

Source-Only Cross-Weather LiDAR via Geometry-Aware Point Drop

YoungJae Cheong¹ and Jhonghyun An¹

Abstract—LiDAR semantic segmentation degrades in adverse weather because refraction, scattering, and point dropouts corrupt geometry. Prior work—weather simulation, mixing-based augmentation, domain randomization, and uncertainty or boundary regularization—improves robustness but still overlooks structural vulnerabilities near boundaries, corners, and sparse regions.

We present a Light Geometry-aware adapter. The module aligns azimuth and applies horizontal circular padding to preserve neighbor continuity across the 0° – 360° wrap-around boundary. A local-window K-Nearest Neighbors gathers nearby points and computes simple local statistics, which are compressed into compact geometry-aware cues. During training, these cues drive *region-aware regularization* that stabilizes predictions in structurally fragile areas. The adapter is *plug-and-play*, complements augmentation, and can be enabled only during training with negligible inference cost.

We adopt a *source-only* cross-weather setup where models train on SemanticKITTI and are evaluated on SemanticSTF without target labels or fine-tuning. The adapter improves mIoU by +7.9% over the *data-centric augmentation* baseline and by +0.6% over the *class-centric regularization* baseline. These results indicate that geometry-driven regularization is a key direction for all-weather LiDAR segmentation.

I. INTRODUCTION

LiDAR enables accurate ranging and scene understanding for autonomous driving. Adverse weather such as rain, snow, and fog causes refraction, scattering, and point dropouts that distort point clouds and degrade performance [1], [2]. *Foreground* classes (vehicles, pedestrians, riders) are vulnerable because of their small size and low density. The resulting errors are **safety-critical**. Generic regularizers like Dropout [3] do not address *structural* distortions such as broken boundaries, corner loss, and noise in sparse regions. We study *source-only* cross-weather transfer where models train on SemanticKITTI and are evaluated on SemanticSTF without target labels or fine-tuning.

Prior work has converged on two principal strategies. The first uses data augmentation and *domain randomization* to mimic weather by point dropping, jittering, or physics-based simulation [4], [5], [6]. The second promotes *invariance and consistency* under distribution shifts with mixing-based generalization, uncertainty mitigation, or boundary-preserving regularization. The first line struggles to capture real-world **structural vulnerabilities**, especially discontinuities at object boundaries and in sparse regions. The second often omits explicit modeling of local geometry, which leads to confusion

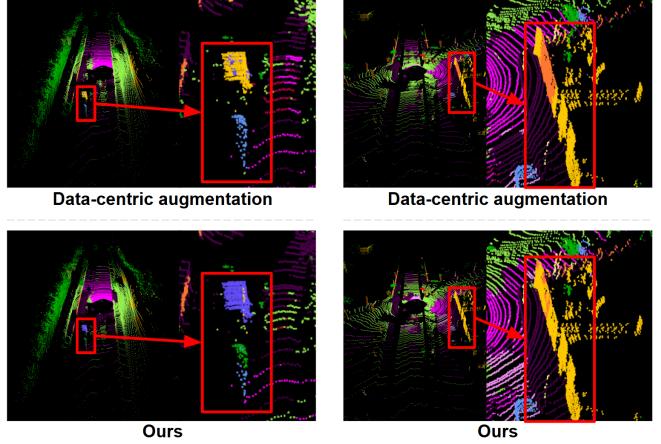


Fig. 1: Data-centric augmentation baseline in adverse weather. Left (a) the baseline merges a bus into a nearby building, while our *Light Geometry-aware adapter* preserves neighbor continuity and segments the bus (blue) instead of the building (yellow). Right (b) the baseline blurs the building–fence boundary and leaks into fence (orange). Our method keeps the boundary sharp and retains the building (yellow) label by respecting local point-to-point structure.

between boundary-adjacent classes. We adopt the *source-only* setting throughout this work.

Fig. 1 illustrates these issues and our remedy. **(a)** On the left, a *data-centric augmentation* baseline [2] merges a *bus* (blue) into a nearby *building* (yellow) because neighbor relations are not explicitly preserved. On the right, **ours** uses a Light geometry-aware adapter with local-window KNN and circular wrapping to maintain neighbor continuity and recovers the correct *bus* segmentation. **(b)** The baseline blurs the *building–fence* boundary and leaks into *fence* (orange). Our method keeps the boundary sharp and retains the *building* (yellow) label by respecting local point-to-point structure.

