LiDAR-UDA: Self-ensembling Through Time for Unsupervised LiDAR Domain Adaptation

Amirreza Shaban* JoonHo Lee* Sanghun Jung* Xiangyun Meng Byron Boots
University of Washington

Abstract

We introduce LiDAR-UDA, a novel two-stage self-training-based Unsupervised Domain Adaptation (UDA) method for LiDAR segmentation. Existing self-training methods use a model trained on labeled source data to generate pseudo labels for target data and refine the predictions via fine-tuning the network on the pseudo labels. These methods suffer from domain shifts caused by different LiDAR sensor configurations in the source and target domains. We propose two techniques to reduce sensor discrepancy and improve pseudo label quality: 1) LiDAR beam subsampling, which simulates different LiDAR scanning patterns by randomly dropping beams; 2) cross-frame ensembling, which exploits temporal consistency of consecutive frames to generate more reliable pseudo labels. Our method is simple, generalizable, and does not incur any extra inference cost. We evaluate our method on several public LiDAR datasets and show that it outperforms the state-of-the-art methods by more than 3.9% *mIoU on average for all scenarios. Code will be available at* https://github.com/JHLee0513/lidar_uda.

1. Introduction

Modern approaches to perception for robotics and autonomous driving rely on supervised LiDAR segmentation methods that can accurately identify objects and scenes from 3D point clouds. These methods have advanced significantly thanks to large public datasets [2, 5, 35, 29] that enable the development of efficient and powerful deep neural networks [47, 8, 43], and they have inspired applications in other domains such as off-road navigation [25, 34], locomotion navigation [13], and construction site mapping [14]. However, supervised LiDAR segmentation often struggles to adapt to new domains (*i.e.,* domains that the model is not trained on) due to distributional shifts [32] between source and target datasets. LiDAR perception poses a unique challenge in this regard because different sensor configurations (*e.g.,* beam patterns, reflectivity estimates, mounting position, etc.) introduce significant distributional shifts [39]. To mitigate such concerns, several Unsupervised Domain Adaptation (UDA) approaches [21, 17, 44, 4] have been proposed, transferring the knowledge of a model trained on one domain to another without requiring additional labels. UDA plays an essential role in LiDAR segmentation since it relieves the necessity of an expensive and labor-intensive labeling process.

Domain adaptation methods based on *self-training*, which work by iteratively generating pseudo labels on target data and retraining the model with these labels, have achieved great success in reducing covariate shift in image-based semantic segmentation tasks [26, 48, 12, 1]. These self-training methods operate under the assumption that a model trained on source data yields mostly accurate predictions on at least a subset of the target dataset, enabling the model to adapt and refine its predictions iteratively through fine-tuning over the pseudo labels. However, in the case of LiDAR segmentation, the beam pattern gap between different LiDAR sensors hampers the source model from predicting reasonably good pseudo labels in the target domain for initializing the self-training approach.

To overcome this gap between the source and target datasets, we propose a simple yet effective structured point cloud subsampling method that simulates different LiDAR beam patterns. Specifically, we randomly subsample rows in the range image [27] of a high-beam LiDAR sensor to simulate low-beam LiDAR sensors. Additionally, we propose *cross-frame ensembling*, a temporal ensembling module, to ensure consistency of pseudo labels across LiDAR scans within each sequence. Cross-frame ensembling aggregates predictions from multiple scans and uses nearest neighbors to refine the pseudo labels. While we could simply calculate the average with uniform weights, this method ignores the temporal (i.e., time from the reference scan) and spatial variations (i.e., distance to sensor origin for each scan) of points captured by the LiDAR sensor when aggregating multiple scans. We address this issue by training a Learned Aggregation Model (LAM) that resembles graph convolution [38, 40]. LAM learns how to aggregate pseudo

*Equal Contribution.

labels within a sequence and weigh labels for each point differently according to its importance. Adopting this approach eliminates the need for ad-hoc approaches to deal with special cases such as moving objects [44, 21].

We show that the combination of proposed modules achieves state-of-the-art performance in domain adaptation scenarios for urban and off-road driving. Moreover, our framework is applicable to off-the-shelf LiDAR segmentation networks since it does not require any architectural modifications or impose additional computational costs during inference. In contrast to previous work [21, 44] that uses aggregated LiDAR scans within a sequence as a dense and sensor-agnostic representation for the segmentation network, our approach maintains the sparsity of the point cloud during the network forward pass. This characteristic enables us to use state-of-the-art network architectures that favor sparse convolutions for efficient LiDAR segmentation.

2. Related Work

LiDAR Semantic Segmentation LiDAR semantic segmentation is a fundamental capability for scene understanding in autonomous driving and robotics. In recent years, deep learning methods have achieved remarkable results on LiDAR segmentation, thanks to several large-scale datasets and benchmarks, such as nuScenes [5], SemanticKITTI [2], and SemanticPOSS [29]. Approaches to LiDAR segmentation can be broadly classified into point-based [27, 16], image-based [11, 41], sparse voxel-based [47], and hybrid [36] categories. Despite remarkable progress, existing methods still face challenges in generalizing to different datasets due to two factors: 1) different datasets have different semantic classes and geometric feature distributions, depending on the environments where they were collected. 2) LiDAR sensors have different mounting positions and produce different beam patterns. Therefore, models trained on one dataset may not perform well on another dataset [44, 17, 21]. This limitation restricts the practical applicability of LiDAR-based segmentation methods because labeling LiDAR points is costly and time-consuming.

Domain Adaptation for LiDAR There has been an increasing interest in developing domain adaptation techniques to improve the generalization ability of LiDAR perception models across different LiDAR sensors and environments. Domain adaptation methods for LiDAR can be grouped into three categories, which we describe here. 1) *Learning domain-invariant representation*. These methods transform the source and target domain point clouds into a common representation that is independent of sensor characteristics. The common representation can be a 3D mesh [21, 44] or a bird's eye view projection [31]. Notably, Complete & Label [44] learns to complete 3D surfaces from sparse LiDAR scans using a sensor-specific network, and then applies a segmentation network on the completed surfaces. However, this method requires a simplified segmentation network to handle the dense point clouds and also needs to remove moving objects from the common representation using heuristic methods [44] or manual annotations [21]. We use a data-driven approach to decide how different semantic classes should be aggregated. 2) *Learning domain-invariant features.* These methods align or adapt the feature representations of source and target domains using various techniques, such as feature alignment [41, 28], adversarial training [17], multi-task learning [33], and graph matching [4]. These methods do not modify the input point clouds but learn to extract features that are robust to domain variations. 3) *Domain transfer*. These methods explicitly model the difference between source and target domains and apply it to transfer one domain to another. For example, some methods learn a noise model from real data and add it to synthetic data to make it more realistic [5]. Our method applies LiDAR beam subsampling to reduce the domain gap without needing heuristics on which rows to drop and instead uses a random selection scheme.

