CP-Net: Contour-Perturbed Reconstruction Network for Self-Supervised Point Cloud Learning

Mingye Xu, Zhipeng Zhou, Hongbin Xu, Yu Qiao, Senior Member, IEEE, and Yali Wang

Abstract—Self-supervised learning has not been extensively investigated in the context of point cloud analysis. Current frameworks are predominantly rely on point cloud reconstruction. Given only 3D coordinates, such approaches tend to learn local geometric structures and contours but struggle to comprehend high-level semantic content. Consequently, they achieve unsatisfactory performance in downstream tasks such as classification, segmentation, etc. To fill this gap, we propose a generic Contour-Perturbed Reconstruction Network (CP-Net), which can effectively guides self-supervised reconstruction to learn semantic content in the point cloud, and thus promote discriminative power of point cloud representation. Initially, we introduce a concise contour-perturbed augmentation module for point cloud reconstruction. With guidance of geometry disentangling, we divide point cloud into contour and content components. Subsequently, we perturb the contour components and preserve the content components on the point cloud. As a result, self supervisor can effectively focus on semantic content, by reconstructing the original point cloud from such perturbed one. Next, we use this perturbed reconstruction as an assistant branch, to guide the learning of basic reconstruction branch via a distinct dual-branch consistency loss. In this case, our CP-Net not only captures structural contour but also learn semantic content for discriminative downstream tasks. Finally, we perform extensive experiments on a number of point cloud benchmarks. Part segmentation results demonstrate that our CP-Net (81.5% of mean Intersection over union) outperforms the previous self-supervised models, and narrows the gap with the fully-supervised methods. For classification, we get a competitive result with the fully-supervised methods on ModelNet40 (92.5% accuracy) and ScanObjectNN (87.9% accuracy).

Index Terms—3D point cloud analysis, classification, part segmentation, unsupervised learning.

Manuscript received 26 March 2022; revised 12 July 2023, 5 December 2023, and 18 March 2024; accepted 19 March 2024. Date of publication 27 March 2024; date of current version 21 August 2024. This work was supported in part by the National Key R&D Program of China under Grant2022ZD010505 and in part by the National Natural Science Foundation of China under Grant62272450. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. S. Lee. (Corresponding author: Yali Wang.)

Mingye Xu is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, also with the University of Chinese Academy of Sciences, Beijing 101408, China, and also with Interactive Entertainment Group, Tencent Inc., Shenzhen 518054, China (e-mail: my.xu@siat.ac.cn).

Zhipeng Zhou is with Alibaba Damo Academy, Hangzhou 310052, China.

Hongbin Xu is with the South China University of Technology, Guangzhou 511442, China.

Yu Qiao and Yali Wang are with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China (e-mail: yl.wang@siat.ac.cn).

Our code is available at https://github.com/MingyeXu/cp-net

Digital Object Identifier 10.1109/TMM.2024.3382512

I. INTRODUCTION

POINT cloud analysis has gradually become an important problem for understanding 3D world, resulting from its wide applications in robotics, AR/VR, autonomous driving, etc [1], [2], [3], [4], [5]. Current mainstream methods of point cloud analysis are mainly driven by fully supervised deep learning [6], [7], [8], [9], [10], [11], [12]. However, these methods require a large number of manual annotations, which could be expensive and infeasible for practical applications. Therefore, it is desirable to obtain discriminative representations of 3D point cloud in a self-supervised manner.

The current self-supervised methods are mainly based on pretext tasks provided by generation or reconstruction [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. However, the performance of these methods significantly lags behind the fully-supervised approaches. To fill this gap, we investigate the underlying problem in the self-supervised point cloud learning. As Fig. 1 shows, we take the point cloud segmentation task of an airplane for illustration. We first visualize point cloud by the corresponding feature response in different methods. Due to the fact that supervised signals are part level information, supervised features reflect clear semantic distinctiveness between different parts. While self-supervised reconstruction excels at capturing structural contours [18], it struggles to differentiate content component. Regarding the content component, it is always flat and associated with the gentle variations in [23]. These regions carry more semantic representations (semantic parts) than the edge component. Therefore, we reinterpret it as bearing the semantic content. For instance, the fuselage and wing contours are distinctive, It is difficult to distinguish different parts using such self supervised features, their internal features exhibit similar responses, making it challenging to differentiate between these parts using self-supervised features. This observation further propels our exploration into self-supervised point cloud representations, considering both the contour and content aspects. Specifically, we employ geometry disentangle module [23] to separate point cloud into contour and content components, and display the feature response distribution for the fuselage and wing across the complete point cloud, contour, and content components in Fig. 1. As anticipated, the feature distribution for distinct semantic parts can be readily segregated in the contour components, whereas they significantly overlap in the content components. This finding suggests that self-supervised reconstruction lacks the ability to distinguish between semantic parts within content components.
To tackle this challenge, we propose a generic Contour-Perturbed Reconstruction Network (CP-Net). This network effectively guides self-supervised reconstruction to focus more on the discriminative content of the object, leveraging point cloud disentangling. Our approach involves geometrically separating a point cloud into contour and content components and augmenting it by perturbing the contour while preserving the content. By reconstructing original point cloud from the contour-perturbed one, we can effectively force self-supervisor to learn semantic content. Additionally, we build up a weight-sharing dual branch structure to boost point cloud representation learning. Since the basic branch excels at learning structural contour while overlooks semantic content, we leverage such perturbed reconstruction as an assistant branch of the original reconstruction task. Via designing a novel dual-branch consistency loss, we can gradually guide the basic branch to focus more on semantic content information under the guidance of the assistant branch.

Experiments demonstrate that our method outperforms the self-supervised methods and narrows the gap between unsupervised and supervised models in part segmentation task, achieving 81.5% mIoU on ShapeNetPart. For classification, our self-supervised method gets a comparable result with the fully-supervised methods on ModelNet40 (92.5% accuracy) and ScanObjectNN (87.9% accuracy).

