# Supplementary Materials for Rethinking LiDAR Domain Generalization: Single Source as Multiple Density Domains

Jaeyeul Kim, Jungwan Woo, Jeonghoon Kim, and Sunghoon Im

DGIST, Daegu, South Korea
{jykim94, friendship1, jeonghoon, sunghoonim}@dgist.ac.kr

This supplementary material provides a more detailed description of the proposed method. We also present a variety of additional experimental results, including per-class IoU and qualitative results.

## 1 Implementation Details

**Augmentation** We set the range of random translation for enhanced-Mix3D to $[-25\,\mathrm{m}, 25\,\mathrm{m}]$ along the $x$-axis and the range of random rotation to $[-30^\circ, 30^\circ]$ around the $z$-axis. After merging two point clouds, the estimated density values for each point are separately calculated based on the origin of each point cloud and then these values are aggregated. Alg. 1 provides a detailed description of the enhanced-Mix3D. Regarding beam sampling augmentation [17], we reduce the LiDAR channels to half by selecting only even-numbered vertical beams. We also incorporate standard 3D point cloud augmentation methods such as scaling $[0.9, 1.1]$, rotation $[0^\circ, 360^\circ]$ around the $z$-axis, flipping, and translation $[-0.1\,\mathrm{m}, 0.1\,\mathrm{m}]$.

**Algorithm 1** Enhanced-Mix3D Density Augmentation

**Input:** Two point clouds $\mathbf{P}_1 \in \mathbb{R}^{N\times 3}$ and $\mathbf{P}_2 \in \mathbb{R}^{M\times 3}$
**Output:** Mixed point cloud $\mathbf{P}_o \in \mathbb{R}^{(N+M)\times 3}$ and it's corresponding density embedding $\mathcal{D} \in \mathbb{R}^{(N+M)\times 4}$

```
1: R_1 ← Z-axis-RandomRotation([−30°, 30°])
2: R_2 ← Z-axis-RandomRotation([−30°, 30°])
3: t ← X-axis-RandomTranslation([−25m, 25m])
4: P̃_2 ← R_2(R_1 P_2 + t)                 ▷ Transform point cloud
5: P_o ← [P_1; P̃_2]                       ▷ Mix coordinates
6: O_2 ← R_2 t                             ▷ sensor center of P̃_2
7: for p_i ∈ P_o do
8:     d_1 ← DensityOf(p_i)                ▷ From Eq. 5 in the main paper
9:     d_2 ← DensityOf(p_i − O_2)          ▷ From Eq. 5 in the main paper
10:    D_i ← sqrt(d_1^2 + d_2^2)           ▷ Mix densities
11: end for
```

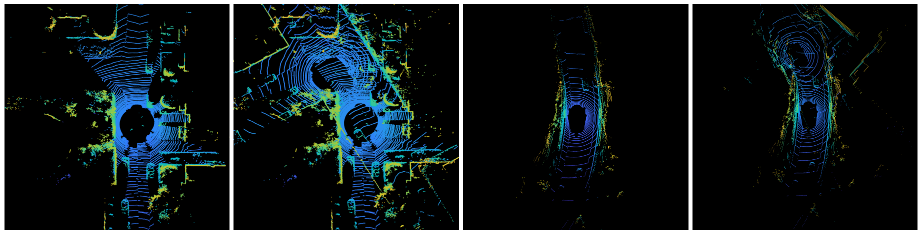
J. Kim and J. Woo—Both authors contributed equally to this work.

**Reservoir Sampling** We employ the density clipping technique to ensure that density values during the test phase do not surpass the $90^{\text{th}}$ or fall below the $10^{\text{th}}$ percentile values established from the training dataset. Due to the resource-intensive nature of sorting the entire training dataset for percentiles, we adopt an on-the-fly approach for percentile estimation using reservoir sampling [14], as detailed in Alg. 2. We sample density values into reservoir memory $\mathcal{R}$ with the size of $N = 1000$, then approximate percentiles in the memory. To prevent decelerating the training process, we estimate percentiles only when sufficient samples are accumulated, subsequently updating them with a cumulative moving average. For the test phase, the established $90^{\text{th}}$ and $10^{\text{th}}$ percentile values, denoted as $\hat{p}$, serve as model parameters.

**Algorithm 2** Point cloud Reservoir Sampling Algorithm

**Initialize** $\mathcal{R}[1,...,N]$, $\hat{p} \leftarrow 0$, $c \leftarrow 0$ ▷ $\mathcal{R}$ is the reservoir memory, $\hat{p}$ is the estimated percentile, $N$ is the max size of reservoir, $c$ is the number of resets

**for** point cloud $\mathcal{P}$ in $\mathcal{S}$ **do** ▷ $\mathcal{S}$ is the stream of training point cloud data<br>
    $\mathcal{D} \leftarrow$ calculate density from point cloud $\mathcal{P}$ ▷ From Eq. 5 in the main paper<br>
    $\rho \leftarrow \frac{N}{k}$ ▷ $k$ is the number of points processed so far<br>
    $s \leftarrow \min(N, \text{round}(\rho \times |\mathcal{D}|))$ ▷ $s$ is the number of point to be sampled<br>
    **if** $s < 1$ **then**<br>
        $s \leftarrow N$<br>
        $k \leftarrow 0$ ▷ Reset the count of processed points<br>
        $c \leftarrow c + 1$<br>
        $p_{\text{curr}} \leftarrow$ Calculate percentile of $\mathcal{R}$<br>
        $\hat{p} \leftarrow \hat{p} + \frac{p_{\text{curr}} - \hat{p}}{c}$ ▷ Calculate Cumulative Moving Average<br>
    **end if**<br>
    Select $s$ random density values of points from $\mathcal{D}$<br>
    Replace $\mathcal{R}$ with these points<br>
    $k \leftarrow k + |\mathcal{D}|$<br>
    **if** $c = 0$ **then** ▷ Exception for first loop<br>
        $\hat{p} \leftarrow$ Calculate percentile of $\mathcal{R}$<br>
    **end if**<br>
    ForwardModel($\mathcal{P}$, $\hat{p}$) ▷ Model forward with estimated percentile<br>
**end for**



