

Improving Multimodal Distillation for 3D Semantic Segmentation under Domain Shift

Björn Michele^{1,2}
 bjoern.michele@univ-ubs.com

xandre Boulch¹
 xandre.boulch@valeo.com

les Puy¹
 s.puy@valeo.com

n-Hung Vu¹
 n-hung.vu@valeo.com

naud Marlet^{1,3}
 aud.marlet@valeo.com

olas Courty²
 las.courty@univ-ubs.com

¹ valeo.ai,
 Paris, France

² CNRS, IRISA, Univ. Bretagne Sud,
 Vannes, France

³ LIGM, Ecole des Ponts, Univ Gustave
 Eiffel, CNRS,
 Marne-la-Vallée, France

Abstract

Semantic segmentation networks trained under full supervision for one type of lidar fail to generalize to unseen lidars without intervention. To reduce the performance gap under domain shifts, a recent trend is to leverage vision foundation models (VFM) providing robust features across domains. In this work, we conduct an exhaustive study to identify recipes for exploiting VFMs in unsupervised domain adaptation for semantic segmentation of lidar point clouds. Building upon unsupervised image-to-lidar knowledge distillation, our study reveals that: (1) the architecture of the lidar backbone is key to maximize the generalization performance on a target domain; (2) it is possible to pretrain a single backbone once and for all, and use it to address many domain shifts; (3) best results are obtained by keeping the pretrained backbone frozen and training an MLP head for semantic segmentation. The resulting pipeline achieves state-of-the-art results in four widely-recognized and challenging settings. The code will be available at: github.com/valeoai/muddos.

Introduction

Understanding scenes at 3D level is key for applications like autonomous driving or robotics. In particular, the semantic segmentation of lidar scans is valuable high-level information that autonomous vehicles can rely upon, e.g., for trajectory planning. However, state-of-the-art networks for semantic segmentation require a large amount of costly annotated training data to achieve good performance, limiting their deployment in new environments or when changing sensors. Domain adaptation (DA) addresses this problem by adapting a network trained on a labeled source domain to a new target domain. Unsupervised domain adaptation (UDA), in particular, conducts this adaptation without using any label on the target domain.

Vision foundation models (VFMs), trained on web-scale datasets, provide features that can be used off-the-shelf for a wide variety of tasks, and are robust to strong domain shifts [63, 52]. In addition, recent works like [25, 22, 40, 48, 57] show that knowledge distillation of VFMs into 3D backbones can provide robust 3D features. In this work, we build upon these distillation methods and conduct an exhaustive study to identify key recipes for exploiting VFMs for 3D UDA on autonomous driving datasets in the best possible way. Notably, our study identifies practices that allow us to outperform by more than 15 mIoU points the previous state of the art in multimodal unsupervised domain adaptation for 3D lidar semantic segmentation.

We structure our study along four axes. First, we analyze the effect of different architectural choices in the 3D backbone to improve its robustness to domain gaps. Second, we benchmark different VFMs to identify the most appropriate for our task. Third, we evaluate different downstream training recipes on source datasets to get the best models on target datasets. Finally, we study how the pretraining datasets used for distillation influence performance under domain shifts.

Our insights are listed below.

- *Backbone.* While we are able to surpass competing methods with MinkowskiUNet [44] (MUNet), which is the default choice of backbone in the DA literature, we advocate the use of more recent networks, which compete with the best adapted MUNet without even leveraging any domain adaptation technique. Further improvements can be achieved by scaling the capacity of the backbone. We also notice that the choice of normalization layers has a high impact on the generalization capabilities of the lidar backbone, where layernorms perform better than batchnorms. Besides, we observe that the use of lidar intensity as input feature is most often detrimental for adaptation.
- *Pretraining by distillation.* We note that ViTs pretrained with DINOv2 provide more robust features than those provided by SAM [65]. Moreover, we show that distillation can be performed *once for all* on a combination of multiple datasets. This contrasts with existing techniques where a new full training is done for each new pair of source and target datasets.
- *Downstream training.* Generalization capabilities are preserved when the backbone is kept frozen after distillation and a classification head is trained for the downstream task of semantic segmentation. Better results are obtained when the distillation and semantic segmentation training are done consecutively, rather than jointly.

The best technique resulting from this study obtains SOTA results with large margins. We call it MuDDoS, for **M**ultimodal **D**istillation for 3D Semantic Segmentation under **D**omain **S**hifts. An overview of the pipeline can be seen in Fig. 1.

2 Related work

Monomodal UDA for semantic segmentation. Taking inspiration from the problem of UDA for images, several works adapted these techniques to lidar scenes such as adversarial training via projection of the points to image-like representations [0, 42, 19, 22, 63], mixing source and target data [21, 43], or using both adversarial training and mixing strategies [69]. Geometric methods target the specificity of the domain gap induced by different sensors, e.g., variable acquisition patterns or location on the vehicle. For example, the source point clouds can be up/down-sampled to resemble the target point clouds [50], possibly with self-ensembling [49]. Along the same line, a reconstruction of the underlying sensor-agnostic

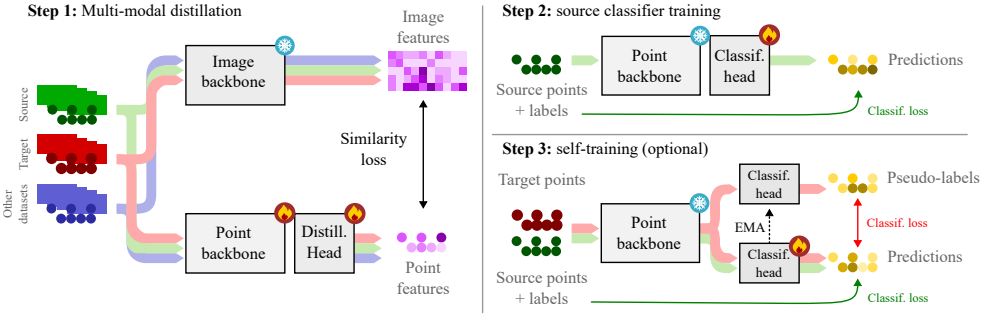


Figure 1: Overview of the multimodal distillation pipeline for 3D domain adaptation. With MuDDoS, adapting from an annotated source dataset to an unannotated target dataset, operating in three steps. **Step 1** is a 2D-to-3D distillation using a frozen visual foundation model (DINOv2) to obtain aligned 3D representations on all datasets. **Step 2** trains a classification head with source labels. The backbone is frozen to prevent the 3D representations from drifting away and to maintain a good performance on the target dataset. **Step 3** is a prediction refinement using self-training obtained via a classical teacher-student scheme.

surface can be used as a unified representation space [68] or as a regularization loss [29]. In addition to the vanilla UDA settings, several others have studied source-free UDA [80], online-adaptation [42] and generalization [18, 20, 65, 45, 46, 47, 62].

