

# No Thing, Nothing: Highlighting Safety-Critical Classes for Robust LiDAR Semantic Segmentation in Adverse Weather

Junsung Park

Hwiyeong Lee

Inha Kang

Hyunjung Shim

Korea Advanced Institute of Science and Technology

108, Taebong-ro, Seocho-gu, Seoul, Republic of Korea

{jshackist, hjee0612, rkswlsj13, kateshim}@kaist.ac.kr

## Abstract

Existing domain generalization methods for LiDAR semantic segmentation under adverse weather struggle to accurately predict “things” categories compared to “stuff” categories. In typical driving scenes, “things” categories can be dynamic and associated with higher collision risks, making them crucial for safe navigation and planning. Recognizing the importance of “things” categories, we identify their performance drop as a serious bottleneck in existing approaches. We observed that adverse weather induces degradation of semantic-level features and both corruption of local features, leading to a misprediction of “things” as “stuff”. To mitigate these corruptions, we suggest our method, *NTN - segmeNt Things for No-accident*. To address semantic-level feature corruption, we bind each point feature to its superclass, preventing the misprediction of things classes into visually dissimilar categories. Additionally, to enhance robustness against local corruption caused by adverse weather, we define each LiDAR beam as a local region and propose a regularization term that aligns the clean data with its corrupted counterpart in feature space. *NTN* achieves state-of-the-art performance with a +2.6 mIoU gain on the SemanticKITTI-to-SemanticSTF benchmark and +7.9 mIoU on the SemanticPOSS-to-SemanticSTF benchmark. Notably, *NTN* achieves a +4.8 and +7.9 mIoU improvement on “things” classes, respectively, highlighting its effectiveness.

## 1. Introduction

LiDAR Semantic Segmentation (LSS) involves predicting semantic labels directly for each 3D point, an important building block for autonomous driving. However, LSS models exhibit a significant performance drop under adverse weather conditions such as fog, rain, and snow [8, 12, 17, 25, 28]. Since safety is the highest priority in autonomous

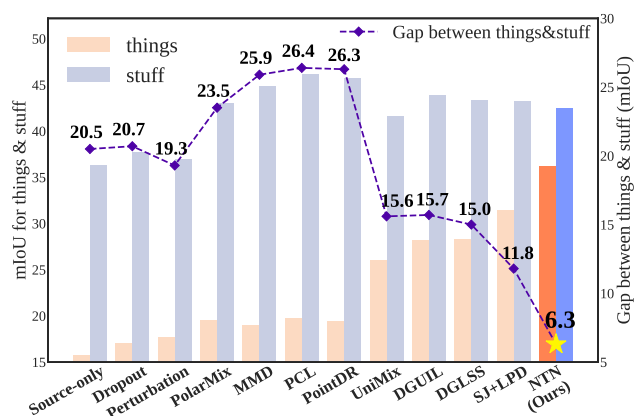


Figure 1. Previous methods struggle to accurately predict the things category in adverse weather. Our approach overcomes this limitation with improved performance for things category.

driving, it is essential to develop robust models that can maintain high accuracy regardless of weather conditions.

To mitigate this issue, existing research on improving robustness under adverse weather conditions can be broadly categorized into domain generalization and domain adaptation methods. Domain adaptation methods [28] utilize real weather training data to train LSS models to be robust against weather corruption. In contrast, domain generalization methods aim to create models robust to weather corruption only using training data under clean weather. Typically, these studies either improve training mechanisms [8, 25] or generate LiDAR data through weather simulations [6, 7, 17]. Domain generalization in adverse weather is particularly important, as real weather varies in type and severity and occurs infrequently, making data collection challenging. In this paper, we focus on the domain generalization problem.

However, existing Domain Generalization methods for LiDAR Segmentation under Adverse Weather (DGLS-AW) show significant performance variations across semantic

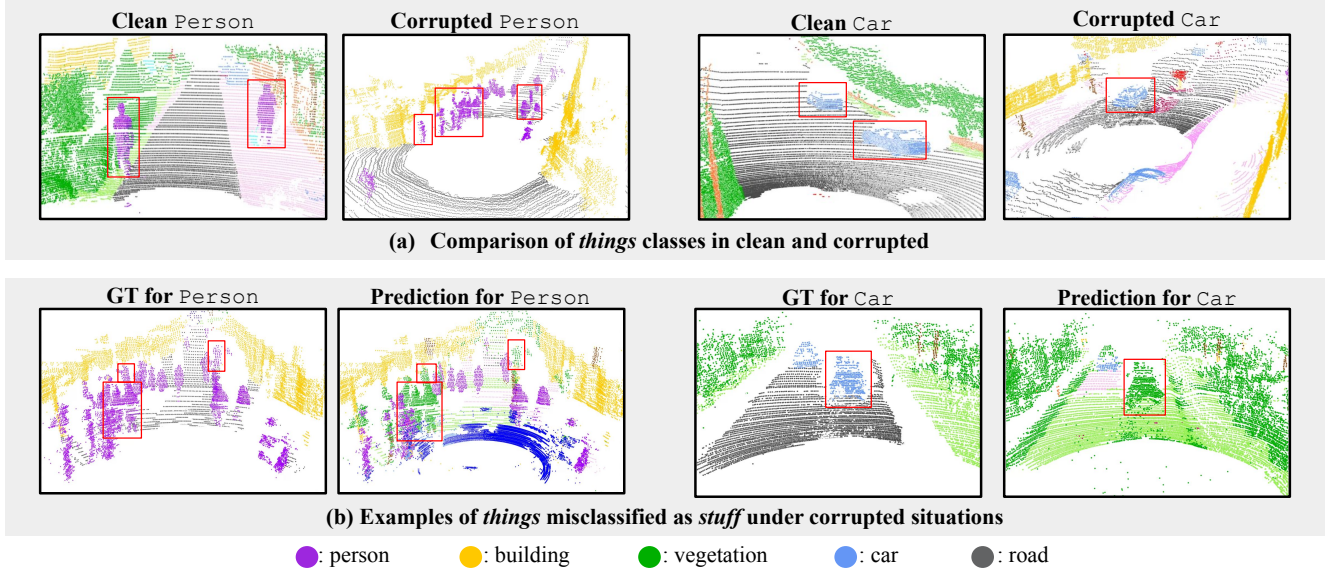


Figure 2. (a) Comparison of *things* classes in SemanticKITTI [3] (clean) and SemanticSTF [25] (corrupted) datasets. (b) Examples of *things* classes misclassified as *stuff* due to weather corruption. The prediction results were obtained with SJ+LPD [17].