II. RELATED WORK

A. Supervised Learning on 3D Point Clouds

Recently 3D point cloud analysis has enjoyed some remarkable progress for various downstream tasks, which benefits from the deep learning methods can directly to consume the point cloud with irregular structure [6], [7], [8], [9], [10], [11], [12]. PointNet [6] and DeepSet [24] propose a groundbreaking way to deal directly with the point clouds. They learn spatial coding separately for each point and use max pooling to deal with the order invariant problem. However, such methods also have the problem of insufficient local geometry structures extraction. To
remedy this, PointNet++ [7] propose a hierarchical grouping architecture to extract local features from the neighbors of each point. Some subsequent works such as PointCNN [25], PointConv [26] and RSCNN [11] also focus on the extraction of local geometric features via the treatment of local structures. To capture the holistic geometric information more efficiently, GS-Net [8] groups distant points with similar and relevant geometric information and aggregates features from neighbors in both Euclidean space and Eigenvalue space. DGCNN [10] utilizes the nearest neighbors in the features space dynamically. While these supervised methods have advanced the state-of-the-art in point cloud deep learning, leveraging extensive supervised signals, their generalization capability might be constrained by the inherent limitations of the supervised learning mechanism. Therefore, it is desirable to obtain features in an unsupervised manner and obtain the general representation of 3D point clouds.

B. Unsupervised Learning on 3D Point Clouds

In order to produce a semantic latent space without relying on annotations, unsupervised networks are trained to perform tasks based on information derived from the point cloud itself. Based on this, recent self-supervised approaches design various pre-tasks such that models need to learn useful information from data itself [27], [28], [29], [30], [31]. Several prior works have attempted on learning representation of point cloud without human supervision [17], [18], [20], [21], [22], [32]. FoldingNet [20] trains an end-to-end auto-encoder that consumes unordered point clouds directly by reconstruction from the point cloud itself. PointGLR [18] focuses on reasoning between local and global representations. However, it can not deal with semantic information well in the unsupervised scheme. GraphTER [33] proposes graph transformation equivalent representation learning to extract unsupervised representations. Chen et al. [34] destroys some local shape parts of the object, and then segments which points belong to the distorted part. Moreover, parallel branches have been proven effective in different studies: Graph-PBN [35] introduces a graph-based parallel branch network, which combines the advantages of PointNet and GCN to achieve efficient and accurate recognition results. So-Net [36] models the spatial distribution of point cloud by building a Self-Organizing Map (SOM) and performs hierarchical feature extraction via parallel branches. Different from these parallel-branched methods, we design a dual-branched CP-Net to improve discriminative self-supervised representation both on semantic content and structural contour.

III. METHOD

This section will introduce our proposed CP-Net in detail. We commence by presenting the overall framework, followed by an introduction to the contour-perturbed augmentation module implemented in the assistant branch. Lastly, we delineate the loss terms incorporated in our CP-Net.

A. Overall Architecture of Our CP-Net

Our CP-Net is a dual-branched network comprising an assistant branch and a basic branch. The assistant branch is used to learn discriminative representation on semantic content, while the basic branch preserves the discriminative representation of the structural contour. Both branches share the weights of the feature extraction and prediction networks, which are used for point cloud reconstruction. Feature extraction network is designed to obtain the global feature and point-wise feature. Prediction network reconstructs the coordinates and estimate the normal vectors. By introducing dual-branch consistency loss as feature consistency regularization, we can leverage the assistant branch to guide the basic branch for distinguishing content information of point clouds.

As shown in Fig. 2, we consider a 3D point cloud with \(N\) points as the input. Generally, point cloud contains \(3D\) coordinates \(P = \{p_1, \ldots, p_N\}\), and normal vectors \(N = \{n_1, \ldots, n_N\}\). The basic branch receives the original point cloud coordinates \(P\) as input, while the assistant branch receives the perturbed point cloud coordinates \(P'\) generated by the contour-perturbation augmentation module. This extractor network receives point cloud coordinates as input, and outputs the point-wise features \(Y\) and global features \(G\). The reconstructed coordinates and normal vectors of prediction network can be defined as \(\hat{P}\) and \(\hat{N}\).

Feature Extraction Network: Like PointNet++ [7], we use a hierarchical structure to learn point cloud feature progressively with skip connections. Specifically, at the \(l\)-th level of the encoder, the point set is downsampled using iterative furthest point sampling to generate a new point set \(P^l \subset P^{l-1}\), which consists of \(N^l\) points derived from the \(N^{l-1}\) points. Meanwhile, we extract the point-wise feature \(f^l \subset F\) by applying the RS-Conv [11] for each point \(p_i^l \subset P^l\). For corresponding \(l\)-th level of decoder, the input feature is \(Q^l\), we use the transition-up module [37] to propagate the points feature via multi-layer perceptron (MLP):

\[
Q^{l-1} = MLP\left(MLP(\xi(Q^l)) + MLP(F^{l-1})\right)
\]

where \(\xi\) is the point interpolation function [7]. Moreover, we also use the transition-up module to propagate the feature \(Q^l\) of each level \(l\) to \(Y^l\) from \(N^l\) points to the original \(N\) points:

\[
Y^l = MLP(\xi(Q^l))
\]

Then we concatenate them together as the point-wise feature \(Y = concatenate[Y^1, \ldots, Y^L]\), where \(L\) is the layers of feature extractor network. While the global feature \(G\) is obtained by a symmetric aggregation function (e.g., max pooling, ...) operating on the point-wise feature.

Prediction Networks: Self-supervised prediction networks are frequently employed in self-supervised learning [18], [29]. These networks typically comprise a standard prediction network and a point cloud reconstruction network. The normal prediction network is used to enhance point cloud representation with geometry structure information. We can take the concatenation of the global feature \(G\), original point coordinates \(p_i\) and point-wise feature \(y_i\) from \(Y\) as input, then obtain the estimated normal \(\hat{n}_i \subset \hat{N}\) through a shared light-weight MLP and \(l^2\) normalization operation II:

\[
\hat{n}_i = II(MLP(\mathbb{I} + y_i + G))
\]

where \(\mathbb{I}\) is the concatenation operator, and \(\hat{n}_i \in \mathbb{R}^3\).
IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 26, 2024

Fig. 2. Network architecture of our CP-Net. “RC” means RS-Conv module, “TU” means transition up module. The basic branch uses the original point cloud to reconstruct its coordinates and normal vectors. Meanwhile, the assistant branch processes the contour-perturbed point cloud as input and reconstructs the original point cloud. The feature extractor network and prediction network from two branches share the parameters.

