– Supplementary Material – HRDA: Context-Aware High-Resolution Domain-Adaptive Semantic Segmentation

Lukas Hoyer[1], Dengxin Dai[2], and Luc Van Gool[1,3]

[1] ETH Zurich, Switzerland {lhoyer,vangool}@vision.ee.ethz.ch
[2] MPI for Informatics, Germany ddai@mpi-inf.mpg.de
[3] KU Leuven, Belgium

A Overview

In the supplementary material for HRDA, we provide the source code (Sec. B), additional experimental analysis (Sec. C and D), comparisons with further baselines (Sec. E), an analysis of the runtime (Sec. F), an extended comparison with previous UDA methods (Sec. G), and a comprehensive qualitative analysis of the predictions from HRDA (Sec. H).

B Source Code

The source code to reproduce HRDA and all ablation studies is provided at https://github.com/lhoyer/HRDA. Please, refer to the contained `README.md` for further instructions to set up the environment and run the experiments. Our implementation is based on the DAFormer framework [7] and the mmsegmentation framework [4].

C Influence of Detail Loss Weight

In Fig. S1, the sensitivity of the UDA performance of HRDA with respect to the detail loss weight λ_d is studied. It is shown that values in the range between 0.1 and 0.3 give a consistently good UDA performance, which is a reasonably broad range for a robust hyperparameter choice. If λ_d is either too small or too large, HRDA focuses too much on LR or HR, respectively.

D Influence of Context Scale

We further study the influence of the downscale factor of the context crop s_c in Tab. S1. It can be seen that the default downscale factor $s_c = 2$ provides the best performance. A context crop with a higher downscale factor $s_c = 4$ performs worse by -2.6 mIoU than the default choice (cf. row 2 and 3) but is still slightly better than just using the detail crop by +0.8 mIoU (cf. row 1 and

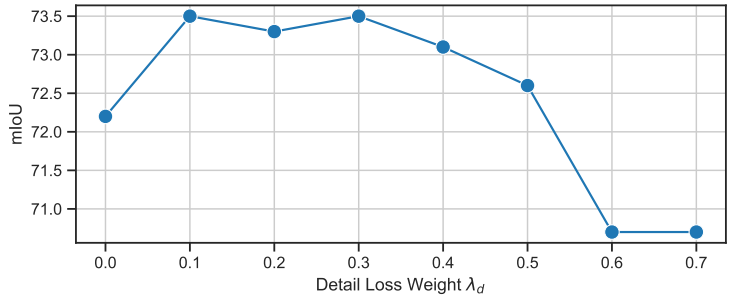

Fig. S1. Study of the UDA performance of HRDA with respect to the detail loss weight λ_d on GTA→Cityscapes.

Table S1. Influence of the context downscale factor s on HRDA performance. A larger downscale factor s results in a lower crop resolution. The relative crop size is $a{=}h/\frac{H_T}{s}$.

	Context s_c	Context Rel. Size a_c	Detail s_d	Detail Rel. Size a_d	mIoU
1	–	–	1 (HR)	0.5	65.1 $\pm$1.9
2	4	0.5	1 (HR)	0.5	65.9 $\pm$1.2
3	2 (LR)	0.5	1 (HR)	0.5	68.5 $\pm$0.6
4	1.33	0.5	1 (HR)	0.5	67.5 $\pm$0.7

2). We assume that with $s_c = 4$ the resolution of the context crop is too low to be useful for UDA. A context crop with a small downscale factor $s_c = 1.33$ performs better than the high downscale factor $s_c = 4$ by +1.6 mIoU (cf. row 2 and 4) but still does not achieve the performance of the default $s_c = 2$ with a difference of -1.0 mIoU. Possibly, the resolution of $s_c = 1.33$ is too similar to the detail crop resolution $s_d = 1$ and, therefore, it does not provide a sufficiently different perspective on the data, which is important for multi-resolution UDA.

E Further Baselines

E.1 Overlapping Sliding Window Inference (OSW)

Prior works in the field of UDA (including DAFormer) use whole image inference while HRDA utilizes overlapping sliding window inference (OSW). To show that the improvement of HRDA is not mainly caused by the OSW inference, we also evaluate prior arts with OSW (see Tab. S2 col. 4). It can be seen that OSW only slightly benefits DAFormer by +0.3 mIoU. Still, HRDA outperforms DAFormer with OSW by +5.2 mIoU.

E.2 Naive High-Resolution UDA

In Sec. 5.5 of the main paper, we compared HRDA with the naive HR crops ($\text{HR}_{0.75}$) for DAFormer [7]. Here, we extend this comparison also to DACS [15],

which uses another UDA method and network architecture. Tab. S2 col. 5 shows that naive HR training improves the performance of DACS similarly to DAFormer. HRDA outperforms naive HR training by +3.6 mIoU for DACS and +3.8 mIoU for DAFormer.

Table S2. Overlapping sliding window inference (OSW) and naive HR training for different UDA methods and network architectures on GTA→Cityscapes.

