ToThePoint: Efficient Contrastive Learning of 3D Point Clouds via Recycling

Xinglin Li[†§], Jiajing Chen[‡], Jinhui Ouyang[†§], Hanhui Deng[†§], Senem Velipasalar[‡], Di Wu[†§]
[†]Hunan University, China, [‡] Syracuse University, NY, USA, [§] ExponentiAI Innovation, China [*]

Abstract

Recent years have witnessed significant developments in point cloud processing, including classification and segmentation. However, supervised learning approaches need a lot of well-labeled data for training, and annotation is labor- and time-intensive. Self-supervised learning, on the other hand, uses unlabeled data, and pre-trains a backbone with a pretext task to extract latent representations to be used with the downstream tasks. Compared to 2D images, self-supervised learning of 3D point clouds is under-explored. Existing models, for self-supervised learning of 3D point clouds, rely on a large number of data samples, and require significant amount of computational resources and training time. To address this issue, we propose a novel contrastive learning approach, referred to as ToThePoint. Different from traditional contrastive learning methods, which maximize agreement between features obtained from a pair of point clouds formed only with different types of augmentation, ToThePoint also maximizes the agreement between the permutation invariant features and features discarded after max pooling. We first perform self-supervised learning on the ShapeNet dataset, and then evaluate the performance of the network on different downstream tasks. In the downstream task experiments, performed on the ModelNet40, ModelNet40C, ScanobjectNN and ShapeNet-Part datasets, our proposed ToThePoint achieves competitive, if not better results compared to the state-of-the-art baselines, and does so with significantly less training time (200 times faster than baselines).

1. Introduction

In recent years, self-supervised methods, which pretrain a backbone with pretext tasks to extract useful latent representations, have become increasingly effective [16]. For example, self-supervised tasks can be set to distinguish positive and negative samples or restore damaged images, and

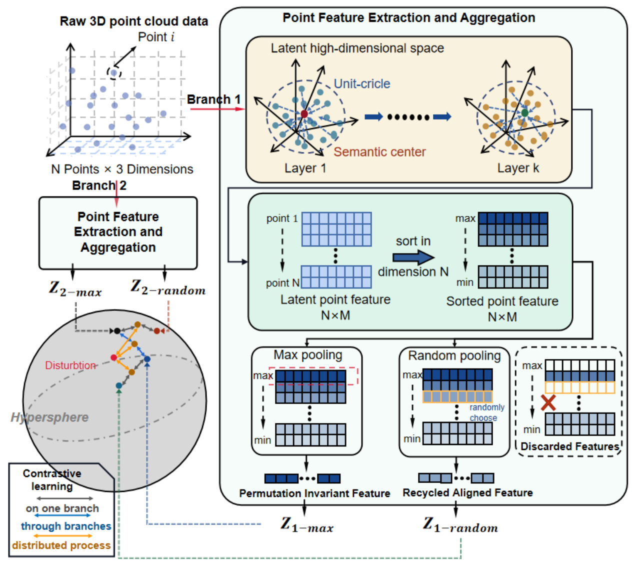


Figure 1. **A running example of ToThePoint.** Raw 3D point cloud data is streamed through two branches; in each branch, normalization and data augmentation are performed followed by traditional max-pooling operations. In our recycling mechanism, the N × M dimensional features are sorted and a row of features is randomly selected (from the remaining rows after max-pooling) as the recycled aligned features to assist the representation of permutation invariant features. The four features extracted from the two branches are next subjected to two stages of contrastive learning. Then the learning result would be mapped on the hypersphere.

these self-supervised pre-training tasks have been proven to provide rich latent feature representations for downstream tasks to improve their performance [5, 8, 10]. For the tasks, for which dataset labeling is difficult, such as detection [33], segmentation [12] or video tracking tasks [27], unsupervised pre-training can be especially helpful by alleviating the issue of insufficient labelled data. Moreover, it has been shown that self-supervised pre-training combined with supervised training provides better performance than traditional fully supervised learning by itself [11, 34, 36]. With the ever increasing availability of LiDAR sensors and stereo cameras, more and more point cloud data can be and have been captured. However, annotating this data is difficult, providing additional incentive for self-supervised algorithms developed for 3D point clouds.

*This work was supported in part by the National Natural Science Foundation of China under Grant No. 61972145 and 61932010, and the Key R&D Program of Hunan Province under Grant No. 2022GK2069. D. Wu and X. Li are the corresponding authors (e-mail: {dwu,lixinglin}@hnu.edu.cn).

There have been some works exploring self-supervised representation learning from point clouds, mainly based on generative models [29], reconstruction [20, 26] and other pretext tasks [34]. However, existing methods require large amounts, even millions of data samples, for self-supervised pre-training [9], making them computationally more expensive and time-consuming. Among traditional point cloud networks, PointNet [18] is a pioneering, end-to-end 3D point cloud analysis work. It obtains permutation-invariant features by adopting the max-pooling operation. There have been many subsequent works adopting this structure [14, 19]. Yet, the max-pooling operation discards a large number of points and their features. Chen et al. [4] have shown that these discarded features are still useful and, when recycled, can boost performance; and proposed recycling to improve the performance of fully-supervised 3D point cloud processing tasks, including classification and segmentation.

In this work, different from [4], we perform recycling differently, and also use the discarded point cloud features as a feature augmentation method for contrastive learning. This augmentation approach can allow having less training samples for self-supervised training, i.e. it can enable the self-supervised pre-training of a point cloud network without requiring large amounts of point cloud data. We achieve this by making good use of the point cloud features discarded by the max-pooling module of the point cloud network. Performing self-supervised learning with a small amount of point cloud data can also allow downstream tasks to get a competitive result.

