Train Till You Drop: Towards Stable and Robust Source-free Unsupervised 3D Domain Adaptation — *Supplementary Material* —

Björn Michele[1,2], Alexandre Boulch[1], Tuan-Hung Vu[1], Gilles Puy[1], Renaud Marlet[1,3], and Nicolas Courty[2]

[1] Valeo.ai, Paris, France
[2] CNRS, IRISA, Univ. Bretagne Sud, Vannes, France
[3] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

Overview

In this document, we provide: experiments on the application of **TTYD** to the image modality (Sec. A), additional implementation details (Sec. B), a guarantee of the soundness (Sec. C), and additional ablations: on the parameters to adapt (Sec. D), on alternative distances for the consistency validator $\mathbf{TTYD}_{stop}$ (Sec. E) and on other reference models (Sec. F). We also report the performance of $\mathbf{TTYD}_{stop}$ with other training schemes (Sec. G) and discuss the SFUDA hypothesis for our training scheme (Sec. H). Additionally, we also provide the per-class results and comparison to non-SF UDA approaches (Sec. I), qualitative results (Sec. J), and more details on the datasets and class mappings (Sec. K).

A Application to image modality

While developed for 3D SFUDA, the formulation of **TTYD** appears to be general enough to be used for other modalities. To study this aspect, we conducted experiments on image segmentation. We used the GTA5 dataset [16] as source, and the Cityscapes (City) dataset [4] as target.

This is also an opportunity to evaluate if different models can be used as reference models for the validation. We remark, nevertheless, that it is common practice for image semantic segmentation to keep the ImageNet-pretrained batchnorm frozen during training on the source dataset. We cannot directly use a PTBN version of such source-only models as reference for $\mathbf{TTYD}_{stop}$, in particular because the ImageNet-pretrained batchnorm statistics differ too much from those we would have obtained on the source training set. Therefore, we use a PTBN model built

Table 5: SFUDA for image modality.

Method	Valid. ref. model	GTA5 → City
Source-only		36.8
URMDA [17]		45.1
SFDA [10]		45.8
SDF [24]		49.4
HCL [6]		48.1
DT-ST [27]		52.1
TTYD	PTBN	**53.4**
TTYD	DT-ST	<u>53.2</u>

using a source-only model trained *without* freezing the BN layers [3]. We also test the DT-ST model from [27].

Our results are presented in Tab. 5. We also reach SOTA performances for the GT5→City adaptation pair. As we use the self-training module of DT-ST, we can conclude that, as for 3D SFUDA, the final performance relies on the quality of the self-training starting point, which is provided here by $\mathbf{TTYD}_{core}$.

B Additional implementation details

We use PyTorch for our implementation [15]. The models for NS→SK_{10}, SL→SK_{19}, and NS→PD_8 are trained on a single NVIDIA GeForce RTX 2080 Ti (11 GB) GPU. For SL→SP_{13}, NS→SP_6, and NS→PD_8, we use a split NVIDIA A100-40GB GPU with 20 GB memory.

Code. AdaBN [8] and PTBN [13] were not designed specifically for 3D point clouds; we implemented them. MeanBN is derived from the idea of MixedBN [11] (rather than the code of MixedBN, which requires source data, see just below); we implemented it ourselves. AdaBN, PTBN and MeanBN are hyperparameter-free. For DT-ST [27], we used the official code repository and default parameters, as recommended. Code for SHOT [9], TENT [21] and URMDA [17] was taken from their official repository, with parameters set as described below.

Note on MixedBN and MeanBN. In the main paper, we introduce MeanBN as a SFUDA version of MixedBN [11]. Indeed, MixedBN computes the average running statistics of the source and target datasets by mixing them during the training, which cannot be done in an SFUDA setting. MeanBN just averages (with equal weight) the running statistics from source training and from passing the target data through the source-trained network: it is the average of the running statistics of Source-only and AdaBN.

Parameters selected for SHOT, TENT and URMDA. For SHOT [9], we obtained the best results on the target validation set with a learning rate of 10^{-6} and a balancing hyperparameter of $\beta = 10^{-5}$. For TENT [21] and URMDA [17], we used a learning rate of 10^{-5}. Additionally, as URMDA relies on [28] for setting per-class confidence thresholds, we achieved optimal results with significantly different values for the target portion p, depending on source and target domains: $p = 0.01$ (NS→SK_{10}, NS→WO_{10}), $p = 0.9$ (SL→SK_{19}, SL→SP_{13}, NS→PD_8), $p = 0.1$ (NS→SP_6).

Self-training (ST) propagates and somehow denoises uncertain pseudo-labels. It has been successfully used in UDA [11,19] and SFUDA [27]. Table 3 in the main paper shows the benefits of adding self-training in our context (line $\mathbf{TTYD}_{core}$ vs line **TTYD**).

We used the self-training from [27], which we adapted for point clouds, e.g., regarding augmentations. This self-training handles a confidence level for each class, making sure to also promote rare classes. This allows us to train on the target data, selecting a mostly-correct set of labels while keeping a sufficient balance of rare classes, also preventing collapse, which may occur when focusing mainly on most frequent classes.

Training time. Our stopping criterion $\mathbf{TTYD}_{stop}$ saves a lot of time and computation at the training stage. For example, we stop the training for NS→SK_{10} after 1.1 hr, compared to 6 hrs for a full 20k-iteration training.

The design of our training scheme itself makes it also faster, as there is no costly centroid generation after each epoch like in SHOT [9], where 20k iterations require 30 hrs, or time-consuming surface reconstruction regularization like in SALUDA [11], which is reported to run in 120 hrs. The self-training step then takes about 10 hrs.

GPU memory footprint. Our training scheme is also memory efficient at training time, as only one semantic segmentation network is needed. This is in contrast, *e.g.*, to DT-ST [27], where an additional teacher network is used, or to SALUDA [11], which uses an additional geometric regularization head during training.