Multimodal UDA for semantic segmentation. Automotive datasets are usually acquired with several synchronized sensors, including lidars and cameras [8, 13, 12, 15, 65]. Using the sensor calibration and their relative position, points can be projected in the accompanying images. Thus, it makes it possible to do a multimodal domain adaptation that leverages the image as well as the lidar modalities.

Training the image features extractor. The pioneering work of xMUDA [16, 17] introduced the task of multimodal domain adaptation for semantic segmentation using both lidar and image data. Their approach involved training separate classifiers for each modality — one for images and one for point clouds — while enforcing consistency through a KL-divergence loss between the semantic predictions of corresponding lidar points and image pixels. At inference, the best performance was achieved by fusing the predictions of both classifiers, leveraging the complementarity of 2D and 3D data. Though effective, this approach requires precisely calibrated lidar-camera data at test time and limits predictions to regions visible in both modalities. For example, in SemanticKITTI [8, 15] only a front-view image is available, in Waymo [13] the rear part of the lidar scan is not covered by the images, and in nuScenes [8], even though using a complete ring of cameras, only 48% of the points have a reprojection in 2D [54]. Several works have built upon xMUDA to enhance multi-modal domain adaptation [0, 24, 56, 60]. Furthermore, multimodal information has been leveraged for test-time adaptation [60, 53].

Using a vision foundation model. VFMs are trained with distinct objectives, resulting in unique characteristics. SAM [63] generates high-quality masks of objects instances in image space. DINOv2 [63] focuses on extracting high-quality semantically-coherent features. In multimodal domain adaptation, methods leverage these models’ specific strengths. Several works [6, 63, 56] rely on the ability of SAM to extract object instances. Adapt-SAM [63] uses it to produce instance masks in each domain, selecting some instances in each

domain, and adding these instances in the point cloud of the other domain. An image-to-lidar distillation loss using SAM features is also used while training the lidar backbone for semantic segmentation. This approach is the closest to ours as it leverages image-to-lidar distillation during training and performs inference using only the lidar data and predicts semantic labels for the entire 360° point cloud (and not only for points with an image projection).

Image-to-lidar distillation. Our method leverages image-to-lidar distillation techniques. Among such techniques, a first set of techniques distill the knowledge of vision language models such as CLIP [41] to make open-vocabulary tasks on lidar data possible [9, 37, 61, 64]. Another set of techniques uses VFM as teacher such as [8, 63] to pretrain lidar backbones without supervision [26, 27, 28, 48, 57]. This pretraining stage permits to reach better performance on downstream tasks such as semantic segmentation or object detection. In this paper, we build upon ScaLR [40] which detailed several techniques to improve knowledge distillation.

3 Study: How to get the best out of VFMs for 3D UDA?

Experimental setup. We conduct our study using three datasets: nuScenes [12] (N), SemanticKITTI [9, 15] (K) and Waymo Open Dataset [13] (W). The pairs of source and target datasets considered, which are common in the literature, are: $N \rightarrow K$, $K \rightarrow N$, $N \rightarrow W$ and $W \rightarrow N$. A different lidar sensor is used in each dataset. In nuScenes, the lidar used has 32 beams, vs. 64 beams in SemanticKITTI and Waymo. All 3 sensors also have a different horizontal resolution. These variations make the settings especially difficult. The precise class mappings between the source and target datasets are provided in the supplementary material, as well as the training protocols (augmentation, losses and optim. parameters). See supp. mat. B and A.

3.1 Choice of lidar backbone architecture

A common choice of backbone in the lidar UDA literature is MinkowskiUNet [10] (MUNet). We show that this default choice actually limits our ability to reach high performance. We advocate for the use of other architectures that are more robust to domain shifts.

We study different choices of lidar backbone architecture. The architectures considered are MinkowskiUNet [10] (MUNet) and WaffleIron [39] (WI, with 256 and 768 channel variants). We also use the MUNet from [38, 43] in Tab. 6. In this first part of the study, all backbones are trained from scratch for semantic segmentation on source datasets *without any pretraining phase*. After training, the backbone is directly tested on target datasets. We monitor the effect of: (a) using the intensity as input feature; (b) changing batchnorms to layernorms; (c) increasing the capacity of the backbones. The results are presented in Tab. 1.

Effect of intensity. We notice in Tab. 1 that using intensity as input feature is detrimental for MUNet with both batchnorms or layernorms, and for WI-256 with batchnorms. This result is in line with the conclusions drawn in [46]. For WaffleIron with layernorms, the outcome is dependent on the setting: removing the intensity leads to a large gain on $N \rightarrow K$, a drop on $N \rightarrow W$, and no effect in the two other settings. The average mIoU over all dataset pairs is nevertheless better when intensity is not used. In the rest of the paper, we do not use intensity as input feature as this is the most robust strategy overall.

Effect of normalization layers. Due to sensor differences, the distribution of points in each dataset widely differs. It impacts feature distribution when going from the source to the

Backbone	Int.	Norm	N+K	K+N	N+W	W+N
MUNet	✓	BN	30.1	22.0	28.3	30.9
	✗	BN	32.3	49.9	31.9	46.2
	✓	LN	35.9	21.6	27.9	31.5
	✗	LN	38.1	49.5	33.1	44.7
WI-256	✓	BN	16.0	48.0	20.2	59.7
	✗	BN	33.8	52.9	19.9	60.1
	✓	LN	22.2	50.6	40.9	63.2
	✗	LN	41.1	49.8	35.3	61.8
WI-768	✗	LN	44.6	55.1	37.1	64.6
<i>State of the art in lidar multi-modal UDA.</i>						
MUNet w. [63]	✓	BN	48.5	42.9	44.9	48.2

Table 1: Generalization capability of different backbones and architecture choices, intensity (Int.) or normalization layer (BN, LN). Trained on source, evaluated on target.