Self-training We leverage self-training, a semi-supervised learning technique [20, 37] that has been successfully applied for unsupervised domain adaptation in the image domain [1, 49, 23], but has not been extensively explored for LiDAR domain adaptation. Self-training, also known as teacher-student training or self-ensembling, iteratively trains a model on a mixture of labeled source data and pseudo labeled target data, where the pseudo labels are generated by the model itself on unlabeled target data. However, since the pseudo labels may be noisy or inaccurate, self-training often requires some regularization strategies to improve their quality and reliability, such as class balancing [49], adversarial pre-training [42], and uncertainty estimation [45]. In our approach, we adopt a teacher-student paradigm and use a data-driven aggregation scheme that selectively aggregates the pseudo labels within each sequence as a regularizer. This further enhances the performance of our model on the target domain.

3. Method

3.1. Definitions and Framework Overview

We consider the problem of point cloud semantic segmentation in a domain adaptation setting. Let $\mathcal{S}$ and $\mathcal{T}$ denote the source and target datasets, respectively. Each element in the source dataset consists of a tuple $(\mathcal{P}, \mathbf{l})$, where $\mathcal{P} \in \mathbb{R}^{P\times 3}$ represents a 3D point cloud and $\mathbf{l} \in \{0,1\}^{P\times K}$ denotes the corresponding one hot semantic labels with K classes. In contrast to the source dataset, the target dataset only contains unlabeled point clouds. We address the closed-set adaptation problem [39], where both the source and target domains share the same semantic classes. Our objective is to train a model on the labeled source

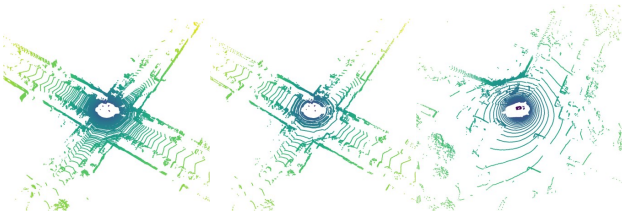

Figure 1. Comparison of a) original SemanticKITTI LiDAR scan with 64 LiDAR beams, b) row-subsampled by randomly dropping range image rows with probability 0.5, and c) nuScenes LiDAR scan with 32 beams. The figure shows that subsampling rows effectively simulates LiDAR scans with fewer laser beams.

dataset and unlabeled target point clouds, and then evaluate it on a held-out target set using ground-truth annotations. Our approach employs a two-stage self-ensembling strategy to learn a performant segmentation model for the target dataset. A LiDAR segmentation model $F_\theta : \mathbb{R}^{P\times 3} \rightarrow \mathbb{R}^{P\times K}$ is first trained on the labeled source dataset, and then adapted to the target dataset. The source model training and domain adaption stages are detailed next.

3.2. Source Model Training

We train the source model using standard supervised learning, which enables our framework to be applied to any generic LiDAR segmentation model. To facilitate generalization to target domains with fewer LiDAR beams than the source domain, we apply *structured point cloud subsampling* along with conventional data augmentations during training. Specifically, we subsample the point cloud on the *range image*, which is created by spherical mapping of a LiDAR scan into a 2D image [27]. The image is represented as $I_\mathcal{P} \in \mathbb{R}^{H\times W\times 3}$, where H and W are the height and width of the projected image. The mapping from the 3D point $\mathbf{p} = (x, y, z)$ to the image coordinate (u, v) is defined as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right] W \\ \left[1 - \left(\arcsin\left(z/||\mathbf{p}||_2\right) + f_{\text{down}}\right) f^{-1}\right] H \end{pmatrix},$$

where f_{down} and f denote lower and vertical LiDAR field-of-view, respectively. Then, we randomly drop all the points on a horizontal line with probability $1 - \min(1, r)$, where $r = n_{\text{target}}/n_{\text{source}}$ and $n_{\text{target}}, n_{\text{source}}$ are the number of laser beams in the target and source datasets, respectively. In cases where the target LiDAR has more beams than the source dataset, we do not apply subsampling to the source dataset. Instead, we address the domain gap by subsampling the target point cloud during the domain adaptation stage in Section 3.3.1. As demonstrated in Figure 1, row subsampling effectively simulates a LiDAR scan with fewer laser beam patterns. We further elaborate on the effectiveness of this data augmentation in Section 4.5.

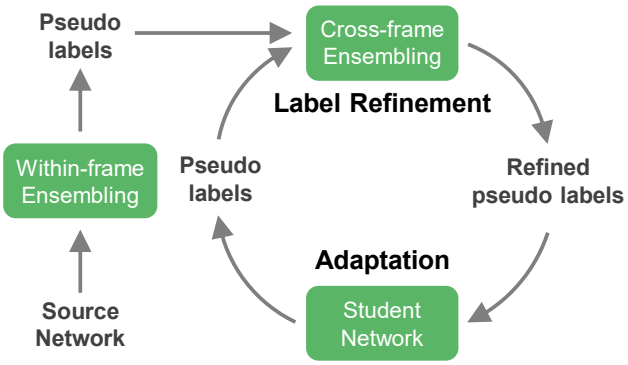

Figure 2. Overview of the domain adaptation process. We first apply within-frame ensembling with the source model (source network) $F_\theta(\cdot)$ to generate the pseudo labels. Subsequently, we apply cross-frame ensembling with the LAM module $g_\omega(\cdot)$ to refine the initially generated pseudo labels. Then, we adapt the student network to the target domain by training it with the refined pseudo labels for a certain number of epochs, and finally, re-generate the pseudo labels from the trained student network. The cross-frame ensembling and adaptation steps are iterated multiple times.

3.3. Target Domain Adaptation

The domain adaptation stage is an iterative process where each iteration involves generating pseudo labels using a teacher model (*i.e.,* label generation step), and training a student network with given pseudo labels (*i.e.,* training step). The adaptation allows for multiple iterations, where any additional iterations after the initial source-to-target adaptation may be viewed as further refinements within the target domain.