classes. In driving scenarios, semantic categories can be divided into *stuff* and *things*. *Stuff* shows continuous distributions and is static such as road and terrain. Meanwhile, *things* consist of distinct and dynamic instances like pedestrians, bicycles, cars, and trucks. As shown in Fig. 1, the segmentation models generalized for adverse weather [8, 12, 17, 25, 28] exhibit significantly lower prediction performance for *things* compared to *stuff*. Even the state-of-the-art model SJ+LPD [17] shows a significant performance gap, with *things* mIoU averaging 11.8 lower than *stuff*. This performance gap poses serious challenges for autonomous driving. Failing to accurately detect objects like pedestrians or vehicles can lead to critical incidents, as it hinders the ability to react quickly to their movements. On the other hand, missing *stuff* category generally has a lower impact on immediate accident risk.<sup>1</sup> For these reasons, accurately detecting *things* is a critical mission considering driving applications.

Our study focuses on mitigating performance limitations for the *things* classes in DGLS-AW. For that, we first identify key factors contributing to the performance gap and propose a solution to address each challenge: weather-induced corruption confuses the LSS model at both (1) semantic-level and (2) local-level features separately, especially for *things* classes.

First, adverse weather causes severe deformation, particularly affecting *things* classes, making their shapes un-

recognizable and hindering the learning of semantic-level features. As shown in Fig. 2 (a), *stuff* classes (building, vegetation, and road) are large enough to cover the entire scene, therefore their geometric features remain stable regardless of whether the data is clean or corrupted. In contrast, *things* classes show surprisingly different appearances in clean and corrupted data.

This difference arises because *things* classes are generally small and dispersed, leaving them vulnerable to missing points that can distort their structure. For example, *person* class, belonging to the *things* category, have dense and clear shapes under clean weather, as shown on the far left of the Fig. 2 (a) and (b). Consequently, under adverse weather conditions, the *person* and *car* points are fewer in number and have obscure shapes. We analyze that weather-induced corruption blurs the boundary between *things* and *stuff*, leading to frequent misclassification. Even the state-of-the-art method, SJ+LPD [17], often confused between *things* and *stuff* as shown in Fig. 2 (b).

To tackle this, we propose a **Feature Binder (FB)** that groups semantically similar classes into superclasses. By binding each point feature to its superclass, FB ensures that even in the presence of corruption, *things* classes can learn and retain their semantic information. This approach effectively reduces misclassification between *things* and *stuff*.

Second, the corruption leads to the loss of local-level features, which is especially critical for *things* classes due to their smaller size. Previous research states that adverse weather effects like point drops affect each laser beam independently, causing local feature corruption [6, 7, 17]. Since *things* classes rely heavily on distinctive local features (e.g.,

<sup>1</sup>It is well-known that misclassifying a *things* category as a *stuff* category can be fatal, as it fails to account for the potential movement of these objects. It was revealed that an accident that occurred in May 2016 while using a driving assistance system was caused by incorrectly predicting the white part of the trailer as the sky [1, 9].

point patterns, intensity) for accurate prediction, this corruption is particularly detrimental to them. Missing points remove essential training signals from these regions, preventing effective feature learning for precise predictions.

To mitigate this issue, we introduce **Beam-wise Feature Distillation (BFD)**, which aligns the features of corrupted data with those of clean data. By matching corrupted points with their clean counterparts, we provide additional information to the *things* classes. This addition allows the model to robustly predict *things* even under severe corruption, as the dense points from the clean data compensate for the missing information in the corrupted data.

We propose **NTN** (segmeNt Things for No-accident), a novel approach that integrates FB and BFD to tackle semantic- and local-level feature corruption in DGLSS-AW. NTN significantly improves the robustness of LSS models against adverse weather conditions, with particular benefits for safety-critical *things* classes. This approach achieves state-of-the-art performance on both the SemanticKITTI-to-SemanticSTF and SemanticPOSS-to-SemanticSTF benchmarks, with overall mIoU improvements of +2.6 and +7.9 for each, respectively. Notably, NTN demonstrates a +4.8 and +7.9 boost in *things* classes on these benchmarks, significantly increasing accuracy for highly mobile or structurally complex objects such as *person*, *bicycle*, and *motorcyclist*. These results show NTN’s ability to maintain precise segmentation of risk-sensitive objects in adverse weather scenarios.

In summary, our contributions are as follows:

1. To our knowledge, we are the first to identify a significant performance degradation in safety-critical *things* classes under adverse weather conditions. We found that damage to both semantic- and local-level features caused by weather corruption is a key factor in performance degradation, particularly for *things* classes.
2. To address these issues, we propose segmeNt Things for No-accident, which integrates two novel modules; *Feature Binder* and *Beam-wise Feature Distillation*.
3. Our approach achieves state-of-the-art results on the SemanticKITTI-to-SemanticSTF and SemanticPOSS-to-SemanticSTF benchmarks, with significant gains of +4.8 and +7.9 in *things* classes performance, respectively.

## 2. Related Works

### 2.1. LiDAR Semantic Segmentation

LSS methods can be broadly categorized into (1) point-based, (2) projection-based, and (3) voxel-based approaches based on their input representations.

Point-based methods [18, 21, 27] process 3D points directly as input. PointNet [18] employs sequential MLPs for directly merging local features of points into global feature vector. KPConv [21] defines 3D kernel points and per-

forms direct convolution with input points. Building upon this approach, KPConvX [22] introduces attention mechanisms within the kernel points, thereby enhancing the model capacity without significantly increasing the size of the 3D convolutional network. The Point-Mixer [4] adapts the MLP-Mixer [23] for point cloud processing. The Point Transformer [27] leverages a transformer architecture to compute query points within each local neighborhood found by k-nearest neighbors. While effective, these methods incur high computational costs due to processing large-scale raw LiDAR data. This is why point-based methods are not suitable for outdoor LSS.

