Self-Supervised Global-Local Structure Modeling for Point Cloud Domain Adaptation with Reliable Voted Pseudo Labels

Hehe Fan\textsuperscript{1} Xiaojun Chang\textsuperscript{2} Wanyue Zhang\textsuperscript{3,4*} Yi Cheng\textsuperscript{4} Ying Sun\textsuperscript{4} Mohan Kankanhalli\textsuperscript{1}

\textsuperscript{1}School of Computing, National University of Singapore \textsuperscript{2}ReLER, University of Technology Sydney \textsuperscript{3}Max Planck Institute for Informatics \textsuperscript{4}Institute for Infocomm Research, Agency for Science, Technology, and Research, Singapore

Abstract

In this paper, we propose an unsupervised domain adaptation method for deep point cloud representation learning. To model the internal structures in target point clouds, we first propose to learn the global representations of unlabeled data by scaling up or down point clouds and then predicting the scales. Second, to capture the local structure in a self-supervised manner, we propose to project a 3D local area onto a 2D plane and then learn to reconstruct the squeezed region. Moreover, to effectively transfer the knowledge from source domain, we propose to vote pseudo labels for target samples based on the labels of their nearest source neighbors in the shared feature space. To avoid the noise caused by incorrect pseudo labels, we only select reliable target samples, whose voting consistencies are high enough, for enhancing adaptation. The voting method is able to adaptively select more and more target samples during training, which in return facilitates adaptation because the amount of labeled target data increases. Experiments on PointDA (ModelNet-10, ShapeNet-10 and ScanNet-10) and Sim-to-Real (ModelNet-11, ScanObjectNN-11, ShapeNet-9 and ScanObjectNN-9) demonstrate the effectiveness of our method.

1. Introduction

Large-scale learning methods based on deep neural networks [7–10, 15, 25, 26, 35, 36, 39] constitute the recent advances in 3D vision, and play an important role for visual perception in intelligent platforms such as robots, drones and self-driving cars. These intelligent platforms often employ real-time depth sensors, such as LiDAR, to capture the accurate geometric information of scenes, which are represented by 3D point clouds. However, deep neural networks usually requires massive amounts of labeled point clouds for representation learning, which limits the scalability to the real world. To alleviate this problem, unsupervised point cloud domain adaptation is recently attracting increasing attention from the community [1, 27, 31, 43]. Domain adaptation aims to transfer the knowledge from a labeled source domain to a related but unlabeled target domain, in which the source and target domains share the same feature space. However, due to different point scales, object sizes, densities, styles, sensor perspectives, \textit{etc}., point cloud representations in the target domain inevitably deviate from the corresponding representations in the source domain, result-
Figure 2. Illustration of reliable voted pseudo label generation. First, target point clouds’ pseudo labels are voted by a few of nearest source neighbors in the feature space. Then, the target samples whose nearest source neighbor labels are consistent enough are selected as reliable training data.

To reduce domain shift, one solution is to directly learn from the target domain via self-supervised learning, i.e., exploiting the relations or correlations between different input signals. However, most of the existing methods focus on leveraging only one of the global or local structure of unlabeled data, such as predicting global vertical rotation [24], reconstructing point clouds from randomly rearranged object local parts [29], reconstructing a collapsed local region via Chamfer Distance loss [1] or localizing a curvature-changed local area [43]. Another domain adaptation solution is to transfer the knowledge from source to target domain via adversarial training [27] and self-training [43].

This paper is devoted to exploiting the self-supervised and transfer learning for point cloud domain adaptation. First, as shown in Fig. 1, we propose to learn point cloud representations by scaling up or down point coordinates of one dimension and then predicting the scale based on the other two unchanged dimensions. In this way, the network is able to capture the global structure in a self-supervised manner. To model the local structure, we propose to project a 3D local area onto a 2D plane by simply setting point coordinates of a randomly selected dimension to the same value and recover the squeezed area via mean squared error loss. Second, as shown in Fig. 2, to enhance the knowledge transfer from the source domain, we propose a voting method to assign reliable pseudo labels to target samples for self-training. Specifically, pseudo labels are voted based on a few nearest source neighbors in the shared feature space. Then, only the target point clouds whose nearest source neighbor labels are consistent enough are selected as reliable training data. With networks becoming stronger during training, our reliable voting method adaptively selects more target data, which in return facilitates learning because the amount of labeled target data increases.