These observations motivate a *geometry-aware* design that injects two signals into training. The first is *neighbor continuity* from the scanning pattern. The second is *structural risk* at boundaries, corners, and sparse zones. We propose a **Light Geometry-aware adapter**. The module aligns azimuth and applies horizontal circular padding to preserve neighbor continuity across the 0° – 360° seam. It then uses *local-window KNN* to gather nearby points and compute simple local statistics such as offsets and dispersion. These compress into *geometry-aware cues*. During training, the cues drive *region-aware regularization* that prioritizes structurally fragile areas and stabilizes predictions. While PointDR [6] broadens the training distribution via large-scale *domain randomization*, it does not inject geometry into the decision process. Our

¹Gachon University, Seongnam, Republic of Korea
bluebull1777@gachon.ac.kr

¹Gachon University, Seongnam, Republic of Korea
jhonghyun@gachon.ac.kr
Corresponding author: Jhonghyun An

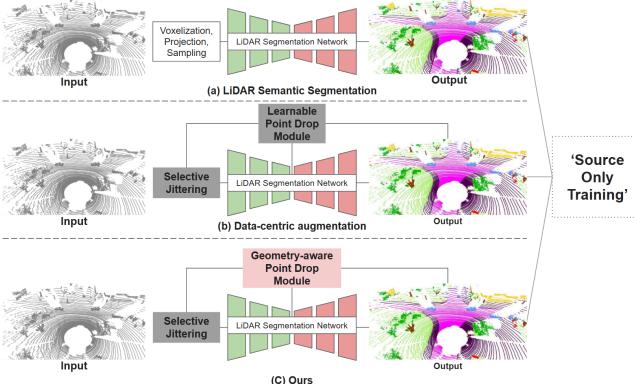


Fig. 2: Three LiDAR segmentation pipelines under the *source-only* training setting. (a) Conventional networks without noise or structure modeling. (b) Data-centric augmentation with selective jittering and a learnable point drop module. (c) Ours with a *Light Geometry-aware adapter*, which enhances region-level dropping by preserving local geometry.

adapter is *complementary*. Augmentation expands coverage, and our cues link local geometry to the learning signal. The module is *plug-and-play* for point- and range-based backbones. On SemanticKITTI→SemanticSTF, it yields consistent gains over strong *data-centric augmentation* baselines. For completeness, we also compare with a *class-centric regularization* baseline [1] and observe improvements. Fig. 2 illustrates how the *Light Geometry-aware adapter* enhances region-level dropping, leading to more robust segmentation under adverse weather.

Our main contributions are:

- 1) **Light Geometry-aware adapter.** We distill the *conventional mixer* [7] into a lightweight module that preserves neighbor continuity via circular padding and *local-window KNN*. By providing geometry-aware cues before the dropping stage, the adapter enhances region-level dropping toward boundaries, corners, and sparse regions while keeping computation low.
- 2) **Geometry-conditioned robustness.** We introduce *region-aware regularization* that uses the derived geometric cues to reduce confusion at boundaries, corners, and sparse regions.

II. RELATED WORK

A. Semantic Segmentation of LiDAR Point Clouds

LiDAR semantic segmentation methods are commonly grouped into *voxel-based*, *projection-based (range-view)*, and *point-based* approaches. **Voxel-based** models use sparse 3D convolutions for large outdoor scenes (SPVCNN, Cylinder3D). Transformer variants introduce spherical or radial attention (SphereFormer) [8], [9], [10]. **Projection-based** methods render scans to range images and apply 2D CNNs (SqueezeSeg, RangeNet++) [11], [12]. **Point-based** models operate directly on points, from PointNet/PointNet++ to kernel-based convolutions and lightweight designs (KPConv, RandLA-Net) [13], [14], [15], [16]. Despite steady accuracy

gains, robustness in adverse weather remains limited, motivating components that expose local geometric structure to the learner.

B. Data Augmentation and Synthesis for LiDAR

Augmentation spans geometric and radiometric perturbations in 3D and range-view spaces. Mixing-based strategies reduce domain gaps by recombining scans or partitions [4], [5]. PolarMix augments along the scanning direction [4]. UniMix builds a bridge domain for adverse-weather adaptation and generalization [5]. Domain adaptation and domain generalization further align features with adversarial or contrastive objectives [17], [18]. These methods broaden or align distributions but do not encode neighborhood continuity or boundary risk. Our adapter complements them by computing local statistics and supplying geometry-conditioned cues to the loss.

C. LiDAR Perception in Adverse Weather

Rain, snow, and fog introduce refraction, scattering, and point dropouts that degrade geometry and segmentation performance [1], [2]. SemanticSTF offers a real-weather benchmark for evaluation [6]. Data-centric approaches improve robustness. Selective Jittering with RL-guided dropping (LPD) targets noise-induced vulnerabilities [2]. TripleMixer adds a Geometry Mixer (GMX) layer to model local geometric relations [7]. Our *Light Geometry-aware adapter* differs by forming non-parametric *geometry-aware cues* from local-window neighborhoods with circular wrapping and injecting them into both the policy and a region-aware regularizer to address noise-driven and structure-driven failures.

