

[5] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, pages 2366–2374. 2014. 1, 2

[7] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1, 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 4

[9] Omid Hosseini Jafari, Oliver Groth, Alexander Kirillov, Michael Ying Yang, and Carsten Rother. Analyzing modular cnn architectures for joint depth prediction and semantic segmentation. In *ICRA*, 2017. 1

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, 2012. 1, 2

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 4

[13] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 1, 2

[14] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 1, 2

[15] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016. 2, 8

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 2

[17] Dominik Scherer, Andreas C. Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN*, pages 92–101, 2010. 1

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 1

[20] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *CoRR*, abs/1908.07919, 2019. 1

[21] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 1, 2

[22] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[23] Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 1

[24] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1

[25] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. PSANet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018. 1

Appendix A: Detailed Qualitative Results

Table 6 and Table 7 show the detailed quantitative results of our proposed models on Cityscapes and GTA-5 datasets respectively.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	IoU
FCN32s-Res34	97.2	78.5	88.5	40.1	48.5	35.4	51.5	63.0	88.9	56.5	89.3	70.4	50.3	91.2	48.3	64.9	43.5	51.8	68.4	64.5
FCN8s-Res34	97.3	80.0	89.6	37.2	49.5	53.5	57.6	69.9	90.9	57.7	92.4	77.1	53.6	92.5	**49.9**	69.4	**46.6**	51.0	72.2	67.8
SMP-34(ours)	**97.3**	**80.6**	**90.5**	**42.2**	**50.6**	**60.5**	**63.7**	**74.5**	**91.4**	**58.7**	**92.8**	**78.8**	**54.1**	**92.9**	48.5	**70.6**	32.3	**52.9**	**73.9**	**68.8**
FCN32s-Res101	97.3	79.6	88.9	38.3	51.3	39.2	58.1	66.9	89.4	55.4	91.3	71.9	51.4	91.8	43.0	65.1	41.3	53.9	70.6	65.5
FCN8s-Res101	97.5	81.2	90.3	**41.0**	**49.9**	56.1	63.1	72.2	91.4	59.8	92.8	78.2	55.4	92.9	**49.9**	**68.6**	**44.7**	52.8	74.7	69.1
SMP-101(ours)	**97.5**	**82.7**	**90.6**	39.6	48.4	**63.3**	**68.7**	**75.8**	**91.9**	**61.0**	**93.4**	**79.9**	**56.1**	**93.2**	41.5	61.2	40.0	**52.9**	**76.6**	**69.2**

Table 6. **Cityscapes - detailed.** SMP models outperforms their corresponding original models on small and thin objects. Both SMP models outperform their corresponding FCN32s models for *pole* by 24%, for *traffic light* by more than 10%, for *traffic sign* by more than 10%, and for *person* by 8%.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	IoU
FCN32s-Res18	95.4	81.8	87.3	62.9	54.8	44.6	45.3	60.7	79.9	70.0	93.3	69.3	64.5	88.3	**83.9**	87.8	82.6	59.8	38.5	71.1
FCN8s-Res18	96.1	84.6	88.6	66.6	**56.4**	54.0	52.7	63.8	83.3	72.5	94.8	74.4	**70.4**	89.5	80.7	**90.4**	85.1	66.5	40.3	74.2
SMP-18(ours)	**96.5**	**85.6**	**89.6**	**67.2**	55.7	**72.7**	**69.2**	**73.6**	**88.4**	**75.0**	**97.9**	**76.8**	67.8	**90.6**	82.3	80.8	**86.1**	**69.3**	**46.5**	**77.5**
FCN32s-Res34	96.5	85.2	88.0	64.2	55.4	46.0	47.7	63.0	80.9	72.4	93.5	70.0	67.9	89.2	86.2	85.1	84.2	63.9	49.5	73.1
FCN8s-Res34	97.0	87.6	89.5	68.9	**59.4**	55.6	57.2	68.3	83.9	74.4	94.9	76.3	**74.0**	90.5	86.8	**90.4**	84.9	65.8	51.3	76.7
SMP-34(ours)	**97.3**	**88.1**	**91.4**	**69.6**	58.3	**74.3**	**71.4**	**73.7**	**89.1**	**77.1**	**98.2**	**81.0**	70.5	**92.6**	**88.3**	83.7	**87.6**	**73.7**	**58.5**	**80.2**
FCN32s-Res101	96.6	86.0	89.2	70.4	59.3	47.3	50.3	68.7	81.6	72.6	93.7	70.3	63.9	89.6	88.2	**89.1**	77.1	66.2	58.1	74.6
FCN8s-Res101	96.6	86.6	89.2	61.7	60.6	57.1	59.0	70.8	84.5	73.9	95.1	75.5	67.1	90.0	**88.6**	86.0	81.9	**70.6**	62.4	76.7
SMP-101(ours)	**97.3**	**88.7**	**91.7**	**71.0**	**62.2**	**76.8**	**74.9**	**78.9**	**89.6**	**79.0**	**97.8**	**79.0**	**69.2**	**90.6**	86.0	71.9	**84.3**	69.6	**67.8**	**80.3**

Table 7. **GTA-5 - detailed.** SMP models outperforms their corresponding original models on small and thin objects for GTA-5 as well. Compare to FCN32s models, our corresponding SMP models improve the categories *pole* by more than 28%, *traffic light* by more than 23%, *traffic sign* by more than 10%, and *person* by more than 7%.