University of Zurich[UZH]

PRS Photogrammetry Remote Sensing

VISUALIZATIONANDMULTIMEDIALAB

# Unsupervised Domain Adaptation for Synthetic-to-real 3D Point Cloud Semantic Segmentation in Outdoor Urban Scenes

Master's Thesis


Yuanzhou Cai

Department of Informatics

20-752-051



**A**dvisor: Binbin Xiang (ETH Zurich)

**S**upervisors: Prof. Dr. Renato Pajarola

Prof. Dr. Konrad Schindler (ETH Zurich)


October 6, 2023

# Abstract

The application of deep learning methods in 3D semantic segmentation has gained significant attention, driven by advancements in 3D data acquisition technologies and their decreasing costs. However, a major bottleneck in this domain is the labor-intensive and costly process of manually labeling vast quantities of 3D data, especially when dealing with voluminous outdoor LiDAR data. One approach to address this challenge is to train deep learning models on synthetic data and subsequently apply them to real-world scenarios. However, However, this approach can result in subpar performance due to the domain gap between synthetic and real-world data. In this thesis, we address this challenge by developing an Unsupervised Domain Adaptation (UDA) pipeline for 3D semantic segmentation in outdoor urban scenes. We propose a 3-stage pipeline that incorporates contrastive learning, transfer learning, and self-training. Within this pipeline, we employ a teacher-student training scheme, focusing on calibrating the teacher model to enhance the quality of pseudo-labels. This calibration process improves both the model's robustness and performance. Our experiments, conducted on UDA from SynLiDAR to SemanticKITTI and SemanticPOSS, demonstrate that our proposed method is on par with the state-of-the-art. Notably, when intensity values of LiDAR data are used as input, which introduces domain discrepancies, our method exhibits better robustness compared to other methods. Moreover, the results reveal that the contrastive learning stage not only aids in learning domain-invariant features but also in acquiring features that are discriminative for specific semantic classes. Lastly, we acknowledge certain limitations, such as the lack of interpretability.

# Acknowledgements

I would like to express my heartfelt gratitude to the following individuals:

- Prof. Dr. Konrad Schindler for this great thesis opportunity and for the comfortable working environment.
- Prof. Dr. Renato Pajarola for being my responsible supervisor at IFI.
- Binbin Xiang for her exceptional guidance and valuable input throughout this journey.
- Yujia Liu for her valuable advice, and the enjoyable lunches we shared.
- My friends, who shared many cozy evenings with me during this thesis period.
- My family, especially my grandparents, for being the best grandparents one could ever ask for.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, we commence by presenting the intricate challenge posed by unsupervised domain adaptation (UDA) in the context of 3D semantic segmentation. Section 1.1 outlines our motivations for undertaking this challenge. Following that, in Section 1.2, we provide a formulated problem description. In Section 1.3, we offer a preliminary glimpse into our specific research focus and the contributions we bring to this domain. Finally, Section 1.4 provides an overview of the thesis's structure, guiding you through the subsequent chapters and their content.

## 1.1 Motivation

In recent years, there has been a growing interest in the domain of computer vision concerning three-dimensional (3D) data [9]. This heightened attention can be attributed to several factors, including the rich spatial geometric information it offers, enabling machines to better perceive their surrounding objects and scenes when compared to traditional two-dimensional (2D) images [10]. Additionally, the rapid advancements in 3D data acquisition technologies, such as Light Detection and Ranging (LiDAR) and RGB-D cameras, have substantially reduced the cost associated with acquiring 3D data [11]. As we observe the progression of autonomous driving technologies, it becomes evident that 3D semantic segmentation holds a pivotal and intricate role within the perception pipeline. This process involves categorizing individual points within LiDAR scans into specific semantic classes, thereby providing crucial semantic information for real-time decision-making and ensuring public safety [12]. Deep learning methods have proven their capabilities in large-scale 3D point cloud semantic segmentation. However, they are facing several challenges.

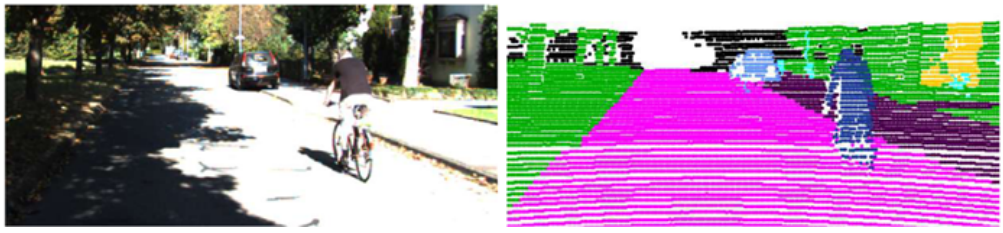

Figure 1.1: Illustration of semantic segmentation for point cloud. [1]

The effective performance of deep learning models in downstream tasks, such as semantic segmentation, necessitates the availability of extensive annotated datasets for training [13, 14]. However, the annotation of 3D data is time-consuming and laborious [13, 15, 14]. To mitigate the scarcity of annotated data, some existing approaches resort to the generation of synthetic 3D point cloud data and automatic annotations

through the use of simulators capable of emulating LiDAR in virtual outdoor environments [15, 16, 17, 18]. Synthetic data can be generated in large quantities and comes with readily available ground truth annotations. However, the real data collected by static or mobile LiDARs often exhibit irregular patterns and inherent noise, including missing points, specular reflections, grazing incidence angles, and other types of noise that are absent in synthetic data. While deep learning models have demonstrated profound capabilities in specific contexts, they can sometimes face challenges in generalization, particularly in the face of domain shifts [19, 14]. This means that their performance might degrade when confronted with tasks or domains that are dissimilar to the ones they were trained on. This limitation arises because deep learning models tend to learn representations that are fine-tuned to the training data but lack transferability and robustness when applied to unseen domains.

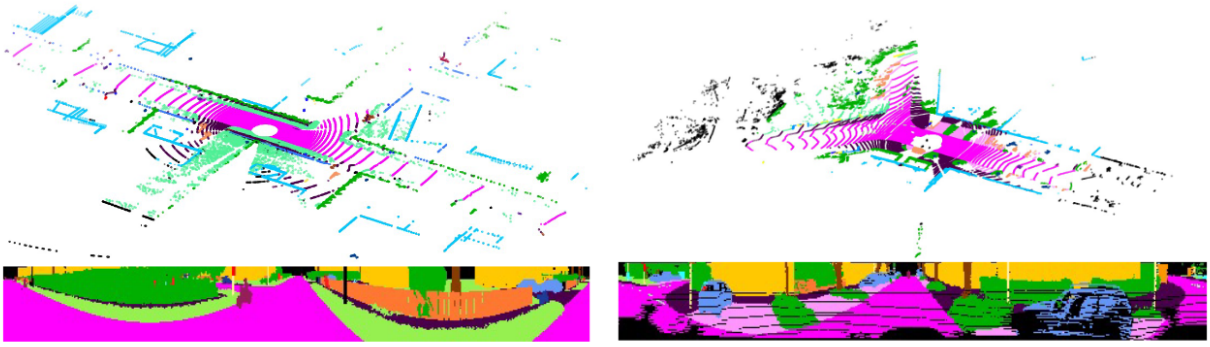

Figure 1.2: Comparison between a synthetic LiDAR scan (left) and a real scan (right). [2]

To tackle the aforementioned challenges, research were conducted toward unsupervised domain adaptation (UDA) within the context of 3D semantic segmentation. UDA aims to transfer knowledge from a labeled source domain to a completely unlabeled target domain. Essentially, UDA encourages deep learning models to learn domain-invariant representations that are robust to domain shift. Current UDA methods that exclusively operate on 3D point cloud data, referred to as uni-modal UDA [3], can be roughly categorized into two groups: adversarial-based methods and self-training-based methods [8].

Adversarial-based approaches involve a min-max game between a discriminator and a generator, which adversarially reduces domain shift by aligning feature distributions between domains [20, 21, 2, 8]. While adversarial-based methods have demonstrated considerable success in 2D image semantic segmentation, they are known for training difficulties due to their inherent instability. In the field of syn-to-real UDA for 3D semantic segmentation, notable advancements [8, 2] have been achieved through the incorporation of adversarial training within a multi-task framework. Nevertheless, the adversarial branch does not consistently yield strong results when functioning on its own.

In contrast, self-training-based methods [7, 22] iteratively retrain a model using confident pseudo-labeled data generated within the target domain, thereby improving the generalization ability of the model thanks to increased size of effective training data. Self-training-based methods offer the advantage of seamless integration with sorts of data augmentation techniques tailored for 3D LiDAR point clouds, allowing model to learn representations in augmented domains and further improving its generalization ability.

In the context of limited label availability, recent research efforts in the domain of self-supervised learning [23, 24, 25] and few-shot learning [26, 27] for 3D point clouds have also provided with label-efficient solutions. Provided with only a small amount of labeled data in the target domain, these endeavors have yielded remarkable success, surpassing the performance of current UDA methods. This success prompts

an intriguing question: Can we integrate these methods into a single framework to benefit from both sides, leveraging the generalizable features obtained from self-supervised learning and the available source domain labels, to further enhance performance?

## 1.2 Problem Statement

Given unlabeled point clouds $P_t$ from the target domain, which consists of $n$ points $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$ with $d$ as the dimensional feature, and point clouds $P_s$ with a set of semantic labels $L_s$ from the source domain, the objective is to determine the optimal mapping function $\Phi_\theta$ that accurately assigns semantic labels to each point within the point clouds $P_t$, where $\theta$ is the parameters. Mathematically, this can be represented as:

$$\Phi_\theta : P_t \rightarrow L^n. \tag{1.1}$$

In a typical UDA pipeline, two distinct tasks are involved: supervised learning from labeled data in the source domain and unsupervised learning from unlabeled data in the target domain, as shown in Fig 1.3. The supervised learning leverages the labeled source domain to transfer knowledge, while the unsupervised learning aims to learn domain-invariant features from the unlabeled target domain.

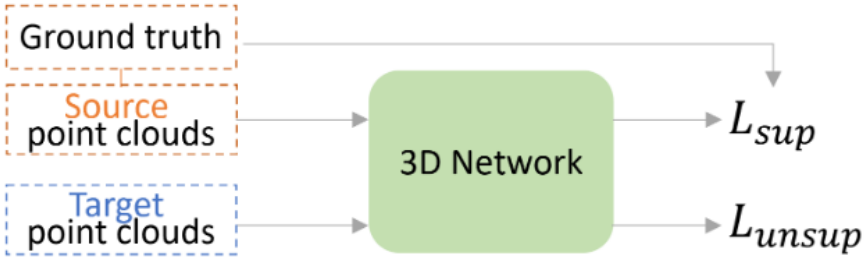


Figure 1.3: A typical pipeline of UDA [3]. $L_{sup}$ is the loss of supervised learning, while $L_{unsup}$ is the one of unsupervised learning.

## 1.3 Focus of This Work

In our work, we establish a connection between prior self-training-based UDA methods and self-supervised representation learning techniques applied to 3D point clouds. Recognizing that both self-supervised learning and self-training techniques are rooted in semi-supervised methods and follow similar principles, we assume a high degree of compatibility between these approaches.

Our approach draws inspiration from representation learning using a contrastive loss objective during a pre-training stage, a concept borrowed from TARL [25]. We integrate this pre-training method with CoSMix [7], a self-training-based UDA technique that trains the model within a mixed augmented space. To bridge these two frameworks and retain the domain-shared representations acquired during the pre-training phase, we devise a fine-tuning technique.

Furthermore, given that self-training methods rely on high-quality pseudo labels, we delve into the calibration of a teacher-student architecture. This investigation aims to enhance the performance and robustness of the trained model by refining the quality of the pseudo labels used during the self-training process.

In summary, our main contributions are the followings:

(1) We present a comprehensive UDA pipeline that seamlessly merges existing self-supervised representation learning techniques and self-training-based UDA methods for 3D point clouds.

(2) Experiments show our method is on par with the state-of-the-art.

(3) Recognizing the crucial role of high-quality pseudo labels in self-training methods, we embarked on a deep exploration of the teacher-student model architecture. Our calibration techniques refine the pseudo-label quality, enhancing both the robustness and performance of the final model.

## 1.4 Thesis Organization

This thesis is structured in the following way. Chapter 2 explores related work in the fields of LiDAR Semantic Segmentation, Self-supervised Representation Learning, Unsupervised Domain Adaptation for 3D Semantic Segmentation, and Calibration for Deep Learning Models. Chapter 3 starts by introducing a theoretical basis, then covers detailed methodology of our approach. Chapter 4 presents our experimental setup and results, followed by a comparison between the proposed method and previous approaches. Chapter 5 engages in a comprehensive discussion, including an ablation study, implications, limitations, and directions for future work. Finally, Chapter 6 concludes the thesis, summarizing key findings and contributions.

# Chapter 2

# Related Work

In this chapter we introduce the literature related to our approach. Section 2.1 We start by exploring LiDAR Semantic Segmentation (Section 2.1) and categories of methods used to tackle this task. Section 2.2 provides a examination of Self-supervised Representation Learning, highlighting its role in acquiring generalized representations without relying on human-labeled data.

Section 2.3 delves into Unsupervised Domain Adaptation for 3D Semantic Segmentation, covering various approaches for adapting models from labeled to unlabeled domains. Lastly, Section 2.4 Calibration for Deep Learning Models introduces the challenges of trustworthiness in deep learning and methods to mitigate overconfidence and enhance model reliability.

