Adversarially Masking Synthetic to Mimic Real:
Adaptive Noise Injection for Point Cloud Segmentation Adaptation

Guangrui Li$^{1,2}$†, Guoliang Kang$^3$, Xiaohan Wang$^4$, Yunchao Wei$^5$, Yi Yang$^4$

$^1$ ReLER, AAIL, University of Technology Sydney  $^2$ Baidu Inc.  $^3$ Beihang University
$^4$ ReLER, CCAI, Zhejiang University  $^5$ Beijing Jiaotong University.

Abstract

This paper considers the synthetic-to-real adaptation of point cloud semantic segmentation, which aims to segment the real-world point clouds with only synthetic labels available. Contrary to synthetic data which is integral and clean, point clouds collected by real-world sensors typically contain unexpected and irregular noise because the sensors may be impacted by various environmental conditions. Consequently, the model trained on ideal synthetic data may fail to achieve satisfactory segmentation results on real data. Influenced by such noise, previous adversarial training methods, which are conventional for 2D adaptation tasks, become less effective. In this paper, we aim to mitigate the domain gap caused by target noise via learning to mask the source points during the adaptation procedure. To this end, we design a novel learnable masking module, which takes source features and 3D coordinates as inputs. We incorporate Gumbel-Softmax operation into the masking module so that it can generate binary masks and be trained end-to-end via gradient back-propagation. With the help of adversarial training, the masking module can learn to generate source masks to mimic the pattern of irregular target noise, thereby narrowing the domain gap. We name our method “Adversarial Masking” as adversarial training and learnable masking module depend on each other and cooperate with each other to mitigate the domain gap. Experiments on two synthetic-to-real adaptation benchmarks verify the effectiveness of the proposed method.

1. Introduction

Recently, point cloud semantic segmentation task attracts increasing attention because of its important role in various real-world applications, e.g., autonomous driving, augmented reality, etc. Despite remarkable progress [5, 19, 30, 33, 39, 40, 62, 63], most algorithms are designed for the fully-supervised setting, where massive annotated data is available. In the real world, it is costly and time-consuming to annotate large amounts of data, especially for labeling each point in the segmentation task. Synthetic data is easy to obtain and its label can be automatically generated, which largely reduces the human effort.

† Work done during an internship at Baidu.
of annotating data. However, it is usually infeasible to directly apply networks trained on synthetic data to real-world data due to the apparent domain gap between them. In this paper, we consider the synthetic-to-real domain adaptation [7, 10, 29, 38, 42, 45, 56, 68] for point cloud segmentation. Specifically, we aim to utilize the fully-annotated synthetic point clouds (source domain) and unlabeled point clouds collected from imperfect real-world sensors (target domain) to train a network to support the segmentation of real-world point clouds (target domain).

Domain adaptation solutions [8, 9, 20, 21] aim to discover and mitigate the domain shift from source to target domain. Through comparing the synthetic and real-world point clouds, we observe that the domain shift can be largely attributed to the unexpected and irregular noise existing in the target domain data. As with [53], we consider “noise” to be the missing points of certain instances/objects, where all pixel channels are zero. Such noise may be caused by various factors such as non-reflective surfaces (e.g., glass). As shown in Fig. 1, the synthetic point cloud is integral and clean, but the real one contains large amounts of noisy points. A model trained on clean source data may find it hard to understand the scene context under the distraction of noises and thus cannot achieve satisfactory segmentation results on target point clouds.

Previous domain adaptation methods [4, 13, 14, 18, 28, 31, 32, 38, 61] (e.g., adversarial training), which have been proven effective in the 2D visual tasks, can be applied to this 3D segmentation setting. For example, SqueezeSegV2 [54] employs geodesic correlation alignment [37] to align the point-wise feature distributions of two domains. However, without explicitly modeling and dealing with the noise, these methods bring quite weak benefits to the adaptation performance. Recently, several works attempt to deal with the target noise to mitigate the domain gap. Rochan et al. [43] randomly select target noise masks and apply the selected mask to source samples. Wu et al. [53] compute one dataset-level mask and apply it to all source samples. Zhao et al. [67] use CycleGAN [69] to perform noise inpainting which is then used to learn synthetic noise generation module. The issues of these previous works are two-fold: 1) they cannot adaptively determine the injected noises according to the context of source samples; 2) the generated mask cannot be guaranteed to reduce the domain shift. Thus, these methods may achieve sub-optimal results.