C Soundness guarantee

We can show that $\mathbf{TTYD}_{stop}$ is *sound* because the agreement $A(f, g)$ (cf. Eq. (5)), which is bounded by 1, can only take at most $|\mathcal{X}| + 1$ different values. Hence, the number of iterations, as defined by Eq. (7), is bounded by $|\mathcal{X}|$. Also, to check the stopping criterion efficiently, we actually only evaluate Eq. (7) after a fixed number N of iterations (typically, $N = 1000$). Even so, the number of iterations remains bounded, by $N|\mathcal{X}|$. In our experiments, the number of iterations at the stopping point is however much smaller than $N|\mathcal{X}|$, typically between 5 and 10k.

However, it is to be noted that we have no *performance* guarantees, as most UDA and SFUDA methods, including validators [12, 18], whose performance results are generally empirical.

D Ablation: Model parameters to adapt

In Tab. 6, we explore a wide range of possible options concerning the parameters to adapt, some of which are already proposed in the literature [8, 9, 11, 21, 27]. Please note that reported values represent the maximum performance over a training for 20k iterations; a stopping criterion is to be used on top of that.

Although they differ in terms of maximum performance, most adaptation strategies make sense, except adapting the classification layer only (Tab. 6.a). On the contrary, adapting the features in the backbone, including before each layer, is key to the performance, to obtain linearly separable features. Adapting the running statistics online both at train and eval time also is detrimental (Tab. 6.b), probably because it does not "see" enough target data. In the end, we adopt for our method the affine transformations before each batch normalization layer as it performs the best, although adapting the backbone is on average nearly as good. Besides, it reduces the memory footprint as fewer parameters have to be updated (although not reducing gradient computation) and it could facilitate investigations for a deeper understanding of the adaptation.

Table 6: Ablation study

(a) Parameters to adapt. Assuming frozen statistics, parameters to update can be replacement of BN by linear layer, or the backbone weights only (without the classification layer) for different learning rates, or the classification layer only, or the complete network (backbone + classification layer).

Adaptation	BN→ Lin. w/o bias	BN→ Lin. w/ bias	Backbone only 10^{-5}	Backbone only 10^{-6}	Backbone only 10^{-7}	Classif. layer	Backbone+classif. 10^{-5}	Backbone+classif. 10^{-6}	Backbone+classif. 10^{-7}
NS→SK_{10}	44.0	**44.7**	40.3	42.1	42.0†	34.4	41.5	41.4†	35.7†
SL→SK_{19}	27.9†	28.2	27.9	**28.5**	26.7†	22.4	28.1	28.0†	23.3†
SL→SP_{13}	36.1	36.0	31.3	36.6	**36.9**†	34.1	36.6	36.9	30.0†
NS→SP_{6}	61.5	61.4	60.5	**61.5**†	60.9†	60.4	**61.5**	61.4	61.0†

(b) Choice of running statistics for BN layers, either fixed or variable (per-instance norm. at train and eval time, or only at train and fixed at eval).

Adaptation	Fixed statistics source	Fixed statistics target	Fixed statistics mean	Online statistics train +eval	Online statistics train
NS→SK_{10}	44.7	43.4	**45.9**	39.1	43.7*
SL→SK_{19}	**28.2**	26.2	27.4	22.2	26.7*
SL→SP_{13}	**36.0**	30.9	34.4	23.7	26.8*
NS→SP_{6}	**61.4**	59.3	61.1	54.7	60.4*

(c) Class distribution to target, uniform or obtained from source data.

Adaptation	Distribution uniform	Distribution source
NS→SK_{10}	35.0	44.7
SL→SK_{19}	23.8	28.2
SL→SP_{13}	25.6	36.0
NS→SP_{6}	60.7	61.4

Maximum mIoU% over 20k iterations, learning rate 10^{-5} unless otherwise stated.
*: performance strongly fluctuating. †: maximum reached at 20k iterations.

Table 7: Performance of our criterion $\mathbf{TTYD}_{stop}$ and other using soft measurements to select a model being trained over 20k iterations (one model for each 1k iteration increment).

Validator \ Adaptation	NS→ SK_{10}
$\mathbf{TTYD}_{stop}$ (i.e., hard choice A)	**44.5**
$\mathbf{TTYD}_{stop}$ L2	**44.5**
$\mathbf{TTYD}_{stop}$ L1	**44.5**
$\mathbf{TTYD}_{stop}$ Symmetric KL	**44.5**

E Ablation: Other distances for consistency validator

We show in Tab. 7 the results of our stopping criterion using various divergences to measure the agreement (symmetric KL divergence, L1 and L2 norms), instead of the default hard counting of identical predictions. As all different options

Table 8: Performance of our $\mathbf{TTYD}_{stop}$ with different reference models to select a model being trained over 20k iterations (one model for each 1k iteration increment).

Validator \ Adaptation	NS→ SK_{10}	SL→ SK_{19}	SL→ SP_{13}	NS→ SP_6	NS→ WO_{10}	NS→ PD_8
Source-only	34.4	22.3	25.6	60.4	46.1	60.4
TTYD-train (last iter.)	39.2	27.8	28.1	23.4	47.7	60.8
TTYD-train (max. value)	44.7	28.2	36.0	61.4	51.4	64.9
$\mathbf{TTYD}_{stop}$ (i.e., w/ PTBN)	44.5	**28.2**	35.9	61.1	**51.4**	63.3
$\mathbf{TTYD}_{stop}$ w/ AdaBn	44.5	**28.2**	**36.0**	61.1	**51.4**	63.3
$\mathbf{TTYD}_{stop}$ w/ MeanBN	39.0	26.9	32.3	61.1	49.8	60.4
$\mathbf{TTYD}_{stop}$ w/ SHOT [9]	43.8	22.3	29.8	60.4	46.1	63.3
$\mathbf{TTYD}_{stop}$ w/ TENT [21]	43.0	27.4	35.9	**61.4**	50.2	**64.5**
$\mathbf{TTYD}_{stop}$ w/ URMDA [17]	39.0	24.7	25.6	60.4	46.1	60.4
$\mathbf{TTYD}_{stop}$ w/ SHOT + ELR [26]	**44.6**	28.1	32.3	60.4	51.0	63.3
$\mathbf{TTYD}_{stop}$ w/ DT-ST [27]	42.4	26.9	32.3	60.4	49.8	63.3

give the same results we keep the simplest one, the hard counting of identical predictions.