Backbone	VFM	N+K	K+N
WI-256	SAM ViT-L	42.5	48.6
	DiNOv2 ViT-L	46.6	59.7

Table 2: Effect of the VFM used for feature distillation with ScaLR.

Downstream training	Intensity	N+K	K+N
Frozen Back. + Linear (a)	✗	43.4	<u>60.1</u>
Frozen Back. + MLP (b)	✗	47.5	62.1
Frozen Back. + MLP (b)	✓	36.9	58.7
Full finetuning (c)	✗	50.5	58.9
Joint distill. & classif. as in [63]	✗	44.9	56.0

Table 3: Downstream training recipe using WI-768 backbone.

target dataset, altering target segmentation performance. When using batch normalization layers, one way to address these changes, at least partially, is to adapt the batchnorm statistics to the target dataset [23, 52]. We study here an alternative: replacing all batchnorms (BNs) with layernorms (LNs). Averaging the mIoUs over all dataset pairs in Tab. 1, we notice that layernorms improve the performance for both backbones considered: +1.4 mIoU pts for MUNet, +5.3 for WI-256. Unlike batchnorms that use training feature statistics at inference, layernorms center and normalize the features using the actually seen statistics at inference. To maximize performance, we use layernorms in the rest of the paper.

Effect of backbone and capacity. WI-256 performs better than MUNet in all four pairs of datasets (Tab. 1). For WaffleIron, we study the effect of increasing its feature size from 256 to 768. While increasing the number of parameters could have led to overfitting to the source dataset, we notice (Tab. 1) that WI-768 actually generalizes better and surpasses WI-256.

Interestingly, WI-768 trained on source without adaptation outperforms the current SOTA in two settings out of four. This advocates for more studies in UDA on architectural choices.

3.2 Choice of a visual foundation model for pretraining

For the remaining part of our study, the backbones are pretrained by distilling a VFM. The choice of VFM is key to get a good performance for downstream semantic segmentation. Several VFMs are studied in [40] and the best results are obtained with DiNOv2 [53]. SAM [65] is another powerful model also leveraged in, e.g., [58] for robust point cloud segmentation across domains. We test whether SAM is a better choice.

Pretraining protocol. The lidar backbones are pretrained using ScaLR [40]. This VFM-based pretraining method requires calibrated and synchronized cameras and lidars to establish correspondences between points and pixels. It does not need manual annotations. The training loss, optimization parameters and augmentations are described in the supp. mat. A.

SAM vs DiNOv2. We distill the knowledge from SAM ViT-L and DiNOv2 ViT-L into WI-256 on nuScenes and SemanticKITTI jointly (the datasets are merged). Then, we freeze the weights of the pretrained backbone and train an MLP classification head on nuScenes and test the performance of the overall network on SemanticKITTI (and vice versa). We present

the results in Tab. 2. We notice a better performance after distillation of DINOv2, showing that the features of DINOv2 are more suited for semantic understanding than those of SAM. For images, let us mention that a study of the properties of DINOv2 and SAM features was, e.g., conducted in [42]. The results showed that SAM performs worse than DINOv2 for “high-level object description” and for “combining the semantics of multiple objects.”. Our results indicate that this conclusion remains valid after distillation into lidar backbones.

3.3 Downstream training

After pretraining, the downstream training recipe is also important to get competitive results. We should make sure to exploit at best the features distilled from the chosen VFM and avoid scenarios where the backbone degenerates to the source-only performance.

Downstream training recipes. We study three different ways for downstream training on semantic segmentation. (a) We keep the lidar backbone weights frozen and train a linear classification head, with a batchnorm followed directly by a linear layer (note that these two layers can be combined into a single linear layer at inference). (b) We keep the lidar backbone weights frozen and train a 2-layer MLP with ReLU activation. (c) We finetune the backbone weights along with the linear classification head (full finetuning). Implementation details, such as epoch number and learning rates, are described in the supplementary material A.

MLP classification head performs the best. We present in Tab. 3 the results obtained with the three considered recipes. First, we notice that training only a linear classification head is underperforming compared to the other alternatives. Second, training an MLP classification head leads to similar results on average than finetuning the whole backbone. Yet, training an MLP is simpler and faster: full finetuning needs a careful choice of the learning rates applied on the pretrained backbones and on the classification head, and requires a full backward pass in the backbone. We also notice that the performance on the target dataset, after a first phase of improvement, keeps decreasing with full finetuning. Instead, when using a simple MLP head, the performance stabilizes both on the source and target datasets. This makes it an easy and practical method to use, as one just need to monitor the performance on the source dataset to stop the training.

Effect of intensity when pretraining. Again in Tab. 3, one can notice that the use of the intensity as input feature remains detrimental, even when pretraining on the source and target datasets and training a simple MLP head. This result stays in line with our design choice made in Sec. 3.1, to disregard intensity when having to change domain.

Joint distillation and semantic segmentation. Instead of first pretraining the source and target datasets, and then finetuning the backbone for semantic segmentation on the source dataset, one can naturally wonder if optimizing for both tasks at the same time (distillation + semantic segmentation), as done in [58], is a better choice or not.

We test this design as follows. Two different heads are added at the end of the lidar backbone: one for distillation and one for classification. The backbone sees both source and target point clouds. The distillation loss is computed for both source and target point clouds. The classification loss (cross-entropy plus Lovász loss) is applied on source point clouds. We present the result of this strategy in the last row of Tab. 3. We observe that the strategy is the worst on $K \rightarrow N$ and is the second worse on $N \rightarrow K$.

