3D Shape Contrastive Representation Learning With Adversarial Examples

Congcong Wen, *Member, IEEE*, Xiang Li, *Member, IEEE*, Hao Huang, *Student Member, IEEE*, Yu-Shen Liu, *Member, IEEE*, and Yi Fang, *Member, IEEE*

***Abstract*—Current supervised methods for 3D shape representation learning have achieved satisfying performance, yet require extensive human-labeled datasets. Unsupervised learning-based methods provide a viable solution by learning shape representations without using ground truth labels. In this study, we develop a contrastive learning framework for unsupervised representation learning of 3D shapes. Specifically, in order to encourage models to pay more attention to useful information during representation learning, we first introduce a new paradigm for critical points search based on the adversarial mechanism. We extract critical points with a larger impact on the global feature by attacking a pre-trained auto-encoder model, and apply data augmentations on these points to generate adversarial examples. Taking a pair of adversarial examples as inputs, we obtain their intermediate embeddings and global representations of corresponding inputs, which are then transformed into latent spaces by two predictor heads. Finally, we train the proposed model by maximizing the agreements on these latent spaces via Normalized Temperature-scaled Cross Entropy (NT-Xent) loss and a newly designed Cross-layer Normalized Temperature-scaled Cross Entropy (Cross-NT-Xent) loss, where the latter is proposed in this article to enforce cross-layer feature similarities. The effectiveness, robustness, and transferability of learned representations are validated on three downstream tasks, including object classification, few-shot classification, and shape retrieval. Experiments on three benchmark datasets show that our learned representations achieve better or competitive performance than current state-of-the-art methods in these downstream tasks. Moreover, our model can easily be extended to 3D part segmentation and scene segmentation tasks.**

***Index Terms*—3D shape representation, contrastive learning, adversarial examples, few-shot learning, shape retrieval.**

I. Introduction

3D GEOMETRIC data has been widely applied in various applications, such as robotics, augmented reality, and autonomous driving. As one of the fundamental problems in 3D computer vision, 3D shape representation learning has recently attracted increasing attention. Although numerous studies [1], [2], [3], [4], [5] using supervised learning with extensive manually-labeled datasets have achieved remarkable performance, the time and manpower required for data collection and annotation poses a significant barrier to real-world applications. Alternatively, unsupervised learning methods, which don't require ground truth labels, provide an attractive solution to this issue and become more appealing in the 3D vision community.

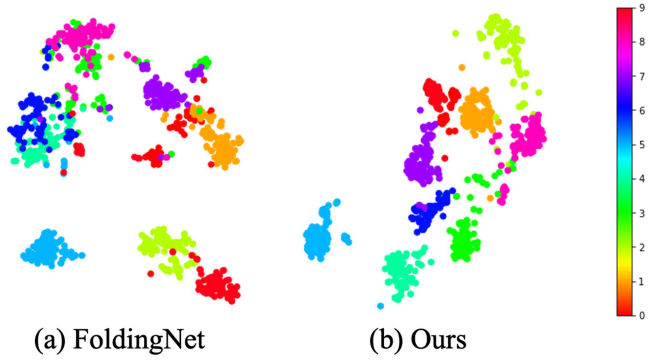

Fig. 1. Visualization of unsupervisedly learned representations by FoldingNet and ShapeContrast on the test set of ModelNet10 using t-SNE.

Several attempts have been made to learn 3D shape representations based on unsupervised learning. One important direction is to train auto-encoder networks by minimizing the reconstruction error to learn shape representations [6], [7], [8]. Specifically, they first employ an encoder network to obtain global shape representations from input 3D shapes and then utilize a decoder network to reconstruct the original inputs. Although such methods perform well on shape reconstruction, they usually have several drawbacks: first, the learned representations are variant as they involve specific information (such as pose information) of objects from input [9]; second, these methods struggle to capture effective semantic information during representation learning [10], [11]. Fig. 1(a) shows the learned representations on ModelNet10 test set by FoldingNet [6] method. It can be seen that learned global shape representations are intersected for several categories.

Another promising direction in unsupervised visual representation learning is contrastive learning [12], [13], [14], [15] which learns 3D shape representations by maximizing the agreement of different augmented views of the same input. Inspired by

Manuscript received 14 October 2022; revised 12 February 2023 and 31 March 2023; accepted 2 April 2023. Date of publication 6 April 2023; date of current version 21 February 2025. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Yang Liu. *(Corresponding author: Yi Fang.)*

Congcong Wen, Xiang Li, Hao Huang, and Yi Fang are with the NYUAD Center for Artificial Intelligence and Robotics, New York University, Abu Dhabi, UAE, also with the New York University, Abu Dhabi, UAE, and also with the NYU Multimedia and Visual Computing Lab, New York University, NY 11201 USA (e-mail: wencc@nyu.edu; xl1845@nyu.edu; hh1811@nyu.edu; yfang@nyu.edu).

Yu-Shen Liu is with the School of Software, Tsinghua University, Beijing 100190, China (e-mail: liuyushen@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TMM.2023.3265177

this, some researchers have applied contrastive learning models to 3D vision tasks [9], [16], [17]. However, their learned representations are limited in effectiveness mainly for two reasons: they pay equal attention to all points in the input point cloud; and they only focus on the similarities between global representations of input shapes. To remedy the first issue, we propose a new paradigm that utilizes an adversarial mechanism to identify "critical points" in the input point cloud of 3D shapes. It's worth noting that "critical points" refer to important points in point clouds with the maximum information [18], [19]. As we know, a trained neural network model can be attacked by adding small perturbations to certain points of an input point cloud, i.e., these points have greater impacts on global features. Accordingly, in this article, we refer to these critical points that contribute more to the vulnerability of neural networks as *adversarial points*. To encourage the proposed model to pay more attention to adversarial points, we apply data augmentations only on the adversarial points to generate *adversarial examples*. To solve the second issue, we additionally maximize the agreements between the local embeddings of one adversarial example and the global representation of the other adversarial example by introducing a Cross-layer Normalized Temperature-scaled Cross Entropy (Cross-NT-Xent) loss function.

To conclude, we propose ShapeContrast, a universal contrastive learning neural network for 3D shape representation learning. Specifically, we identify critical points using the adversarial mechanism that attacks a pre-trained auto-encoder model. Next, we apply data augmentations solely to these adversarial points in order to enable the proposed model to concentrate on the points that have a greater impact on representation learning. The augmented views of adversarial points are then merged with other non-adversarial points to obtain adversarial examples. A pair of adversarial examples are input into a weight-sharing encoder to extract the local embeddings and global representations, which are further transformed into latent spaces by using two lightweight predictor heads. Finally, we maximize the agreements between two adversarial examples in these latent spaces via the NT-Xent loss and the proposed Cross-NT-Xent loss. Our principal contributions are summarized as follows.

- We achieve effective 3D shape representation learning by designing a new paradigm for critical point identification based on the adversarial mechanism, which enables the proposed model to concentrate more on critical points.
- We introduce a Cross-NT-Xent loss function to maximize similarities between local embedding and global representation of different branches to improve the quality of learned representation.
- Extensive experiments on three benchmark datasets show new state-of-the-art or competitive performance in three downstream tasks, including shape classification, few-shot classification, and shape retrieval.
- Our model exhibits promising extensibility and generalizability, and can be easily extended to part segmentation and scene segmentation tasks.

II. Related Work

A. Point Cloud Representation Learning

Supervised Representation Learning: Early work on point cloud representation learning mainly focused on supervised learning methods. PointNet [1], as one of the pioneering deep learning models directly applied to raw point clouds, extracts the features of each point by multilayer perceptron (MLP) layers and then utilizes a symmetric function to obtain a global feature representation. To remedy PointNet's limitation of being unable to consider the local geometry structures, the follow-up work PointNet++ [2] is proposed to iteratively apply PointNet to extract features in local regions of the point cloud generated by sampling and grouping operations. In addition, some researchers design convolutional kernels for 3D point clouds to learn feature representations [3], [4], [5], [20], [21]. For instance, PointConv [4] introduces a convolutional kernel by combining a weighting function and a density function. Moreover, a number of graph networks [22], [23], [24], [25], [26] are proposed to perform representation learning by regarding a point cloud as a graph. DGCNN [25], a representative graph-based method, employs edge convolution to learn features from the graph constructed dynamically in the feature space.

Unsupervised Representation Learning: Recently, unsupervised learning has attracted increasing attention as an alternative method for feature extraction. Compared to supervised learning methods, unsupervised learning methods don't require the time-consuming annotation of point cloud, which provides more possibilities for practical applications. One of the main directions of existing studies for unsupervised representation learning is based on the auto-encoder framework [6], [7], [8], [27], [28], [29], [30], in which an encoder network is applied to extract global features and a decoder network is utilized to reconstruct the point cloud from learned representation. However, the learned representation involves specific information of input shape, like pose information, which is inconsistent with the goal of learning invariant representations. In addition, the learned representation may perform well on low-level tasks like completion and reconstruction, but it has limitations on high-level tasks that require semantic information [10], [11]. To address this problem, PointGLR [10] implements a bidirectional reasoning network by connecting the local structures at different levels and the global shape.

