ULD-Net: 3D Unsupervised Learning by Dense Similarity Learning with Equivariant-Crop

Yu Tian,[1,2] Da Song,[1,2] Mengna Yang,[1,2] Jie Liu,[1,2,3] Guohua Geng,[1,2] Mingquan Zhou,[1,2] Kang Li,[1,2] and Xin Cao[1,2,4]

[1] School of Information Science and Technology, Northwest University, Xi'an, Shaanxi, China
[2] National and Local Joint Engineering Research Center for Cultural Heritage Digitization, Xi'an, Shaanxi 710127, China
[3] jieliu2017@126.com

[4] caoxin@nwu.edu.cn

Received XX Month XXXX; revised XX Month, XXXX; accepted XX Month XXXX; posted XX Month XXXX (Doc. ID XXXXX); published XX Month XXXX

Though many recent deep learning methods have achieved good performance in point cloud analysis, most of them are built upon the heavy cost of manual labeling. Unsupervised representation learning methods have attracted increasing attention due to their high label efficiency. How to learn more useful representations from unlabeled 3D point clouds is still a challenging problem. Addressing this problem, we propose a novel unsupervised learning approach for point cloud analysis, named as ULD-Net, consisting of an Equivariant-Crop (Equiv-Crop) module to achieve dense similarity learning. We propose dense similarity learning that maximizes consistency across two randomly transformed global-local views at both the instance level and point level. To build feature correspondence between global and local views, an Equiv-Crop is proposed to transform features from the global scope to the local. Unlike previous methods that require complicated designs such as negative pairs and momentum encoders, our ULD-Net benefits from the simple Siamese network that relies solely on stop-gradient operation preventing the network from collapsing. We also utilize the feature separability constraint for more representative embeddings. Experimental results show that our ULD-Net achieves the best results of context-based unsupervised methods and comparable performances to supervised models in shape classification and segmentation tasks. On the linear SVM classification benchmark, our ULD-Net surpasses the best context-based method STRL by 1.1% overall accuracy. On tasks with fine-tuning, our ULD-Net outperforms STRL under fully-supervised and semi-supervised settings, in particular, 0.1% accuracy gain on ModelNet40 classification benchmark, and 0.6% mIoU gain on ShapeNet Part segmentation benchmark.

1. INTRODUCTION

As a common 3D representation, the significant advantage of point cloud data over other representations (e.g. volumetric grids, meshes, depth images) lies in its easy availability. With the advancement of 3D acquisition technologies, various types of 3D scanners, LiDARs, and RGB-D cameras (e.g. cameras in Kinect and Apple devices) are becoming ever more accessible, thus point cloud data can be quickly acquired without triangulating data into grids or voxel form. Hence, point cloud data is ideal for wide-ranging applications such as autonomous driving[1], building information modeling (BIM)[2], and digital preservation of ancient artifacts[3].

Recently, deep learning approaches became a dominant source in point cloud analysis in resolving various problems, including 3D shape classification, segmentation, object detection and tracking, registration, and so on. The remarkable advances in point cloud shape understanding rely on the large scale of labeled training data, and the performance improves logarithmically based on the size of annotated training data. The tedious and resource-consuming annotation process became a bottleneck for sustainable success due to the following reasons: (1) because of the sparsity, annotations for low-resolution point clouds are always ambiguous; (2) with the huge amount of points in dense objects which can reach hundreds of millions, point-by-point annotation comes with significant costs; (3) the annotation for 3D objects are inherently more error-prone than for 2D instances as its high complexity; (4) few works have focused on building automatic annotation tools for 3D point clouds, existing tools are still in the early stage manifested in their low accuracy and inconvenience.