F Ablation: Other reference models

In Tab. 8, we compare the performance of the model selected by $\mathbf{TTYD}_{stop}$ using PTBN as a reference model, against the selection of models using AdaBN and MeanBN as reference models. It can be seen that using PTBN or AdaBN as reference model are mostly equivalent. Using MeanBN is clearly inferior, probably because it is too close to the source-only model: it always selects a model trained for less iterations than our proposed alternatives.

We also tested other models as potential reference models: DT-ST, SHOT+ELR, SHOT, TENT and URMDA. We use the model obtained after 20k iterations as reference model for all these methods. DT-ST and and SHOT-ELR are able to select competitive checkpoints, improving performance over the source-only one in 5 out of the 6 domain adaptation scenarios. Although SHOT suffered from a strong performance degradation during training, and therefore would not be a natural choice as reference model, SHOT allows selection a better performing model that the source-only model in half of the domain adaptation settings, and never select a model performing worse than the source-only one. It is to be noted that PTBN, AdaBN, MeanBN are hyperparameter-free. We use default hyperparameters for DT-ST. For SHOT, TENT, URMDA, we use target-validated hyperparameters to study their potential.

Table 9: Performance of our criterion $\mathbf{TTYD}_{stop}$ to select a SHOT or URMDA model being trained over 20k iterations (one model for each 1k iteration increment).

Adaptation / Validator	NS→ SK_{10}	SL→ SK_{19}	SL→ SP_{13}	NS→ SP_6	NS→ WO_{10}	NS→ PD_8
Source-only	34.4	22.3	25.6	60.4	46.1	60.4
TTYD-train (last iter.)	39.2	27.8	28.1	23.4	47.7	60.8
TTYD-train (max. value)	44.7	28.2	36.0	61.4	51.4	64.9
$\mathbf{TTYD}_{core}$	**44.5**	**28.2**	**35.9**	**61.1**	**51.4**	**63.3**
SHOT last iter.	34.9	18.4	21.7	42.4	37.3	43.7
SHOT max.	42.7	27.9	36.7	61.2	50.1	62.9
SHOT w/ $\mathbf{TTYD}_{stop}$	40.7	27.9	35.9	61.2	50.1	62.9
URMDA last iter.	29.4	25.4	24.5	30.8	42.7	56.9
URMDA max.	37.5	25.5	33.4	63.0	48.4	60.4
URMDA w/ $\mathbf{TTYD}_{stop}$	37.2	25.6	25.6	60.4	46.1	60.4

G $\mathbf{TTYD}_{stop}$ for other training schemes

In Tab. 9 we also apply $\mathbf{TTYD}_{stop}$ to SHOT and URMDA, as both methods are facing strong model degradation during training. We report the maximal achieved performance during training (max.), the performance reached after 20k iterations (last iter.), and the performance reached using our stopping criterion ($\mathbf{TTYD}_{stop}$). We see that our stopping criterion is able to pick a model whose performance is close to the best achieved performance during training (max.).

The application of our stop criterion on TENT does not make sense as the starting point for the TENT method is identical to the reference model.

H SFUDA hypothesis

For our training scheme, we use no source data. Besides a source-only trained model $f[\theta^{\mathbf{s}}]$, we only use global statistics $D^{\mathbf{s}} = D(\mathcal{X}^{\mathbf{s}})$ on source data, i.e., a few class frequencies. These class-wise point ratios are in fact often already provided on dataset datasheets, *e.g.*, SemanticKITTI [1], nuScenes [2]. This very minor requirement complies with motivations of source-free approaches, *e.g.*, privacy, lost access or computation saving. As it can be seen in Tab. 10: alternatives to our prior (D^S) in Eq. (2) (main paper) do not perform well on NS→SK_{10}. However, the correct target class data distribution (D^T), which of course is not available, but could be seen of a kind of oracle, helps to further improve the performance.

I Classwise results and related approaches

In this section, we detail classwise results of semantic segmentation after domain adaptation. We also compare to UDA methods.

Table 10: Comparison of different priors in Eq. (2) on NS→SK_{10}. For easier comparison we report the maximal obtained performance with our training scheme without the selection of $\mathbf{TTYD}_{stop}$.

KL($D(P)$|| ?)	unif.	$D(f[\theta^s](\mathcal{X}^t))$	D^s(ours)	D^t (oracle)
Ours (mIoU%)	35.0	34.4	44.7	47.0

Per-class results. We provide in Tabs. 11 to 16 the classwise results for methods and domain adaptation settings reported in Tab. 2 of the main paper. It can be seen that the gain in performance (mIoU) achieved by our $\mathbf{TTYD}_{core}$ originates, on all dataset settings, from a consistent improvement over a broad range of classes, not just a few of them.

UDA (with source data) as a kind of SFUDA upper bound. General UDA is privileged over the SFUDA setting because it has access to the source data at training time. UDA resutls thus represents a kind of upper bound to SFUDA's. To analyze this aspect, we compare to two state-of-the-art UDA methods, namely CoSMix [19] and SALUDA [11], on the domain adaptation settings we experimented with and for which UDA results are available, i.e., NS→SK_{10}, SL→SK_{19}, SL→SP_{13} and NS→SP_6.

Please note that CoSMix has hyperparameters, which have to be (and are) optimized for each setting on the ground-truth target validation set (which somewhat detracts from the lack of supervision). On the contrary, SALUDA uses an unsupervised validator (Entropy [12]), like we do with our own unsupervised stopping criterion and validator.

As can be seen in Tabs. 11 to 12, although CoSMiX and SALUDA do have a better mIoU on average, our method $\mathbf{TTYD}_{core}$ still outperforms CoSMix on 2/4 domain adaptations and is only 1.8 to 4.7 percentage points behind SALUDA, except on SL→SP_{13}, where SALUDA remains 7.0 p.p. ahead. **TTYD** reduces the gaps with SALUDA down to 0.8 to 3.8 p.p., and even outperforms SALUDA by 1.2 p.p. on SL→SK_{19}.

Please note that we compare to values reported in the SALUDA paper [11], including for CoSMix [19], as the evaluation protocol in [19] for mIoU calculation differs from the official evaluation metric [1], which we use instead. Furthermore, [11] report results as an average over 3 runs, whereas we provide here only the results of a single run.

