

Compositional Semantic Mix for Domain Adaptation in Point Cloud Segmentation

Cristiano Saltori, Fabio Galasso, Giuseppe Fiameni, Nicu Sebe, Fabio Poiesi, and Elisa Ricci

Abstract—Deep-learning models for 3D point cloud semantic segmentation exhibit limited generalization capabilities when trained and tested on data captured with different sensors or in varying environments due to domain shift. Domain adaptation methods can be employed to mitigate this domain shift, for instance, by simulating sensor noise, developing domain-agnostic generators, or training point cloud completion networks. Often, these methods are tailored for range view maps or necessitate multi-modal input. In contrast, domain adaptation in the image domain can be executed through sample mixing, which emphasizes input data manipulation rather than employing distinct adaptation modules. In this study, we introduce compositional semantic mixing for point cloud domain adaptation, representing the first unsupervised domain adaptation technique for point cloud segmentation based on semantic and geometric sample mixing. We present a two-branch symmetric network architecture capable of concurrently processing point clouds from a source domain (e.g. synthetic) and point clouds from a target domain (e.g. real-world). Each branch operates within one domain by integrating selected data fragments from the other domain and utilizing semantic information derived from source labels and target (pseudo) labels. Additionally, our method can leverage a limited number of human point-level annotations (semi-supervised) to further enhance performance. We assess our approach in both synthetic-to-real and real-to-real scenarios using LiDAR datasets and demonstrate that it significantly outperforms state-of-the-art methods in both unsupervised and semi-supervised settings.

Index Terms—Domain adaptation, unsupervised learning, semi-supervised learning, semantic segmentation, point cloud.

1 INTRODUCTION

LIDAR is currently the most suitable sensor for capturing accurate 3D measurements of an environment for autonomous driving [1] and robotic navigation [2]. Semantic scene understanding is a crucial component for AI-based perception systems [3]. LiDAR measurements can be analyzed in the form of 3D point clouds, with point cloud semantic segmentation used to assign a finite set of semantic labels to the 3D points [4]. To train accurate deep learning models, large-scale datasets with point-level annotations are necessary [5], [6], [7]. This involves a costly and labor-intensive data collection process, as point clouds need to be captured in the real world and manually annotated. An alternative is to use synthetic data, which can be conveniently generated with simulators [8]. However, deep neural networks are known to suffer from domain shift when trained and tested on data from different domains [8]. Although simulators can reproduce the acquisition sensor with high fidelity, further research is still required to address such domain shift [9].

Data augmentation techniques based on the combination of samples and their labels, such as Mixup [10] or Cut-Mix [11], have been proposed to enhance deep network generalization. The underlying concept involves mixing

samples to expand the training set and reduce overfitting. These methods were initially applied to image classification tasks and later adapted for domain adaptation and domain generalization in image recognition [12], [13]. Similar ideas have also been successfully extended to 2D semantic segmentation [14], [15]. While Unsupervised Domain Adaptation (UDA) for semantic segmentation in the image domain has been extensively studied [14], [15], [16], [17], [18], less attention has been devoted to developing adaptation techniques for point cloud segmentation. Point cloud UDA can be addressed in the input space [9], [19] with dropout rendering [19] or adversarial networks [9], or in the feature space through feature alignment [20]. A few studies have proposed exploiting sample mixing for point cloud data [21], [22], but they target different applications than UDA for semantic segmentation.

In this paper, we present a novel domain adaptation framework for 3D point cloud segmentation, named CoSMix, which extends the approach presented in [23] to the semi-supervised settings (SSDA). CoSMix is designed to mitigate the domain shift by mixing semantically-informed groups of points (patches) across domains. Specifically, we design a two-branch symmetric deep neural network pipeline that concurrently processes point clouds from a source domain (e.g. synthetic or real) and point clouds from a target domain (e.g. real or real but captured with a different sensor). Target point clouds can be either unlabeled or partially labeled if one wants to use CoSMix for UDA or SSDA, respectively. Each branch is domain specific, i.e. the source branch is in charge of mixing a source point cloud with selected patches of a target point cloud, and vice versa for the target branch. We formulate mixing as a composition operation, which is similar to the concatenation operation

- C. Saltori, N. Sebe and E. Ricci are with the Dept. of Information Engineering and Computer Science, University of Trento, Italy. E-mails: cristiano.saltori@unitn.it, niculae.sebe@unitn.it, e.ricci@unitn.it.
- F. Galasso is with the Dept. of Computer Science, Sapienza University of Rome, Italy.
- G. Fiameni is with NVIDIA AI Technology Center, Italy.
- F. Poiesi and E. Ricci are with Fondazione Bruno Kessler (FBK), Trento, Italy. E-mail: poiesi@fbk.eu.

Manuscript received April 19, 2005; revised August 26, 2015.

proposed in [21], [22], but unlike them, we leverage the semantic information to mix domains. Patches from the source point cloud are selected based on the semantic labels of their points. Patches from the target point cloud can be selected based on the predicted semantic pseudo-labels in the case of UDA and based on human annotations in the case of SSDA. We will show that only a handful of manually annotated points are sufficient to significantly improve the domain adaptation performance. When patches are mixed across domains we apply data augmentation both at local and global semantic levels to boost the efficacy of the mixing. An additional key difference between our method and [21], [22] is the teacher-student learning scheme that we implement to improve the accuracy of the pseudo-labels. We evaluate CoSMix on large scale point cloud segmentation benchmarks, featuring both synthetic and real-world data, in several directions such as synthetic to real and real to real. Specifically, we use the following datasets: SynLiDAR [9], SemanticPOSS [6], SemanticKITTI [5], and nuScenes [7]. Our results show that CoSMix can reduce the domain shift, outperforming state-of-the-art methods in both UDA and SSDA settings. We perform detailed analyses of CoSMix and an ablation study of each component, highlighting its strengths and discussing its limitations.

This paper extends our earlier work [23] in several aspects. We extend the original CoSMix in order to tackle the SSDA setup. The current design allows a user to input a few annotated points to significantly improve the semantic segmentation performance on the target domain. Then, we significantly extend our experimental evaluation and analysis by adding new experiments, new comparisons, and new ablation studies to evaluate this new setup. We extend the related work by thoroughly reviewing additional state-of-the-art approaches, and summarizing these approaches in a comprehensive table that highlights key contributions and setups. Lastly, the code is available at <https://github.com/saltoricristiano/cosmix-uda>.

2 RELATED WORK

Point cloud semantic segmentation. Point cloud semantic segmentation can be performed at point level [37], on range views [38] or on a voxelized point clouds [39]. Point-level architectures process the input point cloud without the need for intermediate representation processing. This architectures include PointNet [40], which is based on a series of multilayer perceptrons. PointNet++ [37] improves on PointNet by aggregating global and local point features at multiple scales. RandLA-Net [41] extends PointNet++ [37] by embedding local spatial encoding, random sampling and attentive pooling. KPConv [42] learns weights in the continuous space, and introduces flexible and deformable convolutions for point cloud processing. These methods are computationally inefficient when large-scale point clouds are processed. Computational efficiency can be improved by projecting 3D points on 2D representations [25] or by using 3D quantization approaches [4]. The former includes 2D projection-based approaches that use 2D range maps and exploit standard 2D convolution filters [38] to segment these maps prior to a re-projection in the 3D

space. RangeNet++ [25], SqueezeSeg networks [20], [43], 3D-MiniNet [44] and PolarNet [45] are approaches that belong to this category. Although these approaches are efficient, they tend to lose information when the input data are projected in 2D and re-projected in 3D. The latter includes 3D quantization-based approaches that transform the input point cloud into a 3D discrete representations, and that employ 3D convolutions [39] or 3D sparse convolutions [4], [36] to predict per-point classes. VoxelNet [39] maps input points into a voxel-grid and processes the input voxel-grid with 3D convolutions. SparseConv [36], [46] and MinkowskiNet [4] improves voxel processing and introduce sparse convolutions to improve efficiency. Cylinder3D [47] further improves voxel processing for LiDAR data by using cylindrical and asymmetrical 3D convolutions. In our work, we use MinkowskiNet [4], which provides a trade off between accuracy and efficiency.

Sample mixing for 2D domain adaptation. Unsupervised domain adaptation (UDA) for image-based tasks is a well-studied problem [48], and there exist several methods using domain adversarial learning [49], [50], [51], regularization losses [50], [52], [52], self-training [17], [53], multi-task learning [54], and curriculum learning [55]. Mixup can also be used for domain adaptation [10], [11], [56]. For 2D semantic segmentation, several recent works have used sample mixing for domain adaptation, including BAPA-Net [57], CAMix [58], DACS [14], DSP [15], and DAFormer [59]. BAPA-Net employs a boundary-informed mixing strategy, while CAMix proposes a context-aware mask generation for domain mixing. DACS extends ClassMix [56] for domain adaptation by pasting specific source classes into target images. DSP improves on DACS by introducing self-training, soft-labels, and a double mixing strategy. Unlike these works, our method tackles UDA for 3D semantic segmentation. It should be noted that extending image mixing strategies to point clouds is not as straightforward, as point clouds are sparse in nature. Therefore, we introduce a novel mixing strategy specifically designed for 3D point clouds.

Domain adaptation for point cloud segmentation. Unlike domain adaptation for image-based tasks [48], [60], domain adaptation for point cloud segmentation still lacks a unified experimental setup to compare different approaches. We review domain adaptation approaches for point cloud segmentation by grouping them into range-view methods, multi-modal (2D&3D) methods, and 3D-focused methods. Tab. 1 provides a detailed summary of these approaches.

Range-view (RV) images are computed through a cylindrical projection of the input point cloud onto a 2D plane. After projection, RV images can be processed with existing 2D convolution networks. RV-based networks are affected by domain shift, which can be mitigated by using generative approaches [19], [24], feature alignment [19], [20], [24], and contrastive learning [27]. RayCast [24] tackles the real-to-real UDA problem by transferring the sensor pattern of the target domain to the source domain through ray casting. After training the deep network on the source data, a minimal-entropy correlation alignment loss is used to reduce domain shift [61]. SqueezeSegV2 [20] improves the SqueezeSeg [43] architecture, and reduces domain shift in the synth-to-real setup by aligning source and target features with a geodesic

TABLE 1: Overview of existing methods for unsupervised (UDA) and semi-supervised (SSDA) adaptation in point cloud segmentation. For each approach, we report the sensor setup (Setup), the architecture (Input data type and Model), and the source and target datasets. Then, we classify the adaptation strategy into mixup based, adversarial learning based, alignment based, generative based, self-training based and auxiliary task based. Furthermore, we report whether the implementation (Code) is publicly available.