UDA Method	Network	Baseline	OSW	Naive HR+OSW	HRDA
DACS [15]	DeepLabV2 [2]	53.9 ±0.6	53.9 ±0.6	55.8 ±1.6	59.4 ±1.2
DAFormer [7]	DAFormer [7]	68.3 ±0.5	68.6 ±0.3	70.0 ±1.2	73.8 ±0.3

E.3 Scale-Invariance Loss

As another baseline for HRDA, we integrate the scale-invariance loss of Guan et al. [5] into DAFormer [7]. The optimal loss weight when combined with DAFormer is determined with a grid search as 0.1. As shown in Tab. S3, DAFormer with scale-invariance loss [5] achieves 68.8 mIoU on GTA→Cityscapes, which is only a small gain of +0.5 mIoU over DAFormer, while HRDA achieves +5.5 mIoU. We assume that the effect of the scale-invariance loss is not so pronounced as DAFormer is much stronger (68.3 mIoU) than the original baseline (43.8 mIoU). This further emphasizes that the contribution of HRDA goes beyond scale consistency training.

Table S3. Comparison of scale consistency training and HRDA on GTA→Cityscapes.

	Baseline	w/ Scale-Invariance [5]	w/ HRDA
Original [5]	43.8	48.1	–
DAFormer [7]	63.3	63.8	73.8

F Training and Inference Time

The training of HRDA takes 32h on a Titan RTX. To put this into context, the training of other UDA methods [10, 24, 25] can take several days. The runtime and memory consumption of HRDA during inference is shown in Tab. S4. The inference runs with 0.8 img/s, 3.3 TFLOPs, and 9.4 GB GPU memory. The focus of HRDA is UDA performance and not fast inference as the latter is not an inherent constraint of UDA. Still, if efficient inference is important, *non*-overlapping sliding window inference (stride equal to window size) can be used at test-time resulting in 2.2 img/s, 1.8 TFLOPs, and 3.2 GB with still 73.4 mIoU. Alternatively, the knowledge of HRDA can be distilled into a faster single-scale

DeepLabV2 [2] model, which is commonly used for UDA. For that purpose, the multi-resolution HRDA is utilized to generate high-quality pseudo-labels for the target domain and the single-scale DeepLabV2 model is trained on the target domain with a pixel-wise cross-entropy loss using the pseudo-labels. The distilled DeepLabV2 model achieves 70.4 mIoU at 3.4 img/s, 1.4 TFLOPs, and 1.3 GB. Both results are significantly better than the previous SOTA performance of DAFormer [7], which is 68.3 mIoU.

Table S4. Runtime and memory consumption of HRDA variants during inference on an Nvidia Titan RTX.

	Throughput (img/s)	TFLOPs	GPU Mem. (GB)	mIoU
HRDA	0.8	3.3	9.4	73.8
HRDA w/ Non-Overlapping SW	2.2	1.8	3.2	73.4
HRDA w/ Distilled DeepLabV2	3.4	1.4	1.3	70.4

G Extended Comparison with Previous UDA Methods

We extend the comparison of HRDA with previous UDA methods from the main paper by a large selection of further methods for GTA→Cityscapes in Tab. S5 and for Synthia→Cityscapes in Tab. S6. It can be seen that $\text{HRDA}_{\text{DAFormer}}$ (the default HRDA based on DAFormer) still outperforms previous UDA methods by a large margin both for the class-wise IoU as well as the overall mIoU. The highest performance gains are achieved for classes with fine segmentation details such as pole, traffic light, traffic sign, person, rider, motorbike, and bike. But also large classes such as truck, bus, and train benefit from HRDA. Only a few classes such as road, sidewalk, fence, and vegetation on Synthia→Cityscapes have a lower performance than the respective best comparison method. The comparably low performance is probably inherited from DAFormer [7], which is the basis of HRDA. This effect might be caused by the shape bias of the used Transformer encoder as discussed in DAFormer [7]. Possibly, the performance for the mentioned stuff classes could be improved for HRDA by integrating the depth-clues as done in CorDA [21] or pseudo-label prototypes as used in ProDA [25]. Further, Tab. S5 and Tab. S6 show the performance of HRDA when used with a DeepLabV2 network instead of a DAFormer network. It can be seen that $\text{HRDA}_{\text{DeepLabV2}}$ outperforms all DeepLabV2-based UDA methods (all methods except DAFormer) on GTA→Cityscapes and that it even outperforms DAFormer on Synthia→Cityscapes.

Table S5. Comparison with previous UDA methods on GTA→Cityscapes. The results of HRDA are averaged over 3 random seeds.