In order to resolve the above practical difficulties, researchers explored unsupervised representation learning (URL) in the 3D point cloud analysis field based on the easy availability of unlabeled data. In common URL settings, the network learns knowledge from pretext tasks without supervision in the pre-training stage, then transfers the learned knowledge to other downstream tasks. Most existing works are based on generation tasks[4-8] that rely heavily on the specific architecture designation, such as folding-based decoders for completion and reconstruction tasks, the performance in downstream tasks degenerates when using a general MLP-based decoder. Meanwhile, generation-based tasks concentrate on geometric structures which results in poor transferability on scene-level datasets. To eliminate the dependence on such specific components and improve model transferability to real-world scenes, several recent works consider context similarities[9-16] between samples to explore generic methods for URL.

Inspired by the huge success of self-supervised learning in 2D computer vision domain, several efforts have been devoted to exploring context similarities in 3D point clouds based on Siamese networks. Most works built on top of contrastive learning rely on negative samples, Info3D[15] proposes to learn representations by maximizing mutual information between 3D objects and their local parts. Towards more discriminative features from local patches, Du et al.[16] introduced a hard negative sampling strategy into architecture. PointContrast[9] extracts dense correspondences across two views of scene point clouds for point-level contrastive learning. Without the requirements of negative samples, STRL[10] extends BYOL[17] from 2D image processing to 3D point cloud analysis by learning features between original objects and their augmented views. However, previous works conduct unsupervised pre-training with complicated designations, such as negative pairs sampling[9], memory banks[15], and momentum encoders[10]. Additionally, most methods individually considered context similarities between transformed views at the instance level[10, 15, 16] or point level [9], separately maintaining consistency at both two levels was not taken into account.

To this end, we present a dense representation learning approach named 3D Unsupervised Learning by Dense Similarity Learning with Equivariant-Crop (ULD-Net) based on three common-sense intuitions. First, purely considering instance-level similarity dismisses local spatial information, while learning point-level similarity cannot extract representative abstract semantic information for the entire object. Thus, we jointly optimize the model at both levels, which helps to learn sufficient knowledge for downstream tasks. Second, the two branches of the network output point-level features within different scopes, while point-level similarity learning aims to maximize corresponding features across views, the features should share the same scopes with one-to-one correspondence. Therefore, we propose an Equiv-Crop module equivariant with Cropping transformation to map the global features to the local scope. Third, it is proved that without redundant components which raise the computational cost, a simple stop-gradient design can get the network rid of collapse[18]. Using these inductive biases alone, we can train a Siamese network with a stop-gradient operation on top of SimSiam[18] to output point embeddings, with objectives maximizing similarities between embeddings across local-global views, aiming at pre-training dense representations with strong transferability in downstream tasks.

The process of the proposed method is illustrated in Fig. 1. The pair of augmented point clouds (shown as blue dots) in global-local views are processed by the same encoder network and a projector network to extract features (shown as pentagons or crosses in other colors).

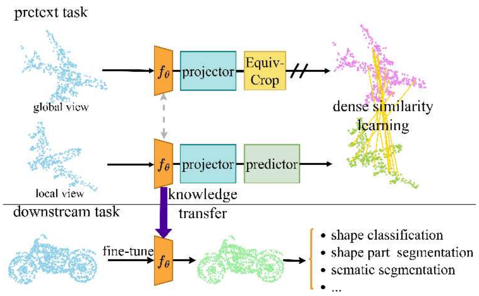

Fig. 1. An illustration of the proposed method.

The Equiv-Crop module is applied to the global view side to project global features to the local scope. The predictor network is applied on one side, and the stop-gradient operation is on the other side. After taking dense similarity learning as a pretext task during pre-training, the trained encoder network transfers the learned knowledge to downstream tasks such as shape classification, part segmentation, semantic segmentation, and so on. We theoretically prove the intuitions can improve the performance through serial experiments conducted, the method we proposed achieves competitive results to existing methods. Our contributions can be summarized as follows:

(1) We propose a novel method for 3D point cloud unsupervised representation learning, which learns dense features by maximizing their local-global similarities at the point level and instance level, eliminating the need for negative samples or other complicated designs.

(2) We introduce a novel point mapping strategy named Equiv-Crop for correspondence across views with local and global scopes, to provide the foundation for point-level feature learning. The local scope is produced by a Cropping operation, and two augmented views are generated by integrated with an Inv-Aug strategy while the robustness is boosted.