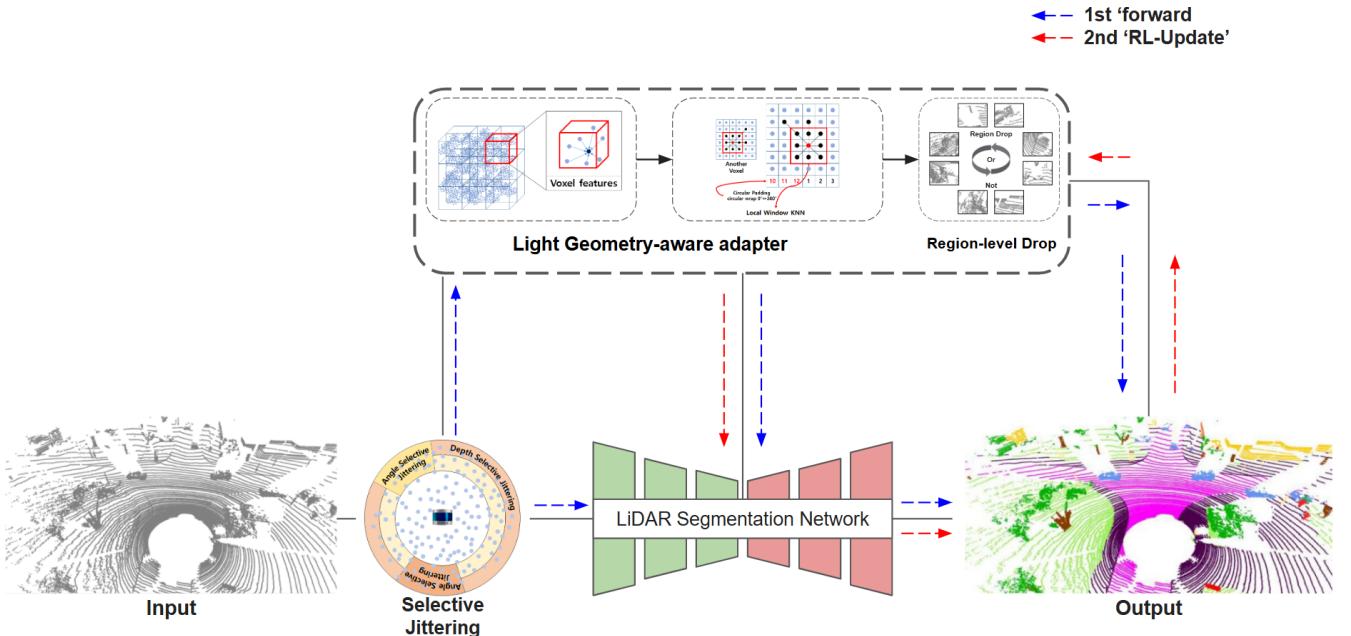
III. METHOD

A. Overview

Our goal is to augment a *data-centric augmentation* pipeline built on Selective Jittering (SJ) and RL-based Learnable Point Drop (LPD) [2] with a **Light Geometry-aware adapter** so that *structural vulnerabilities* under adverse weather are reflected in the training signal (Fig. 3). Inputs are perturbed by SJ to emulate weather-induced corruptions. LPD then selects regions and drop ratios from region-level statistics.

The *Light Geometry-aware adapter* preserves neighbor continuity using *local-window KNN* and horizontal *circular wrapping/padding* at the 0° – 360° seam. It summarizes boundary, corner, and sparse-region geometry into compact *geometry-aware cues*. **The adapter is applied only before LPD, where it enhances region-level dropping by guiding the policy toward structurally fragile areas.** Its cues are injected into the agent *state* so that RL-based decisions become sensitive to geometric risks. This reduces typical confusions such as *bus*↔*building* and broken boundaries such as *building*→*fence* (Fig. 1).

PointDR [6] expands the training distribution via *domain randomization* but does not encode structural risk. By adding local geometric statistics to the agent state,



Our framework

Fig. 3: Overall pipeline. Selective jittering perturbs inputs to emulate weather-induced corruptions. A *Light Geometry-aware adapter*, inserted before learnable point drop, derives compact geometry-aware cues via local-window KNN with circular wrapping and preserves 0° – 360° continuity. These cues guide the drop policy toward **region-level dropping** at boundaries, corners, and sparse structures. After dropping, the backbone processes the perturbed input and produces semantic predictions with improved robustness under adverse weather.

the adapter strengthens LPD’s ability to *prioritize region-level drops* in sensitive areas. The module is *plug-and-play* within the *data-centric augmentation* pipeline and is evaluated under the SemanticKITTI→SemanticSTF *source-only* transfer. For completeness, we also report comparisons with a *class-centric regularization* baseline [1] without altering its pipeline.

