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

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

Abstract. We tackle the challenging problem of source-free unsupervised domain adaptation (SFUDA) for 3D semantic segmentation. It amounts to performing domain adaptation on an unlabeled target domain without any access to source data; the available information is a model trained to achieve good performance on the source domain. A common issue with existing SFUDA approaches is that performance degrades after some training time, which is a by-product of an under-constrained and ill-posed problem. We discuss two strategies to alleviate this issue. First, we propose a sensible way to regularize the learning problem. Second, we introduce a novel criterion based on agreement with a reference model. It is used (1) to stop the training when appropriate and (2) as validator to select hyperparameters without any knowledge on the target domain. Our contributions are easy to implement and readily amenable for all SFUDA methods, ensuring stable improvements over all baselines. We validate our findings on various 3D lidar settings, achieving state-of-the-art performance. The project repository (with code) is: github.com/valeoai/TTYD

Keywords: source-free unsupervised domain adaptation · 3D lidar point cloud · robustness

1 Introduction

The goal of domain adaptation (DA) is to transfer knowledge learned from a source domain, typically with abundant or cheap annotated data, into a model suited for a target domain, typically with less data or data more expensive to annotate, thus saving acquisition or annotation costs. Concretely, DA studies learning schemes to adapt networks to different forms of shifts between source and target data distributions. If no annotation is available for the target domain, the problem is referred to as unsupervised domain adaptation (UDA).

The traditional UDA setup requires the presence of both source and target data during training. However, this is less desirable in practical scenarios for two reasons: (i) source and target data are not always accessible at the same time due to data development cycles or to data privacy constraints, and (ii) many models

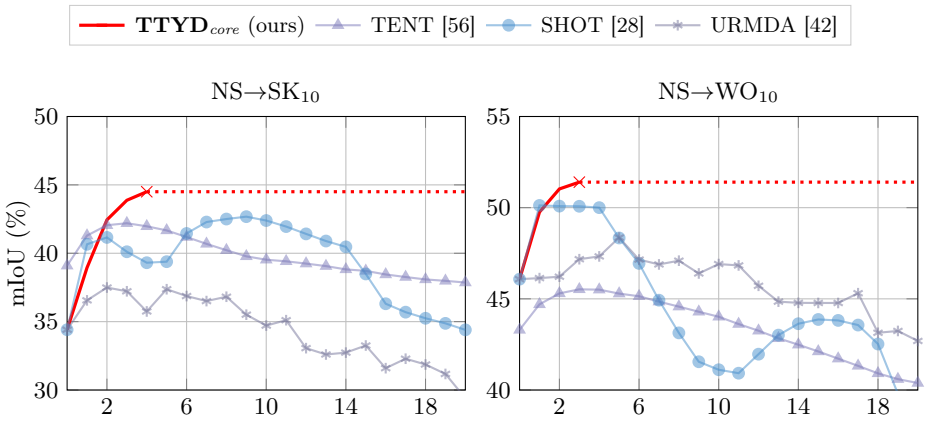

Fig. 1: Evolution of the performance of baselines without degradation prevention strategies as they train over 20k iterations. Our method (**TTYD**$_{core}$) uses an unsupervised criterion to stop training. The horizontal dotted line illustrates that we keep the model obtained at the stopping point (marked with a cross). Models are trained on nuScenes (NS) and *unsupervisedly* adapted to SemanticKITTI (SK$_{10}$) and Waymo Open (WO$_{10}$).

have already been trained on existing source data, and retraining on both source and target data is suboptimal in terms of consumed resources.

In this work, we address *source-free unsupervised domain adaptation* (SFUDA) for 3D semantic segmentation. In this setting, target adaptation is carried out using unlabeled target data and without any access to source data; a model trained on source data is however available. As opposed to vanilla UDA, SFUDA cannot rely on source supervision to prevent the training process from drifting towards collapse [21, 67]. It is illustrated in Fig. 1 for baseline methods, where training first benefits to the models before being detrimental. This phenomenon is often mitigated in papers by rules of thumb, such as qualitative assessment or early stopping based on ground-truth target labels, which are however supposed to be unavailable. Though widely used in existing work, such practices obscure quantitative comparisons and raise concerns about their actual applicability. Our method departs from these practices: it totally ignores any target ground truth.

While widely exploited on image datasets, domain adaptation has recently gained attraction regarding point clouds [66]. This task is particularly challenging because domain shifts are multiple, including specific covariate shifts due to sensors, acquisition conditions heterogeneity, and differences of class proportions between domains [66]. Techniques like self-training and mixing [45], object size adaptation [57] and surface regularization [36] have been proven effective in UDA for 3D semantic segmentation. The SFUDA setup has also been studied for 3D object detection, leveraging the temporal consistency of objects [47].

For SFUDA in 3D segmentation, we resort to a straightforward yet highly effective training scheme involving two losses: one is to encourage model certainty on target samples and the other is to regularize the divergence in class distri-

bution between source and target. To avoid the degradation issue, we propose an unsupervised criterion that indicates when to stop the training. For this criterion, the agreement of the trained model with a reference model is measured. The red curve in Fig. 1 visualizes the evolution of our model's performance during training; the red cross marks the point when training is halted using our criterion. Furthermore, we repurpose the stopping criterion as an unsupervised *validator*, in the sense of Musgrave et al. [38]. We thus can unsupervisedly tune all hyperparameters used in our base SFUDA framework, making it completely hyperparameter-free. To summarize, our contributions are the following:

- We propose an unsupervised stopping criterion targeting the degradation issue of 3D SFUDA.
- To achieve hyperparameter-freedom, we repurpose the stopping criterion as an unsupervised model validator.
- We introduce a SFUDA training scheme that works for 3D lidar data semantic segmentation and show promising results for image semantic segmentation.
- Extensive experiments (real-to-real and synthetic-to-real) show that our method outperforms the SOTA of 3D SFUDA.