In this paper, we aim to mitigate the domain shift caused by the target noise by learning to adaptively mask the source points during the adaptation procedure. To reach this goal, we need to deal with two problems: 1) how to learn a spatial mask that can be adaptively determined according to the specific context of a source sample, and 2) how to guarantee the learned masks help narrow the domain gap. To solve the first problem, we design a learnable masking module named “Adaptive Spatial Masking (ASM)” module, which takes source Cartesian coordinates and features as input, to generate point-wise source masks. We incorporate Gumbel-Softmax operation into the masking module so that it can generate binary masks and be trained end-to-end via gradient back-propagation. To solve the second problem, we incorporate adversarial training into the masking module learning process. Specifically, during training, we add an additional domain discriminator on top of the feature extractor. By encouraging features from two domains (features of masked source samples and those of normal target samples) to be indistinguishable, the masking module is able to learn to generate masks mimicking the pattern of target noise and narrow the domain gap. Note that these two designs cooperate with each other to better align features across domains and improve the adaptation performance.

In a nutshell, our contributions can be summarized as:
- We notice that the pattern of target noise is unexpected and irregular. Thus, we propose to model the target noise in a learnable way. Previous works, which don’t explicitly model the target noise or ignore such characteristics, are less effective.
- We propose to adversarially mask source samples to mimic the target noise patterns. In detail, we design a novel learnable masking module and incorporate adversarial training. Both components cooperate with each other to promote the adaptation.
- Experiments on two synthetic-to-real adaptation benchmarks, i.e. SynLiDAR → SemKITTI and SynLiDAR → nuScenes, demonstrate that our method can effectively improve the adaptation performance.

2. Related Works

Point Cloud Semantic Segmentation aims to classify the point clouds into predefined semantic categories in a point-wise manner. Previous researches in this area could be categorized into three streams: 1) point-based methods propose to handle this task in a point-wise manner and aggregate the contextual information through MLP (Multiple Layer Perception) [11, 40, 41], GCN (Graph Convolutional Network) [52, 60], or newly designed convolutions [48, 55, 59, 66]. These methods typically require massive computation, making them hard to satisfy the latency constraint in real-world applications. 2) Voxel-based methods [26] try to convert point clouds into 3D voxels and employ 3D convolutions to learn the geometric distributions. Some researchers [65, 70] study the partition strategy in the 3D space, while some researchers [16, 47] propose new convolution architectures to handle the sparse 3D voxels. Also, the heavy computation cost of 3D CNN hampers their applicability to real-world applications. 3) projection-based solution is another routine focusing on transforming 3D point
In domain adaptive point cloud segmentation, we are provided with annotated source scans $\mathcal{S} = \{(P_i^s, M_i^s)\}_{i=1}^{N}$ and unlabeled target scans $\mathcal{T} = \{(P_i^t)\}_{i=1}^{N}$, where $P_i \in \mathbb{R}^{n_i \times 4}$ denotes the set of points with coordinates $(x, y, z)$ and intensity, $M_i \in \mathbb{R}^{n_i}$ denotes the ground-truth annotation for the point cloud, and $n_i$ is the number of points in the $i$-th scan. For more efficient processing, we employ spherical projection to transform each raw point cloud $P$ into 2D image $I \in \mathbb{R}^{H \times W \times 3}$, and the labels are transformed to $Y \in \mathbb{R}^{H \times W}$ accordingly. The details are presented below. We aim to train a segmentation model on $\mathcal{S}$ and $\mathcal{T}$ to make accurate predictions on target points.

**Spherical Projection.** For more efficient processing, we transform the sparse point clouds into 2D images with spherical projection like [53, 54]. Specifically, for a point with coordinate $(x, y, z)$, we project it into a 2D LiDAR image as:

$$I_{x,y,z} = \frac{z}{\sqrt{x^2 + y^2 + z^2}}$$

where $(x, y, z)$ is the normalized coordinate of the projection point on the 2D plane.
image with coordinates \((p, q)\):

\[
[p] = \begin{bmatrix}
\frac{1}{2}(1 - \arctan^2(y, x)/\pi) \cdot W \\
(1 - (\arcsin(z \cdot r^{-1}) + f_{up}) \cdot f^{-1}) \cdot H
\end{bmatrix},
\tag{1}
\]

where \(r = \sqrt{x^2 + y^2 + z^2}\) is the range of this point.  
\(f = f_{up} + f_{down}\) is the vertical field-of-view of the LiDAR sensor. For each projected point with coordinate \((p, q)\), we concatenate its Cartesian coordinates \((x, y, z)\), range \((r)\), and intensity, then obtain the projected LiDAR image \(I\) with the shape \(H \times W \times 5\). The intensity channel models the strength of LiDAR beams. As such, raw point clouds with sparse and unordered structures are transformed into 2D images, so that 2D convolutions can be applied directly.