B. In Detail

Selective Jittering (SJ) : We adopt SJ as introduced in prior *data-centric augmentation* work [2]. SJ injects *non-uniform*, frame-wise perturbations that mimic adverse weather by selectively applying small offsets or noise to a subset of points, conditioned on simple factors such as range, bearing, and intensity. Unperturbed points remain unchanged. The perturbed input is fed to both the reference and drop branches. This produces supervision that captures *noise-driven* vulnerabilities alongside *structure-driven* ones.

Light Geometry-aware adapter. *Notation.* p_i is the query point. $N_K(i) = \{p_i^1, \dots, p_i^K\}$ are the K neighbors of p_i selected by local-window KNN. μ_i is the windowed local mean of neighbor coordinates. $d_i^{(1)}$ is the distance between p_i and μ_i , which captures off-center geometry near boundaries and corners. $d_i^{(2)}$ is the average deviation of neighbors from μ_i , which reflects the spread or compactness of the local support. ℓ_i^k is the mixed representation of p_i and its neighbor p_i^k . f_i^{pt} is the aggregated feature after attention pooling. g_i is the final geometry-aware cue.

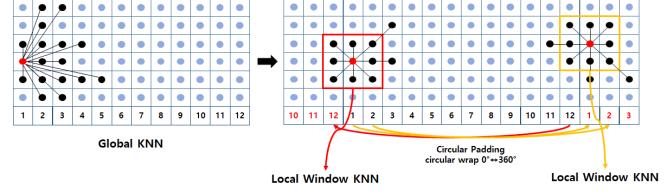


Fig. 4: Global KNN builds neighbors from the full scan and often pulls distant points near the azimuth seam, which increases computation and weakens locality. Our *Light Geometry-aware adapter* uses a horizontal local-window KNN with circular padding to wrap the 0° – 360° seam, preserve edge and corner cues, and reduce computation. This design makes the module light.

Conventional mixers [7] build KNN over a global candidate set on downsampled points. Each query still connects to its closest neighbors, but the search spans the entire scan, which increases computation and often pulls distant points near the azimuth seam. As illustrated in Fig. 4, this *global KNN* design weakens locality and introduces spurious neighbors across boundaries. We instead use a *horizontal local-window KNN*. For each query we restrict candidates to a small window along azimuth and elevation and apply *circular padding* so the horizontal axis is periodic. Queries near the 0° – 360° seam thus include their true neighbors across the seam instead of padded placeholders. This focused search reduces distance evaluations and memory use while preserv-

ing edge and corner cues. Before neighborhood construction, we apply voxel-wise mean pooling to stabilize local features. Inside the window we compute a *windowed local mean* and form two scalar dispersion indicators: $d_i^{(1)}$ measures how far the query lies from the local mean, highlighting off-center geometry at boundaries and corners. $d_i^{(2)}$ measures the average deviation of neighbors from that mean, reflecting how compact or spread the support is. These indicators summarize local geometry without heavy computation and remain robust under varying density. Using these quantities we define point mixing and pooling as follows:

$$\ell_i^k = \phi_p(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus \mu_i \oplus [d_i^{(1)}, d_i^{(2)}]), \quad (1)$$

$$\begin{aligned} \alpha_i^k &= \text{FC}(\ell_i^k), & c_i^k &= \text{Softmax}_k(\alpha_i^k) \\ f_i^{\text{pt}} &= \sum_{k=1}^K c_i^k \ell_i^k. \end{aligned} \quad (2)$$

$$g_i = \text{MLP}(v_i \oplus f_i^{\text{pt}}) \oplus p_i, \quad (3)$$

where g_i forms the *geometry-aware cues*. By replacing the global search with a local-window KNN and circular padding, the module reduces computation, preserves neighbor continuity, and remains lightweight. By design, g_i augments only the *pre-LPD* agent state, so the inference path remains unchanged.

RL-based region-level point drop : We keep the LPD setup from prior *data-centric augmentation* work [2] and *augment* the agent state with the geometry-aware cues above. The original LPD primarily captures *noise-induced* vulnerability via loss and uncertainty statistics. The enhanced state provides an explicit signal for *structural* vulnerability at boundaries, corners, and in sparse zones. This guides the DQN policy to select regions that are *structurally at risk* and couples RL decisions with geometric evidence.

Overall workflow with LPD and the Light Geometry-aware adapter : Algorithm 1 summarizes the procedure. Given SJ-perturbed input \tilde{P} , the adapter \mathcal{A} computes region descriptors $\{(\mathcal{R}_k, F_k)\}$ before LPD, including geometry-aware cues. These descriptors are concatenated with the segmentation network's early predictions and uncertainty to form the Q-network state. An ε -greedy policy selects drop regions and ratios. Rewards and penalties derived from post-drop performance update the Q-network. While PointDR [6] improves generalization through distribution expansion, the adapter is complementary because it injects *local geometric vulnerability* directly into the decision signal.