(a) Raw point cloud (K) (b) Augmented point cloud (K) (c) Raw point cloud (N) (d) Augmented point cloud (N)

**Fig. 1:** Results of point cloud transformation through the proposed augmentation scheme on SemanticKITTI (K) and nuScenes (N) datasets.

**Table 1:** Comparison of per-class performance (IoU) with DGLSS. All methods are trained using the MinkowskiNet on the SemanticKITTI and tested on SemanticKITTI (K), nuScenes (N), and Waymo (W) datasets. The best and the second best results are highlighted in **bold** and underline, respectively.

| | Method | Car | Bicycle | Motor-cycle | Truck | Other vehicle | Pedes-trian | Drivable surface | Side-walk | Walk-able | Vege-tation | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | Base | 91.23 | 10.04 | 35.69 | 52.89 | 37.95 | 40.99 | 83.86 | 62.78 | 66.34 | **91.33** | 57.31 |
| | DGLSS [6] | **92.76** | 11.99 | 27.09 | **72.50** | **45.95** | 36.39 | 84.76 | 65.64 | 67.98 | 91.28 | 59.62 |
| | Ours | 91.86 | **21.65** | **48.11** | 26.84 | 40.04 | **62.32** | **91.90** | **79.65** | **72.29** | 90.31 | **62.50** |
| N | Base | 68.91 | 2.51 | 12.18 | 11.30 | 20.35 | 29.47 | 80.17 | 31.91 | 40.19 | 77.18 | 37.42 |
| | DGLSS [6] | **76.36** | 1.51 | **35.18** | 26.47 | 25.49 | 37.09 | 82.03 | 38.12 | 44.20 | 81.79 | 44.83 |
| | Ours | 75.29 | **3.84** | 29.28 | **30.11** | **31.37** | **51.69** | **87.85** | **50.02** | **51.10** | **83.77** | **49.43** |
| W | Base | 72.12 | 2.52 | 4.52 | 7.77 | 13.36 | 40.86 | 64.92 | 30.12 | **34.84** | 81.40 | 35.24 |
| | DGLSS [6] | 82.26 | **4.85** | 9.72 | 16.80 | 17.67 | 52.55 | **68.20** | 35.91 | 33.33 | 85.41 | 40.67 |
| | Ours | **83.66** | 4.69 | **13.99** | **25.79** | **18.71** | **61.53** | 63.19 | **38.49** | 28.66 | **88.54** | **42.73** |

## 2 Extended Results and Analysis

**Augmentation** Fig. 1 illustrates the results of applying the proposed augmentation scheme on the SemanticKITTI [1] and the nuScenes [2] datasets.

**Per-class Evaluation Results** We provide a comparative analysis of our approach against other methods by examining per-class IoU metrics. Tab. 1 and Tab. 2 detail the per-class results in the DGLSS setting [6]. Tab. 3 and Tab. 4 present the per-class performance in the LiDomAug [11] setting, when using MinkNet42 [3] and C&L [20] as the backbone, respectively. The comparison results in Tab. 3, excluding our proposed method, are sourced from the DCF-Net [22] and LiDomAug papers. The results in Tab. 4 are obtained from the LiDAR-UDA paper [12] and from the authors of LiDomAug.

As shown in Tab. 1, the proposed method outperforms DGLSS [6] in 8 out of 10 classes when trained on SemanticKITTI (64-ch) and tested on nuScenes (32-ch), with an average performance gain of 10.26% over DGLSS. In the (K→W) scenario, our method demonstrates higher performance in 7 classes compared to DGLSS, showing an average improvement of 5.07%. Tab. 2 illustrates the

**Table 2:** Class-wise IoU of the proposed method in the DGLSS setting.

| Scenario | Car | Bicycle | Motor-cycle | Truck | Other vehicle | Pedes-trian | Drivable surface | Side-walk | Walk-able | Vege-tation | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N→N | 89.11 | 7.43 | 44.51 | 69.21 | 74.69 | 67.60 | 96.58 | 75.71 | 73.43 | 83.37 | 68.16 |
| N→K | 86.63 | 4.83 | 29.35 | 24.80 | 13.45 | 38.70 | 80.97 | 48.79 | 52.52 | 85.14 | 46.52 |
| N→W | 72.64 | 2.37 | 12.24 | 22.58 | 35.64 | 66.27 | 77.85 | 48.41 | 37.75 | 84.03 | 45.98 |
| W→W | 93.70 | 56.55 | 66.16 | 50.83 | 64.96 | 91.55 | 94.54 | 72.39 | 75.27 | 95.55 | 76.15 |
| W→K | 90.97 | 32.18 | 58.61 | 17.83 | 8.16 | 68.51 | 84.01 | 57.09 | 62.29 | 91.03 | 57.07 |
| W→N | 77.14 | 9.83 | 26.82 | 58.70 | 53.77 | 62.42 | 93.36 | 50.65 | 49.43 | 85.36 | 56.75 |