2 Related work

2.1 SFUDA in Computer Vision

Traditional Unsupervised Domain Adaptation techniques rely on a variety of approaches to handle potential discrepancies between source and target domains [59]. While some approaches look for *Domain-invariant features* by minimizing statistical divergences between source and target feature representations (*e.g.*, [9, 13, 32, 34, 51, 58]), or through adversarial training (*e.g.*, [14, 33, 54]), another line of work considers finding a *Mapping between domains* [5, 17]. Based on the assumption that both domains are not too different, other strategies were proved efficient, such as reducing prediction uncertainty on target samples in *Self-supervised methods* [55, 56], relying on *Pseudo-labeling* [7, 44, 69, 72] or *Self-ensembling* [19, 26, 52, 53], that maintains a teacher model using a temporal exponential moving average of the student to ensure training stability.

In SFUDA, also called unsupervised model adaptation, and contrary to previous methods, source data is no longer available at adaptation time [29, 42]. Some abstract source information is however sometimes used, *e.g.*, adapting the target statistics of batches to those of the source [22, 27, 37, 39, 49, 56]. The seminal work SHOT [28] freezes the classification layer of the source model and finetunes the remaining parameters by leveraging an information maximization loss, composed of entropy minimization at the sample level to enforce unambiguous predictions, while promoting global diversity by constraining predicted class proportions [20, 24, 50]. Without any prior knowledge, diversity turns into an objective of producing a balanced class distribution. Also, SHOT uses pseudo-labeling based on prototypes obtained by clustering classes in the target domain.

TENT [56] freezes the model trained on source data but learns affine transformations in each normalization layer, whose parameters are trained to minimize classification entropy. Benefits lie in the reduced complexity of the linear adapters, which enforce simple changes in the normalization layer. However, as highlighted in Fig. 1, it is not sufficient to prevent the model from drifting towards collapse. To prevent this behavior, a possibility is to freeze the trainable weights of the source network and work only on batch norm statistics. AdaBN [27] replaces the running statistics (mean and variance) of the source dataset by the running statistics of the target dataset. Rather than computing the running statistics on test data once and for all, PTBN [39] solely relies on the batch statistics of test data at inference time. In a similar spirit, MixedBN in [36], which is not *per se* a SFUDA method, mixes at training time both source and target statistics of the combined source-target dataset, but requires the source data. We showcase in the remainder a small adaptation of it to the SFUDA case.

Performing adaptation at test time, those methods do not show the pathological drift exhibited in Fig. 1. However, their performances compare unfavorably to methods that train a model, *e.g.*, [28, 56]. In this work, we propose to use these non-learned models as guardrails for the optimization process.

Semantic segmentation. URMDA [42] is one of the first methods tackling semantic segmentation in SFUDA, by minimizing an uncertainty loss to make the feature representation more robust to noise, and by exploiting class-balanced pseudo-labeling [72]. Self-training, especially with pseudo-labels is also a popular approach [7, 21, 25, 31, 65, 71]. In [21], the self-training stability is enforced by constraining the current model using consistency with previous models.

3D-specific SFUDA. Applying UDA to 3D data has recently received a lot of attention, with a focus on detection [35, 41, 47, 57, 62–64, 68, 70] and segmentation [36, 45, 66]. But there are only a few works on SFUDA. Some are specifically focusing on object detection [16, 47], leveraging the trackability of cars over several frames [47], or improve the identification of regions-of-interest by using attentive class prototypes [16]. Others target online SFUDA for semantic segmentation [46], relying on spatio-temporal sequential lidar data, as well as on an additional point cloud processing network to produce geometric features.

2.2 Mitigating the drift in SFUDA

Addressing model drift during adaptation is a significant challenge in SFUDA. It is typically done by parameter tuning or early stopping based on target scores. While it offers insight into the upper-bound performance of a method, it does not account for real-world scenarios where target performance is not readily available.

Using validators. Validators have been introduced in UDA as methods for selecting hyperparameters without any access to target labels [10, 38]. In [36], target entropy, information maximization (IM), and source validation have been proven to be reliable in an UDA semantic segmentation task on 3D data. SND [43] is used in [71] as a criterion to guide the update rate of the EMA teacher.

RankME [15] assesses the quality of self-supervised representations without labeled downstream data, and can thus also be used to select models.

Learning stabilization. Another approach is to improve the training stability, *e.g.*, modulating the learning rate or the update rate of the EMA teacher for pseudo-labeling [71]. In DT-ST [71], the update interval of the EMA teacher is selected based on the evolution of the SND [43] or entropy values. In [67], the degradation is explained for pseudo-labeling approaches with the impact of noisy-labels, and an early-learning regularization term is introduced, putting more weight on the early predictions of the network in the training process.

3 Method

Our approach is mostly model-agnostic. We consider a model f, with trainable parameters θ, that takes as input a point cloud P and that outputs, for each point $p \in P$, a probabilistic classification prediction $f[\theta](P)_p \in [0,1]^K$ among K classes (generally after a softmax as final layer). Without loss of generality, we consider that P can also be a batch of point clouds, that are processed in parallel. We assume we are in the more usual white-box SFUDA setting [12]: we know the architecture and have access to the weights. We denote by $f[\theta^{\mathsf{s}}]$ the model trained on source data $\mathcal{X}^{\mathsf{s}}$. ($\mathcal{X}^{\mathsf{s}}$ is unavailable at domain adaptation time.) Finally, we assume we know the source class distribution $D^{\mathsf{s}} = D(\mathcal{X}^{\mathsf{s}}) \in [0,1]^K$. Our goal is to find, without any ground-truth knowledge of the target data $\mathcal{X}^{\mathsf{t}}$, new parameters θ^{t} such that the model $f[\theta^{\mathsf{t}}]$ performs well on $\mathcal{X}^{\mathsf{t}}$.

The framework, coined as **TTYD**, is composed of three elements that can be used independently: (i) a training scheme to regularize the adaptation of the source-only model to target data, (ii) a stopping criterion ($\mathbf{TTYD}_{stop}$) to halt training and prevent performance degradation, which is additionally repurposed as a *validator* ($\mathbf{TTYD}_{valid}$) to unsupervisedly tune training hyperparameters, and (iii) a self-training module using the initially-adapted model (i)+(ii) ($\mathbf{TTYD}_{core}$) as a starting point.