Contrastive learning, as another promising direction in unsupervised representation learning, has recently played a dominant role in the image and natural language processing (NLP) domains [12], [13], [14], [15]. Different from the auto-encoder framework, which require an additional decoder network to reconstruct the input shape, contrastive learning methods only consider the similarity of learned representation among samples, enabling the network concentrate on extracting powerful representations. Recently, a few studies [9], [16], [17] have attempted to incorporate contrastive learning to point cloud tasks. For example, ContrastNet [16] first generates some parts of a 3D object shape and then conducts partial contrastive learning to obtain features. Based on the learned feature, the authors

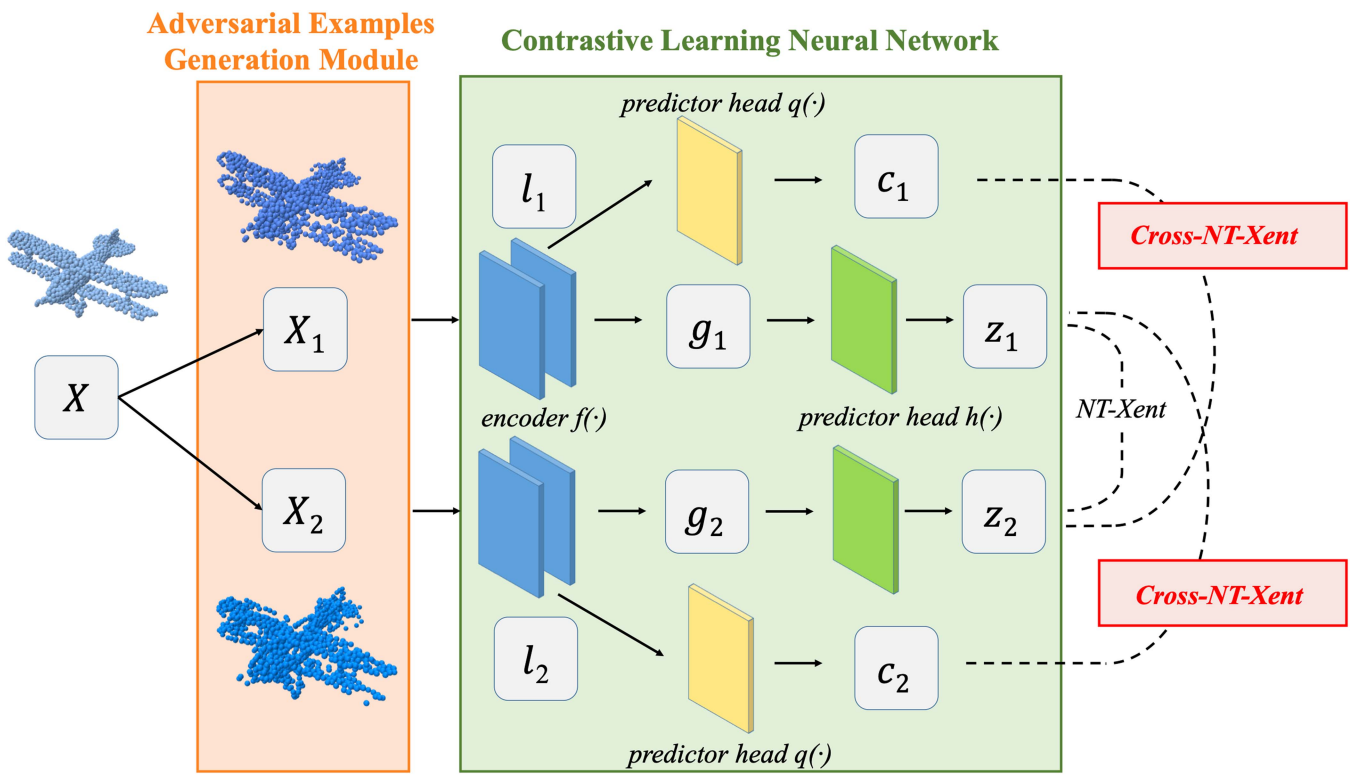

Fig. 2. Overview of the proposed ShapeContrast. We first generate $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ based on the Adversarial Examples Generation Module (see Fig. 3). Then, we input $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ to a weight-sharing feature learning encoder $f(\cdot)$ to obtain global representation $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$, respectively. Besides, we preserve the local features $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ from various intermediate layers of the encoder $f(\cdot)$. Next, we utilize predictor head $h(\cdot)$ to transform $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ to latent vector $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$, and predictor head $q(\cdot)$ to transform $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ to latent vector $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$, respectively. Finally, we adopt NT-Xent and the proposed Cross-NT-Xent functions to optimize the parameters of encoder $f(\cdot)$, predictor head $h(\cdot)$, and predictor head $q(\cdot)$.

design the other two networks for clustering and classification. Besides, Info3D [9] takes different views of a 3D object shape as input and maximizes the mutual information of the features learned by the same encoder. However, these methods fail to consider the contribution difference of each point to the representation learning. Meanwhile, they only take the similarity between global features of different branches into account to optimize the network parameters.

B. Adversarial Attack and Examples on Point Cloud

Extensive studies [31], [32], [33], [34], [35] have shown that deep neural networks of 2D images recognition are vulnerable to adversarial examples, which are generated by adding imperceptible perturbations that cause the models to make wrong predictions. Such vulnerability of deep learning models has also raised great concerns in the 3D vision [36], [37], [38], [39], [40], [41]. Xiang et al. [36] investigate adversarial attacks on point cloud classification task by proposing two types of attacks: adversarial point perturbation and adversarial point generation. Lee [40] inject adversarial noises into the learned latent space of point clouds and obtain adversarial examples by inputting the noised features to a decoder. All these works demonstrate that the global features are very vulnerable to the effect of adding perturbations to some specific points. Inspired by this, we regard these specific points as adversarial points, which are more critical than other points in a point cloud during representation learning. By adopting the adversarial mechanism, we extract critical points and apply data augmentations to these points, which is aimed at enabling model to pay more attention to points with larger contributions to representation learning.

III. Methods

This section will provide a detailed description of the proposed ShapeContrast. In Section III-A, we introduce the overall architecture that is also illustrated in Fig. 2. And then, in Section III-B, we elaborate the *Adversarial Examples Generation Module* that is used to generate a pair of adversarial examples when given a point cloud example. Next, in Section III-C, we delve to the *Contrastive Learning Neural Network*, including its components and the process of obtaining local intermediate embeddings and global representations. Finally, in Section III-D, we present the *Contrastive Loss Functions* that are used to optimize the parameters of the proposed model.

A. Overall Architecture

The overall architecture is illustrated in Fig. 2. Given a 3D point set $\boldsymbol{X} = \{x_1, x_2, \ldots, x_N\}$, where each point x_i is represented by a 3D coordinate, and N is the number of points. We first generate two adversarial examples, $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, based on

the proposed Adversarial Examples Generation Module (Section III-B). And the generated $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ are input into the *Contrastive Learning Neural Network* (Section III-C) to first produce the local intermediate embeddings $\boldsymbol{l}$ and the global representation $\boldsymbol{g}$, and then output the latent vectors $\boldsymbol{c}$ and $\boldsymbol{z}$, which are respectively predicted from $\boldsymbol{l}$ and $\boldsymbol{g}$. Specifically, we input $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ into a weight-sharing *encoder* $f(\cdot)$ to obtain global representation $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$, respectively. Besides, we also preserve the local intermediate embeddings $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ produced from f_I, which represents the first few layers of *encoder* $f(\cdot)$. Next, we introduce *predictor head* $h(\cdot)$ to transform global features $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ to latent vectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$, and a *predictor head* $q(\cdot)$ to map local features $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ to latent vectors $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$. The processing of the entire network can be formulated as:

$$\begin{aligned} \boldsymbol{z_1} &\triangleq h(f(\boldsymbol{X_1})) \quad \boldsymbol{z_2} \triangleq h(f(\boldsymbol{X_2})) \\ \boldsymbol{c_1} &\triangleq q(f_I(\boldsymbol{X_1})) \quad \boldsymbol{c_2} \triangleq q(f_I(\boldsymbol{X_2})). \end{aligned} \tag{1}$$

After obtaining these latent vectors, we define two types of contrastive loss functions: Normalized Temperature-scaled Cross Entropy (NT-Xent) and Cross-layer Normalized Temperature-scaled Cross Entropy (Cross-NT-Xent). The first loss function is aimed to maximize agreement on latent vectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$, since two adversarial examples of the same object should be similar in terms of global representation. And the second loss function is aimed to maximize agreement on latent vectors $\boldsymbol{z_1}$ and $\boldsymbol{c_2}$, and $\boldsymbol{z_2}$ and $\boldsymbol{c_1}$, as the local structures and global shape of an object should be closely correlated, which also applies to different augmented views of the same object. That is, the local structures of one augmented view should also be related to the global shape of the other augmented view. Therefore, the final objective function is defined as:

$$\mathcal{L} = \mathcal{L}_1(\boldsymbol{z_1}, \boldsymbol{z_2}) + \mathcal{L}_2(\boldsymbol{c_2}, \boldsymbol{z_1}) + \mathcal{L}_2(\boldsymbol{c_1}, \boldsymbol{z_2}), \tag{2}$$

where $\mathcal{L}_1$ denotes the NT-Xent loss function, which measures the agreement between the latent spaces of global features to ensure that the global representations of the two augmented examples are similar. And $\mathcal{L}_2$ denotes the Cross-NT-Xent loss function, which measures the agreement between the latent spaces of local features and global features to make the local structure of one augmented example correlated with the global representations of the other augmented example. The algorithm is summarized in Algorithm 2.

B. Adversarial Examples Generation

Existing contrastive learning methods [9], [16], [17] typically conduct augmentations on all points of a point cloud, which do not account for differences in contributions between points to representation learning. However, as previously stated, our model is expected to focus on the critical points, refered to adversarial points in this article, which have a greater contribution to representation learning. Considering this, we only conduct augmentations on these adversarial points of the point cloud in this article.