(3) We present a feature separability constraint that maximizes the separability of feature vectors from different dimensions while boosting the representability of features.

2. RELATED WORKS

A. Deep architectures for point cloud processing

The advances in deep learning and learning-based point descriptors have been helpful to the impressive performance of recent point cloud processing for several 3D understanding tasks. Existing methods focus on alleviating the difficulty caused by the irregularity of 3D point clouds, with most works extracting features directly from points.

PointNet[19] is the seminal work using deep learning that performs directly on raw point clouds, which achieved input order invariance by symmetric functions. Since PointNet learns features independently through point-wise MLP for each point, later works paid attention to capturing local structural information using various methods. PointNet++[20] learns local features by a hierarchical network, which is stacked by set abstraction layers. PointCNN[21] designed discrete convolutional kernel χ-conv particular for point clouds. Considering each point in point clouds as a vertex, DGCNN[22] and RGCNN[23] construct graphs in spatial and spectral space. Kd-Net[24] learns features by constructing hierarchical data structures based on K-d trees. Recently, transformer-based methods[25, 26] are proposed for long-range visual dependencies learning. In this work, common architectures are suitable to be utilized as backbone networks because of our flexible designation.

B. Deep architectures for point cloud processing

URL is drawing increasing interest owing to its superiority in resolving the annotation bottleneck. Since annotations for 3D data take higher costs than 2D vision data, 3D tasks are supposed to benefit much more from URL. However, compared with Natural Language Processing (NLP) and 2D vision, the unsupervised pre-text task defined for 3D point cloud data is much less mature.

Numerous pretext tasks have been proposed for strong presentation acquisition with specific objectives, which can be broadly divided into 2 categories: generation-based and context-based tasks. Generation-based tasks take point clouds themselves as supervised information, including reconstructing original input from low-dimensional vectors[4, 5], generating new point clouds similar to training samples from random noise[6], up-sampling point clouds from sparse to dense[7], and completing missing parts[8]. Learning features through context-based methods is another rising research direction, including performing instance discrimination[9, 10], solving 3D jigsaw puzzles[11], predicting rotation angles[12], predicting the next point in the sequence[13], and disentangling the mixed point clouds[14]. Considering multi-level similarity, a pretext task defined as optimizing the cosine similarities at both the instance level and point level is proposed, accompanied by a feature separability constraint aiming at more representative features.

C. Siamese neural networks

Siamese network consists of two identical artificial neural networks for comparing the projected representations of the two input vectors. The key challenge in siamese methods is how to avoid collapsing solutions. SimCLR[27] and MoCo[28] proposed based on the core idea of contrastive learning that drags positive sample pairs and pushes negative sample pairs away. Different from comparing samples in the current batch in SimCLR, MoCo builds a dynamic dictionary with a queue and a moving-averaged encoder to get rid of the dependence on large batch size and to improve the consistency of the queues. Clustering-based methods construct Siamese networks with clustering intergraded while achieving competitive results without a memory bank. Specifically, SwAV[29] solves degenerate solutions through computing cluster assignments from one view playing as negative samples relying on the Sinkhorn-Knopp algorithm. Asymmetric methods prevent features from collapsing using asymmetric architecture. BYOL[17] uses a momentum encoder accompanied by stop-gradient and moving-average, while SimSiam[18] removes the momentum encoder and keeps minimum core architecture as an elegant realization. Inspired by SimSiam, we build our network based on the Siamese architecture with a stop-gradient operation as its computational advantage.

3. METHOD

The overall pipeline of our ULD-Net is depicted in Fig. 2. Taking unsupervised point cloud datasets as source data, our fundamental idea is to train an encoder network by modeling dense consistency between local-global features from transformed views to extract representations for better transferability on downstream tasks.