**Table 3:** Comparison of per-class performance (IoU) with DA/DG methods. The symbol † denotes methods necessitating ego-motion knowledge. The best and the second best results are highlighted in **bold** and underline, respectively.

| | Method | DA | Car | Bicycle | Motor-cycle | Truck | Other vehicle | Pedes-trian | Drivable surface | Side-walk | Terrain | Vege-tation | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K→N | Base | | 50.7 | 5.7 | 6.0 | 21.7 | 24.8 | 29.2 | **89.1** | 42.0 | 23.1 | 85.8 | 37.8 |
| | FADA [16] | ✓ | 69.1 | 1.9 | 21.4 | 42.9 | 20.5 | 12.6 | 84.3 | 43.6 | 38.3 | 80.0 | 41.5 |
| | CLAN [8] | ✓ | 73.1 | 1.7 | 16.3 | 31.4 | 19.1 | 13.5 | 85.7 | 45.7 | 50.1 | 81.4 | 41.8 |
| | DAST [21] | ✓ | 73.2 | 2.1 | 24.4 | 44.6 | 22.4 | 18.2 | 77.6 | 39.5 | 43.2 | 81.2 | 42.6 |
| | DCF-Net [22] | ✓ | 74.9 | 1.8 | 28.9 | 33.9 | 21.6 | 19.9 | 84.5 | 44.6 | 46.3 | 83.6 | 44.0 |
| | CutMix [23] | | 75.5 | 0.1 | 14.0 | 26.6 | 22.6 | 3.9 | 86.6 | 36.5 | 19.7 | 85.6 | 37.1 |
| | Copy-Paste [5] | | 77.9 | 3.1 | 11.1 | 21.7 | 31.2 | 7.8 | 88.0 | 38.8 | 19.6 | 86.2 | 38.5 |
| | Mix3D [9] | | 72.1 | 0.0 | 34.8 | 11.7 | 26.4 | 28.5 | 83.3 | 41.0 | 46.4 | **86.5** | 43.1 |
| | PolarMix [19] | | 74.1 | 1.7 | 41.9 | 26.9 | 23.8 | 30.5 | 85.1 | 42.7 | 45.3 | 86.2 | 45.8 |
| | LiDomAug† [11] | | **79.2** | **5.8** | 28.0 | **49.3** | **32.1** | 13.8 | 88.0 | 42.0 | 35.4 | 85.1 | 45.9 |
| | Ours | | 76.0 | 5.7 | **42.3** | 33.9 | 26.7 | **49.5** | 88.5 | **49.8** | **50.2** | 78.2 | **50.1** |
| N→K | Base | | 78.5 | 0.0 | 8.2 | 3.4 | 11.1 | 34.5 | 66.3 | 35.8 | 39.4 | 84.2 | 36.1 |
| | FADA [16] | ✓ | 83.3 | 39.0 | 24.5 | 8.7 | 3.6 | 33.6 | 72.7 | 39.4 | 36.0 | 82.1 | 42.3 |
| | CLAN [8] | ✓ | 86.0 | 21.4 | 9.2 | 13.5 | 5.7 | 42.8 | 77.8 | **51.0** | 47.6 | 84.6 | 43.9 |
| | DAST [21] | ✓ | 90.8 | **40.8** | 24.0 | 11.5 | 5.2 | 41.7 | 76.7 | 45.3 | 48.5 | 84.5 | 46.9 |
| | DCF-Net [22] | ✓ | 91.7 | 34.3 | 24.5 | 18.5 | 11.6 | **49.1** | 78.6 | 47.5 | 49.8 | 86.4 | **49.2** |
| | CutMix [23] | | 81.2 | 0.0 | 5.3 | 9.1 | **17.4** | 11.8 | 73.6 | 45.5 | 46.8 | 85.7 | 37.6 |
| | Copy-Paste [5] | | 85.7 | 0.0 | 8.2 | 12.8 | 6.5 | 28.6 | 80.8 | 47.4 | 53.8 | 87.2 | 41.1 |
| | Mix3D [9] | | **93.1** | 10.4 | 31.3 | 17.0 | 14.1 | 34.2 | 71.8 | 40.7 | 44.6 | **89.5** | 44.7 |
| | PolarMix [19] | | 75.9 | 19.4 | 19.7 | 9.6 | 3.0 | 18.3 | 75.0 | 43.1 | 48.9 | 77.8 | 39.1 |
| | LiDomAug† [11] | | 92.6 | 31.6 | **42.5** | 21.6 | 6.4 | 34.4 | 70.0 | 47.1 | **59.4** | 77.5 | 48.3 |
| | Ours | | 89.5 | 4.7 | 29.5 | **22.4** | 6.3 | 33.7 | **81.9** | 50.7 | 56.5 | 88.2 | 46.3 |

class-wise IoU of the proposed method in various scenarios within the DGLSS setting. Furthermore, as presented in Tab. 3, our method shows a 13.9% higher performance in the (K→N) scenario compared to the state-of-the-art domain adaptation method [22]. While the proposed method shows a slightly decreased performance by 6.3% in the (N→K) scenario, considering that our method is a domain generalization approach that does not require additional fine-tuning, this comparable performance to the domain adaptation methods demonstrates the superiority of our approach. As indicated in Tab. 4, even when using C&L as the backbone, the proposed method outperforms both the latest domain adaptation [7, 12, 18, 20] and domain augmentation [11] methods, demonstrating the superiority of the proposed approach.