Fig. 3. Process of contour-perturbed augmentation module.

The self-reconstruction network is designed to recover the coordinate information of the original point cloud. Following the approach of FoldingNet [20], we incorporate a standard two-dimensional grid to deform the reconstructed coordinates with the guidance of the global feature $G$. The self-reconstruction network contains two consecutive 3-layer MLPs. Specially, before feeding the global feature $G$ into this network, we replicate it $N$ times as $\hat{G}$, Subsequently, we concatenate it with an matrix $I \in \mathbb{R}^{N \times 2}$. The reconstructed point cloud $\hat{P}' \in \mathbb{R}^{N \times 3}$ can be obtained by the following operation:

$$\hat{P}' = MLP(\hat{G} \oplus MLP(\hat{G} \oplus I))$$

where $\oplus$ is the concatenation operator. $MLP$ is the multilayer perceptron.

B. Contour-Perturbed Augmentation Module

As stated in the Section I, self-supervised reconstruction primarily concentrates on structural contours the discriminative content information of point cloud. To tackle this problem, we design the contour-perturbed augmentation module for the assistant branch in Fig. 3. Our observations indicate that the geometry-disentangle module [23] can decompose the point cloud into the structural contour information and semantic content information. Motivated by this observation, we can devise our contour-perturbed augmentation module to perturb the point cloud accordingly.

First, for the input point cloud $P$ with $N$ points, we construct the point graph with the eigenvalues which represent the graph frequencies. Second, we collect the contour components $P_s \in \mathbb{R}^{M \times 3}$ and content components $P_g \in \mathbb{R}^{M \times 3}$ through the graph filters [23] on the constructed graph, where $M = N/2$. The points within the contour components $P_s$ are more adept at conveying the local geometric structural information of the point cloud. Conversely, the content points $P_g$ are capable of highlighting relatively common semantic information. Third, we perturb the contour points $P_s$ with normal distributed noise $\Delta \in \mathbb{R}^{M \times 3}$, then we concatenate the perturbed contour points $P'_s = P_s + \Delta$ with original content points $P_g$ as the perturbed point cloud $P' = P'_s \oplus P_g$, where $P' \in \mathbb{R}^{N \times 3}$. Finally, we use the perturbed point cloud $P'$ to reconstruct the original point cloud in the assistant branch. This approach enables the branch to focus more intently on the semantic content information.

Besides of these methodology insights, our experiments in Table VII can also prove our statement, e.g., sharp perturbation (Setting H) vs. gentle perturbation (Setting G): 81.5% vs. 80.8%. it clearly shows that content components are harder to learn in the self-supervised manner. Hence, we use sharp perturbation to make our model focus on reconstructing gentle component. Moreover, we also consider other point cloud decomposition schemes, such as point cloud clustering, spatial domain decomposition, etc.

C. Loss Terms

To learn our model effectively, we introduce three training losses:

$$Loss = \mathcal{L}_{\text{dual}} + \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{normal}}$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
where the dual-branch consistency loss, $L_{\text{dual}}$, is used to guide the basic branch to learn the semantic content representation with assistant branch. $L_{\text{recon}}$ and $L_{\text{normal}}$ are widely used self-reconstruction losses. Besides, dual-branch consistency loss is composed of object-wise consistency loss, point-wise consistency loss and point-to-object consistency loss.

$$L_{\text{dual}} = L_{d-o} + L_{d-p} + L_{d-p2o}$$

(6)

**Dual-branch consistency loss:** Once the perturbed point cloud $P'$ has been obtained, we input both the perturbed and original point clouds into the feature extraction networks to extract the point-wise feature $y_i \subset Y$ and global feature $G$ for basic branch, $y'_i \subset Y'$ and $G'$ for assistant branch. Subsequently, we formulate a dual-branch consistency loss function to facilitate the convergence of corresponding features across different branches. This innovative approach enables the transmission of discriminative content information from the auxiliary branch to the primary branch. Moreover, to capture multi-scale semantics in the point cloud, we introduce such consistency losses in different scales, including object-wise consistency loss, point-wise consistency loss, and point-to-object consistency loss.

The object-wise consistency loss, denoted as $L_{d-o}$, primarily aims to maintain the global consistency between the perturbed global representation and the fundamental original representation. We calculate the similarity of the global features from two branch as follows:

$$L_{d-o} = 1 - \cos(G, G')$$

(7)

In contrast, the point-wise consistency loss $L_{d-p}$ operates on point-wise representations and is utilized to enhance the feature relevance between the basic branch and assistant branch. We can operate the similarity of features of each corresponding point as follows:

$$L_{d-p} = -\sum_{i=1}^{N} \log \frac{\exp \left(y_i \cdot y'_i / \tau \right)}{\sum_{j=1}^{N} \exp \left(y_i \cdot y'_j / \tau \right)}$$

(8)

Moreover, inspired by [18], we employ the point-to-object consistency loss $L_{d-p2o}$ to explore the distinct properties by connecting local and global representations of different branches. This is done in order to bring the point-wise representation as close as possible to the global representation. This loss can be formulated as:

$$L_{d-p2o} = -\sum_{i=1}^{N} \log \frac{\exp \left(y_i \cdot G'/\tau \right)}{\sum_{k=1}^{B} \exp \left(y_i \cdot G'_k / \tau \right)} - \sum_{i=1}^{N} \log \frac{\exp \left(y'_i \cdot G/\tau \right)}{\sum_{k=1}^{B} \exp \left(y'_i \cdot G_k / \tau \right)}$$

(9)

where $B$ represents the batch size, $\{G'_k, k = 1, 2, \ldots, B\}$ are the global features of different point clouds. Different with the loss of PointGLR [18], our point-to-object loss acts on the representation between basic branch and assistant branch. This approach is utilized to learn self-supervised representations for both contour structures and semantic contents.