3.1 Training scheme

Rather than training a new model from scratch, we assume that the target domain is not widely different from the source domain and adapt the already-trained model $f[\theta^{\mathsf{s}}]$ by fine-tuning it on target data $\mathcal{X}^{\mathsf{t}}$, without any label supervision.

General idea. To train on unlabeled target data, we need a guidance that does not require ground-truth knowledge. To that end, we consider two training objectives. First, and quite classically, the trained (adapted) model should be discriminative, i.e., points should be classified with a large margin, which is one way to promote certainty in the predictions. Second, and more originally, the predicted class distribution of the target data should not only be diverse but in fact similar enough to the source class distribution.

As already noted, previous SFUDA work only considers perfect class balancing [28], while autonomous driving data contains severe class imbalance, with

factors of proportion up to three orders of magnitude [30]. Besides, blindly balancing the classes ignores information that is readily available in the distribution of the source data. Additionally, favoring the alignment of the predicated class distribution onto source data is consistent with the fine-tuning strategy, which consists in finding θ^{t} in the neighborhood of θ^{s}. Conversely, if target data is actually very different from source data, domain adaptation makes little sense in the first place. While the first objective (discriminability) is neither particular to the task nor to the target domain, the second one (distribution similarity with source data) is specific both to the task and to the target data.

Formal description. Concretely, to perform the training on target data, we use a loss that does not require ground-truth knowledge. This new loss is composed of two terms, which correspond to the two objectives mentioned above.

The first term penalizes ambiguity in the probabilistic class predictions. To that end, we classically [28, 56] measure the entropy of predictions:

$$\mathcal{L}_{\mathsf{discrim}}(P) = \frac{1}{|P|} \sum_{p \in P} H(f[\theta^{\mathsf{t}}](P)_p) \tag{1}$$

where $|P|$ is the number of points in P, and H is the entropy function.

The second term penalizes the discrepancy between the known class distribution in the source data D^{s}, which we assume is not widely different from the (unknown) class distribution in the target data D^{t}, and the predicted class distribution of D^{t}, estimated as the average on the current point cloud (or batch) P:

$$\mathcal{L}_{\mathsf{simsrc}}(P) = \mathrm{KL}(D(P)||D^{\mathsf{s}})\,, \quad \text{where } D(P) = \frac{1}{|P|} \sum_{p \in P} f[\theta^{\mathsf{t}}](P)_p \tag{2}$$

and KL is the Kullback–Leibler divergence. Of note, our approach differs from prior work [28], which tries to enforce similarity with the uniform class distribution. In urban scene segmentation, while the source's class distribution is not perfectly aligned with the target's, it still serves as a more accurate prior than uniform. While an explicit class distribution prior has already been used in UDA [4, 18], we develop it here in the specific context of SFUDA: whereas source data is inaccessible, we assume the source class distribution remains available.

Our final loss is the sum of these two terms. We do not introduce any balancing factor as the two losses somehow have a similar nature and range of values. Indeed, like $\mathcal{L}_{\mathsf{discrim}}$, $\mathcal{L}_{\mathsf{simsrc}}$ can also be expressed with (cross-)entropies:

$$\mathcal{L}_{\mathsf{simsrc}}(P) = \mathrm{KL}(D(P)||D^{\mathsf{s}}) = H(D(P), D^{\mathsf{s}}) - H(D(P))\,. \tag{3}$$

Yet, to prevent overconfidence in discriminability, we consider a hinge-loss-like variant of $\mathcal{L}_{\mathsf{discrim}}$ that ignores samples with very low entropy. Similarly, to prevent the adapted model from following exactly the estimated distribution of classes in the target set, we clip $\mathcal{L}_{\mathsf{simsrc}}$ under a certain threshold. Our actual loss is:

$$\mathcal{L}(P) = \max(0, \mathcal{L}_{\mathsf{discrim}}(P) - \lambda) + \max(0, \mathcal{L}_{\mathsf{simsrc}}(P) - \lambda)\,. \tag{4}$$

We use the same margin λ for both losses, which is set to 0.02 in all experiments.

The 3D source model is trained from scratch. Once trained, the Batch Normalization (BN) layers [22] within the 3D model profoundly embody the characteristics of the source domain. It results in significant covariate shifts when the model is applied to the target domain. Competitive results in 3D UDA are reported [36] by simply altering BN statistics, with AdaBN [27], PTBN [39] or MixedBN [36]. Despite its low operational cost, the effectiveness of BN adaptation in 3D perception is intriguing. Here, we explore this idea for 3D SFUDA.

We conducted an extensive study to determine which parameters (the entire network, the classifier, or the BN layers) are better to finetune. Our finding is that most parameter schemes yield similar results. (See supp. mat. for details.)

As it is sufficient to only alter few parameters, we keep the model $f[\theta^{\mathsf{s}}]$ completely frozen and replace BN layers by optimizable linear layers initialized with BN statistics, scale and bias. A similar affine transformation is also used, but at inference time, in [56].

3.2 Unsupervised stopping criterion and model validator

As discussed above, training in current SFUDA methods starts to provide gain over the source-only model, before degrading (Fig. 1). Workarounds include adding hyper-parameters that are hard to set without peeking at the inaccessible target ground truth, *e.g.*, fixed number of iterations or learning-rate scheduling.

A rightful solution is to rely on a *validator*, which scores adapted models to choose the best one [38]. Using such a validator to tune hyperparameters (including the number of training iterations), is a way to make domain adaptation methods truly unsupervised [36, 71]. A constraint is to use a validator that is not based on the same principle as the validated domain adaptation method. As an example, using the minimization of entropy both as a validator and as an objective model optimization would lead to an infinite training. As validators tend to measure the same kind of aspects that DA methods try to optimize, i.e., class discriminability and class diversity, this situation is not uncommon.