**Table 4:** Comparison of per-class performance (IoU) with DA/DG methods. All methods are based on C&L architecture. The symbol † denotes methods necessitating ego-motion knowledge. The best and the second best results are highlighted in **bold** and underline, respectively.

| | Method | DA | Car | Bicycle | Motor-cycle | Truck | Other vehicle | Pedes-trian | Drivable surface | Side-walk | Terrain | Vege-tation | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K→N | Base | | 55.3 | 0.1 | 3.4 | 2.5 | 4.5 | 4.7 | 79.2 | 32.2 | 21.7 | 74.0 | 27.8 |
| | SWD [7] | ✓ | - | - | - | - | - | - | - | - | - | - | 27.7 |
| | 3DGCA [18] | ✓ | - | - | - | - | - | - | - | - | - | - | 27.4 |
| | C&L† [20] | ✓ | - | - | - | - | - | - | - | - | - | - | 31.6 |
| | LiDAR-UDA† [12] | ✓ | **73.5** | 0.9 | 15.9 | 0.9 | 25.7 | **40.8** | **87.4** | 42.3 | 47.9 | **83.2** | 41.8 |
| | LiDomAug† [11] | | - | - | - | - | - | - | - | - | - | - | 39.2 |
| | Ours | | **73.5** | **1.7** | **35.9** | **34.2** | **27.4** | 40.3 | 85.1 | **45.4** | **48.5** | 78.9 | **47.1** |
| N→K | Base | | 58.0 | 0.5 | 4.4 | 3.8 | 5.0 | 12.6 | 33.8 | 2.8 | 30.1 | 80.9 | 23.2 |
| | SWD [7] | ✓ | - | - | - | - | - | - | - | - | - | - | 24.5 |
| | 3DGCA [18] | ✓ | - | - | - | - | - | - | - | - | - | - | 23.9 |
| | C&L† [20] | ✓ | - | - | - | - | - | - | - | - | - | - | 33.7 |
| | LiDAR-UDA† [12] | ✓ | **86.2** | 0.0 | 13.9 | 9.3 | 3.2 | 16.5 | 65.7 | 6.1 | 54.1 | 85.7 | 34.0 |
| | LiDomAug† [11] | | 83.7 | 0.0 | 16.6 | 5.4 | **13.4** | **29.6** | 67.4 | 25.4 | **56.5** | 81.2 | 37.9 |
| | Ours | | 84.6 | **8.7** | **27.6** | **15.8** | 2.6 | 27.6 | **75.6** | **27.6** | 46.8 | **86.1** | **40.3** |

**Experiments with the additional setting** We further compare our voxel-based method with the results of applying Domain Adaptation (DA) methods [7, 10, 13, 15] to a range-view backbone [4], using the 11-class setting outlined in Rochan *et al.* [10]. The comparative results in Tab. 5 are directly sourced from the original paper [10], except for the MinkNet42 base and our method. The results show that the MinkNet42 base model exhibits superior generalization performance compared to the DA methods using range-view architectures [7, 10, 13, 15].

With the application of our method, MinkNet42's performance further increases, recording a +13.1% uplift in the (K→N) scenario and a remarkable +44.7% increase in the (N→K) scenario relative to its baseline performance. Compared to [10], our method shows a substantial performance surge, achieving +33.0% and +95.7% improvements in the respective scenarios. This significant enhancement can be attributed to the inherent disadvantages of range-view-based backbone, which struggle with performance dips due to the distortion in the convolutional layer's receptive field caused by the variation in input range image resolution. Moreover, in scenarios like (K→N), where inputs include range images with extensive blank areas not encountered during training, range-view-based backbone face pronounced performance setbacks, a limitation our voxel-based approach effectively circumvents.

**Qualitative Results** To further demonstrate the effectiveness of our proposed method, we provide various qualitative results. We compare the baseline and DGLSS [6] with our proposed method, in accordance with the experimental settings of DGLSS, using the SemanticKITTI (K), Waymo (W), and nuScenes (N) datasets. Fig. 2 and Fig. 3 show the qualitative results after training on nuScenes