	Road	S.walk	Build.	Wall	Fence	Pole	Tr.Light	Sign	Veget.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bike	mIoU
AdaptSeg [16]	86.5	25.9	79.8	22.1	20.0	23.6	33.1	21.8	81.8	25.9	75.9	57.3	26.2	76.3	29.8	32.1	7.2	29.5	32.5	41.4
CyCADA [6]	86.7	35.6	80.1	19.8	17.5	38.0	39.9	41.5	82.7	27.9	73.6	64.9	19.0	65.0	12.0	28.6	4.5	31.1	42.0	42.7
CLAN [11]	87.0	27.1	79.6	27.3	23.3	28.3	35.5	24.2	83.6	27.4	74.2	58.6	28.0	76.2	33.1	36.7	6.7	31.9	31.4	43.2
ADVENT [18]	89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
APODA [23]	85.6	32.8	79.0	29.5	25.5	26.8	34.6	19.9	83.7	40.6	77.9	59.2	28.3	84.6	34.6	49.2	8.0	32.6	39.6	45.9
CBST [28]	91.8	53.5	80.5	32.7	21.0	34.0	28.9	20.4	83.9	34.2	80.9	53.1	24.0	82.7	30.3	35.9	16.0	25.9	42.8	45.9
PatchAlign [17]	92.3	51.9	82.1	29.2	25.1	24.5	33.8	33.0	82.4	32.8	82.2	58.6	27.2	84.3	33.4	46.3	2.2	29.5	32.3	46.5
MRKLD [29]	91.0	55.4	80.0	33.7	21.4	37.3	32.9	24.5	85.0	34.1	80.8	57.7	24.6	84.1	27.8	30.1	26.9	26.0	42.3	47.1
BDL [9]	91.0	44.7	84.2	34.6	27.6	30.2	36.0	36.0	85.0	43.6	83.0	58.6	31.6	83.3	35.3	49.7	3.3	28.8	35.6	48.5
FADA [20]	91.0	50.6	86.0	43.4	29.8	36.8	43.4	25.0	86.8	38.3	87.4	64.0	38.0	85.2	31.6	46.1	6.5	25.4	37.1	50.1
CAG [26]	90.4	51.6	83.8	34.2	27.8	38.4	25.3	48.4	85.4	38.2	78.1	58.6	34.6	84.7	21.9	42.7	41.1	29.3	37.2	50.2
Seg-Uncert. [27]	90.4	31.2	85.1	36.9	25.6	37.5	48.8	48.5	85.3	34.8	81.1	64.4	36.8	86.3	34.9	52.2	1.7	29.0	44.6	50.3
FDA [24]	92.5	53.3	82.4	26.5	27.6	36.4	40.6	38.9	82.3	39.8	78.0	62.6	34.4	84.9	34.1	53.1	16.9	27.7	46.4	50.5
PIT [12]	87.5	43.4	78.8	31.2	30.2	36.3	39.9	42.0	79.2	37.1	79.3	65.4	37.5	83.2	46.0	45.6	25.7	23.5	49.9	50.6
IAST [14]	93.8	57.8	85.1	39.5	26.7	26.2	43.1	34.7	84.9	32.9	88.0	62.6	29.0	87.3	39.2	49.6	23.2	34.7	39.6	51.5
DACS [15]	89.9	39.7	87.9	30.7	39.5	38.5	46.4	52.8	88.0	44.0	88.8	67.2	35.8	84.5	45.7	50.2	0.0	27.3	34.0	52.1
SAC [1]	90.4	53.9	86.6	42.4	27.3	45.1	48.5	42.7	87.4	40.1	86.1	67.5	29.7	88.5	49.1	54.6	9.8	26.6	45.3	53.8
CTF [13]	92.5	58.3	86.5	27.4	28.8	38.1	46.7	42.5	85.4	38.4	91.8	66.4	37.0	87.8	40.7	52.4	44.6	41.7	59.0	56.1
CorDA [21]	94.7	63.1	87.6	30.7	40.6	40.2	47.8	51.6	87.6	47.0	89.7	66.7	35.9	90.2	48.9	57.5	0.0	39.8	56.0	56.6
BAPA [10]	94.4	61.0	88.0	26.8	39.9	38.3	46.1	55.3	87.8	46.1	89.4	68.8	40.0	90.2	60.4	59.0	0.0	45.1	54.2	57.4
ProDA [25]	87.8	56.0	79.7	46.3	44.8	45.6	53.5	53.5	88.6	45.2	82.1	70.7	39.2	88.8	45.5	59.4	1.0	48.9	56.4	57.5
DAFormer [7]	95.7	70.2	89.4	53.5	48.1	49.6	55.8	59.4	89.9	47.9	92.5	72.2	44.7	92.3	74.5	78.2	65.1	55.9	61.8	68.3
$\text{HRDA}_{\text{DeepLabV2}}$	96.2	73.1	89.7	43.2	39.9	47.5	60.0	60.0	89.9	47.1	90.2	75.9	49.0	91.8	61.9	59.3	10.2	47.0	65.3	63.0
$\text{HRDA}_{\text{DAFormer}}$	**96.4**	**74.4**	**91.0**	**61.6**	**51.5**	**57.1**	**63.9**	**69.3**	**91.3**	**48.4**	**94.2**	**79.0**	**52.9**	**93.9**	**84.1**	**85.7**	**75.9**	**63.9**	**67.5**	**73.8**

Table S6. Comparison with previous UDA methods on Synthia→Cityscapes. The results of HRDA are averaged over 3 random seeds.