As shown in Fig. 3, in order to identify adversarial points, we first conduct the adversarial attack on a pre-trained point cloud auto-encoder model to obtain adversarial point cloud $\boldsymbol{X'}$. Note

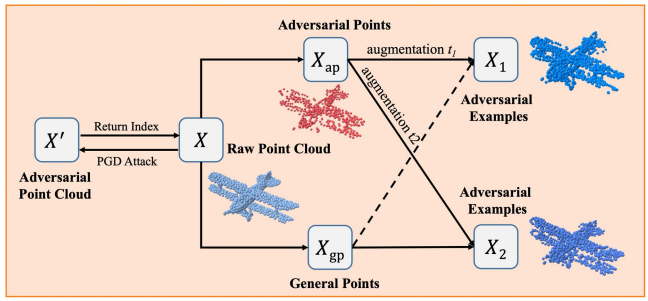

Fig. 3. Illustration of adversarial examples generation. First, we generate adversarial point cloud $\boldsymbol{X'}$ by applying the adversarial attack on raw point cloud $\boldsymbol{X}$. Then, we calculate the Euclidean distance between each point of $\boldsymbol{X}$ and $\boldsymbol{X'}$. Next, we return the indexes of the top K points with the largest distances, and extract the adversarial points $\boldsymbol{X_{ap}}$ according to these indexes, while the rest of the points are used as general points $\boldsymbol{X_{gp}}$. Finally, we obtain adversarial examples $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$. by adding two standard data augmentations t1 and t2 on $\boldsymbol{X_{ap}}$ and merging the augmented views of $\boldsymbol{X_{ap}}$ with $\boldsymbol{X_{gp}}$.

that our goal is to achieve unsupervised representation learning, so pre-trained supervised models are not considered. In this article, we adopt projected gradient descent (PGD) [42] as the attack method for adversarial points identification. The PGD attack can be regarded as an inner maximization problem, which is aimed at finding a small perturbation to maximize the reconstruction loss of target auto-encoder model, formulated as:

$$\max L_{AE}(\theta, M(\boldsymbol{X'}), \boldsymbol{X}), \tag{3}$$

where L_{AE} is reconstruction loss function, θ is parameters of auto-encoder model M, $\boldsymbol{X}$ is input point cloud, $\boldsymbol{X'}$ is generated adversarial point cloud. To achieve this, referring to [42], we iteratively add the sign of the gradient of the loss function with respect to the input, while projecting the perturbation back to ensure the adversarial examples are valid on the geometry. This iterative process can be expressed as:

$$\begin{aligned} \boldsymbol{X^0} &= \boldsymbol{X} \\ \boldsymbol{X^{t+1}} &= \mathscr{P}\left(\boldsymbol{X^t} + \alpha \operatorname{sign}\left(\nabla_{\boldsymbol{X^t}} L_{AE}(M(\boldsymbol{X^t}), \boldsymbol{X^t})\right)\right), \\ & \qquad t \in [0, T-1] \\ \boldsymbol{X'} &= \boldsymbol{X^T} \end{aligned} \tag{4}$$

where X^t refers to the adversarial point cloud at step t, α denotes the learning rate, and $\mathscr{P}$ is the projection onto the ball of interest, and this work uses clipping which is the case of ℓ_∞ norm to ensure the adversarial examples are valid on the geometry. Hence, the perturbation of each point can be calculated by the Euclidean distance between $\boldsymbol{X}$ and $\boldsymbol{X'}$. Then the first K points with the largest perturbation are chosen as adversarial points $\boldsymbol{X_{ap}}$,

$$\boldsymbol{X_{ap}} = \{\boldsymbol{X_k} \mid k \in \operatorname{argmax}(\|\boldsymbol{X'} - \boldsymbol{X}\|_{1:K})\}, \tag{5}$$

and the remaining points are regarded as general points $\boldsymbol{X_{gp}}$. Fig. 4 shows randomly selected identified adversarial points (colored by red) of point clouds on the ModelNet40 test set. Next, we perform two different data augmentations t_1 and t_2 on $\boldsymbol{X_{ap}}$, including random scaling and random shift, following [1], [11]. Finally, the adversarial examples $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ of the input

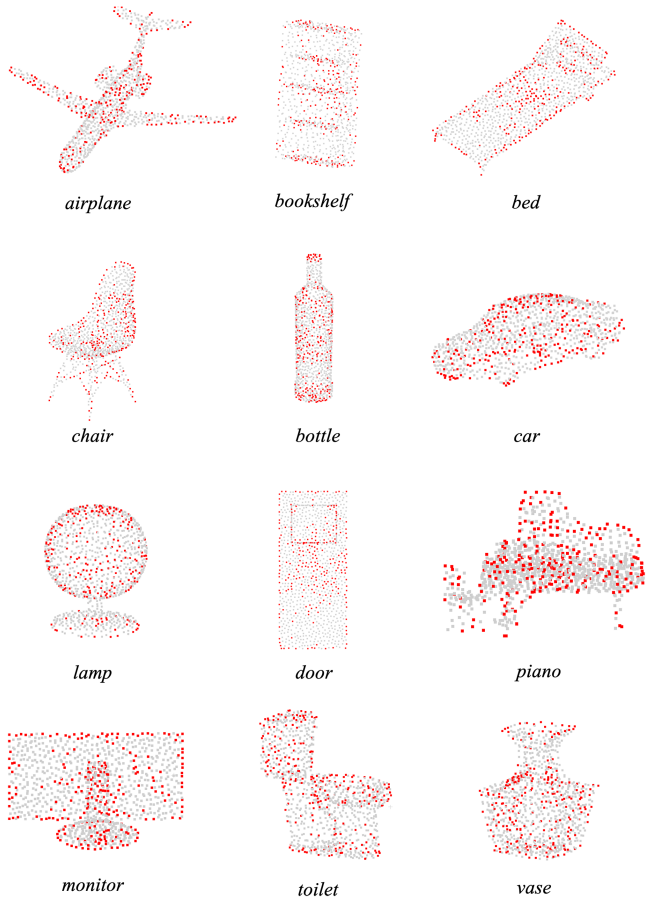

Fig. 4. Visualization of the adversarial points identified on the point cloud examples of the Modelnet40 dataset.

point cloud $\boldsymbol{X}$ can be obtained by:

$$\begin{aligned} \boldsymbol{X_1} &= [t_1(\boldsymbol{X_{ap}}), \boldsymbol{X_{gp}}] \\ \boldsymbol{X_2} &= [t_2(\boldsymbol{X_{ap}}), \boldsymbol{X_{gp}}], \end{aligned} \tag{6}$$

where t_1 and t_2 are two augmentations applied to point cloud $\boldsymbol{X}$, $[\cdot,\cdot]$ is concatenation operation. The algorithm is illustrated in Fig. 3 and summarized in Algorithm 1.

C. Contrastive Learning Neural Network

Our contrastive learning neural network consists of three sub-networks: an encoder $f(\cdot)$, a predictor head $h(\cdot)$ and a predictor head $q(\cdot)$. Two adversarial examples, $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, are input to the encoder $f(\cdot)$ to get the corresponding representation $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$. Various choices of point cloud deep neural networks can be used as the encoder. In this article, we adopt a simple and light-weight PointNet++ model [2], [10] as our encoder, which has only two Single-Scale Grouping (SSG) layers for feature aggregation. For each layer, we further reduce the original three MLP layers to two MLP layers. The outputs of two SSG layers, denoted by f_I, from two branches are preserved as local features $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$, respectively. Then, similar to the previous contrastive learning work [12], [13], [15], a simple neural network *predictor head* $h(\cdot)$ is utilized to transform the global representation $\boldsymbol{g_1}$

Algorithm 1: Adversarial Examples Generation

Input: Point Cloud $\boldsymbol{X}$, pre-trained AE model M with parameter θ, loss function L_{AE}, number of iterations T, a small constant ε that restricts the perturbation, the number of adversarial points N

Output: Adversarial Example $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$

Let $\boldsymbol{X^t} = \boldsymbol{X^0}$;
for t = 1:T **do**
  Calculate loss L_{AE} ;
  Calculate gradient $\nabla_{X^t} L_{AE}$;
  Calculate perturbation $\alpha \operatorname{sign} \nabla_{X^t} L_{AE}$;
  Clip perturbation within the range [-ε, ε] ;
  Update $\boldsymbol{X^t}$;
end
$\boldsymbol{X'} = \boldsymbol{X^T}$;
Calculate the distance between $\boldsymbol{X}$ and $\boldsymbol{X'}$;
Select N points with the largest distance as $\boldsymbol{X_{ap}}$;
Take the other points as $\boldsymbol{X_{gp}}$;
Apply data augmentations on $\boldsymbol{X_{ap}}$;
Generate $\boldsymbol{X_1}$, $\boldsymbol{X_2}$ according to Eq. 6

and $\boldsymbol{g_2}$ to latent vectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$. Moreover, another simple neural network *predictor head* $q(\cdot)$ is designed to transform the local features $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ to latent vectors $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$. We implement these two predictor heads by a two-layer MLP with a ReLU function, and (1) can be rewritten as:

$$\begin{aligned} \boldsymbol{z_1} &\triangleq W_h^{(2)} * (\sigma(W_h^{(1)} * f(\boldsymbol{X_1}) + b_h^{(1)})) + b_h^{(2)} \\ \boldsymbol{z_2} &\triangleq W_h^{(2)} * (\sigma(W_h^{(1)} * f(\boldsymbol{X_2}) + b_h^{(1)})) + b_h^{(2)} \\ \boldsymbol{c_1} &\triangleq W_q^{(2)} * (\sigma(W_q^{(1)} * f_I(\boldsymbol{X_1}) + b_q^{(1)})) + b_q^{(2)} \\ \boldsymbol{c_2} &\triangleq W_q^{(2)} * (\sigma(W_q^{(1)} * f_I(\boldsymbol{X_2}) + b_q^{(1)})) + b_q^{(2)}, \end{aligned} \tag{7}$$

where σ is a ReLU activation function, $*$ denotes matrix multiplication, and W and b represent the weight parameter and bias parameter of the MLP layer.