C. Loss Function

We follow a *data-centric augmentation* training setup [2]. Two branches run in parallel. The reference branch applies selective jittering (SJ) only. The drop branch applies RL-based learnable point drop (LPD) and is augmented with the *Light Geometry-aware adapter*.

Notation. N is the number of points in the batch. C is the number of semantic classes. $Y = \{y_i^{(c)}\}$ is the one-hot ground truth for point i . $\hat{Y} = \{\hat{y}_i^{(c)}\}$ are predicted class

Algorithm 1 Overall workflow of LPD with Light Geometry-aware Adapter

Require: input cloud P , backbone f , decision module Q , drop ratios B , selective jittering SJ , Light Geometry-aware Adapter *adapter*

Ensure: perturbed cloud P' , geometry-aware cues g , region features $\{F_k\}$

- 1: $\tilde{P} \leftarrow SJ(P); u \leftarrow \text{Uncertainty}(f(\tilde{P}))$
 - 2: $g \leftarrow \text{adapter}(\tilde{P})$
 - 3: $\{(\mathcal{R}_k, F_k)\} \leftarrow \text{BuildRegions}(\tilde{P}, g)$
 - 4: $(k^*, b^*) \leftarrow \varepsilon\text{-greedy } \arg \max_{k,b} Q([F_k, u], b)$
 - 5: $P' \leftarrow \text{RegionDrop}(\tilde{P}, \mathcal{R}_{k^*}, B[b^*])$
 - 6: $r \leftarrow \text{SegLoss}(f(P')) - \text{SegLoss}(f(\tilde{P})) - \lambda \cdot \text{gt_ratio}$
 - 7: update Q with $(s = [F_{k^*}, u], a = (k^*, b^*), r, s');$ update f
-

probabilities. $\hat{y}_{i,\text{sj}}^{(c)}$ and $\hat{y}_{i,\text{drop}}^{(c)}$ are predictions from the SJ and LPD branches. w_c are precomputed class weights. $s_i \in [0, 1]$ is the adapter's vulnerability score for point i with larger values indicating higher structural risk. $\kappa \geq 0$ sets the strength of geometry-aware reweighting. $\alpha \geq 1$ and $\eta \geq 0$ weight the post-drop and entropy terms. SJ denotes selective jittering and LPD denotes the RL-based learnable point drop policy.

Base objective : We use a class-weighted cross-entropy to address class imbalance

$$\mathcal{L}_{\text{CE}}(\hat{Y}, Y) = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C w_c y_i^{(c)} \log \hat{y}_i^{(c)}. \quad (4)$$

The reference loss $\mathcal{L}_{\text{before}}$ is \mathcal{L}_{CE} applied to the SJ-only predictions \hat{Y}_{sj} .

Geometry-aware reweighting after drop : The post-drop loss up-weights points that the adapter marks as structurally vulnerable through a score $s_i \in [0, 1]$

$$\mathcal{L}_{\text{after}} = \frac{1}{N} \sum_{i=1}^N (1 + \kappa s_i) \left[-\sum_{c=1}^C w_c y_i^{(c)} \log \hat{y}_{i,\text{drop}}^{(c)} \right]. \quad (5)$$

Unlike distribution expansion via *domain randomization* [6], this reweighting injects *local geometry* into the learning signal by allocating more loss to boundaries, corners, and sparse regions.

Entropy regularization : To discourage overconfidence and to couple the two branches, we add

$$\mathcal{L}_{\text{ent}} = -\frac{1}{2N} \sum_{i=1}^N \left[\sum_{c=1}^C \hat{y}_{i,\text{sj}}^{(c)} \log \hat{y}_{i,\text{sj}}^{(c)} + \sum_{c=1}^C \hat{y}_{i,\text{drop}}^{(c)} \log \hat{y}_{i,\text{drop}}^{(c)} \right]. \quad (6)$$

Final loss :

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{before}} + \alpha \mathcal{L}_{\text{after}} + \eta \mathcal{L}_{\text{ent}}, \quad (7)$$

where $\alpha \geq 1$ emphasizes the post-drop branch and $\eta \geq 0$ controls the strength of the entropy term.

TABLE I: Comparison of methods on the SemanticKITTI→SemanticSTF benchmark. Our method yields consistent improvements in per-class IoU and mIoU over strong baselines. This indicates that region-aware point dropping provides additional robustness in cross-weather adaptation.