We propose ToThePoint to accelerate self-supervised pretraining of 3D point cloud features, as shown by the example in Fig. 1. Compared to previous baselines, which require a large number of training samples and longer training time, our proposed work achieves its accuracy levels with only a fraction of samples during pre-training. The goal of our work is to introduce the distribution of the maximum aggregated features and the recycled point cloud features into the hypersphere space through a contrastive learning method. The maximum aggregated feature and the recycled point cloud feature from the same sample are regarded as a cluster. Contrastive learning is used to make the maximum aggregated feature become the centroid of the cluster, so that the maximum aggregated feature can better represent the sample.

Contributions. The main contributions of this work include the following:

- We first demonstrate that the point cloud features, discarded by the max-pooling module of a point cloud network, can be recycled and used as a feature augmentation method for contrastive learning.
- We propose a two-branch contrastive learning framework, which incorporates a cross-branch contrastive learning loss and an intra-branch contrastive learning loss.
- We perform extensive experiments to evaluate our proposed method on three downstream tasks, namely object classification, few-shot learning, and part segmentation on synthetic and real datasets of varying scales. The results show that our method achieves competitive if not better results compared to the state-of-the-art baselines, and does so with significantly less training time and fewer training samples.
- We perform ablation studies analyzing the effects of individual loss terms and their combinations on the performance.

2. Related Work

2.1. Representation Learning in 3D Point Clouds

Deep neural networks have been proven to be effective models to learn the representations of structured data, such as 2D images. However, the unordered structure of 3D data, and the requirement for permutation invariance introduce additional challenges for representation learning. To address these problems, many works on 3D point clouds has been presented in recent years [14,29,31]. According to the input data type of a neural network, point cloud representation methods can be divided into three categories: multi-view-based, volumetric-based and point-based methods [9].

Multi-view-based methods project a 3D shape onto multiple views and extract view-wise features by 2D image models. Hang *et al.* [22] proposed MVCNN, which uses view-based descriptors to represent 3D shapes. To learn local multi-view descriptors of point clouds, Li *et al.* [15] presented an end-to-end framework that performs in-network multi-view rendering with optimizable view points.

Volumetric-based methods voxelize 3D point clouds into grids, which are suitable for 3D CNN models. Maturana *et al.* [17] introduced VoxNet, which is one of the pioneering works utilizing 3D CNN to predict the occupancy grids generated by voxelized point clouds. Zhou *et al.* [38] presented PVD, which combines denoising diffusion models with the hybrid, point-voxel representation of 3D shapes.

Point-based methods, different from the above two categories, directly work on raw point clouds without any voxelization or projection. A classic backbone network in this area is PointNet [18], which uses a max pooling layer as a symmetric function, which enables permutation invariance. Another popular baseline work is DGCNN [28], which semantically groups points by dynamically updating a graph of relationships from layer to layer. This approach captures local geometric features of point clouds while still maintaining permutation invariance.

2.2. Contrastive Learning

Different approaches have been investigated to devise a pre-training architecture to enhance the learning of repre-

Model	Mean of no. of max pnts	*std* of no. of max pnts	Mean of no. of recycled pnts	*std* of no. of recycled pnts	Mean of utilized pnts	*std* of utilized pnts
PointNet	219.98	36.48	189.93	36.01	283.99	50.43
DGCNN	337.29	78.47	474.36	247.21	693.93	269.10

Table 1. **Point utilization analysis.** This table shows how many points out of 2048 input points are utilized after max pooling, and recycled with our proposed method. std represents the standard deviation. The first two columns show the statistic of points by max pooling. The next two columns show the statistics of recycled points by our proposed ToThePoint. The last two columns show the statistics of total utilized points in our proposed method.

sentation [7]. Sauder *et al.* [20] presented a self-supervised learning method operating on raw point clouds, wherein a neural network is trained to reconstruct a point cloud whose parts have been randomly displaced. Wang *et al.* [26] proposed OcCo, an unsupervised pre-training method, which consists of a mechanism to generate masked point clouds via view-point occlusions, and a completion task to reconstruct the occluded point clouds.

Contrastive learning has been one of the most popular self-supervised representation learning methods for 2D image data in recent years. It encourages augmentations of the same input to have more similar representations compared to augmentations of different inputs [10]. Chen *et al.* [5] proposed SimCLR, which learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. Unlike SimCLR, BYOL, proposed by Grill *et al.* [8], learns the representation by predicting previous versions of its outputs, without using negative pairs.

Motivated by the success of contrastive learning with 2D images, research on applying contrastive learning to 3D point clouds has emerged in the past few years [34, 36]. Based on BYOL, Huang *et al.* [11] devised STRL, which excludes negative pairs in contrastive learning on 3D point clouds and constructs a stable and invariant representation through a moving average target network. Referring to cross-model work in 2D vision, Afham *et al.* [2] proposed CrossPoint, which utilizes joint contrastive learning of imposing intra-modal and cross-modal correspondences to obtain more generic and transferable point cloud features.

3. Proposed Method

We propose a novel contrastive learning approach, referred to as ToThePoint, which achieves comparable, if not better, performance than the state-of-the-art (SOTA) baselines, with a much faster run-time and by using significantly fewer numbers of training data. Different from traditional contrastive learning methods, which maximize agreement between features obtained from a pair of point clouds formed only with different types of augmentation, ToThePoint recycles the features discarded in the max pooling operation of the point cloud network and relabels them as positive samples. An overview of our method is shown in Fig. 1, and the details will be provided in the following sections after providing the motivation first.

3.1. Motivation

SOTA methods on self-supervised learning of 3D point clouds [2, 11] require tens of thousands or even millions of samples (as shown in Tab. 2) and incur long training times. Chen et al. [4] showed that a significant amount of point features are discarded after max-pooling. As depicted in Fig. 1, max-pooling operation simply retains the maximum latent features and completely discards the rest, wasting some valuable latent features in the process. The percentage of discarded points and its effect on the model's performance have been analyzed by Chen et al. [4], which demonstrates the benefit of increasing the number of points used for the model training process. Inspired by this, instead of performing self-supervised training on a large of amount data and discarding point features, our proposed method only needs a small amount of data and recycles the discarded features for self-supervised training and achieves competitive if not better results compared to SOTA baselines.

