– Supplementary Material –
DAFormer: Improving Network Architectures and Training Strategies for Domain-Adaptive Semantic Segmentation

Lukas Hoyer
ETH Zurich
lhoyer@vision.ee.ethz.ch

Dengxin Dai
MPI for Informatics
ddai@mpi-inf.mpg.de

Luc Van Gool
ETH Zurich & KU Leuven
vangool@vision.ee.ethz.ch

A. Overview

The supplementary material provides further details on DAFormer as well as additional experimental results and analysis. In particular, Sec. B provides further implementation details, Sec. C discusses the class statistics of sampling with and without RCS, Sec. D provides an ablation of the training strategies with DeepLabV2, Sec. E studies the parameter sensitivity of RCS and FD, Sec. F ablates the UDA self-training, Sec. G compares the runtime and memory consumption, Sec. H analyzes example predictions, Sec. I compares DAFormer with additional previous UDA methods, and Sec. J discusses limitations of DAFormer.

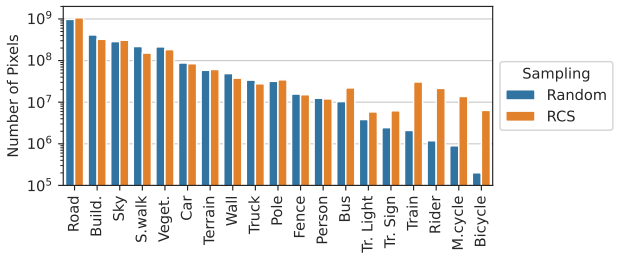
B. Source Code and Further Details

The source code to reproduce DAFormer and all ablation studies is provided at `https://github.com/lhoyer/DAFormer`. Please, refer to the contained `README.md` for further information such as the environment and dataset setup.

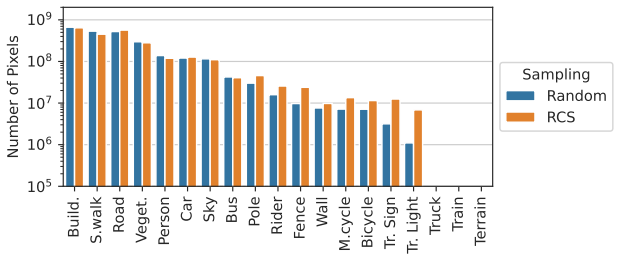
For the distinction of thing- and stuff-classes, we follow the definition by Caesar *et al*. [3]. Applied to Cityscapes, thing-classes are traffic light, traffic sign, person, rider, car, truck, bus, train, motorcycle, and bicycle and stuff-classes are road, sidewalk, building, wall, fence, pole, vegetation, terrain, and sky.

C. Rare Class Sampling Statistics

Most (real-world) datasets have an imbalanced class distribution. This is also the case for the used source datasets as can be seen in Fig. S1 in the blue bars. Please note that the y-axis is scaled logarithmically so that the rare classes are still visible. This problem is addressed by RCS by sampling images with rare classes more often as discussed in Sec. 3.3 of the main paper. Therefore, more pixels of the re-sampled images belong to the rare classes as can be seen in the orange bars of Fig. S1, which directly results in a significantly improved IoU for these rare classes as can be seen



(a) GTA

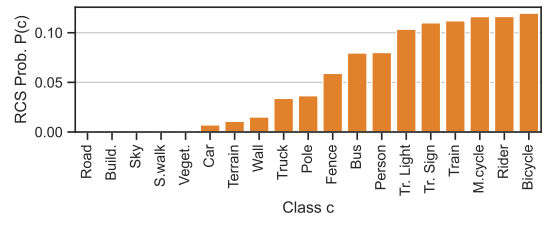


(b) Synthia

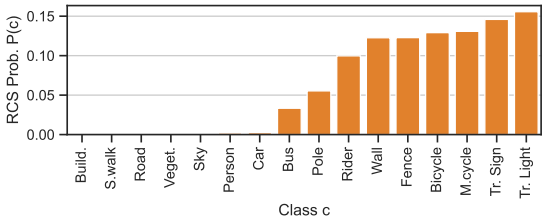
Figure S1. Class statistics of the corresponding dataset for 10k samples. Note that the y-axis is scaled *logarithmically*. RCS samples images with rare classes more often than random sampling.

in Fig. 6 of the main paper. The RCS temperature $T = 0.01$ is chosen to (approximately) maximize the number of re-sampled pixels of the class with the least re-sampled pixels as described in Sec. 3.3 of the main paper. This strategy results in a balance of the number of re-sampled pixels of the classes with the least re-sampled pixels as can be seen in the orange bars of Fig. S1a for the classes *traffic light* and *bicycle*.

Further, Fig. S2 shows the RCS class sampling probabilities $P(c)$ from Eq. 7 in the main paper for the default RCS temperature $T = 0.01$. It can be seen that the class sampling probability for rare classes is higher than for common classes as expected. For some common classes such as *road* and *sky*, $P(c)$ is very close to zero. As these common



(a) GTA



(b) Synthia

Figure S2. RCS class sampling probability $P(c)$ from Eq. 7 in the main paper with an RCS temperature of $T = 0.01$.

Table S1. Ablation of the components of the UDA framework for SegFormer and DeepLabV2. Mean and standard deviation are calculated over 3 random seeds.

