Self-supervised Point Cloud Representation Learning via Separating Mixed Shapes

Chao Sun, Zhedong Zheng, Xiaohan Wang, Mingliang Xu and Yi Yang, *Senior Member, IEEE*

***Abstract*—The manual annotation for large-scale point clouds costs a lot of time and is usually unavailable in harsh real-world scenarios. Inspired by the great success of the pre-training and fine-tuning paradigm in both vision and language tasks, we argue that pre-training is one potential solution for obtaining a scalable model to 3D point cloud downstream tasks as well. In this paper, we, therefore, explore a new self-supervised learning method, called Mixing and Disentangling (MD), for 3D point cloud representation learning. As the name implies, we mix two input shapes and demand the model learning to separate the inputs from the mixed shape. We leverage this reconstruction task as the pretext optimization objective for self-supervised learning. There are two primary advantages: 1) Compared to prevailing image datasets, *e.g.*, ImageNet, point cloud datasets are *de facto* small. The mixing process can provide a much larger online training sample pool. 2) On the other hand, the disentangling process motivates the model to mine the geometric prior knowledge, *e.g.*, key points. To verify the effectiveness of the proposed pretext task, we build one baseline network, which is composed of one encoder and one decoder. During pre-training, we mix two original shapes and obtain the geometry-aware embedding from the encoder, then an instance-adaptive decoder is applied to recover the original shapes from the embedding. Albeit simple, the pre-trained encoder can capture the key points of an unseen point cloud and surpasses the encoder trained from scratch on downstream tasks. The proposed method has improved the empirical performance on both ModelNet-40 and ShapeNet-Part datasets in terms of point cloud classification and segmentation tasks. We further conduct ablation studies to explore the effect of each component and verify the generalization of our proposed strategy by harnessing different backbones.**

***Index Terms*—Point cloud, Pre-training, Self-supervised learning, Graph Neural Network, Representation learning.**

I. Introduction

POINT clouds, as one perception of the 3D world, have wide applications, *e.g.*, autonomous driving [2]–[4] and virtual reality [5]. With recent developments of deep learning technology, deep learning-based approaches [6]–[9] on point cloud processing gradually surpass traditional statistical processing methods [5], [10]. However, these deeply-learned models are data-hungry. Although the point cloud data can be collected by laser sensors and other equipment, the point-wise point cloud annotation still costs a lot of human resources and unaffordable expenses [1], [11]–[13]. In this work, we argue that pre-training is one potential way to relieve data limitation. We are inspired by successes in image recognition, where the pre-training model on ImageNet can efficiently adapt to various computer vision tasks, including image segmentation [14]–[17] and image retrieval [18]–[22]. We also note that pre-training on point cloud is still under-explored. To fill this gap, in this work, we resort to model pre-training via self-supervised learning to reduce the demand for annotated data.

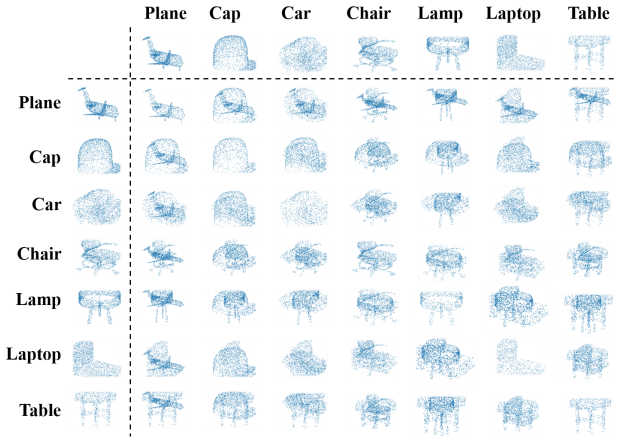


Fig. 1. Visualized results of mixed point clouds. We select seven types of point clouds from ShapeNet-Part [1]. Each row and column corresponds to the original point clouds and the intersection corresponds to the mixed point cloud. M denotes the number of samples in the training set. **In theory, we can generate $O(M \times M)$ different point clouds by sampling various cloud pairs, resulting in a much larger online generated training sample pool.**

Most existing works [23], [24] focus on designing pretext tasks by exploring the spatial characteristics of the single point cloud, which does not solve the problem of data limitation in current open-source datasets [1], [11]. To address this limitation, we introduce a simple solution Mixing to combine two point clouds as a new mixed point cloud. Benefiting from the batch training of deeply-learned models, each point cloud can be mixed with a large number of point clouds during the whole training process to enlarge the training data pool. The proposed approach is memory and resource-efficient, which can directly process point clouds in a single pass. Compared with texture, which is important in 2D images, shape information is critical for point cloud representation. The human can easily figure out the two 3D models, *i.e.*, chair and airplane, in mixed point clouds, according to the prior knowledge of shape (see Fig. 1). Inspired by the work on 2D clothes changing [25], we propose a novel self-supervised task, Mixing and Disentangling (**MD**) for point clouds. As

Chao Sun, Xiaohan Wang and Yi Yang are with School of Computer Science, Zhejiang University, Zhejiang, China. E-mail: c_sun@zju.edu.cn, xiaohan.wang@zju.edu.cn, yangyics@zju.edu.cn

Zhedong Zheng is with Sea-NExT joint lab, School of Computing, National University of Singapore, Singapore 118404. E-mail: zdzheng@nus.edu.sg

Mingliang Xu is with Zhengzhou University, 100 Kexue Ave, Zhongyuan District, Zhengzhou, Henan, China. Email: iexumingliang@zzu.edu.cn

This work is supported by National Key R&D Program of China (No.2020AAA0108800) and Fundamental Research Funds for the Central Universities (No. 226-2022-00087).

the name implies, our intuition underpinning the proposed **MD** is encouraging the network to mine geometric knowledge, *i.e.*, shape-aware features, and such knowledge can be easily transferred to various tasks. As shown in Fig. 2, the pretext task of self-supervised learning is designed to separate original point clouds from the mixed one. Given two input point clouds A of N points and B of N points, the mixing progress outputs mixed point cloud C of N points sampled from A and B. In particular, we adopt a random sampling strategy in each input point cloud and select $\frac{N}{2}$ points in A and B respectively. After the sampling stage, we concatenate 2 sampled $\frac{N}{2}$ points together as a new point cloud C with N points, and we also disrupt the indices of these points again to prevent over-fitting the point order. The disentangling process demands the model to mine the key points of both two original point clouds from the mixed point cloud. The decoder output 2 tensors shape of $N \times 3$, and each reconstructs one input point cloud. In particular, the decoder disentangles a specified point cloud based on its conditional coordinates shown in Fig. 3 as the decoder input. For instance, two generated point clouds shown in Fig. 2 demands two decoding process (the model should forward twice while the sum loss of two results backward once), one for the plane, and another for the chair. Briefly, our contributions are as follows:

- Different from most existing pre-training works on image recognition, there do not exist large-scale datasets like ImageNet [26] for 3D point cloud pre-training. To address this problem, we leverage the mixing process to generate large-scale mixed data for 3D point cloud training;
- Inspired by the human ability to figure out two shapes from one mixed object, we propose a new self-supervised learning method on the point cloud, called Mixing and Disentangling (**MD**) to learn the geometric prior knowledge without the requisite of annotations;
- As one minor contribution, we implement one basic pipeline to verify the effectiveness of the proposed self-supervised learning strategy. It contains one encoder with the learnable aggregation function and one instance-adaptive decoder, to learn from the mixed point cloud and conduct the disentanglement.
- Albeit simple, experimental results on two benchmarks, *i.e.*, ModelNet-40 [11] and ShapeNet-Part [1], show that the self-supervised learning model can effectively and efficiently improve the accuracy of classification and segmentation tasks by the pre-training and fine-tuning paradigm. Self-supervised learning on the point cloud also can reduce the network dependence on labeled data.