To evaluate our method, we conduct experiments on the widely-used 3D domain adaptation benchmark PointDA [27], which consists of 10 shared classes from ModelNet40 [37], ShapeNet [2] and ScanNet [3]. Moreover, we also conduct experiments on a Sim-to-Real dataset [16], which consists of 11 shared classes from ModelNet40 and ScanObjectNN [34], and 9 shared classes from ShapeNet and ScanObjectNN, respectively. The contributions of this paper are threefold:

- To model the structure of unlabeled target point clouds, we propose the global scaling-up-down prediction and local 3D-2D-3D projection-reconstruction methods for point cloud domain adaptation.
- To transfer the knowledge from source domain, we propose a voting method to assign reliable pseudo labels to target samples. The method is able to iteratively select more and more target data during training, which in return facilitates learning.
- Extensive experiments on two datasets show that the proposed method effectively improves the accuracy of unsupervised domain adaptation on point clouds.

2. Related Work

Point Cloud Classification. Point cloud classification is one of the fundamental tasks for point cloud processing. Recently, a number of deep neural networks have been proposed to address this problem [25, 26, 32, 35, 36]. Most of these works aim to directly manipulate point clouds without converting irregular points into regular voxel grids, to avoid the quantization errors and high computational cost of voxelization. Since a point cloud is essentially a set of unordered points and invariant to permutations of its points, the key to point cloud processing is to design effective point-based spatial modeling operations that do not rely on point orderings. Our method is independent of these works and employ them to encode point clouds.

Self-supervised Learning. To learn from internal structures in images, self-supervised learning tries to find or exploit the relations or correlations between different input signals [5, 6, 13, 21–23, 38], e.g., modifying the input and (1) predict what changed or (2) ensure that the output representation does not change, such as learning spatial context of patches from an image [5], learning to count objects [22], predicting the missing pixels [23], recovering a plausible colored version of a grayscale image [38] and solving Jigsaw puzzles [21]. Self-supervised learning can also be applied to point clouds. Sauder and Sievers proposed to split an input point cloud into several parts and reconstruct the point cloud [29]. Poursaeed et al. proposed to rotate a
point cloud and then predict the rotation angle [24]. Shen et al. proposed to employ geometry-aware implicits in point clouds to reduce domain biases [31]. Achituve et al. and Zou et al. proposed to first mix up two point clouds and then predict the mixed labels [1] and their angles [43], respectively. Besides, they also proposed to reconstruct and localize deformed local areas, respectively. When modeling local structure, our method is inspired by the two deformation-based methods but achieves better accuracy.

Unsupervised Domain Adaptation. Unsupervised domain adaptation has been well developed on images [11, 19, 28, 30, 33, 40, 41]. These methods can be divided into three categories. 1) Adversarial training [19, 28, 33], which aims to directly learn unbiased representations via a discriminator to judge whether the learned features are from the target domain or the source domain, and a feature generator to confuse the discriminator. 2) Style transfer [40], which employs Generative Adversarial Networks [14] to transfer source images as the target style for training. 3) Self-paced learning [18], in which self-paced learning gets images. Therefore, target data’s pseudo labels can be used to confuse the discriminator. 2) Style transfer [40], which employs Generative Adversarial Networks [14] to transfer source images as the target style for training. 3) Self-paced learning [18], in which self-paced learning gets images. Therefore, target data’s pseudo labels can be used to confuse the discriminator. 2) Style transfer [40], which employs Generative Adversarial Networks [14] to transfer source images as the target style for training.

3. Proposed Method

In this section, we first introduce and formulate the point cloud domain adaption problem in Sec. 3.1. Second, we introduce the scaling-up-down method for self-supervised global structure modeling in Sec. 3.2. Third, the 3D-2D-3D project-reconstruction method for local structure learning is described in Sec. 3.3. Fourth, in Sec. 3.4, we briefly introduce the adversarial training method used in our paper. Fifth, the reliable voting method for pseudo label assignment is described in details in Sec. 3.5. Finally, the overall training procedure of our method is shown in Sec. 3.6.