Network	Warmup	RCS	FD	mIoU
SegF. [23]	–	–	–	51.8 ±0.8
SegF. [23]	✓	–	–	58.2 ±0.9
SegF. [23]	✓	–	✓	61.7 ±2.6
SegF. [23]	✓	✓	–	64.0 ±2.4
SegF. [23]	✓	✓	✓	66.2 ±1.0
DLv2 [4]	–	–	–	49.1 ±2.0
DLv2 [4]	✓	–	–	54.2 ±1.7
DLv2 [4]	✓	–	✓	55.5 ±1.3
DLv2 [4]	✓	✓	–	55.7 ±0.5
DLv2 [4]	✓	✓	✓	56.6 ±1.2

classes are part of almost every image, it is not necessary to specifically sample images containing these particular common classes.

D. Training Strategies for DeepLabV2

Tab. S1 shows the performance of the improved UDA training strategies for the DeepLabV2 architecture in addition to the SegFormer architecture from the main paper. It demonstrates that learning rate warmup, RCS, FD, and their combination are all beneficial for DeepLabV2 as well. However, the performance improvement for SegFormer is significantly larger than for DeepLabV2, supporting our hypothesis that the network architecture is crucial for UDA performance.

Table S2. Hyperparameter sensitivity study for RCS and FD. The default parameters are marked with *. The evaluation setup is equivalent to row 8 in Tab. 5 of the main paper.

RCS T	mIoU	λ_{FD}	mIoU
0.001	65.6	0.001	65.8
0.002	66.8	0.002	66.2
0.01*	66.8	0.005*	66.8
0.05	66.8	0.01	66.5
0.1	65.5	0.02	65.8

Table S3. Ablation of UDA self-training (ST, see Sec. 3.1 in the main paper) with and without RCS and FD. Warmup is enabled in all configuration. The experiments are conducted with SegFormer [23] and complement Tab. 5 in the main paper.

	w/o (RCS+FD)	w/ (RCS+FD)
w/o ST	45.6±0.6	50.7±0.3
w/ ST	58.2±0.9	66.2±1.0

E. Parameter Sensitivity of RCS and FD

To analyze the sensitivity of the parameters of FD and RCS, Tab. S2 shows a study of T for RCS and λ_{FD} for FD. It can be seen that RCS is stable up to a deviation of factor 5 from the default value. For FD, the weighting is stable up to a factor of about 2. Given that both default values are chosen according to an intuitive strategy (maximization of re-sampled pixels for the class with the least re-sampled pixels for T and gradient magnitude balance for λ_{FD}), the robust range of the hyperparameters is sufficient to select a good value according to the described strategy.

F. Ablation of Self-Training

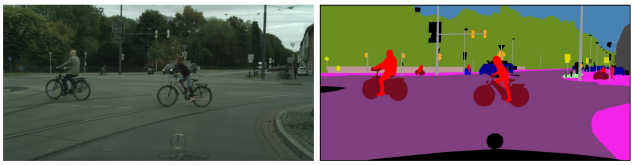
As FD and RCS operate on source data, they can also be used to improve the domain generalization ability of a model trained only on the source domain without self-training (ST) on the target domain. Without ST (row 1 in Tab. S3), RCS and FD increase the network performance by +5.1 mIoU, demonstrating their benefit for domain generalization. Combined with ST (row 2 in Tab. S3), their improvement even increases to +8.0 mIoU showing that RCS and FD reinforce ST, confirming their particular importance for UDA as well.

G. Runtime and Memory Consumption

DAFormer can be trained on a single RTX 2080 Ti GPU within 16 hours (0.7 it/s) while requiring about 9.6 GB GPU memory during training. It has a throughput of 8.7 img/s for inference. In Tab. S4, DAFormer is compared with other network architectures and UDA methods with respect to runtime and memory consumption. Even though DAFormer is heavier than DeepLabV2 and SegFormer, it

Table S4. Runtime and memory consumption of different network architectures and UDA methods on a single RTX 2080 Ti GPU.

UDA Method	Network	Training Throughput (it/s)	Inference Throughput (img/s)	Training GPU Memory	Num. Params
ST	DLv2 [4]	1.24	11.3	5.6 GB	43.2M
ST	SegF. [23]	0.95	8.9	7.7 GB	84.6M
ST+RCS+FD	DLv2 [4]	0.91	11.3	8.6 GB	43.2M
ST+RCS+FD	SegF. [23]	0.75	8.9	8.8 GB	84.6M
ST+RCS+FD	DAFormer	0.71	8.7	9.6 GB	85.2M



(a) Bicycle annotation policy on Cityscapes

(b) Bicycle annotation policy on GTA

Figure S3. Different annotation policies for bicycle on Cityscapes and GTA. It can be seen that the entire wheel is segmented as bicycle for Cityscapes, while only the tire and spokes of the wheel are segmented as bicycle for GTA.

requires only 12% more GPU memory and about 30% more training/inference time than DeepLabV2 when the same UDA configuration is used. When ablating the proposed RCS and FD, the GPU memory consumption is further reduced by 54% and the training time is decreased by 36% mainly due to the additional ImageNet encoder and the feature distance calculation. However, as this is only relevant for training, the inference throughput is the same.