Method	Setup	Architecture		Datasets		Settings		Adaptation						Code
		Input data	Model	Source	Target	UDA	SSDA	Mixup	Adv.	Align.	Gen.	Self-train.	Aux. task	
RayCast [24]	real-to-real	RV	RangeNet++ [25]	Sem.KITTI [5]	nuSc. [7]	✓				✓	✓			
ePointDA [19]	synth-to-real	RV	SqueezeSegV2 [20]	GTA-V [20]	KITTI [26] Sem.KITTI [5]	✓			✓	✓	✓			
SqueezeSegV2 [20]	synth-to-real	RV	SqueezeSegV2 [20]	GTA-V [20]	KITTI [26]	✓				✓				✓
Gated [27]	synth-to-real real-to-real	RV	SalsaNext [28]	GTA-V [20] nuScenes [7] KITTI [26]	KITTI [26] nuScenes [7]	✓					✓		✓	
xMUDA [29]	real-to-real	2D&3D	xMUDA [29]	nuSc. [7] KITTI [30] A2D2 [31]	nuSc. [7] KITTI [30] A2D2 [31]	✓				✓		✓		✓
Cross-modal [32]	real-to-real synth-to-real	2D&3D	xMUDA [29]	nuSc. [7] v.KITTI [33] A2D2 [31] Sem.KITTI [5] Waymo [34]	nuSc. [7] Sem.KITTI [5] Waymo [34]	✓	✓			✓		✓		✓
Complete&Label [35]	real-to-real	3D	SparseConv [36]	KITTI [26] Waymo [34] nuSc. [7]	KITTI [26] Waymo [34] nuSc. [7]	✓					✓			
PCT [9]	synth-to-real	3D	Minkowski [4]	SynLiDAR [9]	Sem.KITTI [5] Sem.POSS [6]	✓	✓				✓			✓
CoSMix [23]	synth-to-real	3D	Minkowski [4]	SynLiDAR [9]	Sem.KITTI [5] Sem.POSS [6]	✓		✓				✓		✓
Ours	real-to-real synth-to-real	3D	Minkowski [4]	SynLiDAR [9] SemKITTI [5]	Sem.KITTI [5] Sem.POSS [6] nuScenes [7]	✓	✓	✓				✓		✓

correlation alignment [61]. ePointDA [19] addresses domain shift in the synth-to-real UDA setup at both input level and feature level. At input level, a generative CycleGAN [62] is trained to simulate real sensor noise on synthetic source data. At feature level, a higher-order momentum loss [63] is used to learn domain agnostic features between source and target input data. Gated [27] states that domain shift between source and target point clouds can be mitigated by solving the sparsity shift and by introducing domain specific parameters. Given an input pair of source and target RV images, they first solve the sparsity difference thorough self-supervised completion and by applying a dropout mask. Then, residual gated adapters are added to the segmentation model to learn target specific parameters. None of the RV-based methods tackle the semi-supervised scenario.

Multi-modal models are designed to process the information captured by multiple input sensors, e.g. RGB cameras and LiDAR sensors are those typically used. Domain shift is tackled by enforcing prediction consistency among modalities and domains, and by using target (pseudo) labels. xMUDA [29] uses cross-modality and cross-domain consistency to learn a domain agnostic model in the real-to-real UDA setup. Cross-modal consistency exploits source labels and target pseudo-labels to produce consistent multi-modal predictions in both the domains. DeepCORAL feature alignment [64] is used to enforce feature alignment between source and target domains. In [32], xMUDA is extended to SSDA settings showing that cross-modal consistency is effective even in the semi-supervised settings.

3D methods can process input point clouds with or without prior voxelization. UDA approaches for 3D segmentation

include voxel-based architectures such as SparseConv [46] and MinkowskiNet [4]. Domain shift can be tackled by focusing on the problem of sparsity [9], [35] or by employing mix up strategies [23]. Complete&Label [35] reduces the sparsity difference between real domains by formulating the domain adaptation problem as a point cloud completion (or densification) problem. A self-supervised completion network is trained to make the sparse input point cloud denser. The pre-processed point clouds can then be used as intermediate domains in order to lower the domain shift. PCT [9] disentangles domain shift between synthetic and real point clouds into appearance and sparsity. Then, PCT learns an appearance translation module and a sparsity translation module. These modules are used for translating source data in the target modality. Translated data are then used together with ST [18] and APE [65] in the UDA and SSDA settings, respectively.

CoSMix [23] is a method that reduces domain shift in point cloud data by introducing a compositional semantic mixup strategy with a teacher-student learning scheme. The method obtains domain-invariant models/features by creating two new intermediate domains of composite point clouds: a mixed source and a mixed target. In the mixed target, source instances pull the target domain closer to the source domain, preventing overfitting from noisy pseudo-labels. In the mixed source, target instances (pseudo-labels) bring the target modality into the source domain, pulling the source domain closer to the target domain. The teacher-student learning scheme enables the iterative improvement of pseudo-labels, progressively reducing the domain gap. In this work, we extend CoSMix [23] to the SSDA settings by

allowing target labels to be mixed in the source and target (unlabeled) point clouds while improving adaptation. We also show how a small amount of target supervision can significantly improve the adaptation performance.

3 COMPOSITIONAL SEMANTIC MIX (CoSMix)

3.1 Preliminaries and definitions

CoSMix implements a teacher-student learning scheme that exploits the supervision from the source domain, the self-supervision from the target domain and, if available, the supervision from a few labeled target samples to improve the semantic segmentation on the target domain. Our method is trained on two different mixed point cloud sets. The first is the composition of the source point cloud with pseudo-labeled portions of points, or *patches*, of the unlabeled target point cloud. Target patches bring the target modality in the source domain making the altered source domain more similar to the target domain. The second is the composition of the unlabeled target point cloud with randomly selected patches of the source point cloud. Source patches make the altered target domain more similar to the source domain, preventing overfitting from noisy pseudo-labels. If available, labeled points of the target point clouds can also be used in both the mixed point cloud sets. This target supervision can further reduce domain shift. The teacher-student learning scheme iteratively improves pseudo labels, progressively reducing the domain gap. Fig. 1 illustrates the block diagram of CoSMix.

Let $\mathcal{S} = \{(\mathcal{X}^s, \mathcal{Y}^s)\}$ be the source dataset that is composed of $N^s = |\mathcal{S}|$ labeled point clouds, where \mathcal{X}^s is a point cloud and \mathcal{Y}^s is its point-level labels, and $|\cdot|$ is the cardinality of a set. Labels take values from a set of semantic classes $\mathcal{C} = \{c\}$, where c is a semantic class. Let $\mathcal{T}_U = \{\mathcal{X}_U^t\}$ be the unlabeled target dataset composed of $N_U^t = |\mathcal{T}_U|$ unlabeled point clouds. Let $\mathcal{T}_L = \{(\mathcal{X}_L^t, \mathcal{Y}_L^t)\}$ be the semi-supervised set of $N_L^t = |\mathcal{T}_L|$ labeled target point clouds with $N_L^t \ll N_U^t$.

On the upper branch, the source point cloud \mathcal{X}^s is mixed with selected patches of the target point cloud \mathcal{X}_U^t and, selected patches of the supervised point cloud \mathcal{X}_L^t when available. The unlabeled target patches from \mathcal{X}_U^t are subsets of points that correspond to the most confident pseudo-labels $\hat{\mathcal{Y}}_U^t$ that the teacher network produces during training. The supervised target patches are subsets of points that are randomly selected based on the class frequency distribution in the source training set. On the lower branch, the target point cloud \mathcal{X}_U^t is mixed with the selected patches of the source point cloud \mathcal{X}^s and with the selected patches of \mathcal{X}_L^t , if available. The source patches are subsets of points that are randomly selected based on their class frequency distribution in the training set.

We define the branch that mixes target point cloud patches to the source point cloud as $t \rightarrow s$ and the branch that does the vice versa as $s \rightarrow t$. Let $\mathcal{X}^{t \rightarrow s}$ be the mixed point cloud obtained from the upper branch, and $\mathcal{X}^{s \rightarrow t}$ be the mixed point cloud obtained from the lower branch. Lastly, let Φ_θ and $\Phi_{\theta'}$ be the student and teacher deep networks with learnable parameters θ and θ' , respectively.

3.2 Semantic selection

To train the student networks with balanced data, we perform a selection of reliable and informative point cloud patches prior to mixing points and labels across domains. To select patches from the source point cloud, we use the class frequency distribution by counting the number of points of each semantic class within \mathcal{S} . Unlike DSP [15] that selects long-tail classes in advance, we exploit the source distribution and the semantic classes available to dynamically sample classes at each iteration.

Let P_Y^s be the class frequency distribution of \mathcal{S} . We create a function f that randomly selects a subset of classes at each iteration based on the labels $\hat{\mathcal{Y}}^s \subset \mathcal{Y}^s$. f performs a weighted random sampling of α classes from the input point cloud by using $1 - P_Y^s$ as the class weight for each class. α is an hyperparameter that regulates the ratio of selected classes for each point cloud. The output of f is a set point-level labels belonging to the sampled classes, i.e. $\hat{\mathcal{Y}}^s$. The likelihood that f selects a class c is inversely proportional to its class frequency in \mathcal{S} . Formally we have

$$\hat{\mathcal{Y}}^s = f(\mathcal{Y}^s, 1 - P_Y^s, \alpha). \quad (1)$$

Example: with $\alpha = 0.5$, the algorithm selects a number of patches corresponding to the 50% of the available classes, i.e. long-tailed classes are selected with a higher likelihood.

Let $\tilde{\mathcal{X}}^s$ be the set of points that correspond to $\hat{\mathcal{Y}}^s$, and let $\tilde{\mathcal{X}}_c^s \subset \tilde{\mathcal{X}}^s$ be a patch (set of points) that belongs to class $c \in \mathcal{C}$. To select patches from the target point clouds, we apply the same set of operations but using the pseudo-labels produced by the teacher network based on their prediction confidence. Specifically, we define a function g that selects reliable pseudo-labels based on their confidence value. The selected pseudo-labels are defined as

$$\hat{\mathcal{Y}}_U^t = g(\Phi_{\theta'}(\mathcal{X}_U^t), \zeta), \quad (2)$$

where $\Phi_{\theta'}$ is the teacher network, ζ is the confidence threshold used by the function g and $\hat{\mathcal{Y}}_U^t \subset \hat{\mathcal{Y}}_U^t$.

Let $\tilde{\mathcal{X}}_U^t$ be the set of points that correspond to $\hat{\mathcal{Y}}_U^t$.

In the case of target supervision, we apply f to the target labels \mathcal{Y}_L^t and randomly select target patches as

$$\hat{\mathcal{Y}}_L^t = f(\mathcal{Y}_L^t, 1 - P_Y^s, \mu), \quad (3)$$

where μ is an hyperparameter that regulates the ratio of selected classes for each point cloud similarly to α .