Table 11: Classwise results for NS→SP$_6$. † from [11].

NS→SP$_6$ (% IoU)	% mIoU	Person	Bike	Car	Ground	Vegetation	Manmade
Strict SFUDA							
Source-only	60.4	56.1	7.5	**65.0**	**79.4**	79.0	**75.7**
AdaBN [8]	57.7	**58.8**	**14.9**	42.8	76.8	79.2	73.7
PTBN [13]	54.7	55.2	10.5	41.0	75.7	74.8	70.9
MeanBN [11]	60.9	58.6	12.4	60.7	78.0	80.0	75.5
TTYD$_{core}$ (ours)	**61.1**	57.0	11.3	64.2	79.0	**80.6**	74.4
Loose SFUDA							
SHOT [9]	42.4	19.0	0.0	13.3	78.7	71.6	72.1
TENT [21]	45.1	36.0	0.1	35.9	76.1	62.0	60.5
URMDA [17]	30.8	36.2	7.7	2.6	71.1	26.2	41.1
SHOT+ELR [26]	59.4	54.0	1.2	67.0	79.9	78.3	75.9
DT-ST [27]	63.1	59.8	7.6	72.9	**81.0**	79.2	78.2
TTYD (ours)	**64.5**	**61.0**	**10.4**	**74.5**	80.9	**81.6**	**78.8**
UDA methods with src data and (for CoSMix) parameters							
CoSMix† [19]	65.2	60.3	24.1	66.4	80.4	81.4	78.3
SALUDA† [11]	65.8	59.0	20.5	70.6	82.6	81.4	81.0

Table 12: Classwise results for NS→SK$_{10}$. † from [11].

NS→SK$_{10}$ (% IoU)	% mIoU	Car	Bicycle	Motorcycle	Truck	Other vehicle	Pedestrian	Driveable surf.	Sidewalk	Terrain	Vegetation
Strict SFUDA											
Source-only	34.4	77.5	8.8	18.3	5.7	4.6	**52.0**	38.8	25.6	29.7	83.2
AdaBN [8]	39.9	80.8	14.5	16.7	8.6	3.8	23.8	**75.0**	**38.9**	**52.9**	84.0
PTBN [13]	39.4	80.0	14.7	27.0	7.3	5.5	23.2	71.3	35.4	48.8	80.6
MeanBN [11]	41.7	87.0	**17.6**	29.6	12.1	4.4	43.8	61.3	33.3	40.2	**87.5**
TTYD$_{core}$ (ours)	**44.5**	**87.4**	7.8	**30.1**	**16.6**	**8.3**	50.1	71.9	33.2	51.9	87.3
Loose SFUDA											
SHOT [9]	34.9	90.2	1.2	8.6	20.9	6.2	1.2	68.9	19.0	**60.4**	72.3
TENT [21]	37.9	58.4	0.1	4.6	**43.1**	10.2	41.6	66.1	20.3	57.8	76.4
URMDA [17]	29.4	72.0	1.4	3.4	3.3	3.1	18.3	36.4	**36.8**	41.4	78.0
SHOT+ELR [26]	40.5	90.1	**2.8**	18.2	16.2	**10.6**	44.9	69.3	15.8	51.2	86.1
DT-ST [27]	35.6	88.6	0.0	26.3	9.1	4.1	**54.9**	39.9	17.2	29.2	87.2
TTYD (ours)	**45.4**	**92.4**	0.0	**37.0**	26.9	2.1	49.0	**72.8**	27.7	56.3	**89.7**
UDA methods with source data and (for CoSMix) hyperparameters											
CoSMix† [19]	38.3	77.1	10.4	20.0	15.2	6.6	51.0	52.1	31.8	34.5	84.8
SALUDA† [11]	46.2	89.8	13.2	26.2	15.3	7.0	37.6	79.0	50.4	55.0	88.3

Table 13: Classwise results for SL→SK$_{19}$. † from [11].

SL→SK$_{19}$ (% IoU)	%mIoU	Car	Bicycle	Motorcycle	Truck	Other vehicle	Pedestrian	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic sign
Strict SFUDA																				
Source-only	22.3	40.7	7.6	9.6	1.5	1.7	21.0	**47.1**	1.6	21.9	4.7	34.0	0.0	36.3	22.2	62.3	28.3	**48.5**	28.8	5.6
AdaBN [8]	24.6	**64.2**	8.5	9.1	2.9	3.3	20.8	27.0	0.4	56.5	**6.8**	30.5	0.0	64.9	17.8	59.2	19.2	36.6	28.0	11.5
PTBN [13]	22.4	53.5	6.5	**11.2**	**4.7**	**3.5**	18.8	30.4	0.3	52.4	3.9	33.2	0.0	58.5	14.4	45.3	20.2	32.7	25.7	10.4
MeanBN [11]	26.9	59.6	9.1	9.8	2.4	3.1	23.6	37.3	1.2	42.5	**6.8**	**34.0**	0.1	60.2	**28.8**	68.9	29.3	42.3	38.0	14.5
TTYD$_{core}$ (ours)	**28.2**	63.9	**11.1**	11.0	3.6	3.0	**26.5**	33.0	**1.7**	**63.2**	5.9	32.3	**0.2**	**67.4**	19.1	**72.6**	**30.5**	35.4	**40.9**	**15.2**
Loose SFUDA																				
SHOT [9]	18.4	49.5	1.0	2.1	4.5	4.2	13.7	8.0	0.5	60.0	4.2	24.0	**0.5**	46.5	16.7	38.0	22.8	15.1	37.4	0.9
TENT [21]	24.5	57.8	3.3	9.5	**12.4**	2.5	11.7	20.3	0.0	52.0	0.3	34.2	0.0	60.8	15.6	66.9	29.9	44.4	40.6	3.5
URMDA [17]	25.4	52.0	3.3	6.3	1.3	1.1	14.7	52.0	1.2	26.2	**5.6**	**37.0**	0.1	46.3	**32.3**	65.3	35.8	51.6	45.8	4.7
SHOT+ELR [26]	27.1	56.7	4.1	10.0	3.3	1.7	31.4	32.7	1.0	62.1	2.8	33.7	0.1	64.9	7.6	71.9	32.3	40.0	46.2	12.2
DT-ST [27]	23.5	34.9	2.1	10.9	2.3	2.0	29.2	**66.7**	1.0	20.6	3.2	35.1	0.0	27.8	5.4	60.4	30.7	**52.9**	48.8	12.6
TTYD (ours)	**32.4**	**77.0**	**5.0**	**12.8**	8.7	**2.9**	**40.0**	43.6	**1.2**	**67.4**	5.5	34.8	0.0	**70.8**	8.4	**77.5**	**40.4**	38.6	**52.8**	**28.1**
UDA methods with source data and (for CoSMix) hyperparameters																				
CoSMix† [19]	28.0	63.9	5.6	11.4	5.7	7.9	20.0	40.3	3.8	56.4	13.2	37.9	0.1	42.6	29.5	66.9	27.9	29.6	46.0	22.5
SALUDA† [11]	31.2	65.4	7.5	13.6	3.2	5.9	23.9	43.7	1.7	52.9	11.6	39.8	0.3	67.8	28.2	74.2	37.6	43.6	47.5	22.7