Projection-based methods [2, 11, 15] project LiDAR points onto a 2D plane, allowing semantic segmentation to leverage architectures originally developed for 2D image tasks. RangeNet++ [15] was the first to address LSS using range images via spherical projection, adapting 2D convolutional neural networks for this purpose. RangeViT [2] applies a pre-trained ViT model on 2D images, demonstrating effective use of 2D image pre-training for range images. RangeFormer [11] introduces “RangeAug,” which enhances 2D-projected range images by generating multiple inputs to boost model performance. Projection-based approaches offer fast inference but are limited by information loss during projection.

Voxel-based methods [5, 13, 29] divide 3D space into voxel grids for efficient computation. MinkowskiNet [5] voxelizes LiDAR data into cubic grids, using sparse convolutions. SPVCNN [20] builds on MinkowskiNet with an added point-wise MLP. Cylinder3D [29] uses cylindrical partitions, addressing LiDAR density variations with distance. SphereFormer [13] applies radial windows in voxelization and a transformer to aggregate long-range information. Voxel-based methods offer a balance of inference speed and segmentation accuracy.

### 2.2. LiDAR under Adverse Weather Conditions

Recently, LiDAR perception research in adverse weather has gained attention for its practical significance in autonomous driving, focusing on (1) weather simulation and (2) task-agnostic domain generalization methods.

Hahner *et al.* [6, 7] use physical modeling to create synthetic datasets for specific weather conditions, integrating them into training. UniMix [28] extends these works for adapting LiDAR segmentation models for adverse weather using these simulations. SJ+LPD [17] deviate from physical modeling, incorporating weather-induced corruption via point dropping and jittering to enhance robustness.

Recent task-agnostic approaches [8, 12, 25] employ general machine learning techniques, such as teacher-student frameworks and prototyping, to enhance robustness. Kong *et al.* [12] introduced random dropout in the teacher-student architecture to build noise-resistant models. PointDR [25]

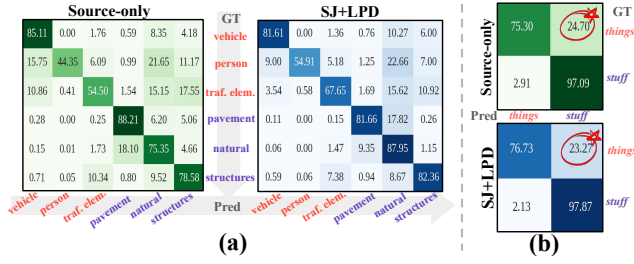


Figure 3. Confusion matrices for (a) superclasses and (b) *things* and *stuff*. It shows a significant performance gap between *things* and *stuff*; *things* are often misclassified as *stuff*, whereas the reverse misclassification is rare.

generates a prototype in the clean weather branch and multiplies corrupted branch features by this prototype matrix, utilizing clean data insights. DGUIL [8] addresses point uncertainty under domain shifts by improving point features across varying distributions in uncertain point clouds.

However, existing weather simulation and domain generalization methods focus on replicating weather conditions and do not address precise recognition of key *things* classes under weather-induced corruption in driving environments. Building on SJ+LPD [17], our work proposes methods to enhance recognition of *things* classes.

### 3. Methods

#### 3.1. Motivation

In this section, we aim to understand why existing LSS methods in adverse weather underperform on the *things* classes. To accomplish this, we analyze the impact of weather-induced corruption in adverse weather on 3D points. The findings in Sec. 1 can be summarized as follows: (1) Weather corruption degrades the semantics of *things* classes. (2) Local-level feature corruption causes significant missing points and leads to information loss.

As shown in Fig. 3, even the current state-of-the-art DGLS-AW method, SJ+LPD [17], misclassifies 22.66% of *person* predictions as *natural*, illustrating the tendency for *things* classes’ predictions to shift toward *stuff*. This misclassification occurs because the model struggles to maintain clear boundaries between visually distinct classes when the input features are corrupted. To address this, we constrain features from the LSS model to remain close within visually similar classes. This approach preserves semantic alignment and mitigates shifts between *things* and *stuff*, even under corruption-induced semantic degradation.

Also, as shown in Fig. 2 (a), significant missing points cause a loss of learning signals during training. This loss is particularly harmful to *things* classes. It is because *things* classes, with fewer points and smaller sizes, heavily rely on these learning signals. Specifically, in SemanticKITTI [3],

| Category | Superclass      | SemanticKITTI                        | SemanticPOSS     |
|----------|-----------------|--------------------------------------|------------------|
| Things   | Vehicle         | car, bicycle, motorcycle, truck, bus | car, bicycle     |
|          | Person          | person, bicyclist, motorcyclist      | pedestrian       |
|          | Traffic Element | fence, pole, traffic-sign            | -                |
| Stuff    | Pavement        | road, parking, sidewalk              | drivable-surface |
|          | Natural         | other-ground, vegetation, terrain    | walkable         |
|          | Structure       | building, trunk                      | -                |

Table 1. Superclasses of each category and its constituent classes for SemanticKITTI and SemanticPOSS datasets.

*things* classes account for only 12.3% of the total points. The limited points in the *things* classes become even scarcer when missing, resulting the poor performance in the *things* category.

By focusing on these key points, we address the degraded performance of the *things* category in existing DGLS-AW methods. We adopt the existing adverse weather LSS, SJ+LPD [17] as a baseline, which achieves state-of-the-art performance through specialized weather simulation augmentations. Importantly, our proposed framework is architecture-independent and applicable to various baseline models beyond SJ+LPD, as it introduces new training schemes.

#### 3.2. Weather-corrupted LiDAR Data Generation

Existing LiDAR weather simulation methods [6, 7, 17] simulate corrupted data  $X^{aug}$  and use them as additional training data to achieve generalized performance under adverse weather. Hahner *et al.* [6, 7] employ physical equations to synthesize weather-specific LiDAR data for training. SJ+LPD [17], which we set as a baseline method, generates output points  $\hat{X}$  by applying selective jittering to the original clean data. Subsequently,  $\hat{X}$  is input into the Learnable Point Drop module  $Q(\cdot)$  to produce the final augmented data  $X^{aug}$ . In subsequent methods, we utilize  $\hat{X}$  as clean data (i.e., the original LiDAR points) and  $X^{aug}$  as augmented data (i.e., the simulated or augmented counterpart).