Backbone	Pretrain			Frozen backbone	N→K	K→N	N→W	W→N
	S	T	E					
WI-256	✗	✗	✗	✗	41.1	49.8	35.3	61.8
	✗	✗	✗	✓	31.0	39.0	30.2	32.8
	✓	✓	✗	✓	46.6	59.7	61.3	59.7
WI-768	✗	✗	✗	✗	44.6	55.1	37.1	64.6
	✓	✓	✗	✓	47.5	62.1	66.1	70.5
	✓	✓	✓	✓	52.1	63.4	67.1	71.8

*Results of ScaLR [40] with intensity. *w/o Waymo.*

WI-768	✗	✗	✗	✗	28.6	-	-	-
	✓	✗	✗	✗	29.6	-	-	-
	✓	✓	✓*	✗	36.6	-	-	-

Table 4: **Adaptation performance** depending on the backbone, pretraining datasets (S, T, E), and backbone freezing status.

3.4 Choice of pretraining datasets

Finally, we study the choice of pretraining datasets. A few results in [40] suggest that the generalization capabilities of the lidar backbone improves when combining multiple datasets for pretraining. In this section, we analyze this behavior much more thoroughly by testing exhaustively different dataset combinations. We specialize this study to the best setup we have constructed so far for UDA.

Pretraining dataset configurations. We study three different settings (in Tab. 4).

- *Source pretraining (S)*: it corresponds to a source-only setting.
- *Source & Target pretraining (S+T)*: pretraining is done on both source and target.
- *Source & Target & External datasets pretraining (S+T+E)*: pretraining is done combining nuScenes, SemanticKITTI and Waymo, to which we also add PandaSet [54].

For each experiment, we pretrain the backbone using ScaLR, the weights of the backbone are then frozen, and the classification head (a 2-layer MLP) is trained using the source labels.

Impact of pretraining on both source and target. First, we observe that the results obtained for WI-256 with source pretraining and downstream training of the classification head are worse than those obtained when training WI-256 from scratch on source datasets. The quality of the features after source pretraining is not good enough to reach a high performance on the target datasets using only an MLP classification head. Nevertheless, as soon as the network is pretrained on both the source and target datasets, tuning the classification head on the source datasets is enough to surpass the results obtained in Tab. 1 in 3 out of 4 settings. This demonstrate that pretraining on Source & Target produces features: (a) that are well aligned between the source and target domain, and (b) that can be used as is by just training an MLP on top of them. As previously, we still observe better results with WI-768 vs WI-256.

Impact of pretraining on more datasets. Recent works [40, 54] highlight that training jointly with multiple datasets can improve feature quality. We show in Tab. 4 that this observation also holds for domain adaptation. Indeed, the performance of WI-768 improves by at least 1.0 mIoU point and up to 4.6 points when pretrained on the mix of all datasets, compared to pretraining only on the source and target datasets.

From a practical point of view, besides obtaining better performances, this result shows that one can pretrain a *single backbone* to address *multiple domain* shifts, rather than one

Backbone	Pretrain		Self-train	N→K	K→N	N→W	W→N
	S+T	S+T+E					
WI-768	-	-	✗	44.6	55.1	37.1	64.6
	-	-	✓	45.7	55.6	39.0	62.9
	✓	-	✗	47.5	62.1	66.1	70.5
	✓	-	✓	49.5	66.5	69.8	69.6
	-	✓	✗	52.1	63.4	67.1	71.8
	-	✓	✓	52.1	66.4	69.1	70.5

Table 5: **Effect of self-training.** We present the consistent benefit of self-training whether the backbone is not pretrained, pretrained on both the source and target dataset (S+T), or pretrained on all the considered datasets (S+T+E).

Method	Back.	Mod.	N+K	K+N	N+W	W+N	Avg.
Target oracle	MUNet	-	70.3	78.3	79.9	78.3	76.7
Target oracle	WI-768	-	72.4	83.8	83.4	83.8	80.9
Source only [†]	MUNet	-	27.7	28.1	29.4	21.8	26.8
Source only	WI-768	-	44.6	55.1	37.1	64.6	50.4
PL [†] [63]	MUNet	U	30.0	29.0	31.9	22.3	28.3
CosMix [†] [63]	MUNet	U	30.6	29.7	31.5	30.0	30.5
MM2D3D [†] [6]	MUNet	M	30.4	31.9	31.3	33.5	31.8
MM2D3D* [†] [6]	MUNet	M	32.9	33.7	34.1	37.5	34.6
Adapt-SAM [†] [68]	MUNet	M	48.5	42.9	44.9	48.2	46.1
MuDDoS (ours)	MUNet	M	41.8	53.3	54.1	53.0	50.6
Self-Training	WI-768	U	45.7	55.6	27.6	48.0	44.2
MuDDoS (ours)	WI-768	M	52.1	66.4	69.1	70.5	64.5

(a) **Comparison to existing approaches.** The results for methods marked with [†] are reported from [63]. The current absence of code for [68] prevents us from testing it with our architecture choices.

	Adapt-SAM [68]	MuDDoS (ours)
Distillation	Joint	Sequential
Ext. Data	✗	✓
Finetuning	Full	Head
VFM	SAM	DINOv2
Backbone	MUNet	WI-768
Scaled backbone	✗	✓
Intensity	✓	✗
Norm	BN	LN

(b) **Main differences** with Adapt-SAM, previous SOTA.

Table 6: Comparison to the state of the art. (a) We compare quantitatively our best setting (MuDDoS) to both unimodal (U) and multimodal (M) UDA methods for 3D semantic segmentation of lidar point clouds. (b) We highlight the main changes introduced by our study with respect to the previous state-of-the-art method, i.e., Adapt-SAM [68].

pretraining for each pair of source and target datasets. In fact, it is possible to pretrain a single backbone that addresses all UDA settings together, and only train one classification head per source dataset. This linear number of trainings contrasts with methods in which the network has to be retrained for each dataset pair, thus with a quadratic number of trainings. Besides, training only a 2-layer MLP in our case is more efficient than training a complete network.

We also remark that our results are much better than those in [40] on N+K despite similar pretrainings. It highlights the importance of our architectural choices (no use of intensity) and downstream training recipe (training a simple MLP head instead of full finetuning).

