Not All Areas Are Equal: Transfer Learning for Semantic Segmentation via Hierarchical Region Selection (Supplementary Material)

Ruoqi Sun [1] Xinge Zhu [2] Chongruo Wu [3] Chen Huang [4] Jianping Shi [5] Lizhuang Ma [1]

[1]Shanghai Jiao Tong University [2]The Chinese University of Hong Kong
[3]University of California, Davis [4]Carnegie Mellon University [5]SenseTime Research

ruoqisun7@sjtu.edu.cn zx018@ie.cuhk.edu.hk crwu@ucdavis.edu
chenh2@andrew.cmu.edu shijianping@sensetime.com ma-lz@cs.sjtu.edu.cn

1. Ablation Studies

Image selection vs. pixel weighting. Here we compare our hierarchical pixel/region/image weighting method with an intuitive baseline: to select some source images with synthetic road and directly add them to our training set. Such holistic image selection scheme is commonly adopted in many vision tasks. Table. 1 shows that such image-level selection is inferior in performance to our method (both shared and multi-channel schemes with W^1 and W^{19}), which benefits from adaptive and arbitrary region selection in a soft weighting manner. Note in our image selection baseline, we filter out most of those non-road pixels leaving more road pixels in an image. Its variant that keeps those non-road pixels works even worse since their distribution in the synthetic source domain largely deviates from the target domain. And this is the exact motivation of our adaptive region selection method.

Pixel weighting on images vs. pixel weighting on predictions In Table. 1, we compare our method with another baseline: to set the images as the inputs of the weighting networks, which is different from our setting that utilizes the predictions as the input data. The experiment shows that our setting has higher performance. The predictions are better than images in encouraging the segmentation network to predict the same for similarly structured regions, regardless of their texture difference from two data domains. This essentially robustifies segmentation to data variance across domains in a transfer learning framework.

*The first two authors contributed equally to this paper.

Method	Base	Backbone	Setting	M IoU
Swami *et al.* [5]	FCN	VGG16	Un-	37.1%
CL [8]	FCN	VGG16	Un-	38.1%
ROAD [1]	FCN	VGG16	Un-	35.9%
Baseline1*	FCN	VGG16	-	65.3%
+GAN	FCN	VGG16	Joint-	64.0%
+GAN+ImageSelect	FCN	VGG16	Joint-	65.4%
+GAN+W^1(Image)	FCN	VGG16	Joint-	67.1%
Ours with W^1	FCN	VGG16	Joint-	**67.6%**
Ours with W^{19}	FCN	VGG16	Joint-	**68.1%**
Baseline2*	PSPNet	ResNet50	-	76.1%
Ours with W^1	PSPNet	ResNet50	Joint-	**77.6%**

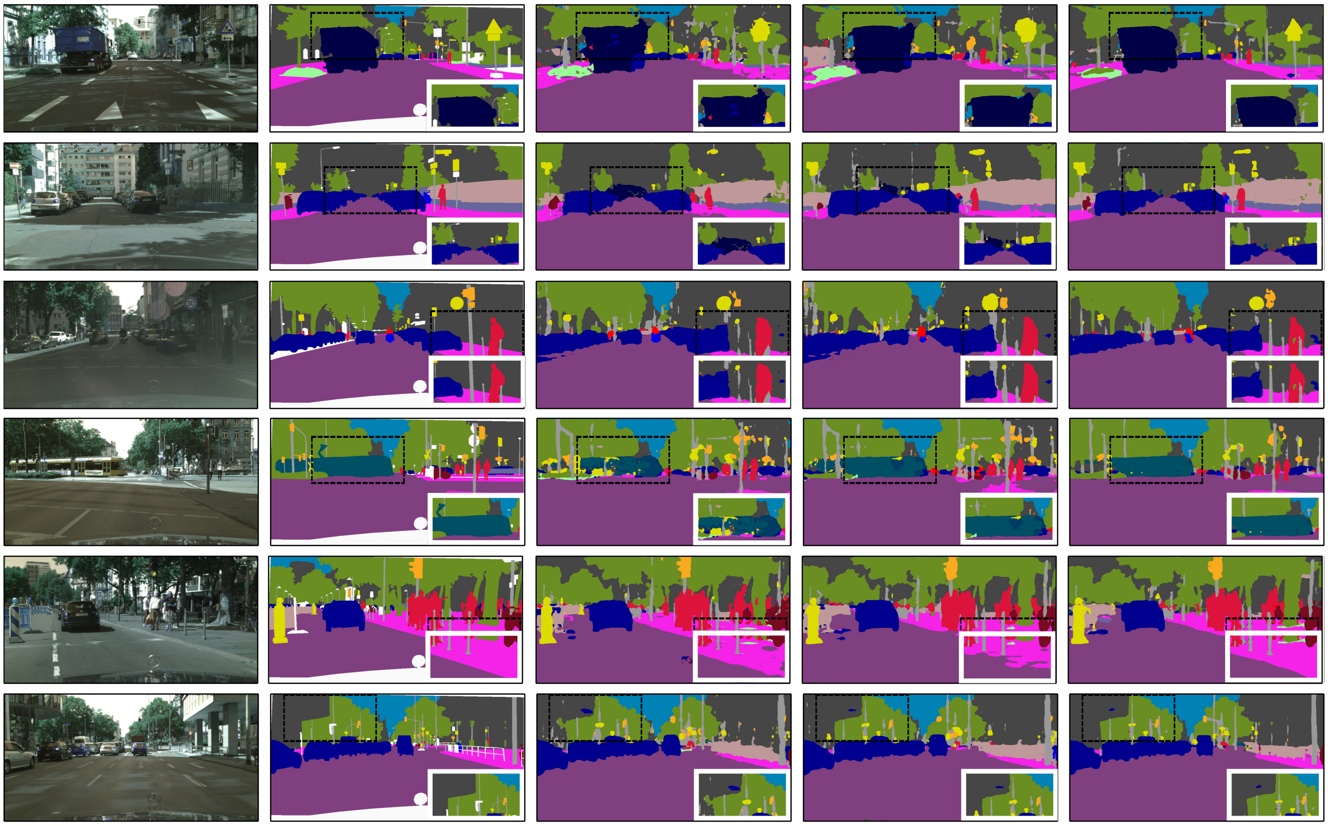
Table 1. Experimental results of transfer learning using GTAV and CITYSCAPES (GTAV + CITYSCAPES $\rightarrow$ CITYSCAPES). W^1 and W^{19} denote our shared and multi-channel weighting schemes, respectively. * denotes the model is trained on CITYSCAPES dataset only, without any source datasets.

