

Point Cloud Classification by Domain Adaptation Using Recycling Max Pooling and Cutting Plane Identification

Hojin Yoo ¹  and Kyungkoo Jun ^{1,2,*}¹ Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, Republic of Korea² Energy Excellence and Smart City Laboratory, Incheon National University, Incheon 22012, Republic of Korea* Correspondence: kjun@inu.ac.kr

Abstract: Deep models have been studied in point cloud classification for the applications of autonomous driving and robotics. One challenging issue is that the point cloud of the same object could be discrepantly captured depending on sensors. Such a difference is the main cause of the domain gap. The deep models trained with one domain of point clouds may not work well with other domains because of such a domain gap. A technique to reduce domain inconsistency is domain adaptation. In this paper, we propose an unsupervised domain adaptation with two novel schemes. First, to improve unreliable pseudo-label assignment, we introduce a voting-based procedure based on the recycling max pooling module, which involves self-paced learning. It helps to increase the training stability of the models. Second, to learn the geometrical characteristics of point clouds in unfamiliar settings, we propose a training method of cutting plane identification, which works in an unsupervised way. Testing with the popular point cloud dataset of PointDA-10 and Sim-to-Real, experiments show that our method increase classification accuracy by 6.5%-points on average, ModelNet and ShapeNet as the source domain and ScanNet, and ScanObjectNN as the target domain. From an ablation study, it was observed that each method contributes to improving the robustness of domain adaptation.

Keywords: point cloud classification; unsupervised domain adaptation; self-paced learning; self-supervised learning



Citation: Yoo, H.; Jun, K. Point Cloud Classification by Domain Adaptation Using Recycling Max Pooling and Cutting Plane Identification. *Sensors* **2023**, *23*, 1177. <https://doi.org/10.3390/s23031177>

Academic Editor: Gemine Vivone

Received: 8 December 2022

Revised: 16 January 2023

Accepted: 17 January 2023

Published: 19 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The most straightforward way to depict a 3D object is as a point cloud. Deep learning techniques that use the point cloud to learn classification and segmentation [1–8], detection [9,10], or completion [11,12] are currently being researched. The ability of the deep model to learn these tasks is crucial in the fields of autonomous driving and robotics.

The point cloud can be constructed by gathering sensor data from real-world objects using a depth camera or Light Detection and Ranging (LiDAR). The point cloud can also be created by sampling a 3D shape. The points in the synthetic point cloud, which is produced in virtual space, are dispersed uniformly, and the data may be easily updated or modified. On the other hand, a real-world point cloud's distribution might not be regular due to equipment noise, and an object might not be entirely made up of obstacles. Furthermore, a single object's point cloud may be captured differently depending on the sensors used. From this perspective, it is considered that a domain shift takes place when attempting to develop the deep model using these point clouds.

The deep model must be adjusted to the environment in which it is intended to work in order to overcome domain shift. However, gathering data is a labor-intensive operation that takes a lot of time. Alternatively, synthetic point clouds could be produced with simplicity utilizing a Computer-Aided Design (CAD). A source domain is an environment rich in labeled data. A target domain is an environment where the deep model functions but where labeled data are scarce. Unsupervised domain adaptation (UDA) should be employed in training to enhance the deep model's performance in the target domain.

There are two basic ways to apply UDA to point clouds, according to recent studies. The first technology utilizes adversarial learning, which makes it harder to distinguish apart different domains [13]. Tasks can be completed regardless of the domain because this technique decreases the deep model's ability to discriminate input domains. The second method makes use of self-supervised learning, which teaches the geometrical properties of the domain on its own [14–18]. Self-supervised learning is an approach for establishing deep models about the properties of the domain so that tasks can be carried out in the domain. The strategy using self-supervised learning is currently the one with the best performance. Geometry-Aware Self-Training (GAST) [14], which encodes features by identifying angle and distorted parts of point clouds, was proposed.

However, the GAST's pseudo-label generation mechanism is unreliable because the pseudo-label is determined by a single prediction result, which has the highest confidence score. It ignores other prediction results even though their scores are high, resulting in wrong label assignments. Research on UDA of the point cloud is also lacking as compared to 2D image fields, where several self-supervised learning methods, such as predicting positional relationships of partial images and predicting the order of partial ones, are examined. Therefore, we suggest voting-based self-paced learning with a recycling max pooling module and unprecedented self-supervised learning method illustrated in Figure 1.

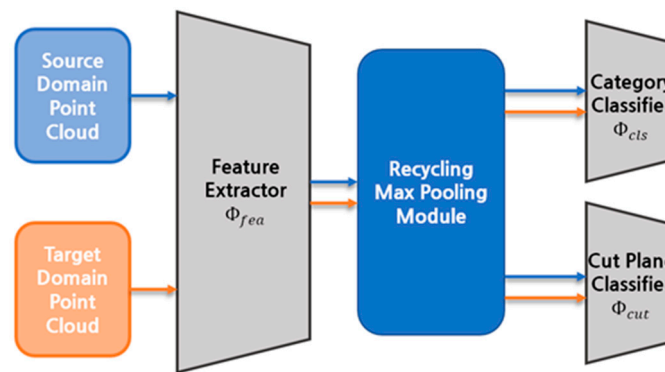


Figure 1. Concept of our domain adaptation method on point cloud classification.

Therefore, this paper introduces a novel UDA method that could be applied to point clouds. Our contributions are summarized as follows.

- We propose a novel voting-based procedure based on recycling the max pooling module and using improved cross-entropy loss function.
- We suggest a training method of cutting plane identification, which is a novel self-supervised learning method for domain adaptation. A corresponding cross-entropy loss function was also presented for learning.
- We showed that the proposed methods improve point cloud classification accuracy through ablation studies, comparison studies with existing methods, and additional studies.

2. Related Work

Point Cloud Classification—There are two sorts of recent point cloud classification networks. The first one is a method for understanding the point cloud's characteristic features using multi-layer perception (MLP), which can represent the association between all points [1,2,5,7]. This tactic includes a methodology for encoding all information between points into the MLP. The method has a weakness in that the deep model could not recognize local features due to a lack of regional information. An approach using local features with surrounding points [19] and was using a graph with those local features [3,4,6,8] were investigated to address the drawbacks. These networks tried to learn global or regional geometric features through point cloud classification. The domain adaptation on the point cloud has not been considered.