3.5 Self-training for further improvements

A classical recipe to improve results is to do a final round of self-training using a teacher-student mechanism. We complement our study by checking if this stage is also beneficial in our case. We train only the MLP head and keep the backbone frozen. We present results in Tab. 5. We notice that it improves the performance in all cases for WI-768, except for W+N where we observe a slight decrease. See supplementary material A for training details.

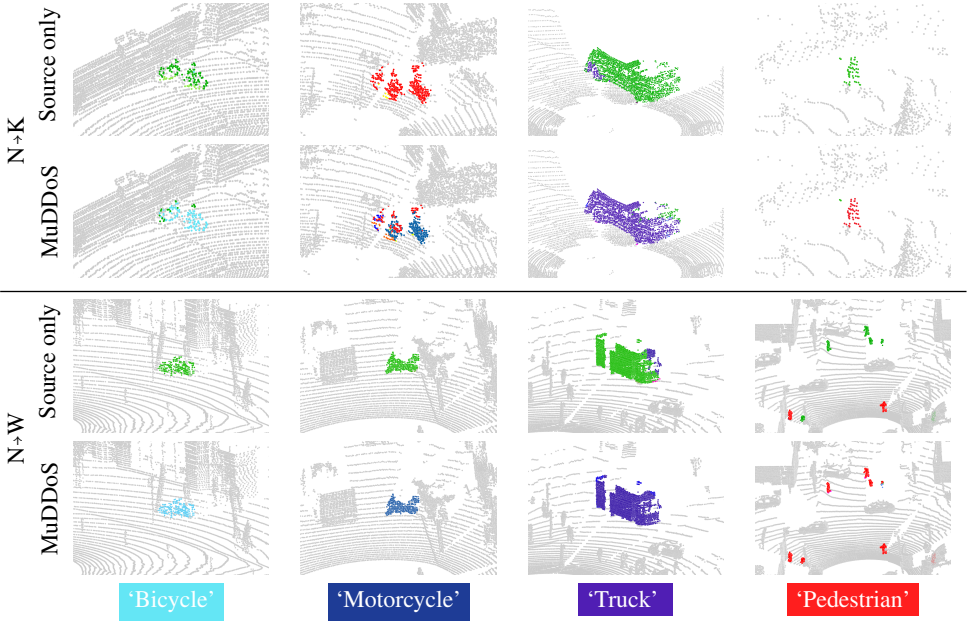


Figure 2: **Qualitative results on $N \rightarrow K$ (Top) and $N \rightarrow W$ (Bottom).** The label colors correspond to ground truth label assigned color. Points with a ground-truth not belonging to the shown class are grayed out. The source only model tends to over predict vegetation and sometimes mistakes dense partially occluded object with other classes, e.g., pedestrian instead of motorcycle in the second example. MuDDoS is able to partially or completely recover the correct classes.

4 Comparison to SOTA methods

In this section, we compare our method to state-of-the-art methods in unsupervised domain adaptation across lidars. We recall the final setup for our method.

- *Backbone.* We use WI-768 or MinkUNet as in [58, 43]. In both cases, we exploit the findings in our study: we use layernorms and remove intensity from the input.
- *Pretraining datasets.* The distillation is done on the combination of nuScenes, SemanticKITTI, Waymo and Pandaset (N&K&W&P). The *same* pretrained backbone is re-used for all cross-domain settings.
- *Downstream training.* After distillation, the backbone weights are frozen and we train an MLP classification head on the source dataset.

High-level comparisons. We present the results in Tab. 6. First, we remark that our technique with MinkUNet performs better than Adapt-SAM with significant margin on three out of the four pairs of dataset considered. We also notice that our best discovered setup with WI-768 outperforms former results with very significant margins (up to 24.2 mIoU pts on $N \rightarrow W$, and 18.4 mIoU pts on average). It shows the interest of optimizing each stage of the training pipeline from the choice of the backbone to the downstream training recipes, by way of the pretraining datasets.

In Tab. 6(b), we contrast directly which of our findings differ from what has been used in the current SOTA method Adapt-SAM [58]. At the time of submission, no code is available

for [88] to integrate our findings in their method.

Per class comparisons. The main comparisons in Tab. 6 are extracted from [88], which, unfortunately, does not provide per class results. To get hints about the classes which perform well or not with our method, we compare it to results reported in [29]. Please refer to Tab. 11 in the supplementary material for numbers. None of the methods in this comparison uses intensity as input features. We notice that easy classes such as car, vegetation and drivable surface perform well with all methods. The most notable differences (also to the source-only models) are on bicycle, motorcycle, truck and pedestrian, where our approach performs significantly better.

Qualitative results. We present in Fig. 2 qualitative results on challenging classes. While not perfect, our method is able to correctly segment part of objects that are completely mislabeled with the source-only model.

5 Conclusion

The main messages of this work are that: (1) the choice of the backbone architecture is a key part in the design of domain adaptation methods: simple changes can significantly improve the performance; (2) it is possible to pretrain a single backbone to address many domain shifts thanks to the distillation of VFM features; (3) the downstream training recipe should be designed with care to avoid degrading the quality of the pretrained features.

Looking ahead, this study opens several interesting perspectives. Indeed, as the distillation process does not require any annotation, scaling the backbone and the size of the pretraining dataset even more could lead to improved generalization capabilities, offering foundation models for lidar point clouds that are as powerful as for images. Future research directions include combining several VFMs, e.g., to benefit from the quality of DINOv2 features for semantic understanding and of SAM features for instance augmentations.

Acknowledgements

We also acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grants ANR-21-CE23-0032 (project MultiTrans), ANR-20-CHIA- 0030 (OTTOPIA AI chair), and the European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617. This work was performed using HPC resources from GENCI-IDRIS (2024-AD011013839R2 and AD011015497R1).

References

- [1] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.
- [2] Alejandro Barrera, Jorge Beltrán, Carlos Guindel, Jose Antonio Iglesias, and Fernando García. Cycle and semantic consistent adversarial domain adaptation for reducing simulation-to-real domain shift in lidar bird’s eye view. In *ITSC*, 2021.