Method	car	bi.cle	mt.cle	truck	bus	pers	bi.clst	mt.clst	road	parki.	sidew.	oth.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	0.2	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	15.7	24.4
Dropout [3]	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	23.3	0.6	19.7	20.0	0.0	0.0	59.3	10.7	32.0	7.2	70.2	45.2	57.1	47.9	28.2	16.2	25.9	
PolarMix [4]	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	58.7	53.8	20.5	45.4	29.3	15.8	26.6
MMD [17]	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	22.4	47.9	28.2	16.2	26.9
PCL [18]	65.9	0.0	0.4	17.3	8.4	8.4	0.0	0.0	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 [6]	67.3	0.0	4.5	19.9	18.8	2.7	20.0	0.0	62.6	12.9	36.8	43.8	73.3	43.8	56.4	32.2	45.7	28.7	27.4	26.4
DGLSS [20]	72.6	0.1	11.7	29.4	13.7	48.3	0.5	21.2	65.0	20.2	36.5	3.8	78.9	51.8	57.0	36.4	42.7	26.9	34.9	34.6
UniMix [5]	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	23.5	39.4	23.5	31.5	31.5
DGUIL [21]	77.9	1.0	19.1	26.0	9.7	46.3	0.6	9.3	69.1	9.8	38.6	9.4	73.3	51.2	59.0	31.8	50.8	31.8	22.3	31.4
LiDARWeather [2]	83.1	1.2	17.2	30.5	18.4	47.5	1.07	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
No Thing, Nothing [1]	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	39.8	38.9	38.9
Ours	85.01	9.88	23.88	38.7	23.58	46.34	9.65	28.8	65.36	13.13	37.54	1.81	77.82	50.64	65.16	39.39	53.83	32.66	40.65	39.15

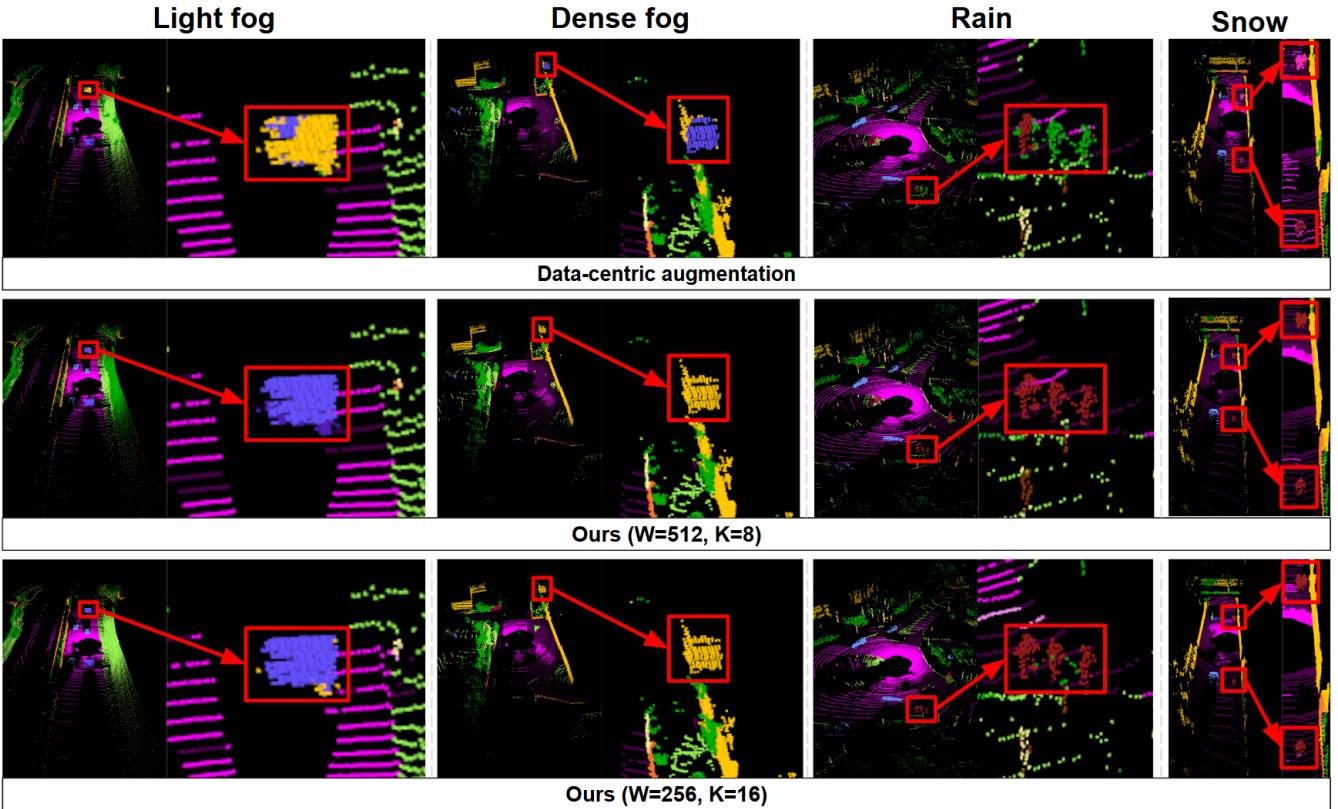


Fig. 5: Qualitative comparison on SemanticSTF across four adverse-weather conditions: **light fog**, **dense fog**, **rain**, and **snow**. The top row shows the *data-centric augmentation* baseline. The middle and bottom rows show our method with ($W = 512, K = 8$) and ($W = 256, K = 16$). The *Light Geometry-aware adapter* preserves neighbor continuity and reduces class confusion at boundaries.