Table 14: Classwise results for SL→SP$_{13}$. † from [11] and uses a voxel size of 5 cm.

SL→SP$_{13}$ (% IoU)	%mIoU	Person	Rider	Car	Trunk	Plants	Traffic sign	Pole	Garbage can	Building	Cone	Fence	Bike	Ground
Strict SFUDA														
Source-only	25.6	43.2	31.4	22.5	20.8	65.8	1.0	4.5	14.9	53.9	7.0	21.5	3.0	43.4
AdaBN [8]	25.4	38.4	17.8	22.4	23.6	55.9	**13.0**	7.8	8.8	61.1	6.9	14.9	**9.3**	50.9
PTBN [13]	23.7	36.3	20.4	27.0	19.9	43.4	10.6	6.8	8.2	58.8	5.2	15.3	8.5	47.7
MeanBN [11]	27.7	38.9	23.2	22.5	26.2	69.5	6.1	7.0	15.6	63.2	9.4	21.2	5.2	52.2
TTYD$_{core}$ (ours)	**35.9**	**46.1**	**37.2**	**43.5**	**31.3**	**71.3**	4.8	**20.5**	**21.8**	**69.1**	**11.5**	**25.4**	4.3	**79.9**
Loose SFUDA														
SHOT [9]	21.7	31.1	5.7	11.8	32.9	37.1	8.0	18.5	4.6	52.3	6.2	18.1	0.1	55.3
TENT [21]	28.3	39.1	30.0	33.4	20.0	63.3	0.0	21.4	3.0	60.0	16.8	31.6	**0.7**	48.7
URMDA [17]	24.5	42.0	37.7	**50.3**	23.5	46.1	0.0	21.5	0.0	41.9	0.0	**51.7**	0.0	3.4
SHOT+ELR [26]	36.9	59.8	29.1	47.7	**30.4**	71.1	1.3	23.1	12.1	70.9	**18.4**	34.4	0.4	**81.9**
DT-ST [27]	36.8	**64.1**	**57.1**	47.3	21.5	65.3	3.6	23.6	**28.3**	58.5	6.2	35.1	0.3	67.1
TTYD (ours)	**39.1**	**64.1**	54.8	48.9	27.8	**73.0**	**8.8**	**29.4**	14.1	**73.6**	5.9	36.8	0.5	70.7
UDA methods with source data and (for CoSMix) hyperparameters														
CoSMix† [19]	40.8	50.9	54.5	34.9	33.6	71.1	19.4	35.6	26.8	65.2	30.4	24.0	6.0	78.5
SALUDA† [11]	42.9	59.9	54.6	59.2	33.7	69.8	14.9	40.9	30.8	64.5	26.2	22.1	2.7	78.0

Table 15: Classwise results for NS→WO$_{10}$.

NS→WO$_{10}$ (% IoU)	% mIoU	Car	Bicycle	Motorcycle	Truck	Other vehicle	Pedestrian	Driveable surf.	Sidewalk	Walkable	Vegetation
Strict SFUDA											
Source-only	46.1	72.2	6.2	14.0	24.9	24.5	68.1	70.8	47.8	43.8	88.6
AdaBN [8]	47.7	70.5	8.9	9.1	27.6	33.2	58.8	**82.2**	51.5	46.4	89.0
PTBN [13]	42.3	65.1	4.5	7.7	21.7	22.1	51.8	80.3	46.4	40.4	83.3
MeanBN [11]	50.3	75.2	**9.6**	12.8	**30.0**	**37.2**	67.5	78.5	52.2	**48.9**	**91.5**
TTYD$_{core}$ (ours)	**51.4**	**77.5**	7.6	**17.3**	27.5	36.1	**74.2**	80.3	**53.8**	48.4	91.1
Loose SFUDA											
SHOT [9]	37.3	56.2	0.8	7.6	15.2	21.7	36.9	61.7	45.9	41.1	85.7
TENT [21]	40.4	56.5	0.4	10.9	18.3	23.8	52.1	82.2	47.8	35.5	76.2
URMDA [17]	42.7	71.9	1.7	1.3	26.2	20.6	60.2	64.9	52.1	41.5	86.5
SHOT+ELR [26]	49.5	79.5	2.2	**24.0**	26.2	29.0	67.6	76.5	51.9	50.0	88.1
DT-ST [27]	51.8	81.0	6.8	18.9	**33.1**	42.9	77.6	72.1	47.5	45.7	92.7
TTYD (ours)	**55.5**	**83.1**	**8.4**	20.4	**33.1**	**46.0**	**79.5**	**82.2**	**55.4**	**53.0**	**93.5**

Table 16: Classwise results for NS→PD$_8$.