We design an experiment to investigate how many points are utilized in the max pooling, and how many points are recycled in each point cloud sample, during the whole pre-training process. If a point has feature value participating the downstream task, we say the point is utilized, otherwise the point is referred to as discarded. We use PointNet and DGCNN as backbones and ShapeNet as the dataset. The number of training epochs is 800. For each point cloud sample in each training epoch, we record the number of utilized points, the number of recycled points by our method described in Sec. 3, and obtain the mean and standard deviation. The results are shown in Tab. 1. For instance, for the DGCNN backbone, in the original DGCNN, on average about 337.29 out of 2048 points are utilized by max pooling for a point cloud sample. However, applying our method, 474.36 more points are recycled on average, which increases the number of utilized points from 337.29 to 693.93. By utilizing more points, the feature embedding output of the backbone network can have a better description of the object's shape.

3.2. Data Augmentation on Point Clouds

The overall structure of our proposed ToThePoint is shown in Fig. 2. ToThePoint is composed of two branches. The input to both branches is a normalized set $\mathbf{P}$ of N-many 3D points. As commonly done in contrastive learning, we apply multiple types of transformations, more specifically,

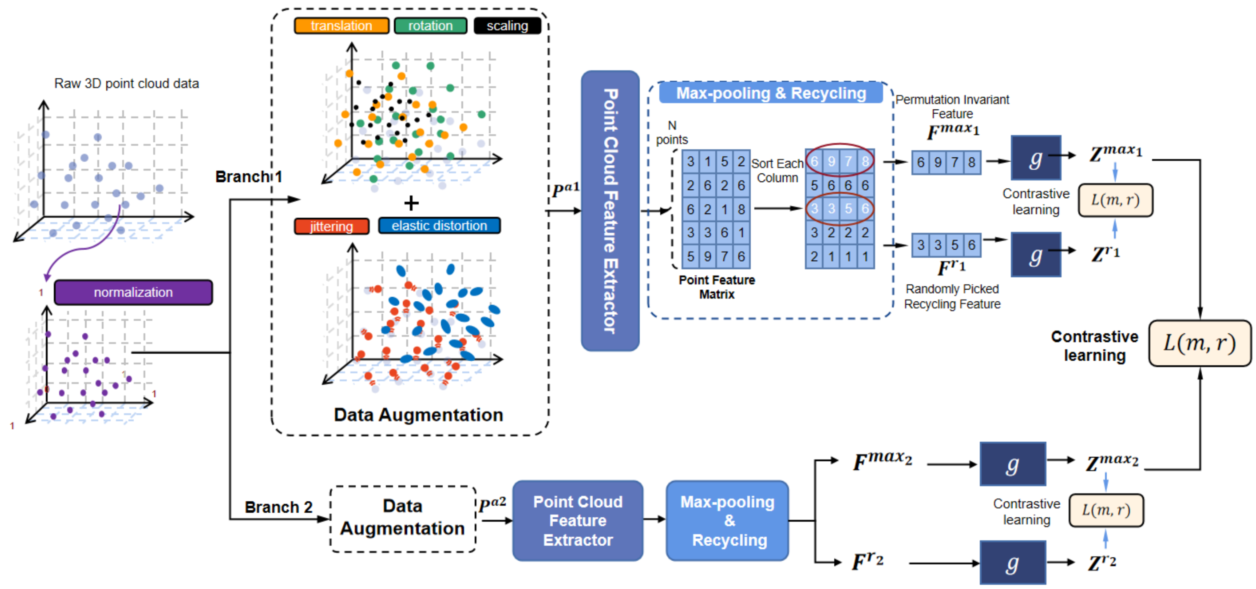


Figure 2. **The overall structure of the ToThePoint framework.** After the two-branch data augmentation, the data would be fed into the point cloud feature extractor. In the Max-pooling & Recycling Module, each branch gets a maximum aggregated feature and a recycled point feature. The Point Cloud Feature Extractor and the projection head g are trained to maximize the agreement between the maximum aggregated features and the recycled point cloud features of the same branch and the agreement between the maximum aggregated features of the two branches. After training is completed, we throw away projection head g and use Point Cloud Feature Extractor and maximum aggregated feature for downstream tasks.

rotation, scaling, translation, jitter and elastic deformation, on the normalized point cloud to augment the data. Data augmentation is done at both branches by using different parameters, e.g. different angles for rotation etc. These two branches allow having more augmented data from the same point cloud input. The augmented point cloud data is denoted by $\mathbf{P^{a1}} \in \mathbb{R}^{\mathbf{N\times 3}}$ and $\mathbf{P^{a2}} \in \mathbb{R}^{\mathbf{N\times 3}}$ for branches 1 and 2, respectively. The two augmented samples are then fed into the same point feature extraction backbone in both branches.

3.3. Recycling Features for Feature Augmentation

The backbone used for point feature extraction learns a function $f(\cdot)$ producing powerful representations of $\mathbf{P^{a1}}$ and $\mathbf{P^{a2}}$, i.e. $f(\mathbf{P^{a1}})$ and $f(\mathbf{P^{a2}})$ in M-dimensional space. After M-dimensional latent features are extracted for N points, traditional max-pooling is applied to obtain permutation-invariant features. In other words, the $N \times M$ matrix is sorted in descending order along the dimension N, as depicted in Fig. 1. Then, features in the first row are kept, which are called 'maximum features' and denoted as $\mathbf{F^{max_1}}$ and $\mathbf{F^{max_2}}$ in Fig. 2 for first and second branches, respectively. $\mathbf{F^{max_1}}$ and $\mathbf{F^{max_2}} \in \mathbb{R}^{M\times 1}$.