	Road	S.walk	Build.	Wall	Fence	Pole	Tr.Light	Sign	Veget.	Sky	Person	Rider	Car	Bus	M.bike	Bike	mIoU16	mIoU13
SPIGAN [8]	71.1	29.8	71.4	3.7	0.3	33.2	6.4	15.6	81.2	78.9	52.7	13.1	75.9	25.5	10.0	20.5	36.8	42.4
GIO-Ada [3]	78.3	29.2	76.9	11.4	0.3	26.5	10.8	17.2	81.7	81.9	45.8	15.4	68.0	15.9	7.5	30.4	37.3	43.0
AdaptSeg [16]	79.2	37.2	78.8	–	–	–	9.9	10.5	78.2	80.5	53.5	19.6	67.0	29.5	21.6	31.3	–	45.9
PatchAlign [17]	82.4	38.0	78.6	8.7	0.6	26.0	3.9	11.1	75.5	84.6	53.5	21.6	71.4	32.6	19.3	31.7	40.0	46.5
CLAN [11]	81.3	37.0	80.1	–	–	–	16.1	13.7	78.2	81.5	53.4	21.2	73.0	32.9	22.6	30.7	–	47.8
ADVENT [18]	85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	41.2	48.0
CBST [28]	68.0	29.9	76.3	10.8	1.4	33.9	22.8	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	42.6	48.9
DADA [19]	89.2	44.8	81.4	6.8	0.3	26.2	8.6	11.1	81.8	84.0	54.7	19.3	79.7	40.7	14.0	38.8	42.6	49.8
MRKLD [29]	67.7	32.2	73.9	10.7	1.6	37.4	22.2	31.2	80.8	80.5	60.8	29.1	82.8	25.0	19.4	45.3	43.8	50.1
BDL [9]	86.0	46.7	80.3	–	–	–	14.1	11.6	79.2	81.3	54.1	27.9	73.7	42.2	25.7	45.3	–	51.4
CAG [26]	84.7	40.8	81.7	7.8	0.0	35.1	13.3	22.7	84.5	77.6	64.2	27.8	80.9	19.7	22.7	48.3	44.5	51.5
PIT [12]	83.1	27.6	81.5	8.9	0.3	21.8	26.4	33.8	76.4	78.8	64.2	27.6	79.6	31.2	31.0	31.3	44.0	51.8
SIM [22]	83.0	44.0	80.3	–	–	–	17.1	15.8	80.5	81.8	59.9	33.1	70.2	37.3	28.5	45.8	–	52.1
FDA [24]	79.3	35.0	73.2	–	–	–	19.9	24.0	61.7	82.6	61.4	31.1	83.9	40.8	38.4	51.1	–	52.5
FADA [20]	84.5	40.1	83.1	4.8	0.0	34.3	20.1	27.2	84.8	84.0	53.5	22.6	85.4	43.7	26.8	27.8	45.2	52.5
APODA [23]	86.4	41.3	79.3	–	–	–	22.6	17.3	80.3	81.6	56.9	21.0	84.1	49.1	24.6	45.7	–	53.1
DACS [15]	80.6	25.1	81.9	21.5	2.9	37.2	22.7	24.0	83.7	90.8	67.6	38.3	82.9	38.9	28.5	47.6	48.3	54.8
Seg-Uncert. [27]	87.6	41.9	83.1	14.7	1.7	36.2	31.3	19.9	81.6	80.6	63.0	21.8	86.2	40.7	23.6	53.1	47.9	54.9
CTF [13]	75.7	30.0	81.9	11.5	2.5	35.3	18.0	32.7	86.2	90.1	65.1	33.2	83.3	36.5	35.3	54.3	48.2	55.5
IAST [14]	81.9	41.5	83.3	17.7	4.6	32.3	30.9	28.8	83.4	85.0	65.5	30.8	86.5	38.2	33.1	52.7	49.8	57.0
SAC [1]	89.3	47.2	85.5	26.5	1.3	43.0	45.5	32.0	87.1	89.3	63.6	25.4	86.9	35.6	30.4	53.0	52.6	59.3
BAPA [10]	91.7	53.8	83.9	22.4	0.8	34.9	30.5	42.8	86.6	88.2	66.0	34.1	86.6	51.3	29.4	50.5	53.3	61.2
ProDA [25]	87.8	45.7	84.6	37.1	0.6	44.0	54.6	37.0	**88.1**	84.4	74.2	24.3	88.2	51.1	40.5	45.6	55.5	62.0
CorDA [21]	**93.3**	**61.6**	85.3	19.6	5.1	37.8	36.6	42.8	84.9	90.4	69.7	41.8	85.6	38.4	32.6	53.9	55.0	62.8
DAFormer [7]	84.5	40.7	88.4	41.5	**6.5**	50.0	55.0	54.6	86.0	89.8	73.2	48.2	87.2	53.2	53.9	61.7	60.9	67.4
$\text{HRDA}_{\text{DeepLabV2}}$	85.8	47.3	87.3	27.3	1.4	50.5	57.8	**61.0**	87.4	89.1	76.2	48.5	87.3	49.3	55.0	**68.2**	61.2	69.2
$\text{HRDA}_{\text{DAFormer}}$	85.2	47.7	**88.8**	**49.5**	4.8	**57.2**	**65.7**	60.9	85.3	**92.9**	**79.4**	**52.8**	**89.0**	**64.7**	**63.9**	64.9	**65.8**	**72.4**

H Further Qualitative Examples

In Fig. S2-S6, we compare the predicted semantic segmentation of HRDA to the two strongest previous UDA methods from Tab. S5, namely ProDA [25] and DAFormer [7]. Further, we visualize the scale attention of HRDA as the weighted sum over the scale attention channels for each class with weight being the softmax of the segmentation prediction. White regions mean that HRDA focuses on the prediction from the HR input.