The rest is organized as follows. We introduce existing works in Section II. Section III describes the proposed method with details. Quantitative and qualitative experiments verify the effectiveness of **MD** pre-training in Section IV, followed by the conclusion in Section V.

II. Related Work

A. 3D Point Cloud Processing

In recent years, deep learning-based approaches [27]–[30] facilitate the development of point cloud processing. Due to the irregular and disordered characteristics of point clouds, the traditional neural networks in the 2D field, *i.e.*, convolutional neural network (CNN), cannot be directly applied to point clouds. Therefore, researchers resort to various methods as follows. **(1).** Some methods are voxel-based. The voxel-based methods [11], [31] voxelize the point cloud to obtain a cube with a grid structure, which can be directly processed by 3DCNN. However, the accuracy of this method is limited by the resolution during voxelization. **(2).** Some methods are projection-based. The projection-based methods [32]–[34] project the point cloud on multiple planes. The existing 2D CNN is used to extract the projected point cloud features on different planes. After fusing multiple features, the descriptor of the point cloud is obtained, which can be used in downstream tasks. **(3).** Some methods are point-based. PointNet [35] proposed a neural network structure that directly processes the raw point cloud data. PointNet++ [36] further considers both the global feature and the local feature. Point-based approaches can be divided into the graph-based method and the convolution-based method. The graphed-based method treats the point set as a graph [37]–[39]. The convolution-based method [40], [41] introduces the concept of the convolution kernel in 2D CNN to the point cloud. Taking one step further, recent methods [42], [43] also explore attention and transformer structures, yielding competitive performance.

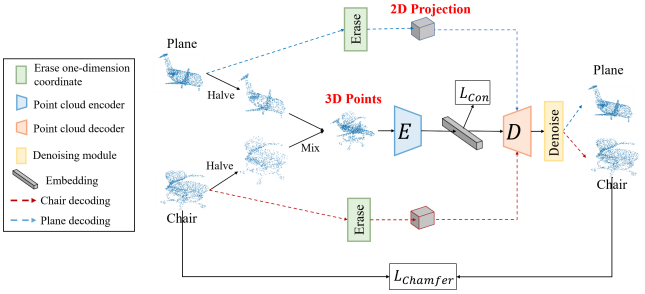


Fig. 2. A schematic overview of our method. The two original point clouds are mixed and input into encoder E to obtain the embedding vector of the mixed point cloud. A coordinate extracting operation, $Erase$, is used to extract partial information from the original point cloud for instance disentanglement. Then the embedding vector and instance information are input to the decoder D and the denoising module to generate the original point cloud, as shown in the red line and the blue line, respectively. The reconstruction loss, *i.e.*, $L_{Chamfer}$, between the generated point cloud and the original point cloud is used as the self-supervision during training.

B. Self-Supervised Learning

Self-supervised learning is a machine learning paradigm that uses the structural information of the data itself to generate labels required for training. **(1).** Some methods are based on the transformation invariance. In particular, image rotations [44], image jigsaw puzzle [45], random erasing [46] , adaptive exploration [47] have been shown to be helpful. **(2).** Some methods are based on the context of data. This type of method learns the feature by image generation, *e.g.*, image colorizing [48], image inpainting [49] and super-resolution methods [50]. [51] adopts a context reconstruction loss for self-supervised temporal modeling. Generative Adversarial Networks (GANs) are used for image generation [25], [52], [53]. **(3).** Other methods are based on contrastive learning. The

type of method mines the structure information from data by designing loss functions. These methods define positive and negative samples firstly and then conduct metric learning by narrowing the distance between positive pairs and widening the distance between negative pairs. For instance, CPC [54] introduces anti-noise estimation, while InfoNCE [54] proposes a loss function based on mutual information. Following the spirits of these works, researchers [55]–[59] further design various model architectures and loss functions. Even worse than 2D image datasets, the annotation of 3D point clouds is usually unavailable due to the annotation time costs and human expenses. In recent years, there are several self-supervised learning methods [60]–[63], which explore the point cloud pre-training. For instance, Sauder *et al.* [23] propose space reconstruction, PointContrast [24] is based on multi-view contrastive learning, and DepthContrast [64] leverages contrastive learning which can handle different input data formats. However, these methods usually require a relatively large dataset, such as Scannet [13], and do not solve the data limitation in real-world tasks. To fill this gap, we adopt a simple point cloud mixing strategy to enlarge the small dataset. Some works have already applied the mixing progress to supervised learning as a data augmentation strategy. For example, PointMixup [30] proposes the shortest path interpolation with EMD (Earth Move Distance), while PointCutMix [65] searches the one-to-one correspondence by EMD. Besides, RSMix [66] proposes to replace some points by extracting subsets from another point cloud. Differently, our method focuses on self-supervised training of point clouds with the disentangling task while these works only apply the mixing progress as a data augmentation in supervised learning. From another point of view, our method is complementary to the existing work. We can apply our method to conduct self-supervised pre-training on the unlabeled dataset and fine-tune the model on the downstream task with the above data augmentation strategies. **The main difference between the proposed method and existing works are: 1). We generate more "within-distribution" training point clouds via Mixing, which have the same mean and variance with the original data. This process largely increases the training pool and lets the model "see" more inlier variants during training. 2). The proposed pretext task is more challenging. Separating original objects from mixed point clouds is more difficult than extracting key points of a single object. 3). We show that the learned models are competitive in various downstream tasks and settings.**

III. Methodology

In this section, we illustrate the pipeline and the basic model, which simply contains two modules, *i.e.*, encoder, and decoder. Then we explain the training strategy of Mixing and Disentangling, followed by the discussion on components.

A. Overview

Our model is to recover two 3D input point clouds according to the corresponding 2D projections from the mixed point cloud. This challenging task motivates the model to learn shape-related geometric knowledge, which benefits downstream tasks. As shown in Fig. 2, the structure is mainly composed of one encoder and one decoder. The encoder is to extract the embedding vector of the mixed point cloud. Given the output features from the encoder, the decoder is used to restore the original point clouds based on the embedding vector and the conditional input from partial coordinates of the original point cloud (see Fig. 3). After pre-training, the encoder can be utilized to extract discriminative features for subsequent tasks, *e.g.*, classification, and segmentation. As shown in Fig. 4, the black arrows denote the fine-tuning workflow for recognition, while the blue arrows are the segmentation workflow. We denote the input point cloud as a set of points $S = \{s_1, s_2, ..., s_n\}$, $s_i \in \mathbb{R}^3$ and each point has 3D coordinates. However, the point cloud does not have a regular spatial structure as 2D images, we can not apply the convolutional neural network (CNN) directly. Therefore, we apply K-Nearest Neighbor (KNN) algorithm to construct the graph structure $\mathcal{G}$ in the feature space. The KNN algorithm based on the feature inputs can efficiently and effectively find two points with the most similar semantics, *i.e.*, the points on the two legs of the chair with similar semantics. The graph can be expressed as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the edge set.

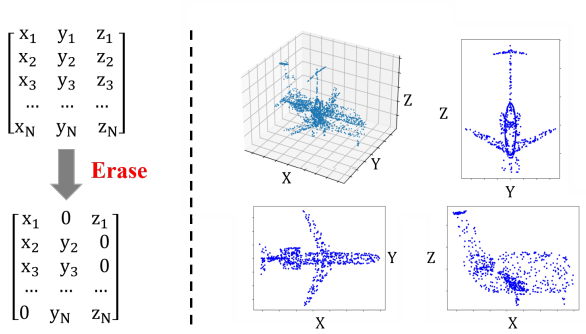


Fig. 3. The left part represents a point cloud of N points, and each point has 3D coordinates. After erasing, the random one-dimensional coordinate of each point is set to zero which intends to make the pre-training process more challenging. The right part is the visualization of an erased point cloud of the plane and its three-view drawing after erasing. The erasing process equals randomly set about $\frac{1}{3}N$ points to the corresponding projection surface.