Figure 2 summarizes the domain adaptation stage. We employ within-frame and cross-frame ensembling techniques to enhance the quality of the pseudo labels and improve the training of the student model. In the initial source-to-target adaptation iteration, we employ within-frame subsampling (Section 3.3.1) to reduce high-beam target beams to match the low-beam source model. We further enhance the pseudo labels by aggregating predictions within each sequence using cross-frame ensembling described in Section 3.3.2.

The student network is randomly initialized and trained with aggressive data augmentation to enforce consistent predictions across different augmentations for domain adaptation, following a common practice in previous work [1]. While recent literature [1, 46] utilizes a momentum network as a teacher, we adopt a fixed teacher model that allows us to pre-compute the pseudo labels at the beginning of each domain adaptation iteration and reduce the training time significantly.

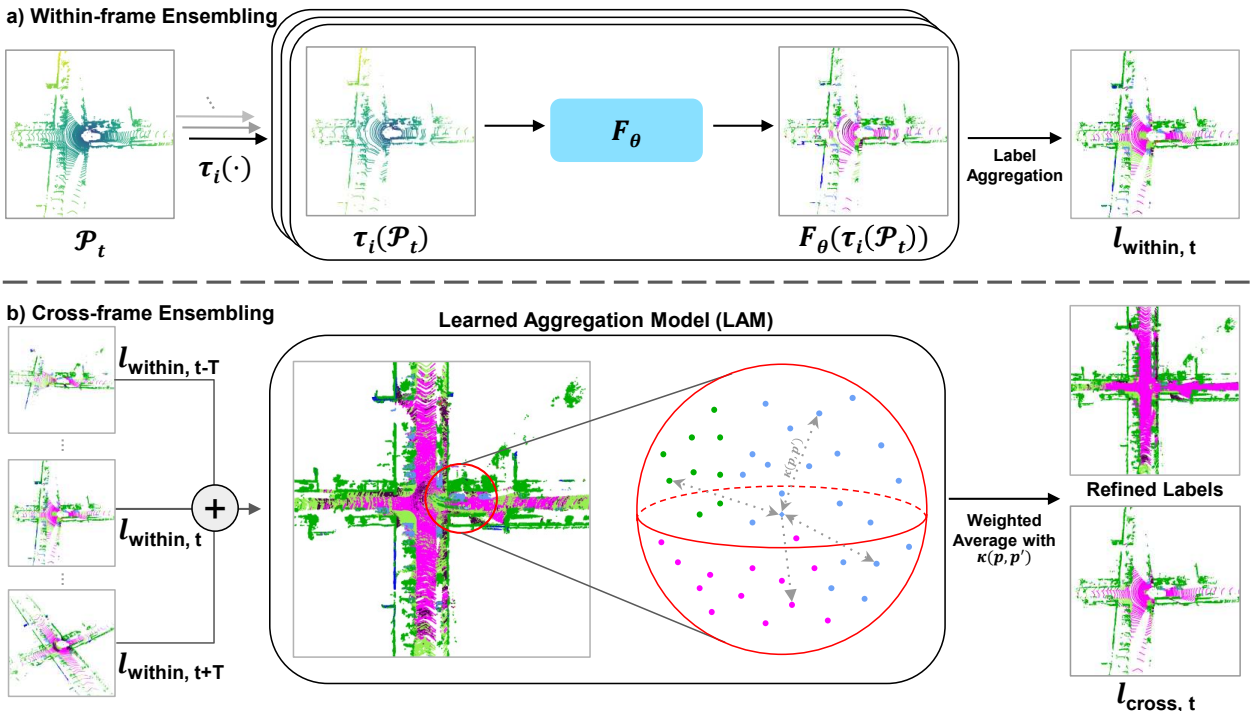

Figure 3. Illustration of our within-frame and cross-frame ensembling modules. All the predictions in the figure are obtained from our nuScenes to SemanticKITTI experiment. a) Within-frame ensembling: we randomly select horizontal rows with a probability of $1 - \min(1, 1/r)$ and drop all the points in the rows to simulate the different beam patterns of the target domain. To obtain more robust predictions, we apply this subsampling several times and average the prediction. b) Cross-frame ensembling: with the obtained predictions from step a), we temporally aggregate the point clouds and their predictions. Afterward, we calculate the nearest neighboring points within ϵ-ball and predict their summing weight using LAM. Finally, we obtain the refined pseudo labels by weight averaging the pseudo labels of the neighboring points.

3.3.1 Within-frame Ensembling

We use within-frame ensembling when we have more beams in the target LiDAR than in the source LiDAR. As shown in Figure 3-a, this approach works as follows: First, we create a batch of randomly subsampled point clouds from an input LiDAR scan. Then, we use the source model to predict labels for each subsampled point cloud. Finally, we average the predictions across all subsampled point clouds to get the final prediction.

Let $\mathcal{P}$ be an input point cloud. We generate a set of subsampled point clouds $\mathbb{T}(\mathcal{P}) = \{\tau_i(\mathcal{P})\}_{i=1}^{N_s}$, where N_s is the number of trials and $\tau(\cdot)$ is a subsampling operation. We set $\tau_1(\cdot)$ as an identity mapping to keep the original input point cloud and follow a similar approach as in Section 3.2, but we use the source-to-target ratio $(1/r)$ to drop points within a row of the LiDAR image with a probability of $1 - \min(1, 1/r)$. Then, we obtain a set of predictions from the pretrained network $F_{\boldsymbol{\theta}}(\cdot)$ by computing $\mathbb{P}(\mathcal{P}) = \{F_{\boldsymbol{\theta}}(\tau_i(\mathcal{P}))\}_{i=1}^{N_s}$. For each point in the original point cloud $\mathbf{p} \in \mathcal{P}$, we compute its final prediction by averaging all the predictions associated with $\mathbf{p}$ within the augmented point clouds $\mathbb{P}(\mathcal{P})$. As the original point cloud is always included, every point appears at least once during aggregation.