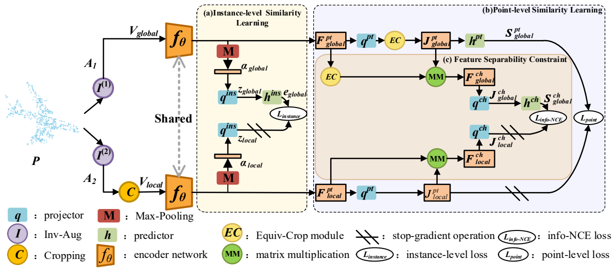

Fig. 2. The overall overview of the proposed pre-training method.

For each point cloud object P, we first transform the original input into two random augmented views in the global and local scope by Inv-Aug and Cropping transformations. Then, we encode the point clouds to generate feature maps in high dimensional space by a weight-shared encoder network f_θ. Inspired by SimSiam, we promote the representational ability of the encoder by minimizing the dissimilarities between feature maps through dense similarity learning. We integrate the instance-level (Fig. 2(a)) and point-level (Fig. 2(b)) similarity learning into a unified framework, and we also utilize feature separability constraint (Fig. 2(c)) for more discriminative features. Taking our approach to pre-train an encoder network from unlabelled data, the learned encoder can be transferred to various downstream tasks for feature extraction.

A. Views and Features Generation

Given each input point cloud $P \in \mathbb{R}^{N\times 3}$ with N elements, we transform its geometric features (XYZ coordinates) by Inv-Aug and Cropping operations with random factors. Inv-Aug is a collection of data augmentations consisting of rotation, translation, scaling, and jittering. We start the transformation with two randomly Inv-Aug augmentations $I^{(1)}$ and $I^{(2)}$ that output two augmented point clouds $A_1 = I^{(1)}(P)$ and $A_2 = I^{(2)}(P)$. Regarding one augmented point cloud A_1 as the global view $V_{global} = A_1$ of the input point cloud P (as in Eq.(3)), we conduct a Cropping operation C on another augmented point cloud to generate the local view. Different from Inv-Aug which keeps the object complete, the Cropping operation transforms points to a random local scope.

1. Cropping Operation

The Cropping operation C consists of two steps conducted at the augmented point set A_2. First, we compute the indices of at least 50% of points inside the coordinate range defined by a random 3D cuboid, with computed indices $m = \{j_i \in [1,\ldots,N], i \in [1,\ldots,L]\}$, the selected L points $A_2[m]$ inside a cuboid local scope are kept as downsampled points. Since the point sequence is invariant to Inv-Aug operations $I^{(1)}$ and $I^{(2)}$, the downsampled point set $A_2[m]$ with L elements corresponds exactly to points with the same indices m in the global scope point set V_{global} (i.e. A_1). Second, we upsample the downsampled point set $A_2[m]$ from size L to the predefined input size N of the encoder network. We choose inverse distance weighted average based on k nearest neighbors (as in Eq.(1), in default we use $p = 2, K = 3$) for interpolation, outputs upsampled point set regarded as local view (as in Eq.(4)),

$$V_{local}[i] = \frac{\sum_{r=1}^{K} w_r(A_2[i]) A_{knn}[i,r]}{\sum_{r=1}^{K} w_r(A_2[i])}, i \in [1,\dots,N] \quad (1)$$

where $A_{knn} \in \mathbb{R}^{N \times K \times 3}$ denotes the K nearest neighbors of N points in the entire point set A_2, the neighbors are searched in the downsampled point set $A_2[m]$, $d(\cdot)$ denotes the euclidean distance between two points, and w_r is computed for the weight of the rth neighbor:

$$w_r(A_2[i]) = \frac{1}{d(A_2[i], A_{knn}[i,r])^p} \quad (2)$$

In conclusion, the correspondence between two scales is constructed by downsampled indices m and the K neighbors A_{knn}. Thus, two different views in local and global scope with transformation in Eq.(3)(4) are produced.