Unsupervised Domain Adaptation for Point Cloud—An UDA method used in a 2D image field could be classified into two types. In the first scheme, domain adaptation is attempted by reducing domain disparities. This approach is broken down into two parts: a way to utilize a statistical distribution and a technique to leverage domain distance. Minimizing the maximum mean discrepancy (MMD) between the source domain and the target domain is a representative way to increase classification accuracy [20,21]. The second way aligns features across domains is by employing adversarial learning. The method for aligning domains with a domain classifier and an inverse gradient layer has been proposed [22,23].

In a recent point cloud study, domain adaptation was attempted using both a method of minimizing domain difference and adversarial learning. Qin et al. [13] proposed a node module that could learn partial features of the point cloud and a way to minimize MMD between nodes. At the same time, adversarial learning was also applied to take in overall characteristics between domains. The amount of information the point cloud contains when learned partially is limited. Therefore an optimal local issue may arise.

Self-supervised Domain Adaptation for Point Cloud—Self-supervised learning is the process of exploiting input data that has been transformed into a format suitable for supervised learning. Through self-supervised learning, deep learning models could explore what the dataset represents. Recent 2D image research has suggested learning representativeness through tasks including predicting the positional relationship of two parts in an image [24], predicting the rotation angle of an image [25], and predicting order using a jigsaw puzzle [26]. These works have demonstrated that when supervised learning is implemented after comprehending the representation of images via self-supervised learning, the performance of deep models may be enhanced.

Studies have sought to increase classification accuracy in point clouds by adopting self-supervised learning. Sauder [15] used a method to predict position after dividing a point cloud into voxel units, and Poursaeed et al. [16] tried to improve classification accuracy by rotating the entire point cloud and then predicting the rotated angle. Recently, researchers have tried to acquire the geometrical properties of point clouds that use self-supervised learning in an effort to adapt the domain. In Achituve et al. [17], a point cloud is partially distorted and then reconstructed. Zou et al. [14] suggested a methodology for self-supervised learning that predicts the rotation angles of two-point clouds and distorted positions. In addition, in Shen et al. [18], the reconstruction of the point cloud is carried out with a feature map of the point cloud and sampled points.

These strive at domain adaption employing self-supervised learning on point clouds. There have been relatively few studies on self-supervised learning algorithms for point clouds since, unlike 2D images, they cannot be described simply using coordinates. For example, the self-supervised learning method using a jigsaw puzzle could not be easily implemented since the three-dimensional coordinate is more intricate than the two-dimensional coordinate. Therefore, research is required into how to effectively employ point clouds as self-supervised learning data.

3. Unsupervised Domain Adaptation for Point Cloud Classification

Source domain labeled point clouds $\mathcal{S} = \{\mathcal{P}_j^s, \mathbf{t}_j^s\}_{j=1}^{n_s}$ and target domain unlabeled point clouds $\mathcal{T} = \{\mathcal{P}_j^t\}_{j=1}^{n_t}$ are utilized in UDA on point cloud classification. The point cloud $\mathcal{P} \in \mathcal{X} \subset \mathbb{R}^{N \times 3}$ consists of N three-dimensional coordinated (x, y, z) . A classification result is obtained by Equation (1).

$$\Phi(\mathcal{P}) = \Phi_{cls}(\Phi_{fea}(\mathcal{P})) \quad (1)$$

where Φ_{fea} creates features of the input point cloud, and Φ_{cls} which is composed of fully connected (FC) layers and classifies a category of the point cloud. The equation of the

classification deep model Φ in GAST [14] has been replaced as follows in Equation (1) because. The reason is that, in our opinion, the output of Φ_{fea} is used as the input of Φ_{cls} .

Figure 2 depicts the suggested UDA procedure. It consists of three main components. The multiple one-dimensional feature vectors from the point cloud are produced by the feature extractor Φ_{fea} and recycling max pooling module. The category classifier Φ_{cls} identifies the class of the point cloud, and the cut plane classifier Φ_{cut} distinguishes the direction of cutting planes. We propose a novel UDA that makes use of self-paced learning and self-supervised learning. Before providing the proposed method, Section 3.1 explains the GAST learning category classification and domain adaptation method. Section 3.2 then introduces the proposed self-paced learning method. Finally, Section 3.3 illustrates the suggested self-supervised learning strategy.

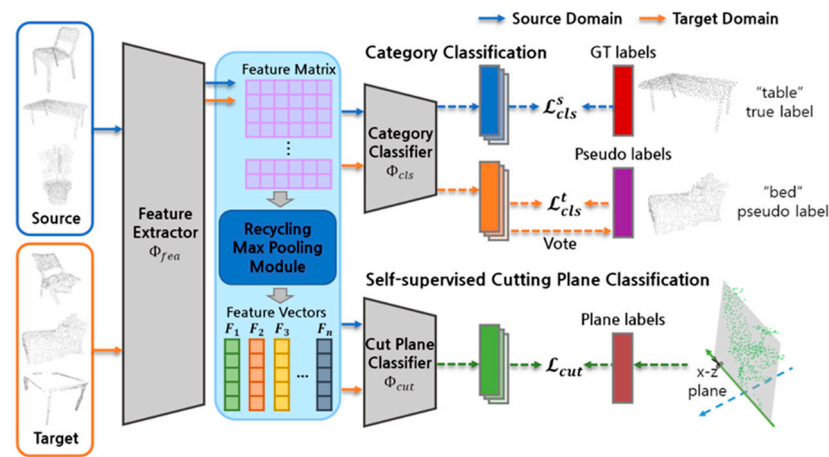


Figure 2. Overview of the proposed domain adaptive point cloud classification model.

3.1. Domain Adaptation on Geometry-Aware Self-Training

3.1.1. Learning Category Classification

For the point cloud $\{\mathcal{P}_i^s, \mathbf{t}_i^s\}_{i=1}^{n_s}$ in the source domain, the model Φ predicts a category with $\{p_i^s\}_{i=1}^{n_s}$, category classification learning in the source domain is conducted using cross-entropy loss, as shown in Equation (2).

$$\mathcal{L}_{cls} = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{c=1}^C I[c = \mathbf{t}_i^s] \log p_{i,c}^s \quad (2)$$

where $p_{i,c}^s$ means c -th predicted value of $\{p_i^s\}_{i=1}^{n_s}$, and $I[\cdot]$ means indicator function. $I[c = \mathbf{t}_i^s]$ means a value in that class. It has a value of 0 or 1, and one value of 1 per object n_s .