3.3.2 Cross-frame Ensembling

Our cross-frame ensembling is illustrated in Figure 3b. To further refine the pseudo labels of individual scans, we utilize predictions on scans from both previous and subsequent timestamps. Given an input query scan $\mathcal{P}_t$ at time t, we aggregate scans from the past and future into a dense point cloud $\mathcal{D} = \bigcup_{i=t-T}^{t+T} A_i(\mathcal{P}_i)$, where A_i represents the transformation from index i to t, and T controls the number of aggregated frames. Let $\mathcal{N}(\mathbf{p}) \subset \mathcal{D}$ denote the set of points that fall in the vicinity of $\mathbf{p} \in \mathcal{P}_t$. We compute the enhanced class probability vector $\tilde{\mathbf{v}}(\mathbf{p})$ as

$$\tilde{\mathbf{v}}(\mathbf{p}) = \frac{1}{Z(\mathbf{p})} \sum_{\mathbf{p}' \in \mathcal{N}(\mathbf{p})} \kappa(\mathbf{p}, \mathbf{p}') \mathbf{v}(\mathbf{p}'), \tag{1}$$

where $\mathbf{v}(\mathbf{p}')$ is the single scan pseudo label of $\mathbf{p}'$, $\kappa : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$ is a positive scoring function, and Z is a normalizer that ensures $\tilde{\mathbf{v}}(\mathbf{p})$ remains a probability vector,

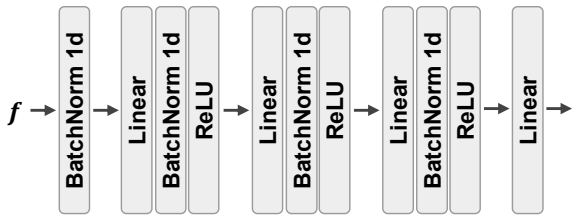

Figure 4. Architecture of $g_\omega(\cdot)$ our Learned Aggregation Model (LAM) module. Note that $\mathbf{f}$ denotes $\mathbf{f} = \mathbf{\Phi}(\mathbf{p}, \mathbf{p}')$. We first apply the batch normalization layer to effectively address the differences in statistics of source and target domains.

i.e.,

$$Z(\mathbf{p}) = \sum_{\mathbf{p}' \in \mathcal{N}(\mathbf{p})} \kappa(\mathbf{p}, \mathbf{p}'). \tag{2}$$

In our experiments, we compute $\mathcal{N}(\mathbf{p})$ by finding the k-nearest neighbors and then selecting the subset that lies within an ϵ-ball (i.e., a ball with radius ϵ) centered on $\mathbf{p}$. This approach guarantees that the point count remains under a specific threshold, and all points are within the ϵ-ball surrounding $\mathbf{p}$.

We can set $\kappa(\mathbf{p}, \mathbf{p}')$ to a constant value to obtain the standard KNN algorithm, but this method is more suitable for static objects. Assigning the same weight to all the points in $\mathcal{N}(\mathbf{p})$ overlooks their semantic classes, their distances from the query point $\mathbf{p}$, and the fact that these points are captured at different times and distances from the LiDAR sensor.

We further improve the quality of refined labels by learning an attention model for label aggregation, which we refer to as the Learned Aggregation Model (LAM). To account for the various sources of aggregation error, LAM considers not only the Euclidean distance between input points $\mathbf{p}$ and $\mathbf{p}'$, but also their single scan pseudo labels $\mathbf{v}(\mathbf{p})$ and $\mathbf{v}(\mathbf{p}')$, the temporal offset between $\mathbf{p}$ and $\mathbf{p}'$, and the distance of $\mathbf{p}'$ to its sensor origin. For ease of notation, we use $\Phi(\mathbf{p}, \mathbf{p}') \in \mathbb{R}^D$ to denote the feature vector that concatenates each of these factors. Then, we use a fully connected network $g_\omega : \mathbb{R}^D \to \mathbb{R}$ to predict an attention score for each feature vector. Finally, enhanced pseudo labels are obtained via attention by setting $\kappa(\mathbf{p}, \mathbf{p}') = \exp(g_\omega(\Phi(\mathbf{p}, \mathbf{p}')))$ in Equation (1).

To train LAM, we use the source model predictions and aforementioned features as inputs, while supervision is provided through the source dataset ground-truth labels. We first compute the single-scan pseudo label $\mathbf{v}(\mathbf{p})$ for each point within the source dataset using the source model network $F_\theta(\cdot)$. Next, we construct the dense point cloud by aggregating scans and pre-compute the nearest neighbors $\mathcal{N}(\mathbf{p})$ for all the points to speed up the training. During training, we estimate the enhanced labels using Equation (1) and use a combination of multi-class cross-entropy and Lovász-Softmax loss [3] to update model parameters ω.

The LAM model g_ω, shown in Figure 4, consists of an input standardization layer and 3 fully connected hidden layers each followed by batch-norm and ReLU layers. The collected statistics (i.e. mean and variance) for the standardization layer captured by LAM are fit to the source domain and hence are not optimal for adaptation. Therefore, we *modulate* the statistics by updating the layer with statistics acquired from the target domain. Since the statistics are collected from input point features including semantic pseudo labels, it does not involve acquiring any privileged information from the target domain. We find that updating the statistics in the first layer serves a similar purpose to the Adaptive Batch Normalization (ABN) domain adaption method [24] that adapts the batch-norm statistics across the network.

4. Experiments

4.1. Experimental Setup

We compare our method against prior domain adaptation methods on publicly available LiDAR segmentation datasets using the mean Intersection over Union (mIoU) metric. In particular, our main experiment is split into two tracks: 1) between SemanticKITTI [2] and nuScenes [5]; 2) between SemanticKITTI [2], SemanticPOSS [29], and SemanticUSL [17]. These tracks demonstrate that our method is superior to prior work in the presence of environmental shifts, sensor configuration shifts, and different sets of semantic classes.

4.2. Implementation Details

As illustrated in Figure 4, the LAM architecture has three fully-connected layers which consist of 32, 64, and 128 channels, respectively. We train LAM on the same train/validation split as the source model. However, for nuScenes [5] totaling around 400K sweeps, in comparison to SemanticKITTI [2] (∼20K), SemanticUSL [17](∼18K), and SemanticPOSS [29](∼3K), running cross-frame ensembling on the entire set of sequences is costly in terms of both memory and storage. Therefore, when using the nuScenes dataset to train LAM, we subsample the training data while validation is kept unmodified. We randomly select 210 sequences from the nuScenes training set, out of the total 700 sequences. The list of these 210 sequences will be included in the future release of the code for reproducibility purposes.

We use the Pytorch [30] library for our training code, and we implement the sparse 3D convolutions with the Spconv [10] library. Our cross-frame ensembling aggregates 60 frames with a stride of 3 for efficient but dense coverage of the scene. The student model is trained for 25 epochs using the Adam [19] optimizer with a learning rate of $1e{-}3$ and with other optimization hyperparameters set to default.

We pre-compute the neighbors $\mathcal{N}(\mathbf{p})$ as well as the model predictions for label generation, which significantly reduces the pre-processing time during the student model training. We use the Faiss [18] library to run a nearest neighbor search to find the 60 nearest neighbors for each query point. We employ an $\epsilon = 0.2m$ radius filtering of found nearest neighbors exclusively in our SemanticKITTI and nuScenes experiments since such radius filtering does not yield any benefits in other experiments. We maintain a fixed size set for all points with zero padding.