In fact, as illustrated in the experiment Sec. 4.2, existing validators are not well suited for our method and fail to select a good model. The reason is that, as a gauge of discriminability, a number of validators are also based or inspired by a measure of entropy, as is our method. Regarding diversity, as existing validators are designed to be general and target-set agnostic, they tend to measure how uniform the class distribution is, which is basically the only thing one can do without any prior on the target set. But as explained above, it is not appropriate for autonomous driving data, which features highly-imbalanced classes. A specific validator-like criterion is needed with our SFUDA training.

Objective. Our goal here is to try to capture the best performance achieved by a model as it trains, without any label knowledge on target data. More precisely, we aim to identify the point when the unknown, underlying performance of a model being trained starts to degrade.

General idea. In a supervised setting, a validation dataset is used to stop training when the performance on this data starts to drop, thus reducing the risk

of over-fitting up to a certain extent. In UDA, the source data can be used either during the training for stabilization purposes, or as a validator [38] to find an optimal hyperparameter setting or an optimal point to stop the training. But in SFUDA, source data is not available; we can thus only use a model trained using source data as a basis to construct a validator or a stopping criterion.

For this construction, rather than just using model $f[\theta^{\mathsf{s}}]$, we propose to use an additional auxiliary model g that is already adapted to the target data in an SFUDA fashion, and which is thus better than $f[\theta^{\mathsf{s}}]$. The idea however remains to explore the space of domain adaptations using our SFUDA training, starting from $f[\theta^{\mathsf{t}}_0] = f[\theta^{\mathsf{s}}]$. The auxiliary model g only acts as a kind of anchor to detect when the model being trained strays too much and degrades. It does not alter the training of $f[\theta^{\mathsf{t}}]$ in any way, and definitely does not act as an upper bound in terms of performance. It merely helps to identify when training $f[\theta^{\mathsf{t}}]$ should stop.

Formal description. Given two models f, g that classify (among K classes) the points x of a dataset $\mathcal{X}$, we measure their *class assignment agreement* $A(f, g)$ by counting the number of times they make identical predictions:

$$A(f,g) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \mathbb{1}\left(\operatorname*{argmax}_{k \in [\![1,K]\!]} f(x)_k = \operatorname*{argmax}_{k \in [\![1,K]\!]} g(x)_k\right). \tag{5}$$

As alternatives to this hard counting, we experimented with various divergences to measure the agreement (symmetric KL divergence, L1 and L2 norms). All options gave similar results (see supp. mat.) and we kept the simplest one.

The measure $A(f, g)$, which is also a metric, can be used to define a model validator in the sense of Musgrave et al. [38], i.e., as a way to select the best model among a set of choices. Given a reference model g and a set of models F, the best model f^* is the one that agrees the most with g:

$$f^* = \operatorname*{argmax}_{f \in F} A(f,g). \tag{6}$$

We now consider a model f being trained, with parameters θ_i at iteration i. In its training trajectory from θ_0, the *closest agreement point* of f with another model g is the smallest iteration i^* that maximizes the agreement, i.e., $i^* = \min \operatorname{argmax}_i A(f[\theta_i], g)$. Given that most on-going trainings tend to improve the performance, before the performance starts to drop continuously (cf. Fig. 1), we consider as stopping point the first reversal in the increasing agreement phase, i.e., the first iteration $\hat{\imath}$ after which the agreement starts to decrease:

$$\hat{\imath} = \operatorname*{argmin}_i A(f[\theta_i], g) \geq A(f[\theta_{i+1}], g). \tag{7}$$

The advantage of this *first disagreement trend* is that it does not have any parameter and is quick to compute, whereas the closest agreement requires a maximum training horizon. In theory, the stopping could be sub-optimal if there are local maxima in the evolution of the class assignment agreement. However, in practice, we do not check the agreement after processing each batch but after a significant number of iterations (typically 1000), which has a smoothing effect.

In our experiments, even checking the agreement as often as every 100 iterations, which is practically useless on our context, yields similar results.

Empirically, the training of model f stops when reaching the maximum agreement of 60-80% with g, but at a performance much higher than g by a large margin. Though using an auxiliary model g as anchor can be seen as a limitation in that it does not favor a disruptive improvement of f, we argue, as shown in our experiments, that the remaining slack of 20-40% is sufficient to provide substantial benefits, while preventing catastrophic outcomes in SFUDA.

Note that taking $g = f[\theta^{\mathsf{s}}]$ would lead to a degenerate case because the closest agreement point for $A(f[\theta_i^{\mathsf{t}}], g)$ is then reached with $i^* = 0$, i.e., $\theta_0^{\mathsf{t}} = \theta^{\mathsf{s}}$, meaning that there is no adaptation on target data from the model trained on source data. Therefore, we have to take as training starting point $f[\theta_0^{\mathsf{t}}]$ a model close to $f[\theta^{\mathsf{s}}]$, but not equal to it. We have several possible choices, among the pure SFUDA methods, as discussed below.

Selection of a reference model g. In theory, the reference model g can be any model at hand and reliable, provided it is not based on the same principles as the training scheme. However, as a practical guideline, low-cost, hyperparameter-free and training-free reference models are more favorable for SFUDA.

Recent 3D UDA SOTA [36] reveals the intriguing effectiveness of low-cost BN adaptation methods. We revisit these methods in the SFUDA context and observe competitive performance. Interestingly, BN adaptation methods do not require training, hence they do not suffer from the degradation issue of training-based methods. In addition, methods like AdaBN or PTBN are hyperparameter-free, which is ideal for the unsupervised setting of SFUDA. BN-adapted models hence become our primary choices to select a reference model g.

In the following, we use PTBN as our default reference model. It gives similar results as AdaBN but can be evaluated on the fly, thus requiring less computation. And we denote by $\mathbf{TTYD}_{stop}$ the corresponding stopping criterion.