Since there are no labeled data, category classification learning in the target domain uses self-paced learning. Creating pseudo labels based on the model's computed prediction results and applying them for training is known as self-paced learning. For the point cloud $\{\mathcal{P}_i^t\}_{i=1}^{n_t}$ in the target domain, the model Φ predicts a category $\{p_i^t\}_{i=1}^{n_t}$. The category with the highest predicted value is taken into consideration as the label, and a pseudo-label is formed. When the label of the corresponding category is $\hat{\mathbf{t}}_{i,c}^t$, a pseudo label is brought about in Equation (3).

$$\hat{\mathbf{t}}_{i,c}^t = \begin{cases} 1, & \text{if } c = \operatorname{argmax}_{c'} p_{i,c'}^t, p_{i,c}^t > \exp(-\gamma) \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

where $p_{i,c}^t$ means c -th predicted value of $\{p_i^t\}_{i=1}^{n_t}$.

It is considered that producing pseudo labels with only the highest value is an unreliable method, so the methods that could generate reliable pseudo labels have been studied.

Fan et al. [27] proposed a voting-based pseudo-label generation method using the k-nearest neighbors (k-NN) algorithm. However, it has the shortcoming that pseudo labels can be determined only after all the feature vectors of input data are fed into the k-NN algorithm, unable to be used during the deep model training in real time. Our proposed method addresses this problem by having the pseudo-label determination process run during the model training at the same time.

3.1.2. Self-Supervised Learning on Geometry-Aware Self-Training

Through encoding geometrical features of point clouds to deep models using two approaches, domain adaptation by self-supervised learning was attempted in GAST. The first method is rotation angle identification (RotCls), and the other method is distortion area identification (LocCls). The identification results are trained using cross-entropy loss in both methodologies.

The angle is the rotated point cloud is anticipated throughout the RotCls phase. In the beginning, the point cloud $\mathcal{P}$ is divided into two-point clouds $\mathcal{P}_a$ and $\mathcal{P}_b$. Then, $\mathcal{P}_a$ is rotated along x -axis clockwise and $\mathcal{P}_b$ is rotated along y -axis in clockwise angle within $\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. A new point cloud $\mathcal{P}_{rot}$ which would be used as input for a deep model consisting of $\mathcal{P}_a$ and $\mathcal{P}_b$ rotated in x -axis and y -axis. Using $\mathcal{P}_{rot}$ as input, $\mathcal{P}_a$ and $\mathcal{P}_b$ are each identified in rotation angle.

LocCls refers to the process of distorting a portion of the point cloud and predicting a number of distorted regions. First, $\mathcal{P}$ is separated into k^3 regions in units of voxels, and then one section is selected for distortion. A new point cloud $\mathcal{P}_{loc}$ is created by applying distortion by sampling with an isotropic Gaussian distribution around the point is selected one point in the chosen area. Using $\mathcal{P}_{loc}$ as input, which of the k^3 region is distorted is classified.

3.2. Preliminary: Recycling Max Pooling

The backbone network of the point cloud deep model creates a feature matrix expressing correlation for each point or graph. By utilizing max pooling on the matrix, a feature vector is produced, after which classification or segmentation is carried out using the feature vector's information. In order to improve the performance of the point cloud deep model, a method of utilizing the feature matrix has been studied.

Chen et al. [28] argued that a lot of information is discarded from the feature matrix in the max pooling. Chen et al. suggested a recycling max pooling module that extracts the second and third max pooling vectors in addition to the first max pooling vector. The experiment was presented, showing similar performance even when the classification was performed using the second or third feature vector. This showed that not only the first max pooling vector but also the subsequent max pooling vectors contributed, have significant values. By using these sequential max pool processes, we can extract multiple label candidates from a single input, contributing to the improved label determination scheme. All feature vectors generated by the recycling max pooling module had their loss values calculated and were reflected in the deep model. Additionally, it reveals that the accuracy increased when the scheme was used with various deep-point cloud models.

3.3. Proposed Self-Paced Learning: Voting Pseudo Label with Recycling Max Pooling

The recycling max pooling is applied to the following procedure to produce multiple feature vectors. When point cloud $\mathcal{P}$ passes through the feature extractor Φ , the feature matrix $P_1^f \in \mathbb{R}^{N_1 \times M}$ can be obtained. N_1 denotes the number of points before the first max pooling, and M means the dimension of the feature matrix. After the first max pooling is performed, we will gain a feature vector $F_1 \in \mathbb{R}^M$ and a feature matrix $P_2^f \in \mathbb{R}^{N_2 \times M}$ which excludes the feature vector F_1 . N_2 implies the number of points after the first max pooling vector is excepted. The process is repeated r times until the r th feature vector $F_r \in \mathbb{R}^M$ is earned.

r feature vectors will be utilized for supervised learning of the source domain point cloud, self-paced learning of the target domain point cloud, and self-supervised learning. Section 3.3.1 introduces the procedure of learning category classification using the feature vectors of the source domain point cloud. The issue with the existing pseudo-generating method is brought up in Section 3.3.2, and a revised approach that makes up for the disadvantage while presenting the voting procedure is suggested in Section 3.3.3.

3.3.1. Supervised Learning in Source Domain with Recycling Max Pooling

When the point clouds in the source domain are given such as $\{\mathcal{P}_i^s, \mathbf{t}_i^s\}_{i=1}^{n_s}$, r feature vectors can be obtained for one point cloud. Each feature vector will pass through the category classifier Φ_{cls} to obtain a category prediction result p_i^s . The loss is calculated with a cross-entropy function using the prediction results as shown in Equation (4) by obtaining the average of the r feature vectors.

$$\mathcal{L}_{cls}^s = -\frac{1}{n_s} \sum_{i=1}^{n_s} \frac{1}{r} \sum_{j=1}^r \sum_{c=1}^C I[c = \mathbf{t}_i^s] \log p_{i,j,c}^s \quad (4)$$

$p_{i,j,c}^s$ means c -th predicted value. The deep model may learn to be more generalized than when employing a single feature vector because learning happens with the outcomes from several feature vectors.