#### 3.3. Feature Binding Loss

Building on the analysis in Sec. 3.1, we first introduce a concept of Feature Binding loss (FB). The purpose of FB loss is to “bind” the output features of LSS model, preventing misprediction of *things* objects as *stuff* when weather corruption severely affects semantic-level features.

To achieve this, we create prototypes for superclasses by grouping visually similar pre-defined classes during LSS training. Tab. 1 summarizes the superclass categories and their corresponding classes. Six superclasses are defined for SemanticKITTI [3], while four superclasses are defined for SemanticPOSS [16]. Following the method in [10], we associated each class with its respective superclass. Classes

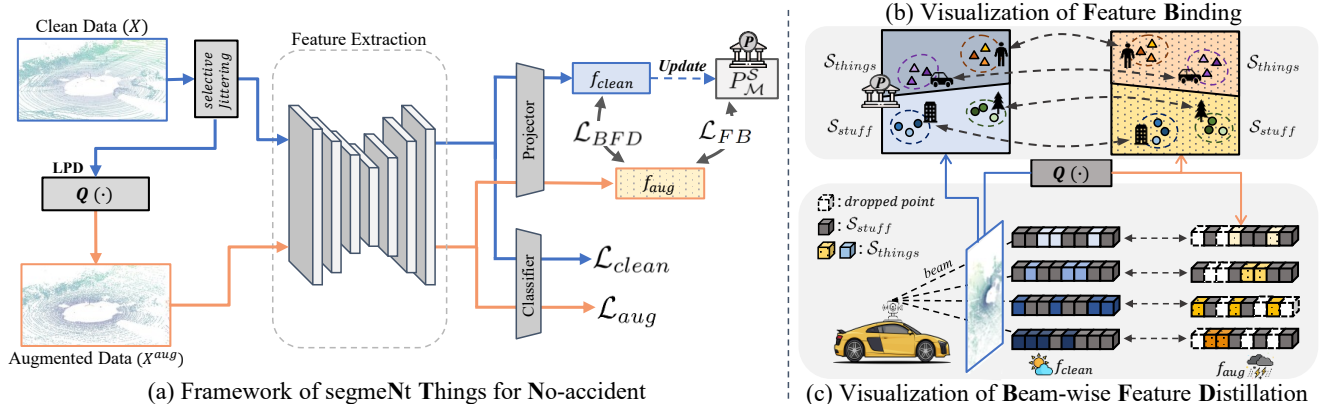


Figure 4. (a) Framework of NTN. NTN builds upon SJ+LPD [17] and enhances the performance of the LSS model on the *things* category through Feature Binding (FB) and Beam-based Feature Distillation (BFD). (b) FB stores prototypes obtained from clean data in a memory bank and encourages features derived from augmented data to align with these prototypes. (c) BFD imposes constraints to ensure that features from clean data and those obtained from augmented data match within a manually defined local region, referred to as a beam.

with significantly different shapes, such as `trunk` and `vegetation` are assigned to separate superclasses. Additionally, classes previously treated as background, like a `motorcyclist`, were reassigned to the `person` class.

We then constrain the output features of the LSS model for augmented data  $x^{aug}$  to remain close to these superclass prototypes. This approach prevents severe misprediction into visually dissimilar *stuff* classes, even if the semantic features of input points are significantly degraded by weather corruption.

Specifically, we compute the prototype for each class  $c_i$  as the mean feature of its constituent classes and store it in the memory bank  $\mathcal{M}$ .

$$P_{\mathcal{M}}^{c_i} = \frac{1}{|\hat{X}_{c_i}|} \sum_{\hat{x}_{c_i} \in \hat{X}_{c_i}} G(F(\hat{x}_{c_i})). \quad (1)$$

$F$  is the LSS model,  $G$  is a projector head with MLP, and  $|\hat{X}_{c_i}|$  denotes the number of points labeled as  $c_i$  in the entire class set  $\hat{X}_C$ . These class-wise prototypes are updated using an exponential moving average during each training iteration. Subsequently, the superclass prototype  $P_{\mathcal{M}}^{S_n}$  is defined using the class-wise prototypes belonging to each superclass  $S_n$  as follows:

$$P_{\mathcal{M}}^{S_n} = \frac{1}{|S_n|} \sum_{c_i \in S_n} P_{\mathcal{M}}^{c_i}. \quad (2)$$

where  $|S_n|$  represents the number of classes in superclass  $S_n$ . We then fit the feature output from augmented data to each superclass prototype to compute the FB loss as follows:

$$\mathcal{L}_{FB} = \left\| \frac{1}{|X_c^{aug}|} \sum_{x_c^{aug} \in X_c^{aug}} G(F(x_c^{aug})) - \frac{1}{|S|} \sum_{S_n \in S} P_{\mathcal{M}}^{S_n} \right\|^2. \quad (3)$$

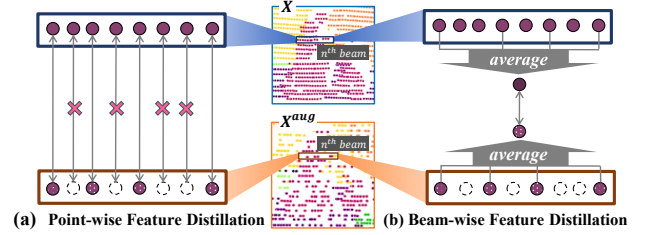


Figure 5. Illustration of (a) Point-wise Feature Distillation and (b) Beam-wise Feature Distillation.

This approach enables the LSS model to bind semantic features to their appropriate superclass by creating prototypes of visually similar, pre-defined classes. This method enables the model to prevent misprediction of *things* as *stuff*, even when handling corrupted inputs.

### 3.4. Beam-wise Feature Distillation

This section presents a method to generate additional learning signals for *things* classes, which are particularly vulnerable to weather corruption due to their limited representation in the dataset. Weather corruption primarily affects localized regions, causing significant point loss and disrupting learning signals. Without sufficient learning signals, the model struggles to capture the necessary features for these classes, leading to degraded performance. To address this issue, we design an additional loss that also operates within localized areas, compensating for the loss of learning signals.