3.3 Compositional mix

The goal of our compositional mixing module is to create mixed point clouds based on the selected semantic patches. The compositional mix involves three consecutive operations: *local random augmentation*, where patches are augmented randomly and independently from each other; *concatenation*, where the augmented patches are concatenated to the point cloud of the other domain to create the mixed point cloud; *global random augmentation*, where the mixed point cloud is randomly augmented. This module is applied twice, once for the $t \rightarrow s$ branch (top of Fig. 1), where target patches are mixed within the source point cloud, and once for the $s \rightarrow t$ branch (bottom of Fig. 1), where source patches are mixed within the target point cloud. Unlike Mix3D [21],

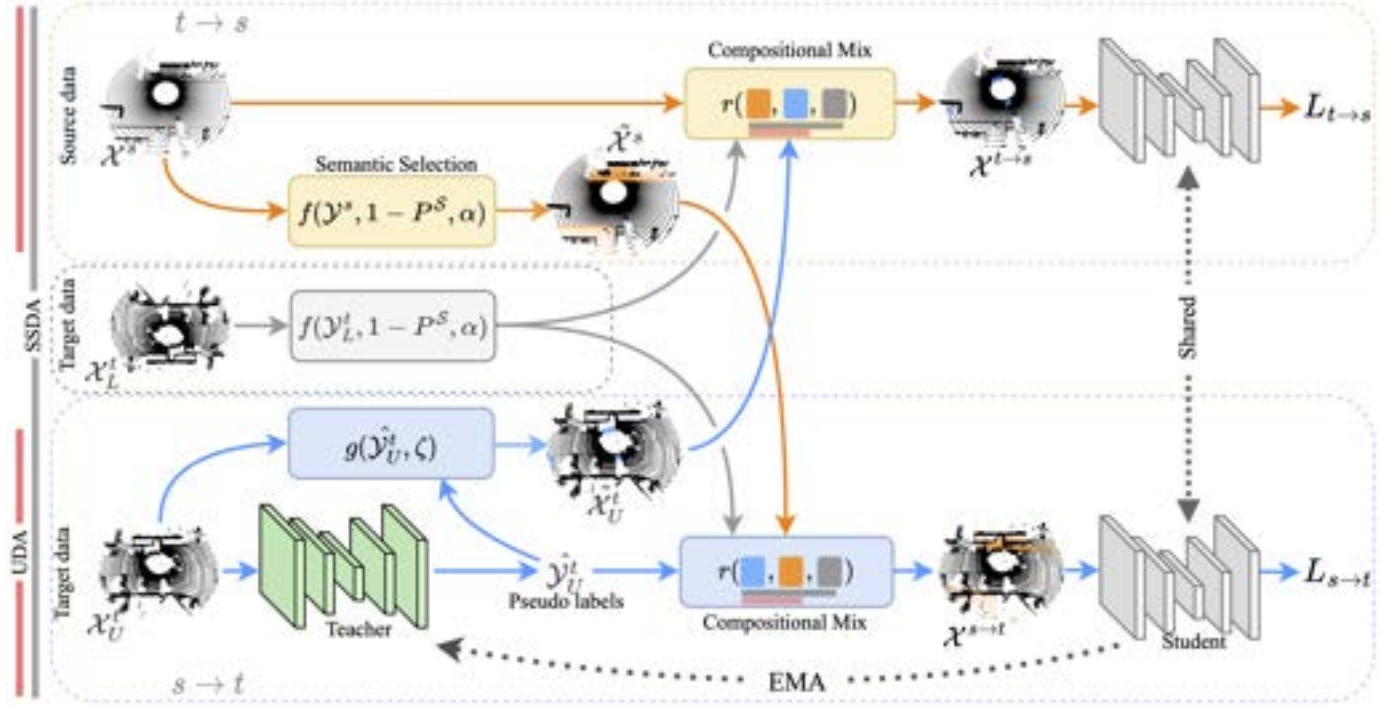


Fig. 1: Block diagram of CoSMix detailing the UDA and SSDA settings. The UDA setting uses the top and bottom branch (red line). The SSDA setting uses also the middle branch in addition to those used in UDA (gray line). In the top branch, the input source point cloud \mathcal{X}^s is mixed with the unsupervised target point cloud \mathcal{X}_U^t obtaining $\mathcal{X}^{t \rightarrow s}$. In the bottom branch, the input target point cloud \mathcal{X}_U^t is mixed with the source point cloud \mathcal{X}^s obtaining $\mathcal{X}^{s \rightarrow t}$. In the SSDA setting, the labeled target data \mathcal{X}_L^t are mixed with the source point cloud \mathcal{X}^s and with the unsupervised target point cloud \mathcal{X}_U^t . A teacher-student learning architecture is used in both the UDA and SSDA settings to improve pseudo-label accuracy while adapting over target domain. This is achieved by updating the teacher network through Exponential Moving Average (EMA). Semantic Selection (f and g) selects subsets of points (patches) to be mixed based on the source labels \mathcal{Y}^s , the target labels \mathcal{Y}_L^t , and target pseudo-labels $\hat{\mathcal{Y}}_U^t$ information. Compositional Mix applies local h and global r augmentations and mixes the selected patches among domains.

our mixing strategy embeds data augmentation at local level and global level.

Let δ be the indicator function that we define as

$$\delta(\mathcal{T}_L) = \begin{cases} 1 & \text{if } \mathcal{T}_L \neq \emptyset \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

which indicates whether the supervised target set \mathcal{T}_L is empty or not. This can be interpreted as the user desire or need to use additional target supervision.

In the $s \rightarrow t$ branch, we apply the local random augmentation h to all the points $\mathcal{X}_c^s \subset \tilde{\mathcal{X}}^s$. We repeat this operation for all $c \in \tilde{\mathcal{Y}}^s$. Note that h is a local and random augmentation that produces a different result each time it is applied to a set of points. We define the result of this operation as

$$h(\tilde{\mathcal{X}}^s) = \{h(\tilde{\mathcal{X}}_c^s), \forall c \in \tilde{\mathcal{Y}}^s\}. \quad (5)$$

If $\delta(\mathcal{T}_L) = 1$ we can apply h also to $\tilde{\mathcal{X}}_L^t$ and obtain

$$h(\tilde{\mathcal{X}}_L^t) = \{h(\tilde{\mathcal{X}}_c^t), \forall c \in \tilde{\mathcal{Y}}_L^t\}. \quad (6)$$

Then, we concatenate the locally augmented patches with the target point cloud \mathcal{X}_L^t and we apply the global

random augmentation, such as

$$\mathcal{X}^{s \rightarrow t} = \begin{cases} r(h(\tilde{\mathcal{X}}^s) \cup h(\tilde{\mathcal{X}}_L^t) \cup \mathcal{X}_U^t) & \text{if } \delta(\mathcal{T}_L) = 1, \\ r(h(\tilde{\mathcal{X}}^s) \cup \mathcal{X}_U^t) & \text{otherwise} \end{cases} \quad (7)$$

Their respective labels are concatenated accordingly as

$$\mathcal{Y}^{s \rightarrow t} = \begin{cases} r(h(\tilde{\mathcal{Y}}^s) \cup h(\tilde{\mathcal{Y}}_L^t) \cup \mathcal{Y}_U^t) & \text{if } \delta(\mathcal{T}_L) = 1, \\ r(h(\tilde{\mathcal{Y}}^s) \cup \mathcal{Y}_U^t) & \text{otherwise,} \end{cases} \quad (8)$$

where r is the global augmentation function.

The same operations of Eq. 7-8 are also performed in the $t \rightarrow s$ branch by mixing target patches within the source point cloud. Instead of using source labels, we use the teacher network to generate pseudo-labels from the target data. Additionally, we use target supervision if $\delta(\mathcal{T}_L) = 1$. Then, we concatenate them with the labels of the source data. This results in $\mathcal{X}^{t \rightarrow s}$ and $\mathcal{Y}^{t \rightarrow s}$. Note that \mathcal{T}_L may be used without compositional mix and without double branched mixing. We implement h and r by using typical augmentation strategies for point clouds [4], i.e. random rotation, scaling, and translation. We report additional information in Sec. 4.2.

3.4 Network update

We leverage the teacher-student learning scheme to facilitate the transfer of knowledge acquired during the course of the training with mixed domains. We use the teacher network $\Phi_{\theta'}$ to produce target pseudo-labels $\hat{\mathcal{Y}}_t^t$ for the student network Φ_{θ} , and train Φ_{θ} to segment target point clouds by using the mixed point clouds $\mathcal{X}^{s \rightarrow t}$ and $\mathcal{X}^{t \rightarrow s}$ based on their mixed labels and pseudo-labels (Sec. 3.3).

At each batch iteration, we update the student parameters Φ_{θ} to minimize a total objective loss \mathcal{L}_{tot} defined as

$$\mathcal{L}_{tot} = \mathcal{L}_{s \rightarrow t} + \mathcal{L}_{t \rightarrow s}, \quad (9)$$

where $\mathcal{L}_{s \rightarrow t}$ and $\mathcal{L}_{t \rightarrow s}$ are the $s \rightarrow t$ and $t \rightarrow s$ branch losses, respectively. Given $\mathcal{X}^{s \rightarrow t}$ and $\mathcal{Y}^{s \rightarrow t}$, we define the segmentation loss for the $s \rightarrow t$ branch as

$$\mathcal{L}_{s \rightarrow t} = \mathcal{L}_{seg}(\Phi_{\theta}(\mathcal{X}^{s \rightarrow t}), \mathcal{Y}^{s \rightarrow t}), \quad (10)$$

the objective of which is to minimize the segmentation error over $\mathcal{X}^{s \rightarrow t}$, thus learning to segment source patches in the target domain. Similarly, given $\mathcal{X}^{t \rightarrow s}$ and $\mathcal{Y}^{t \rightarrow s}$, we define the segmentation loss for the $t \rightarrow s$ branch as

$$\mathcal{L}_{t \rightarrow s} = \mathcal{L}_{seg}(\Phi_{\theta}(\mathcal{X}^{t \rightarrow s}), \mathcal{Y}^{t \rightarrow s}), \quad (11)$$

whose objective is to minimize the segmentation error over $\mathcal{X}^{t \rightarrow s}$ where target patches are composed with source data. We implement \mathcal{L}_{seg} as the Dice segmentation loss [66], which we found effective for the segmentation of large-scale point clouds as it can cope with long-tail classes well.

Lastly, we update the teacher parameters θ' every γ iterations following the exponential moving average (EMA) [67] approach

$$\theta'_i = \beta \theta'_{i-1} + (1 - \beta) \theta, \quad (12)$$

where i indicates the training iteration and β is a smoothing coefficient hyperparameter.

4 EXPERIMENTS

We evaluate our method in both synthetic-to-real and real-to-real UDA and SSDA settings. We use SynLiDAR [9] as synthetic dataset, and SemanticKITTI [5], [26], [30], SemanticPOSS [6], and nuScenes [7] as real-world datasets. We compare CoSMix with five state-of-the-art UDA methods: two general purpose adaptation methods (ADDA [49], EntMin [50]), one image segmentation method (ST [18]), and two point cloud segmentation methods (PCT [9], ST-PCT [9]). Then, we compare CoSMix with five state-of-the-art SSDA methods: three general purpose adaptation methods (MMD [49], MME [68], APE [65]), and two point cloud segmentation methods (PCT [9], APE-PCT [9]). We refer to CoSMix-UDA and CoSMix-SSDA to indicate the version of CoSMix for UDA and SSDA, respectively, we use CoSMix to refer to our method in general otherwise. PCT, ST-PCT and APE-PCT are the only three state-of-the-art methods developed for 360° LiDAR point clouds and have only been applied for synthetic-to-real UDA and SSDA settings. We re-implemented the comparison method and adapted to the same backbone network as that of CoSMix. We refer to these methods as EntMin* [50], ST* [18], MME* [68], MMD* [49], *Source**, *Target** and *Fine-tuned**. Moreover, we extended

EntMin [50] and ST [18] to the SSDA setting, and refer to them as EntMin-SSDA* and ST-SSDA*. For completeness, we also include the results of these methods as they are reported in [9].