3.3.2. Problem of Existing Self-Paced Learning

The pseudo-label generation method of GAST has a disadvantage that could happen while determining the learning direction. The reason is that the deep model recognizes the category with the highest score as the answer. If the deep model sees one object as another category at the beginning of learning and generates an incorrect answer, the issue that the deep model keeps learning with the incorrect label emerges. As a result, classification performance for the target domain point cloud suffers.

In order to reimburse the drawback, we propose the pseudo-label generation method using voting with the recycling max pooling module. In the previous study, the pseudo label was determined by examining just one classification result because the deep model of GAST makes one-dimensional output. Ours uses votes from a variety of results to choose the label. The dependability of the pseudo label is projected to be increased by this method, which is anticipated to be crucial in establishing the learning direction.

3.3.3. Pseudo Label Generation with Voting

Figure 3 illustrates the voting-based pseudo-label generation mechanism leveraging the recycling max pooling module. When the target domain point cloud $\mathcal{P}_i^t$ passes through the feature extractor Φ_{fea} , recycling max pooling module, and the category classifier Φ_{cls} , r category prediction results can be obtained. As shown in Equation (5), the label is determined through voting.

$$\tilde{\mathbf{t}}_i^t = \text{vote}\left(\left\{\mathbf{t}_j^t \mid j \in \{p_i^t\}_{i=1}^r\right\}\right) \quad (5)$$

$\tilde{\mathbf{t}}_i^t$ means a pseudo-label vector generated through voting and is in the form of a one-hot vector. The label is created only when the proportion of the results of a specific category is greater than a threshold $\lambda_{vote} \in [0, 1]$ and learning is performed as shown in Equation (6) through the cross-entropy loss. Self-paced learning will not be used for the remaining point clouds.

$$\mathcal{L}_{cls}^t = -\frac{1}{n_t} \sum_{i=1}^{n_t} \frac{1}{r} \sum_{j=1}^r \sum_{c=1}^C I[c = \tilde{\mathbf{t}}_i^t] \log p_{i,j,c}^t \quad (6)$$

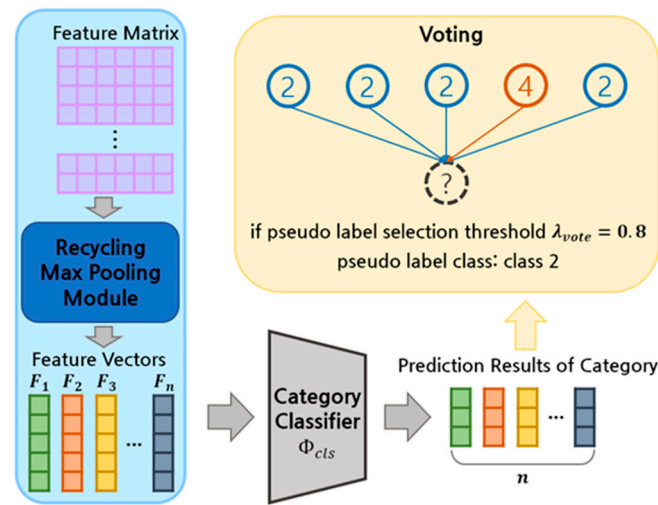


Figure 3. Pseudo-label voting process utilizing recycling max pooling module.

The category classification loss value of the source domain $\mathcal{L}_{cls}^s$ introduced in Section 3.3.1 is combined with $\mathcal{L}_{cls}^t$ as in Equation (7) to proceed with the point cloud category classification learning.

$$\mathcal{L}_{sem} = \mathcal{L}_{cls}^s + \lambda_t \mathcal{L}_{cls}^t \quad (7)$$

$\lambda_t \in [0, 1]$ is a penalty parameter for reducing noise at the beginning of the learning.

3.4. Proposed Self-Supervised Learning: Cutting Plane Identification

We present a novel method for the model to learn geometric information and cutting plane identification (CutCls). As shown in Figure 4, CutCls entails dividing the point cloud in half on a plane in either the x - z or y - z direction and then forecasting the cut direction between the two planes. This approach was motivated by the idea that geometric features could be learned by predicting the cut plane axis of the point cloud. We believe that this technique could extract and encode overall geometrical characteristics into feature vectors. The training process involving the classification of the cut direction x - z or y - z forces the model to learn geometric features from an unfamiliar environment. As a result, the backbone of the model can extract point cloud features even from samples of different domains.

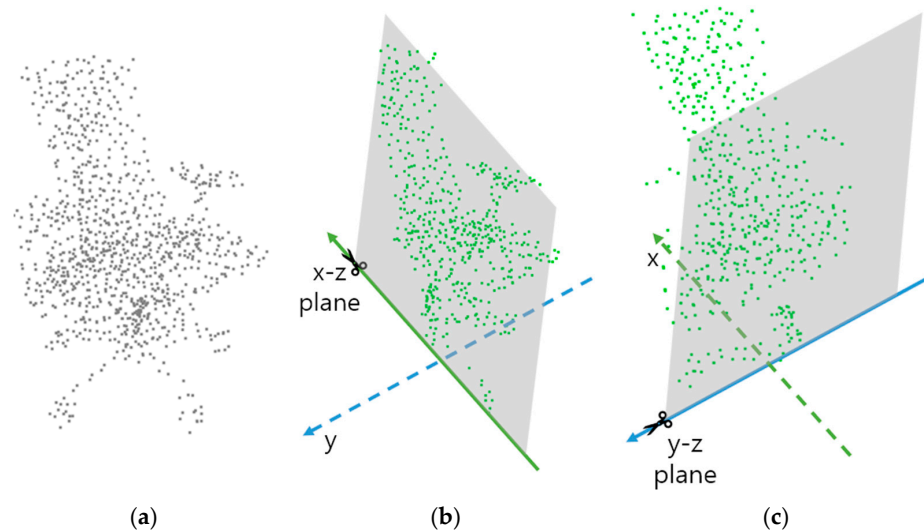


Figure 4. Illustrative examples of generating a cutting plane identification point cloud. (a) Original point cloud. (b) Point cloud cut on x - z plane. (c) Point cloud cut on y - z plane.