NS→PD$_8$ (% IoU)	% mIoU	2-wheeled	Pedestrian	Driveable ground	Sidewalk	Other ground	Manmade	Vegetation	4-wheeled
Strict SFUDA									
Source-only	60.4	27.6	64.2	71.6	45.1	24.2	88.1	75.0	87.2
AdaBN [8]	59.6	31.3	51.6	77.3	44.5	28.5	86.0	73.1	84.3
PTBN [13]	60.2	**32.4**	52.3	76.1	46.0	28.3	86.9	74.1	85.6
MeanBN [11]	61.3	31.3	61.6	75.0	44.8	27.0	87.8	75.0	87.5
TTYD$_{core}$ (ours)	**63.3**	28.8	**65.3**	**78.1**	**49.0**	**30.5**	**88.2**	**76.2**	**90.4**
Loose SFUDA									
SHOT [9]	43.7	0.7	38.4	27.7	40.1	17.1	84.5	67.8	72.5
TENT [21]	59.1	14.8	50.5	**83.6**	**50.8**	25.8	85.5	72.7	89.2
URMDA [17]	56.9	17.0	62.2	68.9	40.1	22.6	88.5	71.9	84.9
SHOT+ELR [26]	60.9	15.2	58.5	78.1	48.3	30.0	88.8	77.4	90.8
DT-ST [27]	62.5	32.7	**64.2**	75.9	43.8	26.6	**89.1**	77.5	90.4
TTYD (ours)	**65.7**	**35.2**	**64.2**	81.7	49.5	**35.9**	88.4	**78.3**	**92.9**

J Qualitative results

Methods with no degradation prevention. We illustrate in Fig. 3 the performance degradation when training is too long for TENT [21], SHOT [9] and URMDA [17]. Note that, for these methods, we select the best trained model by looking at the ground-truth target validation set. It highlights the difference between what can be achieved in theory and what actually happens if training is not stopped with a criterion like ours.

One can observe that the TENT model, which estimates the normalization parameters of the batch norm layers on the target dataset, starts from a better source-only model, although it has not been trained on target data yet. After 20k iterations, the motorcycle, the truck, and part of the vegetation are not correctly classified, although they were correctly classified in the source-only model. A similar degradation behavior can be seen for the SHOT method. The URMDA method does not perform as well as the others. After 20k iterations, it also shows a significant degradation with respect to both the source-only starting point and the best model: while the source-only model correctly segments the vegetation and the truck, the final model incorrectly labels part of the vegetation using various other classes, and wrongly predicts the class on the top of the truck.

Our stopping criterion. In Fig. 4, we show qualitative results for each domain adaptation setting: ground-truth labels (GT), the source-only result, the result obtained by our training scheme with $\mathbf{TTYD}_{stop}$, and the result obtained after 20k iterations. These representations highlight that the stopping criterion achieves a significant, qualitatively visible improvement.

As can be seen, the improvements of our training scheme in combination with our stopping criterion over the source-only model are dominated by changes in the "Road", "Sidewalk", and "Terrain" classes. If the training is pushed to 20k iterations, these large classes are little degraded, while objects of other classes like cars or pedestrians can be totally misclassified. One exception is the NS$\rightarrow$SP$_6$ setting, where we can observe a total collapse into a binary classification after training for 20k iterations.

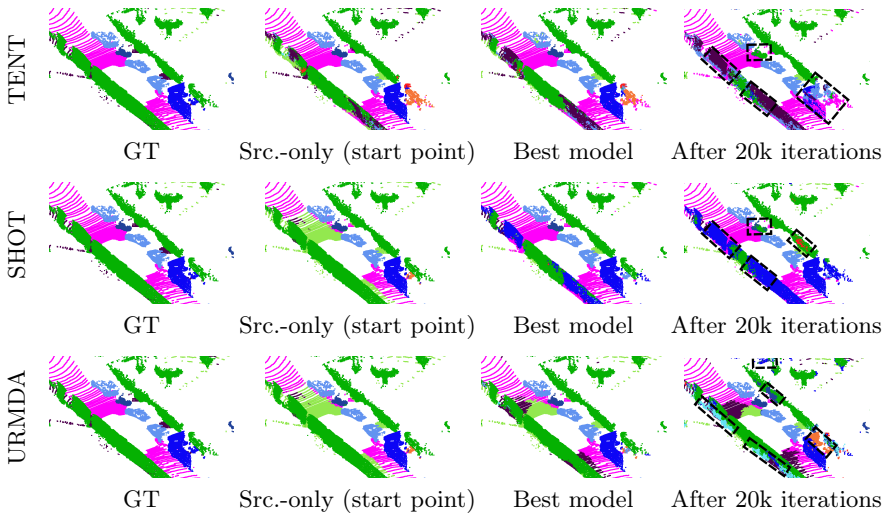

Fig. 3: Examples of results with TENT [21], SHOT [9] and URMDA [17] on NS→SK_{10}: ground truth (GT), initial model trained only on source data, best model as upper bound (using ground-truth knowledge of the target validation set), and "full" training for 20k iterations. "Ignore" points are removed for a better visualisation. Notable errors due to degradation are marked with a dashed rectangle.