## 2.1 LiDAR Semantic Segmentation

Semantic segmentation plays a crucial role in computer vision. In the realm of 3D semantic segmentation, methods can be broadly classified into three groups: point-based, projecton-based, and voxel-based techniques. Moreover, combining these methods can help leverage various sources of information and potentially improve performance.

Point-based methods [28, 29] utilize point features and positions as inputs, employing various operators to gather information from neighboring points. Voxel-based methods [30, 31], on the other hand, partition 3D space into regular voxels and apply sparse convolutions. Modern libraries [31, 32] for sparse computation have dramatically increased the inference speed in 3D space. However, voxelization causes loss of information depending on the chosen resolution. Projecton-based methods [19, 1, 33] project 3D point clouds into 2D range or Bird's Eye View (BEV) images, then apply 2D semantic segmentation methods, which has the advantages of lower computational cost compared to other methods directly working in 3D space [9]. Additionally, methods that combine these modalities have shown significantly improved performance. Leveraging the widely adopted Vision Transformer backbone in 3D vision has also enhanced LiDAR semantic segmentation [33].

## 2.2 Self-supervised Representation Learning

Self-supervised representation learning aims to learn representations capable of generalizing to downstream tasks without relying on human-labeled data [34]. In its early stages, this approach utilized pretext tasks to acquire valuable representations. In the field of LiDAR perception, similar to 2D self-supervised learning methods, reconstruction-based and contrastive learning based methods are widely employed. In addition, prediction-based and flow-based methods have been developed to align with the intrinsic characteristics of urban LiDAR point clouds [34].

Successful reconstruction-based pre-training methods in 2D vision and natural language processing (NLP) have been adapted for 3D point cloud. For example, Zhang et al. [35] proposed using a transformer-based masked auto encoder that reconstructs masked surfels for point cloud. Ye et al. [26] introduced PointBert, a mask point modeling (MPM) based on the masked language modeling (MLM) strategy in BERT [36].

Furthermore, contrastive learning methods that contrast 3D region of interests has proven effective for 3D semantic segmentation as downstream task. SegContrast [23] enhances downstream semantic segmentation by extracting class-agnostic segments from point clouds and applying a segment-wise contrastive loss, producing robust and fine-grained feature representations transferable between datasets. Similarly, Proposal-Contrast [24] contrasts region proposals in a 3D context using spherical proposals instead of bounding boxes, and employs cross-attention for capturing geometric relations within each proposal. Moreover, TARL [25] extracts segments by aggregating sequences of point clouds and applying clustering. It then contrasts the point-wise features to segment mean-wise features with a contrastive loss.

These methods have proven their effectiveness and, to a certain degree, align with the objectives of Unsupervised Domain Adaptation, as they are both label-efficient approaches that minimize the need for labeled data in downstream tasks.

## 2.3 Unsupervised Domain Adaptation for 3D Semantic Segmentation

Unsupervised Domain Adaptation (UDA) aims to adapt a 3D semantic segmentation model from a labeled domain to an unlabeled domain. Recent work has also explored synthetic-to-real adaptation for point clouds.

For indoor 3D scenes, DODA [15] remains the sole UDA research for semantic segmentation. It proposes a syn-to-real approach that mimics occlusion and noise patterns present in real scenes and create an intermediate domain through the manipulation of source and target cuboids.

For urban scenes, some existing approaches [2, 19, 37, 14, 38] attempt to address this challenge by projecting LiDAR point clouds onto depth images and subsequently applying 2D UDA techniques to mitigate domain shifts. Nevertheless, it is important to note that the projection from 3D to 2D introduces a loss of geometric information, which is further compounded by the absence of color information typically present in 2D data.

Another line of approaches [16, 12, 7, 39] directly perform domain adaptive semantic segmentation on LiDAR point cloud. For instance, Complete & Label [12] learns to complete input voxels to represent the underlying surface, thus creates a new domain used as a pivot for domain adaptation. PCT [16] employs GANs to translate synthetic point clouds to match the sparsity and appearance of real ones. SALUDA [39] further proposes to employ an auxiliary implicit surface completion task along with the original semantic segmentation task to learn domain invariant representation. PMAN [8] proposed an adversarial multitask network that incorporates self-supervised learning on LiDAR intensity and an auxiliary nonparametric classifier using class prototypes from the source domain to improve semantic consistency and mitigate traditional adversarial learning side effects.

Additionally, some methods [7, 40] mix point clouds from source and target domains to generate intermediate representations with reduced domain discrepancies. Polarmix [40] uses two cross-scan augmentation techniques. Firstly, it utilizes scene-level swapping, which exchanges point cloud sectors between two LiDAR scans. Secondly, it employs instance-level rotation and paste, which cuts and rotates point cloud instances across scans. On the other hand, CoSMix [7] takes a different approach by extracting segments from both domains by (pseudo) labels and subsequently pasting them into the other domain. This thus creates augmented intermediate domains, enhancing the model's generalization to the target domain.

## 2.4 Calibration for Deep Leaning Models

Trustworthiness is a fundamental requirement for deep learning applications in real-world scenarios [41]. However, achieving this is challenging, as modern deep learning models are known to be overconfident when making predictions [6]. For example, in the context of this thesis topic, self-training methods using teacher-student architecture rely on pseudo-labels with high quality, which could lead to poor performance when the teacher is overconfident in incorrect predictions or underconfident in correct predictions. To mitigate this issue, one of the simplest, fastest and effective calibration methods is Temperature Scaling (TempScale) [6]. However, in our setting, the ground-truth target labels are not accessible. Without ground-truth labels, some methods [42, 43, 44] have achieved good calibrations and uncertainty estimations of deep learning models by training an extra generative model which can yield the likelihood estimation of output logits. For instance, Density-Softmax [42] makes use of a flow-based model to estimate the probability density of backbone output, which is then used to calibrate the model when multiplied with the logits generated by the classifier. TransCal [43] calibrates a model in the context of UDA, by adopting a logistic regression classifier to estimate the density ratio of domains to further estimate target calibration error. Calibrated Teacher [44] utilizes a regression calibrator to further underscore the effectiveness of calibration in self-training. However, training an additional model for processing 3D point cloud can be memory-intensive. As a result, we further explore PseudoCal [45] which proposes to calibrate deep learning models on a pseudo mixed validation set using TempScale in a supervised manner.

# Chapter 3

# Materials and Methods

In this chapter, we begin by laying the groundwork for our methodology. Section 3.1 covers essential aspects such as deep learning in 3D space, point cloud pre-processing techniques, and the calibration of deep learning models. Subsequently, Section 3.2 elaborates on our 3-stage approach for tackling the UDA challenge in 3D semantic segmentation.

## 3.1 Prerequisite

A complete UDA pipeline consists of several key components. In this thesis, we use the popular voxel-based sparse 3D U-Net, implemented in the MinkowskiEngine library [31], as our backbone to facilitate fair comparisons with other Unsupervised Domain Adaptation (UDA) methods.

### 3.1.1 Learning in 3D Space

Convolutional neural networks (CNN) rely on convolution and pooling operations to learn feature maps. When it comes to 3D space, the learning cost of CNN can be expansive. To make it efficient to learn in high dimensions, modern frameworks for 3D computer vision exploit sparse representation of data and utilize sparse convolution to accelerate computation. Fig. 3.1 depicts the process of sparse convolution. For example, Choy et al. [31] proposed a a 4D spatio-temporal convolutional neural network called MinkowskiNet for 3D video perception. A generalized sparse convolution is proposed to effectively process high-dimensional data. For LiDAR point cloud data, we can represent a set of 3D coordinates $\mathbf{C}$ and the associated features $\mathbf{F}$ as:

$$\mathbf{C} = \begin{bmatrix} x_1 & y_1 & z_1 & b_1 \\ x_2 & y_2 & z_2 & b_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & b_N \end{bmatrix}, \mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}, \tag{3.1}$$

where $b_i$ is the batch indices of $i$-th coordinate and $f_i$ is a vector of usually 4 dimensions representing xyz coordinates and intensity value for LiDAR data.

The associated open-source auto-differentiation library, MinkowskiEngine, provides tools to voxelize and batch the point cloud data, and offers sparse implementations of layers commonly used in CNN. Like many other UDA methods for 3D semantic segmentation, our framework is independent to the model architecture. We use a provided backbone MinkowskiUnet34C that is also widely used in many other related works [7, 25, 39], ensuring fair and consistent comparisons. The architecture of a MinkowskiNet can be illustrated in Fig. 3.2.

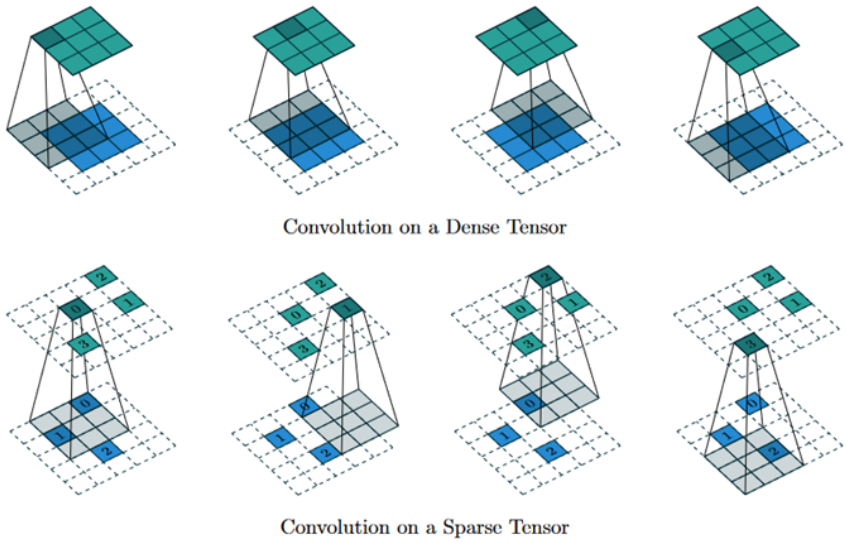


Figure 3.1: Sparse convolution [4].

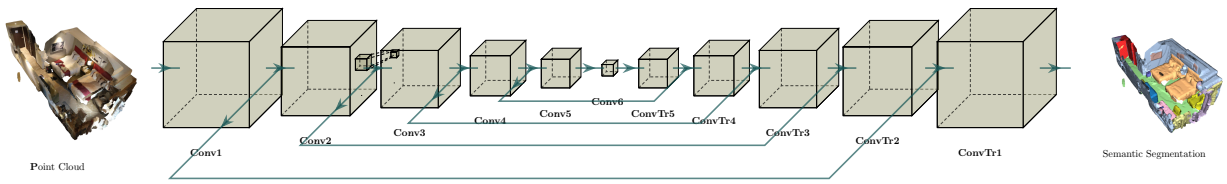


Figure 3.2: Semantic Segmentation Sparse Tensor Network [4].

In semantic segmentation task, the objective is to maximize the mIoU (mean Intersection over Union) as an evaluation metric to assess overall segmentation performance. The loss function used during training is typically the Dice loss [46], Cross Entropy loss or their variants. The Dice loss for the binary-class case can be formulated as follows:

$$\mathcal{L}_{\text{DL2}} = 1 - \frac{2\sum_{i=1}^{N} p_i \cdot g_i}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} g_i^2}, \tag{3.2}$$

where the sums run over the $N$ voxels, of the predicted binary segmentation mask $p_i \in P$ and the ground truth binary mask $g_i \in G$. For multi-class case, the final Dice Loss is calculated as the mean over all $C$ classes:

$$\mathcal{L}_{\text{Dice}} = \frac{1}{C}\sum_{i=1}^{C} \mathcal{L}_{\text{DL2}}. \tag{3.3}$$

On the other hand, the Cross Entropy loss can be expressed as:

$$\mathcal{L}_{\text{CE}}(p, y) = -\sum_{i=1}^{N} y_i \cdot \log(p_i), \tag{3.4}$$

where $y_i$ and $p_i$ represent the ground truth label and predicted class probabilities for class $i$, respectively, $N$ is the number of classes. It is worth noting that in the implementations, variants of these losses may be used.

### 3.1.2 Point Cloud Pre-processing

In this subsection, we break down our point cloud preprocessing pipeline and introduce several fundamental components.

**Ground Removal.** We adopt PatchWork [5], an real-time unsupervised ground detection algorithm for LiDAR point cloud, to seperate ground points and non-ground points, which will help yield better results when clustering points of potential meaningful objects. It encodes a point cloud into a Concentric Zone Model-based representation, which efficiently distributes cloud points among bins while maintaining computational efficiency. It then applies Region-wise Ground Plane Fitting to identify hard samples and Ground Likelihood Estimation to reduce false positives. The algorithm can be illustrated in Fig 3.3.