**Table 5:** Comparison with domain adaptation methods applied to a range-view backbone.

| | Backbone | Method | Car | Bicycle | Motor-cycle | Other vehicle | Pedes-trian | Truck | Drivable surface | Side-walk | Terrain | Vege-tation | Man-made | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K→N | SalsaNext (Range-view) | Base | 35.7 | 0.2 | 0.4 | 5.7 | 7.5 | 8.1 | 73.8 | 15.0 | 14.9 | 8.3 | 51.4 | 20.1 |
| | | CORAL [13] | 51.0 | 0.9 | 6.0 | 4.0 | 25.9 | **29.9** | 82.6 | 27.1 | 27.0 | 55.3 | 56.7 | 33.3 |
| | | MEnt [15] | 57.4 | 2.2 | 4.6 | 6.4 | 22.6 | 19.3 | 82.3 | 28.8 | 29.9 | 46.8 | 64.2 | 33.1 |
| | | AEnt [15] | 57.4 | 1.1 | 8.6 | 6.7 | 24.0 | 10.1 | 81.0 | 25.4 | 26.6 | 34.2 | 58.9 | 30.4 |
| | | (M+A)Ent [15] | 57.3 | 1.1 | 2.3 | 6.8 | 23.4 | 7.9 | 83.5 | 32.6 | 31.8 | 43.3 | 62.3 | 32.0 |
| | | SWD [7] | 45.3 | 2.1 | 2.2 | 3.4 | 25.9 | 10.6 | 80.7 | 26.5 | 30.1 | 43.9 | 60.2 | 30.1 |
| | | Rochan *et al.* [10] | 54.4 | 3.0 | 1.9 | 7.6 | 27.7 | 15.8 | 82.2 | 29.6 | 34.0 | 57.9 | 65.7 | 34.5 |
| | MinkNet42 (Voxel) | Base | 69.7 | **5.3** | 26.2 | **18.3** | 26.5 | 17.5 | 82.2 | 35.6 | 30.6 | 65.2 | 69.7 | 40.6 |
| | | Ours | **75.3** | 1.6 | **44.9** | 15.7 | **28.5** | 19.4 | **87.8** | **48.0** | **44.4** | **67.3** | **72.1** | **45.9** |
| N→K | SalsaNext (Range-view) | Base | 7.7 | 0.1 | 0.9 | 0.6 | 6.4 | 0.4 | 30.4 | 5.7 | 28.4 | 27.8 | 30.2 | 12.6 |
| | | CORAL [13] | 47.3 | 10.4 | 6.9 | 5.1 | 10.8 | 0.7 | 24.8 | 13.8 | 31.7 | 58.8 | 45.5 | 23.2 |
| | | MEnt [15] | 27.1 | 2.0 | 2.3 | 3.4 | 9.5 | 0.4 | 29.3 | 11.3 | 28.0 | 35.8 | 39.0 | 17.1 |
| | | AEnt [15] | 42.4 | 4.5 | 6.9 | 2.8 | 6.7 | 0.7 | 16.1 | 7.0 | 26.1 | 46.1 | 42.0 | 18.3 |
| | | (M+A)Ent [15] | 49.6 | 5.9 | 4.3 | **6.4** | 9.6 | 2.6 | 22.5 | 12.7 | 30.3 | 57.4 | 49.1 | 22.8 |
| | | SWD [7] | 34.2 | 2.7 | 1.5 | 2.0 | 5.3 | 0.9 | 28.8 | 20.5 | 28.3 | 38.2 | 36.7 | 18.1 |
| | | Rochan *et al.* [10] | 49.6 | 4.6 | 6.3 | 2.0 | 12.5 | 1.8 | 25.2 | 25.2 | 42.3 | 43.4 | 45.3 | 23.5 |
| | MinkNet42 (Voxel) | Base | 58.7 | 11.3 | 12.6 | 2.1 | **24.2** | 11.6 | 63.3 | 14.7 | 22.1 | 64.5 | 64.9 | 31.8 |
| | | Ours | **87.1** | **15.0** | **20.1** | 4.1 | 20.4 | **16.9** | **83.6** | **45.9** | **54.7** | **81.7** | **76.8** | **46.0** |

and subsequently evaluating on Waymo and SemanticKITTI, respectively. Fig. 4 and Fig. 5 present the qualitative results on nuScenes after training on Waymo and SemanticKITTI, respectively.