Moreover, the CutCls are expected to be more effective in using a cut point cloud, unlike traditional methods. For instance, it is challenging to find an optimal value in the case of the LocCls of GAST since it is a problem to predict the answer among k^3 classification. On the other hand, CutCls has the advantage of being able to use self-supervised learning that performs binary classification to quickly converge desired geometric feature learning. The binary classification means that the statement is determined between two values. It is possible to rapidly reach the optimal value because there is less information to learn.

When a set of x -axis or y -axis coordinate values of the plane to be cut in the point cloud, $\mathcal{P}$ is $\{o_i^m\}_{i=1}^m$, cutting coordinate is decided as in Equation (8).

$$CutCoord = \frac{\max_o \{o_i^m\}_{i=1}^m - \min_o \{o_i^m\}_{i=1}^m}{2} \quad (8)$$

If the point cloud $\mathcal{P}$ is cut along the selected plane from the obtained coordinates, two-point clouds $\mathcal{P}_l$ and $\mathcal{P}_r$ could be gained. Which of the point cloud should be employed is decided randomly because if only one of the two point clouds is used as input, self-supervised learning may not be successfully carried out. The deep model will have learning biases toward one class, and it will cause class imbalance.

The selected point cloud cut in half needs to be adjusted the number of points. A point cloud has a variable distribution of points for each point cloud, so cutting in the following method will produce a different number of points. On the other hand, if a 2D image were chopped in half, the quantity of information would be exactly halved. Therefore, the number of points should be sampled as $\mathcal{P}_{cut} \in \mathbb{R}^{\frac{m}{2} \times 3}$ through farthest distance sampling used in PointNet [1].

The dataset created for domain adaptation could be indicated as $\{\mathcal{P}_{cut,i}, h_i\}_{i=1}^{n_s+n_t}$. h_i means the direction of the cut plane and has the form of a one-hot vector. Using the point cloud $\mathcal{P}_{cut}$ as input, the direction of the cut plane is predicted among x - z plane or y - z plane through Φ_{cut} . When the set of predicted results is $\{p_{i,j,g}\}_{i=1}^{n_s+n_t}$, the model uses cross-entropy loss as in Equation (9) to encode geometric features.

$$\mathcal{L}_{cut} = -\frac{1}{n_s + n_t} \sum_{i=1}^{n_s+n_t} \frac{1}{r} \sum_{j=1}^r \sum_{g=1}^G I[g = h_i] \log p_{i,j,g} \quad (9)$$

Domain adaptation is conducted by summing $\mathcal{L}_{cut}$ and $\mathcal{L}_{sem}$ which was gained from Section 3.3.3 as in Equation (10).

$$\mathcal{L}_{total} = \mathcal{L}_{sem} + \lambda_{cut} \mathcal{L}_{cut} \quad (10)$$

where λ_{cut} is trade-off hyper-parameter.

4. Experiments

4.1. Datasets

PointDA—The PointDA [13] is used to check whether domain adaptation is performed. PointDA is a dataset that collects point cloud data corresponding to 10 categories common to ModelNet40 [29], ShapeNet [30], and ScanNet [31]. Only objects in the 10 categories are collected in ModelNet40 and are known as ModelNet-10 (M10). ShapeNet-10 (S10) is from ShapeNet, and ScanNet-10 (S*10) is from ScanNet. Table 1 shows the number of training and tests each dataset has and the property of datasets.

Table 1. Characteristics and compositions of the dataset.

Dataset Name	Characteristic	Num. Train	Num. Test
ModelNet-10	Synthetic	4183	856
ShapeNet-10	Synthetic	17,378	2492
ScanNet-10	Real-world	6110	2048

M10 and S10 are data obtained by uniformly sampling points from a 3D model artificially created through 3D CAD. However, S*10 has data acquired by separating corresponding to specific objects from the point cloud collected by a depth camera in real space. For this reason, ScanNet's point distribution is not uniform, in contrast to ModelNet40 and ShapeNet. Additionally, since depth information is not partially gathered, some point clouds may not accurately depict the object.

Sim-to-Real—The Sim-to-Real [32] is a new benchmark, which shares 11 categories with ModelNet40 and shares nine categories with ShapeNet from ScanObjectNN [33]. The dataset is gathered to estimate the effectiveness of meta-learning on point clouds. Our approach, which applies point cloud domain adaptation, is evaluated with the dataset. The dataset has four subsets: ModelNet-11 (M11), ScanObjectNN-11(S*O11), ShapeNet-9 (S9), ScanObjectNN-9 (S*O9).

4.2. Experiment Setup

Comparative Methods—The dataset was tested only when the target domain were real-world datasets, such as ScanNet-10, ScanObjectNN-11, and ScanObjectNN-9. The reason is that in order to check if this technology could be used in the real-world, data of the target domain should be from the real-world. We conduct experiments with four types of scenarios. PointDA: M10 $\rightarrow$ S*10, S10 $\rightarrow$ S*10, Sim-to-Real: M11 $\rightarrow$ S*O11, S9 $\rightarrow$ S*O9.

For evaluation, it was examined whether the proposed method improves the accuracy compared to the case without domain adaptation. In addition, it was also investigated how the accuracy was improved when combined with the two ways (LocCls and RotCls) proposed by GAST [14]. The comparison with the GAST method was inspected on our platform, and the accuracy of the GAST was referred to for supervised learning and remaining domain adaptation methods.

Implementation Details—Backbone network Φ_{fea} for extracting the point, cloud features are based on DGCNN [8] proposed in 2019. The DGCNN is an improved deep model for point cloud than PointNet [1] or PointNet++ [2], and its benchmark is higher. In addition, The DGCNN is most recently used for comparison in the field of point cloud domain adaptation research. The category classifier Φ_{cls} has three FC layers of MLP, each having 512, 256, and 10 nodes, respectively. The model predicts 10 categories of point clouds. A classifier for cutting plane prediction self-supervised learning uses one FC layer. The dimension of our point cloud dataset is all set to 1024.

As in GAST, z-axis direction random rotation data augmentation was applied, the Adam [34] optimizer was utilized, the initial learning rate was 0.001, the weight decay was 0.00005, and a cosine annealing learning rate scheduler was employed. All experiments were performed for 200 epochs with a batch size of 16. The hyperparameter of λ_{cut} is set to 1. The r is set to 5. The experiment was carried out after the most credible label had been produced with λ_{vote} being set to 1. The test was performed by measuring the classification accuracy without data augmentation on the test set of each data set. In total, we train all the methods with the same random seeds on an NVIDIA Geforce RTX 3070 GPU.