**Reconstruction Loss:** Based on our dual-branch framework, we can extract the global representation $G$ and $G'$ respectively for original and perturbed point clouds. To perform self-reconstruction, a FoldingNet based predictor [20] is used to deform the normal 2D grid with $G$ and $G'$ onto the 3D coordinates of the reconstructed point cloud $P$ and $P'$. We calculate the reconstruction loss of reconstructed point cloud and original point cloud, which is defined as the chamfer distance [42]:

$$L_{\text{recon}} = \sum_{p \in P} \min \| \hat{p} - p \|_2 + \sum_{\tilde{p} \in P'} \min \| \hat{\tilde{p}} - p \|_2$$

$$+ \sum_{p \in P} \min \| p' - p \|_2 + \sum_{\tilde{p} \in P'} \min \| p' \hat{p} - p \|_2$$

(10)

**Normal Estimation Loss:** The normal vector of a point cloud is a fundamental feature, crucial to numerous point cloud processing algorithms, as highlighted in studies such as [6], [18]. The task of normal estimation necessitates the construction of a sophisticated representation on the surface of the 3D object, underscoring its significance in the realm of point cloud processing. In the process of self-supervised feature learning, we do not need to pursue the accuracy of the estimated normal vectors [11], but we need to use this task as a self-supervised signal to improve the point-wise level of self-supervised representation. We use cosine loss to measure the estimation error:

$$L_{\text{normal}} = 1 - \frac{1}{N} \sum_{i=1}^{N} \cos(\hat{n}_i, n_i)$$

(11)

where $\hat{n}_i$ and $n_i$ are predicted normal vector and original normal vector.

IV. EXPERIMENTS

This section will introduce the implementation details and the experimental comparisons for point cloud classification and part segmentation.

A. Implementation Details

All our models are trained on a single GTX 2080Ti GPU with the deep learning library Pytorch [44]. Our model is trained using the Adam [45] optimizer with a basic learning rate of 0.001, which is reduced by 0.7 every 20 epochs. The momentum of the batch normalized [46] layer starts from 0.9, and then decays at a rate of 0.5 every 20 epochs.

For self-supervised framework of classification, we employ merely three RSConv modules within the extractor network to effectively extract self-supervised representation. We utilize both the global consistency loss and the local-to-global consistency loss to maintain the consistency of global fine-grained representations. For segmentation tasks, we employ the transition-up component, as proposed in [37], comprising four layers to acquire a diverse set of point-wise self-supervised features. In this context, the intermediate features must be propagated to the original number of points. By concentrating on point-wise representation, we can effectively leverage the local consistency loss as a dual-branch consistency loss, further enhancing the segmentation performance. As for the evaluation, we can randomly
select the training set by category. For the contour-perturbed augmentation module, we jitter the contour points with normal distributed noise with std of 0.02.

### B. Point Cloud Part Segmentation

The purpose of point cloud part segmentation is to predict the part category label of each point in a given point cloud. We evaluate the features of each point learned by our self-supervised model on the ShapeNetPart dataset [48] which is pre-trained on ShapeNetPart and ShapeNet [49] dataset. ShapeNetPart [48] contains 16,881 objects from 16 categories. Each object consists of 2 to 6 parts with total of 50 distinct parts across all categories. While ShapeNet [49] contains 57,000 models across 55 categories.

Regarding the training configuration, as outlined in Table I, we use the training set of ShapeNetPart (without annotations) for unsupervised pretraining, and then use only R% samples of the training set for fine-tuning the unsupervised feature. It can be observed that with only 50% training annotations, our self-supervised CP-Net can achieve 82.5% instance mIoU. Table I also compares our model to fully-supervised models, with our model achieving an mIoU that is only 4.1% lower than the best supervised model. In addition, to disentangle the performance benefits due to unsupervised training, we trained the fully-supervised DGCNN [10] in Table I with only 5% training data. It only achieves 76.6% instance mIoU, which is worse than our semi-supervised method (81.5% mIoU with 5% train set). This indicates that our method has powerful generalization capacity on limited data.

In Table II, we present the results of our experiments using the entire training set of ShapeNet (without annotations) for pretraining and 1%/5% of the ShapeNetPart training set for fine-tuning. These settings follow [17], [21], [22], [39], to ensure fair comparison. To evaluate the transferability of our self-supervised approach, we train our model on ShapeNet [49] and subsequently assess its performance on ShapeNetPart. The results are shown in Table II. CP-Net outperforms other methods on the transfer learning to ShapeNetPart.

Moreover, in the setting of GraphTER [33], they use 100% train set of ShapeNetPart (with labels) for supervised fine-tuning (same pretraining like us). In contrast, our setting is actually more challenging with less data in fine-tuning. Under the same setting of GraphTER [33], our method (83.2%) outperforms GraphTER (81.9%) on ShapeNetPart segmentation task.

### C. Unsupervised Point Cloud Classification

For the unsupervised classification, we first obtain the self-supervised shape features from the ModelNet40 [52] and ScanObjectNN [53] dataset using self-supervised pre-trained model. Then we use a linear SVM [54] to classify self-supervised shape features. ModelNet40 [52] is a benchmark dataset for shape classification, consisting 9,843 training samples, 2,468 testing samples and 40 object categories. ScanObjectNN [53] is a real-world dataset, where 2,902 3D objects are extracted from scans. In our classification experiments, we adopt a consistent approach of sampling 1,024 points from each point cloud for both training and evaluation phases. This uniform sampling strategy ensures a balanced and fair assessment of the model’s performance. All our results are measured using a single view without multi-view voting trick to show the neat performance of different models. Multi-view voting trick is a testing mechanism of multiple votes to select the optimal test result. Surface normal vectors are used to provide self-supervised signals for our models trained on ModelNet40 but were not used as input. For the models trained on ScanObjectNN, normal vectors were not used due to their inaccuracy in real-world data. Following [18], we train the feature using self-supervised learning methods on the source dataset and used a linear SVM [54] trained on target...
dataset to perform classification. The results are presented in Tables III and IV.