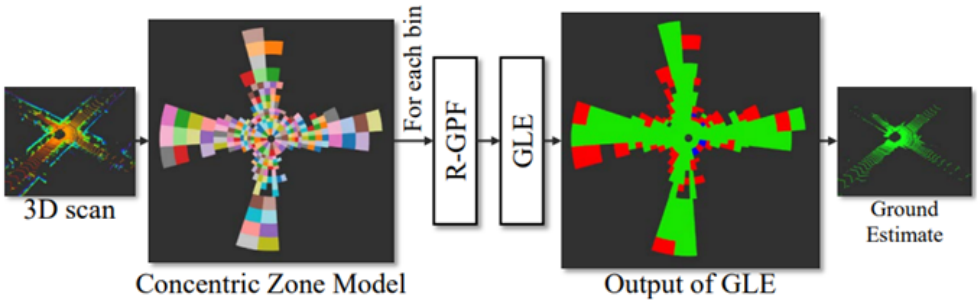


Figure 3.3: Overview of Patchwork [5]

**Registration.** For LiDAR scan data with pose information, we directly transform the point coordinates into a uniform frame. Otherwise, we perform point cloud registration and optimization to align these point clouds into a common coordinate system using tools implemented in the Open3D library. Specifically, the registration process involves pairwise registration of each LiDAR scan to others using an Iterative Closest Point (ICP) algorithm. In the pairwise registration, the algorithm iteratively aligns the source cloud with the target cloud, estimating transformation matrices and information matrices to minimize the differences between corresponding points. The full registration function directs this process among multiple LiDAR scans by creating a pose graph to keep track of the transformations between the point clouds.

**Clustering.** Once we have an aggregated and processed point cloud, the next step is to extract semantically meaningful objects from the data. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [47] is a clustering algorithm ideally suited for this task. We can automatically group points into clusters representing distinct objects or regions in the 3D scene.

**Voxelization.** To able to perform sparse convolution, we need to map the continuous 3D coordinates of point cloud data to a discrete grid. This process is called quantization or voxelization. The MinkowskiEngine library offers convenient functions for this purpose, allowing point cloud coordinates to be quantized with respect to a specified resolution [31].

### 3.1.3 Model Calibration

We introduce calibration for neural networks and relevant metrics. Calibration aims for the model's confidence scores to accurately reflect the likelihood of correctness. Neural networks output "confidence" scores $\hat{p}$ along with predictions $\hat{y}$ in classification. Ideally, these confidence scores should match the true correctness likelihood, which can be expressed as $P(\hat{y} = y|\hat{p} = p) = p, \forall p \in [0, 1]$ . For example, if we assign 80% confidence to 100 predictions, then we would expect that 80% of the predictions are actually correct. If this is the case, we say the network is calibrated [6]. However, this perfect calibration is impossible to achieve. There are several metrics to measure if a model is calibrated.

**Reliability Diagram.** Reliability Diagram [48] is simple way to visualize calibration by plotting accuracy as a function of confidence. Since confidence should reflect accuracy, we would like for the plot to be an identity function. If accuracy falls below the main diagonal, then our network is overconfident. This happens to be the case for most neural networks, such as this ResNet trained on CIFAR100 as shown in Figure 3.4.

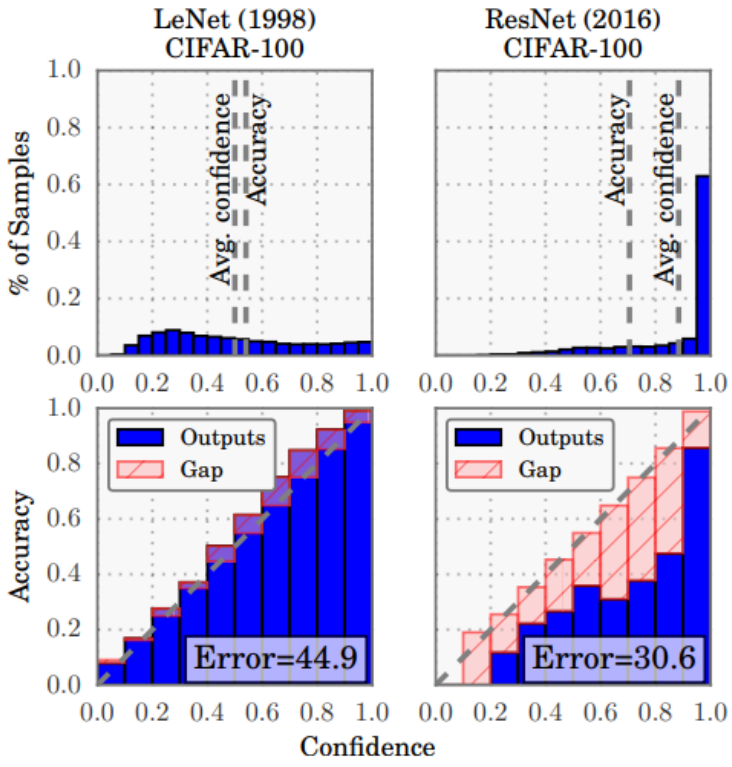


Figure 3.4: Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. [6]

**Expected Calibration Error.** While reliability diagrams provide a visual assessment of calibration, it is more convenient to have a single scalar summary statistic. One widely used metric for evaluating calibration error is the Expected Calibration Error [49] (ECE). The ECE divides probability predictions into $M$ bins, with each bin $B_m$ representing a range of confidence scores. It then calculates the weighted average of the difference between observed accuracy and predicted confidence within each bin.

$$\mathcal{L}_{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|, \tag{3.5}$$

where $n$ represents the number of samples, and for the $m$-th bin, the accuracy is computed as $acc(B_m) = |B_m|^{-1} \sum_{i \in B_m} \mathbf{1}(yi = yi)$, and the confidence is computed as $conf(B_m) = |B_m|^{-1} \sum_{i \in B_m} \hat{p}_i$.

**Temperature Scaling.** Temperature Scaling (TempScal) [6] is a widely-used calibration method in scenarios where data is assumed to be independently and identically distributed (i.i.d.). It is a post-hoc calibration technique that optimizes a temperature scaler $T$ using the negative log-likelihood (NLL) loss on a labeled validation set by comparing softmax predictions with true labels. Let $z$ represent the logit vector associated with input data $x$ and ground-truth labels $y$, and $\sigma(\cdot)$ denote the softmax function. The target temperature, denoted as $\hat{T}$, can be obtained through the optimization formulated as follows:

$$\hat{T} = \arg\min_{T} \mathbb{E}_{(x_i, y_i) \in D} \mathcal{L}_{NLL}(\sigma(z_i/T), y_i) \tag{3.6}$$

## 3.2 Method

In this section, we introduce the 3-stage pipeline of our method, which are Contrastive Learning (Pre-training), Transfer Learning (Fine-tuning) and Calibrated CoSMix. Our approach begins with the pre-training of a neural network using a mean-teacher scheme, followed by fine-tuning of the pre-trained model in the source domain. Finally, we leverage our implementation of CoSMix [7] to further refine the model through a self-training architecture. Overviews of these stages can be found in Fig. 3.6, Fig. 3.7 and Fig. 3.9.

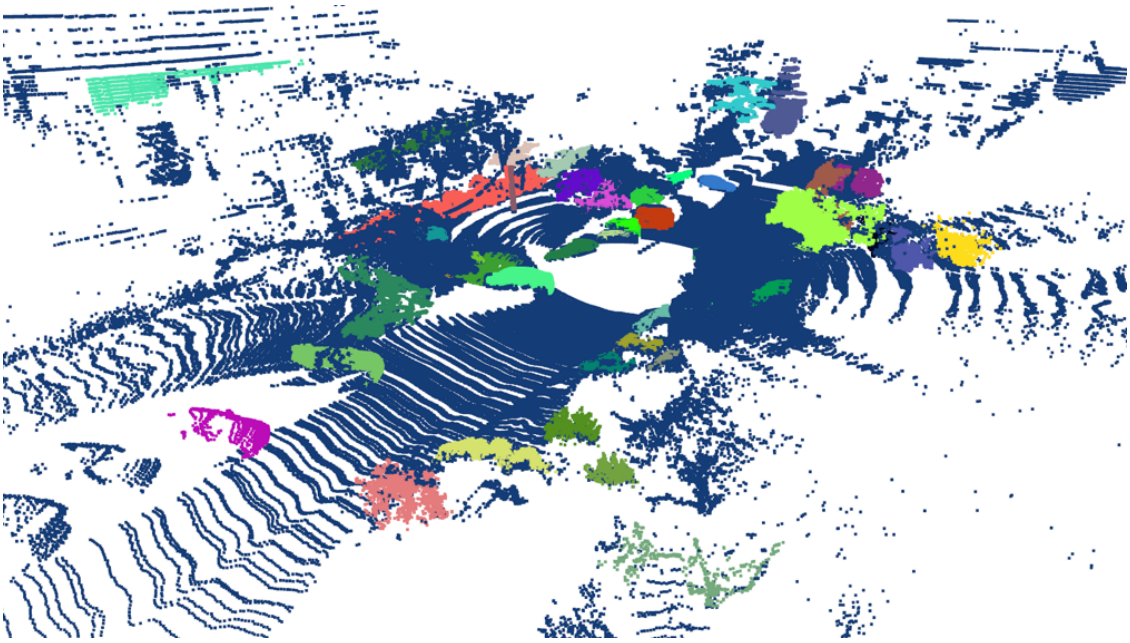

Figure 3.5: Extracted segments from a point cloud aggregated by several scans. Note that the clusters are computed from non-ground points.

### 3.2.1 Contrastive Learning

Inspired by TARL [25] and other UDA methods that employ auxiliary self-supervised learning tasks, we believe that conducting contrastive learning in both source and target domains can enhance domain-invariant representation learning. Our method's contrastive learning process largely aligns with TARL [25]. Given that LiDAR data for urban scenes are captured in time sequences, scans at different times serve as natural augmented views of the same objects, making them suitable for contrastive learning.

**Extract segments.** At each time $t$, we represent the LiDAR data as a point cloud denoted as $P^t = \{p_1^t, ..., p_R^t\}$ as a set of 3D points $p_r \in \mathbb{R}^3$. To extract LiDAR point segments with high-level semantic information, we begin by employing PatchWork [5] to separate the point cloud $P^t$ into two categories: gound points $G^t$ and non-ground points $\hat{P}^t$, where $P^t = G^t \cup \hat{P}^t$ and $G^t \cap \hat{P}^t = \emptyset$. Finally, the HDB-SCAN [47] clustering algorithm is applied to the densified non-ground points $\hat{P}^t$, generating $M$ segments $S^t = \{S_1^t, ..., S_M^t\}$ that represent individual objects. The extracted segments can be illustrated in Fig. 3.5.

**Temproal Contrast.** In order to perform contrastive learning between segment views at different time, we need to extract segments from an aggregated point cloud. We then define a time interval that spans $n$ LiDAR scans, from which we will extract views of objects. Within the interval, we transform the scans

to a common global coordinate frame to be aggregated. For synthetic data with no pose information, we perform point cloud registration as described as the previous Section 3.1.2. This aggregation results in an aggregated point cloud, which we denote as $P = \{P^{t+1}, P^{t+2}, ..., P^{t+n}\}$. As in the individual scan case, we can cluster $\hat{P}$ to get the $M$ segments $S = \{S_1, \ldots, S_M\}$. By keeping the point index mapping from the aggregated point cloud to the individual $n$ scans, we can identify the $n$ segments of the same object viewed at different times as $S_m = \{S^m_{t+1}, \ldots, S^m_{t+n}\}$. We then list the temporal views from each of the $M$ segments as $S_{1:M} = \{S^1_{t+1}, \ldots, S^1_{t+n}, \ldots, S^M_{t+1}, \ldots, S^M_{t+n}\}$.

**Training.** For each batch, we sample two scans $P^{t_1}$ and $P^{t_2}$ at different times, where $t_1 < \frac{n}{3}$ and $t_2 > \frac{2n}{3}$, ensuring that they provide different views of the same object. To ensure the inclusion of the unseen scans, the next batch starts at this unseen interval $\frac{n}{3} < t < \frac{2n}{3}$. By configuring $n$ as a multiple of 3, we include all data within the training sequence for training.

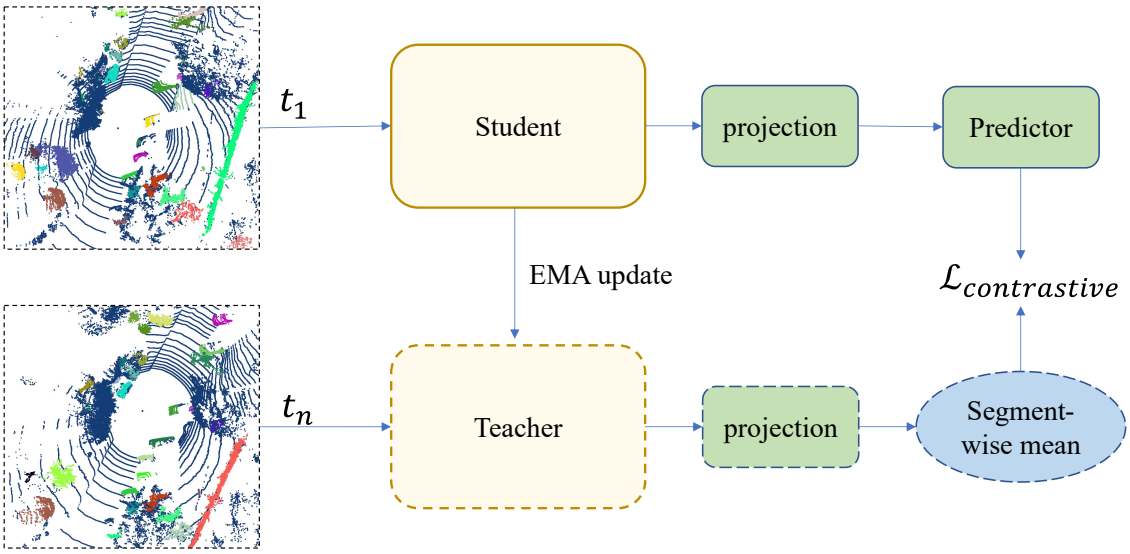


Figure 3.6: Overview of Contrastive Learning stage of our pipeline.