3.1. Problem Formulation

The goal of unsupervised domain adaptation on point clouds is to transfer the knowledge for a labeled source domain \( S = \{(P_i^s, y_i^s)\}_{i=1}^{n_s} \) to an unlabeled target domain \( T = \{(P_i^t)\}_{i=1}^{n_t} \), where \( P \in \mathbb{R}^{m \times 3}, y_i^s \in \mathbb{Y} = \{1, \ldots, c\} \), \( m \) is the number of points and \( c \) is the number of shared classes. The \( n_s \) and \( n_t \) denote the number of source and target point clouds, respectively. The key to domain adaptation is to learn a mapping function or point cloud feature generator \( \Phi \) that projects point clouds from different domains into a shared feature space. The feature generator \( \Phi \) can be implemented by existing deep neural networks, e.g., PointNet [25] and DGCNN [35], which encode a point cloud to a vector, i.e., \( f = \Phi(P) \).

In this paper, we assume that point features, e.g., color, norm or other information, are not available. In this case, domain shifts can be caused by different point scales, point densities, object sizes, sensor perspectives, object styles, etc. Some of the shifts can be reduced by low-level data preprocessing and augmentation. For example, the point scale and object size problems can be addressed by normalizing object coordinates to a fixed range, e.g., \([-1, 1]\]. The density problem can be addressed by sampling, e.g., Farthest Point Sampling (FPS), point clouds to the same number of points. The perspective shift problem can be mitigated by rotation-based data augmentation. However, the other shifts, e.g., object styles, have to be reduced via high-level representations, which is the goal of unsupervised point cloud domain adaption methods.

3.2. Self-Supervised Global (G) Structure Modeling via Scaling-Up-Down Prediction

To enable \( \Phi \) to capture the global structure of target data without human-annotated class labels, we propose to scale up or down coordinates and then employ a regressor \( \Omega \) to predict the scale based on the point cloud feature \( f \). Specifically, suppose \( s_i = (s_i^x, s_i^y, s_i^z) \in \mathbb{R}^{+1 \times 3} \) is a random scale vector for the \( i \)-th target point cloud. Then, the coordinates \( P_i^t \) of the point cloud is scaled by \( s_i \odot P_i^t \), where \( \odot \) is element-wise multiplication and \( s_i \) is broadcasted for the multiplication. Finally, regressor \( \Omega \) is used to predict the scale \( s_i \),

\[
\min_{\Phi, \Omega} \mathcal{L}_g, \quad \mathcal{L}_g = \frac{1}{n_t} \sum_{i=1}^{n_t} ||\Omega(\Phi(s_i \odot P_i^t)) - s_i||_2^2.
\]

Note that, when predicting the scale, regression \( \Omega \) is actually based on the relative scales of the three dimensions, instead of the absolute change. For example, when we scale up two dimensions by 2 and leave the last dimension unchanged, regressor \( \Omega \) would misunderstand that the last dimension is scaled down by 0.5. To avoid this problem, we fix two of three dimensions and only scale
up or down one dimension. Therefore, \( s_i \) is limited to \( \{(s, 1), (1, s), (1, 1, s)\} \), where \( s \in \mathbb{R}^+ \). Moreover, because dramatically scaling up or down point clouds will change their structures, \( s \) is sampled from a small range, e.g., \( [0.5, 1.5] \) in this paper.

### 3.3. Self-Supervised Local (L) Structure Modeling via 3D-2D-3D Projection-Reconstruction

To enable \( \Phi \) to learn the local structure without point-level human annotations, we propose a 3D-2D-3D projection-reconstruction method. Specifically, similarly to [1], we first split the normalized 3D space into several regions. In this way, a point cloud is divided into multiple parts. Then, we randomly select a part that contains enough points for projection-reconstruction. Suppose \( v_i \in \{0, 1\}^{m \times 1} \) is a mask vector to indicate the selected points for the \( i \)-th target point cloud. Recall that \( m \) is the number of points in the cloud. When \( v_i[j] = 1 \), it indicates the \( j \)-point is selected. In this way, the selected local area can be denoted as \( P_i^t[v_i] \in \mathbb{R}^{||v_i|| \times 3} \). Third, we project the selected 3D area \( P_i^t[v_i] \) onto a 2D panel. To do so, we randomly select a dimension \( r \). Then, the \( r \)-dimensional coordinates of the selected points are squeezed to their mean. In this way, we obtain the locally projected point cloud \( P_i^t[r] \).