D. Contrastive Loss Functions

Considering the different adversarial examples of the same object should be similar in terms of global representation, we adopt the normalized temperature-scaled cross entropy loss (NT-Xent) loss [12] to measure the similarity between transformed latent spaces $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ of global representations $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ from two branches. Taking a mini-batch of N examples as input, the two-branch architecture network produces a total of 2 N examples. For each positive pair in a mini-batch, we treat the other 2N-2 examples as negative examples. Hence, this loss function for a positive pair $(\boldsymbol{z}_i, \boldsymbol{z}_j)$ in a mini-batch can be formulated as:

$$\begin{aligned} \mathcal{L}_1(\boldsymbol{z}_i, \boldsymbol{z}_j) &= -\log \frac{\exp(sim(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(sim(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} \\ sim(\boldsymbol{z}_i, \boldsymbol{z}_j) &= \frac{\boldsymbol{z}_i^T \boldsymbol{z}_j}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|}, \end{aligned} \tag{8}$$

where $sim(\cdot)$ denotes cosine similarity, $\|\cdot\|$ indicates the l2 norm, τ is a temperature parameter and $\mathbb{1}_{[k\neq i]}$ is an indicator function that evaluates to 1 if $k \neq i$.

Additionally, we have taken into account another fact that the local structures and global representation of an object should also be correlated, not only within the object itself but also among different adversarial examples of the same object. In other words, the local structures of one adversarial example of the same object should be correlated with the global representation of another adversarial sample as well. To this end, we propose a novel Cross-layer Normalized Temperature-scaled Cross Entropy loss function, named Cross-NT-Xent. Inspired by [12], we apply our loss function to the latent space projected from the feature space, rather than directly on the obtained feature space. To be specific, we design a predictor head $q(\cdot)$ to map local features $\boldsymbol{l}$ generated by intermediate layers of the encoder to latent spaces $\boldsymbol{c}$. After that, we apply the introduced Cross-NT-Xent to measure the similarity between $\boldsymbol{c}$ from one branch and $\boldsymbol{z}$ from the other branch. For instance, the input of model is a mini-batch of N examples, and the output from two branches of the model for an example is $\boldsymbol{z_i}$, $\boldsymbol{z_j}$, $\boldsymbol{c_i}$ and $\boldsymbol{c_j}$, the Cross-NT-Xent can be expressed as:

$$\mathcal{L}_2(\boldsymbol{c_j}, \boldsymbol{z_i}) = -\sum_{\ell} \log \frac{\exp({\boldsymbol{c_j^\ell}}^T \boldsymbol{z_i}/\tau)}{\sum_{k=1}^{N} \exp({\boldsymbol{c_k^\ell}}^T \boldsymbol{z_i}/\tau)}$$

$$\mathcal{L}_2(\boldsymbol{c_i}, \boldsymbol{z_j}) = -\sum_{\ell} \log \frac{\exp({\boldsymbol{c_i^\ell}}^T \boldsymbol{z_j}/\tau)}{\sum_{k=1}^{N} \exp({\boldsymbol{c_k^\ell}}^T \boldsymbol{z_j}/\tau)}, \tag{9}$$

where ℓ indicates ℓ-th intermediate layer of the encoder. In our article, the encoder involves two layers, so $\ell \in [1, 2]$. This loss function enforces local embedding to be closer to the global representation of the same object than any other object, improving the effectiveness of learned representations, as demonstrated by the following experiments.

IV. Experiments

A. Experimental Datasets

Three benchmark datasets, including ModelNet [43], ScanObjectNN [44] and ScanNet [45], are selected for evaluating the performance of representation learned by our proposed model. The ModelNet40/10 dataset consists of 9832/3991 training shapes and 2468/908 testing shapes from 40/10 object classes. ScanObjectNN and ScanNet are both real-world datasets that are used for cross-data evaluation. The former comprises 2902 3D objects from 15 categories and we employ the "object-only" partition in our experiments. And the latter contains 9305/2606 training/testing object instances including 17 categories collected from the original ScanNet in accordance with [3]. In all experiments, 1024 points are randomly sampled from each 3D object for model training and evaluation. Note that no other information, such as normal information, is involved in our network. Moreover, all experimental results are reported using single-view instead of multi-view voting strategy, which is employed in some work to better demonstrate the performance of the proposed model.

Algorithm 2: ShapeContrast's Main Algorithm

Input: Point Cloud X, Batch size N, Epoch E
Output: Network parameters of encoder $f(\cdot)$, predictor head $h(\cdot)$ and predictor head $q(\cdot)$

for e = 1:E **do**
  Sampled a mini-batch $\{\boldsymbol{x_k}\}_{k=1}^{N}$;
  Generate adversarial examples $\{\boldsymbol{x_{k\_1}}\}_{k=1}^{N}$ and $\{\boldsymbol{x_{k\_2}}\}_{k=1}^{N}$ according to Algorithm 1;
  Obtain local features $\left\{\{\boldsymbol{l_{\ell\_k\_1}}\}_{k=1}^{N}\right\}_{\ell=1}^{2}$ and $\left\{\{\boldsymbol{l_{\ell\_k\_2}}\}_{k=1}^{N}\right\}_{\ell=1}^{2}$;
  Obtain global representation $\{\boldsymbol{g_{k\_1}}\}_{k=1}^{N}$ and $\{\boldsymbol{g_{k\_2}}\}_{k=1}^{N}$;
  Calculate transformed latent space $\{\boldsymbol{z_{k\_1}}\}_{k=1}^{N}$ and $\{\boldsymbol{z_{k\_2}}\}_{k=1}^{N}$, $\left\{\{\boldsymbol{c_{\ell\_k\_1}}\}_{k=1}^{N}\right\}_{\ell=1}^{2}$ and $\left\{\{\boldsymbol{c_{\ell\_k\_2}}\}_{k=1}^{N}\right\}_{\ell=1}^{2}$ according to Eq. 7 ;
  Calculate $\mathcal{L}_1$ and $\mathcal{L}_2$ according to Eq. 8 and Eq. 9;
  Optimize the network by minimizing $\mathcal{L}_1$ and $\mathcal{L}_2$
end

B. Implementation Details

Adversarial attack. We adopt projected gradient descent (PGD) [42] to attack a pre-trained FoldingNet [6] model. The step size for each attack iteration is 0.002, and the number of iterations is 20. We restrict the final perturbation to 0.01, and the coordinates of adversarial examples to the range of -1 to 1.

Network architecture. In our experiments, a light-weight PointNet++ model is used as the basic encoder of our model for feature learning, which involves two single-scale grouping set abstraction layers. We sample 512 and 128 points in each layer with a ball radius of 0.23 and 0.32, respectively. For each predictor head, a two-layer MLP only with a ReLU function applied to the hidden layer is employed to map the representation to a 512-dimensional latent vector. To evaluate the effectiveness of learned representation on down-streaming classification task, the most commonly used metric in previous work is the linear separability of the classification task. In line with this, we train a linear SVM [46] on the representation obtained from the training set and then measure the classification accuracy on the testing set. We implement the linear SVM classifier on scikit-learn library.

Network optimization. We employ the ADAM optimizer at a base learning rate of 0.001, which decays with a rate of 0.7 every 20 epochs. And Batch Normalization layers [47] have a momentum of 0.9 decaying with 0.5 every 20 epochs. Our model is trained for 300 epochs with a batch size of 48 on a single Tesla V100 GPU.

C. Object Classification

Results on ModelNet40 and ModelNet10 datasets. To demonstrate the effectiveness of learned representation by our proposed method, we first compare our models with state-of-the-art point

TABLE I
CLASSIFICATION ACCURACY (%) ON MODELNET40 (MN40.) AND MODELNET10 (MN10.) DATASETS

Method	#Points	Supervised	Accuracy	
			MN40	MN10
TL Network [53]	-	✗	74.4	-
3DGAN [54]	-	✗	83.3	91.0
VSL [55]	-	✗	84.5	91.0
VIPGAN [56]	-	✗	92.0	94.1
PointNet[1]	1k	✓	89.2	-
PointNet++[2]	1k	✓	90.5	-
SO-Net[57]	1k	✓	92.5	-
PointCNN[3]	1k	✓	92.5	-
DGCNN[25]	1k	✓	92.5	-
RSCNN[20]	1k	✓	92.9	-
LGAN†[28]	2k	✗	85.7	95.3
LGAN [28]	2k	✗	87.3	92.2
FoldingNet†[6]	2k	✗	88.4	94.4
FoldingNet [6]	2k	✗	84.4	91.9
PointCapsNet [8]	2k	✗	88.9	-
MAP-VAE[58]	1k	✗	90.2	94.8
GraphTER[59]	1k	✗	92.0	-
PointGLR‡[10]	1k	✗	92.2	94.8
PointGLR*‡[10]	1k	✗	93.0	95.5
Shape Self-Correction†[60]	1k	✗	92.4	95.0
CP-Net[61]	1k	✗	92.5	-
ConClu[62]	2k	✗	92.4	95.3
ShapeContrast	1k	✗	92.7	95.1
ShapeContrast*	1k	✗	**93.3**	**95.6**

† Indicates the model is trained on shapenet dataset. ‡ Indicates the model involves normal information. * Denotes the model with multiple channel width.