Overall, HRDA better recognizes small classes and segments finer details. This is especially the case for distant poles, traffic lights, and traffic signs (see Fig. S2) as well as distant pedestrians, riders, bicycles, and motorcycles (see Fig. S3). For these regions, HRDA uses the prediction from the HR input as can be seen in the HRDA scale attention (white encodes a focus on HR). Further, HRDA is able to better recognize difficult stuff classes such as sidewalk and wall (see Fig. S4) as well as to better distinguish different vehicle classes (see Fig. S5). HRDA uses LR input for that purpose as can be seen in the HRDA scale attention (black encodes a focus on LR).

Even though HRDA sets new standards, UDA is still a challenging task. This can be observed for classes that are easy to confuse with others and that have a considerable domain gap such as sidewalk, terrain, or fence, which results in adaptation errors (see Fig. S6).

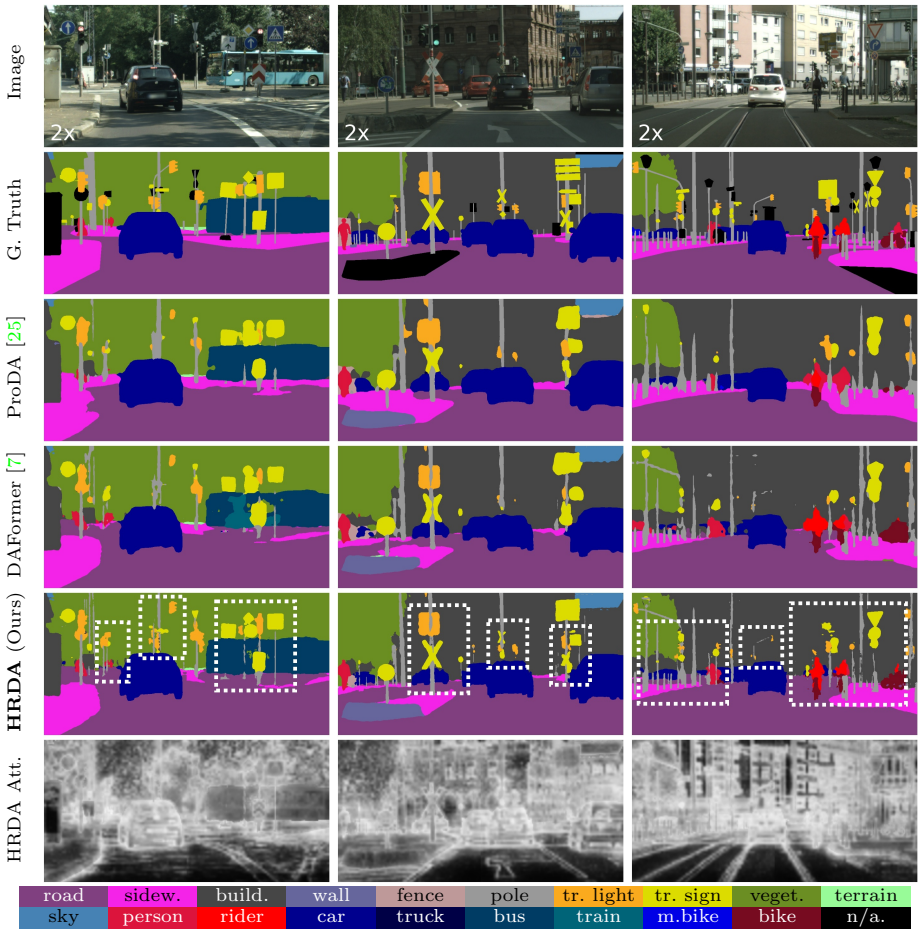

Fig. S2. Example predictions showing a better recognition and finer segmentation details of small classes such as *pole*, *traffic light*, and *traffic sign* on GTA→Cityscapes. Some examples are zoomed in for better visibility of the details. The zoom factor is provided in the bottom left corner of each image.