**Domain Adversarial Training (DAT).** Domain adversarial training \([12, 17, 50]\) has been proven effective in aligning the feature distributions across domains. During training, an additional domain discriminator is introduced to classify the features into different domains. Through adversarial training, the network is encouraged to generate features that are indistinguishable across domains. Consequently, domain-invariant features can be learned, hence benefiting the adaptation performance.

Specifically, let \(D\) denotes the discriminator, the above min-max game can be formed as (the original GAN loss format \([15]\)):

\[
\min_G \max_D \mathcal{L}_{GAN}(G, D) = \mathbb{E}_{I' \sim \mathcal{S}}[\log(D(G(I')))] + \mathbb{E}_{I' \sim \mathcal{S}}[\log(1 - D(G(I')))],
\tag{2}
\]

where \(G\) denotes the feature extractor of the model.

### 3.2. Adversarial Masking

Our framework is illustrated in Fig. 3. Following the general practice in domain adaptation, the network is shared across source and target domain data. The network consists of a backbone \((G)\) to extract features and a task classifier \((F)\) to distinguish the samples into different categories. During training, we insert our designed masking module \((ASM)\) into the backbone to mask source points and attach an additional discriminator \((D)\) on top of the backbone to assign domain labels to features from both domains. On the one hand, the domain discriminator is trained to differentiate masked source samples and target samples. On the other hand, the ASM is encouraged to learn to mask source points to mimic the target noise patterns, and features are trained to be domain-invariant to confuse the domain discriminator. As a result, the adversarial training and the masking module work collaboratively to narrow the domain discrepancy.

**Target Noise Hinders Adversarial Training.** As adversarial training has been proven effective in 2D visual domain adaptation tasks, \(e.g.,\) image classification, semantic segmentation, it is natural to see if adversarial training can help learn domain-invariant features in this 3D synthetic-to-real adaptation scenario. As shown in Fig. 2, we observe an interesting phenomenon that the discriminator converges quickly and can easily differentiate most of target points from the source ones. In contrast, injecting noise (from a random target sample) to source samples helps alleviate such an issue, \(i.e.,\) the features of many source points can confuse the domain discriminator in terms of their domain labels. We assume that in plain adversarial training, target noise may serve as a shortcut for the discriminator to classify samples into different domains. Only with adversarial training, it is hard to discover the noise patterns of the target and meanwhile align feature distributions across domains. Thus, we propose to explicitly model the noise patterns of the target and adaptively inject noise into source samples in order to ease the conventional adversarial training.

**Adaptive Spatial Masking (ASM) Module.** As dis-
Figure 4. Illustration of Adaptive Spatial Masking (ASM). The proposed ASM takes source Cartesian coordinates and source features as input, and outputs two differentiable binary maps which divide points into two groups, *i.e.*, preserved and ignored. Then we impose the first mask on original source features.

Discussed above, the way of randomly injecting target noise to source samples can alleviate the issue of adversarial training to an extent. However, such a way may not be an optimal choice because it cannot adaptively determine the distribution of injected noise according to the specific context of each source sample. Moreover, the pattern of injected noise may be irregular, and the copy-and-paste way of injecting noise may not accurately capture those irregular patterns.

Thus, we design a learnable module to perform spatial masking on source samples. We name our module as Adaptive Spatial Masking (ASM) module. To be concrete, we present the diagram of ASM in Fig. 4. The module consists of two embedding branches and one fusion head that all employ 1×1 convolutions. First, the two branches take source Cartesian coordinates $O^s \in \mathbb{R}^{H' \times W' \times 3}$ (*i.e.*, the $x$, $y$, $z$ channels of projected LiDAR image but downsampled to $H' \times W'$ to match the size of feature map) and source features $E^s \in \mathbb{R}^{H' \times W' \times k}$ as inputs respectively. Then, the embedded features from two branches are fused via a fusion head to generate the desired source mask. Finally, we calculate the element-wise product between the learned mask and original source features. Then the masked feature is forwarded through the remaining layers for predictions.