H. Qualitative Analysis

Comparison with ProDA The better performance of DAFormer compared to ProDA [27] is also reflected in example predictions shown in column 4 and 5 of Fig. S4-S9. The major improvements come from a better recognition of the classes *train* (Fig. S4), *bus* (Fig. S5), *truck* (Fig. S6), *car* (Fig. S7), and *sidewalk* (Fig. S8) across different perspectives, object sizes, and appearances. Fig. S9 also shows a better recognition of *rider*, *bicycle*, and *fence*. Furthermore, in the shown examples, it can generally be seen that DAFormer better segments fine structures, which is especially beneficial for small classes such as *pole*, *traffic sign*, and *traffic light*.

Domain Generalization Additionally, column 2 and 3 of Fig. S4-S9 compare the source-only training of DeepLabV2 and SegFormer. It can be seen that SegFormer better generalizes from the source training to the target domain than DeepLabV2. Still, there is a considerable gap between SegFormer with source-only training and DAFormer, showing that the adaptation to the target domain is essential.

Error Cases To give additional insights into the limitations of DAFormer, Fig. S10-S14 show some of the typical error cases. This includes confusion of *sidewalk* and *road* if the texture is similar or there are cycle path markings (Fig. S10), confusion of *wall* and *fence* (Fig. S11), misclassification of some special vans (Fig. S12), misclassification of partly-occluded *busses* (Fig. S13), misclassification of *persons* close to bikes (Fig. S14 top), misclassification of standing *riders* (Fig. S14 bottom), and segmentation of only the bicycle tires (Fig. S14 bottom). That only the tires and not the inside of the wheel of a bicycle are segmented is caused by the annotation policy of GTA (see Fig. S3). Note that for many of the error cases, also previous methods such as ProDA experience similar issues.

I. Comparison with Previous Methods

In the main paper, we compare DAFormer with a selection of representative UDA methods. However, various other UDA methods were proposed in the last few years. A comprehensive comparison with these is shown in Tab. S5 for GTA→Cityscapes and in Tab. S6 for Synthia→Cityscapes. On the one side, some of the newly shown methods can achieve a higher IoU for specific classes than the previous methods shown in the main paper, but on the other side, their performance suffers for other classes. Therefore, ProDA [27] still achieves the best mIoU of the previous state-of-the-art methods. Overall, DAFormer is able to outperform all previous works both in mIoU and classwise IoU for GTA→Cityscapes, often by a considerable margin. On Synthia→Cityscapes, this statement holds except for the stuff-classes *road*, *sidewalk*, *vegetation*, and *sky*. This might be due to the shape-bias of Transformers [2], which causes the network to focus more on shape than texture. The shape bias could improve the generalization ability for thing-classes as their shape is more domain-robust than their texture. However, for stuff-classes, the texture is sometimes crucial to distinguish similar classes such as *road* and *sidewalk* and a shape-bias could be hindering.

The results in Tab. S5 and Tab. S6 are reported with the training configurations used in the original methods, which do not use learning rate warmup. For a fair comparison, we have re-implemented DACS [15] with our training configuration including learning rate warmup, which achieves 54.2 mIoU on GTA→Cityscapes. Still, DAFormer outperforms it by +14.1 mIoU.

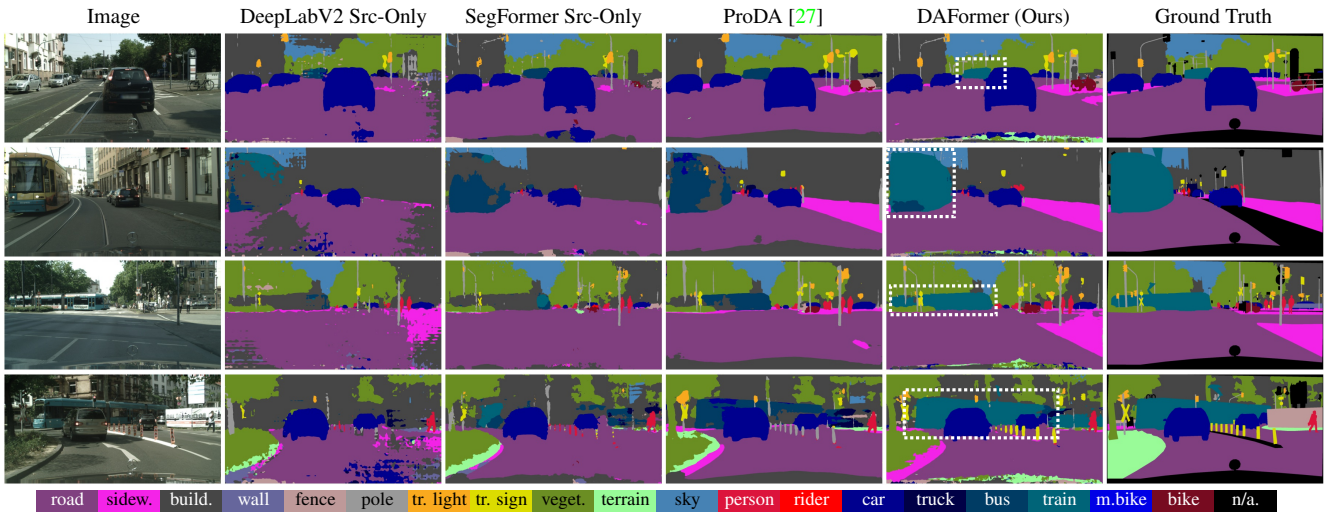


Figure S4. Example predictions showing a better recognition of *train* as opposed to *bus* by DAFormer on GTA→Cityscapes.