To define the local regions, we slice the LiDAR scan into each LiDAR beam. A beam corresponds to the output response of the laser beam that the LiDAR sensor fires at regular pitch angle intervals. Each beam scans the *things* object into upper and lower local regions. Using these beams, we

| Method        | car    | bi.cle | mt.cle  | truck  | oth-v. | pers.  | bi.clst | mt.clst | road   | parki. | sidew. | othe.g. | build. | fence  | veget. | trunk  | terra. | pole   | traf.  | mIoU   |
|---------------|--------|--------|---------|--------|--------|--------|---------|---------|--------|--------|--------|---------|--------|--------|--------|--------|--------|--------|--------|--------|
| Oracle        | 89.4   | 42.1   | 0.0     | 59.9   | 61.2   | 69.6   | 39.0    | 0.0     | 82.2   | 21.5   | 58.2   | 45.6    | 86.1   | 63.6   | 80.2   | 52.0   | 77.6   | 50.1   | 61.7   | 54.7   |
| Source-only   | 55.9   | 0.0    | 0.2     | 1.9    | 10.9   | 10.3   | 6.0     | 0.0     | 61.2   | 10.9   | 32.0   | 0.0     | 67.9   | 41.6   | 49.8   | 27.9   | 40.8   | 29.6   | 17.5   | 24.4   |
| Dropout [19]  | 62.1   | 0.0    | 15.5    | 3.0    | 11.5   | 5.4    | 2.0     | 0.0     | 58.4   | 12.8   | 26.7   | 1.1     | 72.1   | 43.6   | 52.9   | 34.2   | 43.5   | 28.4   | 15.5   | 25.7   |
| Perturbation  | 74.4   | 0.0    | 0.0     | 23.3   | 0.6    | 19.7   | 0.0     | 0.0     | 60.3   | 10.8   | 33.9   | 0.7     | 72.0   | 45.2   | 58.7   | 17.5   | 42.4   | 22.1   | 9.7    | 25.9   |
| PolarMix [24] | 57.8   | 1.8    | 3.6     | 3.7    | 26.5   | 3.7    | 26.5    | 0.0     | 65.7   | 2.9    | 35.9   | 48.7    | 71.0   | 48.7   | 53.8   | 20.5   | 45.4   | 25.9   | 15.8   | 26.5   |
| MMD [14]      | 63.6   | 0.0    | 2.6     | 17.4   | 11.4   | 28.1   | 0.0     | 0.0     | 67.0   | 14.1   | 37.6   | 41.2    | 67.1   | 41.2   | 57.1   | 27.4   | 47.9   | 28.2   | 16.2   | 26.9   |
| PCL [26]      | 65.9   | 0.0    | 0.4     | 17.3   | 8.4    | 8.4    | 8.4     | 8.4     | 59.6   | 12.0   | 35.3   | 63.1    | 74.0   | 47.5   | 60.7   | 15.8   | 48.9   | 26.1   | 27.5   | 26.4   |
| PointDR [25]  | 67.3   | 0.0    | 4.5     | 19.9   | 18.8   | 2.7    | 0.0     | 0.0     | 62.6   | 12.9   | 38.6   | 43.8    | 73.3   | 43.8   | 56.4   | 32.2   | 45.7   | 28.7   | 27.4   | 28.6   |
| DGLSS [10]    | 72.6   | 0.1    | 11.7    | 29.4   | 13.7   | 48.3   | 0.5     | 21.2    | 65.0   | 20.2   | 38.3   | 3.8     | 78.9   | 51.8   | 57.0   | 36.4   | 47.0   | 26.9   | 34.9   | 34.6   |
| UniMix [28]   | 82.7   | 6.6    | 8.6     | 4.5    | 19.9   | 35.5   | 15.1    | 15.5    | 55.8   | 10.2   | 36.5   | 40.1    | 72.8   | 40.1   | 49.1   | 33.4   | 34.9   | 23.5   | 33.5   | 31.4   |
| DGUIL [8]     | 77.9   | 10.6   | 19.1    | 26.0   | 9.7    | 46.3   | 6.0     | 9.3     | 69.1   | 18.0   | 38.6   | 9.4     | 73.3   | 51.2   | 60.8   | 30.9   | 50.8   | 31.8   | 22.3   | 35.5   |
| SJ+LPD [17]   | 83.1   | 1.2    | 17.2    | 30.5   | 18.4   | 47.5   | 10.7    | 18.8    | 64.0   | 15.9   | 38.7   | 4.6     | 77.4   | 50.8   | 59.7   | 37.2   | 47.7   | 31.1   | 35.8   | 36.3   |
| NTN (Ours)    | 83.3   | 3.7    | 31.3    | 36.2   | 18.2   | 53.3   | 6.8     | 55.9    | 67.2   | 18.1   | 37.2   | 5.4     | 72.1   | 41.8   | 58.0   | 36.0   | 46.0   | 28.2   | 39.8   | 38.9   |
| ↑ to SJ+LPD   | (+0.2) | (+2.5) | (+14.1) | (+5.7) | (-0.2) | (+5.8) | (-3.9)  | (+37.1) | (+3.2) | (+2.2) | (-1.5) | (+0.8)  | (-5.3) | (-9.0) | (-1.7) | (-1.2) | (-1.7) | (+2.9) | (+4.0) | (+2.6) |

Table 2. Comparison of methods on the SemanticKITTI-to-SemanticSTF benchmark. Performance improvements of our method over the current state-of-the-art model, SJ+LPD, are shown in the last row, with red text indicating *things* classes increments and green text for *stuff*.

define a loss ensuring that the average of the clean and augmented data is identical for each beam, exclusively for the *things* classes.