- [3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019.
- [4] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovasz-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018.
- [5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.
- [6] Haozhi Cao, Yuecong Xu, Jianfei Yang, Pengyu Yin, Shenghai Yuan, and Lihua Xie. Mopa: Multi-modal prior aided domain adaptation for 3d semantic segmentation. In *ICRA*, 2024.
- [7] Adriano Cardace, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Exploiting the complementarity of 2d and 3d networks to address domain-shift in 3d semantic segmentation. In *CVPRW*, 2023.
- [8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- [9] Runnan Chen, Youquan Liu, Lingdong Kong, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, Yu Qiao, and Wenping Wang. Clip2scene: Towards label-efficient 3d scene understanding by clip. In *CVPR*, 2023.
- [10] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.
- [11] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.
- [12] Robert DeBortoli, Li Fuxin, Ashish Kapoor, and Geoffrey A Hollinger. Adversarial training on point clouds for sim-to-real 3d object detection. *RA-L*, 2021.
- [13] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur’elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *ICCV*, 2021.
- [14] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nusenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *RA-L*, 2021.
- [15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.
- [16] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *CVPR*, 2020.

- [17] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. Cross-modal learning for domain adaptation in 3D semantic segmentation. *T-PAMI*, 2022.
- [18] Jincen Jiang, Qianyu Zhou, Yuhang Li, Xuequan Lu, Meili Wang, Lizhuang Ma, Jian Chang, and Jian Jun Zhang. Dg-pic: Domain generalized point-in-context learning for point cloud understanding. In *ECCV*, 2024.
- [19] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation. In *ICRA*, 2021.
- [20] Hyeonseong Kim, Yoonsu Kang, Changgyoon Oh, and Kuk-Jin Yoon. Single domain generalization for lidar semantic segmentation. In *CVPR*, 2023.
- [21] Lingdong Kong, Niamul Quader, and Venice Erin Liong. Conda: Unsupervised domain adaptation for lidar segmentation via regularized domain concatenation. In *ICRA*, 2023.
- [22] 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 *CVPR*, 2023.
- [23] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *PR*, 2018.
- [24] Wei Liu, Zhiming Luo, Yuanzheng Cai, Ying Yu, Yang Ke, José Marcato Junior, Wesley Nunes Gonçalves, and Jonathan Li. Adversarial unsupervised domain adaptation for 3d semantic segmentation with multi-modal learning. *ISPRS*, 2021.
- [25] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. In *NeurIPS*, 2023.
- [26] Yueh-Cheng Liu, Yu-Kai Huang, Hung-Yueh Chiang, Hung-Ting Su, Zhe-Yu Liu, Chin-Tang Chen, Ching-Yu Tseng, and Winston H Hsu. Learning from 2d: Contrastive pixel-to-point knowledge transfer for 3d pretraining. *arXiv preprint arXiv:2104.04687*, 2021.
- [27] Anas Mahmoud, Jordan SK Hu, Tianshu Kuai, Ali Harakeh, Liam Paull, and Steven L Waslander. Self-supervised image-to-point distillation via semantically tolerant contrastive loss. In *CVPR*, 2023.
- [28] Anas Mahmoud, Ali Harakeh, and Steven Waslander. Image-to-lidar relational distillation for autonomous driving data. In *ECCV*, 2024.
- [29] Björn Michele, Alexandre Boulch, Gilles Puy, Tuan-Hung Vu, Renaud Marlet, and Nicolas Courty. SALUDA: Surface-based automotive lidar unsupervised domain adaptation. In *3DV*, 2024.
- [30] Björn Michele, Alexandre Boulch, Tuan-Hung Vu, Gilles Puy, Renaud Marlet, and Nicolas Courty. Train till you drop: Towards stable and robust source-free unsupervised 3d domain adaptation. In *ECCV*, 2024.

- [31] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv preprint arXiv:1711.10288*, 2017.
- [32] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D’Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.
- [33] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.
- [34] Aljoša Ošep, Tim Meinhardt, Francesco Ferroni, Neehar Peri, Deva Ramanan, and Laura Leal-Taixé. Better call sal: Towards learning to segment anything in lidar. In *ECCV*, 2024.
- [35] Junsung Park, Kyungmin Kim, and Hyunjung Shim. Rethinking data augmentation for robust lidar semantic segmentation in adverse weather. In *ECCV*, 2024.
- [36] Duo Peng, Yinjie Lei, Wen Li, Pingping Zhang, and Yulan Guo. Sparse-to-dense feature matching: Intra and inter domain cross-modal learning in domain adaptation for 3d semantic segmentation. In *ICCV*, 2021.
- [37] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *CVPR*, 2023.
- [38] Xidong Peng, Runnan Chen, Feng Qiao, Lingdong Kong, Youquan Liu, Yujing Sun, Tai Wang, Xinge Zhu, and Yuexin Ma. Learning to adapt SAM for segmenting cross-domain point clouds. In *ECCV*, 2024.
- [39] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Using a waffle iron for automotive point cloud semantic segmentation. In *ICCV*, 2023.
- [40] Gilles Puy, Spyros Gidaris, Alexandre Boulch, Oriane Siméoni, Corentin Sautier, Patrick Pérez, Andrei Bursuc, and Renaud Marlet. Three pillars improving vision foundation model distillation for lidar. In *CVPR*, 2024.
- [41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [42] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *CVPR*, 2024.
- [43] 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 *ECCV*, 2022.
- [44] Cristiano Saltori, Evgeny Krivosheev, Stéphane Lathuilière, Nicu Sebe, Fabio Galasso, Giuseppe Fiameni, Elisa Ricci, and Fabio Poiesi. Gipso: Geometrically informed propagation for online adaptation in 3d lidar segmentation. In *ECCV*, 2022.