4.1 Datasets and metrics

SynLiDAR [9] is a large-scale synthetic dataset that is created with the Unreal Engine [69]. It is composed of 198,396 annotated point clouds with 32 semantic classes. We use 19,840 point clouds for training and 1,976 point clouds for validation [9]. **SemanticPOSS** [6] is composed of 2,988 annotated real-world point cloud with 14 semantic classes. We use the sequence 03 for validation and the remaining sequences for training [6]. For the SSDA settings, we follow [9] and use the point cloud 172 of sequence 02 as the semi-supervised target set. **SemanticKITTI** [5] is a large-scale segmentation dataset consisting of LiDAR acquisitions of the popular KITTI dataset [26], [30]. It is composed of 43,552 annotated real-world point clouds with more than 19 semantic classes. We use sequence 08 for validation and the remaining sequences for training [5]. For the SSDA settings, we follow [9] and use the point cloud 848 from sequence 06 and the point cloud 940 from sequence 02 as semi-supervised target set. **nuScenes** [7] is a large-scale segmentation dataset. It is composed of real-world 850 sequences (700 for training and 150 for validation), for a total of 34,000 annotated point clouds with 32 semantic classes. We use the official training and validation splits in all our experiments. For the SSDA settings, we follow the same selection protocol used in [9] and use the point cloud with token n015-2018-07-24-11-13-19+0800_LIDAR_TOP_1532402013197655 as semi-supervised target set.

We make source and target labels compatible across our datasets, i.e. SynLiDAR \rightarrow SemanticPOSS, SynLiDAR \rightarrow SemanticKITTI and, SemanticKITTI \rightarrow nuScenes. In SynLiDAR \rightarrow SemanticPOSS and SynLiDAR \rightarrow SemanticKITTI, we follow [9] and map labels into 14 segmentation classes and 19 segmentation classes, respectively. In SemanticKITTI \rightarrow nuScenes we map source and target labels into 7 common segmentation classes as in [70].

We evaluate the semantic segmentation performance before and after domain adaptation [9] by using the Intersection over the Union (IoU) [71] for each segmentation class and report the per-class IoU. We average the IoU over all the segmented classes and report the mean Intersection over the Union (mIoU).

4.2 Implementation details

We implemented CoSMix in PyTorch and run our experiments on 4×NVIDIA A100 (40GB SXM4). We use MinkowskiNet as our point cloud segmentation network [4], in particular we use MinkUNet32 as in [9]. We pre-train our network on the source domain with Dice loss [66] starting from randomly initialized weights. In SSDA, we start from the pre-trained source model and finetune on both source and labeled target for two additional epochs. The finetuned model is used as pre-trained model in the semi-supervised settings. In UDA, we initialize student and teacher networks with the parameters obtained after

pre-training. The pre-training and adaptation stage share the same hyperparameters. In both the pre-training and adaptation steps, we use Stochastic Gradient Descent with a learning rate of 0.001.

We set the value of α by examining the long-tailed classes present in the source domain during the adaptation process. Similarly, we set the parameter μ to the same value. We assign the values of α and μ based on our prior experience, rather than optimizing these parameters through a systematic process. In the target semantic selection function g , we establish the value of ζ based on a qualitative assessment of a few target frames, with the aim of producing spatially compact predictions. This approach yields approximately 80% of pseudo-labeled points per scene.

On SynLiDAR \rightarrow SemanticPOSS, we use a batch size of 12 and perform adaptation for 10 epochs. We set source and supervised target semantic selection (f) with $\alpha = 0.5$ and $\mu = 0.5$ while we set target semantic selection (g) with a confidence threshold $\zeta = 0.85$. On SynLiDAR \rightarrow SemanticKITTI, we use a batch size of 16, adapting for 3 epochs. During source and supervised target semantic selection (f) we set $\alpha = 0.5$ and $\mu = 0.5$ while in target semantic selection (g) we use a confidence threshold of $\zeta = 0.90$. We use these last same hyperparameters also on SemanticKITTI \rightarrow nuScenes, and SynLiDAR \rightarrow nuScenes.

Our local augmentations h and global augmentations r are based on data augmentation strategies that are typical in the LiDAR segmentation literature [4]. h involves rigid rotation around the z -axis, scaling along all the axes and random point downsampling. We remove xy rotation to produce coplanar and concentric mixed point clouds, and to preserve point ranges. For the same reason, we remove rigid translations. We bound rotations between $[-\pi/2, \pi/2]$ and scaling between $[0.95, 1.05]$, and perform random downsampling for 50% of the patch points. r involves rigid rotation, translation and scaling along all the three axes. We set the parameters of r the same as those used in [4]. During the network update step (Sec. 3.4), we update the teacher parameters θ'_i with $\beta = 0.99$. On SynLiDAR \rightarrow SemanticPOSS, we set $\gamma = 1$ and do not perform parameter tuning. On SynLiDAR \rightarrow SemanticKITTI, we increase γ to $\gamma = 500$ to obtain a stable teacher behavior, i.e. stable source performance, high average confidence of pseudo-labels, and $\sim 80\%$ of pseudo-labeled points. We use these same hyperparameters also on SemanticKITTI \rightarrow nuScenes, and SynLiDAR \rightarrow nuScenes.

4.3 Quantitative comparisons for UDA

Synthetic-to-real. Tabs. 2&3 report the results in the UDA settings on SynLiDAR \rightarrow SemanticPOSS, and on SynLiDAR \rightarrow SemanticKITTI, respectively. The *Source** model is the lower bound of each scenario with 21.6 mIoU on SynLiDAR \rightarrow SemanticPOSS and 23.8 mIoU on SynLiDAR \rightarrow SemanticKITTI. The *Target** model is the upper bound of each scenario with 44.7 mIoU on SynLiDAR \rightarrow SemanticPOSS and 44.0 mIoU on SynLiDAR \rightarrow SemanticKITTI. Note that *Source** models always outperform *Source*. This may be due to a better parameter choice that leads an improved generalization ability. In SynLiDAR \rightarrow SemanticPOSS (Tab. 2), CoSMix-UDA outperforms the other methods on all the classes, except on *pole* where ST achieves a better result.

On average, we achieve 40.4 mIoU, surpassing ST-PCT by +10.8 mIoU and improving over the *Source** of +18.8 mIoU. CoSMix-UDA improves also on difficult classes as *person*, *traffic-sign*, *cone*, and *bike*, whose performance are rather low before domain adaptation. ST* and EntMin* improve over *Source**. ST* improves over ST while EntMin* achieves lower performance. Tab. 3 reports the results of SynLiDAR \rightarrow SemanticKITTI. SemanticKITTI is challenging as the validation sequence includes a wide range of different scenarios with a large number of semantic classes. CoSMix-UDA improves all the classes when compared to *Source**, except for *traffic-cone*. We believe this is due to the noise introduced by the pseudo labels on these classes and in related classes such as *road*. CoSMix-UDA improves on 10 out of 19 classes, with a large margin in the classes *car*, *motorcycle*, *truck*, *person*, *road*, *parking* and *sidewalk*. On average, we achieve state-of-the-art performance with a 32.2 mIoU, outperforming ST-PCT by +3.3 mIoU and improving over *Source** of about +8.4 mIoU.

Real-to-real. Tab. 4 reports the results on SemanticKITTI \rightarrow nuScenes in the UDA setting. SemanticKITTI \rightarrow nuScenes is a more challenging direction as source and target sensors are different, and nuScenes has rather sparse point clouds. *Source** and *Target** models achieve 40.1 mIoU and 62.0 mIoU, respectively. CoSMix-UDA outperforms the compared methods on 4 out of 7 classes, with the largest margin on the class *person*. On average, EntMin* and ST* achieve 43.4 mIoU and 43.6 mIoU, showing a limited improvement over *Source**. CoSMix-UDA achieves the best results of 46.2 mIoU, outperforming all the compared methods.

4.4 Quantitative comparison for SSDA

Synthetic-to-real. Tabs. 5&6 report the results in the SSDA settings on SynLiDAR \rightarrow SemanticPOSS, and on SynLiDAR \rightarrow SemanticKITTI, respectively. *Source** and *Target** models are the lower and upper bound of the UDA settings. The *Fine-tuned** model is obtained by fine-tuning *Source** with the semi-supervised target samples. It shows the highest possible bound without any adaptation approach. *Fine-tuned** always outperforms *Fine-tuned* from [9]. Similarly, the discrepancy between MMD* and MME*, and the results reported in [9] may due to a different parameter choice. In SynLiDAR \rightarrow SemanticPOSS (Tab. 5), CoSMix-SSDA outperforms all the comparison methods on all the classes, except on *plants*, *fence* and *bike* where MME and MME* achieve better results. On average, we reach 41.0 mIoU, outperforming APE-PCT by +9.8 mIoU and improving over *Source** by +19.4 and over *Fine-tuned** by +15.5. Compared to CoSMix-UDA, CoSMix-SSDA achieves a +0.6 mIoU, getting closer to the *Target* upper bound. In SynLiDAR \rightarrow SemanticKITTI (Tab. 6), CoSMix-SSDA brings a significant improvement on 12 out 19, especially on *car*, *truck*, *motorcyclist*, *road*, *parking*, and *pole*. On average, CoSMix-SSDA achieves 34.3 mIoU, outperforming the best baseline APE-PCT by +7.3 mIoU and improving over *Source** of +10.5 mIoU and over *Fine-tuned** of +9.8 mIoU. Compared to our UDA pipeline, CoSMix-SSDA improves of +2.1 mIoU, showing that the additional target supervision is beneficial for further reducing the domain gap.