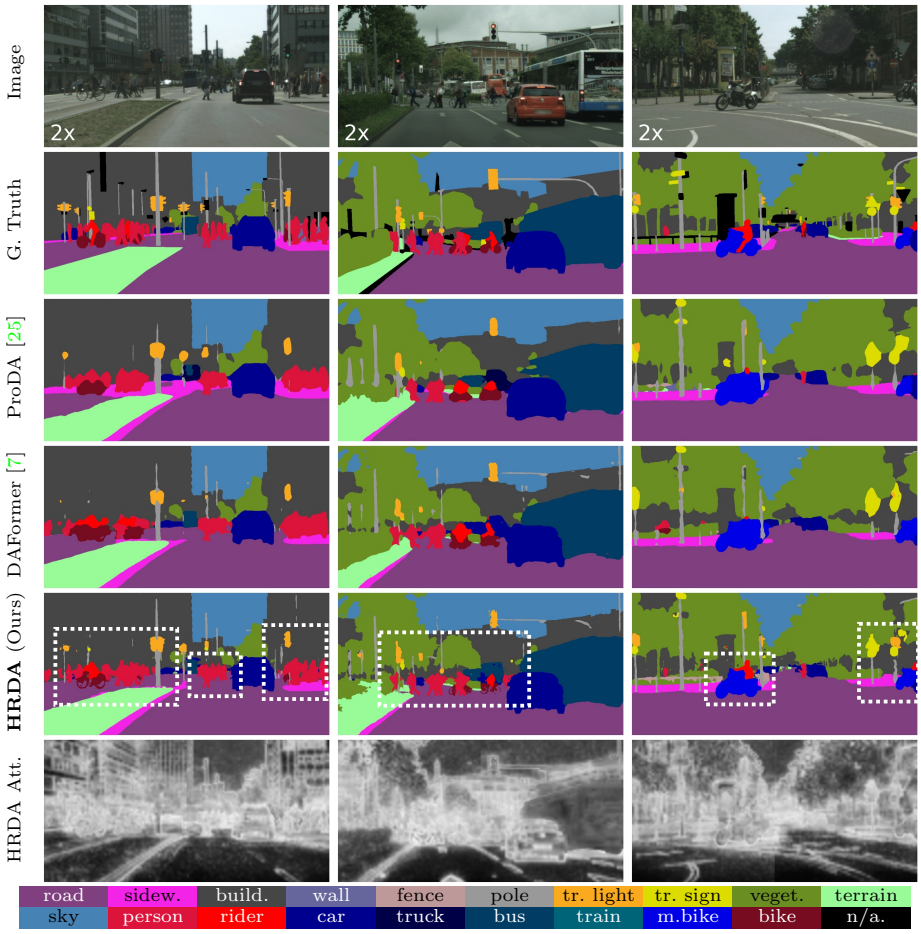

Fig. S3. Example predictions showing a better recognition and finer segmentation of small distant classes such as *pedestrian*, *rider*, *motorcycle* and *bicycle* on GTA→Cityscapes. Some examples are zoomed in for better visibility of the details. The zoom factor is provided in the bottom left corner of each image.

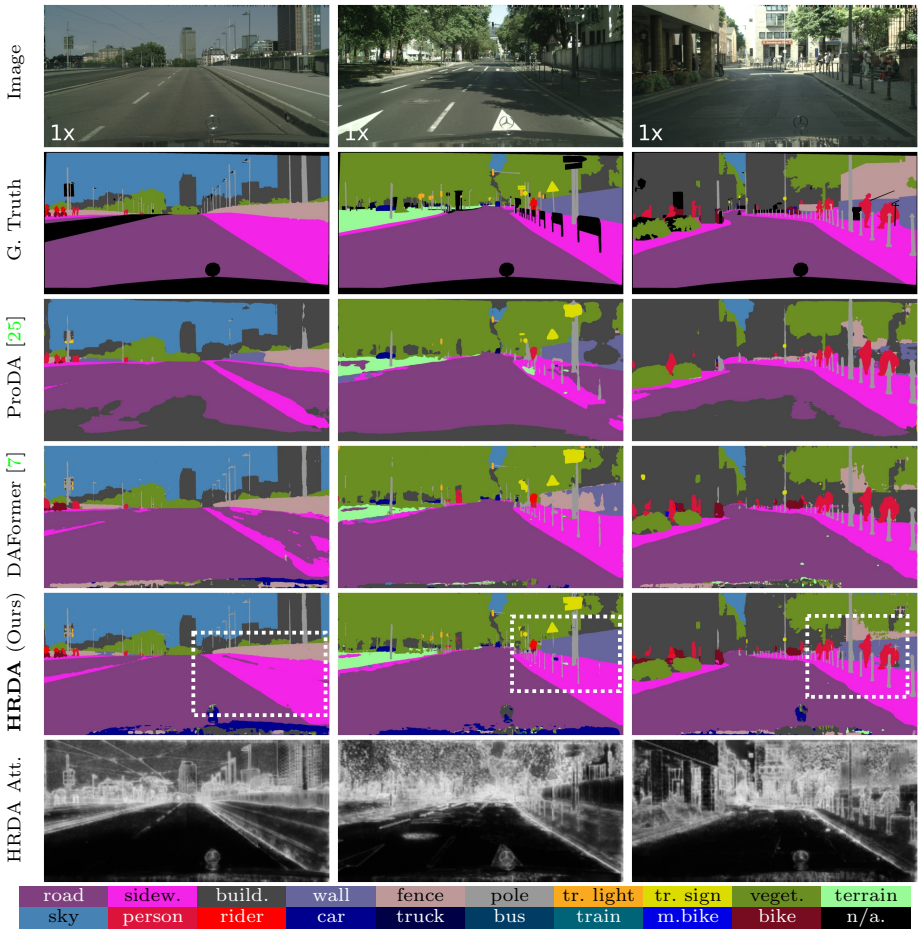

Fig. S4. Example predictions showing a better recognition of difficult stuff classes such as *sidewalk* and *wall* on GTA→Cityscapes. Some examples are zoomed in for better visibility of the details. The zoom factor is provided in the bottom left corner of each image.