Finally, a reconstructor \( \Delta \) is employed to recover the projected area,

\[
\min_{\Phi, \Delta} \mathcal{L}_i, \quad \mathcal{L}_i = \frac{1}{n_i} \sum_{i=1}^{n_i} v_i \odot \|\Delta(\Phi(P_i^t[r])) - P_i^t[r]\|^2, \quad (2)
\]

Figure 4. Illustration of the 3D-2D-3D projection-reconstruction process for the \( i \)-th target point cloud \( P_i^t \). The points of a selected 3D local area is indicated by a mask vector \( v_i \). Then, the local area is projected into 2D by squeezing the \( r \)-dimensional coordinates to their mean, resulting in the locally projected point cloud \( P_i^t[r] \).

Finally, the feature generator \( \Phi \) and reconstructor \( \Delta \) is asked to reconstruct the squeezed area based on \( P_i^t[r] \).

where \( v_i \) is broadcasted for the element-wise multiplication. We illustrate the 3D-2D-3D projection-reconstruction process in Fig. 4.

### 3.4. Adversarial Training for Unbiased Representation Learning

Like most domain adaptation works, we also employ adversarial training [14] to reduce domain shifts and learn unbiased representations. In this paper, we employ Maximum
Classifier Discrepancy (MCD) [28] for adversarial training. MCD uses two classifiers \( \Psi_1 \) and \( \Psi_2 \), which map the feature vector \( f \) to two probability vectors of length \( c \), i.e., \( \Psi_1(f) \in \mathbb{R}^c \) and \( \Psi_2(f) \in \mathbb{R}^c \), respectively. Recall that \( c \) is the number of classes. For labeled data, MCD performs supervised-learning-based classification,

\[
\min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_s, \quad \mathcal{L}_s = -\frac{1}{n^s} \sum_{i=1}^{n^s} \sum_{j=1}^{c} I[j=y_i^t] \cdot \log (\Psi_1(\Phi(P_i^s))[j])
\]

\[-\frac{1}{n^s} \sum_{i=1}^{n^s} \sum_{j=1}^{c} I[j=y_i^t] \cdot \log (\Psi_2(\Phi(P_i^s))[j]).
\]

For unlabeled data, MCD first tries to maximize the prediction discrepancy of the two classifiers with fixed \( \Phi \),

\[
\min_{\Psi_1, \Psi_2} \mathcal{L}_s - \mathcal{L}_{adv}, \quad \mathcal{L}_{adv} = \frac{1}{n^t} \sum_{i=1}^{n^t} ||\Psi_1(\Phi(P_i^t)) - \Psi_2(\Phi(P_i^t))||_1.
\]

Then, the generator \( \Phi \) is trained to minimize the discrepancy with fixed classifiers,

\[
\min_{\Phi} \mathcal{L}_{adv}.
\]

In this way, \( \Phi \) is enforced to learn unbiased representations. Note that the adversarial training method is not our contribution.

3.5. Reliable Voted (RV) Target Pseudo Label Generation for Enhancing Domain Adaptation

Although adversarial training provides a way to reduce domain shifts, its effectiveness is usually limited. In this paper, we propose a self-training method to directly transfer the knowledge from source to target domain via target pseudo labels. Specifically, our method employs a voting strategy to assign pseudo labels to target samples. The pseudo labels of target point clouds are voted based on the labels of a few of their nearest source neighbors in the shared feature space. Suppose \( f_i^t \) and \( f_j^s \) are the features of the \( i \)-th target and the \( j \)-th source point clouds, respectively. Their similarity is calculated as

\[
e_{ij}^{ts} = \frac{f_i^t \cdot f_j^s}{||f_i^t||_2 \times ||f_j^s||_2}.
\]

Then, the \( k \)-nearest source neighbors are selected as follows,

\[
N(P_i^t, k) = \{ j \mid e_{ij}^{ts} \in \text{top}_k(\{e_{i1}^{ts}, \cdots, e_{in}^{ts}\}) \}.
\]

Third, the pseudo label of the \( i \)-th target point cloud is assigned with a voting mechanism,

\[
y_i^t = \text{vote}(\{y_j^s \mid j \in N(P_i^t, k)\}),
\]

where the vote function simply selects the majority as the output.