TABLE 2: Unsupervised adaptation results on SynLiDAR \rightarrow SemanticPOSS. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	pers.	rider	car	trunk	plants	traf.	pole	garb.	buil.	cone.	fence	bike	grou.	mIoU
<i>Source</i>	3.7	25.1	12.0	10.8	53.4	0.0	19.4	12.9	49.1	3.1	20.3	0.0	59.6	20.7
<i>Source*</i>	21.7	20.1	9.7	3.4	56.8	4.8	24.1	6.1	39.9	0.3	15.3	5.3	73.4	21.6
<i>Target*</i>	61.8	54.7	33.0	19.3	73.9	26.7	30.9	11.0	71.3	32.5	44.6	43.2	78.5	44.7
ADDA [49]	27.5	35.1	18.8	12.4	53.4	2.8	27.0	12.2	64.7	1.3	6.3	6.8	55.3	24.9
Ent-Min [50]	24.2	32.2	21.4	18.9	61.0	2.5	36.3	8.3	56.7	3.1	5.3	4.8	57.1	25.5
Ent-Min* [50]	24.8	28.0	13.4	4.1	59.6	2.0	23.3	5.8	47.0	0.0	16.1	5.8	71.6	23.2
ST [18]	23.5	31.8	22.0	18.9	63.2	1.9	41.6	13.5	58.2	1.0	9.1	6.8	60.3	27.1
ST* [18]	47.7	42.6	24.4	13.8	62.5	3.3	36.1	23.5	50.9	18.8	14.6	4.0	68.9	31.6
PCT [9]	13.0	35.4	13.7	10.2	53.1	1.4	23.8	12.7	52.9	0.8	13.7	1.1	66.2	22.9
ST-PCT [9]	28.9	34.8	27.8	18.6	63.7	4.9	41.0	16.6	64.1	1.6	12.1	6.6	63.9	29.6
CoSMix-UDA	55.8	51.4	36.2	23.5	71.3	22.5	34.2	28.9	66.2	20.4	24.9	10.6	78.7	40.4

TABLE 3: Unsupervised adaptation results on SynLiDAR \rightarrow SemanticKITTI. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	bi.cle	mt.cle	truck	oth-v.	pers.	b.clst	m.clst	road	park.	sidew.	oth-g.	build.	fence	veget.	trunk	terra.	pole	traff.	mIoU
<i>Source</i>	42.0	5.0	4.8	0.4	2.5	12.4	43.3	1.8	48.7	4.5	31.0	0.0	18.6	11.5	60.2	30.0	48.3	19.3	3.0	20.4
<i>Source*</i>	60.7	1.9	22.0	10.3	8.0	16.7	11.3	20.3	70.4	6.4	40.4	0.0	25.6	8.6	59.5	18.4	29.1	29.0	13.9	23.8
<i>Target*</i>	90.0	6.3	20.3	63.0	18.1	31.1	39.6	5.8	90.9	29.0	74.7	4.0	85.4	23.3	83.9	46.2	62.2	40.7	20.6	44.0
ADDA [49]	52.5	4.5	11.9	0.3	3.9	9.4	27.9	0.5	52.8	4.9	27.4	0.0	61.0	17.0	57.4	34.5	42.9	23.2	4.5	23.0
Ent-Min [50]	58.3	5.1	14.3	0.3	1.8	14.3	44.5	0.5	50.4	4.3	34.8	0.0	48.3	19.7	67.5	34.8	52.0	33.0	6.1	25.8
Ent-Min* [50]	63.8	8.5	23.0	15.9	5.0	17.2	33.3	22.8	61.6	3.1	34.4	0.2	52.2	6.2	63.3	16.9	19.9	27.5	9.4	25.5
ST [18]	62.0	5.0	12.4	1.3	9.2	16.7	44.2	0.4	53.0	2.5	28.4	0.0	57.1	18.7	69.8	35.0	48.7	32.5	6.9	26.5
ST* [18]	69.7	6.4	18.3	4.4	5.8	14.8	23.3	20.2	54.2	5.3	34.1	0.1	44.3	5.1	63.5	16.8	26.9	30.6	12.2	24.0
PCT [9]	53.4	5.4	7.4	0.8	10.9	12.0	43.2	0.3	50.8	3.7	29.4	0.0	48.0	10.4	68.2	33.1	40.0	29.5	6.9	23.9
ST-PCT [9]	70.8	7.3	13.1	1.9	8.4	12.6	44.0	0.6	56.4	4.5	31.8	0.0	66.7	23.7	73.3	34.6	48.4	39.4	11.7	28.9
CoSMix-UDA	75.1	6.8	29.4	27.1	11.1	22.1	25.0	24.7	79.3	14.9	46.7	0.1	53.4	13.0	67.7	31.4	32.1	37.9	13.4	32.2

TABLE 4: Unsupervised adaptation results on SemanticKITTI \rightarrow nuScenes. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source real dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	pers.	road	side.	terr.	manm.	vege.	mIoU
<i>Source*</i>	29.4	15.6	73.2	29.1	14.7	58.5	59.9	40.1
<i>Target*</i>	35.8	43.2	93.6	62.1	49.0	76.4	73.9	62.0
EntMin* [50]	33.3	12.6	78.3	35.7	18.4	63.1	62.4	43.4
ST* [18]	30.6	20.6	79.1	34.4	18.9	62.4	59.3	43.6
CoSMix-UDA	32.1	26.3	78.1	35.1	20.2	66.4	65.2	46.2

Real-to-real. Tab. 7 reports the results on SemanticKITTI \rightarrow nuScenes in the SSDA settings. *Source** and *Target** models achieve 40.1 mIoU and 62.0 mIoU, respectively. The *Fine-tuned** model improves over *Source** and achieves 43.5 mIoU. CoSMix-SSDA achieves the best results on 3 out of 7 classes, with the largest margin on the class *pedestrian*. On average, MMD* is the best performing method among the comparison methods with 47.0 mIoU. CoSMix-SSDA

achieves 48.9 mIoU, outperforms all the comparison methods and further improving over CoSMix-UDA.

4.5 Domain adaptation between different sensors

We study the domain adaptation performance of CoSMix when the source point cloud is synthetically generated with a certain sensor and the target point cloud is captured in the real world with a different sensor. We use SynLiDAR as the source domain and nuScenes as the target domain. The semantic classes are mapped into the common 11 segmentation classes: *car*, *bicycle*, *motorcycle*, *truck*, *bus*, *pedestrian*, *road*, *sidewalk*, *building*, *vegetation* and *terrain*. This case exhibits a rather strong domain shift, as the SynLiDAR point clouds are dense and nearly noise-free, while the nuScenes point clouds are sparser and noisier. We follow the same implementation details of SemanticKITTI \rightarrow nuScenes, except we change the target domain. Tab. 8 reports the domain adaptation results on SynLiDAR \rightarrow nuScenes in both the UDA and SSDA settings. *Source** and *Target** models achieve 23.7 mIoU and 47.7 mIoU, respectively. *Fine-tuned** improves the performance to 26.4 mIoU. CoSMix-UDA improves over *Source** by achieving 27.3 mIoU. Despite the lack of target supervision, we also outperform the *Fine-tuned** baseline. CoSMix-SSDA further improves the results by achieving 27.6 mIoU when we introduce limited target supervision.

TABLE 5: Semi-supervised adaptation results on SynLiDAR \rightarrow SemanticPOSS. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	pers.	rider	car	trunk	plants	traf.	pole	garb.	buil.	cone.	fence	bike	grou.	mIoU
<i>Source</i>	3.7	25.1	12.0	10.8	53.4	0.0	19.4	12.9	49.1	3.1	20.3	0.0	59.6	20.7
<i>Source*</i>	21.7	20.1	9.7	3.4	56.8	4.8	24.1	6.1	39.9	0.3	15.3	5.3	73.4	21.6
<i>Fine-tuned</i>	25.2	36.1	18.2	12.8	58.6	1.7	30.5	5.6	25.7	3.0	12.0	10.6	75.6	24.3
<i>Fine-tuned*</i>	25.2	27.2	21.6	9.6	60.4	0.7	16.2	10.8	44.5	12.0	24.1	2.5	76.7	25.5
<i>Target*</i>	61.8	54.7	33.0	19.3	73.9	26.7	30.9	11.0	71.3	32.5	44.6	43.2	78.5	44.7
MMD [49]	25.5	35.7	28.9	6.7	64.3	1.7	23.2	5.6	53.3	3.3	30.2	13.9	70.4	27.9
MMD* [49]	28.1	12.2	18.8	11.4	71.5	10.0	14.7	0.0	64.6	0.0	28.1	25.1	78.6	27.9
MME [68]	33.2	40.2	25.0	11.0	61.9	0.4	31.2	7.3	56.1	5.7	37.1	6.7	71.2	29.8
MME* [68]	35.8	16.1	21.4	7.9	73.7	7.9	24.2	1.5	67.6	0.0	32.8	32.0	77.0	30.6
APE [65]	34.3	40.1	21.5	16.3	62.6	0.9	31.1	2.3	55.9	13.3	34.3	9.6	71.6	30.3
EntMin-SSDA*	24.7	9.4	16.5	10.7	69.9	6.7	11.7	0.0	62.6	0.0	25.2	22.5	78.9	26.1
ST-SSDA*	40.1	24.3	22.5	7.9	70.2	13.4	21.7	1.4	66.9	0.1	34.7	32.0	78.1	31.8
PCT [9]	25.8	36.8	27.8	11.3	62.2	1.9	31.2	5.2	58.7	2.6	34.3	8.5	68.7	28.8
APE-PCT [9]	34.7	36.3	27.2	15.8	62.9	0.8	31.6	8.7	62.3	9.8	35.1	9.3	70.9	31.2
CoSMix-SSDA	54.9	50.6	33.4	22.5	73.0	13.6	38.4	26.4	68.5	16.2	29.6	27.4	79.0	41.0

TABLE 6: Semi-supervised adaptation results on SynLiDAR \rightarrow SemanticKITTI. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	bicle	mtcle	truck	oth-v.	pers.	b.clst	m.clst	road	park.	sidew.	oth-g.	build.	fence	veget.	trunk	terra.	pole	traff.	mIoU
<i>Source</i>	42.0	5.0	4.8	0.4	2.5	12.4	43.3	1.8	48.7	4.5	31.0	0.0	18.6	11.5	60.2	30.0	48.3	19.3	3.0	20.4
<i>Source*</i>	60.7	1.9	22.0	10.3	8.0	16.7	11.3	20.3	70.4	6.4	40.4	0.0	25.6	8.6	59.5	18.4	29.1	29.0	13.9	23.8
<i>Fine-tuned</i>	56.2	3.0	15.1	1.0	5.0	20.2	42.1	2.8	52.1	0.7	19.8	0.0	41.3	5.8	62.1	34.0	42.0	24.6	1.4	22.6
<i>Fine-tuned*</i>	61.8	2.8	21.7	10.8	4.2	14.5	18.0	15.9	65.6	6.4	40.2	0.0	34.2	7.0	60.6	22.1	39.8	32.2	8.4	24.5
<i>Target*</i>	90.0	6.3	20.3	63.0	18.1	31.1	39.6	5.8	90.9	29.0	74.7	4.0	85.4	23.3	83.9	46.2	62.2	40.7	20.6	44.0
MMD [49]	56.4	3.3	13.3	1.5	6.1	21.4	34.6	1.6	54.3	0.4	21.4	0.0	50.2	5.8	61.2	37.0	44.9	31.6	2.2	23.5
MMD* [49]	46.5	3.2	6.3	12.1	3.3	8.8	21.7	13.4	47.2	4.6	29.9	0.0	50.6	5.9	62.2	16.2	23.3	19.4	4.7	20.0
MME [68]	51.0	5.6	13.1	1.3	7.3	15.1	54.4	4.4	43.1	0.2	28.3	0.0	60.7	13.3	66.1	30.1	39.9	24.8	6.6	24.5
MME* [68]	28.7	0.2	1.0	1.8	0.9	2.0	1.6	3.5	53.6	1.8	31.1	0.0	40.6	7.2	57.6	12.1	26.7	14.4	0.1	15.0
APE [65]	58.6	6.2	16.6	3.1	11.3	14.2	35.8	3.7	61.5	1.7	30.3	0.0	54.7	15.4	64.6	20.0	45.5	23.9	9.1	25.1
EntMin-SSDA*	52.0	2.6	7.8	10.3	3.5	8.4	20.9	13.2	42.1	3.5	31.2	0.0	44.1	5.8	62.5	15.2	22.7	18.2	5.8	19.5
ST-SSDA*	60.0	2.8	10.6	14.6	5.0	10.4	19.2	20.8	63.9	4.5	35.7	0.1	43.4	7.1	62.2	13.3	26.9	24.8	10.4	22.9
PCT [9]	56.0	7.0	17.1	2.8	9.9	23.7	43.7	5.6	55.3	0.8	22.9	0.0	50.1	8.4	65.3	23.1	43.5	28.8	7.5	24.8
APE-PCT [9]	58.1	7.3	17.8	2.6	13.9	24.7	46.5	5.1	60.5	1.9	31.3	0.0	56.8	14.6	67.9	23.7	44.3	26.1	9.3	27.0
CoSMix-SSDA	76.9	10.4	27.1	23.1	13.4	24.0	21.7	27.9	75.8	17.9	49.7	0.1	60.3	14.7	69.8	36.8	40.9	45.6	16.2	34.3

We observed a lower improvement of CoSMix compared to the other adaptation directions, which we attribute to the different simulated sensor and the large density difference between SynLiDAR and nuScenes scans.