Note that we attach a Gumbel-Softmax [22] layer to the end of the fusion head. The output of Gumbel-Softmax has two channels, each of which has spatial size $H' \times W'$. We use the first channel to indicate which points should be preserved and the second channel to indicate which points should be ignored. Then, the first channel is utilized to mask the source features with the element-wise product. Note that Gumbel-Softmax enables us to apply binary masks during the forward process, while supporting gradient backpropagation to update the parameters of our network. In contrast, the plain softmax layer cannot actually zero out source points, and thus leads to inferior results. We will show an empirical comparison of these designs in Sec. 4.3.

As shown in Fig. 3, we insert ASM (*i.e.*, denoted with color blue) after a specific shallow layer to inject noises to source samples. Note that we don’t directly insert ASM after the input of the projected LiDAR image. It is because that we empirically find the shallow features are also useful to learn a better source mask, while the input only contains the coordinate information. In our implementation, we place ASM after the first convolution block (*i.e.*, conv-bn-relu). Note that ASM is only applied to source samples during the training process. So we simply remove ASM module for the inference on target samples.

**Adversarial Masking.** With the proposed masking module, the model can inject noise to source samples by zeroing out shallow features of partial source points. However, we cannot guarantee that the generated mask can mimic the patterns of target noise and thus mitigating the domain gap. To solve this, we integrate the ASM-equipped model with an adversarial training paradigm. Specifically, with ASM, the corresponding discrimination and generation loss of adversarial training are

$$
\min_{\theta_G} \mathcal{L}_{dis} = \mathbb{E}_{I \sim T}[(1 - D(G(I')))^2] + \mathbb{E}_{I \sim S}[(D(\bar{G}(I')))^2],
$$

$$
\min_{\theta_G, \theta_{ASM}} \mathcal{L}_{gen} = \mathbb{E}_{I \sim S}[(1 - D(\bar{G}(I')))^2],
$$

where $\bar{G}$ is the backbone with ASM module inserted and $\theta_{ASM}$, $\theta_G$, $\theta_D$ denote the parameters of ASM module, backbone and discriminator respectively. Note that different from the loss format in Eq. 2, we choose LSGAN [35] in our implementation for its better stability.

### 3.3. Training Objective

Overall, the model is optimized with three objectives, *i.e.*, cross-entropy loss ($\mathcal{L}_{ce}$), Lovasz-Softmax loss [2] ($\mathcal{L}_{low}$), and the adversarial training loss ($\mathcal{L}_{gen}$, $\mathcal{L}_{dis}$):

$$
\min_{\theta_G, \theta_{ASM}, \theta_F} \mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{low} + \lambda \mathcal{L}_{gen}
$$

$$
\min_{\theta_D} \mathcal{L}_{dis}
$$

where $\theta_F$ denotes the parameters of the classifier and the cross-entropy loss is calculated as:

$$
\mathcal{L}_{ce} = - \sum_{h,w}^{H, W} \log[F(\bar{G}(I^s))(h, w, Y_{h, w})].
$$

The segmentation model and the domain discriminator are updated alternatively with objective Eq. 4 and Eq. 5, respectively. Note that $Y_{h, w}$ is the label for the pixel at position $(h, w)$ of a projected LiDAR image.