B. Encoder

As shown in Fig. 4, the backbone of the encoder is composed of three EdgeConv layers and three learnable aggregation (LA) layers. Specifically, we deploy EdgeConv layers to leverage the graph information. Given the input of $N \times H_{in}$ and the corresponding graph $\mathcal{G}$, the EdgeConv layer performs feature transformation on the N points and outputs the updated point features of $N \times H_{out}$. The input features can be represented as $X = \{x_1, x_2, ..., x_n\}$, $x_i \in \mathbb{R}^{H_{in}}$ where x_i represents the feature of point i. The relation feature $r_i^k \in \mathbb{R}^{N \times H_{out}}$ can be formulated via an EdgeConv layer as:

$$r_i^k = MLP(concat(x_i, x_i - x_k)), k : (i,k) \in \mathcal{E}, k \neq i, \quad (1)$$

explicitly encodes the relationship between x_i and x_k, where x_k is one neighbor of x_i and MLP notes a linear function with ReLU [67]. To consider all the neighbor of x_i, we apply Learnable Aggregation (LA) to the local neighbor relation r_i^k.

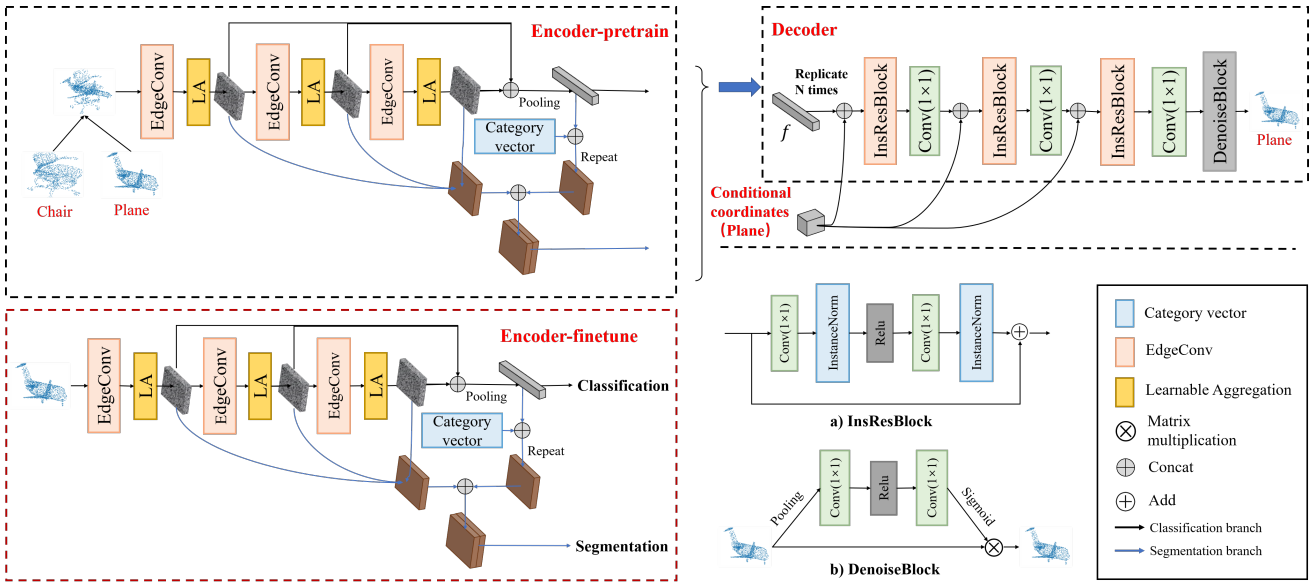


Fig. 4. **Pipeline.** The model is mainly composed of one encoder (left) and one decoder (right). **Encoder:** The encoder takes N points as input and aggregates the special features of each point and its corresponding points in the neighborhood at an EdgeConv layer. The classification network (top branch) contains three EgdeConv layers and three learnable aggregation (LA) layers. The output of the encoder is the embedding vector f of the input point cloud. In the segmentation network, to concatenate with the point-wise feature from the shallow layers, the corresponding embedding vector f is repeated N times for point segmentation, where N is the number of points. The feature maps from shallow layers are concatenated with the embedding vector to obtain the multiple-scale feature map as the output of the encoder. Different from the classification branch, the category vector representing the type of point cloud is fused into the embedding vector for the part segmentation. **Decoder:** The decoder is composed of Instance-aware Residual Block (InsResBlock), 1×1 Convolution Layers and one DenoiseBlock to refine the final reconstructed point cloud. The structure of InsResBlock and DenoiseBlock is shown in a) and b). The random two-dimensional coordinates of the original point cloud are used as conditional information to be fused with the backbone features in three different layers.

We notice that the current widely-used aggregation methods applied in the point cloud include both max pooling and average pooling. It is usually challenging to decide which pooling function should be used. As one minor contribution, we involve Learnable Aggregation (LA), which is a learnable pooling layer, and let the model learn the adaptive weight:

$$LA(x_i) = \alpha \times \underset{(i,k)\ \in \mathcal{E}, k \neq i}{max}(r_i^k) + (1-\alpha) \times \underset{(i,k)\ \in \mathcal{E}, k \neq i}{avg}(r_i^k), \quad (2)$$

where the interpolation ratio α is a learnable parameter of the network. Finally, the encoder backbone outputs the point cloud feature map by combining intermediate features from multiple layers as [68] to enhance the representation capability.For the classification downstream tasks, we add the max pooling layer to compress the feature map to the vector, which is sent to the decoder. When fine-tuning the downstream classification, we simply add one linear classifier to this learned vector. As for the segmentation downstream task, which is a dense point-level task, the vectorization compromises the spatial representation. Therefore, we concatenate the intermediate feature maps to keep the same structure for fine-tuning. We also add the extra category vector to specify part predictions, because different objects contain different semantic parts. For instance, the plane contains the aircraft nose, plane body, plane tail, and wings while the chair has legs, the seat, and the back. When fine-tuning, similarly, we only need to add one linear classifier to the learned feature map.

Discussion. By using the proposed mixing mechanism, we can generate more "within-distribution" point cloud samples, which largely enriches training samples. The previous methods [23] use ShapeNet-Part for pre-training and it can only generate $O(M)$ training samples, where M is the number of point clouds. In theory, our method can generate $O(M \times M)$ different point clouds by sampling various cloud pairs, resulting in a much larger online generated training sample pool. The encoder can mine more prior knowledge from data, which improves downstream tasks. Besides, to extract the features more efficiently, we propose a new feature aggregation method LA in the point cloud. Most existing works [36], [39] apply max pooling to aggregate the neighboring features, which leads to much information loss. LA leverages a learnable parameter to dynamically adjust the aggregation method, which can effectively retain the local structure. More ablation studies on LA are provided in the experiment.