- [45] Cristiano Saltori, Aljosa Osep, Elisa Ricci, and Laura Leal-Taixé. Walking your lidog: A journey through multiple domains for lidar semantic segmentation. In *ICCV*, 2023.
- [46] Jules Sanchez, Jean-Emmanuel Deschaud, and François Goulette. Domain generalization of 3d semantic segmentation in autonomous driving. In *ICCV*, 2023.
- [47] Jules Sanchez, Jean-Emmanuel Deschaud, and François Goulette. Cola: Coarse-label multi-source lidar semantic segmentation for autonomous driving. *IEEE Transactions on Robotics*, 2025.
- [48] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised distillation for autonomous driving data. In *CVPR*, 2022.
- [49] Amirreza Shaban, JoonHo Lee, Sanghun Jung, Xiangyun Meng, and Byron Boots. Lidar-uda: Self-ensembling through time for unsupervised lidar domain adaptation. In *ICCV*, 2023.
- [50] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schuster, Buyu Liu, Sparsh Garg, In So Kweon, and Kuk-Jin Yoon. Mm-tta: multi-modal test-time adaptation for 3d semantic segmentation. In *CVPR*, 2022.
- [51] Yi Wei, Zibu Wei, Yongming Rao, Jiaxin Li, Jie Zhou, and Jiwen Lu. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In *ECCV*, 2022.
- [52] Zhixiang Wei, Lin Chen, Yi Jin, Xiaoxiao Ma, Tianle Liu, Pengyang Ling, Ben Wang, Huaian Chen, and Jinjin Zheng. Stronger fewer & superior: Harnessing vision foundation models for domain generalized semantic segmentation. In *CVPR*, 2024.
- [53] Lisa Weijler, Muhammad Jehanzeb Mirza, Leon Sick, Can Ekkazan, and Pedro Hermosilla. Ttt-kd: Test-time training for 3d semantic segmentation through knowledge distillation from foundation models. In *3DV*, 2025.
- [54] Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *CVPR*, 2024.
- [55] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. Pandaset: Advanced sensor suite dataset for autonomous driving. In *ITSC*, 2021.
- [56] Jingyi Xu, Weidong Yang, Lingdong Kong, Youquan Liu, Rui Zhang, Qingyuan Zhou, and Ben Fei. Visual foundation models boost cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *arXiv preprint arXiv:2403.10001*, 2024.
- [57] Xiang Xu, Lingdong Kong, Hui Shuai, Wenwei Zhang, Liang Pan, Kai Chen, Ziwei Liu, and Liu Qingshan. 4D contrastive superflows are dense 3d representation learners. In *ECCV*, 2024.
- [58] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & Label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *CVPR*, 2021.

- [59] Zhimin Yuan, Wankang Zeng, Yanfei Su, Weiquan Liu, Ming Cheng, Yulan Guo, and Cheng Wang. Density-guided translator boosts synthetic-to-real unsupervised domain adaptive segmentation of 3d point clouds. In *CVPR*, 2024.
- [60] Boxiang Zhang, Zunran Wang, Yonggen Ling, Yuanyuan Guan, Shenghao Zhang, and Wenhui Li. Mx2m: masked cross-modality modeling in domain adaptation for 3d semantic segmentation. In *AAAI*, 2023.
- [61] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *CVPR*, 2022.
- [62] Haimei Zhao, Jing Zhang, Zhuo Chen, Shanshan Zhao, and Dacheng Tao. Unimix: Towards domain adaptive and generalizable lidar semantic segmentation in adverse weather. In *CVPR*, 2024.
- [63] 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. In *AAAI*, 2021.
- [64] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In *CVPR*, 2023.
- [65] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. In *NeurIPS*, 2023.

Appendix

In this supplementary material, we describe the training protocols in Sec. A, provide additional information about the used datasets in Sec. B and show per class results in Sec. C.

A Experimental setups, and implementation details

For further research, the code will be made publicly available at github.com/valeoai/muddos. Here we list out the training details.

Training protocol for supervised training (3.1).

For the supervised training we use standard augmentations on the point cloud: random rotation around the z-axis, random flip of the x or y axes, and scaling of the coordinates by a factor chosen uniformly at random within $[0.9, 1.1]$. The loss is the sum of the cross-entropy and the Lovász loss [4]. We use AdamW as optimizer. The backbones are trained for 45 epochs, using a batch size of 8, a weight decay of 0.003, a base learning rate of 10^{-3} with linear warmup during 4 epochs and a cosine decay towards 0.

Training protocol for multimodal distillation (3.2).

The lidar backbones are pretrained using ScaLR [44]. This pretraining method requires calibrated and synchronized cameras and lidars to establish correspondences between points and pixels. It does not require manual annotations. The backbone is pretrained by minimizing the ℓ_2 distance between normalized pixel-features (coming from the VFM) and normalized point-features (coming from the lidar backbone). The pretraining hyperparameters are adapted from ScaLR. The weights of the lidar backbone are optimized using AdamW, a batch size of 2, a weight decay of 0.03, and a base learning rate of $5 \cdot 10^{-4}$ with a linear warmup phase followed by a cosine decay. The images are resized to 224×448 . No augmentations are used on the images. We use the the point cloud augmentations of Sec.3.1.

Training protocol for downstream training (3.3).

For (a), (b) and (c), the weights are optimized for 10 epochs of the source dataset, using a batch size of 8, a base learning rate of 10^{-3} with linear warmup during 1 epoch and a cosine decay reaching 0 at the end of the last training iteration and a weight decay of 0.03. For full finetuning (c), we also use a layerwise learning rate decay of 0.99 [4, 44] and do not apply weight decay on the pretrained weights, as done in [44].

Training protocol self-training (3.5).

For the self-training we continue to train only the MLP head and keep the weights of the backbone frozen. The teacher weights are a moving average of the student weights with momentum 0.99. We alternate between a batch made of source point clouds and a batch made of target point clouds. The teacher provides pseudo class labels to the student for target point cloud. We keep the pseudo-label only when the corresponding softmax value is above 0.9. The loss is the sum of the cross-entropy and Lovász loss on source point clouds, and the cross-entropy on target point clouds.

B Datasets.

We conduct our study using four datasets in total: nuScenes [12], SemanticKITTI [8, 15], Waymo Open Dataset [13]. The pairs of source and target datasets considered are: (nuScenes, SemanticKITTI) denoted by $N \rightarrow K$, (SemanticKITTI, nuScenes) denoted by $K \rightarrow N$, (nuScenes, Waymo) denoted by $N \rightarrow W$, (Waymo, nuScenes) denoted by $W \rightarrow N$. It should be noted that for each dataset a different lidar sensor is used. In nuScenes the lidar used has 32 beams, whereas both lidars in SemanticKITTI and Waymo have 64 beams. This difference makes the settings especially difficult. The details of the used datasets are outlined in Tab. 7. The performance is evaluated by computing the mIoU after re-mapping the original classes in each dataset to 10 common classes: ‘car’, ‘bicycle’, ‘motorcycle’, ‘truck’, ‘bus’/‘oth. vehicle’, ‘person’, ‘driveable surface’, ‘sidewalk’, ‘vegetation’, and ‘terrain’.