**Unsupervised learning on ModelNet40:** As shown in Table III, we compare the performance of unsupervised classification methods and fully-supervised classification methods. The results presented in the table can be divided into three distinct sections. The upper section (Supervised) is the result of the fully-supervised SOTA methods, the middle section (Unsupervised simple) displays the classification results achieved through self-supervised feature learning on the ModelNet40 dataset, and the lower section (Unsupervised difficult) is the unsupervised transfer learning from the ShapeNet to the ModelNet40 dataset.

As for learning on ModelNet40, many studies [10], [11], [25] have shown that the performance of ModelNet40 has been gradually saturated. CP-Net achieves a comparable performance 92.5% with the SOTA unsupervised method PointGLR [18]. Remarkably, our proposed CP-Net method attains highly competitive results, even when compared to state-of-the-art supervised techniques, achieving an impressive 92.9% accuracy without employing the voting trick in an unsupervised setting. This evidence indicates that CP-Net can discover global semantic representation shared in different kinds of point clouds.

For better evaluation and further explore the generalization ability of the learned representation, we use a more challenging transfer setting (Table III: bottom), we test CP-Net with transfer learning from ShapeNet to ModelNet40, unsupervised pretraining on ShapeNet followed by evaluation of the unsupervised representation using an SVM on ModelNet40. Our method largely outperforms the SOTA approaches. It shows that the unsupervised features from CP-Net are more generic than other methods.

**Unsupervised Learning on ScanObjectNN:** In order to verify the effectiveness of CP-Net more comprehensively, we conduct the same unsupervised classification task on the ScanObjectNN [53]. This dataset is used to investigate the robustness to noisy objects with deformed geometric shape and non-uniform surface density in the real world. We adopt our model on the OBJ ONLY (simplest variant of the dataset). The results are summarized in Table IV. CP-Net achieves the comparable accuracy with the fully-supervised methods, demonstrating its strong practicality in the real world data.

### V. NETWORK ANALYSIS

In this section, we first introduce the ablation studies of our framework. Second, we analyze the details of contour perturbation module. Then we further present the analysis of the normal estimation loss, self-reconstruction loss and the dual-branch consistency loss. Furthermore, we conduct an in-depth analysis to evaluate the robustness of CP-Net concerning sampling density variations.

#### A. Ablation Studies of the Network Architecture

In order to examine the effectiveness of our designs, we conduct architecture ablation studies based on our framework. In Table VI, we present a comparative analysis of the branches within our framework. The results reveal a significant performance gap when utilizing only the basic or assistant branch, as opposed to employing the fully complete dual-branch structure. It can be concluded that the feature distinction of contour components (basic branch) and semantic content components (assistant branch) are both important in the self-supervised feature representation. Of greater significance is the observation that, through consistency learning between the two branches, we can substantially improve the performance by augmenting the representation relevance between semantic content information and structural contour information.

Moreover, concerning the comparison with PointGLR [18], the summary results of CP-Net and PointGLR are shown in Table V. Although our self-supervised classification accuracy on ModelNet40 is slightly lower than PointGLR (92.5% vs 92.9%), our performance surpasses PointGLR in the more challenging outdoor object unsupervised classification task (87.9% vs 86.9%), as shown in Table IV. For other settings, the original paper PointGLR did not conduct relevant experiments, so

### TABLE III

**COMPARISON OF CLASSIFICATION RESULTS ON MODELNET40 DATASET**

| Method          | Acc. (%) |
|-----------------|----------|
| Supervised      |          |
| PointNet [6]    | 89.2     |
| PointNet++ [7]  | 90.5     |
| PointCNN [25]   | 92.5     |
| DGCNN [10]      | 92.9     |
| RSCNN [11]      | 92.9     |
| RSCNN (vote)    | 93.6     |
| DGCNN [10]      | 93.5     |
| SO-Net [39]     | 93.4     |
| Konv [12]       | 92.9     |
| Unsupervised (simple) |      |
| PointHop [47]   | 89.1     |
| PointHop++ [47] | 91.1     |
| PointGLR [18]   | 92.9     |
| Chen et. al. [34]| 92.4    |
| Ours            | 92.5     |
| Unsupervised (difficult) |     |
| FoldingNet [20] | 88.9     |
| PointCapsNet [17]| 88.9   |
| MultiTask [22]  | 89.1     |
| UFF [21]        | 90.4     |
| Ours            | 91.9     |

"Vote" is using the testing voting trick. "Simple" means the self-supervised methods are trained and tested on ModelNet40, while "difficult" means that the self-supervised methods are trained on shapeNet and tested on ModelNet40.

The bold values represent the results of our method.

### TABLE IV

**COMPARISON OF CLASSIFICATION RESULTS ON REAL-WORLD SCANOBJECTNN DATASET (OBJ ONLY)**

| Method          | Acc. (%) |
|-----------------|----------|
| Supervised      |          |
| SDaFV [30]      | 73.8     |
| PointNet [6]    | 79.2     |
| SpiderCNN [9]   | 79.5     |
| DGCNN [10]      | 86.2     |
| PointCNN [25]   | 85.5     |
| GDAE [23]       | 88.1     |
| PointBERT [51]  | 88.1     |
| Ours            | 86.9     |
| Unsupervised    |          |
| PointGLR [18]   | 86.9     |
| Ours            | 87.9     |

The bold value represent the results of our method.
we added our reproduced results of PointGLR. In the ModelNet40 Difficult setting, our results are on par with PointGLR. On ShapeNetPart, our results surpass PointGLR since our self-supervised feature representation is more segmentation-friendly. Furthermore, we compared feature visualization, as shown in

Fig. 6, where the baseline method is PointGLR. It can be observed that our method learns self-supervised features that are more semantically effective.

**B. Analysis of Contour-Perturbed Augmentation Module**

To explore an effective way to perturb the point cloud for self-supervised feature learning, as Table VII and Fig. 4 show, we design a variety of ways to perturb the point cloud to the assistant branch. We choose the downstream task of part segmentation with 5% training data, where the self-supervised features are learned from ShapeNetPart.