C. Decoder

We propose an instance-adaptive decoder, which can restore the input point cloud from the embedding vector according to the conditional coordinates adaptively. The decoder treats the embedding vector as the structure representation of the mixed point cloud and aims to disentangle key points from different objects. In our network, the decoder is utilized as a selective filter, which gradually enhances the feature of points belonging to one object and weakens the feature of the rest points of another object. After several transformations, the feature mainly contains the key points of one object, which can be transformed to the recovered point cloud by the final linear layers. As shown in Fig. 4, the decoder is composed of Instance-aware Residual Block (InsResBlock) [69], 1×1 Convolution Layers and one denoising module *i.e.*, DenoiseBlock. There are two inputs to the decoder: the mixed point cloud embedding vector f, the conditional coordinates C_{coord}

of size $N \times 3$. To recover the original point cloud from the mixed feature, the 2D projection C_{coord} of each point cloud is given during training. We use the random two-dimensional coordinates of the original point cloud as partial information. As shown in Fig. 3, we randomly erase [46] one of the three coordinates of each point cloud. For each point in a point cloud, the erased dimensions are randomly generated instead of taking a certain dimension. Since the direct erasing of a certain dimension will cause the point cloud to change from $N \times 3$ to $N \times 2$, the neural network cannot distinguish the specific dimensions of the remaining two coordinates. For instance, the model can not foreknow the two coordinates are (X,Y) or (X,Z) or (Y,Z). We choose to set the erasing coordinates to zero instead of directly deleting the dimension. Instance-aware Residual Block (InsResBlock) is a basic module used to disentangle the mixed feature from f according to the conditional coordinates C_{coord}. The structure of InsResBlock is shown in Fig. 4 (a), which is composed of two convolution layers and instance normalization [70]. A pair of InsResBlock and 1×1 convolution layer is regarded as a basic decoding unit. Our decoder D is built with three sequential basic decoding units, which are connected sequentially to reduce the feature dimension layer by layer and finally output the generated point cloud of 3 coordinate channels. For each basic encoding unit, we concatenate the conditional coordinates and the feature map from the previous unit to specify the reconstruction target. To refine the generated point cloud, we further introduce a denoising module as the last unit of decoder. As shown in Fig. 4 (b), the denoising module is implemented via a self-attention manner. The module leverages the context information between neighbor points to assign different weights for adjacent points and noisy points, which can ensure each point will be optimized with the global information. Given the noisy point cloud $\widetilde{s}$, the denoising module outputs the denoising point cloud $\hat{s} = Denoise(\widetilde{s}), \widetilde{s} = D(f, C_{coord})$, where D represents the decoder. In particular, given the tensor $\widetilde{s}$ of $N \times 3$, we perform average pooling in the dimension of N to obtain an initial weight of $N \times 1$ for every point. We apply two convolution layers to aggregate the global information from all points and normalize the weight of each key point via the sigmoid function. Then the weight is multiplied by the original point cloud to obtain the denoised point cloud. The multiplication allows every point to refer to the neighbor points and ignore outlier points with low scores.

Discussion. The choice of conditional disentanglement. The choice of C_{coord} has a large impact on the reconstruction results and downstream tasks. If the decoder can only obtain little original point cloud information from C_{coord}, the reconstruction effect of disentanglement will be poor. On the contrary, if the decoder obtains too much information from the original point cloud, the decoder will restore the original point cloud based on the conditional information directly. Then the encoder cannot be well trained to fully mine the geometric information. We think this choice is still open and in this paper, we observe that using random two-dimensional coordinates is a balanced choice, which guarantees the completeness of the disentanglement and the challenge of the pre-training task.

D. Optimization

As the target of the proposed Mixing and Disentangling task (MD), we supervise the model training via reconstruction. Besides, we add the intermediate embedding loss for self-supervision. There are many candidates for embedding loss. In this work, without loss of generality, we select contrastive loss to verify the compatibility of the proposed method instead of pursuing the best loss. Next, we describe the two objectives.

Reconstruction loss. We apply Chamfer distance to measure the distance between the original point clouds s and the reconstructed one $\hat{s}$. As Eq. 3 shows, the distance is a symmetric function. Since we could not know the exact matching between two point sets, the Chamfer distance accumulates one sub-optimal but effective distance between the closest points in both the original and reconstructed point cloud. The reconstruction loss can be formulated as:

$$L_{Chamfer} = \frac{1}{|\hat{s}|} \sum_{\hat{p} \in \hat{s}} \min_{p \in s} ||p - \hat{p}||_2 + \frac{1}{|s|} \sum_{p \in s} \min_{\hat{p} \in \hat{s}} ||\hat{p} - p||_2, \tag{3}$$

where p denotes the point position in s, and $\hat{p}$ denote the points in $\hat{s}$. $||\cdot||_2$ denotes the L2 distance between two points and $|\cdot|$ denotes the number of points. Specifically, we apply the Chamfer distance as the reconstruction loss on s and $\hat{s}$.

Embedding loss. Inspired by contrastive loss [71], we introduce it to our total loss function, which intends to widen the distance between different categories. In particular, since we apply a mix strategy, we view every sample as one single category. Following the existing practise, *e.g.*, Instance loss [72] and MoCo [56], we encourage that these point clouds should have different embeddings and deploy metric as an optimization objective. Given a mini-batch of B different mixed point clouds, we construct $B \times B$ pairs of samples by calculating the distance G_{ij} between every pair (i and j). The basic contrastive loss can be formulated as:

$$L_{Con} = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} [y {G_{ij}}^2 + (1-y) max(0, 1 - G_{ij})^2], \tag{4}$$

We can transfer the distance into the similarity format. Considering $Q_{ij} = 1 - G_{ij}$, Eq. 4 can be rewritten as:

$$L_{Con} = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} [y(1 - Q_{ij})^2 + (1-y) max(0, Q_{ij})^2], \tag{5}$$

where $y = 1$ only if $i == j$, otherwise $y = 0$. In practise, we adopt the cosine similarity $Q_{ij} = \frac{f_i \cdot f_j}{\|f_i\| \|f_j\|}$. Following the previous self-supervised work [56], we ignore the rare case that i and j belong to the cloud of the same mixed category. Moreover, we transfer Eq. 5 to the matric format for better efficiency, and adopt L1 loss, which is relatively stable. Considering $Q_{ij} \in [-1, 1]$, we optimize the distance between $\frac{Q+1}{2}$ and an identity matrix I, which can be formulated as:

$$L_{Con} = |\frac{Q+1}{2} - I|. \tag{6}$$

Finally, to optimize parameters in both encoder and decoder, we deploy the total loss as follows:

$$L_{total} = L_{Chamfer} + \lambda L_{Con}, \tag{7}$$

where λ is the weight to control Contrastive loss. Actually, Contrastive loss is an optional choice. In the ablation studies, we also study the effect of Contrastive loss by setting $\lambda = 0$.

TABLE I
RESULTS FOR OUR BASELINE AND BASELINE + PRE-TRAINING. WE CARRY OUT TWO EXPERIMENTS OF CLASSIFICATION AND SEGMENTATION ON MODELNET-40 AND SHAPENET-PART RESPECTIVELY AND COMPARE THE ACCURACY WITH SEVERAL COMPETITIVE APPROACHES. THE BASELINE MODEL WITH PRE-TRAINING ACHIEVES THE HIGHEST ACCURACY.

Methods	Publication	ModelNet-40 *Classification*		ShapeNet-Part *Segmentation*
		OA (%)	mA (%)	mIoU (%)
3DShapeNets [11]	CVPR'15	84.7	77.3	-
VoxNet [31]	IROS'15	85.9	83.0	-
PointNet [35]	CVPR'17	89.2	86.0	83.7
PointNet++ [36]	NeurIPS'17	91.9	-	85.1
SpecGCN [73]	ECCV'18	91.5	-	-
PCNN by Ext [74]	SIGGRAPH'18	92.2	-	85.1
DGCNN [39]	TOG'19	92.9	90.2	85.1
Point Trans. [43]	ICCV'21	93.7	90.6	86.6
PAConv [41]	CVPR'21	93.9	-	86.1
CurveNet [75]	ICCV'21	94.2	-	86.8
Ours (from scratch)	-	92.74	89.88	85.27
Ours (pre-training)	-	**93.39**	**90.26**	**85.50**

OA: Overall Accuracy; mA: Mean Class Accuracy; mIoU: mean IoU

IV. EXPERIMENT

A. Implementation Details

Datasets. (1) ModelNet-40 [11] is sampled from the mesh surfaces of CAD models, containing 40 categories, 12,311 models. 9,843 models are used for training and 2468 are reversed for testing. Each point cloud contains 2048 points and the coordinates of all points are normalized into the unit sphere. Following existing works [39], [76], we sample 1024 points from each object and augment the data by randomly scaling objects and perturbing point locations. (2) ShapeNet-Part [77] is sampled on the CAD models, having a total of 16,881 point clouds, of which 12,137 point clouds are used for training, 1870 point clouds are used for verifying, and 2874 point clouds are used for testing. Each point cloud is composed of 2048 points, and the coordinates of all points are normalized into the unit sphere. We sample 1024 points from each object. Each point has 4 attributes including the 3D coordinates of each point and the category label of each point. (3) S3DIS [78] is a real-scan dataset composed of six large scale indoor areas with 271 rooms. Each room is split with $1m \times 1m$ area into blocks. We sample 4096 points from each block. Each point has 9 attributes including 3D coordinates, RGB channels and normalized 3D coordinates of each point.