We train the source model with 4 types of data augmentation. We apply 1) random rotation around the z-axis with an angle sampled from $[-45°, 45°]$, 2) flip augmentation by randomly flipping x and/or y coordinates, 3) random scaling with a value sampled from $[0.95, 1.05]$, and 4) random translation with zero-mean Gaussian noise and a standard deviation of 0.1. We refer to this setting as the *basic augmentation* scheme.

As shown in the ablation study, the student model with self-ensembles performs better when trained with a more aggressive, *intense augmentation* scheme. In this setting, we use the same set of augmentations but increase the range of values. Specifically, we increase the random scale to $[0.9, 1.1]$ and set the standard deviation for random translation to 0.5.

4.3. Comparisons on SemanticKITTI and nuScenes

SemanticKITTI and nuScenes datasets focus on urban driving, and hence their semantic labels are specific to the urban roads, including but not limited to road surface, sidewalk, pedestrian, car, etc. For this experiment, we adopt the results from four prior works as the baseline. The baseline methods and our method all use the MinkowskiNet [9] architecture to make fair comparisons.

The SemanticKITTI and nuScenes datasets present challenges from significant sensor configuration shifts, collected with a 64-beam HDL-64E and with a 32-beam VLP-32, respectively. Additionally, the sensor from nuScenes is facing the right side of the vehicle, while the sensor from SemanticKITTI is facing forward, resulting in a $-90°$ rotation difference between them around the z-axis. From minor differences such as axis rotation, sensor height, and viewpoint angle, to major variances such as beam pattern and resolution, the domain gap in sensor configuration makes these adaptation scenarios highly challenging.

As shown in Table 1, the source models without any DA experience severe degradation in target domain performance. In stark contrast, our method improves over the source model by 14.1% mIoU and 10.9% mIoU in each scenario. We also compare our method with two state-of-the-art methods Complete & Label [44], which uses aggregated LiDAR scans as a dense representation, and Graph Matching [4], which uses graph-based feature extraction to align local features across domains. Our method surpasses both methods by a large margin. As we elaborate in Section 4.5, our success stems from our structural point cloud subsampling augmentation, cross-frame ensembling, and LAM.

Source	Target	Method	mIoU (%) ↑
KITTI	KITTI	Source	45.80
	NUS	Source	27.75
		SqueezeSegV2* [41]	10.10
		SWD* [22]	27.70
		Complete & Label [44]	31.60
		Graph Matching [4]	37.30
		LiDAR-UDA	**41.84**
NUS	NUS	Source	50.72
	KITTI	Source	23.17
		SqueezeSegV2* [41]	13.40
		SWD* [22]	24.50
		Complete & Label [44]	33.70
		LiDAR-UDA	**34.04**

Table 1. Comparison of methods for SemanticKITTI (KITTI) and nuScenes (NUS) datasets. MinkowskiNet [9] architecture is adopted for all methods. * Results from [44]

4.4. Comparisons on SemanticKITTI, SemanticPOSS, and SemanticUSL

We additionally evaluate our method on adaptation scenarios using SemanticKITTI and SemanticPOSS as the source domains, and SemanticKITTI, SemanticPOSS, and SemanticUSL as the target domains. In contrast to the semanticKITTI/nuScenes scenarios, the corresponding three datasets have relatively similar sensor configurations. Instead, the primary domain gap lies in shifts caused by the different environments since SemanticKITTI is collected strictly from on-road scenarios while SemanticPOSS is collected on the campus area, and SemanticUSL is collected on both campus and off-road testing sites.

We compare our method to LiDARNet [17], the current state-of-the-art in UDA for LiDAR segmentation across the SemanticKITTI, SemanticUSL, and SemanticPOSS datasets. LiDARNet adopts the SalsaNext [11] architecture backbone and employs adversarial training for adaptation. The SalsaNext backbone utilizes LiDAR intensities in conjunction with the 3D point cloud. However, we have found that the LiDAR intensity domain gap adversely affects the quality of the pseudo labels generated for the target dataset by our source model. Consequently, we choose not to use LiDAR intensities when training the source model and generating pseudo labels for the target set during the first iteration of adaptation. However, we do employ LiDAR intensities in subsequent adaptation steps when training the student model and generating pseudo labels.

As shown in Table 2, our method outperforms LiDARNet by 3.2% mIoU and 3.9% mIoU in the SemanticKITTI to Semantic USL and SemanticPOSS domain adaptation tasks, respectively. When using SemanticPOSS as the

Source	Target	Method	Person	Rider	Car	Trunk	Vegetation	Sign	Pole	Object	Building	Fence	Bike	Ground	mIoU
KITTI	KITTI	Source (LiDARNet)	62.09	74.21	93.59	61.15	91.11	37.99	57.94	50.36	84.82	54.64	15.48	94.13	64.79
		Source (Ours)	42.06	69.11	94.89	60.53	85.58	31.86	59.00	39.94	88.50	47.58	9.49	94.34	60.24
	USL	Source	33.90	0.00	27.45	10.68	36.89	16.20	12.72	5.68	41.61	3.55	**31.60**	75.95	24.69
		CyCADA	0.38	0.00	28.70	13.83	57.11	20.70	23.83	3.78	53.14	22.30	9.24	72.36	25.45
		LidarNet	33.17	0.00	67.75	38.95	**85.60**	49.94	43.44	8.94	**72.86**	**44.06**	23.07	**93.18**	46.75
		Source	42.00	0.00	69.18	35.08	82.94	6.80	43.41	13.23	68.02	42.75	2.13	92.45	41.51
		LiDAR-UDA	**49.50**	0.00	**78.42**	**53.78**	85.47	**58.56**	**59.97**	**19.42**	69.70	32.16	0.00	92.70	**49.97**
	POSS	Source	22.77	1.78	35.91	16.86	39.84	7.08	9.73	0.18	57.03	1.64	**18.17**	41.99	21.08
		CyCADA	0.00	0.00	0.00	1.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12
		LidarNet	31.39	**23.98**	**70.78**	21.43	60.68	**9.59**	17.48	**4.97**	**79.53**	12.57	0.78	**82.41**	34.63
		Source	31.76	9.07	46.81	22.69	60.61	0.05	26.51	2.51	70.87	23.3	1.05	75.06	30.86
		LiDAR-UDA	**65.59**	2.19	64.12	**27.49**	**65.40**	6.44	**36.57**	4.19	75.21	**40.31**	0.00	75.06	**38.55**
POSS	POSS	Source (LiDARNet)	64.47	48.25	85.77	29.71	62.71	27.29	38.19	8.07	84.90	48.50	65.56	72.56	53.00
		Source (Ours)	73.65	34.18	69.48	27.58	71.11	28.24	24.43	13.90	79.71	44.11	47.73	78.93	49.42
	KITTI	Source	5.20	0.50	22.57	0.54	44.00	1.90	12.83	0.08	43.09	0.70	0.40	5.62	11.45
		LiDARNet	**23.64**	24.86	71.31	23.67	72.38	4.17	31.28	**2.48**	59.41	0.36	0.53	68.68	32.06
		Source	14.17	**48.21**	63.57	18.93	65.43	1.63	9.47	0.16	55.75	1.06	0.52	84.07	30.25
		LiDAR-UDA	11.65	39.1	**83.17**	**27.46**	**76.95**	**8.60**	**33.14**	0.28	**66.43**	**1.73**	**1.82**	**92.95**	**37.77**
	USL	Source	2.45	0.00	16.15	1.21	27.94	1.34	4.52	0.62	44.37	0.12	1.16	8.05	8.99
		LiDARNet	30.38	0.00	45.73	28.69	63.08	**22.29**	**33.92**	4.12	63.70	1.89	9.42	77.49	31.73
		Source	35.07	0.00	51.59	27.51	75.84	13.36	25.98	0.05	65.47	1.10	**11.21**	90.95	33.18
		LiDAR-UDA	**45.49**	0.00	**61.74**	**31.37**	**82.67**	15.30	25.19	**15.87**	**68.10**	**8.20**	8.19	**92.30**	**37.87**