4.3. Results

The experiments were conducted by classifying them into three purposes. First, the ablation study signifies that each of the proposed methods is effective. Second, the comparison studies show how much performance improvement was achieved when compared to the domain adaptation methods that were previously applied. Finally, in additional studies, we analyze how changing different hyperparameters affects accuracy.

4.3.1. Ablation Studies

The ablation studies were performed to see if the proposed self-paced learning technique and self-supervised learning method are effective for domain adaptation. For the two scenarios M10 $\rightarrow$ S*10 and S10 $\rightarrow$ S*10, the accuracy without applying the domain

adaptation methods was calculated, and the effectiveness of each method was proved. The accuracies of the experiments are shown in Table 2.

Table 2. Accuracy (%) of ablation study on M10→S*10 and S10→S*10 scenario.

Adaptation Method		M10→S*10	S10→S*10
Voting SPL	CutCls		
		46.8	47.5
✓		53.9	55.1
	✓	49.6	46.4
✓	✓	56.3	56.4

In the M10→S*10 scenario, both self-paced learning and cutting plane identification show improved accuracy than when there is no domain adaptation. In addition, when learning by combining the two methods, better accuracy was recorded than when each was applied, with a 9.5%-point increase. When self-paced learning was used in the S10→S*10 scenario, the accuracy was improved. However, the accuracy decreased when the CutCls were used alone.

We had held the opinion that the ShapeNet-10's data imbalance was to blame for the result because it has a disproportionate amount of data in some categories. For example, chair and table point clouds accounted for 60.4% of the dataset. Therefore, we evenly adjusted the number of samples in each category and executed the experiment in same scenario. The results are different from our expectations; the accuracy does not improve at all. It even drops. It means the class imbalance was not the cause of the dropped accuracy.

To investigate the reason from another perspective other than the class imbalance, we analyzed the classification accuracy for each category as shown in Table 3. As a result, we find that 3 out of 10 categories are responsible for lowered accuracy (18% points drop in average); they are bathtub, sofa, and table, while the accuracy of other four categories improves, and the remaining categories do not change. The common characteristic of the three categories is that they are all relatively simple and big objects, while the improved four categories are bookshelf, lamp, monitor, and plant, which are small and have a complex shapes.

Table 3. Category-wise classification accuracy (%) on the S10 → S*10 scenario.

	Bathtub	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table
w/o Adapt	61.5	0	0.1	0.1	63.9	41.5	44.3	0	38.1	65.8
Only CutCls	42.3 (−19.2)	0 (0)	23.3 (+23.2)	0 (−0.1)	65.8 (+1.9)	58.5 (+17.0)	60.7 (+16.4)	64.0 (+64.0)	26.1 (−12.0)	45.5 (−20.3)

Furthermore, when the two methods were put on learning at the same time, the accuracy increased by 8.9%-points. This finding demonstrates that each suggested strategy for domain adaptation affects classification in the target domain.

4.3.2. Comparison Studies

The accuracy of the four scenarios introduced in Section 4.2 is assessed to see if the proposed self-paced learning and cutting-plane self-supervised learning approaches are successful. The accuracy results are compared with the accuracy of the existing method. Furthermore, we observed the outcomes of integrating our method with the existing self-supervised learning approach. LocCls and RotCls of GAST were merged with the CutCls, respectively, and an experiment combining all three was also conducted. Table 4 shows the results of the comparison studies.

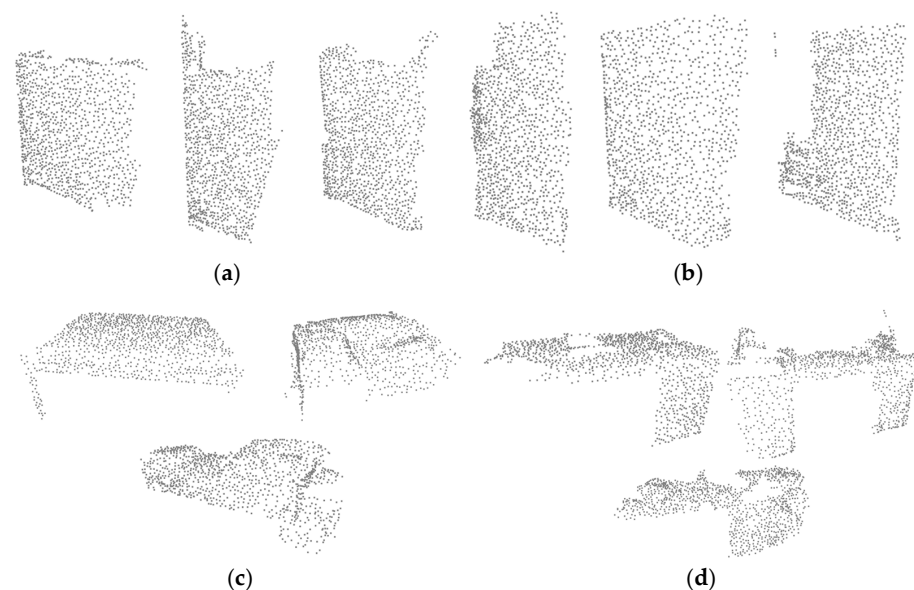
Table 4. Accuracy (%) on the PointDA and Sim-to-Real datasets. The accuracy for models marked with † is tested on our platform.

	M10→S*10	S10→S*10	M11→S*O11	S9→S*O9
Supervised	78.4	78.4	-	-
w/o Adaptation†	46.8	47.5	58.8	46.1
DANN [35]	42.1	50.9	-	-
PointDAN [13]	44.8	45.7	-	-
RS [36]	46.7	51.4	-	-
DefRec+PCM [17]	51.8	54.5	-	-
GAST† [14]	55.3	54.1	57.1	41.9
Ours†	56.3	56.4	63.8	48.5
Ours + LocCls†	55.2	54.7	61.2	48.1
Ours + RotCls†	57.3	55.8	61.3	47.9
Ours + LocCls + RotCls†	56.5	55.2	58.9	48.6

In the M10 → S*10 and S10 → S*10 scenarios, using the existing GAST method increased by 8.5%-points and 6.6%-points, respectively. When the proposed method was exploited, the accuracy separately rose by 9.5%-points and 8.9%-points. Even in the M11 → S*O11 and S9 → S*O9 scenarios, the accuracy rather decreased when learning with existing domain adaptation. Conversely, when we employed our strategy, the accuracy was enhanced by 5.0%-points and 2.4%-points, respectively.