Implementation. The pre-trained model is trained with Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.99$) of a minibatch of 12 for 200 epochs. The initial learning rate is set to 1e-4. We gradually decrease the learning rate via the cosine policy [79]. Following existing works [39], [76], we adopt position jittering as data augmentation and set $k = 20$ for the KNN algorithm to build the dynamic graph. In the classification network, to fairly compare with other works, we deploy four EdgeConv layers and the channel number is {64, 64, 128, 256} by default. In the segmentation network, we adopt three EdgeConv layers and the channel number is {64, 64, 64}. Two dropouts with 0.5 drop rate are used when fusing the conditional coordinates and the embedding vector. We deploy one Nvidia RTX 3090 by default. The pre-training time is about one day. FLOPs (Floating Point Operations) of the whole model is 2.798 GFLOPs and the number of parameters is 2.070 *M*. The test time is about 0.0019 seconds per point cloud. FLOPs and the number of parameters of the decoder are 0.729 GFLOPs and 0.712 *M*. It is worth noting that the complexity of our model mainly depends on the encoder structure. Our method is open to different encoder choices, which can be selected according to the computing resource. The proposed method can converge smoothly as shown in Fig. 5. In the fine-tuning stage, we add a linear classifier for both classification and segmentation tasks. The segmentation task demands extra category vector (see Fig. 4). We keep a random category vector during pre-training to hold the position and provide the real category vector during fine-tuning.

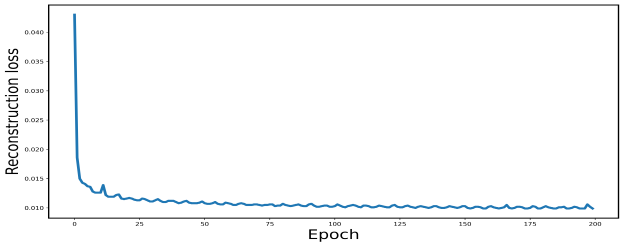


Fig. 5. The loss curve on ModelNet-40 test set during pre-training. We observe that the proposed method can converge smoothly.

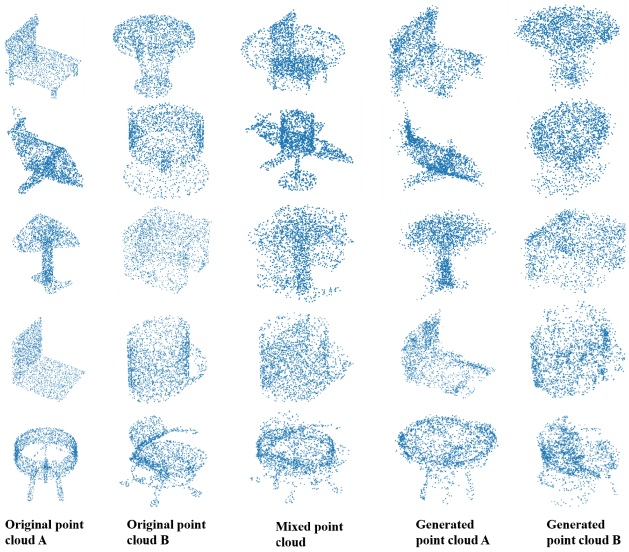


Fig. 6. Visualization of recovered point clouds after disentangling. The experiment is performed on the Shapenet-Part. We observe that the proposed method can successfully disentangle the mixed point cloud into two separate point clouds. In line 5, the chair and the table have a similar structure. There is a hole in the middle of the table and the model tends to fill this hole which makes the disentangling more challenging.

B. Quantitative & Qualitative Results

The pre-training improves the baseline. We evaluate how the pre-training affects the performance in Tab. I. For the classification task, the model is pre-trained and fine-tuned on ModelNet-40. We note that the pre-trained model gets an overall accuracy of 93.39%, which have +0.65% than the baseline (92.74%). Similarly, the mean class accuracy also increases from 89.88% to 90.26% with +0.38% accuracy improvement.

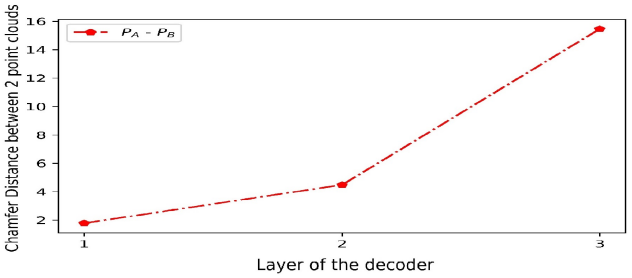
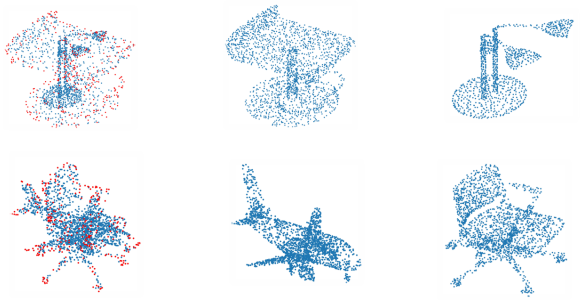


Fig. 7. Visualization of the distance between input point clouds P_A and P_B in 3 layers of the decoder. We observe that the distance between P_A and P_B becomes larger from the shallow layer to the deep layer in the decoder.



a) Key points of the mixed point cloud **b) Original point cloud A** **c) Original point cloud B**

Fig. 8. Visualization of the key points of the mixed point cloud. a) is the point cloud mixed by b) and c), and the key points of a) is indicated by red points. The result shows that our encoder can successfully capture the key points of two original point clouds after pre-training.

For the segmentation task, the model is pre-trained and fine-tuned on ShapeNet-Part. The pre-trained model achieves a competitive mean Intersection-over-Union (mIoU) of 85.50%. Experimental results show that the accuracy of our baseline surpasses three existing works, and pre-training can still bring gains to the accuracy both in classification and segmentation tasks, which verifies the effectiveness of our pretext task.

Visualization of the reconstructed point cloud. (1) As shown in Fig. 6, we visualize original point clouds, mixed point clouds, and generated point clouds on ShapeNet-Part. Our method achieves good reconstruction results, recovering the key geometry. (2) We further track the decoder activation changes during reconstruction. Given one mixed embedding (extracted from mixed P_A and P_B), we visualize the disentangle process via Chamfer distance (see Fig. 7). There are three layers in our decoder and the channel of each layer output is 128, 64, and 3. We normalize the output feature in the channel dimension and divide the Chamfer distance by the channel dimension for distance calculation. We observe that the distance between P_A and P_B becomes larger from the shallow layer to the deep layer in the decoder. It indicates that the decoder follows the conditional information and chooses different features for reconstructing P_A and P_B respectively.