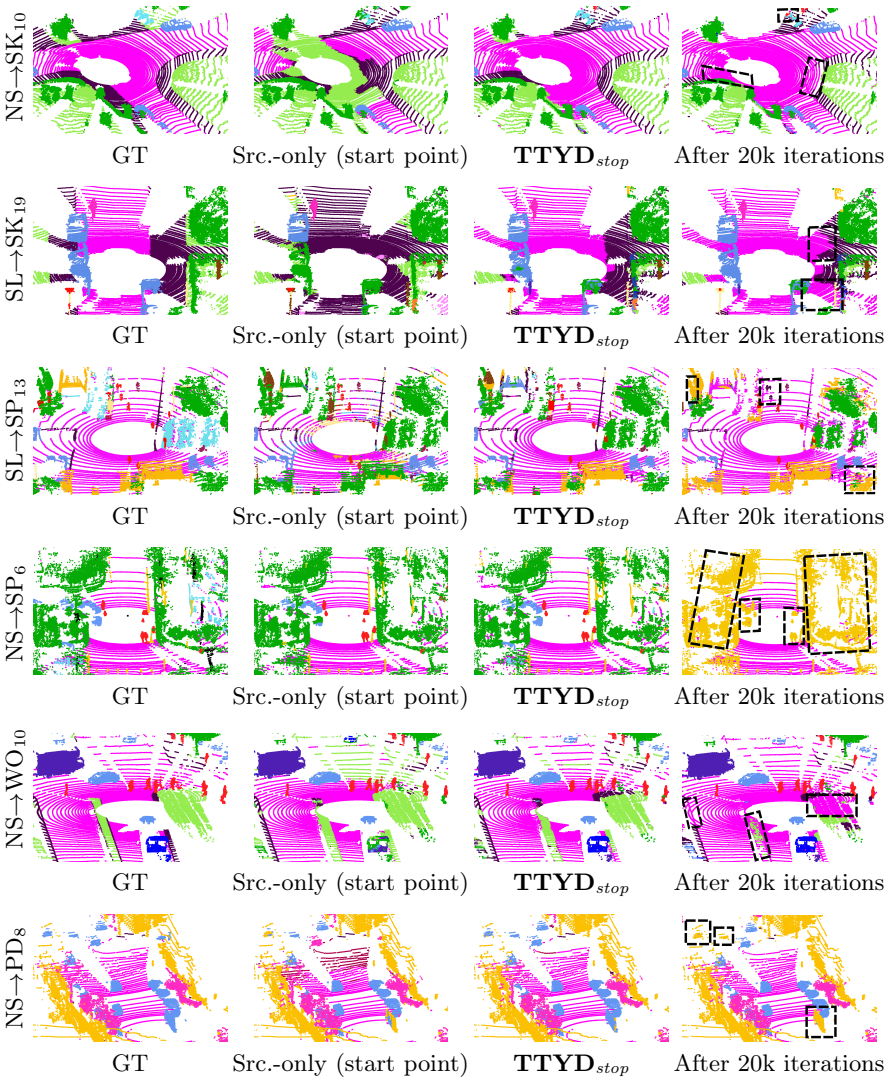

Fig. 4: Examples of results with $\mathbf{TTYD}_{stop}$: ground truth (GT), initial model trained only on source data, training with our training scheme when using our stopping criterion, and "full" training for 20k iterations. "Ignore" points are removed for a better visualisation. Notable errors due to degradation are marked with a dashed rectangle. Due to different class mappings, coloring can vary between the different settings.

K Datasets and class mappings

Tab. 17 summarizes the main characteristics of the datasets we used in experiments, including details about the lidars used for data capture. As can be seen, there is a lot of variety among the lidar sensors, not counting variations that are not even reported here, such as sensor height or laser range. This sensor gap yields significant dissimilarities at point cloud level. Considering on top of that the geographical variety of the driving landscapes over 3 continents, including synthetic scenery, the total domain gap between most of these datasets can be considered as severe.

Note that the number of classes we report is the number used for the standard benchmarking of semantic segmentation on each dataset, which may be lower than the number of finer-grained classes actually annotated in the ground-truth data. Also, for SemanticKITTI, the class of a moving object is merged with the class of the same static object.

In Tabs. 18 to 23, we provide the exact class mapping. Unnamed classes are mapped to 'Ignore'.

Table 17: Datasets used in our domain adaptation experiments. For each dataset, we provide: abbreviation in the paper, main reference, lidar sensor used for data capture, number of beams, vertical field of view (V. FoV), vertical resolution (V. res.), horizontal resolution (H. res.), number of classes used for standard benchmarking (which may be lower than the number of finer-grained actually annotated classes), number of frames for training and/or testing, and region of the world where the data was captured. The V. FoV of the Pandora (Pandar40) lidar is variable, denser when closer to horizontality: 0.33° for the FoV -6° to +2°, and 1° for the FoV -16° to -6° and +2° to +7°. The V. FoV of the Pandar64 is even more variable: 0.167° (-6° to +2°), 1° (-14° to -6°, +2° to +3°), 2° (+3° to +5°), 3° (+5° to +11°), 4° (+11° to +15°), 5° (-19° to -14°), 6° (-25° to -19°).

Dataset		Ref.	Lidar	Beams	V. FoV	V. res.	H. res.	Classes	Train	Test	Region of the world
nuScenes	(NS)	[2]	Velodyne HDL-32E	32	-30.7° to +10.7°	1.33°	0.33°	16	28,130	–	Boston, Singapore
SynLiDAR	(SL)	[22]	*synthetic*	64	-25.0° to + 3.0°			22	19,840	–	3D experts using Unreal Engine 4
SemanticPOSS	(SP)	[14]	Pandora (Pandar40)	40	-16.0° to + 7.0°	0.20°	0,33°/1°	14	2,484	499	Peking University (many dynamic objects)
SemanticKITTI	(SK)	[1]	Velodyne HDL-64E	64	-24.8° to + 2.0°	0.42°	0.18°	19	19,130	4,071	Karlsruhe
Pandaset	(PD)	[23]	Pandar64	64	-25.0° to +15.0°	0.17°	0.20°/6°	37	3,800	2,280	San Francisco, El Camino Real
Waymo Open	(WO)	[5]	Laser Bear Honeycomb	64	-17.6° to + 2.4°			23	23,691	5,976	Phoenix, San Francisco, Mountain View

Table 18: Class mapping for NS→SK_{10} (from [25]).

nuScenes	NS→SK_{10}	SemanticKITTI
Car	Car	Car
Bicycle	Bicycle	Bicycle
Motorcycle	Motorcycle	Motorcycle
Truck	Truck	Truck
Construction vehicle, Bus	Other vehicle	Other-vehicle, Bus
Pedestrian	Pedestrian	Person
Driveable Surface	Driveable surface	Road, Parking, Lane marking
Sidewalk	Sidewalk	Sidewalk
Terrain	Terrain	Terrain
Vegetation	Vegetation	Vegetation, Trunk

Table 19: Class mapping for NS→SP_6 (from [20]).

nuScenes	NS→SP_6	SemanticPOSS
Pedestrian	Person	Person
Bicycle, Motorcycle	Bike	Rider, Bike
Car, Bus, Constriction vehicle, Trailer,Truck	Car	Car
Driveable surface, Other flat, Sidewalk, Terrain	Ground	Ground
Vegetation	Vegetation	Plants
Barrier, Manmade, Traffic cone	Manmade	Traffic sign, Pole, Garbage can, Building, Cone/Stone, Fence