Model validator. The stopping criterion $\mathbf{TTYD}_{stop}$, can serve as a model validator, referred to as $\mathbf{TTYD}_{valid}$, whose score is simply defined as the agreement level at the stopping point, i.e., $A(f[\theta_i], g)$. The validator helps unsupervisedly tune the hyperparmeters to obtain the best model f^*, thanks to Eq. (6).

3.3 Self-training module

The proposed training scheme along with the criterion $\mathbf{TTYD}_{stop}$ and the validator $\mathbf{TTYD}_{valid}$ allow us to adapt the pretrained source model to a target domain using only target data; more importantly the entire process is hyperparameter-free. As later demonstrated in Sec. 4.4, this adapted model alone, referred to as $\mathbf{TTYD}_{core}$, performs better or is on par with SOTA baselines.

To obtain the final **TTYD** model, we conduct the second phase of self-training [7, 21, 25, 31, 65, 71]. Specifically, starting from $\mathbf{TTYD}_{core}$, pseudo-labels are computed on the fly for unlabeled target data, and then are used to self-train the network with the standard cross-entropy loss. To stabilize training, the EMA teacher model is used for pseudo-labeling [1]. Additionally, we employ the

Table 1: Datasets used in our domain adaptation experiments.

Dataset			Lidar beams	cls.	Region of the world	Adaptation pairs
nuScenes	[3]	(NS)	HDL-32E / 32	16	Boston, Singapore	
SynLiDAR	[60]	(SL)	*synthetic* / 64	22	Unreal Engine 4	
PandaSet	[61]	(PD)	Pandar64 / 64	37	2 US cities	NS→PD_8 [48]
Waymo Open	[11]	(WO)	L.B.H. / 64	23	3 US cities	NS→WO_{10} [23]
SemanticPOSS	[40]	(SP)	Pandora / 40	14	Peking University	NS→SP_6 [48], SL→SP_{13} [45]
SemanticKITTI	[2]	(SK)	HDL-64E / 64	19	Karlsruhe	NS→SK_{10} [66], SL→SK_{19} [45]

In adaptation pairs, subscript number on target indicate the number of mapped classes (cls.).

Dynamic Teacher Update (DTU) [71] to adjust the update rate of the teacher model dynamically, further stabilizing SFUDA self-training.

4 Experiments

4.1 Experimental setup

Datasets. The datasets we use for evaluation are listed in Tab. 1. It is worth noting the variety of (rotating) lidar sensors (in particular number of beams), labeled classes, and world scenes. Besides, one of the six datasets is synthetic.

Class mapping. The SFUDA setting assumes that source and target domains share semantic classes. In practice, when comparing existing datasets with ground-truth data, not all classes are shared and there are sometimes partial class overlaps. For each source-target pair, we therefore have to select and aggregate common classes in the two datasets to evaluate the quality of the domain adaptation. However, we do not train source-only models based on class mappings; we use the official classes of each dataset. The class mapping (Tab. 1 and supp. mat.) is only used at evaluation time, to map source-domain classes inferred on target data onto common classes that can be compared based on target ground truth.

Adapted domains. The different domain adaptation settings we experiment with are summarized in Tab. 1. In the following, we write as subscript the number of aggregated common classes that we use to evaluate the quality of the adaptation. We address different types of domain shifts: real-to-real and sparse-to-dense (NS→SK_{10}, NS→SP_6, NS→PD_8, NS→WO_{10}), as well as synthetic-to-real (SL→SK_{19}) including dense-to-sparse (SL→SP_{13}).

Network setting. For all evaluated methods, we use the same sparse-voxel Minkowski U-Net [6] with 10 cm voxel size. It is a commonly used model for automotive lidar semantic segmentation. The model contains 49 batch normalization layers, thus, adapted parameters represent 0.06% of the model parameters.

As in [45, 66], we do not use lidar intensity as input feature. Lidar intensities are difficult to synthesize in simulated datasets and, for real datasets, reflectance calibration may vary a lot from one sensor to another.

To train our method, we use AdamW with a learning rate of 10^{-5}, a weight decay of 0.01, and a batch size of 4. We use $\lambda = 0.02$ in all settings and train for

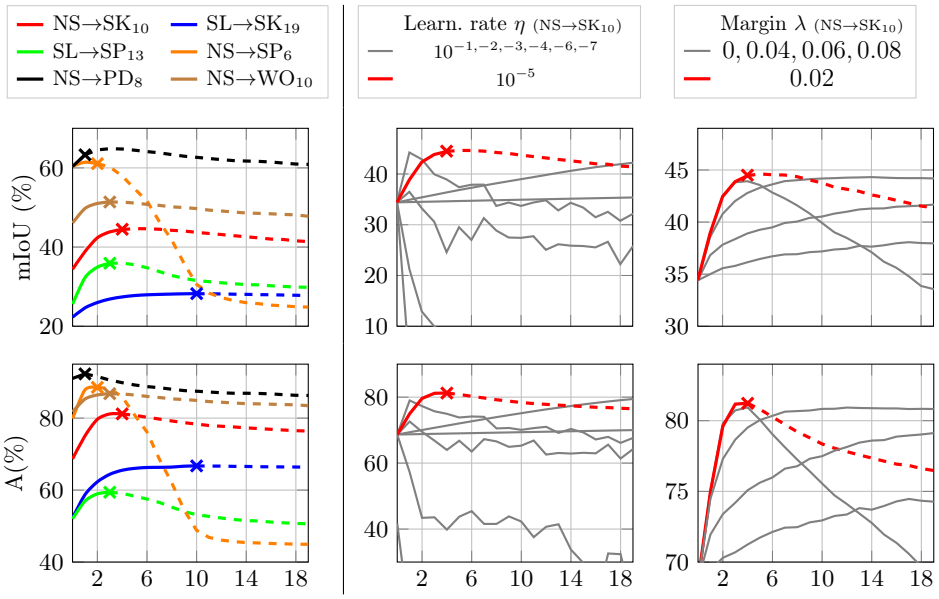

Fig. 2: Performance %mIoU (top), as reference, and class agreement in % (bottom), for training over 20k iterations. **(1st column)** the crosses indicate when $\mathbf{TTYD}_{stop}$ stops the training in different SFUDA setups. Dashed lines after the crosses just illustrate the expected degradation issue. In reality, we do not continue training once the criterion is triggered. **(2nd and 3rd columns)** the red curves correspond to the hyperparameters η and λ selected using $\mathbf{TTYD}_{valid}$ in NS→SK$_{10}$, showing we pick the best ones.

at most 20k iterations on target data, creating checkpoints every 1k iterations to test our stopping criterion. The source-only models are trained to achieve high performance on the source validation set, regardless of the target data and without considering class mapping.