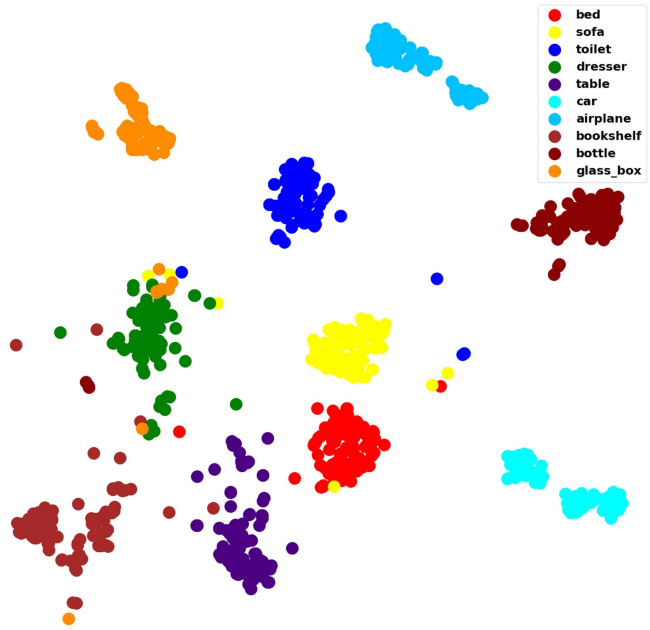

Fig. 5. Visualization of unsupervisedly learned representations by ShapeContrast on the test set of ModelNet40 using t-SNE. We randomly select 10 categories from the original 40 categories to show results more clearly.

cloud representation learning methods, including supervised learning and unsupervised learning. And for unsupervised learning methods, we also select representative voxel-based methods and multi-views-based methods, except for popular point-based methods. Recent work [10] indicates that increasing channel width can improve accuracy with lower speed costs. So we report the performance of our model using the basic model (1x channel width) and the larger model (5x channel width). Table I displays the results of our model and the comparison models on the ModelNet40 and ModelNet10 dataset. While existing supervised learning methods [48], [49], [50], [51], [52] have produced superior results, it is important to note that our approach is aimed at learning representations in an unsupervised manner and trains on point clouds consisting of only 1000 points.To make a fair comparison, we have not listed these methods in Table I. It can be found from Table I that our method achieves new state-of-the-art accuracy with 93.3% and 95.6% for ModelNet40 and ModelNet10 datasets, respectively.

Three points should be noted here: 1) our model is trained on the ModelNet40 dataset, while some models (marked as †) like LGAN, FoldingNet and Shape Self-Correction are obtained on the larger ShapeNet [43] dataset; 2) we input 1024 original 3D points without any other information, while some models (marked as ‡) like PointGLR either input the normal information or use it as a constraint for the decoder; 3) it can be found that our model achieves a comparable or superior accuracy compared with the supervised model with the same input of 1 k points. In short, our model only needs to be trained with a small number of points on a smaller dataset, but achieves state-of-the-art performance for unsupervised classification. In particular, we adopt a lighter PointNet++ as the encoder of our model, which achieves a better classification performance than the original PointNet++ [2] model trained in the supervised manner.

Moreover, we visualize the learned representations on the testing set of ModelNet40 by using t-SNE. To show the learned features more clearly, we randomly select ten categories from the original forty categories, and display their visualization results in Fig. 5. We find that the learned representations of the same category are roughly clustered together.

Transferability Validation on ScanObjectNN and ScanNet datasets. As demonstrated in the last section, our model outperforms other models in terms of learned representation when trained and tested on the same dataset. In this section, we further present the transferability of learned representation on different datasets. Experimental datasets include ScanObjectNN [44] and ScanNet [45], which contain different categories of 3D object shape from the ModelNet dataset. PointNet++ model with 5x channel width is employed as our encoder and trained on ModelNet40 dataset. The representation of 3D shape from the targeted dataset are directly extracted by the trained model **without fine-tuning**. To evaluate the quality of learned representations, we train a linear SVM classifier on unsupervised representations of target training data, and present the classification results in Table II.

As can be seen in Table II, the learned representations by unsupervised learning methods have better generalization ability compared to supervised learning methods, which can be attributed to the fact that supervised learning models heavily rely on labeled data. Moreover, by comparing our ShapeContrast model with the state-of-the-art method PointGLR [10], we

TABLE II
CLASSIFICATION ACCURACY (%) ON ON SCANOBJECTNN (SON.) AND SCANNET (SN.) DATASETS

Method	Supervised	Accuracy	
		SON.	SN.
PointNet [1]	✓	79.2	-
SpiderCNN [63]	✓	79.5	-
PointNet++[2]	✓	84.3	-
PointCNN [3]	✓	85.5	-
DGCNN [25]	✓	86.2	-
PointGLR* [10]	✗	**87.2**	89.2
#PointGLR* [10]	✗	86.2	89.2
Shape Self-Correction†[60]	✗	-	89.0
ShapeContrast*	✗	86.9	**90.4**

† Indicates the model is trained on shapenet dataset.
* Denotes the model with multiple channel width. # Denotes reproduced result.

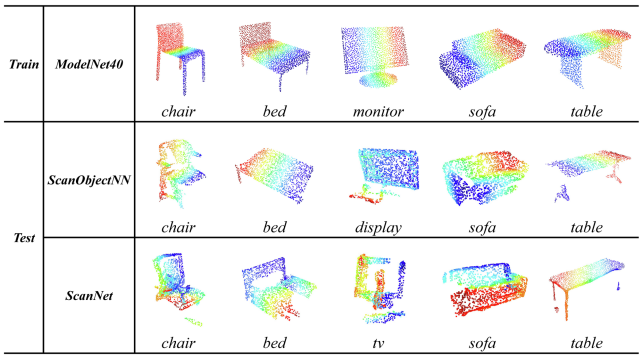

Fig. 6. Visualization of randomly selected examples from train dataset (ModelNet40) and test dataset (ScanObjectNN and ScanNet).

find that our model achieves superior performance on the ScanNet dataset. But for the ScanObjectNN dataset, our model has comparable classification accuracy to that reported in the PointGLR article and outperforms the accuracy reproduced by using the official code of PointGLR. These results indicate the satisfying transferability of learned representations to other unseen datasets.

Furthermore, we show some examples from the ModelNet40, ScanObjectNN, and ScanNet datasets in Fig. 6. As can be seen, the point clouds of synthetic training examples are clean and complete, while those of real-scanned test examples are partial and incomplete due to occlusions. However, our method still performs well on these test datasets, which demonstrates that we learned the generic representation of 3D objects from data structures instead of labels.

D. Few-Shot Learning

In addition, we test our proposed model by conducting experiments in a few-shot learning (FSL) setting. Specifically, we train our model on a support set with K classes and N samples for each class, and then evaluate the trained model on a query set with 20 unseen samples from each class. Following previous work [65], we set K to 5 and 10, and N to 10 and 20, respectively. Therefore, we conduct 10 independent experiments under four settings, i.e. "5-way 10-shot", "5-way 20-shot", "10-way 10-shot" and "10-way 20-shot". The classification results, including average accuracy and standard deviation, are reported in Table III. It can be found from Table III that our proposed model achieves new state-of-the-art accuracy in all few-shot settings, which further demonstrates the pleasing transferability of our model to new tasks, even with limited training data.

E. Shape Retrieval

We also implement 3D shape retrieval on ModelNet40 to further evaluate the performance of learned representation. Given a query 3D shape from the testing set, a ranked list of the remaining 3D shape in the testing set will be returned according to the similarity of the shapes. We adopt the L2 distance between each pair of samples as the similarity measure, which is consistent with [66]. We report the mean average precision (MAP) of retrieval results in Table IV. From Table IV, it can be seen that our model outperforms other state-of-the-art methods. To intuitively show the performance of shape retrieval, we list the retrieval results in Fig. 7 by taking some randomly selected queries as input. We can see from Fig. 7 that most of 3D shapes are correctly retrieved by the proposed model. Although there are a few 3D shapes retrieved incorrectly, it can be seen from Fig. 7 that they are highly geometrically similar to the query shape, such as stool vs. chair, vase vs. lamp and wardrobe vs. piano. Thereby, this further demonstrates that our model effectively learned geometric structure information of the object shape, even without the input of labels.

V. DISCUSSION

A. Ablation Study

In this section, we conduct detailed ablation studies to evaluate the effectiveness of our model designs, including tests on the number of adversarial points, methods for selecting points, different loss functions, and selection of hyper-parameters.

1) Number of Adversarial Points: As introduced before, some critical points contribute more to the global features of the point cloud during model learning and thus should receive more attention. To identify these points, we conduct an adversarial attack on a pre-trained FoldingNet [6] model for generating an adversarial point cloud. Then we obtain adversarial points by selecting the N points with the greatest distance between the adversarial point cloud and the original point cloud. The adversarial examples are generated by merging the augmented views of adversarial points with other non-adversarial points, and then input to the proposed model to learn the representation. In this section, to explore the effect of the number of adversarial points N, we conduct experiments by setting N to 64, 128, 256, 512, and 1024, respectively.