### 4. Experiments

#### 4.1. Setup

**Datasets.** In this paper, we perform experiments on two synthetic-to-real benchmarks, *i.e.*, SynLiDAR→SemKITTI and SynLiDAR→nuScenes.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>car</th>
<th>bicycle</th>
<th>motorcycle</th>
<th>truck</th>
<th>other-vehicle</th>
<th>person</th>
<th>bicyclist</th>
<th>road</th>
<th>parking</th>
<th>sidewalk</th>
<th>building</th>
<th>fence</th>
<th>vegetation</th>
<th>truck</th>
<th>terrain</th>
<th>pole</th>
<th>traffic-merge</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Only</td>
<td></td>
<td>4.2</td>
<td>12.8</td>
<td>6.7</td>
<td>0.9</td>
<td>3.7</td>
<td>9.6</td>
<td>14.3</td>
<td>1.4</td>
<td>0.3</td>
<td>23.2</td>
<td>47.6</td>
<td>5.0</td>
<td>54.3</td>
<td>21.4</td>
<td>27.0</td>
<td>23.2</td>
<td>3.1</td>
<td>15.2</td>
</tr>
<tr>
<td>AdaptSeg [49]</td>
<td>2D</td>
<td>0.6</td>
<td>5.9</td>
<td>3.4</td>
<td>1.9</td>
<td>5.4</td>
<td>8.2</td>
<td>13.6</td>
<td>60.0</td>
<td>1.7</td>
<td>37.8</td>
<td>36.4</td>
<td>3.7</td>
<td>31.8</td>
<td>17.3</td>
<td>40.1</td>
<td>20.9</td>
<td>4.4</td>
<td>17.2</td>
</tr>
<tr>
<td>ADVENT [51]</td>
<td>2D</td>
<td>16.6</td>
<td>9.3</td>
<td>2.1</td>
<td>2.8</td>
<td>13.6</td>
<td>8.4</td>
<td>16.5</td>
<td>30.0</td>
<td>3.1</td>
<td>34.7</td>
<td>34.7</td>
<td>6.8</td>
<td>44.5</td>
<td>20.4</td>
<td>28.9</td>
<td>20.0</td>
<td>4.0</td>
<td>17.9</td>
</tr>
<tr>
<td>CBST [71]</td>
<td>2D</td>
<td>7.6</td>
<td>9.0</td>
<td>4.8</td>
<td>1.2</td>
<td>11.1</td>
<td>10.5</td>
<td>11.1</td>
<td>8.8</td>
<td>1.9</td>
<td>20.0</td>
<td>48.5</td>
<td>8.9</td>
<td>35.8</td>
<td>20.3</td>
<td>25.2</td>
<td>28.6</td>
<td>4.9</td>
<td>14.6</td>
</tr>
<tr>
<td>CCM [27]</td>
<td>2D</td>
<td>10.2</td>
<td>8.1</td>
<td>5.1</td>
<td>1.9</td>
<td>3.3</td>
<td>9.7</td>
<td>14.4</td>
<td>52.1</td>
<td>2.7</td>
<td>30.6</td>
<td>39.3</td>
<td>3.9</td>
<td>38.8</td>
<td>18.8</td>
<td>26.9</td>
<td>24.2</td>
<td>4.2</td>
<td>17.3</td>
</tr>
<tr>
<td>PLCA [25]</td>
<td>2D</td>
<td>7.8</td>
<td>7.1</td>
<td>4.0</td>
<td>3.1</td>
<td>3.8</td>
<td>14.3</td>
<td>18.1</td>
<td>44.4</td>
<td>10.2</td>
<td>19.8</td>
<td>44.9</td>
<td>4.6</td>
<td>43.9</td>
<td>12.6</td>
<td>21.2</td>
<td>26.3</td>
<td>3.6</td>
<td>17.0</td>
</tr>
<tr>
<td>SqueezeSegV1 [53]</td>
<td>3D</td>
<td>43.7</td>
<td>7.4</td>
<td>5.1</td>
<td>4.6</td>
<td>5.8</td>
<td>6.5</td>
<td>13.9</td>
<td>36.7</td>
<td>3.1</td>
<td>31.0</td>
<td>37.4</td>
<td>7.4</td>
<td>28.9</td>
<td>26.2</td>
<td>29.9</td>
<td>27.1</td>
<td>5.0</td>
<td>17.4</td>
</tr>
<tr>
<td>SqueezeSegV2 [54]</td>
