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 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.
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, 19]. 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-
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean of no. of max pnts</th>
<th>std of no. of max pnts</th>
<th>Mean of no. of recycled pnts</th>
<th>std of no. of recycled pnts</th>
<th>Mean of utilized pnts</th>
<th>std of utilized pnts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet</td>
<td>219.98</td>
<td>66.48</td>
<td>189.93</td>
<td>20.01</td>
<td>251.99</td>
<td>50.43</td>
</tr>
<tr>
<td>DGCNN</td>
<td>337.29</td>
<td>78.47</td>
<td>474.36</td>
<td>247.21</td>
<td>693.91</td>
<td>269.10</td>
</tr>
</tbody>
</table>

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 amount of 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 693.91. 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 P of N-many 3D points. As commonly done in contrastive learning, we apply multiple types of transformations, more specifically,
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 $\mathbf{P}_{a1} \in \mathbb{R}^{N \times 3}$ and $\mathbf{P}_{a2} \in \mathbb{R}^{N \times 3}$ 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 $\mathbf{F}^{\text{max}1}$ and $\mathbf{F}^{\text{max}2}$ in Fig. 2 for first and second branches, respectively. $\mathbf{F}^{\text{max}1}$ and $\mathbf{F}^{\text{max}2} \in \mathbb{R}^{M \times 1}$.

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 $\mathbf{F}^{r1}$ and $\mathbf{F}^{r2}$ 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 $\mathbf{F}^{\text{max}1}$ and $\mathbf{F}^{\text{max}2}$ are denoted by $\mathbf{z}^{\text{max}1}$ and $\mathbf{z}^{\text{max}2}$, respectively, where $\mathbf{z}^{\text{max}1} = g(\mathbf{F}^{\text{max}1})$ and $\mathbf{z}^{\text{max}2} = g(\mathbf{F}^{\text{max}2})$ and $j \in \{1, 2\}$. Then, contrastive learning is applied between $\mathbf{z}^{\text{max}1}$ and $\mathbf{z}^{r1}$, between $\mathbf{z}^{\text{max}2}$ and $\mathbf{z}^{r2}$, and between $\mathbf{z}^{\text{max}1}$ and $\mathbf{z}^{\text{max}2}$.

The aforementioned steps are done for each point cloud sample $\mathbf{P}_s$ to obtain $\mathbf{z}^{\text{max}1}_s$ and $\mathbf{z}^{r1}_s$. 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 $\mathbf{z}^{\text{max}1}_i$ and $\mathbf{z}^{r1}_i$ while minimizing the similarity to the projection vectors of all the other point cloud samples ($\mathbf{z}^{\text{max}1}_k$, $\mathbf{z}^{r1}_k$), 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 $\mathbf{z}^{\text{max}1}_i$ and $\mathbf{z}^{r1}_i$ is defined as:

$$
L(i, m, r) = -\log \frac{\exp(s(\mathbf{z}^{\text{max}1}_i, \mathbf{z}^{r1}_i)/\tau)}{\sum_{k \neq i} \exp(s(\mathbf{z}^{\text{max}1}_i, \mathbf{z}^{\text{max}1}_k)/\tau) + \sum_{k \neq i} \exp(s(\mathbf{z}^{\text{max}1}_k, \mathbf{z}^{r1}_i)/\tau)}
$$

(1)
where $B$ is the mini-batch size, $\tau$ is a temperature parameter and $s(\cdot)$ denotes the cosine similarity function. Our intra-branch contrastive loss function $L_{ib-cl}$ for branch $j$ and for a mini-batch is expressed as:

$$L_{ib-cl} = \frac{1}{2B} \sum_{i=1}^{B} [L(i, m, r) + L(i, r, m)].$$ (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 $F_{\text{max}1}$ and $F_{\text{max}2}$ are denoted by $z_{\text{max}1}$ and $z_{\text{max}2}$, respectively, where 1 and 2 refer to first and second branches. The loss function $L(i, \text{max}_1, \text{max}_2)$, for positive examples $z_{\text{max}1}$ and $z_{\text{max}2}$, is written as:

$$L(i, \text{max}_1, \text{max}_2) = -\log \frac{\exp(s(z_{\text{max}_1}, z_{\text{max}_2})/\tau)}{\sum_{k \neq i} \exp(s(z_{\text{max}_1}, z_{\text{max}_k})/\tau) + \sum_{k=1}^{B} \exp(s(z_{\text{max}_i}, z_{\text{max}_k})/\tau)}.$$ (3)

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

$$L_{ib-cl} = \frac{1}{2B} \sum_{i=1}^{B} [L(i, \text{max}_1, \text{max}_2) + L(i, \text{max}_2, \text{max}_1)].$$ (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 $L_{ib-cl}^{1}$, $L_{ib-cl}^{2}$ and $L_{ib-cl}$, as in Eq. (5), during training, where $L_{ib-cl}^{1}$ and $L_{ib-cl}^{2}$ enforce similarity between the maximum aggregated features and recycled point cloud features, while $L_{ib-cl}$ enforces similarity between different point cloud transformations.

$$\mathcal{L} = L_{ib-cl}^{1} + L_{ib-cl}^{2} + L_{ib-cl}.$$ (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.

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 $z$ 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 $1 \times 10^{-4}$ and initial learning rate of $1 \times 10^{-3}$.

**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-
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.

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.
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 datasets, 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.

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 \(\mathcal{L}_{cb}^{\text{cl}}\), 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 ScanObjectNN 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 backbone) when compared to the single-branch construction.
and 7.06% with PointNet backbones) on the ScanObjectNN dataset (obtained from actual indoor scans of real-world scenes, containing occluded objects and points that 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 $L_{cb-cl}$.

**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 ScanObjectNN 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 $L_{1}^{cl-cl}$ and $L_{2}^{cl-cl}$ have a greater impact on the feature extraction capabilities of a simpler backbone (PointNet), and $L_{cb-cl}$ 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.

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


[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. Visualiz-