Table V lists the classification accuracy of our model with an altered number of adversarial points for augmentation. It can be found that an appropriate number of adversarial points is beneficial to improving the model performance to some extent. Too few adversarial points will lead to insufficient variability between the two augmented examples, while augmenting all the points (i.e. N = 1024) will introduce some irrelevant information

TABLE III
COMPARISON WITH THE STATE-OF-THE-ART METHODS FOR FEW-SHOT CLASSIFICATION ON MODELNET40 DATASET

Methods	Supervised	5-way		10-way	
		10-shot	20-shot	10-shot	20-shot
3D-GAN [54]	✗	55.8 ± 10.7	65.8 ± 9.9	40.3 ± 6.5	48.4 ± 5.6
Latent-GAN [28]	✗	41.6 ± 16.9	46.2 ± 19.7	32.9 ± 9.2	25.5 ± 9.9
PointCapsNet [8]	✗	42.3 ± 17.4	53.0 ± 18.7	38.0 ± 14.3	27.2 ± 14.9
FoldingNet [6]	✗	33.4 ± 13.1	35.8 ± 18.2	18.6 ± 6.5	15.4 ± 6.8
PointNet++ [2]	✓	38.5 ± 16.0	42.4 ± 14.2	23.1 ± 7.0	18.8 ± 5.4
PointCNN [3]	✓	65.4 ± 8.9	68.6 ± 7.0	46.6 ± 4.8	50.0 ± 7.2
PointNet [1]	✓	52.0 ± 12.2	57.8 ± 15.5	46.6 ± 13.5	35.2 ± 15.3
DGCNN [25]	✓	31.6 ± 9.0	40.8 ± 14.6	19.9 ± 6.5	16.9 ± 4.8
PointNet-Rand [64]	✗	52.0 ± 3.8	57.8 ± 4.9	46.6 ± 4.3	35.2 ± 4.8
PointNet-OcCo [64]	✗	89.7 ± 1.9	92.4 ± 1.6	83.9 ± 1.8	89.7 ± 1.5
DGCNN-Rand [64]	✗	31.6 ± 2.8	40.8 ± 4.6	19.9 ± 2.1	16.9 ± 1.5
DGCNN-OcCo [64]	✗	90.6 ± 2.8	92.5 ± 1.9	82.9 ± 1.3	86.5 ± 2.2
*DGCNN-rand [64]	✗	91.8 ± 3.7	93.4 ± 3.2	86.3 ± 6.2	90.9 ± 5.1
*DGCNN-OcCo [64]	✗	91.9 ± 3.3	93.9 ± 3.1	86.4 ± 5.4	91.3 ± 4.6
*Transformer-Rand [65]	✗	87.8 ± 5.2	93.3 ± 4.3	84.6 ± 5.5	89.4 ± 6.3
*Transformer-OcCo [65]	✗	94.0 ± 3.6	95.9 ± 2.3	89.4 ± 5.1	92.4 ± 4.6
*Point-BERT [65]	✗	94.6 ± 3.1	96.3 ± 2.7	91.0 ± 5.4	92.7 ± 5.1
Ours	✗	**95.0 ± 3.5**	**96.5 ± 2.5**	**91.2 ± 4.4**	**93.0 ± 5.4**

The average accuracy (%) and the standard deviation are reported over 10 independent experiments. '*' denotes the reproduce results under the same data for a fair comparison.

TABLE IV
COMPARISON WITH THE STATE-OF-THE-ART METHODS FOR 3D SHAPE RETRIEVAL ON MODELNET40 DATASET

Method	Data Representation	mAP
SPH [67]	Mesh	33.3
LFD [68]	Voxels	40.9
3DShapeNet [66]	Voxels	49.2
Deeppano [69]	Image	76.8
MVCNN [70]	Image	80.2
MeshNet [71]	Mesh	81.9
GIFT [72]	Image	81.9
SPNet [73]	Image	85.2
RED [74]	Volume	86.3
Panorama-ENN [75]	Image	86.3
DLAN [76]	Points	85.0
TCL [77]	Image	88.0
SequenceView [78]	Image	89.1
VNN [79]	Image	89.3
Densepoint [80]	Points	88.5
PVNet [81]	Points and Image	89.5
Ours	Points	**89.9**

TABLE V
PERFORMANCE COMPARISON WITH THE DIFFERENT NUMBER OF ADVERSARIAL POINTS SELECTION ON MODELNET40 TEST SET USING POINTNET++ MODEL WITH SINGLE-CHANNEL WIDTH

Number of Adversarial Points	Accuracy
64	92.3
128	92.4
256	**92.7**
512	92.6
1024	92.4

and distract the model. Therefore, it is reasonable and effective to augment some important parts of examples, which can also be applied to other domain tasks.

2) Methods for Selecting Points: To validate the effectiveness of adversarial point selection, we compare it with two other methods: taking all points as input and randomly selecting points as adversarial points. The results of these methods on the ModelNet40 datasets are shown in Table VI. It can be observed that adversarial point selection performs better than the other two methods, improving the accuracy by 1.3% and 1.4%, respectively.

TABLE VI
COMPARISON OF DIFFERENT METHODS FOR SELECTING POINTS

Methods for Selecting Points	Accuracy
N/A	91.4
Random	91.3
Adversarial	**92.7**

N/A stands for taking all points as input, random stands for selecting points randomly, and adversarial stands for selecting points by adversarial mechanism.

TABLE VII
PERFORMANCE COMPARISON WITH THE DIFFERENT LOSS FUNCTIONS ON MODELNET40 TEST SET USING POINTNET++ MODEL WITH SINGLE-CHANNEL WIDTH

Model	$\mathcal{L}_{NT-Xent}$	$\mathcal{L}_{Cross-NT-Xent}$	Accuracy
A	✓	✗	91.8
B	✗	✓	92.3
C	✓	✓	**92.7**

3) Effect of Different Loss Function: The NT-Xent loss function is introduced by SimCLR [12] to maximize agreement between augmented views of the same example in the latent space. Inspired by this, we first design an additional predictor head to transform the intermediate feature embeddings to another latent space, and then design a Cross-NT-Xent loss function based on the original NT-Xent to maximize agreement on the cross-layer latent spaces. Related details can be found in Section III-D. To evaluate the effectiveness of different loss functions, we implement an ablation experiment to compare three models: Model A trained with NT-Xent loss, Model B trained with Cross-NT-Xent loss, and Model C trained with both NT-Xent and Cross-NT-Xent losses.

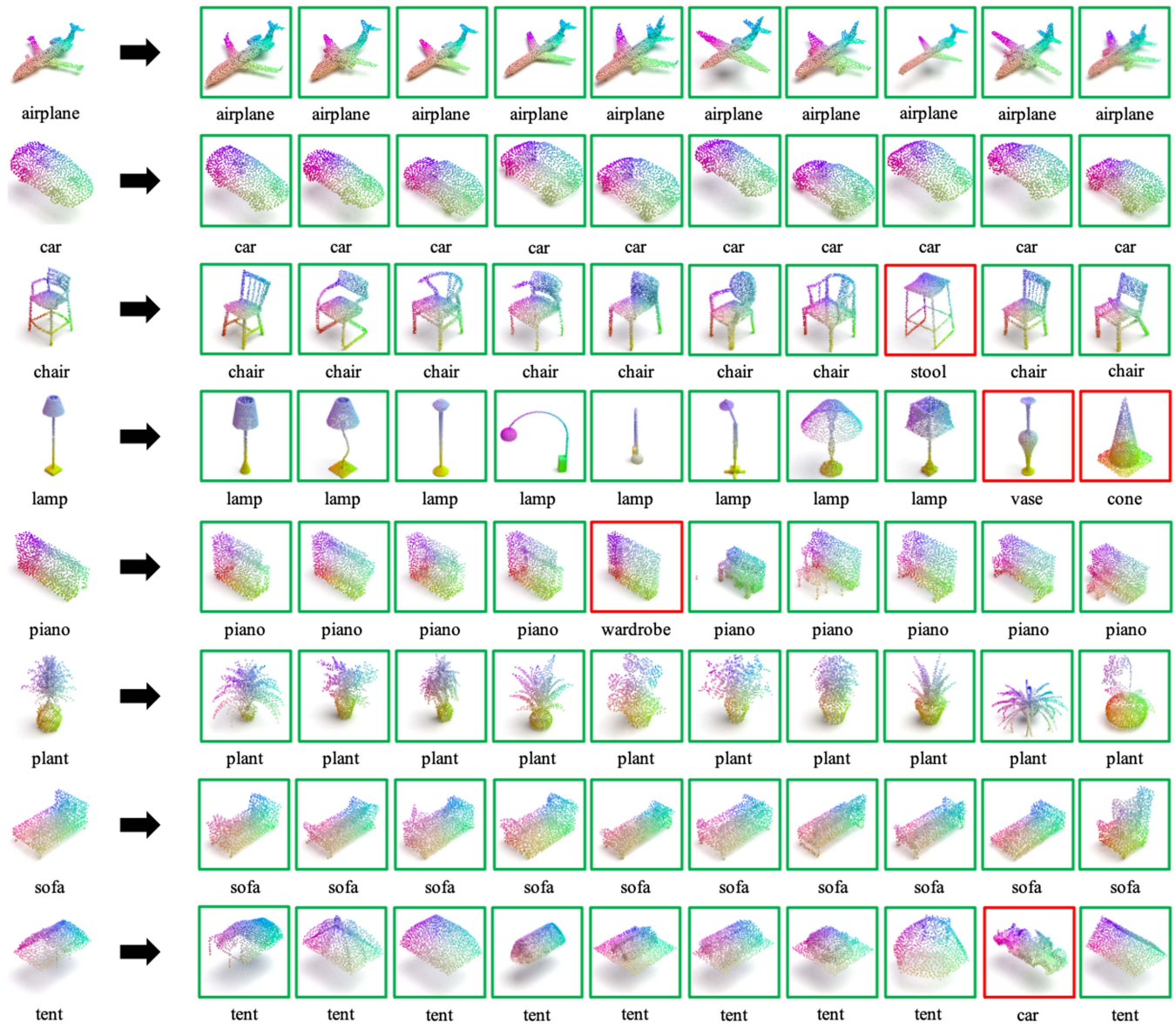

Fig. 7. Examples of 3D shape retrieval on ModelNet40 dataset. The first column denotes the query model and the rest columns are the corresponding top-10 retrieval results. The correct retrieval models are boxed in green, while the incorrect ones are boxed in red.