After this step, instead of discarding all the remaining features, we randomly pick one of the remaining $N-1$ rows, and use its features for data augmentation in feature space. Since selecting a fixed row does not provide better performance, we select it randomly. These features are called 'recycled point features' and denoted as $\mathbf{F^{r_1}}$ and $\mathbf{F^{r_2}}$ in Fig. 2 for the first and second branches, respectively. Subsequently, these feature vectors are passed through g, which is composed of two fully connected layers. The projection vectors of $\mathbf{F^{max_j}}$ and $\mathbf{F^{r_j}}$ are denoted by $\mathbf{z^{max_j}}$ and $\mathbf{z^{r_j}}$, respectively, where $\mathbf{z^{max_j}}=g(\mathbf{F^{max_j}})$ and $\mathbf{z^{r_j}}=g(\mathbf{F^{r_j}})$ and $j \in \{1,2\}$. Then, contrastive learning is applied between $\mathbf{z^{max_1}}$ and $\mathbf{z^{r_1}}$, between $\mathbf{z^{max_2}}$ and $\mathbf{z^{r_2}}$, and between $\mathbf{z^{max_1}}$ and $\mathbf{z^{max_2}}$.

The aforementioned steps are done for each point cloud sample $\mathbf{P_s}$ to obtain $\mathbf{z_s^{max_j}}$ and $\mathbf{z_s^{r_j}}$. For clarity, we dropped the subscript s in the above description.

3.4. Contrastive Learning Incorporating Recycled Features

The goal of contrastive learning is to maximize the similarity between $\mathbf{z_i^{max_j}}$ and $\mathbf{z_i^{r_j}}$ while minimizing the similarity to the projection vectors of all the other point cloud samples ($\mathbf{z_k^{max_j}}$, $\mathbf{z_k^{r_j}}$), where $k \neq i$, in the same batch. We use NT-Xent loss without memory bank for contrastive representation. The loss function $L(i,m,r)$ for a positive pair of examples $\mathbf{z_i^{max_j}}$ and $\mathbf{z_i^{r_j}}$ is defined as:

$$L(i,m,r) = -log\frac{exp(s(\mathbf{z_i^{max_j}},\mathbf{z_i^{r_j}})/\tau)}{\sum_{\substack{k=1\\k\neq i}}^{B} exp(s(\mathbf{z_i^{max_j}},\mathbf{z_k^{max_j}})/\tau)+\sum_{k=1}^{B} exp(s(\mathbf{z_i^{max_j}},\mathbf{z_k^{r_j}})/\tau)} \tag{1}$$

where B is the mini-batch size, τ is a temperature parameter and $s(\cdot)$ denotes the cosine similarity function. Our intra-branch contrastive loss function $\mathcal{L}_j^{ib-cl}$ for branch j and for a mini-batch is expressed as:

$$\mathcal{L}_j^{ib-cl} = \frac{1}{2B} \sum_{i=1}^{B} [L(i, m, r) + L(i, r, m)]. \tag{2}$$

As mentioned above, in addition to performing contrastive learning and feature alignment between the 'maximum features' and 'recycled features' of point cloud data, we also introduce feature alignment between 'maximum features' obtained from the two branches of our network. This enhances the representation and learning capabilities of the network, which is supported by the experimental results provided in Sec. 4.2. As mentioned above, the projection vectors of $\mathbf{F}^{\mathbf{max_1}}$ and $\mathbf{F}^{\mathbf{max_2}}$ are denoted by $\mathbf{z}^{\mathbf{max_1}}$ and $\mathbf{z}^{\mathbf{max_2}}$, respectively, where 1 and 2 refer to first and second branches. The loss function $L(i, max_1, max_2)$, for positive examples $\mathbf{z}_\mathbf{i}^{\mathbf{max_1}}$ and $\mathbf{z}_\mathbf{i}^{\mathbf{max_2}}$, is written as:

$$L(i, max_1, max_2) = \\ -\,log \frac{exp(s(\mathbf{z}_\mathbf{i}^{\mathbf{max_1}}, \mathbf{z}_\mathbf{i}^{\mathbf{max_2}}))/\tau}{\sum_{\substack{k=1 \\ k\neq i}}^{B} exp(s(\mathbf{z}_\mathbf{i}^{\mathbf{max_1}}, \mathbf{z}_\mathbf{k}^{\mathbf{max_1}})/\tau) + \sum_{k=1}^{B} exp(s(\mathbf{z}_\mathbf{i}^{\mathbf{max_1}}, \mathbf{z}_\mathbf{k}^{\mathbf{max_2}})/\tau)} \tag{3}$$

where B, τ and $s(\cdot)$ are the same as those in Eq. (1). Our inter-branch contrastive loss function $\mathcal{L}^{cb-cl}$ for a mini-batch is expressed as:

$$\mathcal{L}^{cb-cl} = \frac{1}{2B} \sum_{i=1}^{B} [L(i, max_1, max_2) + L(i, max_2, max_1)]. \tag{4}$$

In summary, this intra-branch contrastive learning process maximizes the agreement between the permutation-invariant features (coming from the max-pooling operation) and the recycled features that are randomly picked from the discarded ones. The inter-branch or cross-branch contrastive learning, using the 'maximum features' from two branches allows sharing of further semantic information.

Finally, the total loss function is obtained by combining $\mathcal{L}_1^{ib-cl}$, $\mathcal{L}_2^{ib-cl}$ and $\mathcal{L}^{cb-cl}$, as in Eq. (5), during training, where $\mathcal{L}_1^{ib-cl}$ and $\mathcal{L}_2^{ib-cl}$ enforce similarity between the maximum aggregated features and recycled point cloud features, while $\mathcal{L}^{cb-cl}$ enforces similarity between different point cloud transformations.

$$\mathcal{L} = \mathcal{L}_1^{ib-cl} + \mathcal{L}_2^{ib-cl} + \mathcal{L}^{cb-cl}. \tag{5}$$

4. Experimental Results

We compare our ToThePoint with SOTA baselines and present its efficiency and effectiveness through extensive performance evaluations. Note that some experimental results are marked with / due to the fact that either the baseline is not self-supervised or does not report results on that dataset.