<td>3D</td>
<td>20.4</td>
<td>8.0</td>
<td>4.0</td>
<td>2.4</td>
<td>5.1</td>
<td>8.0</td>
<td>17.3</td>
<td>59.4</td>
<td>3.2</td>
<td>34.9</td>
<td>39.4</td>
<td>8.4</td>
<td>41.3</td>
<td>21.9</td>
<td>32.2</td>
<td>29.0</td>
<td>4.7</td>
<td>20.0</td>
</tr>
<tr>
<td>ePointDA [67]</td>
<td>3D</td>
<td>18.8</td>
<td>8.0</td>
<td>6.1</td>
<td>2.6</td>
<td>3.6</td>
<td>6.4</td>
<td>14.6</td>
<td>50.8</td>
<td>9.4</td>
<td>32.7</td>
<td>33.3</td>
<td>8.2</td>
<td>27.6</td>
<td>22.4</td>
<td>30.2</td>
<td>26.9</td>
<td>7.2</td>
<td>18.2</td>
</tr>
<tr>
<td>LiDAR-Net [24]</td>
<td>3D</td>
<td>22.3</td>
<td>7.2</td>
<td>10.3</td>
<td>2.0</td>
<td>2.3</td>
<td>8.1</td>
<td>18.8</td>
<td>43.2</td>
<td>5.6</td>
<td>33.5</td>
<td>32.6</td>
<td>4.9</td>
<td>29.8</td>
<td>26.5</td>
<td>22.5</td>
<td>21.4</td>
<td>5.4</td>
<td>17.4</td>
</tr>
<tr>
<td>CoSMix [44]</td>
<td>3D</td>
<td>15.1</td>
<td>4.3</td>
<td>3.2</td>
<td>1.0</td>
<td>1.4</td>
<td>5.4</td>
<td>5.5</td>
<td>47.6</td>
<td>2.9</td>
<td>32.9</td>
<td>54.1</td>
<td>7.8</td>
<td>58.1</td>
<td>24.8</td>
<td>41.0</td>
<td>32.1</td>
<td>3.0</td>
<td>20.0</td>
</tr>
<tr>
<td>RandMask+ADV</td>
<td>3D</td>
<td>42.0</td>
<td>10.0</td>
<td>13.0</td>
<td>2.4</td>
<td>4.7</td>
<td>8.2</td>
<td>20.7</td>
<td>30.5</td>
<td>3.7</td>
<td>29.2</td>
<td>37.7</td>
<td>5.0</td>
<td>30.7</td>
<td>22.9</td>
<td>26.7</td>
<td>24.7</td>
<td>4.1</td>
<td>18.6</td>
</tr>
<tr>
<td>FreqMask+ADV</td>
<td>3D</td>
<td>22.9</td>
<td>9.3</td>
<td>5.3</td>
<td>2.2</td>
<td>3.0</td>
<td>5.4</td>
<td>13.9</td>
<td>50.6</td>
<td>3.8</td>
<td>31.9</td>
<td>38.1</td>
<td>6.5</td>
<td>32.0</td>
<td>25.4</td>
<td>38.1</td>
<td>25.5</td>
<td>4.6</td>
<td>18.7</td>
</tr>
<tr>
<td>SpatialDropout+ADV</td>
<td>3D</td>
<td>30.1</td>
<td>7.3</td>
<td>3.2</td>
<td>3.1</td>
<td>5.1</td>
<td>10.3</td>
<td>11.6</td>
<td>44.3</td>
<td>3.2</td>
<td>30.4</td>
<td>40.1</td>
<td>7.7</td>
<td>30.8</td>
<td>22.3</td>
<td>27.9</td>
<td>23.9</td>
<td>3.9</td>
<td>18.0</td>
</tr>
<tr>
<td>Ours</td>
<td>3D</td>
<td>19.7</td>
<td>13.8</td>
<td>9.7</td>
<td>2.1</td>
<td>4.1</td>
<td>8.0</td>
<td>8.2</td>
<td>64.5</td>
<td>8.0</td>
<td>36.0</td>
<td>54.6</td>
<td>6.7</td>
<td>58.0</td>
<td>24.7</td>
<td>35.8</td>
<td>24.2</td>
<td>4.2</td>
<td>22.3</td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td>91.9</td>
<td>25.5</td>
<td>42.0</td>
<td>42.7</td>
<td>26.1</td>
<td>32.9</td>
<td>54.7</td>
<td>94.2</td>
<td>42.8</td>
<td>82.0</td>
<td>80.8</td>
<td>39.9</td>
<td>84.5</td>
<td>48.9</td>
<td>72.2</td>
<td>54.0</td>
<td>28.8</td>
<td>55.5</td>
</tr>
</tbody>
</table>