Specifically, we first apply off-the-shelf beam separation algorithms [6, 7, 12] for LiDAR data to identify the LiDAR beam source of the points corresponding to each voxel. Then, for each *things* classes and each LiDAR beam, we apply an MSE loss between the feature of clean data and that of augmented data.

$$\mathcal{F}_{mean}^{c,b} = \frac{1}{N_{c,b}} \sum_{i=1}^{N_{c,b}} F(\hat{x}_i), \quad \mathcal{F}'_{mean}{}^{c,b} = \frac{1}{N'_{c,b}} \sum_{i=1}^{N'_{c,b}} F(x_i^{aug}). \quad (4)$$

$$\mathcal{L}_{BFD} = \sum_{c \in \mathcal{S}_{things}} \sum_{b \in \mathcal{B}} \left\| \mathcal{F}_{mean}^{c,b} - \mathcal{F}'_{mean}{}^{c,b} \right\|^2. \quad (5)$$

$\mathcal{S}_{things}$  denote the *things* classes, and  $c$  a specific class. Let  $\mathcal{B}$  represent all LiDAR beams, and  $b$  a specific beam.  $N_{c,b}$  is the number of points for class  $c$  and beam  $b$ .  $\mathcal{L}_{BFD}$  minimizes the difference between the average features of clean data,  $\mathcal{F}_{mean}^{c,b}$ , and that of augmented data,  $\mathcal{F}'_{mean}{}^{c,b}$ , for each beam, specifically for the *things* classes, as shown in Fig. 5 (b).

Our proposed beam-wise feature matching focuses on local regions, enabling learning from all points in  $\hat{X}$  and leveraging contextual information around each beam unlike point-wise feature distillation in Fig. 5 (a). Points-wise feature distillation matches individual point features for distillation without averaging. In this case, no distillation loss can be computed for points dropped from  $\hat{X}$  when obtaining  $X^{aug}$ . Furthermore, surrounding points contribute no learning information, limiting the clean data branch to learning only from a subset of points in the augmented data branch. In contrast, our beam-wise feature matching overcomes these limitations by effectively handling local feature corruptions in *things* category objects, both before and after weather-induced corruption. Therefore, BFD is a effective

solution for handling local feature corruption in *things* category objects, both before and after weather-induced corruption.

### 3.5. Overall Pipeline

The overall training pipeline is illustrated in Fig. 4. The process for obtaining clean and augmented data follows SJ+LPD [17]. In SJ+LPD, Selective Jittering (SJ) is applied to obtain clean data  $\hat{x}_i$ . The loss and entropy from this data are used as the state for the Learnable Point Drop (LPD). LPD then drops points, and the LSS model outputs features for the augmented data  $x_i^{aug}$ . Thus, we obtain the output feature of clean data  $F(\hat{x}_i)$  and that of augmented data  $F(x_i^{aug})$ . These features are used to compute the FB loss and BFD loss, as described earlier. Additionally, we include the cross-entropy loss from both clean and augmented data. The overall loss is:

$$\mathcal{L}_{total} = \mathcal{L}_{clean} + \lambda_{aug} \mathcal{L}_{aug} + \lambda_{FB} \mathcal{L}_{FB} + \lambda_{BFD} \mathcal{L}_{BFD}. \quad (6)$$

where  $\lambda$ . means the weight for each loss.

## 4. Experiments

### 4.1. Experimental Setup

**Dataset.** We utilize three datasets for autonomous driving to train and evaluate the generalization performance of the LSS model under adverse weather conditions. SemanticKITTI dataset uses a 64-beam LiDAR and provides 19 class labels. Following the official sequence split, we use scenes 00-07 and 09-10 for training, comprising 19,130 frames. The SemanticPOSS dataset uses a 40-beam LiDAR and focuses on campus scenes with a high density of cars, riders, and pedestrians. This dataset provides 14 class labels, and we use scenes 00-02 and 04-05 for training, comprising 2,488 frames. SemanticSTF dataset is collected un-

| Method           | Car         | Bicycle    | Person      | Road        | Walkable    | mIoU        |
|------------------|-------------|------------|-------------|-------------|-------------|-------------|
| Source-only [25] | 46.6        | 3.1        | 54.2        | 47.0        | 37.2        | 37.6        |
| PointDR [25]     | 35.0        | 3.1        | 59.7        | <b>60.7</b> | 14.9        | 34.7        |
| DGLSS [10]       | 60.0        | 3.1        | 38.6        | 54.0        | 40.1        | 39.2        |
| SJ+LPD [17]      | <b>67.0</b> | 0.9        | 45.9        | 43.3        | 34.7        | 38.3        |
| NTN (Ours)       | 65.5        | <b>5.8</b> | <b>61.4</b> | 54.6        | <b>43.6</b> | <b>46.2</b> |
| ↑ to SJ+LPD      | (-1.5)      | (+4.9)     | (+15.5)     | (+11.3)     | (+8.9)      | (+7.9)      |

Table 3. Comparison of methods on the SemanticPOSS-to-SemanticSTF benchmark.

der real-world adverse weather conditions, including rain, fog, and snow. This dataset follows the same 19 class labels used in the SemanticKITTI benchmark. We validate our model on the SemanticSTF *validation set* to evaluate robustness in adverse weather.

**Class mappings.** We apply different class mappings depending on the generalization scenario. For SemanticKITTI  $\rightarrow$  SemanticSTF, we use all 19 classes since they share identical classes. For SemanticPOSS  $\rightarrow$  SemanticSTF, we follow the training and evaluation settings of DGLSS [10], using only 5 classes: {car, bicycle, pedestrian, road, walkable}.

**Evaluation Metrics.** We adopt Intersection over Union (IoU) as the evaluation metric for each segmentation class and calculate the mean IoU (mIoU) across all classes.

**Implementation Details.** We adopted MinkowskiNet-18 architectures [5] as our baseline models. The training was conducted using a stochastic gradient descent (SGD) optimizer with a learning rate of 0.24 and a weight decay of 0.0001. The configurations for SJ+LPD are the same as the previous work [17]. All experiments were executed on four NVIDIA A6000 GPUs for 30 epochs with a batch size of 2. The total runtime for each experiment ranged from 6 to 8 hours. The loss function weights were set to  $\lambda_{\text{aug}} = 0.5$ ,  $\lambda_{\text{SPA}} = 0.5$ , and  $\lambda_{\text{BFD}} = 0.1$ .