We show in our ablation study and the application to image modality (both see supp. mat.) that a wide range of models can serve as reference. However BN adaptation models are the most readily available and remain competitive in performance.

Evaluation. We measure performance with the classwise intersection over union (IoU) and the mean IoU (mIoU) over all classes, as done in the official SK benchmark [2], i.e., computed over the whole evaluation dataset.

4.2 Stopping criterion $\mathbf{TTYD}_{stop}$

In this section, we evaluate the quality of our stopping criterion $\mathbf{TTYD}_{stop}$.

First, Fig. 2(left) shows that our training scheme, while being relatively stable (little performance gap between the last and maximal mIoUs) on the majority of adaptation scenarios, can still suffer from a sharp drop of performance: -38.0 pp. on NS→SP$_6$. This highlights the need for using a good stopping criterion.

Table 2: Unsupervised stopping criteria to select the best checkpoint in 20k training iterations (one checkpoint every 1k iterations). *Oracle w/ GT* gives the upper bound.

Stop. Criterion	NS→SK$_{10}$	SL→SK$_{19}$	SL→SP$_{13}$	NS→SP$_6$	NS→WO$_{10}$	NS→PD$_8$
Entropy [38]	41.4	27.8	29.7	23.7	47.8	60.9
SND [43]	41.4	22.3	30.5	23.7	47.8	60.9
IM [38]	42.4	27.8	<u>34.8</u>	57.5	<u>51.1</u>	63.0
BNM [8]	<u>43.9</u>	27.8	32.1	<u>59.9</u>	<u>51.1</u>	63.0
RankME [15]	42.4	**28.2**	32.1	57.5	51.0	**63.3**
TTYD$_{stop}$	**44.5**	**28.2**	**35.9**	**61.1**	**51.4**	**63.3**
Oracle w/ GT	44.7	28.2	36.0	61.4	51.4	64.9

Second, on the six practical domain adaptation cases we study, our stop criterion **TTYD**$_{stop}$ is able to identify a model reaching a performance close to the best achievable one. This highlights the effectiveness of our method. In none of the observed runs was **TTYD**$_{stop}$ misled by a local maxima of A. Computation is thus saved without giving up performance.

Third, we benchmark **TTYD**$_{stop}$ in Tab. 2. It outperforms other validators used as stopping criteria by a significant margin. RankMe, which is designed to score feature quality, always chooses a model close to the source-only model. As expected, the 'Entropy' validator selects suboptimal models as it relies on one of the ingredients that we also use for our actual domain adaptation (cf. Eq. (1).

In conclusion, we see that our stopping criterion **TTYD**$_{stop}$ systematically selects high-performing models. We however do not claim it is applicable beyond SFUDA, but that it is well suited for that problem.

4.3 Model validator TTYD$_{valid}$

Fig. 2(right) shows performance curves on NS→SK$_{10}$, for a range of learning rates η and margins λ, aligned with their agreement score A. We observe that the agreement, which we can easily compute, is a good proxy for the actual mIoU, which cannot be known for selecting the highest one as the ground truth is not accessible. The weighted Spearman correlation (as in [38]) between performance and agreement is 0.95 for the learning rates and 0.75 for the margins. Selecting the highest agreement thus is close to selecting the highest mIoU. In fact, **TTYD**$_{valid}$ selects $\eta = 10^{-5}$ and $\lambda = 0.02$.

4.4 3D-SFUDA benchmark

Strict SFUDA setting (hyperparameter free). We consider here a strict SFUDA setting: any hyperparameter, if it exists, must be tuned without any access to target scores, thus, *e.g.*, using to a SFUDA validator.

We compare **TTYD**$_{core}$ to methods that do not have any hyperparameter and that can thus be used in a pure SFUDA setting: *Source-only*, which is the model

Table 3: Performance (mIoU%) on target validation sets in two SFUDA settings: strict (without hyperparameters, or with hyperparameters tuned with a validator) and vanilla (with hyperparameters set using target ground truth). For additional comparison, we provide UDA results (using source data at adaptation time).

	Domains / Method	Src. free	H.P. free	NS→ SK_{10}	SL→ SK_{19}	SL→ SP_{13}	NS→ SP_6	NS→ WO_{10}	NS→ PD_8
strict SFUDA	Source-only	✓	✓	34.4	22.3	25.6	60.4	46.1	60.4
	AdaBN [27]	✓	✓	39.9	24.6	25.4	57.7	47.7	59.6
	PTBN [39]	✓	✓	39.4	22.4	23.7	54.7	42.3	60.2
	MeanBN [36]	✓	✓	41.7	26.9	27.7	60.9	50.3	61.3
	TTYD$_{core}$	✓	✓	**44.5**	**28.2**	**35.9**	**61.1**	**51.4**	**63.3**
(loose) SFUDA	SHOT [28]	✓	✗	34.9	18.4	21.7	42.4	37.3	43.7
	TENT [56]	✓	✗	37.9	24.5	28.3	45.1	40.4	59.1
	URMDA [42]	✓	✗	29.4	25.4	24.5	30.8	42.7	56.9
	SHOT + ELR [67]	✓	✗	40.5	27.1	36.9	59.4	49.5	60.9
	DT-ST [71]	✓	✚	35.6	23.5	36.8	63.1	51.8	62.5
	TTYD	✓	✚	**45.4**	**32.4**	**39.1**	**64.5**	**55.5**	**65.7**
UDA	CoSMix [45]	✗	✗	38.3	28.0	40.8	65.2	–	–
	SALUDA [36]	✗	✗	46.2	31.2	42.9	65.8	–	–

H.P. free (no hyperparameter or selected with validator): ✓ = Yes; ✗ = No and parameter sets specific to each setting either reported in literature [36, 45] or re-run by ourselves when default parameter do not perform correctly [28, 42, 56]; ✚ = No but using one single set of parameters for all settings taken from image SFUDA literature [67, 71]). Src. free: not using any source data at adaptation time.