![Fig. 4. Different perturbation manners for the assistant branch (corresponding to Table VII).](image1)

**TABLE V**

**COMPREHENSIVE COMPARISON WITH POINTGLR WITH IDENTICAL RSCNN BACKBONE**

|                        | PointGLR [18] | CP-Net |
|------------------------|--------------|--------|
| ShapeNetPart seg.      | 77.5%*       | 79.3%  |
| (Table II self 1%)     |              |        |
| ShapeNetPart seg.      | 78.4%*       | 81.5%  |
| (Table II self 5%)     |              |        |
| ModelNet40 cls.        | 92.9%        | 92.5%  |
| (Table III simple)     |              |        |
| ModelNet40 cls.        | 92.2%*       | 91.9%  |
| (Table III difficult)  |              |        |
| ScanobjectNN cls.      | 86.9%        | 87.9%  |
| (Table IV)             |              |        |

* denotes our reproduced result.

![Fig. 5. Sampling density robustness test compared to the supervised version on classification and part segmentation. (a). Test results on ModelNet40 of using sparser points as the input to a model trained with 1,024 points. (b) Test results on ShapeNetPart of using sparser points as the input to a model trained with 2,048 points.](image2)

**TABLE VI**

**ABSTRACTION STUDY OF THE BRANCHES**

|                      | mIoU(%) |
|----------------------|---------|
| only basic branch    | 78.4    |
| only assistant branch| 78.2    |
| basic branch         | 81.5    |

We report the mIoU on ShapeNetPart semi-supervised segmentation with 5% train data, where the self-supervised features are learned from Shapenetpart.

![Fig. 6. Comparison of Feature representation. We show the supervised version (a), self-supervised baseline (b) and our self-supervised CP-Net (c). The point color refers to the activation value of each point which is obtained by averaging all the entries in the feature vector.](image3)

**TABLE VII**

**ABLATION STUDIES OF COUNTER-PERTURBED AUGMENTATION MODEL: DIFFERENT PERTURBATION MANNERS**

| Perturbation Manners          | mIoU(%) |
|-------------------------------|---------|
| A Randomly delete a cluster part | 78.7    |
| B Delete contour points       | 80.8    |
| C Jitter all points           | 81.0    |
| D Randomly delete contour or content points | 79.9 |
| E Jitter all points           | 79.6    |
| F Randomly jitter a cluster part | 78.8    |
| G Jitter content points       | 80.8    |
| H Jitter contour points       | 81.5    |
| I Randomly jitter contour or content points | 81.1 |

We report the mIoU on shapenetpart semi-supervised segmentation with 5% train data, where the self-supervised features are learned from ShapeNetPart.
first consider the aggregation part information of the original point cloud. Specially, we use the non-negative matrix factorization method [55] to extract similar clustering effects from the original point cloud, and then randomly select one cluster as the cluster part mentioned in Model A and F. However, no matter jitter or delete a cluster part randomly, the results are not particularly ideal, because there extends a significant discrepancy exists between the segments generated through clustering and the actual ground truth. This may result in the network acquiring irrelevant information. To verify the effectiveness of contour-perturbed augmentation module, we conduct a series of comparative experiments (Model B,C,D,G,H,I). Model C indicates that the contour points have certain useful information; Model B and D show that deleting the content points can harm the performance (0.7% ↓). Model G, H and I verify that jittering contour points can highlight the learning of holistic and generic information. All the above experiments verified that the contour-perturbed augmentation module can extract relatively efficient self-supervised representation and show better performance in downstream tasks.

Based on Model H, we further analyze the perturbation details. Table VIII shows the mIoU evaluation when increasing the number of jittered points of the contour points, when we use 1024 points ($N/2$), we get the best performance. As for the normal distributed noise, we conduct a series of experiments based on the std of the normal distributed noise, which is shown in Table IX. Furthermore, To verify the performance difference of the backbone network in the ablation experiment of the counter-perturbed augmentation module, we conduct different ablation experiments for the backbone network of this method. These experiments are detailed in Tables VIII and IX. Specifically, we adopt three different backbones: RSCNN [11], PointNet++[7], and GSNet [8] backbone. Conversely, RSCNN demonstrates commendable performance when employed as a backbone, further substantiating its potential as a versatile and high-performing backbone for self-supervised tasks.

### Table VIII

| # Jittered Points | 512 | 1024 | 1536 | 2048 |
|-------------------|-----|------|------|------|
| CP-Net(RSCNN)     | 80.9 | 81.5 | 81.3 | 79.6 |
| CP-Net(PointNet++)| 80.1 | 81.0 | 81.5 | 80.2 |
| CP-Net(GSNet)     | 79.6 | 80.3 | 79.8 | 78.5 |

We report the mIoU on ShapeNetPart segmentation with 5% train data.

### Table IX

| STD  | 0.01 | 0.02 | 0.03 |
|------|------|------|------|
| CP-Net(RSCNN) | 80.5 | 81.5 | 80.7 |
| CP-Net(PointNet++) | 80.4 | 81.3 | 80.9 |
| CP-Net(GSNet) | 79.7 | 80.8 | 80.0 |

We report the mIoU on ShapeNetPart segmentation with 5% train data.

### C. Analysis of Network Losses

In Table X, we report the mIoU of segmentation with 5% train data, with the self-supervised representation learned from the ShapeNetPart. Model a, b, and d are based on our baseline model which is training with the basic branch, without assistant branch. Model b can be viewed as a variant of FoldingNet [20], which is trained by self-reconstruction loss only and get a low segmentation mIoU of 71.7%, while model b and d show slight improvements with normal estimation loss, because normal estimation is a point-wise task, this self-supervised signal can affect the self-supervised feature for each point, thereby improving the performance of part segmentation. When we only use dual-branch consistency loss without reconstruction and normal estimation loss, we can still get a result of 79.8%. Compared with model a, b and d, model e shows that the contour-perturbed augmentation module with dual-branch consistency loss has a great improvement in performance. Based on dual-branch consistency loss, we add the normal estimation module, self-reconstruction module to model c, which can be indicated as model e, f and g. They boost significant improvements, and the best performance is 81.5% of model g.

In this section, we delve deeper into the analysis of dual-branch consistency losses, which serve to minimize the feature distance between point clouds originating from the basic and assistant branches.