IV. EXPERIMENTS

A. Experimental Setup

We evaluate under the **SemanticKITTI**→**SemanticSTF** setting. We follow a *source-only* cross-weather setting where models train on **SemanticKITTI** and are evaluated on **SemanticSTF** with no target labels or fine-tuning. All methods use the same backbone and training schedule. We report *mIoU* and *per-class IoU* over the standard 19

evaluation classes.

We compare against augmentation baselines (Dropout, generic perturbations, PolarMix [4]), domain generalization or adaptation methods (PCL [18], DGLSS [20], UniMix [5], DGUIL [21]), domain randomization (PointDR [6]), a *data-centric augmentation* baseline [2], and a *class-centric regularization* baseline [1]. Unless noted, hyperparameters follow previous work. For our adapter, we sweep the *local-window* size W and the number of neighbors K .

TABLE II: Ablation of the Light Geometry-aware Adapter hyperparameters (window W , neighbors K) on SemanticKITTI→SemanticSTF. We report IoU on safety-critical classes (*car*, *bicycle*, *motorcycle*, *truck*, *bus*, *person*) and overall mIoU.

Method	car	bi.cle	mt.cle	truck	bus	pers	mIoU
LiDARWeather [2]	83.10	1.20	17.20	30.50	18.40	47.50	36.30
No Thing, Nothing [1]	83.30	3.70	31.30	36.20	18.20	53.30	38.90
Ours ($W=256$, $K=8$)	85.60	7.09	27.98	31.52	26.26	51.10	37.77
Ours ($W=256$, $K=16$)	85.01	9.88	23.88	38.70	23.58	46.34	39.15
Ours ($W=512$, $K=8$)	84.52	8.35	20.53	40.95	27.28	48.13	38.07
Ours ($W=512$, $K=16$)	85.97	6.76	19.10	37.73	30.61	49.53	37.05

B. Overall Results

Summary. As shown in Table I, injecting geometric information with the *Light Geometry-aware adapter* consistently outperforms the *data-centric augmentation* baseline. We obtain **+7.9% mIoU** over the *data-centric augmentation* baseline and a further **+0.6%** over the *class-centric regularization* baseline, setting a new state of the art on this transfer. Gains are pronounced on safety-critical classes such as *car*, *bus*, and *person*. Boundary-sensitive classes such as *building* and *fence* also improve. Qualitatively, the adapter reduces *bus*↔*building* confusions. It also mitigates *building*→*fence* boundary breaks by preserving neighbor continuity with a local-window KNN and circular padding (Fig. 5).

Per-class Analysis. Improvements are observed in 15 of 19 classes. Large gains appear for safety-related and large rigid categories *truck* +8.20, *bus* +5.18, *bicyclist* +8.58, *bicycle* +8.68, *motorcycle* +6.68, *motorcyclist* +10.0, and *car* +1.91. Background categories change modestly *building* +0.42, *fence* 0.16, *other-ground* 2.79. Boundary-sensitive items improve *trunk* +2.19, *terrain* +6.13, *pole* +1.56, *traffic-sign* +4.85. These trends indicate that geometry-aware cues stabilize region selection in the dropping policy and strengthen the backbone representation.

Qualitative Analysis As shown as Fig. 5, In *light fog*, the *data-centric augmentation* baseline merges the *bus* (blue) into a nearby *building* (yellow). Our method preserves neighbor continuity and segments the entire region as *bus* (blue). In *dense fog*, the baseline flips *building* (yellow) to *bus* (blue). Our method keeps the boundary intact and retains the correct *building* label. Under *rain*, the baseline confuses *person* (dark red) with *vegetation* (green), especially near thin structures and range discontinuities. Our method restores the correct *person* label and reduces leakage into vegetation. Under *snow*, the baseline often mislabels *person* (dark red) as *parking* (pink). Our method predicts *person* consistently and produces cleaner contours around the feet and adjacent ground. These results align with the per-class gains and show how the *Light Geometry-aware adapter* reduces class flips and boundary breaks by maintaining neighbor continuity with a local-window KNN and circular padding.

C. Ablation Study

Table II varies the local window $W \in \{256, 512\}$ and the number of neighbors $K \in \{8, 16\}$. All configurations

surpass the *data-centric augmentation* baseline. The best mIoU is **39.15** at $W=256$ and $K=16$. A compute-friendly option, $W = 512$ and $K = 8$, reaches **38.07**. Increasing K aggregates more neighbors and benefits large objects, but very large neighborhoods may blur boundary detail. Enlarging W expands spatial context. A mid-range K offers a good balance. We report $W=256$, $K=16$ as the default and include $W=512$, $K=8$ as a practical choice.