Table 2. Comparison of methods for SemanticKitti (KITTI), SemanticPOSS (POSS), and SemanticUSL (USL) datasets. MinkowskiNet [9] architecture is adopted for LiDAR-UDA, while CyCADA [15] and LiDARNet adopted the LiDARNet architecture [17], a boundary-aware variant of SalsaNext [11].

source domain, our method outperforms LiDARNet by 5.7% mIoU on SemanticKITTI and 6.1% mIoU on SemanticUSL.

The aforementioned intensity domain gap becomes apparent when comparing the LiDARNet [17] model, trained with intensity values as input, with our *Source (Ours)* model, trained using the MinkowskiNet architecture without intensity. Despite the SalsaNext model outperforming our MinkowskiNet architecture on the source domains, our source model demonstrates significantly better performance on the target domains, revealing a domain gap in the LiDAR intensities. Further details can be found in the appendix.

4.5. Ablation Study

In this section, we analyze the individual contributions made by our proposed modules, namely the structural point cloud subsampling, within-frame / cross-frame ensembling, and LAM. Additionally, we explore the applicability of LiDAR-UDA to other point cloud segmentation architectures and self-training strategies.

Structural Point Cloud Subsampling In the upper section of Table 3, we compare the nuScenes pseudo labels obtained from source models trained on SemanticKITTI using various subsampling methods. With a target-to-source ratio of $r = 0.5$, our method (Random) drops each row in the range image with a 50% chance. This simulates diverse LiDAR patterns and significantly reduces the domain gap, resulting in an 8.9% improvement in mIoU compared to when no subsampling is applied. On the other hand, regular subsampling, which drops every other row in the LiDAR image, is not as robust to the variability of LiDAR patterns (second row).

Cross-frame Ensembling We also compare the impact of cross-frame ensembling using constant weights (Uniform) and LAM in the lower section of Table 3. The table shows that uniform aggregation boosts the performance by 3.3% in mIoU, and LAM further enhances it by 2.2% over the uniform method. Therefore, Table 3 demonstrates that both point cloud subsampling and attention-based cross-frame aggregation are essential for improving the adaptation performance without making specific assumptions on the sensor shift.

To further demonstrate the advantage of LAM, we show the confusion matrices of Uniform and LAM in Figure 5, where we compare their performance on static (road, terrain, trunk, etc.) and dynamic (car, pedestrian, bicycle, etc.) semantic label classes. We notice that both models perform similarly on static objects, but LAM has significantly fewer false negatives on dynamic objects (*i.e.,* dynamic objects predicted as one of the static classes) than Uniform. This could be explained by the fact that static objects have a higher density in the aggregated point cloud, and uniform weight assignment is potentially biased towards the objects with higher density.

Within-frame Ensembling We examine the effect of using the original point cloud and different numbers of subsampled point clouds in within-frame ensembles in Table 4. We observe consistent improvement using a larger number of ensembles, which effectively bridges the gap between different beam patterns. Additionally, comparing the sec-

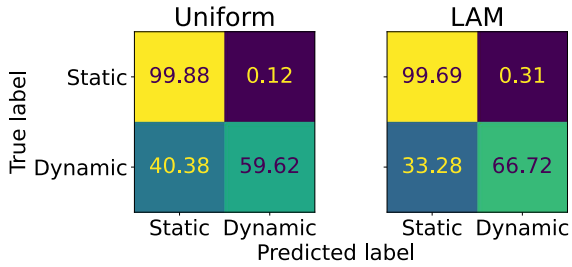

Figure 5. Normalized confusion matrices for static and dynamic classes in SemanticKITTI → nuScenes DA experiment. The original row-normalized 10x10 histogram matrix is condensed into a 2x2 matrix by grouping the static and dynamic classes. The results show that LAM outperforms standard uniform weights (Uniform) in predicting dynamic objects.

Method	Source Subsampling	Cross-frame Ensembling	mIoU (%) ↑
1	✗	✗	27.75
2	Regular	✗	34.97
3	Random	✗	36.64
4	Random	Uniform	39.96
5	Random	LAM	**42.10**

Table 3. Ablation study for SemanticKITTI → nuScenes adaptation. Note that cross-frame ensembling methods in the second part report the **teacher model** performance on the target domain dataset, and not the student model trained using the teacher as ground truth, which are the final results shown on Table 1.

Method	mIoU (%) ↑
input	23.17
2 x random	23.39
input + 2 x random	25.50
input + 4 x random	26.13
input + 8 x random	26.59
input + 16 x random	26.84

Table 4. Comparison of the within-frame ensembling on the target SemanticKITTI domain for the source model trained on the nuScenes dataset. No adaptation is applied to the model for this comparison. We use MinkowskiNet [9] architecture for all methods.