4.6 Qualitative results

Fig. 2 shows some domain adaptation results on SynLiDAR \rightarrow SemanticPOSS. Predictions of *Source** are often incorrect. CoSMix-UDA improves the segmentation results with more homogeneous regions and correctly assigned classes, and CoSMix-SSDA further improves the segmentation quality.

Fig. 3 shows the results on SynLiDAR \rightarrow SemanticKITTI that follows the same result pattern as in Fig. 2. Some classes (e.g. *car*, *vegetation*, *pole*) greatly improve when CoSMix-SSDA is used. An evident increment of performance can be observed from *Source** to CoSMix-UDA to CoSMix-SSDA in both the studied domains. Although the limited amount of target supervision used in CoSMix-SSDA, these experiments show evidence of the benefits of our SSDA method.

5 ABLATION STUDY

We investigate the performance of CoSMix in both its UDA and SSDA variants by using the SynLiDAR \rightarrow SemanticPOSS setup. The first three experiments are designed to study CoSMix in the UDA setting. In Sec. 5.1, we analyze CoSMix-UDA components. In Sec. 5.2, we compare our mixing approach with three recent point cloud mixing strategies, namely, Mix3D [21], PointCutMix [72] and PolarMix [73]. In Sec. 5.3, we investigate the robustness of CoSMix to noisy pseudo-labels by changing the confidence threshold ζ and with different pre-trained models. In the last experiment (Sec. 5.4), we analyze CoSMix in the SSDA setting, comparing our semi-supervised mixing approach with three variations of our approach.

5.1 Method components

We analyze CoSMix by organizing its components into three groups: mixing strategies (*mix*), augmentations (*aug*s) and other components (*others*). In the *mix* group, we assess the

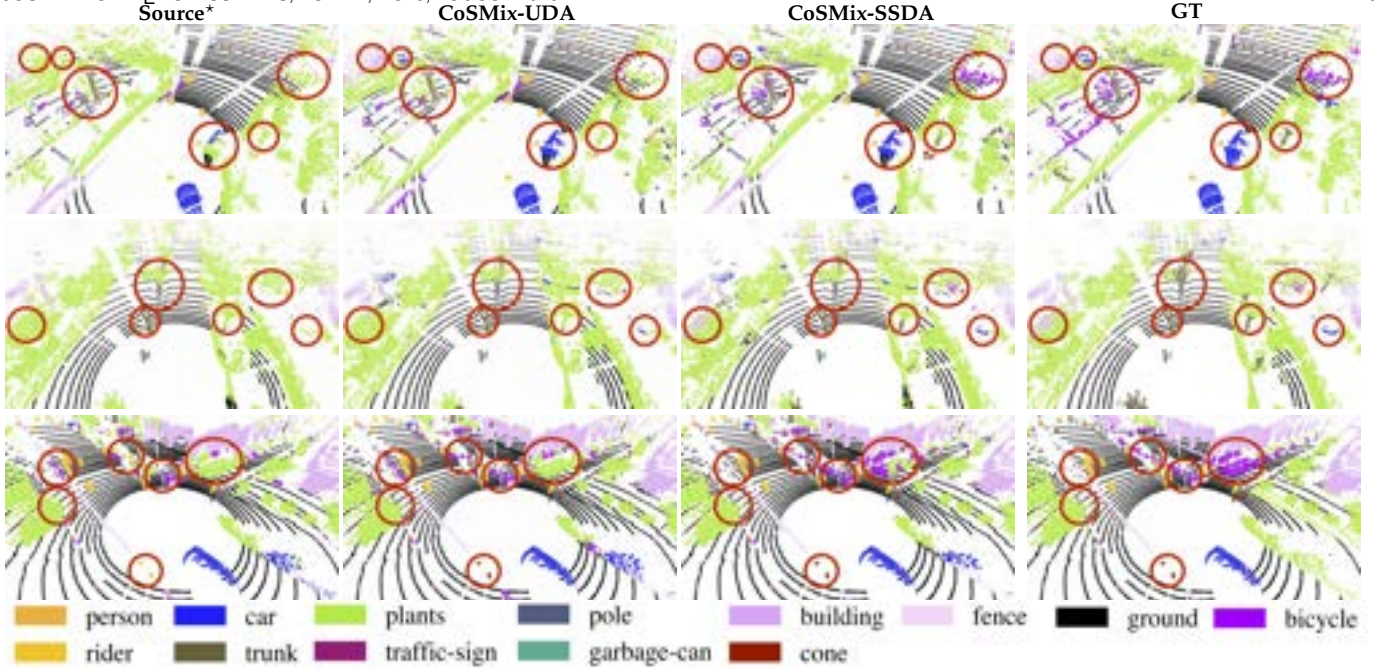


Fig. 2: Results on SynLiDAR \rightarrow SemanticPOSS. *Source** predictions are often wrong and mingled in the same region. After adaptation, CoSMix-UDA and CoSMix-SSDA improves segmentation with homogeneous predictions and correctly assigned classes. The red circles highlight regions with interesting results.

TABLE 7: Semi-supervised adaptation results on SemanticKITTI \rightarrow nuScenes. We denote our reproduced base-lines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source real dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	pers.	road	side.	terr.	manm.	vege.	mIoU
<i>Source*</i>	29.4	15.6	73.2	29.1	14.7	58.5	59.9	40.1
<i>Fine-tuned*</i>	47.0	22.9	75.1	28.6	15.3	61.8	53.7	43.5
<i>Target*</i>	35.8	43.2	93.6	62.1	49.0	76.4	73.9	62.0
MMD* [49]	38.3	14.1	83.2	32.4	33.9	63.7	63.2	47.0
MME* [68]	41.0	9.5	83.9	31.7	32.9	63.3	57.8	45.7
EntMin-SSDA*	31.8	13.6	81.8	35.5	30.7	66.2	65.7	46.5
ST-SSDA*	44.9	13.9	71.9	22.6	34.1	68.0	67.7	46.2
CoSMix-SSDA	45.3	26.9	80.1	34.5	20.8	68.0	67.0	48.9

importance of the mixing strategies ($t \rightarrow s$ and $s \rightarrow t$) used in our compositional mix (Sec. 3.3) after semantic selection. In the *aug*s group, we assess the importance of the local h and global r augmentations that are used in the compositional mix (Sec. 3.3). In the *others* group, we assess the importance of the mean teacher update (β) (Sec. 3.4) and of the long-tail weighted sampling f (Sec. 3.2). When the $t \rightarrow s$ branch is active, also the pseudo-label filtering g is utilized, while when f is not active, $\alpha = 0.5$ source classes are selected randomly. With different combinations of components, we obtain different versions of CoSMix which we name CoSMix (a-h). The complete version of our method is named *Full*, where all the components are activated. The *Source** performance is also added as a reference for the

lower bound. See Tab. 9 for the definition of these different versions.

When the $t \rightarrow s$ branch is used, CoSMix (a) achieves an initial 31.6 mIoU showing that the $t \rightarrow s$ branch provides a significant adaptation contribution over the *Source**. When we also use the $s \rightarrow t$ branch and the mean teacher β , CoSMix (b-d) further improve performance achieving a 35.4 mIoU. By introducing local and global augmentations in CoSMix (e-h), we can improve performance up to 39.1 mIoU. The best performance of 40.4 mIoU is achieved with CoSMix Full where all the components are activated.

5.2 Point Cloud Mix

We compare CoSMix with Mix3D [21], PointCutMix [72] and PolarMix [73] to show the effectiveness of the different mixing designs. As per our knowledge, Mix3D and PolarMix are the only mixup strategies designed for 3D semantic segmentation, while PointCutMix and PolarMix are the only strategies for mixing portions of different point clouds. We implement Mix3D and PointCutMix based on authors descriptions: we concatenate point clouds (random crops for PointCutMix) of the two domains, i.e., \mathcal{X}^s and \mathcal{X}^t , as well as their labels and pseudo-labels, i.e., \mathcal{Y}^s and $\hat{\mathcal{Y}}^t$, respectively. PolarMix [73] uses our same experimental settings and backbone therefore we consider the results reported in their manuscript. We refer to these mixing strategies as Mix3D*, PointCutMix* and, PolarMix*. CoSMix *double* is our two-branch network with sample mixing. For a fair comparison, we deactivate the weighted sampling and the mean teacher update. We keep local and global augmentations activated.

