Stable Neighbor Denoising for Source-free Domain Adaptive Segmentation (Supplementary Material)

Dong Zhao[1], Shuang Wang[1] ✉, Qi Zang[1], Licheng Jiao[1], Nicu Sebe[2], Zhun Zhong[3,4] ✉
[1] School of Artificial Intelligence, Xidian University, Shaanxi, China
[2] Department of Information Engineering and Computer Science, University of Trento, Italy
[3] School of Computer Science and Information Engineering, Hefei University of Technology, China
[4] School of Computer Science, University of Nottingham, NG8 1BB Nottingham, UK

1. Why using the inner optimization for the teacher model.

Applying Bi-Level optimization to the same model (BL on same) also works well, as shown in the table below. We employ it in the student-teacher framework (S-T) for two reasons. First, teachers obtained by Exponential Moving Average (EMA) often outperform student models [1,3], providing more accurate pseudo-labels for students and promoting the stability of self-training. Second, most existing SFDA works are employed based on it, we keep this design for a fair comparison.

	BL on same	BL on S-T	BL on same + DTST	BL on S-T + DTST
GTA → Cityscapes (CS)	55.1	55.6	57.5	58.1
Synthia → CS	53.4	54.1	54.7	55.4
CS → ACDC	47.7	48.1	48.9	49.6
GTA → BDD100k	47.1	47.8	48.7	49.5

2. Does S-T cause bias in bi-level optimization?

We supplement the ablations of querying strategies (Table 5) on both implements in the below TABLE. It shows that retrieving nearest neighbors ($\mathcal{Q}_{layout} + Q_{style}$) brings significant improvements on both 'S-T model' and 'same model' implementations. This verifies that *domain factors are the main factors causing bias in Bi-level optimization instead of using S-T model.*

	BL on same model		BL on S-T	
Query method	GTA → CS	CS → ACDC	GTA → CS	CS → ACDC
Random	53.5	43.4	54.8	44.7
$\mathcal{Q}_{layout}$	54.9	44.2	55.3	44.9
$\mathcal{Q}_{style}$	54.4	47.1	55.6	47.5
$\mathcal{Q}_{layout} + Q_{style}$	55.1	47.7	55.6	48.1

3. Whether using multiple inner loops with multiple neighbors are useful?

In principle, using multi-step inner loops with multiple neighbors will indeed lead to more stable gradient optimization. However, the computational overhead of multi-step optimization is too high and difficult to accept in actual training. We show (Table 1) that using "multi-step inner loops" obtains very similar results as "one-step one", but significantly increases the training time. In addition, we show that replacing our "retrieving nearest neighbors" by "random neighbors" clearly reduces the performance for "multi-step inner loops". These results indicate that 1) one-step inner loop is sufficient for our method and that 2) our "retrieving nearest neighbors" is important and saves a lot of training time compared to the "random one". Therefore, retrieving nearest neighbors is an effective way to eliminate bi-level optimization bias under limited iterations and acceptable cost.

	0-step	1-step	2-step	3-step	random 3-step
Ours (mIoU %)	50.5	55.6	56.1	56.4	53.5
Training Time (hours)	12.5	18.4	36.7	70.5	70.1
Ours + DTST (mIoU %)	54.4	58.1	58.3	58.6	56.5
Training Time (hours)	13.1	20.5	40.9	80.5	79.9

Table 1. The impact of the number of inner loops on performance and time-consuming in GTA → Cityscapes task.

4. Whether the estimated ω^* in the current iteration can be used as initialization for the next optimization? And is there any other manner?

We further implement the predicted probabilities as initial values for ω. The Table below shows that using the last ω^* achieves largely lower results than ours and the probability-based variant. The main reason is: ω measures the status of the current pseudo-labels. When the model optimizes that sample again, it has been updated multiple times, so its pseudo-labels undergo significant changes. It results in the last ω^* not matching with the current pseudo-labels at all.

	Last ω	All Ones (Ours)	Probability
GTA $\rightarrow$ Cityscapes	52.1	55.6	55.2
Cityscapes $\rightarrow$ ACDC	45.1	48.1	47.7

5. Work on both CNN and Transformer.

In the table below, we add the results of using the Segformer [2], indicating that our method consistently improves the performance on both CNN and Transformer.

	Segformer		
	Source	DTST	Ours
GTA $\rightarrow$ Cityscapes	52.6	60.7	62.9
Cityscapes $\rightarrow$ ACDC	47.7	50.2	51.6
GTA $\rightarrow$ BDD100k	47.1	51.6	53.7

6. Detailed weather results on Cityscapes $\rightarrow$ ACDC.

	Fog	Night	Rain	Snow	mIoU
DTST [3]	56.1	32.1	46.7	46.9	45.4
SND	57.2	34.1	51.3	49.8	48.1
DTST+SND	60.1	34.5	53.6	51.6	49.6

References

[1] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9924–9935, 2022. 1

[2] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021. 2

[3] Dong Zhao, Shuang Wang, Qi Zang, Dou Quan, Xiutiao Ye, and Licheng Jiao. Towards better stability and adaptability: Improve online self-training for model adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11733–11743, 2023. 1, 2