Visualization of key points. We visualize the top 25% key points of the mixed point cloud, which has max activation after the last pooling layer in the encoder (see Fig. 8). The result shows that the key points of a) include the key points of both b) and c), which verifies that the pre-trained embedding learns the salient geometric information of both input point clouds.

TABLE II
THE PERFORMANCE OF OUR METHOD IN THE REAL-SCAN DATASET. WE PRE-TRAIN THE MODEL ON S3DIS AND FINE-TUNE THE MODEL IN 3 DIFFERENT TASKS. THE RESULTS SUGGEST A CONSISTENT IMPROVEMENT.

Methods	ModelNet-40 *Classification*		ShapeNet-Part *Segmentation*	S3DIS *Segmentation*
	OA (%)	mA (%)	mIoU (%)	mIoU (%)
Ours (from scratch)	92.74	89.88	85.27	50.78
Ours *	**92.78**	**90.06**	**85.37**	**51.74**

OA: Overall Accuracy; mA: Mean Class Accuracy; mIoU: mean IoU;
* means pre-training on S3DIS

TABLE III
SEGMENTATION RESULTS ON PARTIAL LABELED DATA. THE PERFORMANCE BOOST IS MORE SIGNIFICANT WHEN LABELED DATA IS LIMITED, VERIFYING OUR INTUITION TO BENEFIT THE REAL-WORLD MODEL TRAINING UNDER THE DATA LIMITATION.

Labeled Data Ratio (%)	Pre-trained	Average Accuracy (%)	mIoU (%)
10		74.02	81.59
10	✓	77.04	82.23
25		80.39	83.76
25	✓	81.11	84.49
50		81.46	84.71
50	✓	83.31	85.04

Visualization of embeddings. To verify the scalability of learned embeddings, we extract embeddings of ShapeNet-Part via the pre-trained model on ModelNet-40. We apply the T-SNE [80] to reduce the dimension of the embedding vector to $\mathbb{R}^2$ for plotting. As shown in Fig. 9, the distance between intra-class samples is small and the distance between inter-class is large. The above result supports the generalization of the pre-trained encoder to unseen point clouds.

C. Ablation Studies and Further Discussion

Performance on the real-scan dataset. We further conduct our pre-training process on S3DIS, a real-scan dataset. S3DIS has 6 areas and we follow existing works [43] using area 5 for testing and others for training. We pre-train our model in the training part and then we fine-tune the pre-trained model on ModelNet-40 for the classification task, on ShapeNet-part for the part segmentation task and on S3DIS for the segmentation task. As shown in Tab. II, our method benefits the performance both on ModelNet-40 and ShapeNet-part. It verifies the scalability of the model pre-trained on the large dataset. Furthermore, in the real-scan dataset, our model with pre-training achieves better performance 51.74% mIoU than the model trained from scratch (50.78%) by a clear margin.

Illustration of "within-distribution" data. The "within-distribution" means that generated point clouds have similar characteristic (*e.g.*, mean and variance) with the original point clouds. Some existing works apply GAN to generate 3D point clouds. This line of works may change the data distribution (mean and variance), since 3D point clouds are typically generated from random Gaussian noise [81]–[83]. Compared with these GAN-based methods, new point clouds generated by our method are more similar to the original ones from a statistics view. We sample 1000 point clouds from ModelNet-40 and plot the distance between each point to its center point. In Fig. 10, we observe that the distribution between the original data and mixed 3D point clouds is almost identical.

TABLE IV
SEMI-SUPERVISED SEGMENTATION RESULTS ON 10% LABELED DATA. WE CAN OBSERVE TWO POINTS: (1) BOTH PSEUDO LABEL-BASED APPROACH AND OUR METHOD CAN IMPROVE THE PERFORMANCE. (2) OUR METHOD CAN BE COUPLED WITH OTHER SEMI-SUPERVISED LEARNING METHODS TO IMPROVE THE PERFORMANCE.

Methods	Pre-trained	Average Accuracy (%)	mIoU (%)
baseline		74.02	81.59
baseline	✓	**77.04**	**82.23**
pseudo label		72.39	82.50
pseudo label	✓	**73.57**	**82.81**

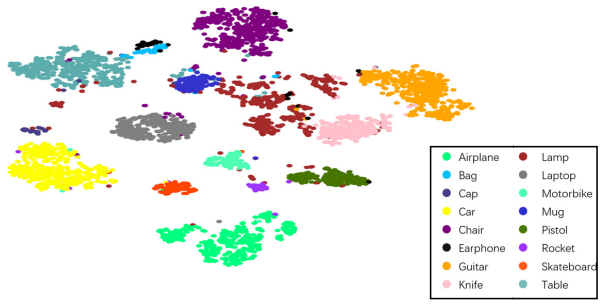


Fig. 9. The visualization of embeddings on ShapeNet-Part in the semantic space via T-SNE [80]. The encoder is pre-trained on ModelNet-40 and is applied to encode the point clouds on ShapeNet-Part directly. The results show that the distance between intra-class samples is small while the distance between inter-class data is large, which verifies that our encoder learns a robust and general structure embedding vector.

Comparisons under the semi-supervised setting. In some real scenarios, labeled data is inadequate. To simulate this harsh situation, we divide the original dataset into two parts A and B, and regard A as unlabeled data. We use $A + B$ to pre-train the model and use B to fine-tune the model. **(1)** We set the percentages of labeled data as 10%, 25%, and 50% respectively. We verify the effect of pre-training by comparing the encoder trained from scratch with the encoder pre-trained on ShapeNet-Part in Tab. III. The results show that as the amount of labeled data increases, the accuracy of segmentation gradually increases. The performance of the pre-trained model generally surpasses that of the model trained from scratch, especially when the labeled data is extremely limited. This verifies that our method can successfully leverage unlabeled data to improve the accuracy of the model with limited annotated data. **(2)** Although our work is not designed for the semi-supervised setting, it can nevertheless be coupled with other semi-supervised learning methods. To verify this point, we build a preliminary semi-supervised baseline based on predicted pseudo labels. We adopt 10% labeled data to train the baseline model. Pseudo labels for the rest 90% unlabeled data are then predicted by this baseline model. We select the unlabeled data with the pseudo label confidence greater than 0.7 and the original 10% labeled data to form the new training set, and then fine-tune the segmentation model. In Tab. IV, the model trained on the pseudo-labeled data improves the mIoU performance from 81.59% to 82.50%, but is suffer from the label noise, which compromises average accuracy. Our pre-training method can be coupled with the pseudo label to relieve the negative impact from noisy annotations. Specifically, initializing the model with our pre-training weight

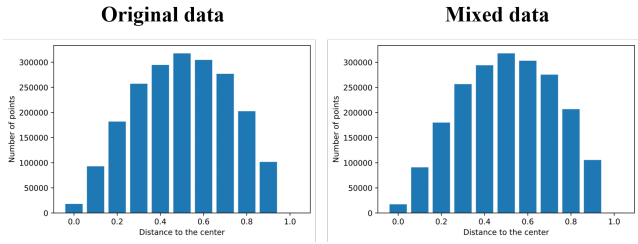


Fig. 10. We sample 1000 point clouds and count the distance from each point to the center of its point cloud. The X-axel means the distance from each point to the center point and the Y-axel means the number of points belonging to each distance interval. We observe that both the mean and std by mixing 3D point clouds are identical to the original data distribution.

TABLE V
WE SELECT 20 CLASSES OF 40 CATEGORIES ON MODELNET-40 FOR PRE-TRAINING, AND THE REST 20 CLASSES FOR FINE-TUNING AND TESTING. THE ACCURACY OF OUR METHOD SURPASSES THE MODEL TRAINED FROM SCRATCH BY A CLEAR MARGIN.