$$V_{global} = A_1 = I^{(1)}(P) \quad (3)$$

$$V_{local} = C(A_2) = C(I^{(2)}(P)) \quad (4)$$

2. Dense Feature Map Generation

The global-local views from the same point cloud are then processed by a backbone encoder network f_θ with parameters θ. The encoder shares the same weights between views. For local input V_{local}, the encoder f_θ computes a point-level feature map $F_{local}^{pt} = f_\theta(V_{local})$ including representations for each point in the local view V_{local}, the feature vector for the ith point is noted as $F_{local}^{pt}[i]$. Simultaneously, f_θ yields a high-dimensional vector α_{local} after max-pooling describing the entire local view V_{local} at the instance level. Following the same computation pipeline with local features, a point-level feature map $F_{global}^{pt} = f_\theta(V_{global})$ and an instance-level representation α_{global} from the global view are generated.

$$\alpha_{global} = maxpooling(f_\theta(V_{global})) \quad (5)$$

$$\alpha_{local} = maxpooling(f_\theta(V_{local})) \quad (6)$$

3. Equiv-Crop

Denoting high dimensional features from global and local view without max-pooling as $F_{global}, F_{local} \in \mathbb{R}^{N \times D}$, where D denotes the number of feature dimensions, global view features F_{global} can be transformed to local scope through a module EC equivalent to Cropping operation. Since the network is permutation invariant, the sequences of output features correspond to network input sequences. Namely, there is a one-to-one correlation between the input point and output feature. For example, the ith point $V_{global}[i]$ in the global view is represented by the feature vector $F_{global}[i]$. Based on such a principle, global features F_{global} can be directly mapped into the corresponding local scope using the same correspondence in the Cropping operation done in views transformation. Specifically, we gather the global features of points in the same local cuboid scope in Cropping following the same downsampling and upsampling steps. First, we downsample the features by selecting indices m saved in Cropping, the downsampled features $F_{global}[m]$ represent features of points in downsampled points $A_2[m]$ in Cropping. Then, with the downsampled features $F_{global}[m]$, we upsample the features from the searched K nearest neighbors A_{knn} same as in Cropping,

$$EC(F_{global}[i]) = \frac{\sum_{r=1}^{K} w_r(A_2[i]) F_{knn}[i,r]}{\sum_{r=1}^{K} w_r(A_2[i])}, i \in [1,\dots,N] \quad (7)$$

where $F_{knn} \in \mathbb{R}^{N \times K \times D}$ denotes features of neighbors A_{knn}, $d(\cdot)$ denotes the euclidean distance between two vectors, and w_r denotes the weight of the rth neighbor. The Equiv-Crop module then transforms global features into local scope defined in Cropping.

B. Dense Similarity Learning

To achieve sophisticated similarity measurement, we learn local-global consistency through dense similarity learning. Solely learning consistency between local and global views at the instance level would cause most of the spatial information to be discarded during pooling. To tackle this question, we jointly learn instance-level and point-level similarities. Moreover, for point-level feature learning, we utilize the Equiv-Crop module in Sec. A.3 towards mapping point embeddings from the global scope to the local one.

1. Instance-Level Similarity Learning

We learn instance-level similarity from representations α_{local} and α_{global}, the pipeline is shown in Fig.2(a). We first transform features by the same projector network q^{ins}, which is a three-layer MLP head output with features $z_{global} = q^{ins}(\alpha_{global})$ and $z_{local} = q^{ins}(\alpha_{local})$. Then, a predictor network h^{ins} transforms the projected feature from one view to predict another, outputs predictions $e_{global} = h^{ins}(z_{global})$ and $e_{local} = h^{ins}(z_{local})$. Meanwhile, a stop-gradient operation is applied to the projected features from another view. We symmetrically minimize the distance of feature maps and predictions from another view:

$$L_{instance} = \frac{1}{2}\mathrm{D}(e_{local}, \mathrm{sg}(z_{global})) + \frac{1}{2}\mathrm{D}(e_{global}, \mathrm{sg}(z_{local})) \quad (8)$$

where sg is the stop-gradient operation to avoid the outputs of the network collapsing to constant and $\mathrm{D}(\cdot)$ in Eq.(9) is a distance function measuring negative cosine similarity in high-dimensional feature space:

$$\mathrm{D}(e_{local}, z_{global}) = -\frac{e_{local}}{\| e_{local} \|_2} \cdot \frac{z_{global}}{\| z_{global} \|_2} \quad (9)$$

where $\|\cdot\|_2$ denotes $l2$ normalization.