Fig. 5a shows that Mix3D* outperforms the *Source** model, achieving 28.5 mIoU, followed by PolarMix* which achieves 30.4 mIoU. PointCutMix* reaches 31.6 mIoU, outperforming the previous strategies. When we use the $t \rightarrow s$

TABLE 8: Adaptation results on SynLiDAR→nuScenes. We denote our reproduced baselines and results with *, e.g., *Source**. *Source** and *Target** correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	bicycle	motorcycle	truck	bus	pedestrian	road	sidewalk	building	vegetation	terrain	mIoU
<i>Source*</i>	23.7	2.8	10.3	15.8	4.9	20.8	63.6	18.0	47.7	50.0	3.3	23.7
<i>Fine-tuned*</i>	27.9	3.4	14.2	15.9	5.0	22.1	66.5	18.6	58.8	54.5	3.9	26.4
<i>Target*</i>	38.4	12.3	22.2	33.4	37.0	40.1	91.6	56.7	75.3	71.6	46.0	47.7
CoSMix-UDA	27.8	4.8	11.9	16.1	6.4	24.2	67.1	19.6	60.3	59.2	2.7	27.3
CoSMix-SSDA	28.8	4.6	7.8	11.1	10.1	22.7	72.8	21.6	60.4	60.5	3.2	27.6

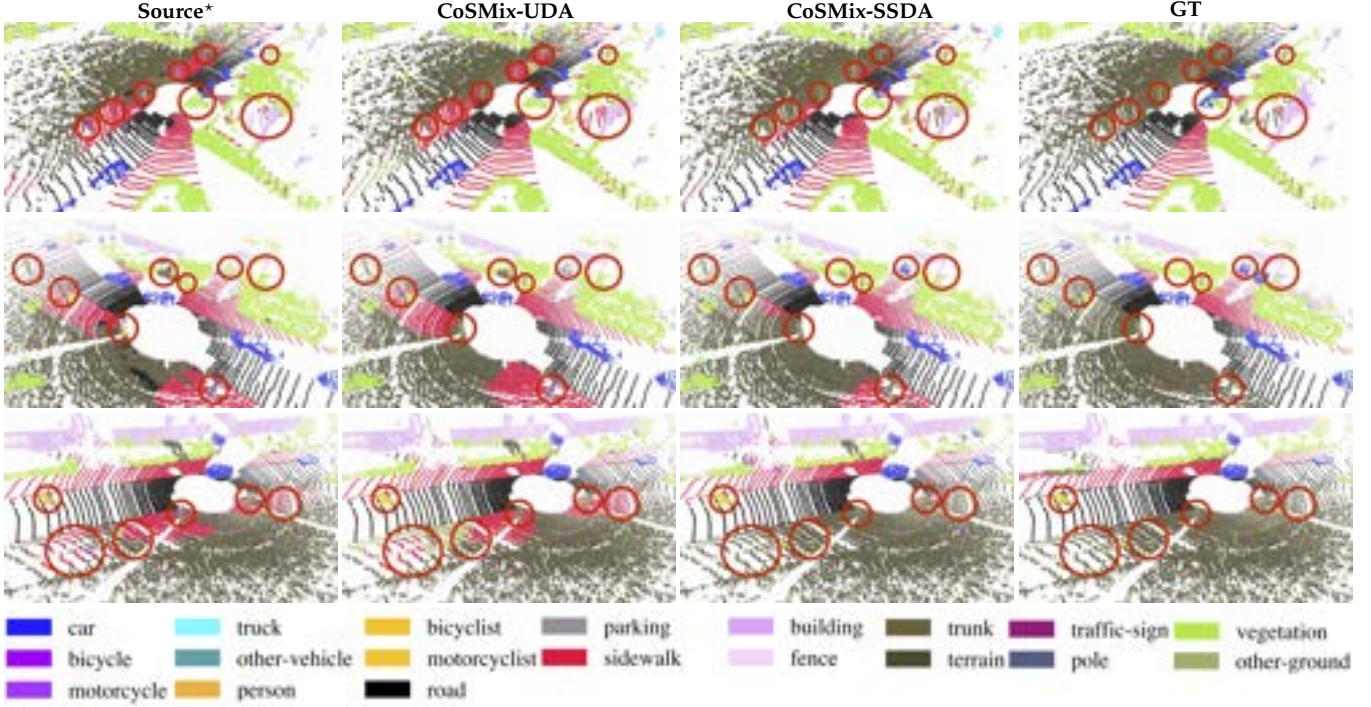


Fig. 3: Results on SynLiDAR → SemanticKITTI. *Source** predictions are often wrong and mingled in the same region. After adaptation, CoSMix-UDA and CoSMix-SSDA improves segmentation with homogeneous predictions and correctly assigned classes. The red circles highlight regions with interesting results.

TABLE 9: Ablation study of the CoSMix components: mixing strategy ($t \rightarrow s$ and $s \rightarrow t$), compositional mix augmentations (local h and global r), mean teacher update (β) and, weighted class selection in semantic selection (f). Each combination is named with a different version (a-h). *Source** performance are added as lower bound and highlighted in gray to facilitate the reading.

CoSMix version	mix		aug		others		mIoU
	$t \rightarrow s$	$s \rightarrow t$	h	r	β	f	
<i>Source*</i>	-	-	-	-	-	-	21.6
(a)	✓						31.6
(b)	✓				✓		31.9
(c)	✓	✓					35.0
(d)	✓	✓			✓		35.4
(e)	✓	✓	✓		✓		36.8
(f)	✓	✓		✓	✓		37.3
(g)	✓	✓	✓	✓	✓		39.0
(h)	✓	✓	✓	✓	✓	✓	39.1
Full	✓	✓	✓	✓	✓	✓	40.4

branch alone we can achieve 32.9 mIoU and when we use the $s \rightarrow t$ branch alone, CoSMix can further improve the results, achieving 34.8 mIoU. This shows that the supervi-

sion from the source to target is effective for adaptation on the target domain. When we use the contribution from both branches simultaneously, CoSMix achieves the best result with 38.9 mIoU.

5.3 Robustness to noisy pseudo-labels

We investigate the robustness of CoSMix to increasingly noisier pseudo-labels. Firstly, we study the effect of different confidence thresholds ζ . Secondly, we evaluate different versions of pre-trained models that we use for generating pseudo-labels.

Confidence threshold. We study the importance of setting the correct confidence threshold ζ for pseudo-label distillation in g (Sec. 3.2). We repeat the experiments with a confidence threshold from 0.65 to 0.95 and report the obtained adaptation performance in Fig. 5b. CoSMix is robust to noisy pseudo-labels reaching a 40.2 mIoU with the low threshold of 0.65. The best adaptation performance of 40.4 mIoU is achieved with a confidence threshold of 0.85. By using a high confidence threshold of 0.95 performance is affected reaching 39.2 mIoU. With this configuration, too few pseudo-labels are selected to provide an effective contribution for the adaptation.

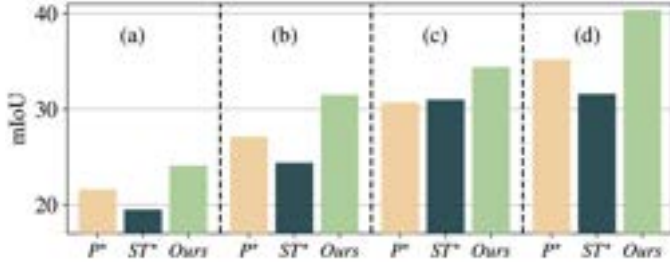


Fig. 4: Adaptation results on SynLiDAR→SemanticPOSS with different pre-trained models. We compare the adaptation results of CoSMix (Ours) with ST* starting from different initialization points (P^*) indicated with (a-d).

Model pre-training. We quantify the robustness of CoSMix and ST* [18] in response to pseudo-labels generated with different pre-trained models on SynLiDAR and tested on SemanticPOSS. In this experiment, we only utilize ST* as it is the sole method from those we benchmarked that is based on pseudo-labels. We denote the pre-trained model as P^* . Fig. 4 displays its performance at different epochs: (a) 1, (b) 2, (c) 4, and (d) 9. Unlike CoSMix, ST* proves sensitive to pseudo-labels as it underperforms P^* in three out of the four cases. A plausible explanation for this is that ST* refines the pre-trained model using filtered pseudo-labels during adaptation, depending on the quality of pseudo-labels. This dependency may cause ST* to drift during the adaptation process, thus impacting performance. Differently, CoSMix blends source and target (pseudo) labels, producing two intermediate domains with mixed labels. In the mixed point clouds, pseudo-labels are integrated with (noise-free) source labels ($t \rightarrow s$) or noise-free selections ($s \rightarrow t$), thus mitigating the negative effects of noisy and imprecise regions. Furthermore, our application of a teacher-based approach allows us to rely on progressively more precise pseudo-labels, thereby minimizing undesirable drift effects.

5.4 Mixing target supervision

We compare CoSMix-SSDA to three alternative mixing strategies: naive, $sup \rightarrow s$ and $sup \rightarrow t$. In Fig. 5c, we name each strategy as CoSMix-SSDA (a-c). In version CoSMix-SSDA (a), we apply CoSMix-UDA without mixing \mathcal{T}_L in $\mathcal{X}^{s \rightarrow t}$ and $\mathcal{X}^{t \rightarrow s}$. Dice segmentation loss is applied separately on \mathcal{T}_L and averaged with our total objective loss in Eq. 9. In the single branch mixing with source point clouds ($sup \rightarrow s$) and with target point clouds ($sup \rightarrow t$), versions (b-c), we apply only the upper or lower branch of CoSMix-SSDA, respectively. Full is our proposed double branched CoSMix-SSDA.

CoSMix-SSDA (a) approach reaches 33.7 mIoU, which shows that traditional training by using labeled target points as is leads to inferior performance than using our SSDA approach. Both the single branch mixing strategies achieve better performance with 38.9 mIoU and 40.5 mIoU for $sup \rightarrow s$ and $sup \rightarrow t$, respectively. The version (b) shows that the mixed target modality with noise-free annotations helps in reducing the domain shift. The version (c) suggests that the addition of target noise-free labels helps us in achieving higher performance. However, both the single

branch approaches are not sufficient to outperform the Full mixing strategy.

6 CONCLUSIONS

We introduced the first method for domain adaptation in 3D semantic segmentation, featuring a novel 3D point cloud mixing strategy that harnesses both semantic and structural information simultaneously. We developed two variations of our approach: one for unsupervised adaptation (CoSMix-UDA) and another for semi-supervised adaptation (CoSMix-SSDA). We performed comprehensive evaluations in both synthetic-to-real and real-to-real contexts within UDA and SSDA settings, utilizing large-scale, publicly available LiDAR datasets. Experimental results demonstrated that our approach significantly surpasses current state-of-the-art methods in both contexts. Moreover, detailed analyses underscored the significance of each component within CoSMix, confirming that our mixing strategy effectively addresses the issue of domain shift in 3D LiDAR segmentation. A primary limitation of CoSMix is its reliance on pseudo-labels, making the quality of the initial warm-up model on the source domain crucial to the adaptation performance on the target domain. An alternative approach could involve the implementation of self-supervised learning in lieu of using source data. Future avenues for research might encompass the incorporation of self-supervised learning tasks, domain generalization, extending CoSMix to source-free adaptation tasks, and its application to 3D object detection.

ACKNOWLEDGMENTS

This work was partially supported by OSRAM GmbH, by the MUR PNRR project FAIR (PE00000013) funded by the NextGenerationEU, by the EU project FEROX Grant Agreement no 101070440, by the PRIN PREVUE (Prot. 2017N2RK7K), the EU ISFP PROTECTOR (101034216) and the EU H2020 MARVEL (957337) projects and, it was carried out in the Vision and Learning joint laboratory of FBK and UNITN.

REFERENCES

- [1] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *IV*, 2011.
- [2] Y. Zhou, Y. Wang, F. Poiesi, Q. Qin, and Y. Wan, "Loop closure detection using local 3D deep descriptors," *RAL*, 2022.
- [3] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE SPM*, 2020.
- [4] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019.
- [5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *ICCV*, 2019.
- [6] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao, "Semanticpos: A point cloud dataset with large quantity of dynamic instances," *arXiv*, 2020.
- [7] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.
- [8] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv*, 2017.
- [9] A. Xiao, J. Huang, D. Guan, F. Zhan, and S. Lu, "Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation," in *AAAI*, 2022.