Although we employ a \( k \)-NN based voting method, the pseudo labels can be still unreliable, which may add noise into training data and lead to accuracy drop. To address this problem, we propose to only exploit reliable target point clouds, of which nearest source neighbor labels are consistent enough, to train the model with their voted pseudo labels,

\[
h_i = \begin{cases} 0, & \frac{\sum_{j \in N(P_i^t, k)} 1[y_i^t = y_j^s]}{k} < \lambda, \\ 1, & \frac{\sum_{j \in N(P_i^t, k)} 1[y_i^t = y_j^s]}{k} \geq \lambda, \end{cases}
\]

where \( \lambda \in (0, 1) \) is the consistency threshold and \( h_i \) indicates where the \( i \)-th target point cloud is selected as a reliable training sample. Finally, the selected reliable target data is used to train the feature generator in a supervised manner,

\[
\min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_t, \quad \mathcal{L}_t = -\frac{1}{||h||_1} \sum_{i=1}^{n^t} h_i \sum_{j=1}^{c} I[j=y_i^t] \cdot \log (\Psi_1(\Phi(P_i^t))[j])
\]

\[-\frac{1}{||h||_1} \sum_{i=1}^{n^t} h_i \sum_{j=1}^{c} I[j=y_i^t] \cdot \log (\Psi_2(\Phi(P_i^t))[j]),
\]

where \( h = (h_1, \cdots, h_{n^t}) \) is the selection indicator vector.

Note that, \( k \) and \( \lambda \) are the only two hyper-parameters of our reliable voting method. Moreover, as shown in the following experiments, we can further simplify this method by fixing \( \lambda \) to 1.0. In this case, \( k \) is the only hyper-parameter of this method and a bigger \( k \) indicates a higher reliability threshold.

Even though with fixed \( k \) and \( \lambda \), our reliable voting method has the ability to automatically and adaptively select an increasing amount of target data during training. At the early stage of training, because the feature generator \( \Phi \) is weak and the domain shift is large, only a few target point clouds, which are similar to source samples and easy to be recognized, reach the consistency threshold and are selected as reliable training data. When \( \Phi \) becomes stronger and domain shift reduces, target representations become more discriminative and the consistency of nearest source neighbors’ labels increases. Consequently, more target point clouds are added into the training set. In return, the increasing of labeled target data facilitates training. In this way, the feature generator \( \Phi \) is improved progressively.

3.6. Overall Training

In summary, our approach includes two self-supervised learning methods, i.e., scaling-up-down prediction and 3D-2D-3D reconstruction-reconstruction, and two transfer learning
methods, i.e., adversarial training and our reliable voted self-training method. The framework of our approach is illustrated in Fig. 3. The overall training process is shown in Alg. 1. The training contains multiple rounds. After each round, we perform reliable voted pseudo label generation. Each round contains several epochs. In each epoch, we perform supervised learning, scaling-up-down prediction, 3D-2D-3D projection-reconstruction and adversarial training.

4. Experiments

4.1. Datasets

**PointDA.** The PointDA [27] dataset is a widely-used benchmark for point cloud domain adaptation evaluation, which extracts the samples in 10 shared classes from ModelNet40 [37], ShapeNet [2] and ScanNet [3], respectively. In this way, PointDA consists of three subsets: ModelNet-10 (M10), ShapeNet-10 (S10) and ScanNet-10 (S*10). Given the three subsets, we can conduct six types of adaptation scenarios: M10 → S10, M10 → S*10, S10 → M10, S10 → S*10, S*10 → M10 and S*10 → S10.

**Sim-to-Real.** The Sim-to-Real [16] dataset is a fairly new benchmark, which consists of 11 shared classes from ModelNet40 and ScanObjectNN [34], and 9 shared classes from ShapeNet and ScanObjectNN, respectively. The dataset is built to evaluate meta-learning on point clouds. In this paper, we also employ the dataset for evaluating point cloud domain adaptation. The dataset consists of four subsets: ModelNet-11 (M11), ScanObjectNN-11 (S*O11), ShapeNet-9 (S9) and ScanObjectNN-9 (S*O9). Different from PointDA, Sim-to-Real asks models to transfer knowledge from simulated ModelNet or ShapeNet to real-world ScanObjectNN. Therefore, there are two types of adaptation scenarios Sim-to-Real: M11 → S*O11 and S9 → S*O9.