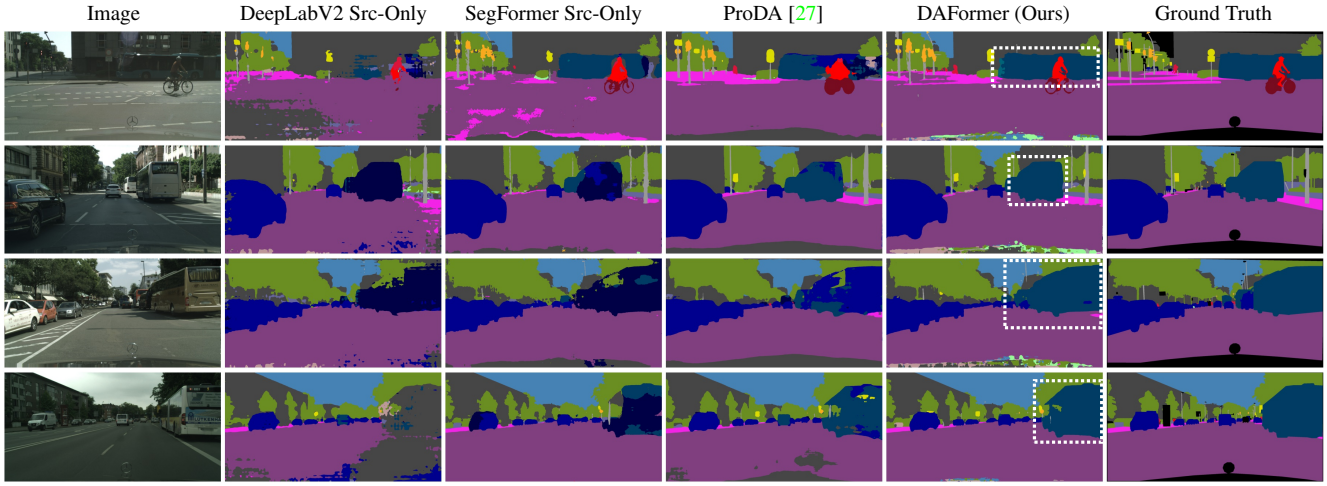


Figure S5. Example predictions showing a better recognition of *bus* by DAFormer on GTA→Cityscapes.

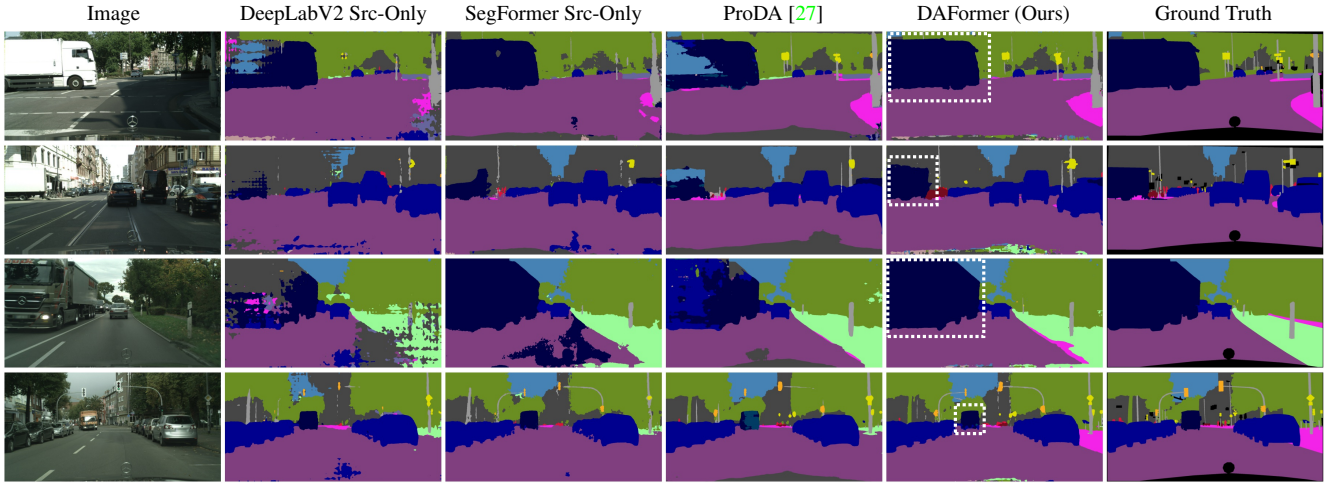


Figure S6. Example predictions showing a better recognition of *truck* by DAFormer on GTA→Cityscapes.