During the forward pass, we compute point-wise features $F^{t_1}$ and $F^{t_2}$ from the backbone, respectively. As the target embedding, we list from $F^{t_2}$ the set of $M$ segments $S^{t_2}$. For each segment in $S^{t_2}$, we compute a mean representation using its associated point-wise features. These mean representations are then processed with a self-attention Transformer encoder, serving as a projection head, to get $M$ target mean feature vectors $\bar{s}^{t_2} \in \mathbb{R}^{M \times D}$, where D denotes the feature dimension.

For each point in a segment from $t_1$, we keep the features at point level, using the Transformer encoder to compute point-wise intra-class correspondences. Segment points will be re-sampled with a maximum number of points $P$ to deal with memory requirement of attention mechanism. Then we input the re-sampled segment points-wise feature $\hat{F}^{t_1}$ to the projection head, follow by another Transformer encoder as predictor. The output is the point-wise feature vectors $s^{t1} \in \mathbb{R}^{M \times P \times D}$.

Using the segment target mean representations $\bar{s}^{t2}$ and the predicted point-wise feature vectors $s^{t1}$ for each segment, we compute a loss to minimize the differences between the point features and the corresponding segment mean representations. We calculate the temperature-scaled cosine similarity $\delta^{t1 \to t2}$ with respect to the corresponding segment-mean representation from $t_2$ as follows:

$$\delta^{t1 \to t2}_{m,p,k} = \frac{(s^{t1}_{m,p})^T \bar{s}^{t2}_k}{\tau}. \tag{3.7}$$

Next, we use the cross-entropy loss to maximize the similarity between each point $p$ from a segment $m$ for all $M$ segments and the corresponding target segment mean representation as follows:

$$\mathcal{L}^{t1\to t2} = -\sum_{m=1}^{M}\sum_{p=1}^{P}\log\left(\frac{\exp(\delta_{m,p,m}^{t1\to t2})}{\sum_k \exp(\delta_{m,p,k}^{t1\to t2})}\right). \tag{3.8}$$

This loss encourages points within the same segment to converge toward a mean representation while separating them from other segments. To ensure bidirectional learning, we repeat the forward pass, swapping $t_1$ and $t_2$. This enables the model to learn correspondences from $t_1$ to $t_2$ and vice versa. The final loss is the sum of losses computed for both directions.

$$\mathcal{L}_{final} = \mathcal{L}^{t1\to t2} + \mathcal{L}^{t2\to t1}. \tag{3.9}$$

To acquire domain-invariant representations, we employ an alternating approach between the source and target domains at each epoch in the training process described above. When dealing with a synthetic source domain, it is possible that this domain may contain repetitive or redundant samples. We implement a sub-sampling strategy on the source data, which balances the number of samples from both domains and facilitates the model to gain more distinctive insights from the target domain.

### 3.2.2 Transfer Learning

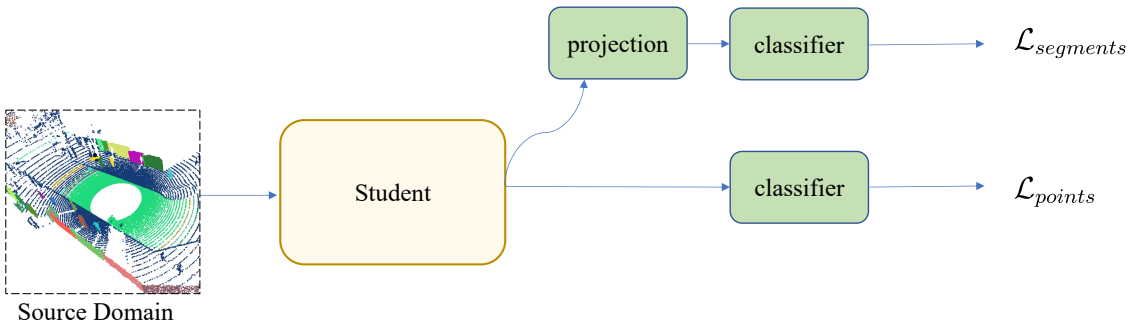


Figure 3.7: Overview of Transfer Learning stage of our pipeline.

After pre-training in both the source and target domains, we proceed with fine-tuning the model for the semantic segmentation task, focusing on the source domain where we have access to semantic labels. To achieve considerable semantic segmentation ability while avoiding over-fitting into the source domain, we limit the number of training epochs for fine-tuning. We define the student network as $\Phi_\theta$, including the backbone and a classification head, where $\theta$ is the parameters. Then, the student's output logits can be denoted as:

$$z^{point} = \Phi_\theta(\mathcal{X}^s). \tag{3.10}$$

Additionally, we leverage the projection head from the contrastive learning phase to perform an additional auxiliary segment-wise prediction task. This auxiliary task acts as a regularization, maintaining the contrastive features learned to the fullest extend and helping to prevent overfitting to the source domain. We denote the output logits of segment-wise head as $z_{segment}$, and the corresponding segment labels as $\mathcal{Y}^s_{segment}$. For each segment, we label the semantic class according to the major class of the points. If the percentage of points from the major class is less than a threshold $th$, this segment is labeled as ignored class.

For segment-wise and point-wise prediction, we use the focal loss [50] and the Lovász-Softmax loss [51] as the loss functions $\mathcal{L}_{segment}$ and $\mathcal{L}_{point}$ respectively, since we found this combination performs better than

Dice loss [46] in fine-tuning. Additionally, a weight factor $\lambda$ is introduced to scale $\mathcal{L}_{segment}$. The objective is to minimize the following loss:

$$\mathcal{L}_{fine-tuning} = \mathcal{L}_{point}(\sigma(z_{point}), \mathcal{Y}^s) + \lambda \mathcal{L}_{segment}(\sigma(z_{segment}/\tau), \mathcal{Y}^s_{segment}). \tag{3.11}$$

We select a small temperature value $\tau$ to steer $\mathcal{L}_{segment}$ toward segment instance classification, which aligns with the previous contrastive learning objective, preserving the learned features and preventing their degradation during fine-tuning.

### 3.2.3 Calibrated CoSMix.

CoSMix [7] is a 3D UDA method that leverages data augmentation techniques for point cloud. It utilizes two mixed point cloud datasets. The first dataset combines the source and confident pseudo-labeled target patches, narrowing the domain gap. The second dataset blends the target with randomly selected source patches to prevent overfitting. Moreover, it employs a teacher-student learning paradigm to iteratively refines pseudo-labels, reducing domain disparity. However, the pseudo-labels used rely on the teacher model's predictions, which are thresholded using Softmax-based confidence values, a method known to produce overconfident labels. Recent calibration methods not only improve label quality but also are shown effective in domain adaptation tasks. For instance, PseudoCal [45] is a model calibration technique for 2D image semantic segmentation that also uses mixing strategy and pseudo labeling. Given the shared similarities between PseudoCal and CoSMix, incorporating these methods has potential for further improvement.

**Semantic Selection.** Before mixing points and labels across domains, class frequency distribution in the source domain is calculated and stored. A point cloud patch corresponds to a subset of points of the same semantic class. In the source domain, patches are chosen based on the frequency distribution of semantic classes. Specifically, we define a function $f$ that randomly selects a subset of classes $\widetilde{\mathcal{Y}}^s$ based on the available labels $\mathcal{Y}^s$ and the corresponding class frequency distribution $\mathcal{P}^s_{\mathcal{Y}}$ at each iteration.

$$\widetilde{\mathcal{Y}}^s = f(\mathcal{Y}^s, 1 - \mathcal{P}^s_{\mathcal{Y}}, \alpha), \tag{3.12}$$

where $\alpha$ denotes the ratio of selected classes for each point cloud and $\widetilde{\mathcal{Y}}^s \in \mathcal{Y}^s$. For instance, by setting $\alpha = 0.5$, a number of patches corresponding to the 50% of the available classes will be selected by sampling them based on their class frequency distribution, i.e., long-tailed classes will have a higher likelihood to be selected.

For the target domain, patches are selected using pseudo-labels $\hat{\mathcal{Y}}^t$ generated by the teacher network, considering their confidence levels and adhering to a predefined threshold. Specifically, we define a pseudo-label selection function $g$. The selected pseudo-labels are defined as

$$\widetilde{\mathcal{Y}}^t = g(\Psi_{\theta'}(\mathcal{X}^t), \zeta), \tag{3.13}$$

where $\Psi_{\theta'}$ is the teacher network, $\zeta$ is the confidence threshold used by the function $g$ and $\widetilde{\mathcal{Y}}^t \in \hat{\mathcal{Y}}^t$. $\mathcal{X}^t$ denotes set of points that correspond to $\hat{\mathcal{Y}}^t$. We further denote the selected point clouds as $\widetilde{\mathcal{X}}^s$ and $\widetilde{\mathcal{X}}^t$.

**Compositional Mix.** This compositional mix module is to blend the selected semantic patches into mixed point clouds. This module involves three key operations: **local random augmentation**, where patches are independently augmented; **concatenation**, where augmented patches are combined with the point cloud of the other domain; and **global random augmentation**, where the mixed point cloud undergoes random augmentation. A mixed point cloud is shown in Fig. 3.8. This mixing process is executed separately for the target-to-source $(t \rightarrow s)$ and source-to-target $(s \rightarrow t)$ branches. We define augmentation function $Aug_{\mathcal{L}}$,

$Aug_{\mathcal{G}}$ for local and global augmentations respectively. Then the new point cloud mixed in the source-to-target $(s \rightarrow t)$ branches can be represented as:

$$\mathcal{X}^{s \rightarrow t} = Aug_{\mathcal{G}}(Aug_{\mathcal{L}}(\widetilde{\mathcal{X}}^{s}) \cup \mathcal{X}^{t}), \tag{3.14}$$

If we denote the above process as a mixing function $mix$, then the augmented mixed point clouds and new labels can also be represented as:

$$\mathcal{X}^{s \rightarrow t} = mix(\mathcal{X}^{s}, \mathcal{X}^{t}, 0), \mathcal{Y}^{s \rightarrow t} = \widetilde{\mathcal{Y}}^{s} \cup \hat{\mathcal{Y}}^{t}, \tag{3.15}$$

$$\mathcal{X}^{t \rightarrow s} = mix(\mathcal{X}^{t}, \mathcal{X}^{s}, 1), \mathcal{Y}^{t \rightarrow s} = \widetilde{\mathcal{Y}}^{t} \cup \mathcal{Y}^{s}, \tag{3.16}$$

where 0 (1) means treating the labels of the to-be-mixed point cloud as pseudo-labels (ground-truth labels) respectively.

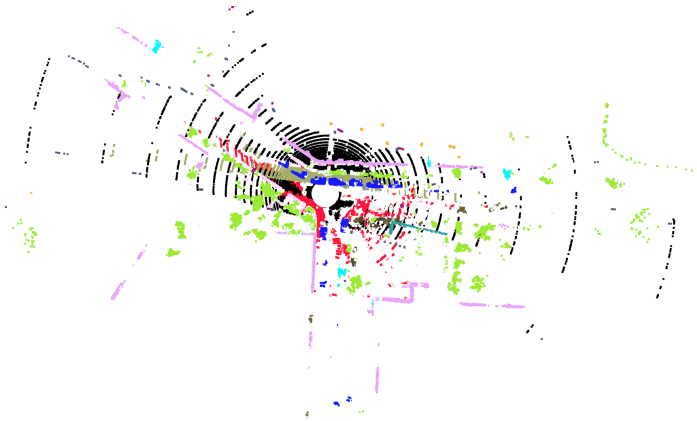

Figure 3.8: An example scan of the mixed point cloud.

**Calibration.** In our implementation, a calibration step takes place before the first training epoch. Essentially, we calculate a temperature value to calibrate the teacher model so that qualitative pseudo labels are accessed. Reported by [45], if two data sets exhibit a similar correct-wrong pattern, they should also share a similar temperature when using TempScal [6]. We split the source dataset into training set and validation set beforehand. Then, we mix the source validation set $\mathcal{X}^{s}_{val}$ and target training set $\mathcal{X}^{t}$ to a mixed validation set $\mathcal{X}^{cal}$ to mimic the correct-wrong pattern as the real target for calibration, on which we further apply temperature scaling to calibrate the teacher network in a supervised manner. Different from the mixing strategy used during training the student network, we consider pseudo target labels as ground-truth labels and source validation labels as pseudo labels. This strategy is designed to create a validation domain that resembles a more unseen domain, thereby resulting in a more conservative temperature value when applying temperature scaling. The mixed validation set is mathematically represented as:

$$\mathcal{X}_{cal} = mix(\mathcal{X}^{s}_{val}, \mathcal{X}^{t}, 1), \mathcal{Y}_{cal} = \mathcal{Y}^{s}_{val} \cup \hat{\mathcal{Y}}^{t}. \tag{3.17}$$

Subsequently, we can obtain a temperature value $\hat{T}$ by referring to Equation 3.6. This value is applied when the teacher network generates pseudo-labels. It's important to note that temperature scaling does not influence the model's accuracy but rather adjusts the confidence-accuracy curve. The calibration in this context can stablize the training by help us choose the confidence threshold for generating pseudo-labels.