| Table 1. Experiments results of SynLiDAR [57] → SemKITTI [1] with SqueezeSegV3-21 [58] as the backbone. |

<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Only</td>
<td>2D</td>
<td>11.5</td>
</tr>
<tr>
<td>AdaptSeg [49]</td>
<td>2D</td>
<td>14.3</td>
</tr>
<tr>
<td>ADVENT [51]</td>
<td>2D</td>
<td>27.5</td>
</tr>
<tr>
<td>CBST [71]</td>
<td>2D</td>
<td>25.0</td>
</tr>
<tr>
<td>CCM [27]</td>
<td>2D</td>
<td>30.8</td>
</tr>
<tr>
<td>PLCA [25]</td>
<td>2D</td>
<td>24.6</td>
</tr>
<tr>
<td>SqueezeSegV1 [53]</td>
<td>2D</td>
<td>23.5</td>
</tr>
<tr>
<td>SqueezeSegV2 [54]</td>
<td>2D</td>
<td>23.5</td>
</tr>
<tr>
<td>ePointDA [67]</td>
<td>2D</td>
<td>28.7</td>
</tr>
<tr>
<td>LiDAR-Net [34]</td>
<td>2D</td>
<td>30.8</td>
</tr>
<tr>
<td>CoSMix [44]</td>
<td>2D</td>
<td>26.5</td>
</tr>
<tr>
<td>RandMask+ADV</td>
<td>2D</td>
<td>19.5</td>
</tr>
<tr>
<td>FreqMask+ADV</td>
<td>2D</td>
<td>14.8</td>
</tr>
<tr>
<td>SpatialDropout+ADV</td>
<td>2D</td>
<td>26.5</td>
</tr>
<tr>
<td>Ours</td>
<td>3D</td>
<td>17.7</td>
</tr>
<tr>
<td>Oracle</td>
<td>3D</td>
<td>17.5</td>
</tr>
</tbody>
</table>

Table 3. Ablation studies on Adaptive Spatial Masking. Experiments are conducted on SynLiDAR [57] → SemKITTI [1].

<table>
<thead>
<tr>
<th>Module</th>
<th>Modification</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Two Branches</td>
<td>Branch e (embedding only)</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>Branch o (coordinate only)</td>
<td>22.0</td>
</tr>
<tr>
<td></td>
<td>Both Branch</td>
<td>22.8</td>
</tr>
<tr>
<td>(b) Mask Type</td>
<td>Plain Softmax (soft)</td>
<td>19.6</td>
</tr>
<tr>
<td></td>
<td>Gumbel-Softmax (binary)</td>
<td>22.8</td>
</tr>
<tr>
<td>(c) Masking Layer</td>
<td>Input</td>
<td>22.0</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>22.8</td>
</tr>
<tr>
<td></td>
<td>Middle</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>End of the backbone</td>
<td>21.6</td>
</tr>
<tr>
<td>(d) Update Strategy</td>
<td>$\mathcal{L}<em>{gen}$ optimizes $\theta</em>{ASM}$ only</td>
<td>21.6</td>
</tr>
<tr>
<td></td>
<td>$\mathcal{L}<em>{gen}$ optimizes $\theta</em>{ASM}$ and $\theta_g$</td>
<td>22.8</td>
</tr>
</tbody>
</table>

4.3. Ablation Studies

Effect of Adaptive Spatial Masking (ASM). First, in Fig. 2, we show qualitatively that injecting noise can ease the adversarial training. And the quantitative comparison in Table 1 and 2 with other masking strategies (Spatial-Dropout, RandMask, FreqMask) also verifies that ASM derives better masks for easing the adaptation.

Effect of different branches of masking module. As discussed in Sec. 3.2, we use two embedding branches in the proposed ASM module. We evaluate the contribution of the two branches in Table 3 (a), where branch e receives source feature as input and branch o receives source Cartesian coordinates as input. It can be seen that removing either of them leads to an obvious drop in mIoU compared to the result using both branches. This verifies that both branches contribute to generating more effective masks.

Effect of Gumbel-Softmax. To contribute the distribution of Gumbel-Softmax, we compare the results with training using plain Softmax which generates soft masks (i.e., each mask value is within $[0, 1]$) for both forward and backward processes. As shown in Table 3 (b), using plain Softmax results in an obvious drop of mIoU, i.e., -3.2% mIoU. This is because plain Softmax cannot actually zero out source points, rendering it hard to mimic the target noise patterns to mitigate the domain shift.