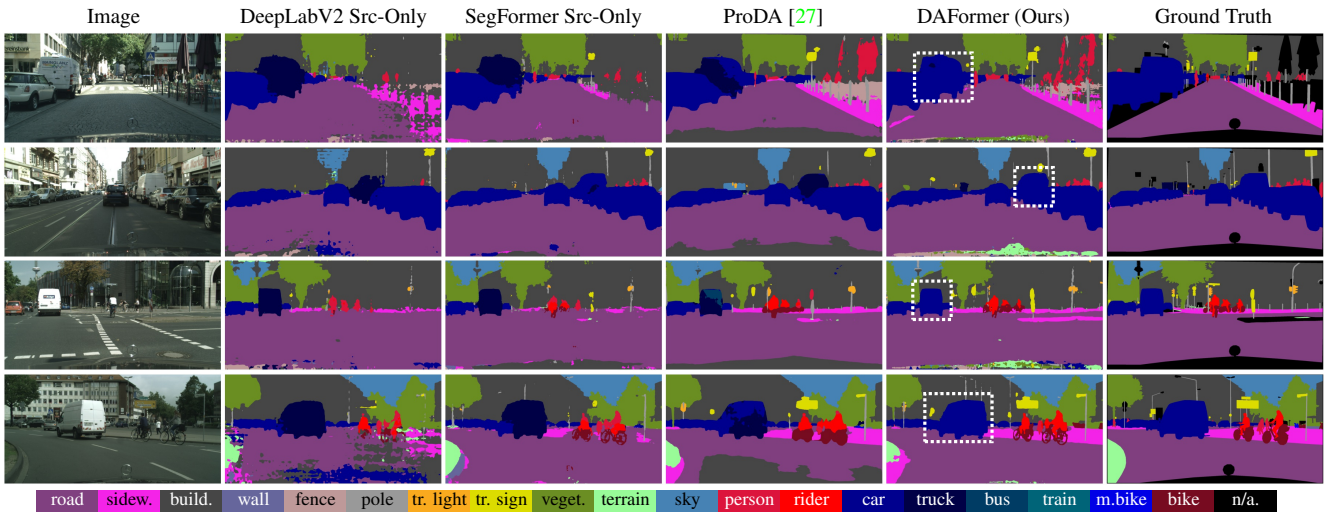


Figure S7. Example predictions showing a better recognition of *car* as opposed to *truck* by DAFormer on GTA→Cityscapes.

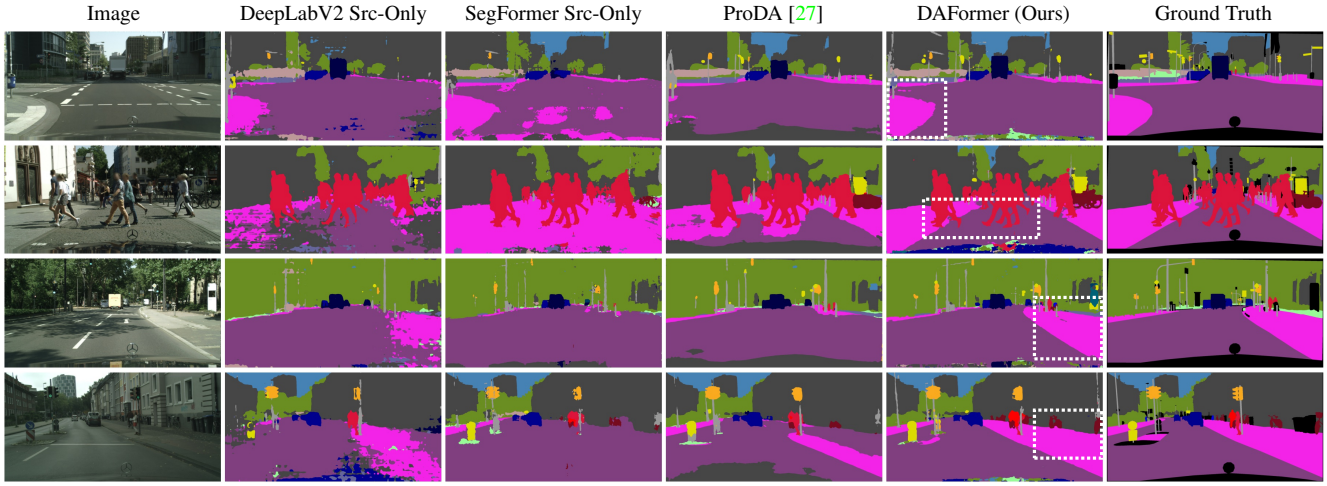


Figure S8. Example predictions showing a better recognition of *sidewalk* as opposed to *road* by DAFormer on GTA→Cityscapes.