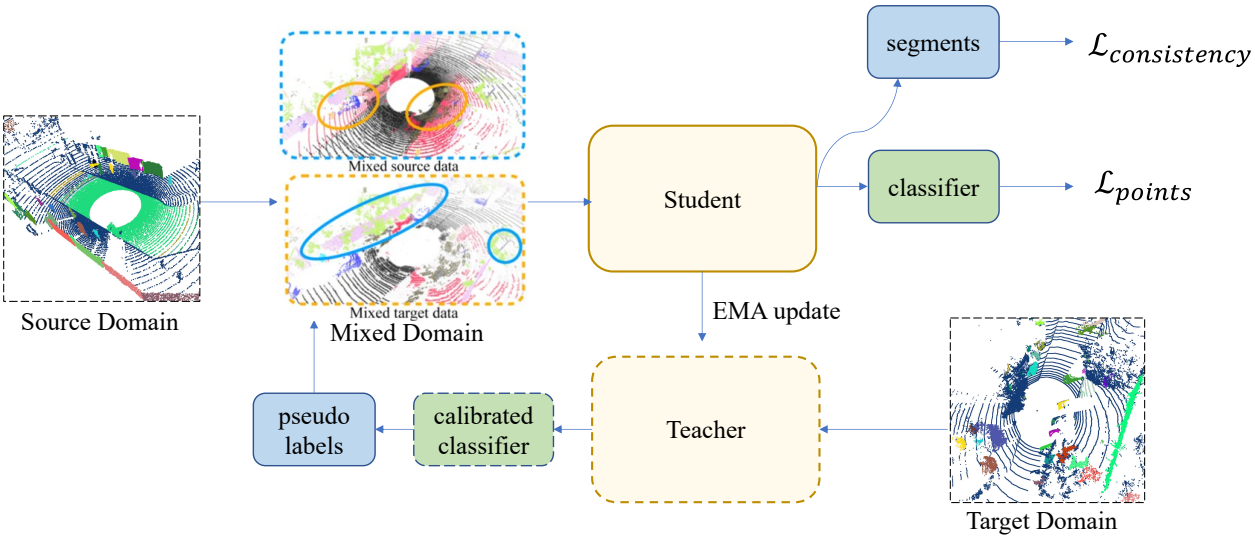


Figure 3.9: Overview of proposed calibrated CoSMix. Part of the figure is adapted from [7].

**Training.** CoSMix leverages a teacher-student learning approach to refine pseudo-labels iteratively and minimize domain disparities. For each batch, the teacher network $\Phi_{\theta\prime}$ starts to generate pseudo-labels on the target domain by retaining predictions with Softmax-based confidence above a threshold $\zeta$. When utilizing intensity values as input, we employ a two-step normalization process. First, we normalize the batch source intensity $I_s$ using its own mean $\mu_s$ and variance $\sigma_s$. Subsequently, we recover it using the mean $\mu_t$ and variance $\sigma_t$ derived from the batch target intensity. The same procedure is applied to the batch target intensity. Mathematically, these can be represented as follows:

$$\hat{I}_s = \frac{I_s - \mu_s}{\sigma_s}, \tag{3.18}$$

$$I'_s = \hat{I}_s \cdot \sigma_t + \mu_t, \tag{3.19}$$

$$\hat{I}_t = \frac{I_t - \mu_t}{\sigma_t}, \tag{3.20}$$

$$I'_t = \hat{I}_t \cdot \sigma_s + \mu_s, \tag{3.21}$$

where $\hat{I}_s$, $\hat{I}_t$ represent the normalized batch source and target intensity respectively, $I'_s$ and $I'_t$ represents the recovered batch source and target intensity respectively.

Then the above described Semantic Selection, Compositional Mix are applied to get mixed point clouds $\mathcal{X}^{s\to t}$, $\mathcal{X}^{t\to s}$ and labels $\mathcal{Y}^{s\to t}$, $\mathcal{Y}^{t\to s}$. To update the student network $\Phi_\theta$, segmentation loss on $(s \to t)$ branch is define as:

$$\mathcal{L}_{s\to t} = \mathcal{L}_{seg}(\Phi_\theta(\mathcal{X}^{s\to t}), \mathcal{Y}^{s\to t}), \tag{3.22}$$

where $\mathcal{L}_{seg}$ is implemented as the Dice Loss [46]. Similarly on $(t \to s)$ branch, the segmentation loss is defined as:

$$\mathcal{L}_{t\to s} = \mathcal{L}_{seg}(\Phi_\theta(\mathcal{X}^{t\to s}), \mathcal{Y}^{t\to s}). \tag{3.23}$$

Furthermore, we utilize the projection head to extract point-wise features of each segment $s^{s\to t}$ and $s^{t\to s}$. A cosine embedding loss serves as an additional constraint since $s^{s\to t}$ and $s^{t\to s}$ correspond to the

same points within the augmented space:

$$\mathcal{L}_{CosEmbedding} = 1 - cos(s^{s \to t}, s^{t \to s}). \tag{3.24}$$

Finally, a total segmentation loss as follow is to be minimized.

$$\mathcal{L}_{total} = \mathcal{L}_{s \to t} + \mathcal{L}_{t \to s} + \omega \mathcal{L}_{CosEmbedding} \tag{3.25}$$

Additionally, the teacher network's parameters are updated periodically through an exponential moving average mechanism, contributing to the ongoing improvement of pseudo-labels and domain adaptation. Every $\gamma$ iterations, we update the teacher parameters $\theta'$ as follows:

$$\theta'_i = \beta \theta'_{i-1} + (1 - \beta)\theta, \tag{3.26}$$

where $i$ indicates the training iteration and $\beta$ is a smoothing coefficient hyperparamenter.

# Chapter 4

# Experiments and Results

This chapter starts from introducing the Dataset and Metrics in Section 4.1 used for evaluating our method. Moreover, we provide a comprehensive overview of the Implementation Details in Section 4.2, encompassing essential parameters and configurations for the various stages of our UDA pipeline. Finally, we compare the proposed method with previous approaches in Section 4.3.

## 4.1 Datasets and Metrics

Let us revisit the goal of this thesis: adapting a model from the synthetic source domain to the real-world target domain without relying on ground-truth target labels for LiDAR semantic segmentation in urban environments. Our choice for the source domain is SynLiDAR [16], a synthetic dataset that has the largest number of scans, annotated points and semantic classes among all synthetic autonomous driving datasets. For target domains, we consider SemanticKITTI [52] and SemanticPOSS [53], which are widely used in previous UDA for LiDAR semantic segmentation research. These two dataset offer quite different outdoor environments and are captured from different spatial perspectives. Although SemanticPOSS is smaller than SemanticKITTI in terms of dataset size, it contains a larger quantity of moving objects than SemanticKITTI, such as person and rider. This contrast between the two target datasets serves as a valuable benchmark for evaluating the generalization capabilities of UDA methods in our experiments. Furthermore, Table 4.1 provides an overview of various outdoor LiDAR point cloud datasets.

**SemanticKITTI.** The LiDAR data of SemanticKITTI is collectd from urban environments in Germany, using Velodyne HDL-64E sensor mounted on the vehicle's roof. It provides dense semantic annotations for each individual scan of sequences 00–10 in KITTI dataset [54]. According to the official setting, sequence 08 is the validation split, while the remaining are the train split. SemanticKITTI uses sequences 11–21 in KITTI as the test set, whose labels are held on for blind online testing. We follow the setting of previous UDA methods, where all results are reported from evaluation on sequence 08.

**SemanticPOSS** is generated in a university campus using vehicle equipped with a Pandora 40-line sensor module and a GPS/IMU localization system to collect point clouds data. The dataset contains 5 sequences, where we follow the same validation protocal as [7, 8] and use sequence 03 for validation and others for training.

**SynLiDAR.** SynLiDAR provides high-quality synthetic point cloud data collected from various virtual scenes created using Unreal Engine 4. These virtual scenes simulate outdoor environments such as cities, towns, and harbors, offering a wide range of scenarios. The dataset includes precise point-wise annotations

for 32 semantic classes. Moreover, Its intensity values are simulated using a rendering model's prediction for the synthetic point clouds. It is worth noting that the rendering model is trained on SemanticKITTI. Therefore, the intensity values of SynLiDAR exhibit similarity with those of SemanticKITTI, while showing a significant gap with those of SemanticPOSS. We use the 'mini' version of this dataset defined by [7] for training, which is essentially a subsampled representation of the complete dataset.

**Class Mapping.** We map SynLiDAR labels from 32 classes into 13 classes for SemantickPOSS and 19 classes for SemanticKITTI.

**Metrics.** The commonly used metric for evaluating the performance of semantic segmentation models is the mean Intersection over Union (mIoU). It quantifies the degree of overlap between predicted and ground truth segmentation masks using a single value. It is calculated as follows:

$$\text{mIoU} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i + \text{FN}_i}, \tag{4.1}$$

where N is the number of classes or categories, $\text{TP}_i$ is the number of true positives for class $i$, $\text{FP}_i$ is the number of false positives for class $i$, $\text{FN}_i$ is the number of false negatives for class $i$.

In addition to the mIoU, we also examine IoU for each individual class, providing assessment of our model's performance on a per-class basis.

Table 4.1: Overview of Outdoor Autonomous Driving Datasets.

| Dataset | Year | #Samples | #Classes | Representation | Label |
|---|---|---|---|---|---|
| KITTI [54] | 2013 | 15K frames | 8 | RGB & LiDAR | Bounding box |
| nuScenes [55] | 2020 | 40K | 32 | RGB & LiDAR | Point category & Bounding box |
| Waymo [56] | 2020 | 200K | 23 | RGB & LiDAR | Point category & Bounding box |
| STF [57] | 2020 | 13.5K | 4 | RGB & LiDAR & Radar | Bounding box |
| ONCE [58] | 2021 | 1M scenes | 5 | RGB & LiDAR | Bounding box |
| SemanticKITTI [52] | 2019 | 43,552 scans | 28 | LiDAR | Point category |
| SemanticPOSS [53] | 2020 | 2,998 scans | 14 | LiDAR | Point category |
| SynLiDAR [16] | 2022 | 198,396 scans | 32 | Synthetic LiDAR | Point category |
| SemanticSTF [59] | 2023 | 2,086 scans | 21 | RGB & LiDAR | Point category |

**Intensity as input.** Given that SynLiDAR's intensity values are generated from SemanticKITTI data, they share a similar distribution when compared to SemanticPOSS. However, incorporating intensity as an input feature introduces additional dissimilarity between the source and target domains. Following [8], we will investigate the impact of including intensity as an input feature in the upcoming experiments and present the results in this chapter, both with and without intensity as input.

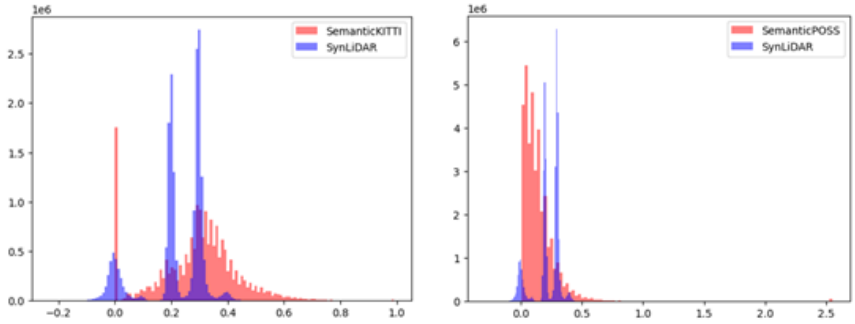


Figure 4.1: Intensity distribution of 3 datasets [8]. Note that the intensity values of SemanticPOSS have been devided by 100.

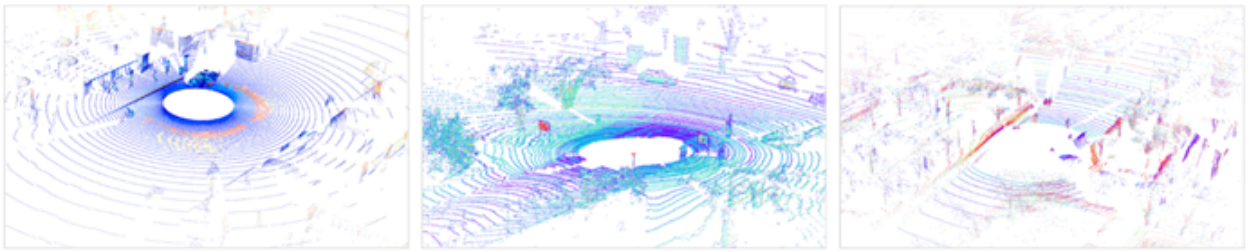


Figure 4.2: Visualization of intensity values of scenes from 3 domains [8]. Red regions in the each image correspond to high intensity, while blue ones correspond to low intensity.

## 4.2 Implementation Details

We employed the voxel-based neural network MinkowskiUnet34C as our backbone. The training pipeline was implemented using PyTorch [60] and PyTorch Lightning [61]. Contrastive learning pre-training was performed with a batch size of 4 using 4 Quadro RTX 6000 GPUs on the Euler cluster provided by ETH Zurich. And the Fine-tuning and Calibrated Cosmix stages are conducted on a single NVIDIA GeForce RTX 3090. To complement the following paragraphs, we list the hyperparameter choices in Table 4.2, Table 4.3 and Table 4.4.