$f[\theta^s]$ trained on source data without any adaptation, *AdaBN* [27], *PTBN* [39] and *MeanBN*, which is a simple source-free adaptation of MixedBN [36] (see supp. mat. for a detailed description).

The strict SFUDA setting is presented in the upper part ofTab. 3. **TTYD**$_{core}$ systematically outperforms all other parameterless approaches, sometimes with a large margin (up to +8.2 pp. on SL→SP_{13}).

Loose SFUDA setting. In this setting, we allow the use of hyperparameters tuned by looking at the target performances. These hyperparameters may be specific to each adaptation pair (indicated by ✗ in Tab. 3) or tuned once and for all (indicated by ✚ in Tab. 3), possibly on other modalities, *e.g.*, images.

As the default hyperparameters of SHOT [28], TENT [56], and URMA [42] do not transfer to 3D SFUDA, we retrained these approaches with various sets of hyperparameters and selected the best performing ones for each adaptation pair.

Regarding SHOT + ELR [67], we used a grid-searched hyperparameter for SHOT and the two default hyperparameters for ELR, which are described as robust [67]. As DT-ST [71] is designed for stability and robustness in the SFUDA setting, we used its default set of hyperparameters (experimented on images), which we also use for the DTU self-training module of **TTYD**. Last, we report UDA scores (use of source data at adaptation time) for CosMix [45] and SALUDA [36], as expected upper-bounds exploiting extra information.

The results obtained in the common "vanilla" SFUDA setting are presented in the middle part of Tab. 3. First, we observe that **TTYD** reaches state-of-the-art performance on all adaptation scenarios. Second, comparing the results of **TTYD**$_{core}$ and **TTYD** highlights the interest of using a self-training scheme for SFUDA. Third, if not for **TTYD**, **TTYD**$_{core}$ ranks first or second in the vanilla benchmark, which shows that hyperparameter-less or hyperparameter-validated approaches are competitive. Last, **TTYD** closes the gap between SFUDA methods and UDA approaches with an average gap of 1.2 mIoU point on four adaptation pairs.

4.5 Application to image modality

The formulation of **TTYD** appears to be general enough to be used for other modalities than 3D lidar data. To study this aspect, we conducted experiments on image segmentation and obtained promising results. Please refer to the supp. material for more details.

4.6 Ablations

Loss terms. Ablation of the two loss terms are presented in Tab. 4, showing the relevance of each ingredient.

Prior class distribution. In Tab. 4, we compare the performance obtained with a uniform prior, to the one obtained using the source class statistics. It clearly shows the advantage of taking into account the strong class imbalances in the data, even though they are approximated by the source statistics.

Table 4: Loss and distribution study (NS→SK$_{10}$).

$\mathcal{L}_{\text{discrim}}$	$\mathcal{L}_{\text{simsrc}}$ unif.	$\mathcal{L}_{\text{simsrc}}$ src	max mIoU%
			34.4
✓			34.4
	✓		34.4
		✓	40.9
✓		✓	**44.7**

5 Conclusion

In this work, we propose simple and effective strategies to stabilize the performance of Source-Free Unsupervised Domain Adaptation in 3D semantic segmentation. Our contributions include a novel stopping criterion that measures an agreement with a reference model, and prevents catastrophic drifting of performance due to the under-constrained nature of the optimization problem. We also provide an easy to apply, yet efficient training scheme, that is well suited for the task of semantic segmentation in autonomous driving scenarios. We demonstrate the effectiveness of our proposal through extensive comparisons with state-of-the-art methods in 3D semantic segmentation, which is a challenging SFUDA instance, and we show its applicability in the image domain.

Acknowledgements

We also acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grants ANR-21-CE23-0032 (project MultiTrans), ANR-20-CHIA-0030 (OTTOPIA AI chair), and the European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617. This work was performed using HPC resources from GENCI–IDRIS (2022-AD011013839, 2023-AD011013839R1).

References

1. Araslanov, N., Roth, S.: Self-supervised augmentation consistency for adapting semantic segmentation. In: CVPR (2021)
2. 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)
3. 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)
4. Chen, Y.H., Chen, W.Y., Chen, Y.T., Tsai, B.C., Frank Wang, Y.C., Sun, M.: No more discrimination: Cross city adaptation of road scene segmenters. In: ICCV (2017)
5. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: CVPR (2018)
6. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR (2019)
7. Corbiere, C., Thome, N., Saporta, A., Vu, T.H., Cord, M., Perez, P.: Confidence estimation via auxiliary models. IEEE TPAMI (2021)
8. Cui, S., Wang, S., Zhuo, J., Li, L., Huang, Q., Tian, Q.: Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In: CVPR (2020)
9. Damodaran, B.B., Kellenberger, B., Flamary, R., Tuia, D., Courty, N.: Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In: ECCV (2018)
10. Ericsson, L., Li, D., Hospedales, T.M.: Better practices for domain adaptation. In: AutoML (2023)
11. 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)
12. Fang, Y., Yap, P.T., Lin, W., Zhu, H., Liu, M.: Source-free unsupervised domain adaptation: A survey. arXiv preprint arXiv:2301.00265 (2022)
13. Fatras, K., Séjourné, T., Courty, N., Flamary, R.: Unbalanced minibatch optimal transport; applications to domain adaptation. In: ICML (2021)
14. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. JMLR (2016)