TABLE VIII
THE EFFECT OF OUR MODEL ACCURACY AT DIFFERENT LEARNING RATES

Learning rate	Accuracy
0.1	91.2
0.01	92.3
0.001	**92.7**
0.0001	92.2

TABLE IX
THE EFFECT OF OUR MODEL ACCURACY AT DIFFERENT BATCH SIZE

Batch size	Accuracy
24	92.0
36	92.3
48	**92.7**
60	92.5

As shown in Table VII, the classification accuracy of Model A and Model B are both inferior to that of Model C. By comparing Model A and Model C, and Model B and Model C, we can find that involving either of these two losses can help improve model accuracy, while our proposed Cross-NT-Xent loss function contributes more than the original NT-Xent loss function. By combining these two loss functions, our model enables the differently adversarial examples to be more similar to each other in the latent space, thus obtaining a more effective global representation.

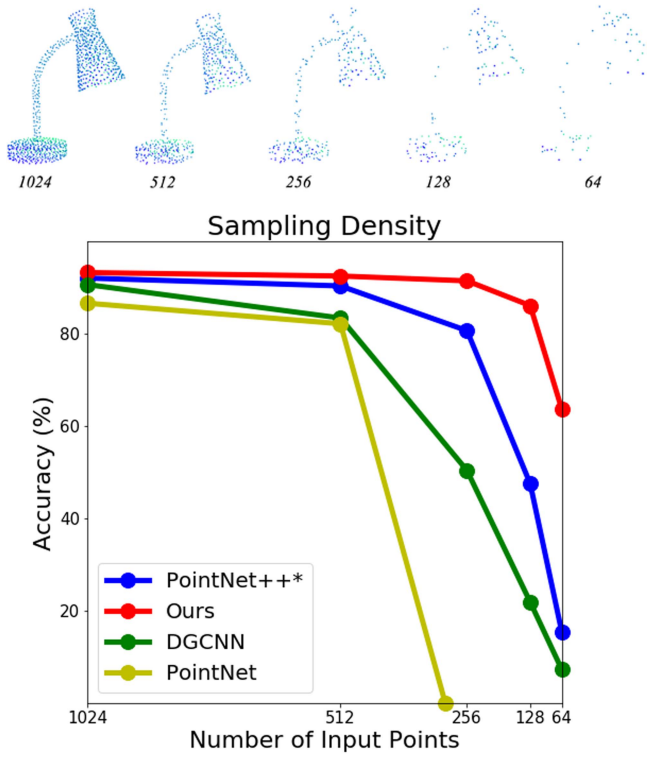

Fig. 8. Top part: Point cloud with random point dropout. Bottom part: The robustness test of our model on different number of input points compared to some supervised models on ModelNet40 test set. *Denotes the model with multiple channel width.

4) Selection of Hyper-Parameters: During the training process of our model, we have tested different hyper-parameter settings. In this section, we mainly discuss the settings of two major hyper-parameters on the ModelNet40 dataset: learning rate and batch size. We evaluate the learning rate of our model at 0.1, 0.01, 0.001, and 0.0001, and found that the highest classification accuracy is obtained with a learning rate of 0.001, as demonstrated in Table VIII. Additionally, we explore the effect of batch size on the model's performance, with values of 24, 36, 48 and 60. The results, presented in Table IX, indicate that our model exhibits the highest classification accuracy when the batch size is set to 48.

B. Robustness Analysis

To test the robustness of learned representation, we evaluate the classification accuracy of our model with varying numbers of input points and training samples.

1) Number of Input Points: We first test the robustness of our model on the ModelNet40 dataset with sparser points of 1024, 512, 256, 128, and 64. Note that our model was trained with 1024 points without random input dropout. In this experiment, we utilize the PointNet++ model with 5x channel width as the encoder of our model. And we choose PointNet [1], PointNet++ [2] with 5x channel width, and DGCNN [25] as baseline methods.

As shown in Fig. 8, recognizing shapes becomes more difficult as the number of points decreases. However, our model is more robust in learning representation of point cloud than other models. The accuracy of the compared methods decreases rapidly as the number of the point cloud gradually decreases. In particular, when the number of input points is set to 64, the accuracy of the other compared models on ModelNet40 is lower than 20%, while the accuracy of our proposed model is higher than 60%.

TABLE X
THE ROBUSTNESS TEST OF OUR MODEL ON DIFFERENT NUMBER OF TRAINING SAMPLES COMPARED TO SUPERVISED POINTNET++ MODEL (WITH 5X CHANNEL WIDTH) ON MODELNET40 TEST SET

Training Samples	PointNet++	Ours	$\lvert\Delta\rvert$
100%	92.1	93.3	1.2
50%	89.4	91.5	2.1
25%	87.4	91.2	3.8
10%	83.6	91.0	7.4
1%	74.5	89.6	15.1

$\lvert\Delta\rvert$ represents the absolute value of the accuracy difference.

2) Number of Training Samples: Moreover, we evaluate the robustness of learned representations of our model when trained with the different number of training samples. The randomly sampled 100%, 50%, 25%, 10% and 1% data from ModelNet40 training sets are individually input into training models, while the linear SVM classifier is still trained on the whole learned representations. As described in the last section, the PointNet++ model with 5x channel width is selected as one of the baseline models and the encoder in this work. It can be seen from Table X that the classification performance of our model is highly robust to decreases in the amount of training data. Particularly, the classification accuracy is kept at nearly 90% even when the model is trained with only 1% data, an improvement of more than 15% when compared with the same model trained in a supervised learning manner.

C. Extensibility of Segmentation Tasks

In this section, we verify the extensibility of the proposed model for segmentation tasks. To make a fair comparison, we follow the experimental setup of PointConstrast and pre-train our model on a larger dataset, ScanNet. We evaluate the segmentation accuracy of our model on two popular segmentation tasks, shape part segmentation and scene segmentation.

1) Part Segmentation: We evaluate the effectiveness of our learned representation on the ShapeNet Part [82] dataset for the 3D shape part segmentation. The dataset consists of 16,681 objects from 16 categories, each of which is labeled with 2 to 6 part labels. We compare several baseline models, including SO-Net [57], PointCapsNet [8], Multitask Unsupervised [83], and PointConstrast [11], and report the mean IoU across all categories of these models in Table XI. Table XI presents the results of these models when fine-tuning the pre-trained model using 1%, 5%, and 100% of the training data, respectively. It can be observed that our model achieves superior segmentation accuracy compared to these models. This indicates that our pre-trained model can easily be extended to the object segmentation task with a small amount of data fine-tuning.

TABLE XI
RESULTS OF PART SEGMENTATION ON THE SHAPENET PART DATASET

Methods	IoU(1% data)	IoU(5% data)	IoU(100% data)
SO-Net [57]	64.0	69.0	-
PointCapsNet [8]	67.0	70.0	-
Multitask [83]	68.2	77.7	-
PointConstrast [11]	74.0	79.9	85.1
Ours	**75.6**	**81.2**	**86.3**

TABLE XII
RESULTS OF SCENE SEGMENTATION ON THE S3DIS DATASET AREA5

Methods	mIoU	mAcc
PointNet [1]	41.1	49.0
PointCNN [3]	57.3	63.9
MinkowskiNet32 [85]	65.4	71.7
PointConstrast [11]	70.9	77.0
Ours	**71.7**	**78.1**

2) Scene Segmentation: In addition, the effectiveness of our pre-trained model is further evaluated on the S3DIS [84] dataset for 3D scene segmentation. The S3DIS dataset contains the point cloud of 271 rooms in 6 areas, which have been manually annotated into 13 classes. We evaluate our model on Area 5, while the point cloud from the other areas is used for fine-tuning our pre-trained model. To ensure a fair comparison with PointContrast, we selected the Sparse Residual U-Net, i.e. MinkowskiNet34, as the backbone for learning representations, following the same setting used in the PointContrast. As shown in Table XII, we compare our results with other models and find that our model still achieves superior segmentation accuracy compared to the others. Especially compared with PointConstrast, our model improves 0.8 and 1.1 in both accuracy and precision metrics, respectively. This superiority in performance is a demonstration to the extensibility and generalizability of our model, which can be applied to a wide range of segmentation tasks with confidence.

VI. CONCLUSION

In this work, we present a simple and universal framework for unsupervised representation learning of 3D shapes. We first introduce an innovative paradigm for adversarial example generation that merges augmented views of critical points, identified through the adversarial attack mechanism, with non-critical points. The adversarial examples are then input into the contrastive learning neural network to extract local embeddings and global representations. To improve the quality of the learned representations, we not only use the NT-Xent loss to enforce the similarity of the global representations but also propose a novel Cross-NT-Xent loss function to enforce the similarity between the local structure and the global shape. Extensive experimental results on three downstream tasks, including classification, few-shot learning, and shape retrieval, have demonstrated the effectiveness, robustness, and transferability of learned representations by the proposed ShapeContrast model. Furthermore, we extend our model to segmentation tasks, demonstrating the extensibility and generalization of the proposed model.

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5099–5108, 2017.

[3] Y. Li et al., "PointCNN: Convolution on X-transformed points," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 820–830, 2018.