## 4.2. Main Results

**SemanticKITTI to SemanticSTF.** As shown in Table 2, our method achieves a state-of-the-art performance with a **+2.6** mIoU improvement over the prior leading model, SJ+LPD [17]. Additionally, our approach shows a significant boost of +14.5 mIoU over the source-only model, the original MinkUnet. Furthermore, our method achieves notable improvements across most *things* classes, yielding gains of **+2.5**, **+14.1**, **+5.7**, **+5.8**, **+37.1**, **+2.9**, and **+4.0** mIoU in *bicycle*, *motorcycle*, *truck*, *person*, *motorcyclist*, *pole*, and *traffic sign* classes, respectively. These results provide strong evidence that our method significantly enhances performance specifically for *things* classes, as intended. Notably, previous methods struggled to predict tail classes with small-sized, such as *motorcycle* and *motorcyclist*, accurately. It indicates that our method offers precise and reliable signals for identifying *things* classes under adverse conditions.

| Prototype  | Feature Distillation |          | Performance                    |       |                                |
|------------|----------------------|----------|--------------------------------|-------|--------------------------------|
|            | Matching             | Classes  | Things                         | Stuff | mIoU                           |
| $\times$   | $\times$             | $\times$ | 31.4                           | 43.2  | 36.3                           |
| Class-wise | $\times$             | $\times$ | 31.6                           | 42.2  | 36.1                           |
| Superclass | $\times$             | $\times$ | 34.9 ( $\uparrow 3.5$ )        | 42.5  | 38.1 ( $\uparrow 1.8$ )        |
| Coarse     | $\times$             | $\times$ | 35.6                           | 41.0  | 37.9                           |
| $\times$   | beam-wise            | all      | 34.0                           | 43.3  | 37.9                           |
| $\times$   | point-wise           | things   | 33.1                           | 41.7  | 36.7                           |
| $\times$   | beam-wise            | things   | 34.4 ( $\uparrow 3.0$ )        | 44.0  | 38.4 ( $\uparrow 2.1$ )        |
| Superclass | beam-wise            | things   | <b>36.2</b> ( $\uparrow 4.8$ ) | 42.5  | <b>38.9</b> ( $\uparrow 2.6$ ) |

Table 4. Ablation study for different prototype and feature distillation settings. Coarse refers to the use of two prototypes divided into *things* and *stuff* for feature binding

**SemanticPOSS to SemanticSTF.** As shown in Table 3, our method establishes a new state-of-the-art on the SemanticPOSS-to-SemanticSTF benchmark, surpassing the performance of the previous model, SJ+LPD [17], by an impressive **+7.9** mIoU. Importantly, in the *person* category, our method achieves a **+15.5** mIoU gain over SJ+LPD. These results underscore the effectiveness of our approach in providing highly accurate segmentation signals for *things* classes under challenging adverse conditions. The SemanticPOSS dataset contains a higher proportion of *things* classes at 28.9%, compared to SemanticSTF, and is known for its diversity. Consequently, the model has already learned relatively rich features, resulting in less performance improvement for *things* classes compared to SemanticKITTI-to-SemanticSTF benchmark.

## 4.3. Ablation Study

All ablation studies are conducted on the SemanticKITTI-to-SemanticSTF benchmark.

**Feature Binding Loss.** We conduct an ablation study to evaluate the impact of various prototype strategies of Feature Binding. We compare three approaches: assigning prototypes to each class individually, grouping semantically similar classes into superclasses, and broadly categorizing classes into coarse prototypes as *stuff* or *things*. As shown in Table 4, using superclass prototypes improves *things* classes performance by **+3.3** mIoU compared to the class-wise method. This improvement indicates that our FB method effectively bounds *things* classes, which are prone to be misclassified as *stuff*, within their respective superclasses. The coarse prototype approach performs well for *things* classes but reduces *stuff* class performance due to the use of a single prototype for numerous samples. In contrast, the superclass prototype strategy achieves balanced results across both *things* and *stuff*, providing a **+3.5** mIoU boost in *things* classes. This shows that creating prototypes by grouping classes with high visual similarity into superclasses is more effective for feature binding. Although the coarse prototype method enhances *things* classes, it causes

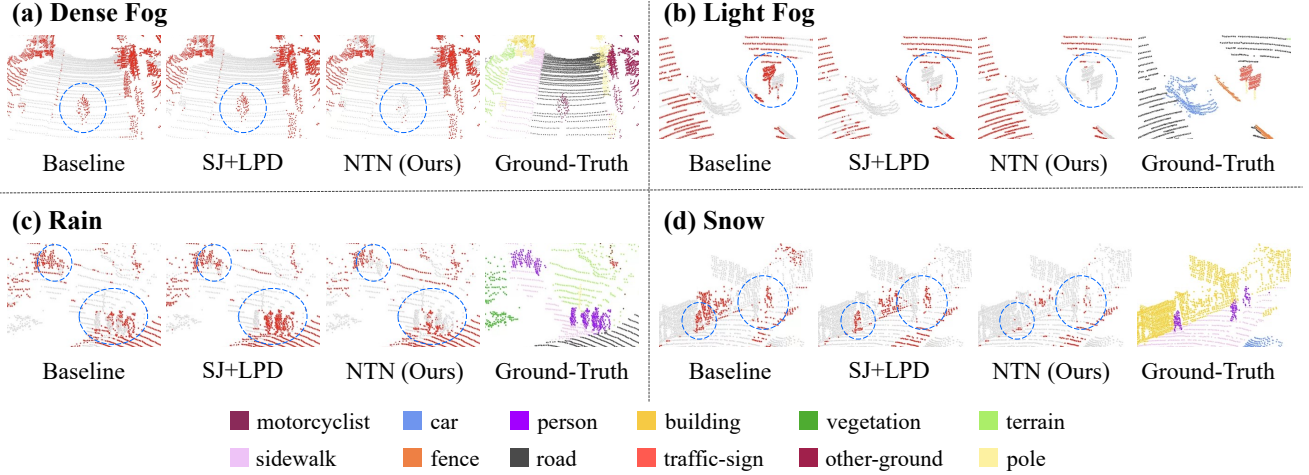


Figure 6. Qualitative results of our method on *validation set* of SemanticSTF. All models are trained on the *train set* of SemanticKITTI. Gray points indicate correct predictions, red points highlight errors, and ground-truth is shown with color-coded labels. Dashed circles highlight the predictions of *things* classes. Best viewed in color.

a 1.5 mIoU decrease in *stuff* and a 0.2 mIoU drop in overall mIoU compared to the superclass prototype approach. This result shows that combining classes with low visual similarity into a single prototype provides suboptimal gains. Besides, improved prediction of *things* in driving scenes is beneficial, but poor prediction of *stuff* can disrupt long-term planning. Thus, the coarse prototype approach cannot be considered to produce ideal results.