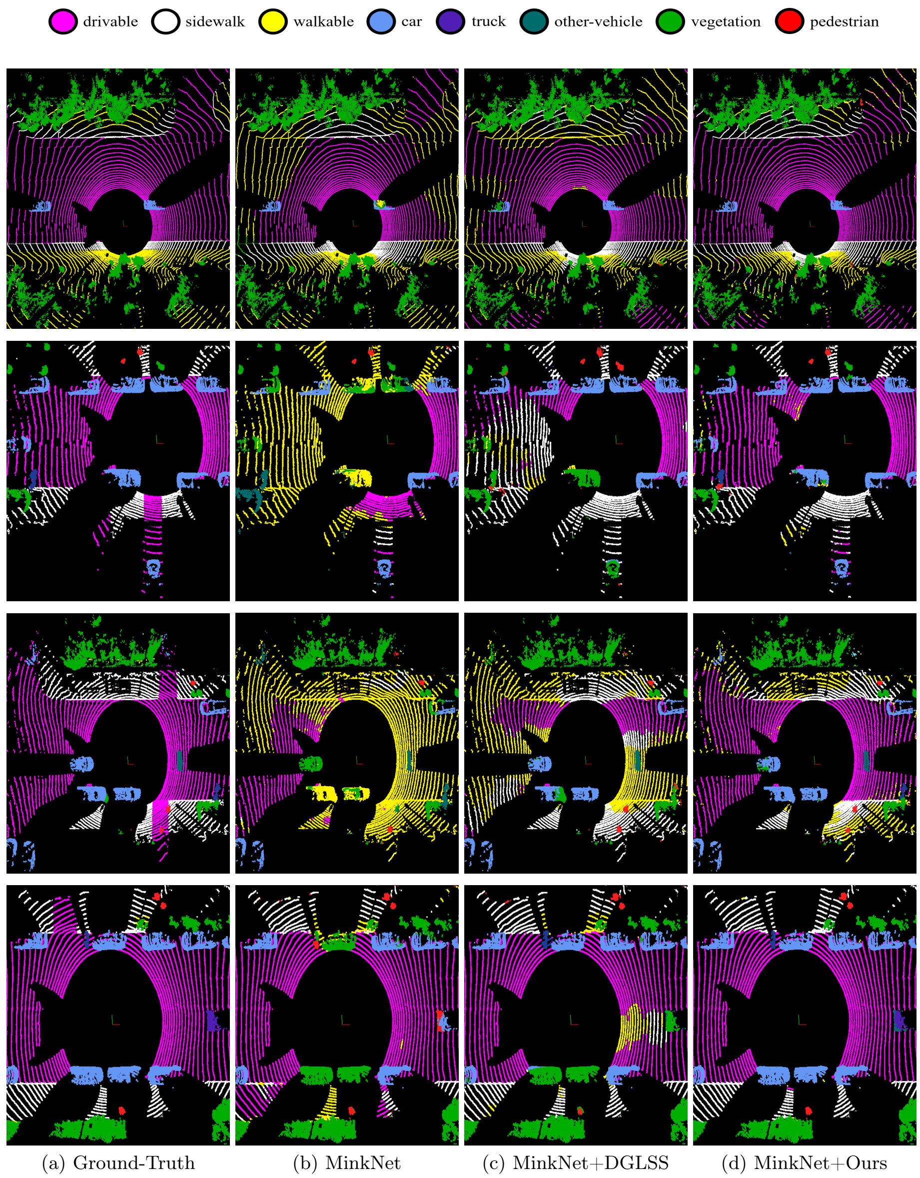



**Fig. 2:** Qualitative Results. Trained with nuScenes, tested on Waymo dataset.

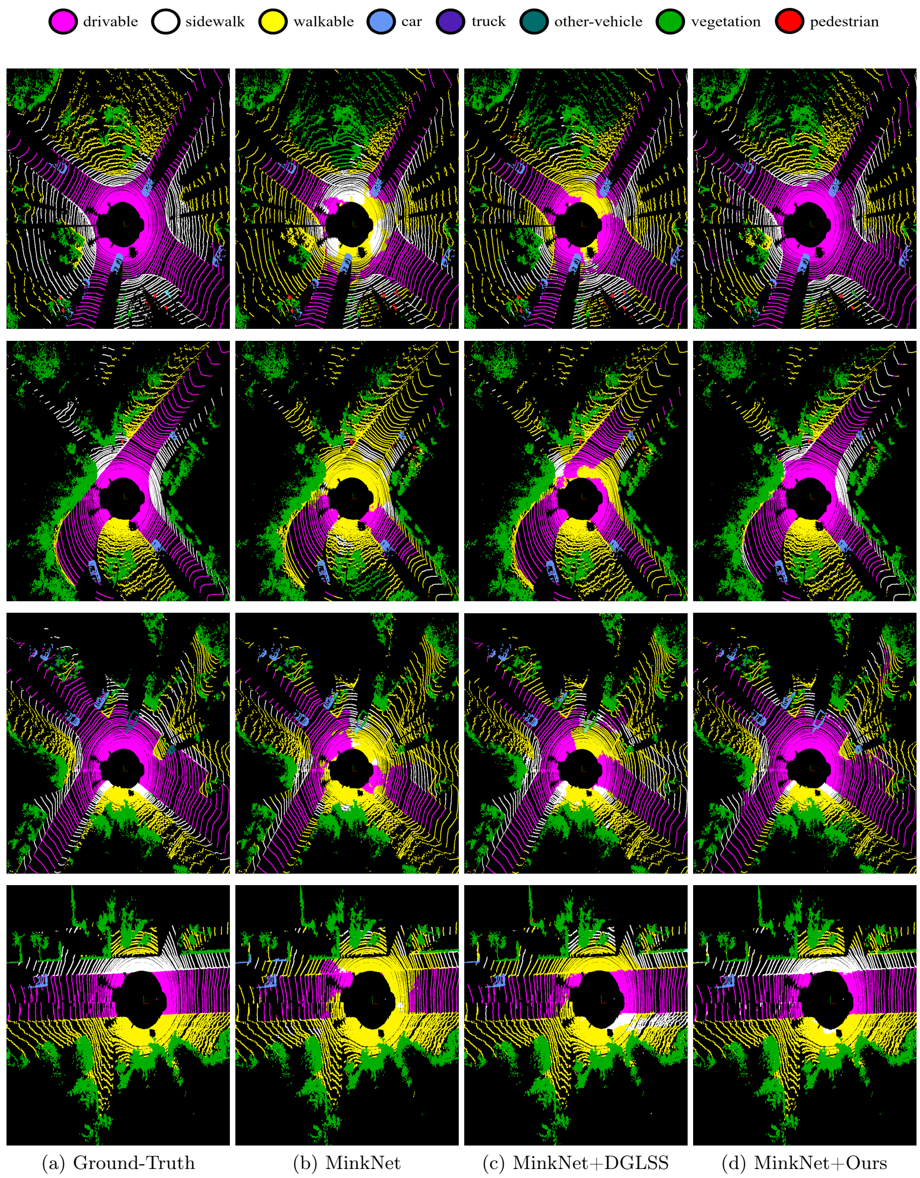



**Fig. 3:** Qualitative Results. Trained with nuScenes, tested on SemanticKITTI dataset.

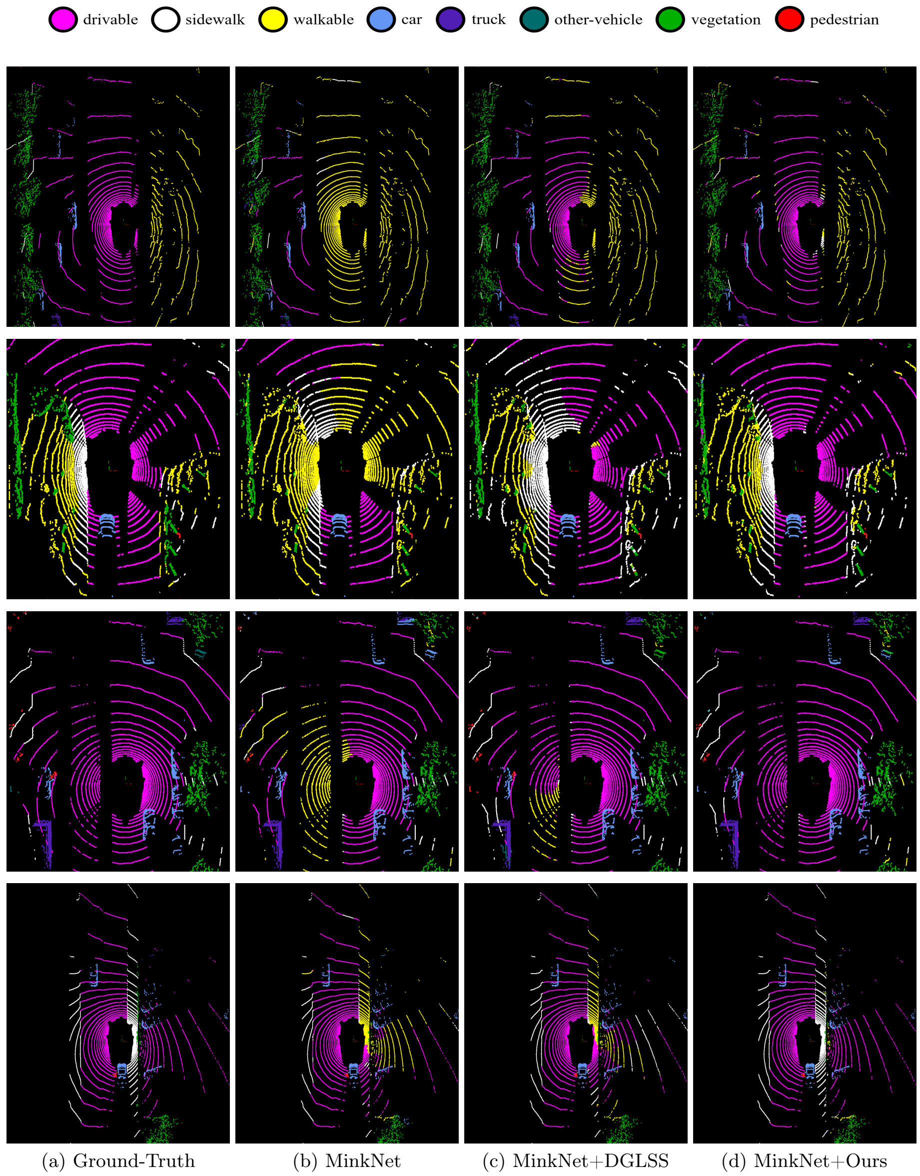



**Fig. 4:** Qualitative Results. Trained with Waymo, tested on nuScenes dataset.

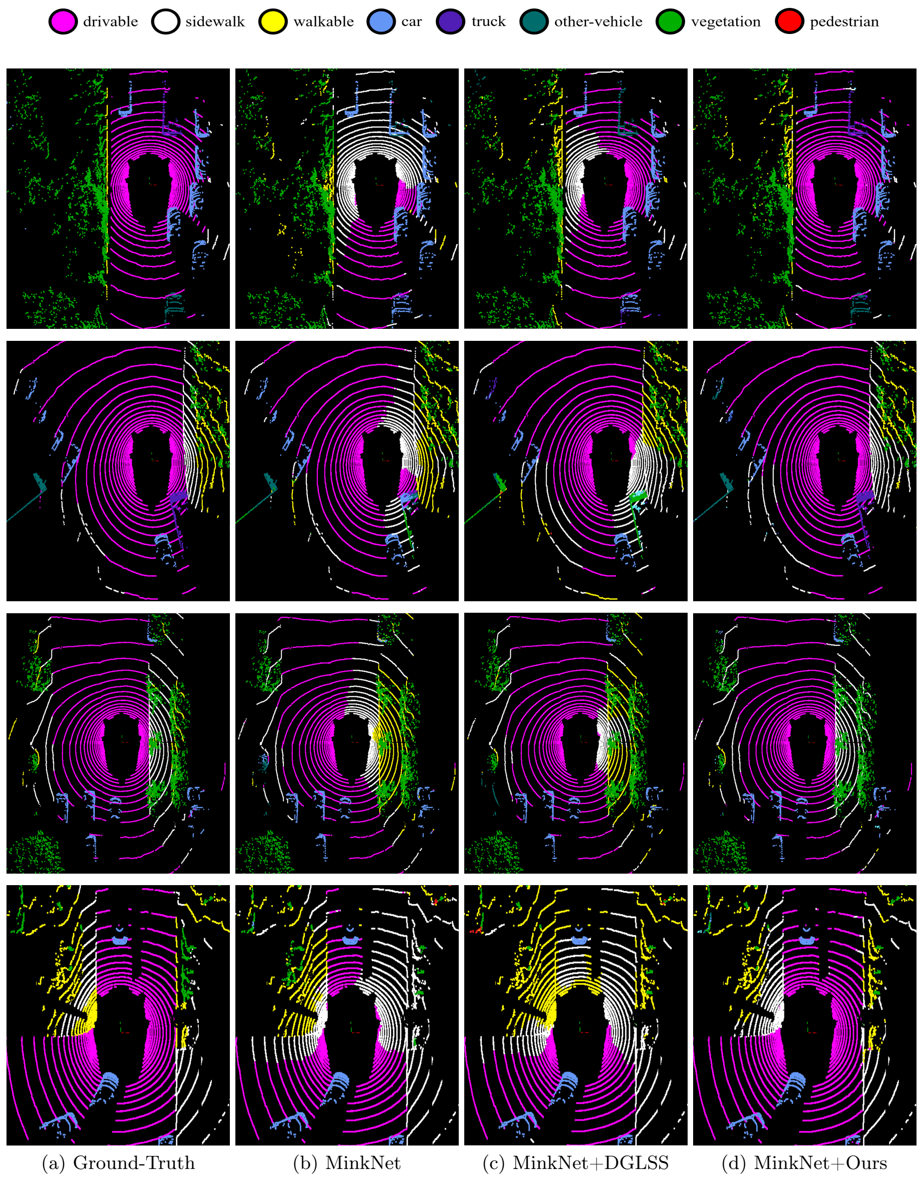



**Fig. 5:** Qualitative Results. Trained with SemanticKITTI, tested on nuScenes dataset.

## References


1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of IEEE International Conference on Computer Vision (ICCV). pp. 9297–9307 (2019) 3
2. Caesar, H., et al.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11621–11631 (2020) 3
3. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3075–3084 (2019) 3
4. Cortinhal, T., Tzelepis, G., Erdal Aksoy, E.: Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In: Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15. pp. 207–222. Springer (2020) 5
5. Ghiasi, G., Cui, Y., Srinivas, A.J., Qian, R., Lin, T.Y., Cubuk, E.D., Le, Q.V., Zoph, B.: Simple copy-paste is a strong data augmentation method for instance segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2918–2928 (2021) 4
6. Kim, H., Kang, Y., Oh, C., Yoon, K.J.: Single domain generalization for lidar semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17587–17598 (2023) 3, 5