Effect of different masking layers. In Table 3 (c), we compare the results of inserting ASM at different layers of the network, including input (i.e., masking the projected LiDAR image), ours (i.e., after the first conv. layer), middle (between the encoder and the decoder), and end of the backbone. From the table, we observe that inserting ASM at the shallower layer can achieve better results, which avoids features being affected by domain shift from the early stage. Compared with the result of inserting ASM directly after the input, ours achieves better results, verifying the important role of exploiting shallow feature information in learning better masks. Besides, inserting ASM at the end of the backbone is worse but not that far from “Ours”. This is be-

its superiority, e.g., our method outperforms AdaptSeg by 5.6% and 2.5% mIoU on SynLiDAR → SemKITTI and SyncLiDAR → nuScenes respectively. Especially, we notice that CBST shows inferior performance, which may be because of the low quality of pseudo labels resulting from the large gap between source and target point clouds. Third, compared with 3D solutions, our method also attains superior results, e.g. on SemKITTI, we achieve 2.8% and 4.6% absolute gain compared to SqueezeSegV2 and ePointDA, respectively. Moreover, even with adversarial training, various non-learnable masking strategies (RandMask, SpatialDropout, FreqMask) fail to achieve competitive results against ours. This is because these masking strategies cannot be adaptively adjusted according to the different contexts, and adversarial training is not able to impact the imposed source masks as they are not learnable.
Figure 5. **Visualization of Segmentation Results** (SynLiDAR → SemKITTI). We compare our method (d) with (a) ground truth, (b) source-only, and (c) AdaptSeg [49]. We present visualizations of both raw points (the first row) and projected point clouds (the second row). We show representative crops of projected 2D images due to the space limit.

Figure 6. **Statistics of ignored source points** (SynLiDAR → SemKITTI). Compared with performing masking randomly, our method exhibits a different preference toward different classes. For example, contrary to SpatialDropout, fewer points from class “Building” are ignored and more points from “Road” are dropped.

cause masking at the end of the backbone also introduces noises to the discriminator and the classifier.

**Optimization strategy with ASM**. We investigate the optimization with the ASM module and present it in Table 3 (d). Only adversarially updating ASM leads to inferior results than updating both (i.e., $\theta_G$ and $\theta_{ASM}$ in Eq.(4)). This shows that besides adversarially updating ASM, adversarially updating features also contributes to a better adaptation.

**Analysis of masked samples** To better understand the masking module, in Fig. 6, we present the class distribution of points that are ignored and compare with random dropout using a similar ignore ratio. Compared with random dropout, our result exhibits a different pattern/distribution, e.g., our method ignores more points of class “Road” but fewer points of class “Building” and “Vegetation”. However, our method outperforms it with a large margin, i.e., +4.8% on SemKITTI and +2.1% on nuScenes. This indicates that our method can derive more reasonable noise distributions for mitigating the domain gap.

**Sensitivity to hyper parameters.** In Table 4, we present the sensitivity of our method to $\lambda$ on both datasets. The performance of our method first increases and then decreases a little bit with the increase of $\lambda$ from $5 \times 10^{-4}$ to $5 \times 10^{-3}$. The bell shape of change verifies the regularization effect of adversarial training on the adaptation performance. Note that, within a wide range of choices of $\lambda$, our method consistently outperforms previous solutions by a large margin, which further verifies the effectiveness of our design.

**Visualization** In Fig. 5, we present the visualization of segmentation results on SynLiDAR→SemKITTI. From these figures, we observe that our method attains obvious improvement against source-only baseline and previous approach, which is in line with the superior results of our method shown in Table 1 and 2.

5. Conclusion

In this paper, we aim to mitigate the domain gap caused by target noises in synthetic-to-real point cloud segmentation adaptation. To this end, we propose Adversarial Masking, where a masking module is designed to derive learnable masks and the adversarial training paradigm encourages the masking module to mimic injecting target noises to source samples. The adversarial training and the masking module cooperate with each other to promote domain-invariant feature learning. Extensive experiments are conducted to prove the effectiveness of the proposed method.

**Broader Impact and Limitations.** Our method will not introduce bias but it may be impacted by the bias contained in the dataset. In terms of limitations, although our method outperforms the source-only baseline by a large margin, there is still a large gap to the oracle results. In future, we will explore more effective ways to narrow the domain gap.
References


[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Gi-an Carlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. CVPR, 2020. 6, 7


[19] Qingyong Hu, Bo Yang, Linhao Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. IEEE TPAMI, 2021. 1, 6


[22] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. ICLR, 2017. 5, 6


[32] Qing Lian, Fengmao Lv, Lixin Duan, and Boqing Gong. Constructing self-motivated pyramid curriculums for cross-


[67] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. In AAAI, 2021. 2, 3, 6, 7