Methods	OA (%)	mA (%)
baseline	95.07	93.82
pre-training	**96.65**	**94.14**

can help to predict more robust pseudo labels. Therefore, semi-supervised methods with our pre-training can arrive at better average accuracy from 72.39% to 73.57% and mIoU from 82.50% to 82.81%. **(3)** We further study category-wise semi-supervised setting. In particular, we select 20 classes of 40 categories on ModelNet-40 for pre-training, and the rest 20 classes for fine-tuning and testing, while the compared baseline model is training and testing only on the remaining 20 classes. We compare the performance of the model initialized randomly (baseline) and initialized with our pre-training method in Tab. V. The accuracy of our method surpasses the model trained from scratch by a clear margin.

Scalability of learned embeddings. We apply a simple linear classifier to directly classify learned embeddings. The classifier consists of a linear layer, a batch norm layer and a linear layer. We arrive at a competitive accuracy of 89.63% with the fine-tuning result of 93.39% on ModelNet-40. Similar to the observation in the large pre-training model CLIP [84], fine-tuning helps the pre-trained model to achieve better performance on specific sub-tasks, while directly using learned features also is a sub-optimal but efficient choice.

Effect of the learnable aggregation. We deploy LA to aggregate neighbor point features. We apply the same network architecture to compare the accuracy of LA, max pooling, and attentive pooling [38]. As shown in Tab. VI, regardless of pre-training, the accuracy of using LA is higher than that of using max pooling or attentive pooling. The observation verifies that LA can better preserve the local context of the point cloud. There are 3 LA modules in our model and concrete numbers of learned α in each layer are 0.6996, 0.6503, and 0.5500 from shallow layer to deep layer in the encoder. It means that in shadow layers, max pooling plays an important role to reduce the redundant points, and in deep layers, average pooling is as important as max pooling to aggregate the feature.

Effect of the contrastive loss. We apply the contrastive loss to narrow the distance between similar samples and widen the

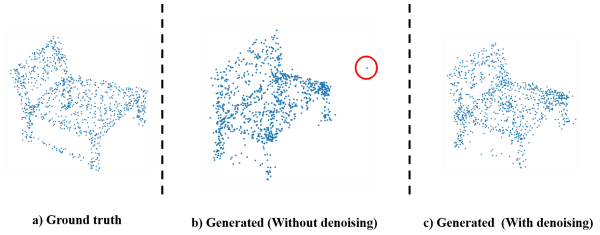


Fig. 11. Visualization results of generated point clouds. The original point cloud is shown in a). b) is the generated point cloud without denoising and c) is the point cloud after denoising. The result shows that the denoise module can help to remove the noise point circled in b).

TABLE VI
ABLATION STUDY ON DIFFERENT COMPONENTS DURING TRAINING. WE VERIFY THE EFFECTS OF LA MODULE AND CONTRASTIVE LOSS.

Pooling Method	Pre-trained	Contrastive loss	mIoU (%)
Max Pooling			85.17
Max Pooling	✓		85.34
Attentive Pooling [38]			85.12
Attentive Pooling	✓	✓	85.36
LA			85.27
LA	✓		85.40
LA	✓	✓	85.50

distance between different samples. The encoder can extract more robust embedding from data, which can improve downstream tasks. We train two pre-trained models with or without the contrastive loss and then fine-tune the two models on ShapeNet-Part, and finally, compare the segmentation accuracy of the two models in Tab. VI. We observe that the contrastive loss can further improve the segmentation performance.

Effect of DenoiseBlock. The visualization results are shown in Fig. 11. a) is the original point cloud, and b) is the generated point cloud obtained by the network without the denoising module. c) is the generated point cloud obtained by the network with the denoising module. We can find that there is a noise point circled in the red circle of b). The denoising module successfully pulls the noise point back to the point cloud by considering the global feature.

Compatibility with different backbone structures. To verify that our pretext task is free from the choice of backbones, we further explore leveraging several widely-adopted architecture as the backbone of the point cloud encoder, such as PointNet++ [36], OGNet [76], PAConv [41], Point Transformer [43] and Point Cloud Transformer [42]. We compare the accuracy of the model trained from scratch and initialized with the pre-trained model. We carry out experiments of classification respectively and results are shown in Tab. VII. We observe consistent improvements with these backbones which verifies the generality of the proposed method.

V. CONCLUSION

In this paper, we propose a new self-supervised learning method, called Mixing and Disentangling (MD), for point cloud pre-training. Different from existing works, we propose to mix the original point clouds in the training set to form "new" data and then demand the model to "separate" the mixed point cloud. In this way, the model is asked to mine

TABLE VII
RESULTS OF USING DIFFERENT BACKBONES. WE CAN OBTAIN ONE CONSISTENT RESULT THAT OUR METHOD IS COMPATIBLE WITH DIFFERENT NETWORK STRUCTURES AND CAN STILL IMPROVE THE ACCURACY OF CLASSIFICATION AND SEGMENTATION TASKS.

Methods	Pre-trained	ModelNet-40 *Classification*	
		OA (%)	mA (%)
Ours	-	92.74	89.88
Ours	ShapeNet-Part	92.79	90.10
Ours	ModelNet-40	**93.39**	**90.26**
PointNet++ [36] *	-	92.07	88.89
PointNet++ * + Ours	ShapeNet-Part	92.19	89.63
PointNet++ * + Ours	ModelNet-40	**92.57**	**89.96**
OGNet* [76]	-	93.23	89.82
OGNet* + Ours	ShapeNet-Part	**93.35**	90.51
OGNet* + Ours	ModelNet-40	93.31	**90.71**
PAConv* (PN) [41]	-	92.50	-
PAConv* (PN) + Ours	ShapeNet-Part	92.70	-
PAConv* (PN) + Ours	ModelNet-40	**92.79**	-
PT* [43]	-	91.47	89.32
PT* + Ours	ShapeNet-Part	92.03	89.50
PT* + Ours	ModelNet-40	**92.07**	**89.58**
PCT* [42]	-	92.71	89.36
PCT* + Ours	ShapeNet-Part	93.07	90.27
PCT* + Ours	ModelNet-40	**93.15**	**90.56**

PT donates Point Transformer. PCT donates Point Cloud Transformer. PAConv (PN) donates using PointNet as the backbone (without voting). OA: Overall Accuracy; mA: Mean Class Accuracy; mIoU: mean IoU; *: We re-implement the model, which achieves a slightly different performance.

the geometric knowledge, *e.g.*, the shape-related key points for reconstruction. To verify the effectiveness of the proposed method, we build a simple baseline to implement our method. We use an encoder to obtain the embedding of the mixed point cloud and then an instance-adaptive decoder is harnessed to separate the original point clouds. During the self-supervised training, the encoder learns the prior knowledge of point cloud structure, which is scalable and can improve the downstream tasks. Experiments show consistent performance improvement through classification and segmentation tasks and verify the effectiveness of our method especially when the number of labeled data is limited. We hope that our approach can benefit the future point cloud works, and take one closer step to the harsh real-world setting, *i.e.*, limited annotations. In the future, we will continue to study the point cloud pre-training method on large-scale datasets, and focus on finding an efficient way to take advantage of the large-scale point data.

REFERENCES

[1] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM TOG*, 2016.
[2] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao, "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
[3] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: a review," *TNNLS*, 2020.
[4] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *TNNLS*, 2020.
[5] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *TPAMI*, 2020.
[6] T. Wan, S. Du, W. Cui, R. Yao, Y. Ge, C. Li, Y. Gao, and N. Zheng, "Rgb-d point cloud registration based on salient object detection," *TNNLS*, 2021.

[7] Y. Chen, H. Li, R. Gao, and D. Zhao, "Boost 3-d object detection via point clouds segmentation and fused 3-d giou-l loss," *TNNLS*, 2020.

[8] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.

[9] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Learning semantic segmentation of large-scale point clouds with random sampling," *TPAMI*, 2021.