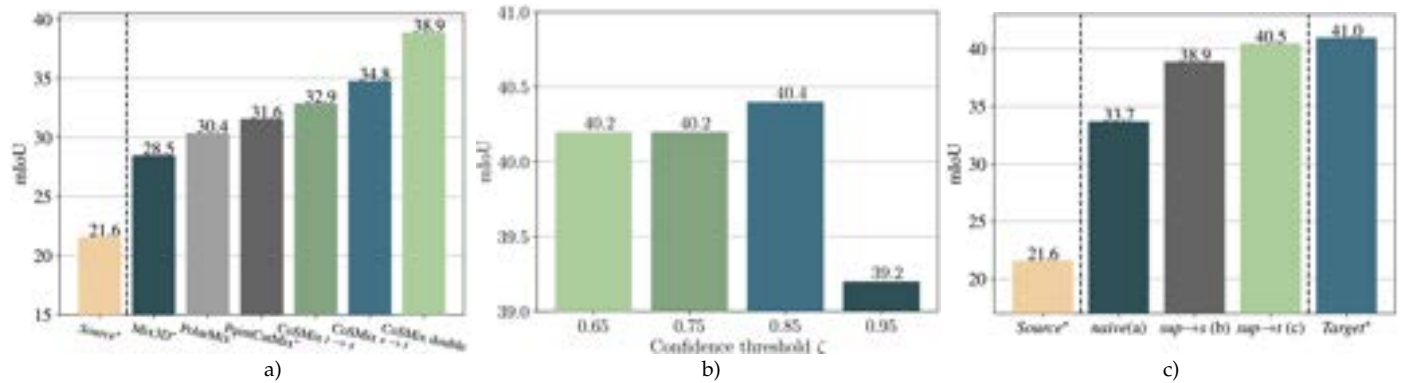


Fig. 5: **a)** Comparison of the adaptation performance with different point cloud mix up strategies. Compared to the recent mixing strategies Mix3D [21], PointCutMix [72] and, PolarMix [73], our mixing strategy and its variations achieve superior performance. **b)** Comparison of the adaptation performance on confidence threshold values. Adaptation results show that ζ should be set such that to achieve a trade-off between pseudo-label correctness and object completeness. **c)** Comparison of the SSDA performance with different mixing strategies: optimization without mix (naive), single branch mixing with source point clouds ($sup \rightarrow s$), single branch mixing with unsupervised target point clouds ($sup \rightarrow t$). Each variation is named with a different version (a-c). In all the experiments, *Source** and *Target** performance is the lower and upper bound.

- [10] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.
- [11] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019.
- [12] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang, "Adversarial domain adaptation with domain mixup," in *AAAI*, 2020.
- [13] M. Mancini, Z. Akata, E. Ricci, and B. Caputo, "Towards recognizing unseen categories in unseen domains," in *ECCV*, 2020.
- [14] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson, "Dacs: Domain adaptation via cross-domain mixed sampling," in *WACV*, 2021.
- [15] L. Gao, J. Zhang, L. Zhang, and D. Tao, "Dsp: Dual soft-paste for unsupervised domain adaptive semantic segmentation," in *ACMM*, 2021.
- [16] Q. Wang, D. Dai, L. Hoyer, L. V. Gool, and O. Fink, "Domain adaptive semantic segmentation with self-supervised depth estimation," in *ICCV*, 2021.
- [17] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *ECCV*, 2018.
- [18] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, "Confidence regularized self-training," in *ICCV*, 2019.
- [19] S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell, and K. Keutzer, "epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation," *arXiv*, 2020.
- [20] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *ICRA*, 2019.
- [21] A. Nekrasov, J. Schult, O. Litany, B. Leibe, and F. Engelmann, "Mix3D: Out-of-Context Data Augmentation for 3D Scenes," in *3DV*, 2021.
- [22] L. Zou, H. Tang, K. Chen, and K. Jia, "Geometry-aware self-training for unsupervised domain adaptation on object point clouds," in *CVPR*, 2021.
- [23] C. Saltori, F. Galasso, G. Fiameni, N. Sebe, E. Ricci, and F. Poiesi, "Cosmix: Compositional semantic mix for domain adaptation in 3d lidar segmentation," in *ECCV*, 2022.
- [24] F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss, "Domain transfer for semantic segmentation of LiDAR data using deep neural networks," in *IROS*, 2021.
- [25] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *IROS*, 2019.
- [26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *IJRR*, 2013.
- [27] M. Rochan, S. Aich, E. R. Corral-Soto, A. Nabatchian, and B. Liu, "Unsupervised domain adaptation in lidar semantic segmentation with self-supervision and gated adapters," in *ICRA*, 2022.
- [28] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *ISVC*, 2020.
- [29] M. Jaritz, T.-H. Vu, R. de Charette, E. Wirbel, and P. Pérez, "xMUDA: Cross-modal unsupervised domain adaptation for 3d semantic segmentation," in *CVPR*, 2020.
- [30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *CVPR*, 2012.
- [31] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2D2: Audi Autonomous Driving Dataset," *arXiv*, 2020.
- [32] M. Jaritz, T.-H. Vu, R. D. Charette, E. Wirbel, and P. Pérez, "Cross-modal learning for domain adaptation in 3d semantic segmentation," *T-PAMI*, 2022.
- [33] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," *arXiv*, 2020.
- [34] P. Sun, H. Kretschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *CVPR*, 2020.
- [35] L. Yi, B. Gong, and T. Funkhouser, "Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds," *arXiv*, 2021.
- [36] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv*, 2017.
- [37] C. Qi, L. Yi, H. Su, and L. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv*, 2017.
- [38] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.
- [39] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018.
- [40] C. Qi, H. Su, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.
- [41] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *CVPR*, 2020.
- [42] H. Thomas, C. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and J. L. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCV*, 2019.
- [43] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *ICRA*, 2018.
- [44] I. Alonso, L. Riazuelo, L. Montesano, and A. Murillo, "3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation," in *IROS*, 2020.
- [45] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *CVPR*, 2020.

- [46] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *CVPR*, 2018.
- [47] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *CVPR*, 2021.
- [48] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: a review," *Technologies*, 2020.
- [49] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *CVPR*, 2017.
- [50] T. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *CVPR*, 2019.
- [51] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *ICML*, 2018.
- [52] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci, "Unsupervised domain adaptation using feature-whitening and consensus loss," in *CVPR*, 2019.
- [53] A. Pilzer, S. Lathuilière, N. Sebe, and E. Ricci, "Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation," in *CVPR*, 2019, pp. 9768–9777.
- [54] T. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Dada: Depth-aware domain adaptation in semantic segmentation," in *ICCV*, 2019.
- [55] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *T-PAMI*, 2019.
- [56] V. Olsson, W. Tranheden, J. Pinto, and L. Svensson, "Classmix: Segmentation-based data augmentation for semi-supervised learning," in *WACV*, 2021.
- [57] Y. Liu, J. Deng, X. Gao, W. Li, and L. Duan, "Bapa-net: Boundary adaptation and prototype alignment for cross-domain semantic segmentation," in *ICCV*, 2021.
- [58] Q. Zhou, Z. Feng, Q. Gu, J. Pang, G. Cheng, X. Lu, J. Shi, and L. Ma, "Context-aware mixup for domain adaptive semantic segmentation," *TCSVT*, 2022.
- [59] L. Hoyer, D. Dai, and L. V. Gool, "Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation," in *CVPR*, 2022.
- [60] W. Kouw and M. Loog, "A review of domain adaptation without target labels," *T-PAMI*, 2021.
- [61] P. Morerio, J. Cavazza, and V. Murino, "Minimal-entropy correlation alignment for unsupervised deep domain adaptation," *arXiv*, 2017.
- [62] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.
- [63] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua, "Homm: Higher-order moment matching for unsupervised domain adaptation," in *AAAI*, 2020.
- [64] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *ECCV*, 2016.
- [65] T. Kim and C. Kim, "Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation," in *ECCV*, 2020.
- [66] S. Jadon, "A survey of loss functions for semantic segmentation," in *CIBCB*, 2020.
- [67] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.
- [68] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *ICCV*, 2019.
- [69] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *ACRL*, 2017.
- [70] C. Saltori, E. Krivosheev, S. Lathuilière, N. Sebe, F. Galasso, G. Fiameni, E. Ricci, and F. Poiesi, "Gipso: Geometrically informed propagation for online adaptation in 3d lidar segmentation," in *ECCV*, 2022.
- [71] M. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *ISVC*, 2016.
- [72] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu, "Pointcutmix: Regularization strategy for point cloud classification," *arXiv*, 2021.

- [73] A. Xiao, J. Huang, D. Guan, K. Cui, S. Lu, and L. Shao, "Polarmix: A general data augmentation technique for lidar point clouds," *NeurIPS*, 2022.



Cristiano Saltori is a PhD candidate in Computer Vision at the University of Trento, Italy. Before, he received his Information and Communication Engineering Master's degree from the University of Trento. His research interests include deep learning, 3D scene understanding, perception and domain adaptation.



Fabio Galasso heads the Perception and Intelligence Lab (PINlab) at the Dept. of Computer Science, Sapienza University of Rome (Italy). His research interests include distributed and multi-agent intelligent systems, perception (detection, recognition, re-identification, forecasting) and general intelligence (reasoning, meta-learning, domain adaptation), within sustainable and interpretable AI frameworks.



Giuseppe Fiameni is a Data Scientist at NVIDIA where he oversees the AI Technology Center in Italy, a collaboration among NVIDIA, CINI and CINECA to accelerate academic research in the field of Artificial Intelligence through collaboration projects. He has been working as HPC specialist and consultant at CINECA, the largest HPC facility in Italy, providing support for large-scale data analytics workloads.



Nicu Sebe is Professor with the University of Trento, Italy, leading the research in the areas of multimedia information retrieval and human behavior understanding. He was the General Co-Chair of ACM Multimedia 2013 and 2022, and the Program Chair of ACM Multimedia 2007 and 2011, ECCV 2016, ICCV 2017 and ICPR 2020. He is a fellow of the International Association for Pattern Recognition.



Fabio Poiesi is a Researcher in the Technologies of Vision (TeV) Lab at Fondazione Bruno Kessler in Trento, Italy. He received his Ph.D. degree from Queen Mary University of London (UK). He was a Postdoctoral Researcher in Queen Mary University of London before moving to TeV. His research interests are 3D scene understanding, object detection and tracking, and extended reality.



Elisa Ricci is an associate professor at the University of Trento and a head of research unit at Fondazione Bruno Kessler. She received the Honorable mention award at ICCV 2021 and the Best Paper Award at ACM MM 2021. Her research interests are mainly in the areas of computer vision and deep learning. She is an ELLIS fellow.