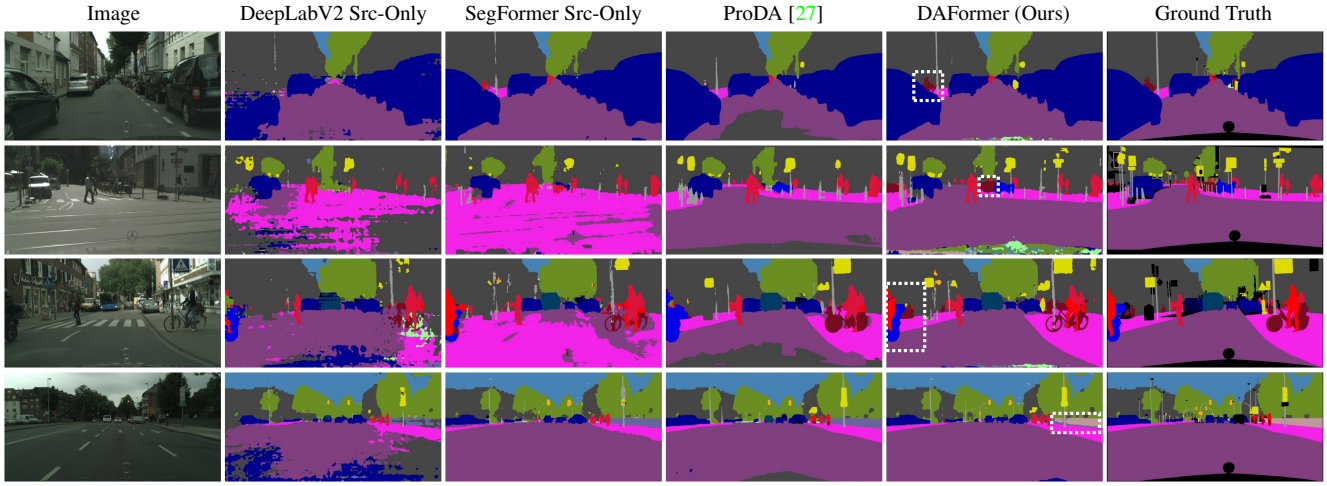


Figure S9. Further example predictions on GTA→Cityscapes showing a better recognition of *bicycle*, *rider*, and *fence*.

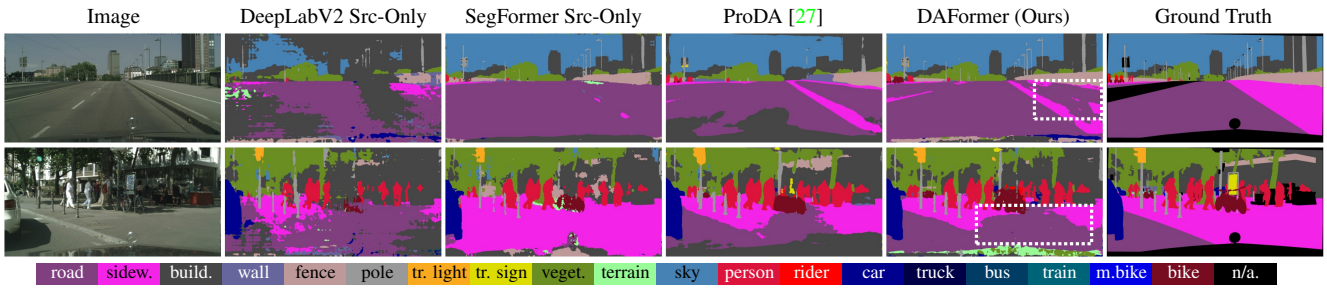


Figure S10. Typical error cases on GTA→Cityscapes: Confusion of *sidewalk* and *road* if the texture is similar.

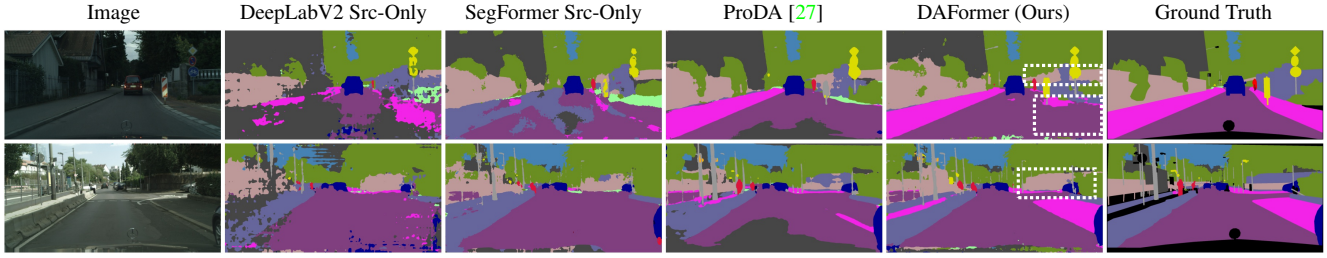


Figure S11. Typical error cases on GTA→Cityscapes: Confusion of *wall* and *fence*.

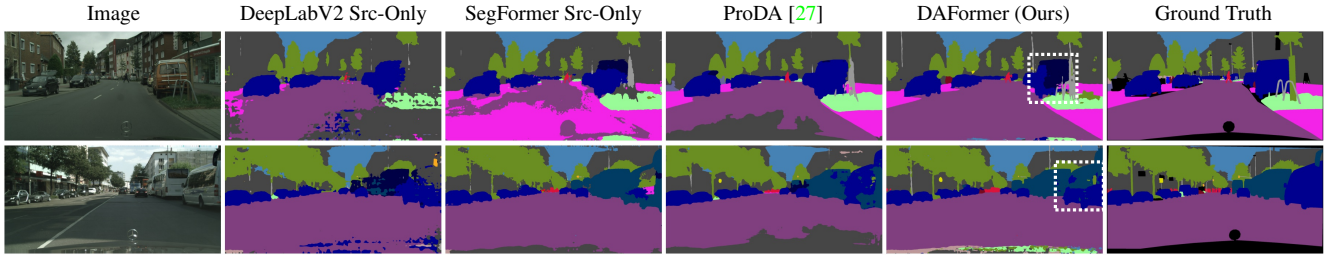


Figure S12. Typical error cases on GTA→Cityscapes: Misclassification of special vans.