[4] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.

[5] H. Thomas et al., "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.

[6] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 206–215.

[7] H. Deng, T. Birdal, and S. Ilic, "PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 602–618.

[8] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1009–1018.

[9] A. Sanghi, "Info3D: Representation learning on 3D objects using mutual information maximization and contrastive learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 626–642.

[10] Y. Rao, J. Lu, and J. Zhou, "Global-local bidirectional reasoning for unsupervised representation learning of 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5376–5385.

[11] S. Xie et al., "Pointcontrast: Unsupervised pre-training for 3D point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 574–591.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[13] J.-B. Grill et al., "Bootstrap your own latent-a new approach to self-supervised learning," *Adv. Neural Formation Process. Syst.*, vol. 33, pp. 21 271–21 284, 2020.

[14] M. Caron et al., "Unsupervised learning of visual features by contrasting cluster assignments," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9912–9924, 2020.

[15] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15750–15758.

[16] L. Zhang and Z. Zhu, "Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks," in *Proc. IEEE Int. Conf. 3D Vis.*, 2019, pp. 395–404.

[17] B. Du, X. Gao, W. Hu, and X. Li, "Self-contrastive learning with hard negative sampling for self-supervised point cloud learning," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 3133–3142.

[18] E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, and J. Luo, "Adaptive hierarchical down-sampling for point cloud classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12956–12964.

[19] J. Tang, Y. Wang, X. Ning, and K. Lv, "Point cloud decomposition by internal and external critical points," *Comput. Graph.*, vol. 102, pp. 18–29, 2022.

[20] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8895–8904.

[21] C. Wen, L. Yang, X. Li, L. Peng, and T. Chi, "Directionally constrained fully convolutional neural network for airborne LiDAR point cloud classification," *ISPRS J. Photogrammetry Remote Sens.*, vol. 162, pp. 50–62, 2020.

[22] G. Te, W. Hu, A. Zheng, and Z. Guo, "RGCNN: Regularized graph CNN for point cloud segmentation," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 746–754.

[23] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4548–4557.

[24] Y. Zhang and M. Rabbat, "A graph-CNN for 3D point cloud classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 6279–6283.

[25] Y. Wang et al., "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
[26] C. Wen, X. Li, X. Yao, L. Peng, and T. Chi, "Airborne LiDAR point cloud classification with global-local graph attention convolution neural network," *ISPRS J. Photogrammetry Remote Sens.*, vol. 173, pp. 181–194, 2021.
[27] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3D surface generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 216–224.
[28] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 40–49.
[29] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3D point clouds via graph convolution," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.
[30] Y. Sun et al., "PointGrow: Autoregressively learned point cloud generation with self-attention," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2020, pp. 61–70.
[31] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–15.
[32] C. Xie et al., "Adversarial examples for semantic segmentation and object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1378–1387.
[33] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 4579–4589, 2018.
[34] Q. Guo et al., "Spark: Spatial-aware online incremental attack against visual tracking," in *Proc. Comput. Vis. 16th Eur. Conf.*, 2020, pp. 202–219.
[35] A. Subramanya, V. Pillai, and H. Pirsiavash, "Fooling network interpretation in image classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2020–2029.
[36] C. Xiang, C. R. Qi, and B. Li, "Generating 3D adversarial point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9136–9144.
[37] Q. Zhang et al., "Adversarial attack and defense on point sets," 2019, *arXiv:1902.10899*.
[38] H. Zhou et al., "Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 356–10 365.
[39] H. Zhou et al., "LG-GAN: Label guided adversarial network for flexible targeted attack of point cloud based deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10356–10365.
[40] K. Lee, Z. Chen, X. Yan, R. Urtasun, and E. Yumer, "Shapeadv: Generating shape-aware adversarial 3D point clouds," 2020, *arXiv:2005.11626*.
[41] Y. Li, C. Wen, F. Juefei-Xu, and C. Feng, "Fooling lidar perception via adversarial trajectory perturbation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7898–7907.
[42] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–23.
[43] Z. Wu et al., "3D shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
[44] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1588–1597.
[45] A. Dai et al., "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
[46] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[48] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5010–5019.
[49] Z. Zhang, H. Lin, X. Zhao, R. Ji, and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3D object classification," *IEEE Trans. Image Process.*, vol. 27, no. 12, pp. 5957–5968, Dec. 2018.
[50] L. Luciano and A. Ben Hamza, "Deep similarity network fusion for 3D shape classification," *Vis. Comput.*, vol. 35, pp. 1171–1180, 2019.
[51] J. Jiang, D. Bao, Z. Chen, X. Zhao, and Y. Gao, "MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 8513–8520.
[52] Z. Han et al., "3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 3986–3999, Aug. 2019.
[53] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 484–499.
[54] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 82–90.
[55] S. Liu, L. Giles, and A. Ororbia, "Learning a hierarchical latent-variable model of 3D shapes," in *Proc. IEEE Int. Conf. 3D Vis.*, 2018, pp. 542–551.
[56] Z. Han, M. Shang, Y.-S. Liu, and M. Zwicker, "View inter-prediction GAN: Unsupervised representation learning for 3D shapes by learning global shape memories to support local view predictions," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 8376–8384.
[57] J. Li, B. M. Chen, and G. H. Lee, "So-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9397–9406.
[58] Z. Han, X. Wang, Y.-S. Liu, and M. Zwicker, "Multi-angle point cloud-VAE: Unsupervised feature learning for 3D point clouds from multiple angles by joint self-reconstruction and half-to-half prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 10441–10450.
[59] X. Gao, W. Hu, and G.-J. Qi, "Graphter: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7163–7172.
[60] Y. Chen et al., "Shape self-correction for unsupervised point cloud understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 8382–8391.
[61] M. Xu, Z. Zhou, H. Xu, Y. Wang, and Y. Qiao, "CP-Net: Contour-perturbed reconstruction network for self-supervised point cloud learning," 2022, *arXiv:2201.08215*.
[62] G. Mei, L. Yu, Q. Wu, J. Zhang, and M. Bennamoun, "Unsupervised learning on 3D point clouds by clustering and contrasting," 2022, *arXiv:2202.02543*.
[63] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.
[64] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9782–9792.
[65] X. Yu et al., "Point-bert: Pre-training 3D point cloud transformers with masked point modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19 313–19 322.
[66] A. X. Chang et al., "Shapenet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[67] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *Proc. Symp. Geometry Process.*, 2003, vol. 6, pp. 156–164.
[68] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3D model retrieval," in *Computer Graphics Forum*, vol. 22. Hoboken, NJ, USA: Wiley, 2003, pp. 223–232.
[69] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, Dec. 2015.
[70] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.
[71] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "Meshnet: Mesh neural network for 3D shape representation," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 01, pp. 8279–8286.
[72] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki, "Gift: A real-time and scalable 3D shape search engine," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5023–5032.
[73] M. Yavartanoo, E. Y. Kim, and K. M. Lee, "SPNet: Deep 3D object classification and retrieval using stereographic projection," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 691–706.
[74] S. Bai et al., "Ensemble diffusion for retrieval," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 774–783.

[75] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval," *Comput. Graph.*, vol. 71, pp. 208–218, 2018.
[76] T. Furuya and R. Ohbuchi, "Deep aggregation of local 3D geometric features for 3D model retrieval," in *Proc. Brit. Mach. Vis. Conf.*, 2016, vol. 7, pp. 121.1–121.12.
[77] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai, "Triplet-center loss for multi-view 3D object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1945–1954.
[78] Z. Han et al., "Seqviews2seqlabels: Learning 3D global features via aggregating sequential views by RNN with attention," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 658–672, Feb. 2019.
[79] X. He, T. Huang, S. Bai, and X. Bai, "View n-gram network for 3D object retrieval," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7515–7524.
[80] Y. Liu et al., "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5239–5248.
[81] H. You, Y. Feng, R. Ji, and Y. Gao, "PVNet: A joint convolutional network of point cloud and multi-view for 3D shape recognition," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1310–1318.
[82] L. Yi et al., "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
[83] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8160–8171.
[84] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1534–1543.
[85] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3075–3084.

Congcong Wen (Member, IEEE) received the B.S. degree in geographic information system from the China University of Petroleum, Beijing, China, and the Ph.D. degree from the Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing. His research interests include remote sensing image recognition and three-dimensional computer vision.

Xiang Li (Member, IEEE) received the B.S. degree in remote sensing science and technology from Wuhan University, Wuhan, China, in 2014, and the Ph.D. degree in cartography and GIS from the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, China, in 2019. His research interests include deep learning, computer vision, and remote sensing image recognition.

Hao Huang (Student Member IEEE) received the B.S. degree in digital media technology from the Beijing University of Posts and Telecommunications, Beijing, China, and the M.S. degree in computer science from the University of Rochester, Rochester, NY, USA. He is currently working toward the Ph.D. degree with New York University, New York, NY. His research interests include three-dimensional computer vision and bioinformatics.

Yu-Shen Liu (Member, IEEE) received the B.S. degree in mathematics from Jilin University, Changchun, China, in 2000, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a Postdoctoral Researcher with Purdue University, West Lafayette, IN, USA. He is currently an Associate Professor with the School of Software, Tsinghua University. His research interests include shape analysis, pattern recognition, machine learning, and semantic search.

Yi Fang (Member, IEEE) received the B.S. and M.S. degrees in biomedical engineering from Xi'an Jiaotong University, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in mechanical engineering from Purdue University, West Lafayette, IN, USA, in 2011. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE. His research interests include three-dimensional computer vision and pattern recognition.