2. Stronger Baseline

We replace FCN with the more recent segmentation network PSPNet (using ResNet50 backbone). Table. 1 shows that our method still outperforms the baseline and achieves state-of-the-art performance, which verifies the efficacy of our method.

3. More Visualizations

Fig. 1 provides more of our segmentation results on the target, real-world dataset CITYSCAPES [2]. One observation from the results is that our method can better preserve the object boundaries and details. We attribute this to our hierarchical weighting networks that can distill useful information from source domain to enrich the modelling ability for detailed textures.



Target Image | Ground Truth | FCN | FCN+GAN (GTAV) | Ours with W^{19} (GTAV)

Figure 1. Segmentation results on CITYSCAPES dataset. Note how our full model with W^{19} can preserve object details and boundaries.

4. Network Architecture

Our backbone network for segmentation is FCN [4] + VGG16 [6] and PSPNet [7] + ResNet50 [3]. The detailed architectures of hierarchical weighting networks (W_p^1, W_r^1, W_i^1, and W_p^{19}, W_r^{19}, W_i^{19}), generator G, and discriminator D are shown in Fig. 2.

References

[1] Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7892-7901*, 2018.

[2] 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 *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 3213–3223, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 770–778, 2016.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[5] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 2881–2890, 2017.

[8] Xinge Zhu, Hui Zhou, Ceyuan Yang, Jianping Shi, and Dahua Lin. Penalizing top performers: Conservative loss for semantic segmentation adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 568–583, 2018.

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	4	2	1
LeakyReLU					
Convolution	64	128	4	2	1
LeakyReLU					
Convolution	128	256	4	2	1
LeakyReLU					
Convolution	256	512	4	2	1
LeakyReLU					
Convolution	512	1	4	2	1
Upsample					

(a) Pixel-level weighting network W_p^1

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	4	2	1
LeakyReLU					
Convolution	64	128	4	2	1
LeakyReLU					
Convolution	128	256	4	2	1
LeakyReLU					
Convolution	256	512	4	2	1
LeakyReLU					
Convolution	512	1	4	2	1

(b) Region-level weighting network W_r^1

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	4	2	1
LeakyReLU					
Convolution	64	128	4	2	1
LeakyReLU					
Convolution	128	256	4	2	1
LeakyReLU					
Convolution	256	512	4	2	1
LeakyReLU					
Convolution	512	1	4	2	1
mean value					

(c) Image-level weighting network W_i^1

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	3	2	1
LeakyReLU					
Convolution	64	128	3	2	1
LeakyReLU					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Conv_Transpose	128	19	1	1	0
Upsample					

(d) Pixel-level weighting network W_p^{19}

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	3	2	1
LeakyReLU					
Convolution	64	128	3	2	1
LeakyReLU					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Conv_Transpose	128	19	1	1	0

(e) Region-level weighting network W_r^{19}

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	19	64	3	2	1
LeakyReLU					
Convolution	64	128	3	2	1
LeakyReLU					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Convolution	128	128	3	1	1
InstanceNorm					
Conv_Transpose	128	19	1	1	0
mean value					

(f) Image-level weighting network W_i^{19}

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	512	512	3	1	1
InstanceNorm					
RELU					
Convolution	512	512	3	1	1
InstanceNorm					
Convolution	512	512	3	1	1
InstanceNorm					
RELU					
Convolution	512	512	3	1	1
InstanceNorm					
Conv_Transpose	512	256	3	2	1
InstanceNorm					
RELU					
Conv_Transpose	256	128	3	2	1
InstanceNorm					
RELU					
Conv_Transpose	128	64	3	2	1
InstanceNorm					
RELU					
Conv_Transpose	64	32	3	2	1
InstanceNorm					
RELU					
Conv_Transpose	32	16	3	2	1
InstanceNorm					
RELU					
Conv_Transpose	16	3	1	1	0
Than					

(g) Generator G

Type	Input channels	Output channels	Kernel size	Stride	padding
Convolution	3	64	3	2	1
LeakyReLU					
Convolution	64	128	3	2	1
LeakyReLU					
Convolution	128	256	3	2	1
LeakyReLU					
Convolution	256	512	3	2	1
LeakyReLU					
Convolution	512	1024	3	2	1
LeakyReLU					
Convolution	1024	2048	3	2	1
LeakyReLU					
Convolution	2048	1	1	1	1

(h) Discriminator D

Figure 2. The detailed architectures of our hierarchical weighting networks, including the pixel- (W_p^1), region- (W_r^1), and image-level (W_i^1) weighting networks under shared scheme, and pixel- (W_p^{19}), region- (W_r^{19}), and image-level (W_i^{19}) weighting networks under multi-channel weighting scheme, as well as the generator G and the discriminator D networks.