Due to different attentions on point cloud representation of the different downstream tasks, here we conduct some ablation studies of the dual-branch consistency losses for part segmentation downstream task and classification downstream task. For part segmentation, which focus more on point-wise local representation, the point-wise consistency loss is more critical as Table XI indicates. Hence, for the pre-training phase of the part segmentation task, it is feasible to employ the point-wise consistency loss as a dual-branch consistency loss. As Table XII shows, for classification, object-wise consistency loss and point-to-object...
Table XII
Ablation Study of Consistent Loss for Classification Downstream Task

| L_{d-o} | L_{d-p} | L_{d-p2o} | Acc. (%) |
|--------|---------|-----------|---------|
| ✓      | ✓       | ✓         | 91.3    |
| ✓      | ✓       | ✓         | 91.5    |
| ✓      | ✓       | ✓         | 92.5    |

We report the accuracy on ModelNet40 test set.

Table XIII
Sampling Density Robustness Testing of Different Self-Supervised Scheme

| # input points | 2048 | 1024 | 512 | 256 | 128 | 64 |
|----------------|------|------|-----|-----|-----|----|
| Reconstruction as [20] | 76.2 | 74.4 | 75.3 | 72.6 | 72.1 | 71.7 |
| Completion as [56] | 78.9 | 75.3 | 74.3 | 72.5 | 72.3 | 72.1 |
| Normal Estimation | 71.7 | 71.3 | 70.9 | 71.1 | 70.7 | 70.5 |
| Contrast as [32] | 79.9 | 76.9 | 75.6 | 74.5 | 73.1 | 72.5 |
| PointGLR [18] | 78.4% | 76.3% | 74.9% | 74.2% | 73.4% | 72.5% |
| Ours | 81.5 | 77.4 | 75.3 | 74.2 | 73.2 | 72.3 |

We report the mIoU on shapenetpart segmentation with 5% training data.

consistency loss play a critical role in performance improvement.

D. Robustness Analysis

Fig. 5 shows the robustness of CP-Net on sampling density compared to the supervised version. Following [18], we use sparse points (1024, 512, 256, 128, 64) as the input of classification model and part segmentation model for testing. For classification (Fig. 5(a)), we feed sparse point clouds to the model trained with 1024 points, and obtain the self-supervised feature, then use a linear SVM to perform the classification results. Our self-supervised classification model proved much more robust to sampling density than the supervised model. Even we use 128 points for testing, the accuracy can achieve to 84.3%. For part segmentation (Fig. 5(b)), we also feed the point clouds with different densities to a segmentation model which is trained with 2048 points. Then we use the same setting with Table I to perform the segmentation results. It can be concluded that our self-supervised method is more robust to the point cloud density.

Additionally, to facilitate a fair comparison with other self-supervised state-of-the-art methods under varying conditions, we have chosen several conventional self-supervised approaches and evaluated their performance using the same backbone network. Specifically, we use the component part segmentation experiment setting with 5% training data in Table II, and input 2048, 1024, 512, 256, 128 and 64 points respectively during the testing, aiming to evaluate the robustness performance under sampling density. All results are shown in Table XIII. All self-supervised methods use RSCNN [11] as feature extraction backbone to reproduce. In this study, we reconstruct the FoldingNet [56] and utilize point cloud reconstruction as a pretext task for self-supervised learning, as described by Deng et al. [56]. Additionally, we explore the point cloud completion task in the Occo framework [57] and normal prediction as alternative pretext tasks. Furthermore, we implement a contrastive learning approach, following the methodology of Xie et al. [32].

In this approach, we replicate the infoNCE loss used by PointContrast [32] and conduct comparative learning on point cloud objects. Our proposed CP-Net demonstrates superior robustness under varying sampling densities compared to most other conventional pre-training schemes. When the input point count is minimal, the performance of CP-Net, PointGLR, and the contrastive learning method are found to be comparable.

VI. VISUALIZATION

A. Reasonable Segmentation

In Tables I and II, we show the part segmentation results of our CP-Net on ShapeNetPart dataset [48]. The mean Intersection over Union (mIoU) is a crucial statistical evaluation metric that indicates the overall segmentation performance. Occasionally, the ground truth may be confused in some situations. In order to show our results more comprehensively, here we visualize some test segmentation results of CP-Net with the ground truth in Fig. 7. Although our results are different from the manual annotation, they are both reasonable manners of segmenting the objects. CP-Net divides the lamp bracket and lamp rope into the same category, while the ground truth labels consider that bases and brackets belong to the same category. The situation is similar for shank, chair leg brackets...

B. Feature Representation Visualization

In order to gain an intuitive understanding of our models, we visualize the point cloud by coloring it according to the point cloud feature response on the test set of ShapeNetPart, as shown in Fig. 6. Notably, points belonging to the same part exhibit similar activation patterns. Upon observation, it becomes evident that the fully supervised feature representations can be effectively distinguished among different parts. In contrast, the common self-supervised feature representation from baseline model (PointGLR) reflect confusion among the different parts. Compared with the baseline, our self-supervised CP-Net shows more distinction which verifies that our model can learn semantic content more effectively. However, there is still a gap between our
self-supervised feature visualization and supervised discriminative feature representation to some extent. For example, on the samples of table lamps and chairs, the self-supervised feature differentiation between different components is not obvious.

To fairly compare self-supervised features with other state-of-the-art methods, we employ various self-supervised training schemes based on the same RSCNN backbone and present the t-SNE [58] visual comparison of self-supervised features under these different schemes. Specifically, we use the ModelNet40 test dataset as visual samples, input them into different models to extract self-supervised global feature of each point cloud, and then apply t-SNE clustering for visualization. The detailed comparison is shown in Fig. 8. It can be seen that our CP-Net has better discrimination of self-supervised features of different categories, which is more conducive to the convergence of downstream tasks.

VII. CONCLUSION

We propose a dual-branched CP-Net for point cloud self-supervised learning. Equipped with contour-perturbed augmentation module and dual-branch consistency loss, our CP-Net is designed to effectively preserve the discriminative representation of easily-learned structural contour information while simultaneously extracting the semantic content information that is typically challenging for self-supervised learning approaches. Extensive experiments have shown the performances, transferability and decent robustness of our CP-Net.