<table>
<thead>
<tr>
<th>Algorithm 1: GLRV Training Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> labeled source dataset ( S = { (P_i^s, y_i^s) }<em>{i=1}^{n_s} ), unlabeled target dataset ( T = { (P_i^t) }</em>{i=1}^{n_t} ), number of source neighbors for voting ( k ), reliability or consistency threshold ( \lambda ), feature generator ( \Phi ), classifiers ( \Psi_1 ) and ( \Psi_2 ), regression ( \Omega ), reconstructor ( \Delta ), number of training rounds ( R ), number of epochs ( E ) for each round.</td>
</tr>
<tr>
<td><strong>Output:</strong> ( \Phi, \Psi_1, \Psi_2, \Omega ) and ( \Delta ).</td>
</tr>
<tr>
<td><strong>Initialization:</strong> randomly initialize ( \Phi, \Psi_1, \Psi_2, \Omega ) and ( \Delta ); randomly initialize pseudo labels ( { \tilde{y}<em>i }</em>{i=1}^{n_t} ); zero-initialize selection indicator ( h = 0 ).</td>
</tr>
<tr>
<td>for 1 to ( R ) do</td>
</tr>
<tr>
<td>for 1 to ( E ) do</td>
</tr>
<tr>
<td>for ( (P_i^s, y_i^s), (P_i^t, \tilde{y}_i^t) ) in ( S, T ) do</td>
</tr>
<tr>
<td>( \min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_s \text{ with } (P_i^s, y_i^s); )</td>
</tr>
<tr>
<td>( \min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_t \text{ with } P_t^t; )</td>
</tr>
<tr>
<td>if ( h_i = 1 ) then</td>
</tr>
<tr>
<td>( \min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_t \text{ with } (P_t^t, \tilde{y}_i^t); )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>( \min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}<em>s - \mathcal{L}</em>{adv} \text{ with } (P_i^s, \tilde{y}_i^t) ) and ( P_t^t; )</td>
</tr>
<tr>
<td>( \min_{\Phi, \Psi_1, \Psi_2} \mathcal{L}_{adv} \text{ with } P_t^t; )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>update pseudo labels ( { \tilde{y}<em>i }</em>{i=1}^{n_t} ) and selection indicator ( h ) based on ( \Phi, k ) and ( \lambda );</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

**Table 1. Ablation study on each component of our method. Experiments are conducted on PointDA with the S*10 → S10 scenario.** When none of the components is employed, the model is directly transferred to the target domain without adaptation.

<table>
<thead>
<tr>
<th>Method</th>
<th>M→S</th>
<th>M→S*</th>
<th>S→M</th>
<th>S→S*</th>
<th>S*→M</th>
<th>S*→S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale (ours)</td>
<td>84.0</td>
<td>46.0</td>
<td>76.4</td>
<td>48.3</td>
<td>66.2</td>
<td>69.8</td>
</tr>
<tr>
<td>Rotation [24]</td>
<td>82.8</td>
<td>41.7</td>
<td>74.0</td>
<td>49.0</td>
<td>64.7</td>
<td>68.7</td>
</tr>
</tbody>
</table>

**Table 2. Comparison of vertical rotation and our scaling-up-down method for self-supervised global structure modeling on PointDA.**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M10 → S10</td>
<td>83.3</td>
<td>78.6</td>
<td>83.5</td>
</tr>
<tr>
<td>M10 → S*10</td>
<td>46.6</td>
<td>52.3</td>
<td>53.4</td>
</tr>
<tr>
<td>S10 → M10</td>
<td>79.8</td>
<td>75.0</td>
<td>75.7</td>
</tr>
<tr>
<td>S10 → S*10</td>
<td>49.9</td>
<td>51.4</td>
<td>48.7</td>
</tr>
<tr>
<td>S*10 → M10</td>
<td>70.7</td>
<td>69.3</td>
<td>68.2</td>
</tr>
<tr>
<td>S*10 → S10</td>
<td>64.4</td>
<td>63.6</td>
<td>67.9</td>
</tr>
</tbody>
</table>

**Table 3. Comparison of deformation-reconstruction (Def-Rec), deformation-localization (Def-Loc) and our 3D-2D-3D method for self-supervised local structure modeling on PointDA.**

S*10, S*10 → M10 and S*10 → S10.