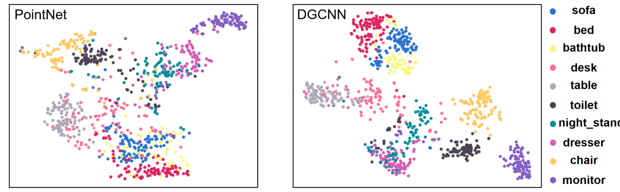


Figure 3. **Visualization of learned features.** The extracted features for each sample in the ModelNet10 test set are visualized using t-SNE. Both models are pre-trained on ShapeNet.

4.1. Pre-training

Dataset. We use the ShapeNet [3] dataset for self-supervised training of the proposed ToThePoint and the baselines. ShapeNet contains more than 50,000 CAD models from 55 categories. As for the downstream tasks, we perform fully supervised and few-shot point cloud classification on the ScanObjectNN [24], ModelNet40 [32] and ModelNet40-C [23] datasets, and parts segmentation on the ShapeNet-Part [35]. The details about the datasets used for the downstream tasks are provided in Sec. 4.2.

Implementation Details. We use PointNet [18] and DGCNN [28] as the backbones for point cloud feature extraction. PointNet is an MLP-based framework and DGCNN is built on graph convolutional networks. As shown in Fig. 2, feature extraction is followed by max-pooling and recycling, and the features are then sent to a 2-layer MLP. M is 1024 in our experiments, and the $\mathbf{z}$ vectors are 256-dimensional. For fair comparison, the same backbone is pre-trained first by different self-supervised learning methods, and then used with different downstream tasks. We use Adam [13] as the optimizer, with weight decay of 1×10^{-4} and initial learning rate of 1×10^{-3}.

Visualization of Learned Features. We first obtain the point cloud embeddings, from PointNet and DGCNN backbones, by using the proposed ToThePoint self-supervised learning method. We use the ModelNet10 dataset, which contains points from 10 categories. We use t-SNE [25] to visualize these embeddings and show them in Fig. 3. We observe that both pre-trained models can successfully differentiate the majority of samples in most categories, with the exception of dressers and nightstands. This is expected, since objects in these two categories usually look similar.

4.2. Evaluation of Downstream Tasks

(i) 3D object classification. We perform the classification experiments on the ModelNet40 [32], ModelNet40C [23] and ScanObjectNN [24] datasets to demonstrate the generalizability of ToThePoint in learning 3D shape representations. The samples in ModelNet40 are obtained from 3D CAD models. It contains 12,331 objects (9,843 for training and 2,468 for testing) from 40 categories. ModelNet40-C is a comprehensive dataset for benchmark-

Self-supervised Learning				Downstream Task			
Method	Num of Samples	Running Time (s.)		ModelNet40C		ScanObjectNN	
		DGCNN	PointNet	DGCNN	PointNet	DGCNN	PointNet
Rand	/	/	/	81.82±0.07	79.63±0.25	85.66±0.45	78.16±0.54
Jigsaw [20]	9.8K	161.27	17.97	83.19±0.26	80.14±0.35	86.33±0.06	79.46±0.17
OcCo [26]	9.8K	2762.7	1307.73	82.47±0.15	79.89±0.52	86.19±0.39	79.63±0.16
STRL [11]	57.4 k	238.65	175.4	82.66±0.38	80.43±0.19	86.17±0.32	80.32±0.21
CrossPoint [2]	43.7 k pnts & 1.05M img	1115.86	344.95	83.69±0.29	**81.12±0.44**	86.32±025	79.90±0.03
ToThePoint	**260**	**5.38**	**1.3**	**83.80±0.32**	80.97±0.27	**86.46±0.21**	**80.64±0.31**

Table 2. **3D object classification comparison.**. We report mean and standard deviation over 3 runs ToThePoint outperforms all the other methods on the ScanObject dataset with both backbones. On the ModelNet40C dataset, ToThePoint provides the best and second-best performance when DGCNN and PointNet are used as backbones, respectively. ToThePoint achieves these accuracies with only a fraction of training samples needed by other methods.

ing corruption robustness of 3D point cloud classification approaches. It contains 2,468 point clouds from 40 classes, 15 corruption types, and 5 severity levels. It also provides a detailed taxonomy of the constructed corruption types. To be able to evaluate on close to real-life examples, we choose LiDAR corruption and the severity 1. ScanObjectNN is a more realistic and challenging 3D point cloud classification dataset, which includes occluded objects taken from actual indoor scans of real world scenes. It has 2,880 objects (2304 for training and 576 for testing) from 15 categories.

We first evaluate the performance of different approaches on the downstream tasks by using DGCNN and PointNet as the backbones. The results are presented in Tab. 2, where the left side shows the average run-time of 30 epochs during pre-training process of self-supervised learning as well as the number of samples needed to well-train the backbone. The right side in Tab. 2 presents the 3D object classification accuracy on the ModelNet40C and ScanObjectNN datasets. As can be seen, ToThePoint outperforms all the other methods on the ScanObject dataset with both backbones. On the ModelNet40C dataset, ToThePoint provides the best and second best performance when DGCNN and PointNet are used as backbones, respectively. It should be noted that ToThePoint achieves these accuracy levels with only a fraction of training samples needed by other methods. For instance, CrossPoint, which provides slightly better performance on the ModelNet40C dataset (with PointNet as the backbone) has average epoch run-time of 344.95 s and needs 43,700 point cloud samples. It also uses over 1 million 2D images to enrich representation ability. In contrast, our method needs only 260 point cloud samples with an average epoch run-time of 1.3 s. It should be noted that for ModelNet40C, we use the part with LiDAR corruption, providing a more realistic test scenario. The high performance of ToThePoint on this dataset demonstrates its ability to generalize well. In summary, our ToThePoint needs fewer training samples for the pre-training stage compared to CrossPoint, and this, in turn, significantly reduces the pre-training time for our approach. At the same time, our method (with fewer training samples and not requiring any other input with different modality) achieves comparable if not better accuracy than CrossPoint.