V. CONCLUSIONS

We introduced a **Light Geometry-aware adapter** that injects local, structure-sensitive cues into the LPD training loop. The adapter uses *local-window KNN* with horizontal *circular padding* to preserve neighbor continuity across the 0° – 360° seam. It guides region-level dropping toward edges, corners, and sparse zones and reduces computation compared with *conventional mixers* [7] by avoiding global neighbor search. On SemanticKITTI→SemanticSTF, it improves mIoU by **+7.9%** over the *data-centric augmentation* baseline. It also exceeds the *class-centric regularization* baseline by **+0.6%**. The module is *plug-and-play* and *model-agnostic*. These properties support practical deployment under adverse weather. *Future work* includes adaptive selection of the window size and the neighbor count K , few-label adaptation, extensions to 3D detection and multi-sensor fusion, and real-time on-board evaluation.

REFERENCES

- [1] J. Park, K. Kim, Y. Kim, J. Lee, and H. Shim, “No Thing, Nothing: Highlighting Safety-Critical Classes for Robust LiDAR Semantic Segmentation in Adverse Weather,” in *Proc. CVPR*, 2025.
- [2] J. Park, K. Kim, and H. Shim, “Rethinking Data Augmentation for Robust LiDAR Semantic Segmentation in Adverse Weather,” in *Proc. ECCV*, 2024.
- [3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res. (JMLR)*, vol. 15, pp. 1929–1958, 2014.
- [4] A. Xiao, J. Gu, Y. Zhang, S. Chen, and Y. Cui, “PolarMix: A General Data Augmentation Technique for LiDAR Point Clouds,” in *NeurIPS*, 2022, pp. 11035–11048.
- [5] H. Zhao, H. Qiu, Y. Yao, Y. Wang, S. Wang, and Z. Liu, “UniMix: Towards Domain Adaptive and Generalizable LiDAR Semantic Segmentation in Adverse Weather,” in *Proc. CVPR*, 2024.
- [6] A. Xiao, J. Huang, W. Xuan, R. Ren, K. Liu, D. Guan, A. El Saddik, S. Lu, and E. P. Xing, “PointDR: 3D Semantic Segmentation in the Wild via Domain Randomization,” in *Proc. CVPR*, 2023, pp. 9382–9392.
- [7] X. Zhao, Y. Chen, H. Zhao, Y. Qiao, and D. Lin, “TripleMixer: A 3D Point Clouds Denoising Model for Adverse Weather,” *arXiv:2408.13802*, 2024.
- [8] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Su, and S. Han, “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution,” in *Proc. ECCV*, 2020.
- [9] H. Zhou, Y. Zhu, X. Cheng, D. Lu, X. Yang, and R. Yang, “Cylinder3D: An Effective 3D Framework for Driving-Scene LiDAR Semantic Segmentation,” *arXiv:2008.01550*, 2020.
- [10] X. Lai, Y. Chen, S. Jiang, X. Liu, S. Pu, and H. Xiong, “Spherical Transformer for LiDAR-Based 3D Recognition,” in *Proc. CVPR*, 2023.
- [11] B. Wu, A. Wan, X. Yue, and K. Keutzer, “SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Clouds,” in *Proc. ICRA*, 2018.
- [12] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “RangeNet++: Fast and Accurate LiDAR Semantic Segmentation,” in *Proc. IROS*, 2019.
- [13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” in *Proc. CVPR*, 2017.

- [14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” in *NeurIPS*, 2017.
- [15] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “KPConv: Flexible and Deformable Convolution for Point Clouds,” in *Proc. ICCV*, 2019.
- [16] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds,” in *Proc. CVPR*, 2020.
- [17] H. Li, S. J. Pan, S. Wang, and A. C. Kot, “Domain Generalization with Adversarial Feature Learning,” in *Proc. CVPR*, 2018, pp. 5400–5409.
- [18] X. Yao, Y. Bai, X. Zhang, Y. Zhang, Q. Sun, R. Chen, R. Li, and B. Yu, “PCL: Proxy-based Contrastive Learning for Domain Generalization,” in *Proc. CVPR*, 2022, pp. 7097–7107.
- [19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences,” in *Proc. ICCV*, 2019.
- [20] H. Kim, Y. Kang, C. Oh, and K.-J. Yoon, “Single-Domain Generalization for LiDAR Semantic Segmentation,” in *Proc. CVPR*, 2023, pp. 17587–17598.
- [21] P. He, L. Jiao, L. Li, X. Liu, F. Liu, W. Ma, S. Yang, and R. Shang, “Domain Generalization-aware Uncertainty Introspective Learning for 3D Point Cloud Segmentation,” in *Proc. ACM MM*, 2024, pp. 651–660.
- [22] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Playing Atari with Deep Reinforcement Learning,” *arXiv:1312.5602*, 2013.