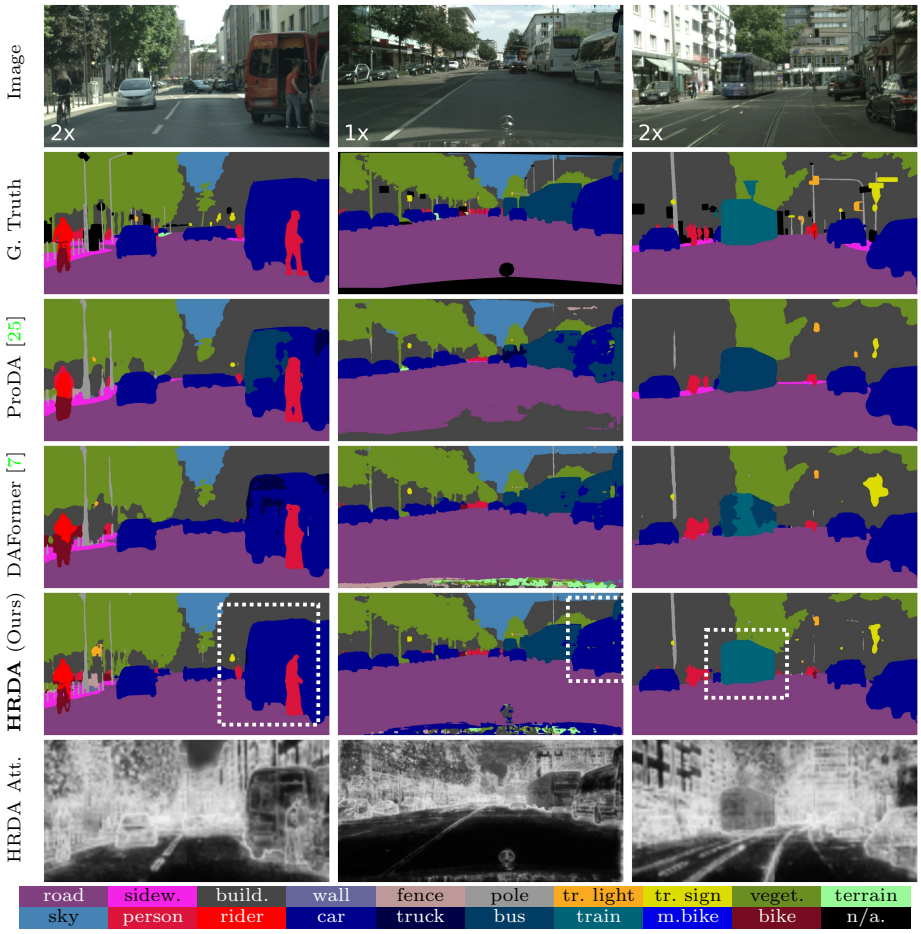

Fig. S5. Example predictions showing a better differentiation of vehicle classes such as *car*, *truck*, *bus*, and *train* on GTA→Cityscapes. Some examples are zoomed in for better visibility of the details. The zoom factor is provided in the bottom left corner of each image.

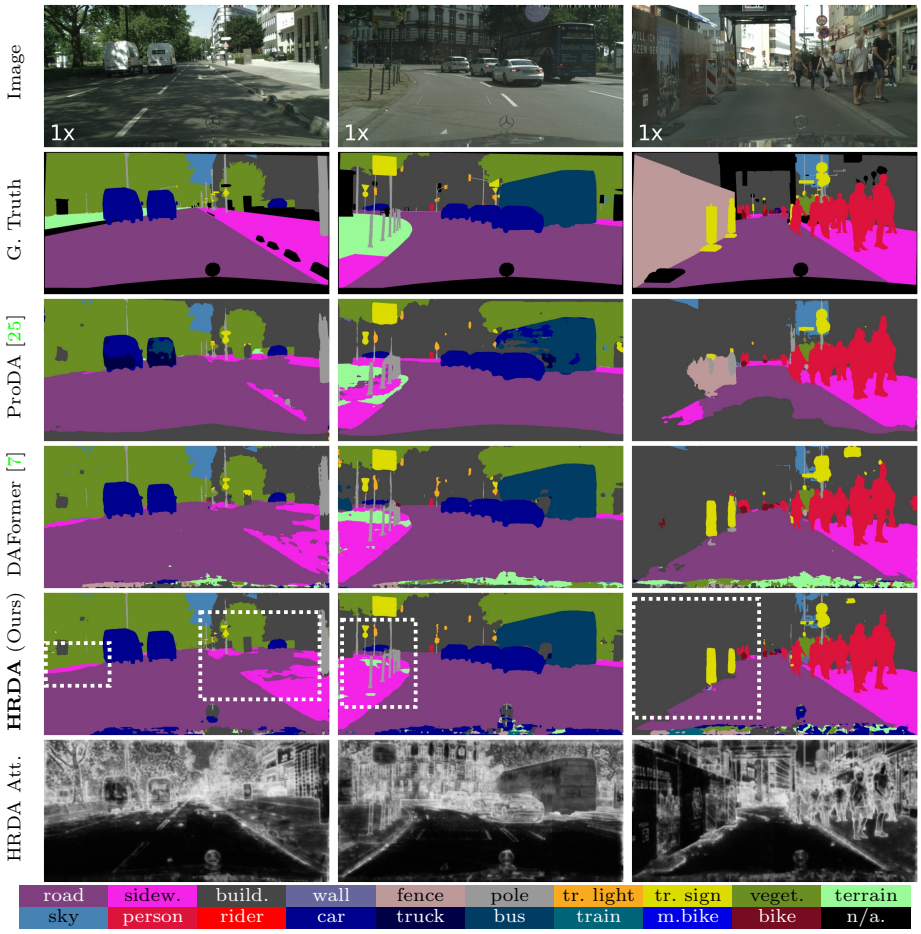

Fig. S6. Failure cases of classes with a low UDA performance such as *sidewalk*, *terrain*, and *fence* on GTA→Cityscapes. Some examples are zoomed in for better visibility of the details. The zoom factor is provided in the bottom left corner of each image.