15. Garrido, Q., Balestriero, R., Najman, L., Lecun, Y.: Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In: ICML (2023)
16. Hegde, D., Patel, V.M.: Attentive prototypes for source-free unsupervised domain adaptive 3d object detection. In: WACV (2024)
17. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. In: ICLR (2018)
18. Hoffman, J., Wang, D., Yu, F., Darrell, T.: FCNs in the wild: Pixel-level adversarial and constraint-based adaptation. arXiv preprint arXiv:1612.02649 (2016)
19. Hoyer, L., Dai, D., Van Gool, L.: Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In: CVPR (2022)
20. Hu, W., Miyato, T., Tokui, S., Matsumoto, E., Sugiyama, M.: Learning discrete representations via information maximizing self-augmented training. In: ICML (2017)
21. Huang, J., Guan, D., Xiao, A., Lu, S.: Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. In: NeurIPS (2021)
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. PMLR (2015)
23. Kim, H., Kang, Y., Oh, C., Yoon, K.J.: Single domain generalization for lidar semantic segmentation. In: CVPR (2023)
24. Krause, A., Perona, P., Gomes, R.: Discriminative clustering by regularized information maximization. In: NeurIPS (2010)
25. Kundu, J.N., Kulkarni, A., Singh, A., Jampani, V., Babu, R.V.: Generalize then adapt: Source-free domain adaptive semantic segmentation. In: ICCV (2021)
26. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: ICLR (2017)
27. Li, Y., Wang, N., Shi, J., Hou, X., Liu, J.: Adaptive batch normalization for practical domain adaptation. PR **80** (2018)
28. 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)
29. 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)
30. Liu, M., Zhou, Y., Qi, C.R., Gong, B., Su, H., Anguelov, D.: LESS: Label-efficient semantic segmentation for lidar point clouds. In: ECCV (2022)
31. Liu, Y., Zhang, W., Wang, J.: Source-free domain adaptation for semantic segmentation. In: CVPR (2021)
32. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: ICML (2015)
33. Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. In: NeurIPS (2018)
34. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: ICML (2017)
35. Luo, Z., Cai, Z., Zhou, C., Zhang, G., Zhao, H., Yi, S., Lu, S., Li, H., Zhang, S., Liu, Z.: Unsupervised domain adaptive 3d detection with multi-level consistency. In: ICCV (2021)
36. Michele, B., Boulch, A., Puy, G., Vu, T.H., Marlet, R., Courty, N.: SALUDA: Surface-based automotive lidar unsupervised domain adaptation. In: 3DV (2024)
37. Mirza, M.J., Micorek, J., Possegger, H., Bischof, H.: The norm must go on: Dynamic unsupervised domain adaptation by normalization. In: CVPR (2022)

38. 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)
39. 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)
40. 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)
41. Peng, X., Zhu, X., Ma, Y.: Cl3d: Unsupervised domain adaptation for cross-lidar 3d detection. In: AAAI (2023)
42. S, P.T., Fleuret, F.: Uncertainty reduction for model adaptation in semantic segmentation. In: CVPR (2021)
43. 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)
44. Saito, K., Ushiku, Y., Harada, T.: Asymmetric tri-training for unsupervised domain adaptation. In: ICML (2017)
45. 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)
46. Saltori, C., Krivosheev, E., Lathuiliére, S., Sebe, N., Galasso, F., Fiameni, G., Ricci, E., Poiesi, F.: Gipso: Geometrically informed propagation for online adaptation in 3d lidar segmentation. In: ECCV (2022)
47. Saltori, C., Lathuiliére, S., Sebe, N., Ricci, E., Galasso, F.: Sf-uda 3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection. In: 3DV (2020)
48. Sanchez, J., Deschaud, J.E., Goulette, F.: Domain generalization of 3d semantic segmentation in autonomous driving. In: ICCV (2023)
49. Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. In: NeurIPS (2020)
50. Shi, Y., Sha, F.: Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In: ICML (2012)
51. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: ECCV (2016)
52. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: NeurIPS (2017)
53. Tranheden, W., Olsson, V., Pinto, J., Svensson, L.: Dacs: Domain adaptation via cross-domain mixed sampling. In: WACV (2021)
54. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: CVPR (2017)
55. Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: CVPR (2019)
56. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: ICLR (2021)
57. Wang, Y., Chen, X., You, Y., Li, L.E., Hariharan, B., Campbell, M., Weinberger, K.Q., Chao, W.L.: Train in germany, test in the usa: Making 3d object detectors generalize. In: CVPR (2020)
58. Wang, Y., Li, W., Dai, D., Van Gool, L.: Deep domain adaptation by geodesic distance minimization. In: CVPRW (2017)

59. Wilson, G., Cook, D.J.: A survey of unsupervised deep domain adaptation. ACM TIST (2020)
60. 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)
61. 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)
62. Xu, Q., Zhou, Y., Wang, W., Qi, C.R., Anguelov, D.: Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In: ICCV (2021)
63. Yang, J., Shi, S., Wang, Z., Li, H., Qi, X.: St3d: Self-training for unsupervised domain adaptation on 3d object detection. In: CVPR (2021)
64. Yang, J., Shi, S., Wang, Z., Li, H., Qi, X.: St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. IEEE TPAMI (2022)
65. Ye, M., Zhang, J., Ouyang, J., Yuan, D.: Source data-free unsupervised domain adaptation for semantic segmentation. In: ACM MM (2021)
66. Yi, L., Gong, B., Funkhouser, T.: Complete & Label: A domain adaptation approach to semantic segmentation of lidar point clouds. In: CVPR (2021)
67. 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)
68. You, Y., Diaz-Ruiz, C.A., Wang, Y., Chao, W.L., Hariharan, B., Campbell, M., Weinbergert, K.Q.: Exploiting playbacks in unsupervised domain adaptation for 3d object detection in self-driving cars. In: ICRA (2022)
69. 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 (2021)
70. Zhang, W., Li, W., Xu, D.: Srdan: Scale-aware and range-aware domain adaptation network for cross-dataset 3d object detection. In: CVPR (2021)
71. 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)
72. Zou, Y., Yu, Z., Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: ECCV (2018)