REFERENCES

[1] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum PointNets for 3D object detection from RGB-D data,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 918–927.
[2] J.-Y. Chen, C.-H. Lin, P.-C. Hsu, and C.-H. Chen, “Point cloud encoding for 3D building model retrieval,” IEEE Trans. Multimedia, vol. 16, pp. 337–345, 2014.
[3] J. Li et al., “Spatio-temporal attention networks for action recognition and detection,” IEEE Trans. Multimedia, vol. 22, pp. 2990–3001, 2020.
[4] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, “Graph-based static 3D point clouds geometry coding,” IEEE Trans. Multimedia, vol. 21, pp. 284–299, 2019.
[5] Y. Xu, W. Zhang, F. Yang, and G. Li, “Rate-distortion optimized geometry compression for spinning LiDAR point cloud,” IEEE Trans. Multimedia, vol. 25, pp. 2993–3005, 2023.
[6] R. Q. Charles, H. Su, M. Kaichun, 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. 77–85.
[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 5105–5114.
[8] M. Xu, Z. Zhou, and Y. Qiao, “Geometry sharing network for 3D point cloud classification and segmentation,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 12500–12507.
[9] 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. 90–105.
[10] Y. Wang et al., “Dynamic graph CNN for learning on point clouds,” ACM Trans. Graph., vol. 38, no. 5, pp. 1–12, 2019.
[11] 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. 8887–8896.
[12] H. Thomas et al., “KPCConv: Flexible and deformable convolution for point clouds,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 6411–6420.
[13] M. Gadelha, R. Wang, and S. Maji, “Multiresolution tree networks for 3D point cloud processing,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 105–122.
[14] 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.
[15] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov, “Point cloud GAN,” 2018, arXiv:1810.05795.
[16] X. Liu, Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker, “L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention,” in Proc. 27th ACM Int. Conf. Multimedia, 2019, pp. 989–997.
[17] 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.
[18] 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. 5375–5384.
[19] M. Shoef, S. Fogel, and D. Cohen-Or, “PointWise: An unsupervised point-wise feature learning network,” 2019, arXiv:1901.04544.
[20] 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.
[21] M. Zhang, P. Kadam, S. Liu, and C.-C. J. Kuo, “Unsupervised feedforward feature (UFF) learning for point cloud classification and segmentation,” in Proc. IEEE Int. Conf. Vis. Commun. Image Process., 2020, pp. 144–147.
[22] K. Hassani and M. Haley, “Unsupervised multi-task feature learning on point clouds,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 8159–8170.
[23] M. Xu et al., “Learning geometry-disentangled representation for complementary understanding of 3D object point cloud,” in Proc. AAAI Conf. Artif. Intell., vol. 35, no. 4, 2021, pp. 3056–3064.
[24] M. Zaheer et al., “Deep sets,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 3394–3404.
[25] Y. Li et al., “PointCNN: Convolution on X-transformed points,” in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., 2018, pp. 828–838.
[26] 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. 9613–9622.
[27] P. Bachman, R. D. Hjelm, and W. Bachwaltcher, “Learning representations by maximizing mutual information across views,” Adv. Neural Inf. Process. Syst., vol. 32, 2019.
[28] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1422–1430.
[29] C. Doersch and A. Zisserman, “Multi-task self-supervised visual learning,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2070–2079.
[30] O. Henaff, “Data-efficient image recognition with contrastive predictive coding,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 4182–4192.
[31] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” presented at the Comput. Vis.–ECCV 2020: 16th Eur. Conf., Glasgow, UK, Aug. 23–28, 2020, pp. 776-794.
[32] S. Xie et al., “PointContrast: Unsupervised pre-training for 3D point cloud understanding,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 574–591.
[33] 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. 7161–7170.
[34] Y. Chen et al., “Shape self-correction for unsupervised point cloud understanding,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 8362–8371.
[35] C. Zhang, H. Chen, H. Wan, P. Yang, and Z. Wu, “Graph-PBN: Graph-based parallel branch network for efficient point cloud learning,” Graphical Models, vol. 119, 2022, Art. no. 101120.

[36] J. Li, B. M. Chen, and G. H. Lee, “SO-Net: Self-organizing network for point cloud analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9397–9406.

[37] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Kolm, “Point transformer,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 16259–16268.

[38] R. Klokov and V. Lempitsky, “Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 863–872.

[39] J. Li, B. M. Chen, and G. H. Lee, “SO-Net: Self-organizing network for point cloud analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9397–9406.

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

[41] Q. Huang, W. Wang, and U. Neumann, “Recurrent slice networks for 3D segmentation of point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 2626–2635.

[42] H. Fan, H. Su, and L. J. Guibas, “A point set generation network for 3D object reconstruction from a single image,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2462–2471.

[43] 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.

[44] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” Adv. Neural Inf. Process. Syst., vol. 32, 2019.

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[46] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. 32nd Int. Conf. Mach. Learn., 2015, pp. 448–456.

[47] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, “PointHop: An explainable machine learning method for point cloud classification,” IEEE Trans. Multimedia, vol. 22, pp. 1744–1755, 2020.

[48] L. Yi et al., “A scalable active framework for region annotation in 3D shape collections,” ACM Trans. Graph., vol. 35, 2016, Art. no. 210.

[49] A. X. Chang et al., “ShapeNet: An information-rich 3D model repository,” 2015, arXiv:1512.03012.

[50] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, “3DmFV: Three-dimensional point cloud classification in real-time using convolutional neural networks,” IEEE Robot. Automat. Lett., vol. 3, no. 4, pp. 3145–3152, Oct. 2018.

[51] 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. 19313–19322.

[52] Z. Wu et al., “3D ShapeNets: A deep representation for volumetric shapes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1912–1920.

[53] 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.

[54] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, pp. 273–297, 1995.

[55] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” Nature, vol. 401, pp. 788–791, 1999.

[56] 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. 620–638.

[57] 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.

[58] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 11, pp. 2579–2605, 2008.