**Beam-wise Feature Distillation.** We compare performance based on how feature distillation is applied and to which classes. Table 4 shows that limiting BFD to *things* classes provides better performance across both *things* and *stuff* compared to applying BFD to all classes. This likely occurs because *stuff* classes, with more points in the pitch direction, create class imbalance when BFD is applied. Additionally, point-wise feature matching in row 6 significantly reduces *things* classes performance compared to beam-wise distillation, as it fails to capture learning signals from dropped points, as noted in Sec. 3.4.

**Overall Comparison.** Applying superclass prototypes with beam-wise feature distillation only to *things* classes improves mIoU by 2.6 mIoU over the previous SoTA model, with a remarkable 4.8 mIoU increase in *things* classes performance.

#### 4.4. Qualitative Results

Fig. 6 shows qualitative results of our method and baselines on the SemanticSTF validation set, using SemanticKITTI as the source domain. Correct predictions are marked in gray, errors are in red, and ground-truth classes are color-coded. Notably, our method outperforms SJLPD in segmenting *things* category, underscoring its intended gains for these critical classes. Specifically, in the dense fog scenario

(a), our method accurately segments the *motorcyclist* class, while the original MinkowskiNet and SJ+LPD struggle to recognize it. This aligns with our quantitative result, with a 37.1 IoU gain for the *motorcyclist* class. Additionally, in rain and snow scenarios (c-d), our model excels in segmenting the *person* class, a critical class for autonomous driving due to its direct impact on safety.

## 5. Conclusion

In this study, we introduced our method, NTN, to address the limitations of existing DGLS-AW, particularly their poor performance on *things* classes. We identified two primary causes of this issue: (1) corruption of semantic-level features in *things* objects and (2) their vulnerability to local feature corruption caused by adverse weather conditions. To alleviate semantic-level feature corruption, we proposed a feature-binding approach that leverages superclass prototypes. To address the vulnerability of *things* classes to corruptions on local features, we introduced a beam-wise feature distillation loss that utilizes clean-weather beam-level features. Our methods establish a state-of-the-art on the SemanticKITTI-to-SemanticSTF and SemanticPOSS-to-SemanticSTF benchmarks, significantly boosting *things* classes performance.

**Acknowledgements.** This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the MSIP (RS-2025-00520207, RS-2023-00219019), KEIT grant funded by the Korean government (MOTIE) (No. 2022-0-00680, No. 2022-0-01045), the IITP grant funded by the Korean government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project, No. 2021-0-02068 Artificial Intelligence Innovation Hub, RS-

2019-II190075 Artificial Intelligence Graduate School Program (KAIST)), Samsung Electronics Co., Ltd (IO230508-06190-01) and SAMSUNG Research, Samsung Electronics Co., Ltd.

## References

- [1] National Highway Traffic Safety Administration. Tesla crash preliminary evaluation report. Technical Report PE 16-007, U.S. Department of Transportation, National Highway Traffic Safety Administration, 2017. 2
- [2] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5240–5250, 2023. 3
- [3] 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*, pages 9297–9307, 2019. 2, 4
- [4] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In *European Conference on Computer Vision*, pages 620–640. Springer, 2022. 3
- [5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019. 3, 7
- [6] Martin Hahner, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Fog simulation on real lidar point clouds for 3d object detection in adverse weather. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15283–15292, 2021. 1, 2, 3, 4, 6
- [7] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. Lidar snowfall simulation for robust 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16364–16374, 2022. 1, 2, 3, 4, 6
- [8] Pei He, Licheng Jiao, Lingling Li, Xu Liu, Fang Liu, Wenping Ma, Shuyuan Yang, and Ronghua Shang. Domain generalization-aware uncertainty introspective learning for 3d point clouds segmentation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 651–660, 2024. 1, 2, 3, 4, 6
- [9] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017. 2
- [10] Hyeonseong Kim, Yoonsu Kang, Changgyoon Oh, and Kuk-Jin Yoon. Single domain generalization for lidar semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17587–17598, 2023. 4, 6, 7
- [11] 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. 3
- [12] Lingdong Kong, Youquan Liu, Xin Li, Runnan Chen, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Robo3d: Towards robust and reliable 3d perception against corruptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19994–20006, 2023. 1, 2, 3, 6
- [13] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17545–17555, 2023. 3
- [14] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018. 6
- [15] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4213–4220. IEEE, 2019. 3
- [16] Yancheng Pan, Biao Gao, Jilin Mei, Sibao Geng, Chengkun Li, and Huijing Zhao. Semanticpos: A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693. IEEE, 2020. 4
- [17] Junsung Park, Kyungmin Kim, and Hyunjung Shim. Rethinking data augmentation for robust lidar semantic segmentation in adverse weather. In *European Conference on Computer Vision*. Springer, 2025. 1, 2, 3, 4, 5, 6, 7
- [18] 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*, pages 652–660, 2017. 3
- [19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 6
- [20] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. 3
- [21] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotequi, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 3
- [22] Hugues Thomas, Yao-Hung Hubert Tsai, Timothy D Barfoot, and Jian Zhang. Kpconvx: Modernizing kernel point convolution with kernel attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5535, 2024. 3

- [23] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021. [3](#)
- [24] Aoran Xiao, Jiaying Huang, Dayan Guan, Kaiwen Cui, Shijian Lu, and Ling Shao. Polarmix: A general data augmentation technique for lidar point clouds. *Advances in Neural Information Processing Systems*, 35:11035–11048, 2022. [6](#)
- [25] Aoran Xiao, Jiaying 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*, pages 9382–9392, 2023. [1](#), [2](#), [3](#), [6](#), [7](#)
- [26] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7097–7107, 2022. [6](#)
- [27] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. [3](#)
- [28] Haimei Zhao, Jing Zhang, Zhuo Chen, Shanshan Zhao, and Dacheng Tao. Unimix: Towards domain adaptive and generalizable lidar semantic segmentation in adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14781–14791, 2024. [1](#), [2](#), [3](#), [6](#)
- [29] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020. [3](#)