Method	Accuracy	
	ModeNet40	**ScanObjectNN**
3D-GAN [29]	81.85	37.01
Latent-GAN [1]	87.64	71.94
3D-PointCapsNet [37]	76.62	53.70
SO-Net [14]	87.03	/
PointNet + Jigsaw [20]	51.90	35.11
PointNet + OcCo [26]	86.67	68.84
PointNet + STRL [11]	88.05	73.67
PointNet + CrossPoint [2]	**88.82**	72.29
PointNet + ToThePoint (Ours)	85.62	**74.70**
DGCNN + Jigsaw [20]	55.71	36.31
DGCNN + OcCo [26]	88.61	78.14
DGCNN + STRL [11]	**90.60**	78.14
DGCNN + CrossPoint [2]	90.03	81.43
DGCNN + ToThePoint (Ours)	89.22	**81.93**

Table 3. **SVM classification results on ModelNet40 and ScanObjectNN.** We perform the SVM evaluation method [1], to compare ToThePoint and baselines with PointNet and DGCNN used as backbones. On the more challenging ScanObjectNN dataset, proposed ToThePoint achieves the best performance with both backbones. On ModelNet40 dataset, ToThePoint provides the 3rd best performance after CrossPoint and STRL, which require a lot more training samples.

In the next set of 3D object classification experiments, we follow the same procedure as in [1, 11, 26, 30]. We train a model in a self-supervised manner on the ShapeNet dataset [3] and freeze it. Let's denote this model by M_{sst}. Then, we train a linear Support Vector Machine (SVM) [6] by using the training data of the ModelNet40 dataset. 1024 points are randomly sampled from each object in the dataset. We use the samples in the training set of ModelNet40 as input, feed them into M_{sst} and use the embeddings provided by M_{sst} to train the SVM. Then, we evaluate the SVM on the test data of ModelNet40. We do this with the ScanObjectNN dataset as well. This SVM classifier is used to evaluate the models' representation ability of 3D point cloud data. For the self-supervised methods we use PointNet [18] and DGCNN [28] as the backbone models. The results are summarized in Tab. 3. From the table, for ScanObjectNN, which is obtained from LiDAR scanning of real-world scenes, and thus, is more challenging, ToThePoint provides the best performance with both backbone models. For ModelNet40, ToThePoint provides the third best performance after CrossPoint and STRL when PointNet and DGCNN are used as the backbone respectively. However, as shown in Tab. 2, compared to both CrossPoint and STRL, ToThePoint requires significantly less number of training samples, and training time.

(ii) Few-shot 3D object classification. Few-shot learning (FSL) aims to perform prediction on objects belonging

Self-supervised Learning				Downstream Task (Few-shot point cloud classification)			
Method	Num of Samples	Running Time (s.)		5-way		10-way	
		DGCNN	PointNet	10 shot	20 shot	10 shot	20 shot
3D-GAN [29]	/	/	/	87.72 ± 5.44	91.98 ± 3.91	81.31 ± 4.75	84.87 ± 5.10
DGCNN Rand	/	/	/	81.13 ± 8.68	85.96 ± 6.60	72.86 ± 7.33	81.03 ± 5.12
DGCNN cTree [21]	200	**2.53**	2.53	86.37 ± 6.29	89.60 ± 5.62	81.03 ± 4.14	83.98 ± 4.75
DGCNN Jiasaw	9.8K	161.27	17.97	87.06 ± 5.93	88.60 ± 6.07	79.20 ± 4.41	83.21 ± 4.40
DGCNN OcCo [26]	9.8K	2762.7	1307.73	88.46 ± 8.15	94.13 ± 3.73	85.21 ± 3.91	87.11 ± 3.93
DGCNN CrossPoint [2]	43.7 k pnts & 1.05 M images	1115.86	344.95	91.12 ± 4.93	94.56 ± 3.23	86.29 ± 4.77	88.96 ± 4.39
DGCNN ToThePoint	260	5.38	**1.3**	**92.73 ± 4.79**	**95.10 ± 2.95**	**87.91 ± 4.29**	**91.06 ± 3.58**

(a) Experiment results on ModelNet40

Self-supervised Learning				Downstream Task (Few-shot point cloud classification)			
Method	Num of Samples	Running Time (s.)		5-way		10-way	
		DGCNN	PointNet	10 shot	20 shot	10 shot	20 shot
3D-GAN [29]	/	/	/	68.20 ± 7.84	72.68 ± 9.76	53.93 ± 4.73	59.62 ± 4.66
DGCNN Rand	/	/	/	61.80 ± 7.60	64.10 ± 8.67	42.13 ± 3.96	49.11 ± 6.08
DGCNN cTree [21]	200	**2.53**	2.53	50.76 ± 7.11	72.68 ± 9.76	37.46 ± 4.03	41.76 ± 4.72
DGCNN Jiasaw	9.8K	161.27	17.97	67.16 ± 8.32	72.76 ± 9.39	50.75 ± 5.40	58.75 ± 5.33
DGCNN OcCo [26]	9.8K	2762.7	1307.73	75.80 ± 5.48	82.06 ± 5.90	63.43 ± 4.60	71.48 ± 4.28
DGCNN CrossPoint [2]	43.7 k pnts & 1.05 M images	1115.86	344.95	77.24 ±7.13	83.68 ±5.51	66.61 ±3.96	73.57±4.72
DGCNN ToThePoint	260	5.38	**1.3**	**78.13 ± 7.29**	**83.80 ± 6.07**	**66.71 ± 5.40**	**74.21 ± 4.95**

(b) Experiment Results on ScanObjectNN

Table 4. **Few-shot object classification results.** We report mean and standard deviation over 30 runs. The top results for each backbone are shown in bold. Our proposed ToThePoint needs only a few samples in the few-shot learning task and improves the few-shot accuracy in all the reported settings.