References

1. Araslanov, N., Roth, S.: Self-supervised augmentation consistency for adapting semantic segmentation. In: CVPR. pp. 15384–15394 (2021) 5
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. PAMI **40**(4), 834–848 (2017) 3, 4
3. Chen, Y., Li, W., Chen, X., Gool, L.V.: Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In: CVPR. pp. 1841–1850 (2019) 5
4. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020) 1
5. Guan, D., Huang, J., Lu, S., Xiao, A.: Scale variance minimization for unsupervised domain adaptation in image segmentation. PR **112**, 107764 (2021) 3
6. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. In: ICML. pp. 1989–1998 (2018) 5
7. Hoyer, L., Dai, D., Van Gool, L.: DAFormer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In: CVPR (2022) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
8. Lee, K.H., Ros, G., Li, J., Gaidon, A.: Spigan: Privileged adversarial learning from simulation. In: ICLR (2018) 5
9. Li, Y., Yuan, L., Vasconcelos, N.: Bidirectional learning for domain adaptation of semantic segmentation. In: CVPR. pp. 6936–6945 (2019) 5
10. Liu, Y., Deng, J., Gao, X., Li, W., Duan, L.: Bapa-net: Boundary adaptation and prototype alignment for cross-domain semantic segmentation. In: ICCV. pp. 8801–8811 (2021) 3, 5
11. Luo, Y., Zheng, L., Guan, T., Yu, J., Yang, Y.: Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In: CVPR. pp. 2507–2516 (2019) 5
12. Lv, F., Liang, T., Chen, X., Lin, G.: Cross-domain semantic segmentation via domain-invariant interactive relation transfer. In: CVPR. pp. 4334–4343 (2020) 5
13. Ma, H., Lin, X., Wu, Z., Yu, Y.: Coarse-to-fine domain adaptive semantic segmentation with photometric alignment and category-center regularization. In: CVPR. pp. 4051–4060 (2021) 5
14. Mei, K., Zhu, C., Zou, J., Zhang, S.: Instance adaptive self-training for unsupervised domain adaptation. In: ECCV. pp. 415–430 (2020) 5
15. Tranheden, W., Olsson, V., Pinto, J., Svensson, L.: DACS: Domain Adaptation via Cross-domain Mixed Sampling. In: WACV. pp. 1379–1389 (2021) 2, 3, 5
16. Tsai, Y.H., Hung, W.C., Schulter, S., Sohn, K., Yang, M.H., Chandraker, M.: Learning to adapt structured output space for semantic segmentation. In: CVPR. pp. 7472–7481 (2018) 5
17. Tsai, Y.H., Sohn, K., Schulter, S., Chandraker, M.: Domain adaptation for structured output via discriminative patch representations. In: ICCV. pp. 1456–1465 (2019) 5
18. Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: CVPR. pp. 2517–2526 (2019) 5
19. Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Dada: Depth-aware domain adaptation in semantic segmentation. In: ICCV. pp. 7364–7373 (2019) 5

20. Wang, H., Shen, T., Zhang, W., Duan, L.Y., Mei, T.: Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In: ECCV. pp. 642–659 (2020) 5
21. Wang, Q., Dai, D., Hoyer, L., Fink, O., Van Gool, L.: Domain adaptive semantic segmentation with self-supervised depth estimation. In: ICCV. pp. 8515–8525 (2021) 4, 5
22. Wang, Z., Yu, M., Wei, Y., Feris, R., Xiong, J., Hwu, W.m., Huang, T.S., Shi, H.: Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In: CVPR. pp. 12635–12644 (2020) 5
23. Yang, J., Xu, R., Li, R., Qi, X., Shen, X., Li, G., Lin, L.: An adversarial perturbation oriented domain adaptation approach for semantic segmentation. In: AAAI. pp. 12613–12620 (2020) 5
24. Yang, Y., Soatto, S.: Fda: Fourier domain adaptation for semantic segmentation. In: CVPR. pp. 4085–4095 (2020) 3, 5
25. Zhang, P., Zhang, B., Zhang, T., Chen, D., Wang, Y., Wen, F.: Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In: CVPR. pp. 12414–12424 (2021) 3, 4, 5, 6, 7, 8, 9, 10, 11
26. Zhang, Q., Zhang, J., Liu, W., Tao, D.: Category anchor-guided unsupervised domain adaptation for semantic segmentation. In: NeurIPS. pp. 435–445 (2019) 5
27. Zheng, Z., Yang, Y.: Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. IJCV **129**(4), 1106–1120 (2021) 5
28. Zou, Y., Yu, Z., Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: ECCV. pp. 289–305 (2018) 5
29. Zou, Y., Yu, Z., Liu, X., Kumar, B., Wang, J.: Confidence regularized self-training. In: ICCV. pp. 5982–5991 (2019) 5