**Contrastive Learning.** In the contrastive learning phase, we use the AdamW optimizer [62] with a learning rate of $2 \times 10^{-4}$ and a weight decay of $10^{-4}$. Training is performed with a batch size of 4. For SynLiDAR $\rightarrow$ SemanticKITTI, we train for 600 epochs, alternating between the SemanticKITTI and SynLiDAR datasets every epoch. For SynLiDAR $\rightarrow$ SemanticPOSS, we randomly select a subset of samples in SynLiDAR with a ratio of 0.1 to match the size of SemanticPOSS and train for 4500 epochs. Following the settings of [25], we set $n = 12$ for the scans of SemanticKITTI to be aggregated. For SemanticPOSS, we set $n = 6$. The voxel resolution is configured to 0.05 m for the input point clouds, with a maximum of 40,000 points per point cloud. During segment pooling, we limit it to a total of $M = 50$ segments with a maximum of $P = 300$ points per segment to avoid memory overflow. We use $\tau = 0.1$ to compute the temperature-scaled cosine similarity $\delta$ in Equation 3.7 and a momentum of 0.999 to update the momentum network based on the online network weights every step.

Table 4.2: Contrastive Learning Parameters.

| Parameter | Value | Description |
|---|---|---|
| learning rate | $2 \times 10^{-4}$ | Learning rate |
| weight decay | $10^{-4}$ | Weight decay coefficient of optimizer |
| batch size | 4 | Batch Size |
| epoch$_{S \to K}$ | 600 | Number of epochs (SynLiDAR $\to$ SemanticKITTI) |
| epoch$_{S \to P}$ | 4500 | Number of epochs (SynLiDAR $\to$ SemanticPOSS) |
| sub-sample rate | 0.1 | The percentage of selected source data samples (SynLiDAR $\to$ SemanticPOSS) |
| $n_{S \to K}$ | 12 | Number of scans aggregated for SemanticKITTI |
| $n_{S \to P}$ | 6 | Number of scans aggregated for SemanticPOSS |
| voxel size | 0.05 m | Voxel resolution |
| num. points | 40,000 | Max number of points per point cloud |
| $M$ | 50 | Max number of segments per scan |
| $P$ | 300 | Max number of points per Segment |
| $\tau$ | 0.1 | Temperature coefficent for Eq. 3.7 |
| $\beta$ | 0.999 | Momentum coefficent for updating teacher |

**Fine-tuning.** We fine-tune the model for the semantic segmentation task using only the source dataset SynLiDAR. If the ratio of major point class within a segment is less then the threshold $th = 0.7$, the segment will be labeled as the ignored class. A weight factor of $\lambda = 0.005$ is chosen for the segment-wise loss. For SynLiDAR $\to$ SemanticKITTI, we train using the AdamW optimizer with a batch size of 8 and a learning rate of $10^{-4}$ for 1 epoch. For SynLiDAR $\to$ SemanticPOSS, we down-sampled SynLiDAR with a ratio of 0.2 and trained for 6 epochs with the Stochastic Gradient Descent (SGD) optimizer and a learning rate of $10^{-3}$. Using a smaller number of epochs helps prevent overfitting to SynLiDAR. The temperature value $\tau = 0.1$ was set for the segment-wise prediction loss, as the same as in the previous phase, to better preserve segment-wise features.

Table 4.3: Fine-tuning Parameters.

| Parameter | Value | Description |
|---|---|---|
| $th$ | 0.7 | Threshold for labeling a segment from based on the major class of its points |
| $\lambda$ | 0.005 | Segment-wise loss weight as in Eq. 3.11 |
| epoch$_{S \to K}$ | 1 | Number of epochs (SynLiDAR $\to$ SemanticKITTI) |
| epoch$_{S \to P}$ | 6 | Number of epochs (SynLiDAR $\to$ SemanticPOSS) |
| sub-sample rate | 0.2 | The percentage of selected source data samples |
| $lr_{S \to K}$ | $2 \cdot 10^{-4}$ | Learning rate (SynLiDAR $\to$ SemanticKITTI) |
| $lr_{S \to P}$ | $10^{-3}$ | Learning rate (SynLiDAR $\to$ SemanticPOSS) |
| $\tau$ | 0.1 | Temperature coefficent for $\mathcal{L}_{segment}$ as in Eq. 3.11 |

**Calibrated Cosmix.** We set the ratio of selected class $\alpha = 0.5$, the weight for the Cosine Embedding loss $\omega = 0.001$, and the confidence threshold $\zeta$ is set to the average confidence of the calibrated teacher network before the first training epoch. Our settings of data augmentation are consistent with those in [7], where we perform rotations between $[-\pi/2, \pi/2]$ and scaling between [0.95, 1.05] for both local and global augmentation, and random downsampling for 50% of the patch points. For SynLiDAR $\to$ SemanticKITTI, we train using the AdamW optimizer with a batch size of 8 and a learning rate of $2 \cdot 10^{-4}$ for 3 epoch. For SynLiDAR $\to$ SemanticPOSS, we used SGD optimizer with a learning rate of $10^{-3}$. The teacher

parameters $\theta_i'$ with a momentum of $\beta = 0.99$ updates every $\gamma = 500$ steps for SemanticKITTI and $\gamma = 1$ step for SemanticPOSS. The same parameter setting is applied to both mixing branches.

Table 4.4: Calibrated Cosmix Parameters.

| **Parameter** | **Value** | **Description** |
|---|---|---|
| $\alpha$ | 0.5 | Class Selection Ratio |
| $\omega$ | 0.001 | Cosine Embedding Loss Weight |
| $\zeta$ | Avg. Confidence | Confidence Threshold |
| rotation | $[-\pi/2, \pi/2]$ | Random rotation of point cloud |
| scaling | [0.95, 1.05] | Random scaling of point cloud |
| down-sample rate | 0.5 | The percentage of selected points in a point cloud |
| $epoch_{S\to K}$ | 3 | Number of epochs (SynLiDAR $\to$ SemanticKITTI) |
| $epoch_{S\to P}$ | 6 | Number of epochs (SynLiDAR $\to$ SemanticPOSS) |
| $lr_{S\to K}$ | $2 \cdot 10^{-4}$ | Learning rate (SynLiDAR $\to$ SemanticKITTI) |
| $lr_{S\to P}$ | $10^{-3}$ | Learning rate (SynLiDAR $\to$ SemanticPOSS) |
| $\beta$ | 0.99 | Momentum coefficent for updating teacher |
| $\gamma_{S\to K}$ | 500 | Number of iterations to update teacher (SynLiDAR $\to$ SemanticKITTI) |
| $\gamma_{S\to P}$ | 1 | Number of iterations to update teacher (SynLiDAR $\to$ SemanticPOSS) |

## 4.3 Comparisons with Previous Methods

In this section, we present our quantitative results for both benchmarks. Our main focus is to compare the results of our proposed method with CoSMix [7], which serves as the baseline for our approach. Additionally, we include a comparison with a set of adversarial-based methods, which employ a different approach. The results for these methods are sourced from PMAN [8], whose backbone is MinkUnet14A, a slightly different variant of our backbone MinkUnet34C. We use the notation "XYZ" to represent that only XYZ coordinates are used as input. Similarly, XYZ coordinates with Intensity as input is denoted as "XYZI". "S" and "A" denote self-training-based method and adversarial-based method respectively.

**SynLiDAR $\to$ SemanticKITTI (XYZ).** Table 4.5 reports the results for adaptation from SynLiDAR to SemanticKITTI, where the proposed method outperforms all other methods achieving an mIoU score of 34.4%. First, compared with the source-only baseline, our method achieves +14% absolute gain. We obtain considerable performance improvement (+4.5%) compared to CoSMix. The per-class IoUs show that the proposed method outperforms the other methods by a large margin on certain classes which are relatively easy to be clustered by HDBSCAN. For instance, we achieve an impressive + 14.8% gain on 'car', + 23.6% on 'truck', and + 9.9% on 'motocycle'.

Table 4.5: Results on SynLiDAR → SemanticKITTI (XYZ).

| Methods | Mech. | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source only | - | 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 |
| AdaptSegNet [20] | A | 52.1 | 10.8 | 11.2 | 2.6 | 9.6 | 15.1 | 35.9 | 2.6 | 62.2 | 10.4 | 41.3 | 0.1 | 58.1 | 17.1 | 68.0 | 38.4 | 38.7 | 35.9 | 20.4 | 27.9 |
| CLAN [63] | A | 51.0 | 15.8 | 16.8 | 2.2 | 7.8 | 18.7 | 46.8 | 3.0 | 68.9 | 11.1 | 44.9 | 0.1 | 59.6 | 17.5 | 71.7 | 41.1 | 44.0 | 37.7 | 19.8 | 30.5 |
| AdvEnt [21] | A | 59.9 | 13.8 | 14.6 | 3.0 | 8.0 | 17.7 | 45.8 | 3.0 | 67.6 | 11.3 | 45.6 | 0.1 | 61.7 | 15.8 | 72.4 | 41.5 | 47.0 | 34.5 | 15.3 | 30.5 |
| DAST [64] | A | 50.6 | 9.7 | 8.2 | 2.6 | 6.9 | 13.1 | 34.4 | 2.2 | 65.2 | 11.0 | 41.7 | 0.1 | 58.0 | 14.1 | 58.6 | 34.4 | 35.6 | 32.7 | 18.8 | 26.2 |
| FADA [65] | A | 49.9 | 6.7 | 5.1 | 2.5 | 10.0 | 5.7 | 26.6 | 2.3 | 65.8 | 10.8 | 37.8 | 0.1 | 60.3 | 21.5 | 60.4 | 37.2 | 31.9 | 35.4 | 17.4 | 25.6 |
| MRNet [66] | A | 49.5 | 11.0 | 12.2 | 2.2 | 8.6 | 16.0 | 46.4 | 2.7 | 60.0 | 10.5 | 41.9 | 0.1 | 55.1 | 16.5 | 68.1 | 38.0 | 40.7 | 36.5 | 20.8 | 28.3 |
| SALUDA [39] | S | 52.1 | 8.6 | 15.0 | 1.9 | 3.8 | 21.4 | 43.4 | 1.7 | 53.7 | 7.8 | 38.0 | 0.1 | 59.2 | 22.3 | 69.1 | 30.8 | 45.1 | 38.0 | 12.6 | 27.6 |
| CoSMix [7] | S | 56.4 | 10.2 | 20.8 | 2.1 | **13.0** | **25.6** | 41.3 | 2.2 | 67.4 | 8.2 | 43.4 | 0 | 57.9 | 12.2 | 68.4 | **44.8** | 35.0 | **42.1** | 17.0 | 29.9 |
| PMAN [8] | A | 71.0 | **14.9** | 24.8 | 1.6 | 6.6 | 23.6 | **61.1** | **5.5** | **75.3** | **10.5** | **54.1** | 0.1 | 47.9 | 17.4 | 69.6 | 38.6 | **61.5** | 37.0 | 18.6 | 33.7 |
| Proposed | S | **85.8** | 2.6 | **34.7** | **25.6** | 11.3 | 24.3 | 50.6 | 4.0 | 69.9 | 1.2 | 43.8 | 0.0 | **68.1** | **17.6** | **75.3** | 38.2 | 37.7 | 41.0 | **21.9** | **34.4** |

**SynLiDAR → SemanticKITTI (XYZI).** Table 4.6 shows that, with intensity as input, the proposed method achieved with an mIoU score of 34.5% with a +12.8% gain over source-only baseline, showing a similar performance as Table 4.5. We can observe that PMAN outperforms us by +1.8% and has a gain of +2.7% over not using intensity, demonstrating the effectiveness of the auxiliary regression task on intensity value. However, it is interesting to note that the proposed method seems to be less sensitive to the inclusion of intensity values as input. The similarity of intensity between both domains might explain why the proposed method achieves consistent performance regardless of whether intensity information is included in the input data.