to classes, which were not seen during training, with only a few labeled samples. We conduct N-way K-shot experiments, wherein the model is tested on N classes and each class contains K samples. In the FSL experiments, we use ModelNet40 and ScanObjectNN for evaluation. Since there is no standard split for FSL in either of these datases, for a fair comparison with earlier approaches cTree [21] and OcCo [26], we run K-way N-shot experiments 30 times, and report the mean and standard deviation in Tab. 4. Similar to above, the left side of the table shows the average run-time of 30 epochs during pre-training process of self-supervised learning as well as the number of samples needed to well-train the backbone. As can be seen, our ToThePoint outperforms all the prior methods in all the FSL settings, using DGCNN as backbones, on both datasets. It should be noted that, in the few-shot object classification task, ToThePoint outperforms CrossPoint while CrossPoint has slightly higher accuracy in the linear SVM evaluation presented in Tab. 3. These results support that ToThePoint has better representation and generalization ability considering that there are fewer samples in the few-shot learning task. We attribute this to the fact that the two-branch framework, incorporating recycling of discarded features, significantly enhances the ability of representing 3D point clouds, which allows the pre-trained model to perform well in the few-shot classification task.

(iii) 3D object part segmentation. We perform object part segmentation evaluation on the widely used ShapeNet-Part dataset [35]. It contains 16,881 3D objects from 16 classes, with a total of 50 parts annotated. We first pre-train the DGCNN and PointNet backbones, perform part segmentation using our method on the ShapeNet-Part dataset, and fine-tune the sequence segmentation in the ShapeNet-Part dataset in an end-to-end manner. We present the mean Intersection-over-Union (IoU) and the overall accuracy (OA) in Tab. 5. Mean IoU is obtained by averaging the IoU for each part of an object before averaging the values for each object class. Part segmentation utilizing ToThePoint with pre-trained DGCNN backbone performs 0.34% better than the randomly initialized DGCNN backbone. This demonstrates that ToThePoint gives the feature extractors a better weight initialization. Gains in overall accuracy over the previous self-supervised learning frameworks show that ToThePoint tends to capture fine-grained part-level properties that are important for part segmentation by incorporating cross-branch and intra-branch losses together.

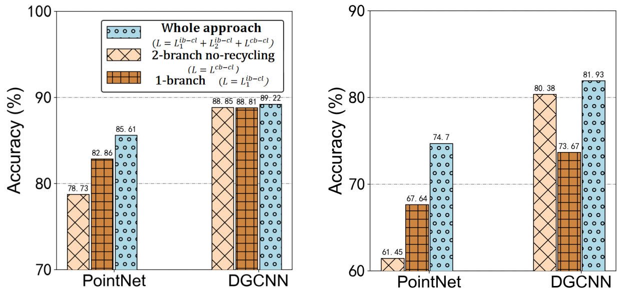
Self-supervised Learning				Downstream Task (Part segmentation)			
Method	Num of Samples	Running Time (s.)		DGCNN		PointNet	
		DGCNN	PointNet	Mean IoU	OA	Mean IoU	OA
Rand	/	/	/	85.16	94.43	84.48	93.82
Jigsaw [20]	9.8K	161.27	17.97	85.34	94.42	84.27	93.67
OcCo [26]	9.8K	2762.7	1307.73	85.32	**94.5**	84.56	93.77
CrossPoint [2]	43.7 k pnts & 1.05 M images	1115.86	344.95	85.38	94.44	84.77	93.97
ToThePoint	**260**	**5.38**	**1.3**	**85.5**	94.44	**84.91**	**94.05**

Table 5. **Part segmentation results on ShapeNet-Part dataset.** Mean IoU and overall accuracy (OA) are reported. All self-supervised models are initialized with pre-trained feature extractors.

4.3. Ablation Studies

In this section, we present the results of our ablation studies investigating the effects of the individual loss terms in Eq. (4) and their combinations on the performance.

Effect of the two-branch construction. Our approach aims to pre-train a model effectively while requiring much fewer 3D cloud point samples, as detailed in Sect. 3. We hypothesize that two-branch construction and the interaction between branches, via our cross-branch contrastive loss $\mathcal{L}^{cb-cl}$, allow to capture a better representation than using a single branch. Moreover, with two-branch construction data and feature variation increases both through different ways of data augmentation and also through feature augmentation via recycling. To verify this, we performed experiments to compare the performances of two-branch and one-branch networks by training the models on the ModelNet40 and ScanObjectiveNN datasets. We used PointNet and DGCNN as backbones, and performed linear SVM classifier-based evaluation during training. The results in Fig. 4 show that the whole approach with the two-branch construction performs better than the one-branch construction. The improvement margin is much higher (8.27% with DGCNN



(a) Accuracy on ModelNet40

(b) Accuracy on ScanObjectNN

Figure 4. **The ablation study results on effects of individual loss terms**. Blue bar represents the whole approach using all 3 loss terms, light orange corresponds to using two branches but no recycling and dark orange corresponds to using one branch with recycling. Classification accuracies are presented on ModelNet40 and ScanObjectNN datasets.

and 7.06% with PointNet backbones) on the ScanObjectNN dataset (obtained from actual indoor scans of real-world scenes, containing occluded objects and points are more unevenly distributed) compared to ModelNet40 (sampled from CAD models and evenly distributed). These findings support the benefit of two-branch construction and the cross-branch contrastive loss $\mathcal{L}^{cb-cl}$.

Effect of recycling. In this experiment, we compare the performances with and without doing recycling (i.e. with and without feature aggregation) on the ModelNet40 and ScanObjectiveNN datasets. We used PointNet and DGCNN as backbones, and performed linear SVM classifier-based evaluation during training. Fig. 4 clearly demonstrates the increased accuracy when recycling is employed, especially on the ScanObjectNN dataset, and when PointNet is used as the backbone. More specifically, on the ScanObject dataset, the accuracy values with recycling are 74.70% and 81.93% compared to 61.45% and 80.38% with no recycling when PointNet and DGCNN are used as backbones, respectively. These results show that recycled features contain additional semantics that cannot be captured by the 'maximum features', kept after max-pooling, alone. This could be utilized to adjust the classified position on the hypersphere by minimizing the difference between the 'maximum features' and 'recycled features'.