[10] X. Huang, G. Mei, J. Zhang, and R. Abbas, "A comprehensive survey on point cloud registration," *arXiv:2103.02690*, 2021.

[11] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.

[12] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv:1702.01105*, 2017.

[13] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[17] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[18] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *ECCV*, 2016.

[19] H. Yao, S. Zhang, D. Zhang, Y. Zhang, and T. Qi, "Large-scale person re-identification as retrieval," in *ICME*, 2017.

[20] Z. Zheng, Z. Liang, and Y. Yi, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *ICCV*, 2017.

[21] L. Zhang, M. Luo, J. Liu, X. Chang, Y. Yang, and A. G. Hauptmann, "Deep top-k ranking for image–sentence matching," *TMM*, vol. 22, no. 3, pp. 775–785, 2019, doi:10.1109/TMM.2019.2931352.

[22] Z. Zheng, T. Ruan, Y. Wei, Y. Yang, and T. Mei, "Vehiclenet: Learning robust visual representation for vehicle re-identification," *TMM*, vol. 23, pp. 2683–2693, 2020, doi:10.1109/TMM.2020.3014488.

[23] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," *NeurIPS*, 2019.

[24] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," in *ECCV*, 2020.

[25] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *CVPR*, 2019.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, 2015.

[27] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun, "3d shape segmentation via shape fully convolutional networks," *Computers & Graphics*, vol. 76, pp. 182–192, 2018.

[28] L. Yi, L. Shao, M. Savva, H. Huang, Y. Zhou, Q. Wang, B. Graham, M. Engelcke, R. Klokov, V. Lempitsky *et al.*, "Large-scale 3d shape reconstruction and segmentation from shapenet core55," *arXiv:1710.06104*, 2017.

[29] Q. Meng, W. Wang, T. Zhou, J. Shen, Y. Jia, and L. Van Gool, "Towards a weakly supervised framework for 3d point cloud object detection and annotation," *TPAMI*, 2021.

[30] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. Snoek, "Pointmixup: Augmentation for point clouds," in *ECCV*, 2020.

[31] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IROS*, 2015.

[32] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *ICCV*, 2015.

[33] Z. Li, H. Wang, and J. Li, "Auto-mvcnn: Neural architecture search for multi-view 3d shape recognition," *arXiv:2012.05493*, 2020.

[34] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *CVPR*, 2020.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[38] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *CVPR*, 2020.

[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM TOG*, vol. 38, no. 5, pp. 1–12, 2019.

[40] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCV*, 2019.

[41] M. Xu, R. Ding, H. Zhao, and X. Qi, "Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds," in *CVPR*, 2021.

[42] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, 2021.

[43] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *ICCV*, 2021.

[44] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.

[45] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*, 2016.

[46] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.

[47] Y. Ding, H. Fan, M. Xu, and Y. Yang, "Adaptive exploration for unsupervised person re-identification," *ACM TOMM*, vol. 16, no. 1, pp. 1–19, 2020, doi:10.1145/3369393.

[48] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.

[49] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.

[50] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Computer Society*, 2016.

[51] L. Zhu, H. Fan, Y. Luo, M. Xu, and Y. Yang, "Temporal cross-layer correlation mining for action recognition," *TMM*, pp. 1–1, 2021, doi:10.1109/TMM.2021.3057503.

[52] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *NeurIPS*, 2014.

[53] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[54] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv:1807.03748*, 2018.

[55] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*.

[56] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[57] T. Han, W. Xie, and A. Zisserman, "Self-supervised co-training for video representation learning," *NeurIPS*, 2020.

[58] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *NeurIPS*, 2020.

[59] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *ICML*, 2021.

[60] L. Zhang and Z. Zhu, "Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks," in *3DV*, 2019.

[61] 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 *ICCV*, 2019.

[62] Y. Rao, J. Lu, and J. Zhou, "Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds," in *CVPR*, 2020.

[63] P.-S. Wang, Y.-Q. Yang, Q.-F. Zou, Z. Wu, Y. Liu, and X. Tong, "Unsupervised 3D learning for shape analysis via multiresolution instance discrimination," 2021.

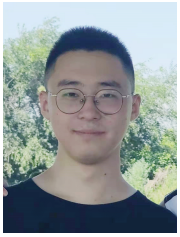
[64] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra, "Self-supervised pretraining of 3d features on any point-cloud," in *ICCV*, 2021.

[65] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu, "Pointcutmix: Regularization strategy for point cloud classification," *Neurocomputing*, vol. 505, pp. 58–67, 2022.

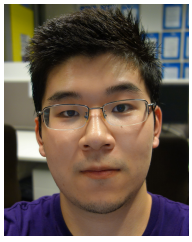
[66] D. Lee, J. Lee, J. Lee, H. Lee, M. Lee, S. Woo, and S. Lee, "Regularization strategy for point cloud via rigidly mixed sample," in *CVPR*, 2021.

[67] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[68] Y. Yang, Y. Zhuang, and Y. Pan, "Multiple knowledge representation for big data artificial intelligence: framework, applications, and case studies," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 12, pp. 1551–1558, 2021, doi:10.1631/FITEE.2100463.
[69] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017.
[70] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv:1607.08022*, 2016.
[71] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, 2006.
[72] Z. Zheng, L. Zheng, M. Garrett, Y. Yang, M. Xu, and Y.-D. Shen, "Dual-path convolutional image-text embeddings with instance loss," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2, pp. 1–23, 2020, doi:10.1145/3383184.
[73] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *ECCV*, 2018.
[74] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *ACM Transactions on Graphics*, 2018.
[75] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, "Walk in the cloud: Learning curves for point clouds shape analysis," in *ICCV*, 2021.
[76] Z. Zheng and Y. Yang, "Parameter-efficient person re-identification in the 3d space," *arXiv:2006.04569*, 2020.
[77] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv:1512.03012*, 2015.
[78] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *CVPR*, 2016.
[79] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2016.
[80] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *JMLR*, 2008.
[81] J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C. K. Yeo, B. Dai, and C. C. Loy, "Unsupervised 3d shape completion through gan inversion," in *CVPR*, 2021.
[82] D. W. Shu, S. W. Park, and J. Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *ICCV*, 2019.
[83] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *ICCV*, 2019.
[84] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.



Chao Sun received the B.E. degree in software engineering from Zhejiang University, China, in 2021. He is currently a master student with the School of Computer Science at Zhejiang University, China. His current research interests include the 3D point cloud and the generative models.



Zhedong Zheng received the Ph.D. degree from the University of Technology Sydney, Australia, in 2021 and the B.S. degree from Fudan University, China, in 2016. He is currently a postdoctoral research fellow at Sea-NExT joint lab, School of Computing, National University of Singapore. He was an intern at Nvidia Research (2018) and Baidu Research (2020). His research interests include robust learning for image retrieval, generative learning for data augmentation, and unsupervised domain adaptation.

Xiaohan Wang received the Ph.D. degree in computer science from University of Technology Sydney, Australia, in 2021. He received the B.E. degree from University of Science and Technology of China, China, in 2017. He is currently a postdoctoral researcher with the College of Computer Science and Technology, Zhejiang University, China. His research interest includes video analysis, egocentric vision and multi-modal understanding.

Mingliang Xu is a professor in the School of Information Engineering of Zhengzhou University, China. He received his Ph.D. degree in computer science and technology from the State Key Lab of CAD&CG at Zhejiang University, Hangzhou, China, and the B.S. and M.S. degrees from the Computer Science Department, Zhengzhou University, Zhengzhou, China, respectively.

Yi Yang received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently a professor with University of Technology Sydney, Australia. He was a Post-Doctoral Research with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. His current research interest includes machine learning and its applications to multimedia content analysis and computer vision.