It is presumed that the cause of the diminished accuracy with the existing method in the M11 → S*O11 and S9 → S*O9 scenarios is the problem with the self-paced learning method. When generating a pseudo label in the self-paced learning of the GAST, the category with the highest score in the predicted value is created as a label to learn. If an object's label is produced wrongly at the start of learning, there's a chance that the inaccurate label will continue to be learned later.

On the one hand, the characteristics of ScanNet-10 are markedly different for each category. On the other hand, in ScanObjectNN-11, there are categories with overlapping features. For instance, in bed and desk or cabinet and door, the shape of these objects looks similar when visualized in 3D. Studies would suffer in the long term if a model confused these objects to produce pseudo labels. An example of the objects, each having similarities, is shown in Figure 5.

**Figure 5.** Examples of object categories with similar characteristics in ScanObjectNN-11. (a) Cabinet. (b) Door. (c) Bed. (d) Desk.

The proposed voting-based pseudo-label generation method could reduce confusion among similar objects. This is because the pseudo label is not used for learning if the voting rate falls short of the threshold. Since learning proceeds only when the deep model is confident in the category, it can be expected that the accuracy is improved by generating steady pseudo labels. Table 5 shows that the accuracies of some categories increased when our proposed method was utilized. For the two categories of desk and bed, the accuracy improved by 20% points on average, proving that our voting scheme is effective, while the results of the door were similar. However, in the cabinet category, the accuracy dropped from 8.5% to 0%. We carefully investigated the cabinet samples from two datasets, ModelNet and ScanObjectNN. Surprisingly, we found that the cabinets are quite different in the two datasets, as shown in Figure 6. For example, the cabinets of ModelNet look similar to desks, while those of ScanObjectNN look similar to wardrobes. Such a difference is likely to be the main cause of the low accuracy.

Table 5. Part of category-wise classification accuracy (%) on the M11→S*O11 scenario.

	Desk	Bed	Door	Cabinet
GAST [14]	39.0	33.3	99.5	8.3
Ours	76.6	45.2	97.6	0

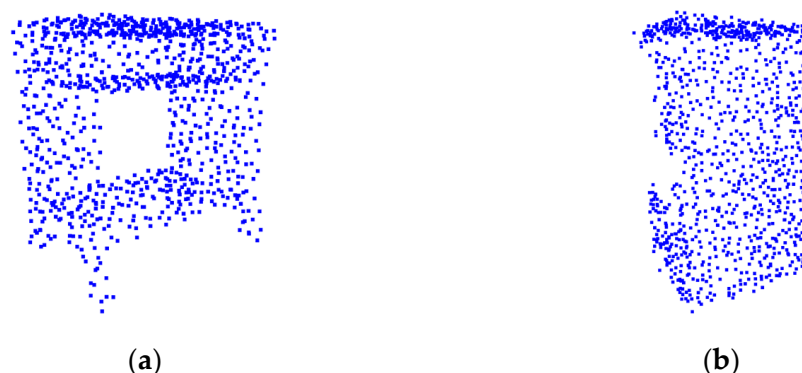


Figure 6. The example of a cabinet point cloud of ModelNet and ScanObjectNN. (a) ModelNet. (b) ScanObjectNN.

When learning in blend with GAST's self-supervised learning method, the highest accuracy was recorded if RotCls was merged in the M10 → S*10 scenario. In addition, in the S9 → S*O9 scenario, the highest accuracy is accounted for when all methods are combined. On the other hand, when LocCls was mingled, the accuracy was higher than the state without adaptation, but it did not record the highest accuracy. This shows that the performance improvement of the LocCls is insignificant compared to the other methods. The reason for this phenomenon is that the learning process of LocCls is more complicated than others.

4.3.3. Additional Studies

In order to examine the impact of the pseudo label generation threshold value on learning, an experiment was run while varying the value of λ_{vote} . In the M10 → S*10 scenario, when the λ_{vote} value was 1.0, 0.8, and 0.6, respectively. The proposed method was used, and the accuracies were compared. As shown in Table 6, the learning results show that the accuracy is highest when the λ_{vote} value is 1.0. In addition to that, the lower λ_{vote} value, the greater drop in accuracy. It is speculated that this is because the false label occurs more frequently as the threshold value decreases.

Table 6. Accuracy (%) of pseudo label generation threshold change experiment on M10 $\rightarrow$ S*10.

$\lambda_{vote}=1.0$	$\lambda_{vote}=0.8$	$\lambda_{vote}=0.6$
56.3	50.7	50.2

Moreover, an experiment was conducted to investigate how the accuracy change with the r value or recycling max pooling number. In the M10 $\rightarrow$ S*10 scenario, when the value was sequentially increased to 3, 5, 15, and 20, the change in accuracy was checked. The learning results are shown in Table 7. The findings indicate that when the r value rises, accuracy may also rise. Since there are fewer recycling max pooling and the voting dependability is worst, the accuracy is lowest when $r = 3$. The accuracy is maximum when the r value is 15, but it is lower than 15 when the value is 20. This suggests that overfitting occurred because there were too many recycled vectors employed for learning. Although the accuracy is the highest when the value is 15, the inference time of the deep model will increase because of the rise of r value. Therefore, it is important to choose a value that takes operational time into account.

Table 7. Accuracy (%) of the number of recycling max pooling change experiments on M10 $\rightarrow$ S*10.

$r=3$	$r=5$	$r=15$	$r=20$
49.8	56.3	56.4	53.4

For the purpose of researching the effect of the angle of the cutting plane, an experiment was handled by adjusting the cutting plane. The cutting plane was rotated by $\{10^\circ, 30^\circ, 45^\circ\}$ in y -axis direction, respectively, and then cut to generate $\mathcal{P}_{cut}$. The result of the experiment is shown in Table 8. It can be seen that the highest accuracy is achieved when the cutting plane was without rotation in the M10 $\rightarrow$ S*10 scenario. Since the input point's angle is not rotated in the y -axis when we test it, we anticipate that this is the reason. This implies that the optimal domain adaptation strategy is cutting the plane vertically.