Table 4.6: Results on SynLiDAR → SemanticKITTI (XYZI).

| Methods | Mech. | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source only | - | 42.5 | 8.6 | 14.6 | 0.6 | 3.8 | 15.3 | 49.0 | 1.3 | 19.4 | 5.5 | 28.0 | 0.1 | 37.8 | 10.3 | 63.1 | 31.5 | 41.6 | 28.1 | 11.3 | 21.7 |
| AdaptSegNet [20] | A | 61.5 | 15.8 | 15.4 | 2.1 | 7.3 | 17.1 | 45.9 | 2.5 | 63.0 | 10.2 | 43.2 | 0.1 | 55.4 | 14.2 | 67.2 | 39.4 | 42.6 | 31.9 | 18.9 | 29.1 |
| CLAN [63] | A | 61.9 | 20.1 | 27.2 | 2.5 | 9.6 | 20.1 | 55.2 | 4.1 | 68.9 | 11.5 | 46.2 | 0.1 | 60.9 | 16.2 | 72.8 | 41.8 | 42.0 | 34.9 | 18.9 | 32.4 |
| AdvEnt [21] | A | 63.5 | 19.0 | 18.1 | 2.8 | 8.7 | 17.3 | 48.4 | 2.2 | 72.1 | 11.6 | 47.2 | 0.1 | 60.0 | 17.8 | 71.5 | 41.8 | 47.7 | 36.0 | 20.0 | 31.9 |
| DAST [64] | A | 60.8 | 16.7 | 19.5 | 2.1 | 7.2 | 17.5 | 39.6 | 2.4 | 65.2 | 11.1 | 43.7 | 0.1 | 57.0 | 16.4 | 56.6 | 38.8 | 35.5 | 31.1 | 17.7 | 28.4 |
| FADA [65] | A | 61.4 | 14.0 | 12.7 | 1.1 | 1.6 | 17.6 | 31.9 | 1.9 | 64.6 | 10.4 | 38.5 | 0.1 | 59.5 | 15.8 | 56.6 | 39.3 | 39.3 | 33.8 | 19.6 | 27.3 |
| MRNet [66] | A | 62.2 | 16.1 | 16.7 | 2.4 | 7.8 | 17.9 | 41.6 | 2.1 | 65.7 | 10.8 | 43.9 | 0.1 | 55.4 | 15.4 | 66.6 | 40.9 | 40.5 | 31.7 | 17.8 | 29.2 |
| PCT [16] | S | 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 |
| PolarMix [40] | S | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 31.0 | |
| CoSMix [7] | S | 61.4 | 8.6 | 24.0 | 0.9 | 5.3 | 25.6 | **55.5** | 3.6 | 58.0 | 8.3 | 41.8 | 0 | 58.7 | 16.4 | 73.5 | 41.5 | 36.7 | **44.7** | 14.4 | 30.5 |
| PMAN [8] | A | 74.9 | **18.0** | 33.3 | 1.9 | 7.8 | **28.7** | 52.7 | 4.0 | **77.6** | **12.6** | **56.9** | 0.1 | 63.8 | 17.9 | 74.1 | **43.8** | **63.5** | 38.3 | 19.3 | **36.3** |
| Proposed | S | **87.2** | 4.4 | **39.0** | **25.6** | **11.3** | 25.6 | 52.6 | **8.5** | 46.9 | 11.3 | 39.1 | 0.0 | **73.9** | **18.5** | **76.3** | 39.0 | 31.8 | 40.1 | **25.1** | 34.5 |

**SynLiDAR → SemanticPOSS (XYZ).** Table 4.7 provides the results for adaptation from SynLiDAR to SemanticPOSS without intensity as input. The proposed method outperforms the baseline method CoSMix, showcasing a performance gain of +2.4%. Nevertheless, making a direct comparison between our results and those of PMAN is challenging in this setting, because of large performance variations between the baseline methods employed by our approach and PMAN. We show the results reported by PMAN in Table 4.8, where the performances of source-only and CoSMix exhibit gaps of 8.8% and 3.8% respectively.

Table 4.7: Results on SynLiDAR → SemanticPOSS (XYZ).

| Methods | Mech. | pers. | bi.clst | car | trunk | veget. | traf. | pole | garb. | build. | cone. | fence | bi.cle | ground | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source only | - | 34.6 | 32.4 | 27.2 | 17.0 | 61.8 | 0.5 | 27.0 | 13.1 | 50.2 | 14.6 | 24.3 | 3.19 | 78.0 | 29.5 |
| CoSMix [7] | S | **58.4** | 46.6 | 37.2 | **22.0** | **71.0** | 17.2 | **35.7** | 29.3 | 69.3 | 23.0 | 30.6 | **10.1** | **80.1** | 40.8 |
| Proposed | S | 58.2 | **60.1** | **39.8** | 21.1 | 69.5 | **24.4** | 34.4 | **32.9** | **70.5** | **23.2** | **39.1** | 3.6 | 79.2 | **42.8** |

Table 4.8: Results from [8] on SynLiDAR → SemanticPOSS (XYZ).

| Methods | Mech. | bi.clst | car | trunk | veget. | traf. | pole | garb. | build. | cone. | fence | bi.cle | ground | pers. | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source only | - | 47.2 | 43.6 | 37.8 | 70.3 | 11.1 | 33.8 | 19.5 | 67.9 | 11.2 | 19.9 | 9.6 | 77.9 | 47.8 | 38.3 |
| AdaptSegNet [20] | A | 43.9 | 48.2 | 39.0 | 69.6 | 15.5 | 33.6 | 21.3 | 64.3 | 12.7 | 25.0 | 11.6 | 76.0 | 49.9 | 39.3 |
| CLAN [63] | A | 43.9 | 46.6 | 41.3 | 71.0 | 15.1 | 34.3 | 20.4 | 69.6 | 9.5 | 23.2 | 12.0 | 75.1 | 51.3 | 39.5 |
| AdvEnt [21] | A | 44.6 | 47.6 | 40.3 | 71.2 | 15.6 | 35.6 | 22.0 | 68.4 | 10.6 | 25.9 | 10.4 | 76.7 | 52.3 | 40.1 |
| DAST [64] | A | 44.6 | 46.1 | 39.7 | 71.1 | 17.7 | 34.0 | 20.3 | 68.7 | 11.4 | 23.0 | 11.6 | 76.2 | 50.3 | 39.6 |
| FADA [65] | A | 39.6 | 41.2 | 38.8 | 69.2 | 16.3 | 32.1 | 18.1 | 67.9 | 11.5 | 22.0 | 13.0 | 71.4 | 47.9 | 37.6 |
| MRNet [66] | A | 43.5 | 47.2 | 39.1 | 70.4 | 15.5 | 32.8 | 22.0 | 66.1 | 13.2 | 24.2 | 11.2 | 76.8 | 50.0 | 39.4 |
| CoSMix [7] | S | 53.6 | 47.6 | 44.8 | 75.1 | 16.8 | 37.9 | 25.3 | 72.7 | 19.9 | 39.7 | 10.8 | 80.0 | 56.5 | 44.6 |
| PMAN [8] | A | 52.6 | 61.5 | 46.8 | 75.1 | 18.8 | 36.5 | 21.4 | 74.7 | 18.3 | 25.8 | 37.5 | 73.7 | 61.9 | 46.5 |

**SynLiDAR → SemanticPOSS (XYZI).** Table 4.9 shows that our method achieves an mIoU of 37.9 outperforming other methods, while performance of all methods have dropped compared to Table 4.7 due to dramatically different intensity distribution of both domains.

Table 4.9: Results on SynLiDAR → SemanticPOSS (XYZI).

| Methods | Mech. | bi.clst | car | trunk | veget. | traf. | pole | garb. | build. | cone. | fence | bi.cle | ground | pers. | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Source only | - | 6.8 | 27.8 | 13.3 | 53.5 | 2.0 | 13.8 | 5.9 | 52.5 | 1.9 | 6.9 | 1.3 | 61.2 | 6.8 | 19.5 |
| AdaptSegNet [20] | A | 47.2 | 39.5 | 23.8 | 68.9 | 5.1 | 32.4 | 15.4 | 58.4 | 2.9 | 19.3 | 8.8 | 76.5 | 53.1 | 34.7 |
| CLAN [63] | A | 49.5 | 40.6 | 28.4 | 71.1 | 6.2 | 32.7 | 15.6 | 63.9 | 2.3 | 18.9 | 8.1 | 75.8 | 56.1 | 36.1 |
| AdvEnt [21] | A | 49.1 | 39.5 | 21.0 | 69.8 | 7.0 | 33.0 | 15.8 | 61.8 | 1.9 | 28.5 | 6.2 | 77.8 | 55.0 | 35.9 |
| DAST [64] | A | 45.9 | 38.2 | 22.4 | 69.3 | 3.7 | 23.1 | 13.8 | 61.5 | 3.1 | 14.3 | 7.6 | 73.7 | 48.9 | 32.7 |
| FADA [65] | A | 48.6 | 38.4 | 23.0 | **72.2** | 5.4 | 31.4 | 13.9 | 64.5 | 2.9 | 19.3 | **9.4** | 76.0 | 51.0 | 35.1 |
| MRNet [66] | A | 47.5 | 39.4 | 24.5 | 79.0 | 5.3 | 33.7 | 14.8 | 60.9 | 3.5 | 18.5 | 8.3 | 76.3 | 53.8 | 35.1 |
| PCT [16] | S | 34.8 | 27.8 | 18.6 | 63.7 | 4.9 | **41.0** | 16.6 | 64.1 | 1.6 | 12.1 | 6.6 | 63.9 | 28.9 | 29.6 |
| PolarMix [40] | S | - | - | - | - | - | - | - | - | - | - | - | - | - | 30.4 |
| CoSMix [7] | S | 31.6 | **42.9** | **34.1** | 59.5 | 0.5 | 23.0 | 17.3 | 52.3 | 1.8 | 1.1 | 2.3 | 78.5 | 16.6 | 27.8 |
| PMAN [8] | A | 50.4 | 39.1 | 28.4 | 71.3 | 4.3 | 35.2 | **18.3** | 63.3 | 3.6 | 30.3 | 5.6 | 76.8 | **59.5** | 37.4 |
| Proposed | S | **53.4** | 35.9 | 18.4 | 68.1 | **21.2** | 32.2 | 15.8 | **67.3** | **6.5** | **34.9** | 2.3 | **78.8** | 57.6 | **37.9** |

# Chapter 5

# Discussion

In this chapter, we take a close look at our proposed method's performance by conducting an Ablation Study in Section 5.1, where we break down our approach into its fundamental parts, examining how each component influences the results. Additionally, we consider the Implications and Limitations in Section 5.2 of our work, acknowledging both its strengths and the hurdles it faces in practical applications. Finally, we outline potential research directions for Future Work in Section 5.3.

## 5.1 Ablation Study

Table 5.1 reports the our method's performance on SynLiDAR → SemanticKITTI (XYZI) with certain components ablated. The ablations are categorized into three main sections: Pre-training, Fine-tuning, and CoSMix.

Table 5.1: Ablations.

| **Module** | **Modification** | **mIoU** |
|---|---|---|
| Pre-training | no pre-training | 21.7 |
| | on target domain | 25.6 |
| | on both domains | 27.0 |
| Fine-tuning | Dice loss | 26.3 |
| | Lovasz-Softmax loss | 27.0 |
| | + Segment-wise loss | 28.9 |
| CoSMix | pre-trained model from [7] | 30.5 |
| | our pre-trained model | 33.0 |
| | + AdamW | 34.2 |
| | + Calibration | 34.5 |

In the Pre-training section, we observe how pre-training affects the model's performance. Without any pre-training, the mIoU is 21.7. When pre-training is done on the target domain alone, the performance improves (+ 3.9%) to an mIoU of 25.6. Leveraging pre-training on both domains further increases mIoU (+ 1.4%) to 27.0.

Moving to the Fine-tuning section, we explore different loss functions and observe their impact. When using Dice loss, the mIoU is 26.3, and it increases (+ 0.7%) to 27.0 with the use of the Lovasz-Softmax loss. The addition of the segment-wise loss $\mathcal{L}_{segment}$ as in Eq. 3.11 leads to a notable performance boost (+ 1.9%), reaching an mIoU of 28.9.

Lastly, using the original pre-trained model yields an mIoU of 30.5. However, our pre-trained model surpasses this performance by + 2.5%, reaching an mIoU of 33.0. Further enhancements are achieved by

switching to the AdamW optimizer, the same optimizer used in previous training stages, resulting in an mIoU of 34.2, improving by (+ 1.2%). Finally, when calibration is employ we obtain an mIoU of 34.5.

**Intensity normalization.** The impact of utilizing intensity as an input feature in the context of the SynLiDAR → SemanticPOSS task is evident from the results presented in Table 4.7 and Table 4.9. In both tables, it is clear that the inclusion of intensity values has noticeably decreased the performance of the models. This decline in performance suggests that intensity values introduce further domain discrepancy.

Moreover, the results in Table 5.2 shows that when intensity normalization is applied, a significant improvement of + 3.4% in the final performance is observed. These findings underscore the importance of carefully considering the inclusion of intensity as an input feature in cross-domain tasks.

Table 5.2: Effect on intensity normalization on SynLiDAR → SemanticPOSS (XYZI).

| | **no** intensity | **with** intensity normalization | **w/o** intensity normalization |
|---|---|---|---|
| mIoU | 42.8 | 37.4 | 34.0 |

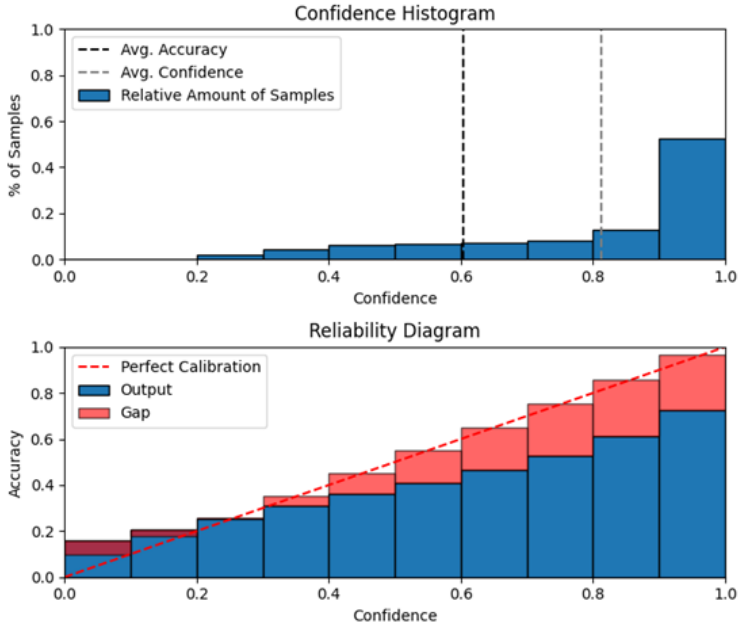


Figure 5.1: Confidence histogram and reliability diagram for **uncalibrated** teacher.