The exact class mapping, as also used in [68], can be seen in Tab. 8 and Tab. 9.

C Per class results.

The per class results corresponding to the class mappings of Tab. 8 and Tab. 9 and the results in Tab. 6 are presented in Tab. 10.

In Tab. 11, we also report the results of MuDDoS with the class mapping from [29], this allows to compare to AdaBN [23], PTBN [52], CoSMix [43] and SALUDA [29]. Consistently with the results of the main paper, our method (the only one to be multimodal) surpasses the other approaches by a large margin (+5.9% mIoU) on $N \rightarrow K$.

Dataset		Lidar	Beams/ Channels	Cameras	cls.	Nb. Train/ Val. Frames	Region of the world
nuScenes	 (N)	HDL-32E	32	6	16	28,130 / 6,019	Boston, Singapore
SemanticKITTI	 (K)	HDL-64E	64	1	19	19,130 / 4,071	Karlsruhe
Waymo Open	 (W)	L.B.H.	64	5	23	23,691 / 5976	3 US cities
PandaSet [†]	 (P)	Pandar64/ -GT	64/150	6	37	3,800	2 US cities

Table 7: **Datasets used in our domain adaptation experiments.** [†]: Only used for distillation.

Original nuScenes classes	Shared set of classes	Original SemanticKITTI classes
Car	Car	Car
Bicycle	Bicycle	Bicycle [†]
Motorcycle	Motorcycle	Motorcycle [†]
Truck	Truck	Truck
Bus	Oth. vehicle	Oth. vehicle
Pedestrian	Pedestrian	Person
Driveable Surface	Driveable surface	Road, Parking
Sidewalk	Sidewalk	Sidewalk
Terrain	Terrain	Terrain
Vegetation	Vegetation	Vegetation, Trunk

Table 8: **Class mapping used for N→K and K→N used in [38] and our work.** [†]: the classes ‘bicycle’ and ‘motorcycle’ do not include the original classes ‘motorcyclist’ and ‘bicyclist’, respectively, which are both mapped to ‘ignore’, like all official classes not mentioned in this table.

Original nuScenes classes	Shared set of classes	Original Waymo classes
Car	Car	Car
Bicycle	Bicycle	Bicycle/ist
Motorcycle	Motorcycle	Motorcycle/ist
Truck	Truck	Truck
Bus	Bus	Bus
Pedestrian	Pedestrian	Person
Driveable Surface	Driveable surface	Road, Lane marking
Sidewalk	Sidewalk	Sidewalk
Terrain	Terrain	Walkable
Vegetation	Vegetation	Vegetation, Tree Trunk

Table 9: **Class mapping used for N→W and W→N in [38] and our work.** All official classes in the original datasets not mentioned in this table are mapped to ‘ignore’.

Method	Backbone	mIoU%	Car	Bicycle	Motorcycle	Truck	Bus/Oth. veh.	Pedestrian	Drive. surf.	Sidewalk	Terrain	Vegetation
N→K												
Source only	WI-768	44.6	89.9	4.1	25.5	18.3	1.7	53.8	75.8	47.2	45	84.7
Ours	WI-768	52.1	89.7	19.7	48.0	26	4.9	63.2	76.2	43.6	61.3	88.4
K→N												
Source only	WI-768	55.1	75.9	5.1	48.4	38.2	38.9	55.2	90.0	53.2	56.5	89.8
Ours	WI-768	66.4	83.2	10.5	74.4	57.0	58.8	67.9	91.6	60.0	70.1	91.0
N→W												
Source only	WI-768	37.1	49.5	2.1	17.3	21.5	9.2	41.3	72.3	43.1	39.7	75.4
Ours	WI-768	69.1	85.1	49.5	48.9	41.1	75.9	81.2	88.9	61.8	66.6	92.2
W→N												
Source only	WI-768	64.6	81.7	14.2	38.9	65.8	74.9	68.1	91.8	59.8	61.2	89.5
Ours	WI-768	70.5	80.8	29.7	75.8	71.1	71.7	68.9	91.1	55.0	69.2	91.8

Table 10: **Classwise IoU% for N→K, K→N, N→W and W→N.** The class mapping is the one used in Tab. 6 of the main paper.

Method	Backbone	UDA Modal.	Norm.	mIoU%	Car	Bicycle	Motorcycle	Truck	Oth. vehicle	Pedestrian	Drive. surf.	Sidewalk	Terrain	Vegetation
Source only [†]	MUNet.	-	Batch	35.9	73.7	8.0	17.8	12.0	7.4	49.4	50.2	27.0	31.6	82.1
Source only	WI-768	-	Layer	44.6	89.8	4.2	25.6	18.3	1.6	53.8	75.8	47.2	45.0	76.8
AdaBN [†] [23]	MUNet.	U	Batch	40.1	84.1	16.5	24.0	7.6	3.5	19.2	76.0	35.6	51.0	83.1
PTBN [‡] [60]	MUNet.	U	Batch	39.4	80.0	14.7	27.0	7.3	5.5	23.2	71.3	35.4	48.8	80.6
CoSMix [†] [43]	MUNet.	U	Batch	38.3	77.1	10.4	20.0	15.2	6.6	51.0	52.1	31.8	34.5	84.8
SALUDA [29]	MUNet.	U	Batch	46.2	89.8	13.2	26.2	15.3	7.0	37.6	79.0	50.4	55.0	88.3
Ours	WI-768	M	Layer	52.1	88.9	19.8	48.1	26.0	5.0	63.2	76.2	43.6	61.3	88.5

Table 11: **Classwise IoU% for N→K.** The results are obtained from [29] for all methods marked with [†], and from [60] for those marked with [‡]. Note that the class mapping in this table is the one used in [29], which has minor differences with the one used in Tab. 6.