7. Lee, C.Y., Batra, T., Baig, M.H., Ulbricht, D.: Sliced wasserstein discrepancy for unsupervised domain adaptation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10285–10295 (2019) 4, 5, 6
8. Luo, Y., Zheng, L., Guan, T., Yu, J., Yang, Y.: Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2507–2516 (2019) 4
9. Nekrasov, A., Schult, J., Litany, O., Leibe, B., Engelmann, F.: Mix3d: Out-of-context data augmentation for 3d scenes. In: Proceedings of International Conference on 3D Vision (3DV). pp. 116–125 (2021) 4
10. Rochan, M., Aich, S., Corral-Soto, E.R., Nabatchian, A., Liu, B.: Unsupervised domain adaptation in lidar semantic segmentation with self-supervision and gated adapters. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA). pp. 2649–2655 (2022) 5, 6
11. Ryu, K., Hwang, S., Park, J.: Instant domain augmentation for lidar semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9350–9360 (2023) 3, 4, 5
12. Shaban, A., Lee, J., Jung, S., Meng, X., Boots, B.: Lidar-uda: Self-ensembling through time for unsupervised lidar domain adaptation. In: Proceedings of IEEE International Conference on Computer Vision (ICCV). pp. 19784–19794 (2023) 3, 4, 5
13. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14. pp. 443–450. Springer (2016) 5, 6
14. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. **11**(1), 37–57 (mar 1985). https://doi.org/10.1145/3147.3165 2

15. Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2517–2526 (2019) 5, 6