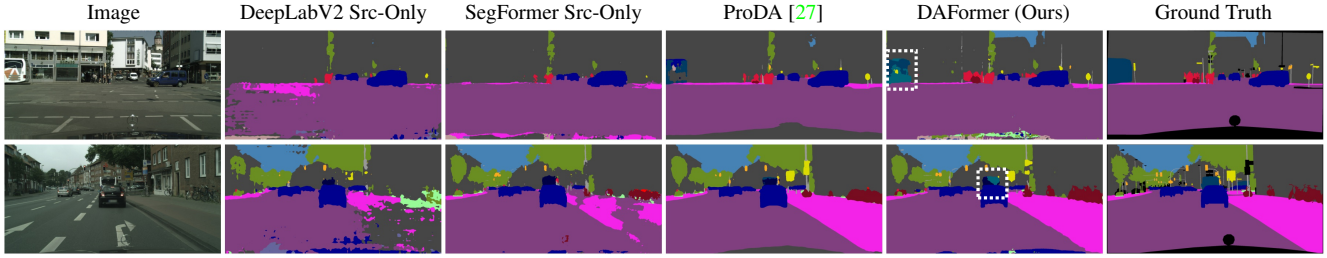


Figure S13. Typical error cases on GTA→Cityscapes: Misclassification of partly-occluded *busses*.

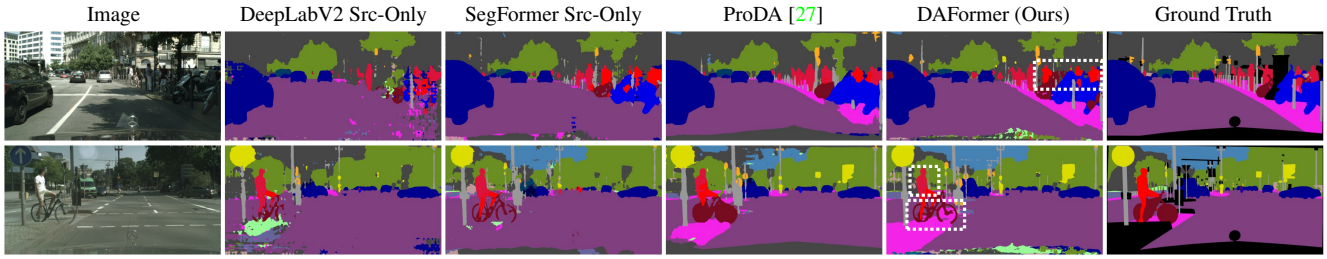


Figure S14. Typical error cases on GTA→Cityscapes: Misclassification of *persons* close to bikes (top row), standing *riders* (bottom row), and missing segmentation of the inside of the *bicycle* wheel (bottom row).

Table S5. Comparison with previous methods on GTA→Cityscapes. Our results (DAFormer) 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 [24]	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 [31]	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 [32]	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 [28]	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. [29]	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 [25]	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
ProDA [27]	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 (Ours)	**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**

Table S6. Comparison with previous methods on Synthia→Cityscapes. Our results (DAFormer) 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 [5]	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 [31]	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 [32]	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 [28]	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 [25]	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 [24]	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. [29]	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
ProDA [27]	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 (Ours)	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**

DAFormer uses the last checkpoint of a training for evaluation. The reported results are averaged over three training runs and have a standard deviation of 0.5 mIoU on both benchmarks, which shows that the training process is stable. We do not use the target validation dataset for checkpoint selection in contrast to some other works [27, 28].

J. Discussion

J.1. Limitations

Due to computational constraints, we only use a selection of network architectures to support our claims. However, there are further interesting architectures that could be explored in the future such as [10, 26]. Also, other training aspects such as larger batch sizes could be of relevance.

In Fig. 3 of the main paper and in the discussion of the Synthia results in Sec. I, there have been some indications that a Transformer architecture is not ideal for stuff-classes. This might be due to the shape-bias of Transformers [2]. The focus on shape instead of texture might be disadvantageous for the distinction of stuff-classes as the texture is an important aspect for their recognition. Even though a further investigation is out of the scope of this work, we believe that this is an interesting aspect for future work, which could potentially lead to a network architecture, even better suited for UDA.

Context-aware fusion assumes that context correlations are domain-invariant. This is often the case for the typical UDA benchmarks [30]. However, this assumption can break down for some special cases in other domains, where the context misleads the model (e.g. misclassification of a cow on road as a horse) [7].

RCS is designed to counter a long-tail data distribution of the source dataset. It is unproblematic for RCS if a class is more common in the target dataset than in the source dataset (e.g. bicycle) as it is balanced by RCS on the source domain and regularly sampled on the target domain. If a class is extremely rare in the target dataset, it might happen that this class is not sampled often enough for efficient adaptation. Therefore, the pseudo-labels would not contain this class and, conceptually, it would not be possible to specifically select samples with this class from the target data.