	Method	mIoU (%) ↑
Effects of LAM	Single-scan + Intense Aug.	37.52
	LAM + Intense Aug. (Ours in Table 1)	41.84
Effects of Augmentation	Single-scan + Basic Aug.	37.36
	Single-scan + Intense Aug.	37.52
	LAM + Basic Aug.	40.61
	LAM + Intense Aug. (Ours in Table 1)	41.84

Table 5. Comparision of student models for SemanticKITTI → nuScenes DA. The single-scan method trains the student model directly from source model pseudo labels without any ensembling. The source model used by all the methods in the table is trained with structural point cloud subsampling.

ond and third rows reveals the importance of having the original point cloud in the ensemble. We used two random samples alongside the original point cloud in Section 4.3 to balance time complexity and performance.

Source	Target	Method	mIoU (%) ↑
KITTI	KITTI	Source	61.62
	NUS	Source	32.72
		LiDAR-UDA	**48.79**
NUS	NUS	Source	74.70
	KITTI	Source	32.05
		LiDAR-UDA	**46.58**

Table 6. Comparison of methods for SemanticKITTI (KITTI) and nuScenes (NUS) datasets. Cylinder3D [47] architecture is adopted for all methods.

Effects of LAM & Data Augmentation on Student Table 5 compares the student models trained with 1) single scan pseudo labels, i.e., directly adapting to the target domain model with pseudo labels from the source model, 2) basic data augmentation scheme with LAM, and 3) our intense augmentation with LAM.

Using LAM yields significant gains over using the single scan pseudo labels with an improvement of 4.3% mIoU. Meanwhile, the performance gains of 1.2% mIoU also align with the general understanding that applying stronger augmentation on the student model in a self-training or semi-supervised training framework is beneficial [1, 6, 7]. Lastly, we also observe that the basic data augmentation scheme reduces performance in the single scan pseudo label scenario.

Applicability to Other Architectures To demonstrate the model-agnostic nature of our proposed framework, we test integrating Cylinder3D [47] into LiDAR-UDA. Cylinder3D utilizes asymmetric cylindrical 3D convolutions, resulting in superior performance compared to MinkowskiNet [9]. While maintaining the experimental setup outlined in Section 4.3, we opt to use constant weights (Uniform) for this experiment, avoiding the time taken to train LAM. The results in Table 6 demonstrate significant improvements for LiDAR-UDA over the source model: 16.1% mIoU improvement for SemanticKITTI → nuScenes and 14.5% mIoU improvement for nuScenes → SemanticKITTI. Furthermore, when compared to the results in Table 1, using Cylinder3D yields significant improvements, including over our own method using MinkowskiNet.

LiDAR-UDA with Class-Balanced Self-Training While our method achieves state-of-art performance in the SemanticKITTI to SemanticPOSS in the adaptation experiment, we observe a reduction of rider and bike classes IoU. We hypothesize that this drop in performance is due to class imbalance, as the rider and bike classes account for only approximately 0.5% and 5% of the entire dataset, respectively. To test this hypothesis, we employ CBST [48], a class-balanced self-training framework that avoids the dominance of large classes in pseudo label generation by performing class-wise confidence normalization and selecting a portion of pseudo labels with higher confidence.

Table 7 demonstrates that using CBST with LAM im-

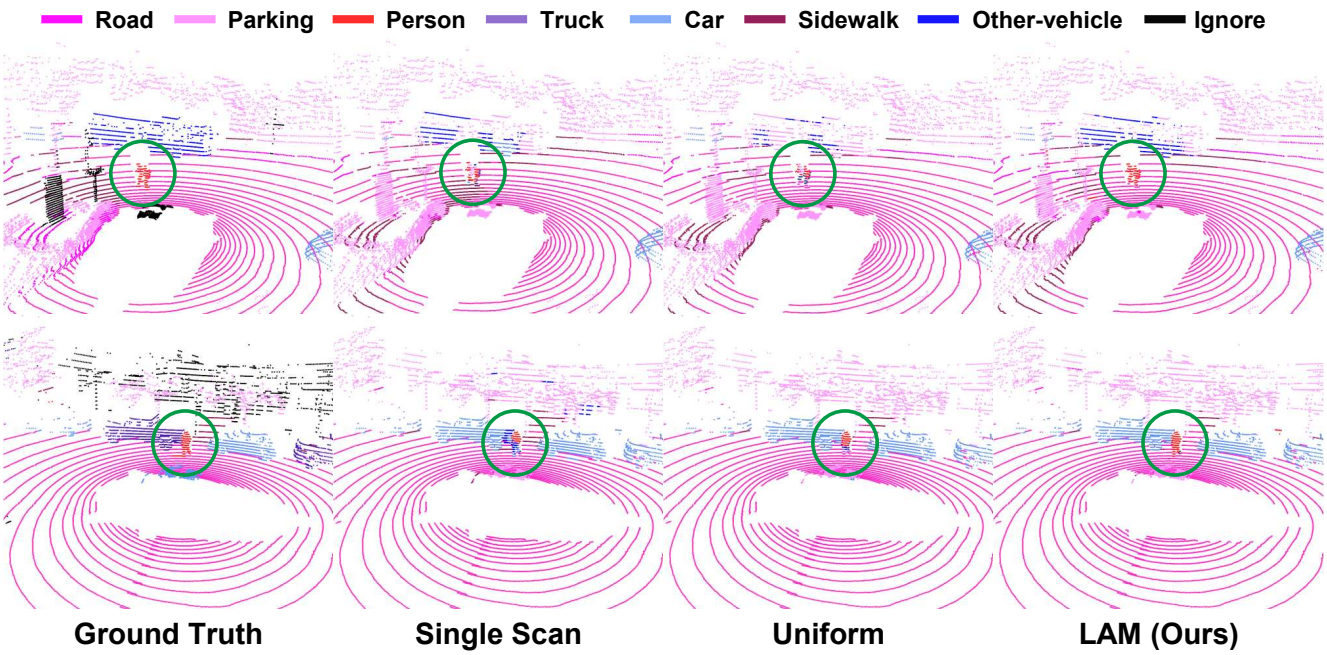

Figure 6. Visualization of two example frames from the held-out target domain data for SemanticKitti → nuScenes adaptation scenario. We compare the ground truth against pseudo labels from the base model (single scan), the cross-frame ensembling using uniform weights, and LAM. We circle the specific points of interest, where we see a noticeable improvement in segmenting small objects or sparse parts of a scene with LAM compared to other methods. Note that the unlabeled points are colored in black in the ground truth.

proves the IoU for minor classes, such as rider, traffic sign, pole, and bike. However, this improvement also results in a slight decrease in the overall mIoU. The optimal performance is achieved by setting the pseudo label portion to be $p = 20\%$. Please refer to Algorithm 2 of CBST [48] for further details on the implemented CBST framework.