16. Wang, H., Shen, T., Zhang, W., Duan, L.Y., Mei, T.: Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In: Proceedings of European Conference on Computer Vision (ECCV). pp. 642–659. Springer (2020) 4
17. Wei, Y., Wei, Z., Rao, Y., Li, J., Zhou, J., Lu, J.: Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In: Proceedings of European Conference on Computer Vision (ECCV). pp. 179–195. Springer (2022) 1
18. Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K.: Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA). pp. 4376–4382 (2019) 4, 5
19. Xiao, A., Huang, J., Guan, D., Cui, K., Lu, S., Shao, L.: Polarmix: A general data augmentation technique for lidar point clouds. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 11035–11048 (2022) 4
20. Yi, L., Gong, B., Funkhouser, T.: Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15363–15373 (2021) 3, 4, 5
21. Yu, F., Zhang, M., Dong, H., Hu, S., Dong, B., Zhang, L.: Dast: Unsupervised domain adaptation in semantic segmentation based on discriminator attention and self-training. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). vol. 35, pp. 10754–10762 (2021) 4
22. Yuan, Z., Wen, C., Cheng, M., Su, Y., Liu, W., Yu, S., Wang, C.: Category-level adversaries for outdoor lidar point clouds cross-domain semantic segmentation. IEEE Transactions on Intelligent Transportation Systems **24**(2), 1982–1993 (2023) 3, 4
23. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of IEEE International Conference on Computer Vision (ICCV). pp. 6023–6032 (2019) 4