Table 8. Accuracy (%) of cutting plane angle rotation experiment on M10 $\rightarrow$ S*10.

0°	10°	30°	45°
56.3	52.2	52.5	52.8

5. Conclusions

In this paper, we proposed a novel unsupervised domain adaptation method applicable to point cloud classification. The novel voting-based procedure based on the recycling max pooling module was presented. In addition, we introduced the self-supervised learning method and raised the possibility of expanding the self-supervised learning in point cloud classification.

According to the experiments, the proposed methods were helpful for the domain adaptation of point clouds. Each strategy improved accuracy, it was found through an ablation study. Furthermore, the most appropriate mechanism for domain adaptation was discovered by investigations combined with the existing approaches. Finally, the optimal conditions were known via additional studies adjusting the hyperparameters of the proposed method.

The suggested point cloud domain adaptation method does, however, have certain limitations. When generating pseudo labels with voting on multiple vectors, the problem will occur when the voting rate does not exceed the threshold. Our self-paced learning process was abandoned if the rate failed to surpass the threshold. In other words, the dataset's information was not being utilized to its full potential. If adversarial learning or other pseudo-label generation methods can be used for the data in this situation, higher accuracy can be expected. We will investigate this approach in future work.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education 2021R1F1A1055796.

Data Availability Statement: Not acceptable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
2. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proceedings of the Advances in Neural Information Processing System (30), Long Beach, CA, USA, 4–9 December 2017.
3. Chen, C.; Li, G.; Xu, R.; Chen, T.; Wang, M.; Lin, L. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4994–5002.
4. Zhao, H.; Jiang, L.; Fu, C.W.; Jia, J. Pointweb: Enhancing local neighborhood features for point cloud processing. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5565–5573.
5. Li, J.; Chen, B.M.; Lee, G.H. So-net: Self-organizing network for point cloud analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9397–9406.
6. Duan, Y.; Zheng, Y.; Lu, J.; Zhou, J.; Tian, Q. Structural relational reasoning of point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 949–958.
7. Yan, X.; Zheng, C.; Li, Z.; Wang, S.; Cui, S. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5589–5598.
8. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *Acm Trans. Graph. (tog)* **2019**, *38*, 1–12. [[CrossRef](#)]
9. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA, 18–23 June 2023; pp. 4490–4499.
10. Shi, S.; Wang, X.; Li, H. Pointcnn: 3d object proposal generation and detection from point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 770–779.
11. Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; Le, X. Pf-net: Point fractal network for 3d point cloud completion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 7662–7670.
12. Zhang, J.; Chen, X.; Cai, Z.; Pan, L.; Zhao, H.; Yi, S.; Yeo, C.K.; Dai, B.; Loy, C.C. Unsupervised 3d shape completion through gan inversion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 1768–1777.
13. Qin, C.; You, H.; Wang, L.; Kuo, C.C.J.; Fu, Y. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
14. Zou, L.; Tang, H.; Chen, K.; Jia, K. Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 6403–6412.
15. Jonathan Sauder, B.S. Self-Supervised Deep Learning on Point Clouds by Reconstructing Space. In Proceedings of the Advances in Neural Information Processing Systems 32. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
16. Poursaeed, O.; Jiang, T.; Qiao, H.; Xu, N.; Kim, V.G. Self-supervised learning of point clouds via orientation estimation. In Proceedings of the 2020 International Conference on 3D Vision (3DV), Fukuoka, Japan, 25–28 November 2020; pp. 1018–1028.
17. Achituve, I.; Maron, H.; Chechik, G. Self-supervised learning for domain adaptation on point clouds. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 123–133.
18. Shen, Y.; Yang, Y.; Yan, M.; Wang, H.; Zheng, Y.; Guibas, L.J. Domain Adaptation on Point Clouds via Geometry-Aware Implicits. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 7223–7232.
19. Cohen, T.S.; Geiger, M.; Köhler, J.; Welling, M. Spherical cnns. *arXiv* **2018**, arXiv:1801.10130.
20. Long, M.; Zhu, H.; Wang, J.; Jordan, M.I. Deep transfer learning with joint adaptation networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2208–2217.
21. Long, M.; Cao, Y.; Wang, J.; Jordan, M. Learning transferable features with deep adaptation networks. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 97–105.
22. Ganin, Y.; Lempitsky, V. Unsupervised domain adaptation by backpropagation. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1180–1189.
23. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7167–7176.

24. Doersch, C.; Gupta, A.; Efros, A.A. Unsupervised visual representation learning by context prediction. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1422–1430.
25. Komodakis, N.; Gidaris, S. Unsupervised representation learning by predicting image rotations. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
26. Kim, D.; Cho, D.; Yoo, D.; Kweon, I.S. Learning image representations by completing damaged jigsaw puzzles. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 793–802.
27. Fan, H.; Chang, X.; Zhang, W.; Cheng, Y.; Sun, Y.; Kankanhalli, M. Self-Supervised Global-Local Structure Modeling for Point Cloud Domain Adaptation with Reliable Voted Pseudo Labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 6377–6386.
28. Chen, J.; Kakillioglu, B.; Ren, H.; Velipasalar, S. Why Discard if You Can Recycle?: A Recycling Max Pooling Module for 3D Point Cloud Analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 559–567.
29. Vishwanath, K.V.; Gupta, D.; Vahdat, A.; Yocum, K. Modelnet: Towards a datacenter emulation environment. In Proceedings of the 2009 IEEE Ninth International Conference on Peer-to-Peer Computing, Seattle, WA, USA, 9–11 September 2009; pp. 81–82.
30. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. Shapenet: An information-rich 3d model repository. *arXiv* **2015**, arXiv:1512.03012.
31. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5828–5839.
32. Huang, C.; Cao, Z.; Wang, Y.; Wang, J.; Long, M. Metasets: Meta-learning on point sets for generalizable representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8863–8872.
33. Uy, M.A.; Pham, Q.H.; Hua, B.S.; Nguyen, T.; Yeung, S.K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1588–1597.
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
35. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 2030–2096.
36. Shu, R.; Bui, H.H.; Narui, H.; Ermon, S. A dirt-t approach to unsupervised domain adaptation. *arXiv* **2018**, arXiv:1802.08735.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.