As shown in the error cases in Sec. H, our method struggles with differences in the annotation policy between source training data and target evaluation data. One example is the bicycle wheel. While the entire wheel is segmented in Cityscapes (see Fig. S3a), only the tires and spokes are segmented in GTA (see Fig. S3b). Also, there are corner cases, where the annotation policy is not defined by source labels such as cycle paths on road (see the top row in Fig. S11) or small busses (see the bottom row in Fig. S12). In order to resolve these issues, additional information about the annotations policy and corner cases would be necessary. A potential solution might be the use of a few target training labels as studied in semi-supervised domain adaptation.

J.2. Potential Negative Impact

Our work improves the adaptability of semantic segmentation, which can be used to enable many good applications such as autonomous driving. However, UDA might also be utilized in undesired applications such as surveillance or military UAVs. This is a general problem of improving semantic segmentation algorithms. A possible countermeasure could be legal restrictions of the use cases for semantic segmentation algorithms.

References

[1] Nikita Araslanov and Stefan Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 15384–15394, 2021. 7

[2] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Int. Conf. Comput. Vis.*, pages 10231–10241, 2021. 3, 8

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1209–1218, 2018. 1

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 2, 3

[5] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1841–1850, 2019. 7

[6] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Int. Conf. Mach. Learn.*, pages 1989–1998, 2018. 7

[7] Lukas Hoyer, Mauricio Munoz, Prateek Katiyar, Anna Khoreva, and Volker Fischer. Grid saliency for context explanations of semantic segmentation. In *Adv. Neural Inform. Process. Syst.*, pages 6462–6473, 2019. 8

[8] Kuan-Hui Lee, German Ros, Jie Li, and Adrien Gaidon. Spigan: Privileged adversarial learning from simulation. In *Int. Conf. Learn. Represent.*, 2018. 7

[9] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6936–6945, 2019. 7

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, pages 10012–1110022, 2021. 8

[11] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2507–2516, 2019. 7

[12] Fengmao Lv, Tao Liang, Xiang Chen, and Guosheng Lin. Cross-domain semantic segmentation via domain-invariant interactive relation transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4334–4343, 2020. 7

[13] Haoyu Ma, Xiangru Lin, Zifeng Wu, and Yizhou Yu. Coarse-to-fine domain adaptive semantic segmentation with photometric alignment and category-center regularization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4051–4060, 2021. 7

[14] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *Eur. Conf. Comput. Vis.*, pages 415–430, 2020. 7

[15] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. DACS: Domain Adaptation via Cross-domain Mixed Sampling. In *IEEE Winter Conf. on Applications of Comput. Vis.*, pages 1379–1389, 2021. 3, 7

[16] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7472–7481, 2018. 7

[17] Yi-Hsuan Tsai, Kihyuk Sohn, Samuel Schulter, and Manmohan Chandraker. Domain adaptation for structured output via discriminative patch representations. In *Int. Conf. Comput. Vis.*, pages 1456–1465, 2019. 7

[18] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2517–2526, 2019. 7

[19] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Dada: Depth-aware domain adaptation in semantic segmentation. In *Int. Conf. Comput. Vis.*, pages 7364–7373, 2019. 7

[20] Haoran Wang, Tong Shen, Wei Zhang, Ling-Yu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *Eur. Conf. Comput. Vis.*, pages 642–659, 2020. 7

[21] Qin Wang, Dengxin Dai, Lukas Hoyer, Olga Fink, and Luc Van Gool. Domain adaptive semantic segmentation with self-supervised depth estimation. In *Int. Conf. Comput. Vis.*, pages 8515–8525, 2021. 7

[22] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-mei Hwu, Thomas S Huang, and Honghui Shi. Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12635–12644, 2020. 7

[23] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *Adv. Neural Inform. Process. Syst.*, 2021. 2, 3

[24] Jihan Yang, Ruijia Xu, Ruiyu Li, Xiaojuan Qi, Xiaoyong Shen, Guanbin Li, and Liang Lin. An adversarial perturbation oriented domain adaptation approach for semantic segmentation. In *AAAI*, pages 12613–12620, 2020. 7

[25] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4085–4095, 2020. 7

[26] Yuhui Yuan, Lang Huang, Jianyuan Guo, Chao Zhang, Xilin Chen, and Jingdong Wang. Ocnet: Object context for semantic segmentation. *Int. J. Comput. Vis.*, pages 1–24, 2021. 8

[27] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12414–12424, 2021. 3, 4, 5, 6, 7, 8

[28] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *Adv. Neural Inform. Process. Syst.*, pages 435–445, 2019. 7, 8

[29] Zhedong Zheng and Yi Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *Int. J. Comput. Vis.*, 129(4):1106–1120, 2021. 7

[30] Qianyu Zhou, Zhengyang Feng, Qiqi Gu, Jiangmiao Pang, Guangliang Cheng, Xuequan Lu, Jianping Shi, and Lizhuang Ma. Context-aware mixup for domain adaptive semantic segmentation. In *IEEE Winter Conf. on Applications of Comput. Vis.*, pages 514–524, 2021. 8

[31] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Eur. Conf. Comput. Vis.*, pages 289–305, 2018. 7

[32] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Int. Conf. Comput. Vis.*, pages 5982–5991, 2019. 7