Our ensembling and LiDAR augmentation techniques are applicable to various self-training strategies. Thus, exploring different self-training and class-balancing approaches offers promising avenues for future research in LiDAR unsupervised domain adaptation.

Classes	Source	LiDAR-UDA	+ CBST ($p = 0.2$)	+ CBST ($p = 0.5$)
Person	31.76	**65.59**	40.01	29.28
Rider	9.07	2.19	**19.89**	11.72
Car	46.81	**64.12**	51.55	60.84
Trunk	22.69	**27.49**	25.73	27.41
Vegetation	60.61	**65.40**	59.77	63.75
Traffic-sign	0.05	6.44	**21.07**	4.47
Pole	26.51	36.57	37.31	**39.22**
Object	2.51	**4.19**	1.31	1.95
Building	70.87	**75.21**	64.58	72.90
Fence	23.30	**40.31**	28.17	31.50
Bike	1.05	0.00	**10.94**	0.27
Ground	75.06	75.06	75.73	**77.34**
mIoU	30.86	**38.55**	36.34	35.05

Table 7. Comparison of source, Lidar-UDA, and Lidar-UDA + CBST methods on the SemanticKITTI to SemanticPOSS adaptation scenario. CBST denotes class-balanced self-training from Zou *et al.* [48].

4.6. Qualitative Evaluation

Figure 6 shows the effectiveness of our method with a set of examples comparing the base model prediction, uniform weight aggregation, and our method with LAM. We circle the areas of noticeable improvement, where we are able to observe that LAM correctly aggregates model predictions with regard to the geometric and temporal information of the points to segment pedestrians, which are either missed by the base model or washed out when the predicted semantic classes are uniformly aggregated.

5. Conclusion

We present LiDAR-UDA, a novel unsupervised domain adaptation framework for LiDAR segmentation. Based on self-training, the framework enables the transfer of model knowledge from the labeled source domain to the unlabeled target domain. Using structural LiDAR point cloud subsampling that reduces the geometric structural gap between the source and target domain input, and cross-frame ensembling that regularizes the self-training, LiDAR-UDA offers an efficient, model-agnostic adaptation method. We demonstrate the effectiveness of our method by surpassing the current state-of-the-art UDA methods on various publicly available LiDAR segmentation datasets. We hope this paper lays a foundation for further exploration of self-training methods for domain adaptation in LiDAR perception.

References

[1] Nikita Araslanov and Stefan Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15384–15394, 2021. 1, 2, 3, 8

[2] 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 *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 5

[3] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018. 5

[4] Yikai Bian, Le Hui, Jianjun Qian, and Jin Xie. Unsupervised domain adaptation for point cloud semantic segmentation via graph matching. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9899–9904. IEEE, 2022. 1, 2, 6

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 5

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 8

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 8

[8] Ran Cheng, Ryan Razani, Ehsan Moeen Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12542–12551, 2021. 1

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 6, 7, 8

[10] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022. 5

[11] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving, 2020. 2, 6, 7

[12] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*, 2018. 1

[13] Jonas Frey, David Hoeller, Shehryar Khattak, and Marco Hutter. Locomotion policy guided traversability learning using volumetric representations of complex environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2022. 1

[14] Tianrui Guan, Zhenpeng He, Ruitao Song, Dinesh Manocha, and Liangjun Zhang. Tns: Terrain traversability mapping and navigation system for autonomous excavators. *arXiv preprint arXiv:2109.06250*, 2021. 1

[15] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018. 7

[16] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020. 2

[17] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation, 2020. 1, 2, 5, 6, 7

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 6

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[20] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 2

[21] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270. IEEE, 2020. 1, 2

[22] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10285–10295, 2019. 6

[23] Ruihuang Li, Shuai Li, Chenhang He, Yabin Zhang, Xu Jia, and Lei Zhang. Class-balanced pixel-level self-labeling for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11593–11603, 2022. 2

[24] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *International Conference on Learning Representations*, 2017. 5

[25] Daniel Maturana, Po wei Chou, Masashi Uenoyama, and Sebastian A. Scherer. Real-time semantic mapping for autonomous off-road navigation. In *International Symposium on Field and Service Robotics*, 2017. 1

[26] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *European Conference on Computer Vision*, 2020. 1

[27] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4213–4220. IEEE, 2019. 1, 2, 3

[28] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv preprint arXiv:1711.10288*, 2017. 2

[29] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. Semanticposs: A point cloud dataset with large quantity of dynamic instances, 2020. 1, 2, 5

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[31] Florian Piewak, Peter Pinggera, and Marius Zöllner. Analyzing the cross-sensor portability of neural network architectures for lidar-based semantic labeling. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3419–3426. IEEE, 2019. 2

[32] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008. 1

[33] Christoph B Rist, Markus Enzweiler, and Dariu M Gavrila. Cross-sensor deep domain adaptation for lidar detection and segmentation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1535–1542. IEEE, 2019. 2

[34] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. Semantic terrain classification for off-road autonomous driving. In *5th Annual Conference on Robot Learning*, 2021. 1

[35] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[36] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pages 685–702. Springer, 2020. 2

[37] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017. 2

[38] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1

[39] Larissa T. Triess, Mariella Dreissig, Christoph B. Rist, and J. Marius Zöllner. A Survey on Deep Domain Adaptation for LiDAR Perception. In *Proc. IEEE Intelligent Vehicles Symposium (IV) Workshops*, 2021. 1, 2

[40] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2589–2597, 2018. 1

[41] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019. 2, 6

[42] Haifeng Xia, Handong Zhao, and Zhengming Ding. Adaptive adversarial network for source-free domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9010–9019, 2021. 2

[43] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European Conference on Computer Vision*, pages 677–695. Springer, 2022. 1

[44] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15363–15373, June 2021. 1, 2, 6

[45] Zhedong Zheng and Yi Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *International Journal of Computer Vision*, 129(4):1106–1120, 2021. 2

[46] Qianyu Zhou, Chuyu Zhuang, Xuequan Lu, and Lizhuang Ma. Domain adaptive semantic segmentation via regional contrastive consistency regularization. *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2021. 3

[47] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *arXiv preprint arXiv:2011.10033*, 2020. 1, 2, 8

[48] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *European Conference on Computer Vision*, 2018. 1, 8, 9

[49] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991, 2019. 2