Table 20: Class mapping for NS→WO_{10} (from [7]).

nuScenes	NS→WO_{10}	Waymo Open
Pedestrian	Person	Person
Bicycle, Motorcycle	Bike	Rider, Bike
Car, Bus, Constriction Vehicle, Trailer,Truck	Car	Car
Driveable Surface, Other Flat, Sidewalk, Terrain	Ground	Ground
Vegetation	Vegetation	Vegetation, Plant
Barrier, Manmade, Traffic Cone	Manmade	Traffic Sign, Pole, Garbage Can, Building, Cone/Stone, Fence

Table 21: Class mapping for NS→PD_8 (from [20]).

nuScenes	NS→PD_8	Pandaset
Bicycle, Motorcycle	2-wheeled	Bicycle, Motorcycle, Motorized scooter Pedicab, Personal Mobility Device
Pedestrian	Pedestrian	Pedestrian, Pedestrian w/ objects
Driveable ground	Driveable ground	Driveway, Road, Road marking
Sidewalk	Sidewalk	Sidewalk
Other flat, Terrain	Other ground	Ground
Barrier, Manmade, Traffic cone	Manmade	Building, Cones, Construction Barriers/Signs, Other static object, Pylons, Road Barriers, Rolling containers, Signs
Vegetation	Vegetation	Vegetation
Bus, Car, Construction vehicle, Trailer, Truck	4-wheeled	Car, Construction vehicle, Emergency vehicle, Bus, Towed object, Truck (all kinds of)
		Uncommon vehicle

Table 22: Class mapping for SL→SK_{19} (from [19]).

SynLiDAR	SL→SK_{19} & SemanticKITTI
Car	Car
Bicycle	Bicycle
Motorcycle	Motorcycle
Truck	Truck
Bus, Other vehicle	Other vehicle
Person	Pedestrian
Bicyclist	Bicyclist
Motorcyclist	Motorcyclist
Road	Road
Parking	Parking
Sidewalk	Sidewalk
Other ground	Other ground
Building	Building
Fence	Fence
Vegetation	Vegetation
Trunk	Trunk
Terrain	Terrain
Pole	Pole
Traffic sign	Traffic sign

Table 23: Class mapping for SL→SP_{13} (from [19]).

SynLidar	SL→SP_{13} & SemanticPOSS
Person	Person
Bicyclist,Motorcyclist	Rider
Car	Car
Trunk	Trunk
Vegetation	Plants
Traffic sign	Traffic sign
Pole	Pole
Garbage can	Garbage can
Building	Building
Traffic-cone	Cone
Fence	Fence
Bicycle	Bike
Road	Ground

References

1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In: ICCV (2019)
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: CVPR (2020)
3. Chen, M., Xue, H., Cai, D.: Domain adaptation for semantic segmentation with maximum squares loss. In: ICCV (2019)
4. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
5. Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., Yang, Z., Chouard, A., Sun, P., Ngiam, J., Vasudevan, V., McCauley, A., Shlens, J., Anguelov, D.: Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In: ICCV (2021)
6. Huang, J., Guan, D., Xiao, A., Lu, S.: Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. In: NeurIPS (2021)
7. Kim, H., Kang, Y., Oh, C., Yoon, K.J.: Single domain generalization for lidar semantic segmentation. In: CVPR (2023)
8. Li, Y., Wang, N., Shi, J., Hou, X., Liu, J.: Adaptive batch normalization for practical domain adaptation. PR **80** (2018)
9. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In: ICML (2020)
10. Liu, Y., Zhang, W., Wang, J.: Source-free domain adaptation for semantic segmentation. In: CVPR (2021)
11. Michele, B., Boulch, A., Puy, G., Vu, T.H., Marlet, R., Courty, N.: SALUDA: Surface-based automotive lidar unsupervised domain adaptation. In: 3DV (2024)
12. Musgrave, K., Belongie, S., Lim, S.N.: Three new validators and a large-scale benchmark ranking for unsupervised domain adaptation. arXiv preprint arXiv:2208.07360 (2022)
13. Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., Snoek, J.: Evaluating prediction-time batch normalization for robustness under covariate shift. arXiv preprint arXiv:2006.10963 (2020)
14. Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., Zhao, H.: SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. In: IV (2020)
15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
16. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: ECCV (2016)
17. S, P.T., Fleuret, F.: Uncertainty reduction for model adaptation in semantic segmentation. In: CVPR (2021)
18. Saito, K., Kim, D., Teterwak, P., Sclaroff, S., Darrell, T., Saenko, K.: Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In: ICCV (2021)
19. Saltori, C., Galasso, F., Fiameni, G., Sebe, N., Ricci, E., Poiesi, F.: Cosmix: Compositional semantic mix for domain adaptation in 3d lidar segmentation. In: ECCV (2022)

20. Sanchez, J., Deschaud, J.E., Goulette, F.: Domain generalization of 3d semantic segmentation in autonomous driving. In: ICCV (2023)
21. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: ICLR (2021)
22. Xiao, A., Huang, J., Guan, D., Zhan, F., Lu, S.: Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In: AAAI (2022)
23. Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al.: Pandaset: Advanced sensor suite dataset for autonomous driving. In: ITSC (2021)
24. Ye, M., Zhang, J., Ouyang, J., Yuan, D.: Source data-free unsupervised domain adaptation for semantic segmentation. In: ACM MM (2021)
25. Yi, L., Gong, B., Funkhouser, T.: Complete & Label: A domain adaptation approach to semantic segmentation of lidar point clouds. In: CVPR (2021)
26. Yi, L., Xu, G., Xu, P., Li, J., Pu, R., Ling, C., McLeod, A.I., Wang, B.: When source-free domain adaptation meets learning with noisy labels. In: ICLR (2023)
27. Zhao, D., Wang, S., Zang, Q., Quan, D., Ye, X., Jiao, L.: Towards better stability and adaptability: Improve online self-training for model adaptation in semantic segmentation. In: CVPR (2023)
28. Zou, Y., Yu, Z., Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: ECCV (2018)