Effect of loss. The results in Fig. 4 show that $\mathcal{L}_1^{ib-cl}$ and $\mathcal{L}_2^{ib-cl}$ have a greater impact on the feature extraction capabilities of a simpler backbone (PointNet), and $\mathcal{L}^{cb-cl}$ has more impact on a more complex backbone (DGCNN).

Analysis of Results. To see the effects of different constructions and loss terms, i.e. the results of the ablation studies, side-by-side, we present the accuracy reductions for each configuration in Tab. 6. As can be seen with only 1 branch, the accuracy would decline by 7.06% and 8.27% on the ScanObjectNN dataset, when PointNet and DGCNN are used as backbones, respectively. The decline would be 2.76% and 0.41% on the ModelNet40 dataset, with PointNet and DGCNN backbones, respectively.

Component	2-Branch, no recycling		1-branch with recycling	
Backbone	PointNet	DGCNN	PointNet	DGCNN
ModelNet40	6.89%	0.37%	2.76%	0.41%
ScanObjectNN	13.25%	1.56%	7.06%	8.27%

Table 6. **The accuracy reduction caused by different configurations.**

With two branches and no recycling, the accuracy would decline by 13.25% and 1.56% on the ScanObjectNN dataset, when PointNet and DGCNN are used as backbones, respectively. The decline would be 6.89% and 0.37% on the ModelNet40 dataset, with PointNet and DGCNN backbones, respectively.

It can be observed that, the two-branch construction with recycling provides more performance increase on the ScanObjectNN dataset than ModelNet40. Since ScanObjectNN dataset includes point clouds with unevenly distributed points, feature extraction is more challenging, and we can argue that recycling features could be more helpful.

As for the backbones, the improvement obtained is much higher when PointNet (a simpler model with less feature extraction ability) is used as the backbone, compared to DGCNN (a more complicated model with better feature extraction ability). In addition, with PointNet, recycling approach is more beneficial than using two-branches with no recycling. This two-branches with no recycling provide better performance than one-branch with recycling in DGCNN. These findings show that the combination of the two-branch construction and recycling mechanism has the ability to enhance the performance of models with varying complexity and feature extraction ability.

5. Conclusion

We have proposed ToThePoint as a novel and very efficient contrastive learning framework. In addition to using traditional data augmentation, ToThePoint performs feature augmentation by recycling point cloud features, which would otherwise be discarded after max-pooling operation of a point cloud feature extraction network. ToThePoint is a fast, self-supervised pre-training architecture to learn 3D point cloud representations. It has been evaluated on several benchmark datasets for various downstream tasks, such as 3D object classification, few-shot classification and parts segmentation. Ablation studies have been conducted based on different backbones. Results have shown that ToThePoint achieves comparable if not better performance than baselines while requiring significantly fewer training samples and less training time. In future work, we will investigate whether using more branches or recycling more features can provide additional benefit.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80, pages 40–49, 2018. 6

[2] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 9892–9902, 2022. 3, 6, 7

[3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv*, 1512.03012, 2015. 5, 6

[4] Jiajing Chen, Burak Kakillioglu, Huantao Ren, and Senem Velipasalar. Why discard if you can recycle?: A recycling max pooling module for 3d point cloud analysis. In *CVPR*, pages 549–557, 2022. 2, 3

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020. 1, 3

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 6

[7] Benjamin Eckart, Wentao Yuan, Chao Liu, and Jan Kautz. Self-supervised learning on 3d point clouds by learning discrete generative models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 8248–8257, 2021. 3

[8] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. 1, 3

[9] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4338–4364, 2021. 2

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 9726–9735, 2020. 1, 3

[11] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *ICCV*, pages 6515–6525, 2021. 1, 3, 6

[12] Xu Ji, Andrea Vedaldi, and João F. Henriques. Invariant information clustering for unsupervised image classification and segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*, pages 9864–9873. IEEE, 2019. 1

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[14] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 9397–9406, 2018. 2, 6

[15] Lei Li, Siyu Zhu, Hongbo Fu, Ping Tan, and Chiew-Lan Tai. End-to-end learning local multi-view descriptors for 3d point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 1916–1925, 2020. 2

[16] Jianming Lv, Weihang Chen, Qing Li, and Can Yang. Unsupervised cross-dataset person re-identification by transfer learning of spatial-temporal patterns. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 7948–7956, 2018. 1

[17] Daniel Maturana and Sebastian A. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pages 922–928, 2015. 2

[18] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 77–85, 2017. 2, 5, 6

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5099–5108, 2017. 2

[20] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 12942–12952, 2019. 2, 3, 6, 7

[21] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. 7

[22] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*, pages 945–953, 2015. 2

[23] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z. Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv*, 2201.12296, 2022. 5

[24] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classifi-

cation model on real-world data. In *ICCV*, pages 1588–1597, 2019. 5

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 5

[26] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9762–9772. IEEE, 2021. 2, 3, 6, 7

[27] Ning Wang, Yibing Song, Chao Ma, Wengang Zhou, Wei Liu, and Houqiang Li. Unsupervised deep tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 1308–1317, 2019. 1

[28] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019. 2, 5, 6

[29] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 82–90, 2016. 2, 6, 7

[30] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016. 6

[31] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 9621–9630, 2019. 2

[32] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 5

[33] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, pages 8372–8381, 2021. 1

[34] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision - ECCV 2020 - 16th European Conference, Proceedings, Part III*, volume 12348, pages 574–591, 2020. 1, 2, 3

[35] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6):210:1–210:12, 2016. 5, 7

[36] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, pages 10232–10243. IEEE, 2021. 1, 3

[37] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 1009–1018, 2019. 6

[38] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, pages 5806–5815, 2021. 2