**Self-Supervised** | **Transfer** | **Adversarial** | **Pseudo Label** | **Accuracy** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>69.8</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>71.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>66.7</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>67.4</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>73.8</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>74.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>76.2</td>
</tr>
</tbody>
</table>

The PointDA [27] dataset is a widely-used benchmark for point cloud domain adaptation evaluation, which extracts the samples in 10 shared classes from ModelNet40 and ScanObjectNN [34], and 9 shared classes from ShapeNet and ScanObjectNN, respectively. The dataset is built to evaluate meta-learning on point clouds. In this paper, we also employ the dataset for evaluating point cloud domain adaptation. The dataset consists of four subsets: ModelNet-11 (M11), ScanObjectNN-11 (S*O11), ShapeNet-9 (S9) and ScanObjectNN-9 (S*O9). Different from PointDA, Sim-to-Real asks models to transfer knowledge from simulated ModelNet or ShapeNet to real-world ScanObjectNN. Therefore, there are two types of adaptation scenarios Sim-to-Real: M11 → S*O11 and S9 → S*O9.
cloud domain adaptation method, i.e., PointDAN [27], and a meta-learning method, i.e., MetaSets [16]. We perform each adaptation scenarios three times and report the average and the standard deviation of the results in Table 5. Our method outperforms both the domain adaptation and meta-learning methods.

### 4.4. Ablation Study

#### A) Influence of scaling-up-down prediction, 3D-2D-3D projection-reconstruction, adversarial training and reliable voted pseudo label.

To investigate the influence of each component in our method, we conduct an ablation study on PointDA with the S*10 → S10 scenario. The results are show in Table 1. All the four components effectively improve domain adaption. Among them, the reliable voted pseudo label method (Vote + Reliable) is the most effective, which increases the baseline (64.2%) by 9.6%. Note that, without the consistency-based reliable target sample section, the improvement drops significantly. This is because the single voting method fails to obtain accurate pseudo labels and inevitably adds noise into training data.

#### B) Progressively selecting more and more target data and gradually improving accuracy.

To verify the ability of the reliable voting method to iteratively and adaptively select more and more target data during training, we show the number of selected reliable target samples, the accuracy of their pseudo labels and the accuracy on the test dataset in Fig. 5a ~ Fig. 5f. Under the premise of fixed or slightly varying accuracy of pseudo labels, more and more target training data is selected, which means that the number of correctly labeled target point clouds increases. With the correct labeled target data increasing, the feature generator is gradually improved on the target domain.

#### C) Impact of $k$ and $\lambda$ on voted pseudo label generation and adaptation performance.

Our reliable voting method contains two hyper-parameters, i.e., $k$ and $\lambda$. To investigate the impact of the two hyper-parameters, we conduct the S*10 → S10 adaptation with different $k$ and $\lambda$. The results are shown in Fig. 5j ~ Fig. 5l. To investigate $k$, we fix $\lambda$ to 1.0. When $k = 5$, a
Figure 5. Influence of reliable voted pseudo labels. Experiments are conducted on PointDA. (a)–(f): With training, the proposed method adaptively selects more and more target data and test accuracies increase gradually. (g)–(l): Impact of $k$ and $\lambda$ on voted pseudo label generation and adaptation performance ($S^{*}10 \rightarrow S10$). To show the real trend, test accuracies are evaluated without source validation.

large number of target samples with noisy pseudo labels are selected. In this case, the method plays a negative role to adapting. When $k$ increases, the reliability increases, leading to the accuracy improvement. When $k$ is too large, although the pseudo label accuracy increases, too few target data is selected. Consequently, the advantage of pseudo labels diminishes and the corresponding improvement weakens. To investigate $\lambda$, $k$ is fixed to 10. When $\lambda$ decreases from 1.0 to 0.6, the selected target data becomes less reliable. Because noise is added, the accuracy decreases.

5. Conclusion

In this paper, we propose two self-supervised learning methods, i.e., scaling-up-down prediction and 3D-2D-3D projection-reconstruction, and one reliable voted pseudo label method for point cloud domain adaptation. Experiments on two datasets demonstrate the effectiveness of our approach. However, when selecting target data, our reliable voting method does not take the class balance problem into consideration. A promising improvement is to integrate class diversity into selection, instead of only reliability.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (#A18A2b0046).
References