**Calibration.** We generated confidence and reliability diagrams for both uncalibrated and calibrated models, depicted in Fig 5.1 and Fig 5.2 respectively. Notably, calibration plays a crucial role in addressing the issue of model overconfidence. It reduces the average confidence value from approximately 0.8 to around 0.6. Moreover, we observe that calibration has the effect of moving the model's reliability curve towards the diagonal line. As a result, the model's training becomes less sensitive to the choice of confidence threshold. This enables us to cancel this hyperparameter, opting instead to use the calibrated average confidence as the threshold for selecting pseudo-labels in our approach. This not only simplifies our method but also enhances its robustness.

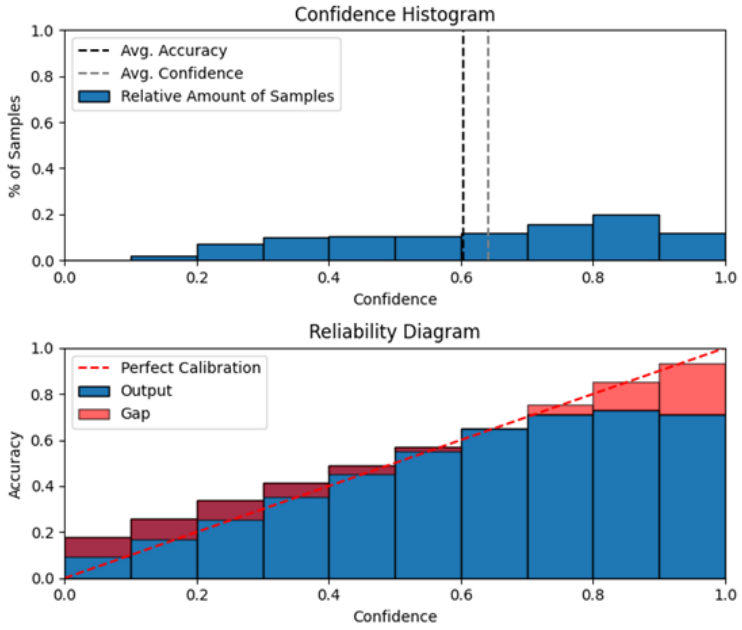


Figure 5.2: Confidence histogram and reliability diagram for **calibrated** teacher.

## 5.2 Implications and Limitations

### 5.2.1 Implications

Our proposed method exhibits improved performance on two UDA benchmarks, making it practical for various LiDAR-based applications.

One of the notable strengths of our approach is its robustness in handling LiDAR intensity values. Unlike [8], which requires additional effort to leverage intensity data, our method shows resilience, regardless of whether intensity data is provided as input, as indicated in Table 4.7 and Table 4.9. This robustness enhances its practicality in various LiDAR-based tasks.

Furthermore, our results suggest that the proposed method outperforms other methods in specific classes that are relatively easier to cluster using HDBSCAN [47], such as cars, trucks, motorcycles, and buildings as shown in Table 4.5. This highlights its competence in handling common objects encountered in LiDAR-based perception tasks.

In the Pre-training section of Table 5.1, we have demonstrated that unsupervised learning can facilitate the acquisition of domain-invariant features in both source and target domains. In the Fine-tuning section, the use of the same optimizer in both pretraining and fine-tuning stages consistently yields improved results. Additionally, the introduction of constraints that mimic the pretraining objectives into the fine-tuning stage enhances feature preservation, further contributing to enhanced performance.

In CoSMix, we cancel the the pseudo-label confidence threshold hyperparameter by directly using the calibrated average confidence. While calibration itself does not directly impact the model's accuracy, the observed accuracy gain of +0.3% in Table 5.1 can be attributed to the improved selection of the confidence threshold achieved through the calibration process. This underscores the significance of calibration in stabilizing the training process.

### 5.2.2 Limitations

Our approach does come with the trade-off of increased training time and computational resources in exchange for enhanced performance. Additionally, it involves a multi-stage training process, with each stage having its own set of hyperparameters, increasing complexity compared to single-stage training methods.

Despite the simplicity and effectiveness of intensity normalization in mitigating domain discrepancies, it is unable to fully align the data distributions between source and target domains. This limitation may require further exploration of domain adaptation techniques to achieve stronger distribution alignment, so that intensity information can be more effectively used for domain adaptation without widening the domain gap.

While CoSMix demonstrates significant performance improvements in terms of evaluation metrics, it is essential to acknowledge its inherent unexplainability. The synthetic scenes created by CoSMix may not faithfully replicate real-world environments, making it less suitable for safety-intensive applications like autonomous driving, where interpretability and trustworthiness are paramount. Future research should consider addressing this limitation to make CoSMix more suitable for such critical domains.

In conclusion, our method showcases promising implications for LiDAR-based applications, offering adaptability, robustness, and competitive performance. However, it is crucial to address the identified limitations to ensure its applicability and reliability in real-world, safety-critical scenarios.

## 5.3 Future Work

In future research, we intend to explore synergies between our method and other clustering techniques to generate more semantically meaningful segments. Additionally, we plan to delve deeper into unsupervised learning techniques, possibly incorporating concepts like GrowSP [67] to expand the size of segments or group segments into classes progressively.

Another crucial aspect of our future research is reducing the gap between UDA and fully supervised learning. Despite the noticeable performance gains of our UDA pipeline compared to previous methods, there remains a significant performance gap between UDA and supervised learning. This observation points us towards two directions for future research: Weakly Supervised Learning and Uncertainty-Aware Domain Adaptation. Weakly Supervised Learning allows to label only valuable data, enhancing model adaptation results while reducing annotation costs. On the other hand, Uncertainty-Aware Domain Adaptation aims to mitigate risks associated with unreliable predictions. This is particularly crucial for safety-intensive real-world applications like autonomous driving, where mistakes are not tolerated.

# Chapter 6

# Conclusion

In this thesis, we present a 3-stage pipeline designed to address the challenge posed by unsupervised domain adaptation (UDA) for synthetic-to-real 3D Lidar semantic segmentation. The first stage of our method leverages the capabilities of contrastive learning to facilitate the learning of domain-invariant representations from both synthetic and real-world LiDAR data. Our second stage involves transfer learning, where we fine-tune the pre-trained model using labeled data from the source domain. Within this stage, we have devised a fine-tuning strategy to effectively preserve domain-invariant features. In our third stage, we present calibrated CoSMix, a refinement of an existing 3D UDA technique. This stage incorporates a calibration technique to counteract the model's overconfidence, thereby enhancing stability in the self-training process. Data from both domains are merged to create augmented mixed domains, progressively enhancing the model's generalization capabilities. These three stages together form a comprehensive solution to the UDA challenge in 3D LiDAR semantic segmentation. Our experimental results on two benchmarks indicate that the proposed method is on par with the current state-of-the-art. As exemplified in UDA task from SynLiDAR to SemanticKITTI with xyz coordinates as input, our method outperforms previous ones by achieving an mIoU of 34.4%, with the second-best method and the source-only method achieving 33.7% and 20.4%, respectively. While the proposed method shows competitive performance in UDA benchmarks, we acknowledge the increased training time and complexity of our approach. We also recognize the lack of explainability of CoSMix-generated-scenes, which may draw concerns when applied to safety-intensive applications. This also motivates us to reduce the performance gap between UDA and fully supervised learning, and to explore uncertainty-aware approaches to mitigate risks.

# Bibliography


[1] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Perez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] Guangrui Li, Guoliang Kang, Xiaohan Wang, Yunchao Wei, and Yi Yang. Adversarially masking synthetic to mimic real: Adaptive noise injection for point cloud segmentation adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20464–20474, June 2023.

[3] Aoran Xiao, Xiaoqin Zhang, Ling Shao, and Shijian Lu. A survey of label-efficient deep learning for 3d point clouds, 2023.

[4] Christopher B Choy. *High-Dimensional Convolutional Neural Networks for 3D Perception*. Stanford University, 2020.

[5] Hyungtae Lim, Oh Minho, and Hyun Myung. Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor. *IEEE Robotics and Automation Letters*, 2021.

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017.

[7] Cristiano Saltori, Fabio Galasso, Giuseppe Fiameni, Nicu Sebe, Elisa Ricci, and Fabio Poiesi. Cosmix: Compositional semantic mix for domain adaptation in 3d lidar segmentation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 586–602, Cham, 2022. Springer Nature Switzerland.

[8] Zhimin Yuan, Ming Cheng, Wankang Zeng, Yanfei Su, Weiquan Liu, Shangshu Yu, and Cheng Wang. Prototype-guided multitask adversarial network for cross-domain lidar point clouds semantic segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–13, 2023.

[9] Alok Jhaldiyal and Navendu Chaudhary. Semantic segmentation of 3d lidar data using deep learning: a review of projection-based methods. *Applied Intelligence*, 53(6):6844–6855, 2023.

[10] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[11] Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[12] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15358–15368, 2021.

[13] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Gläser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2021.

[14] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4):3500–3509, May 2021.

[15] Runyu Ding, Jihan Yang, Li Jiang, and Xiaojuan Qi. Doda: Data-oriented sim-to-real domain adaptation for 3d semantic segmentation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 284–303, Cham, 2022. Springer Nature Switzerland.

[16] Muhammad Imad, Oualid Doukhi, and Deok-Jin Lee. Transfer learning based semantic segmentation for 3d object detection from point cloud. *Sensors*, 21(12), 2021.

[17] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529, 2019.

[18] Jean-Emmanuel Deschaud, David Duque, Jean Pierre Richa, Santiago Velasco-Forero, Beatriz Marcotegui, and François Goulette. Paris-carla-3d: A real and synthetic outdoor point cloud dataset for challenging tasks in 3d mapping. *Remote Sensing*, 13(22):4713, 2021.

[19] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019.

[20] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Perez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] Yuefan Shen, Yanchao Yang, Mi Yan, He Wang, Youyi Zheng, and Leonidas J. Guibas. Domain adaptation on point clouds via geometry-aware implicits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7223–7232, June 2022.

[23] Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Segcontrast: 3d point cloud feature representation learning through self-supervised segment discrimination. *IEEE Robotics and Automation Letters*, 7(2):2116–2123, 2022.

[24] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. 2022.

[25] Lucas Nunes, Louis Wiesmann, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Temporal consistent 3d lidar representation learning for semantic perception in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5217–5228, June 2023.

[26] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19313–19322, June 2022.

[27] Laurenz Reichardt, Nikolas Ebert, and Oliver Wasenmüller. 360° from a single camera: A few-shot approach for lidar segmentation, 2023.

[28] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[30] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[31] 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.

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

[33] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 228–240, 2023.

[34] Ben Fei, Weidong Yang, Liwen Liu, Tianyue Luo, Rui Zhang, Yixuan Li, and Ying He. Self-supervised learning for pre-training 3d point clouds: A survey, 2023.

[35] Yabin Zhang, Jiehong Lin, Chenhang He, Yongwei Chen, Kui Jia, and Lei Zhang. Masked surfel prediction for self-supervised point cloud learning, 2022.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[37] 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, 2020.

[38] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2457–2464, 2021.

[39] Bjoern Michele, Alexandre Boulch, Gilles Puy, Tuan-Hung Vu, Renaud Marlet, and Nicolas Courty. Saluda: Surface-based automotive lidar unsupervised domain adaptation, 2023.

[40] Aoran Xiao, Jiaxing Huang, Dayan Guan, Kaiwen Cui, Shijian Lu, and Ling Shao. Polarmix: A general data augmentation technique for lidar point clouds. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11035–11048. Curran Associates, Inc., 2022.

[41] Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art, 2023.

[42] Ha Manh Bui and Anqi Liu. Density-softmax: Scalable and calibrated uncertainty estimation under distribution shifts, 2023.

[43] Ximei Wang, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable calibration with lower bias and variance in domain adaptation. In *Advances in Neural Information Processing Systems 33*, 2020.

[44] Haohan Wang, Liang Liu, Boshen Zhang, Jiangning Zhang, Wuhao Zhang, Zhenye Gan, Yabiao Wang, Chengjie Wang, and Haoqian Wang. Calibrated teacher for sparsely annotated object detection, 2023.

[45] Dapeng Hu, Jian Liang, Xinchao Wang, and Chuan-Sheng Foo. Pseudocal: A source-free approach to unsupervised uncertainty calibration in domain adaptation, 2023.

[46] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[47] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[48] Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32(1/2):12–22, 1983.

[49]

[50] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[51] 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 (CVPR)*, June 2018.

[52] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[53] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. Semanticposs: A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693, 2020.

[54] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[55] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[56] 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.

[57] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[58] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, et al. One million scenes for autonomous driving: Once dataset. *NeurIPS*, 2021.

[59] Aoran Xiao, Jiaxing Huang, Weihao Xuan, Ruijie Ren, Kangcheng Liu, Dayan Guan, Abdulmotaleb El Saddik, Shijian Lu, and Eric P. Xing. 3d semantic segmentation in the wild: Learning generalized models for adverse-condition point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9382–9392, June 2023.

[60] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[61] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019.

[62] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[63] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[64] Fei Yu, Mo Zhang, Hexin Dong, Sheng Hu, Bin Dong, and Li Zhang. Dast: Unsupervised domain adaptation in semantic segmentation based on discriminator attention and self-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10754–10762, May 2021.

[65] Haoran Wang, Tong Shen, Wei Zhang, Ling-Yu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In Andrea Vedaldi, Horst Bischof, Thomas

Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 642–659, Cham, 2020. Springer International Publishing.

[66] Zhedong Zheng and Yi Yang. Unsupervised scene adaptation with memory regularization in vivo. In *IJCAI*, 2020.

[67] Zihui Zhang, Bo Yang, Bing Wang, and Bo Li. Growsp: Unsupervised semantic segmentation of 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17619–17629, June 2023.