2. Point-Level Similarity Learning

We formulate point-level similarity learning as shown in Fig.2(b) to maximize the similarity of point predictions. Input with point-level feature maps F_{local}^{pt} and F_{global}^{pt}, following the same pipeline with instance-level similarity learning, a projector q^{pt} is used to transform the point-level features first. For each point, we predict its feature from another view. However, due to the input point represented by ith feature mismatch between local and global scope, it is incompatible with common sense to predict directly between two features from the same indices. To bridge the gap, an Equiv-Crop module EC maps the projected features from global to local scope, and the projected features are noted as $J_{global}^{pt} = EC(q^{pt}(F_{global}^{pt}))$, $J_{local}^{pt} = q^{pt}(F_{local}^{pt})$. After that, the predictions $S_{global}^{pt} = h^{pt}(J_{global}^{pt})$ and $S_{local}^{pt} = h^{pt}(J_{local}^{pt})$ for each point are outputted from the point-level predictor h^{pt}.

We symmetrically maximize the similarity between the projected feature for the ith point and its prediction:

$$L_{point} = \sum_{i=1}^{N} \frac{1}{2} \mathrm{D}(S_{local}^{pt}[i], \mathrm{sg}(J_{global}^{pt}[i])) + \frac{1}{2} \mathrm{D}(S_{global}^{pt}[i], \mathrm{sg}(J_{local}^{pt}[i])) \tag{10}$$

C. Feature Separability Constraint

It is common that projected features and predictions (such as $J_{global}^{pt}[i]$ and $S_{local}^{pt}[i]$) contain different information after random augmentations, but similarity learning forces these embeddings to be close to each other, which leads to a risk of features from different dimensions degenerating to the same value. To address the degenerating issue, besides the stop-gradient operation, we further propose a feature separability constraint as illustrated in Fig.2(c) to boost the expressiveness of features.

The channel embeddings are obtained by the sum of multiplication between the feature maps and predictions:

$$F_{global}^{ch} = \sum_{i}^{N} J_{global}^{pt}[i] \cdot EC(F_{global}^{pt})[i] \tag{11}$$

$$F_{local}^{ch} = \sum_{i}^{N} J_{local}^{pt}[i] \cdot F_{local}^{pt}[i] \tag{12}$$

where $F_{global}^{ch}, F_{local}^{ch} \in \mathbb{R}^{D \times D'}$, D and D' represents the number of output feature channels in the predictor and encoder.

Similar to similarity learning, we transformed the embeddings by a projector composed of an MLP head q^{ch} and predictor h^{ch} output embeddings $J_{global}^{ch} = q^{ch}(F_{global}^{ch})$, $J_{local}^{ch} = q^{ch}(F_{local}^{ch})$ and predictions $S_{global}^{ch} = h^{ch}(J_{global}^{ch})$, $S_{local}^{ch} = h^{ch}(J_{local}^{ch})$. By using info-NCE loss[30], the similarities of features in different channels decreased, which leads to higher separability. Specifically, we optimize the feature separability by Eq. (13):

$$L_{separability} = \frac{1}{2} L_{info-NCE}(S_{local}^{ch}, \mathrm{sg}(J_{global}^{ch})) + \frac{1}{2} L_{info-NCE}(S_{global}^{ch}, \mathrm{sg}(J_{local}^{ch})) \tag{13}$$

where $L_{info-NCE}$ is the info-NCE loss as:

$$L_{info-NCE}(S, J) = -\sum_{r=0}^{R} log \frac{\exp(S[r] \cdot J[r] / \tau)}{\sum_{r'=0}^{R} \exp(S[r] \cdot J[r'] / \tau)} \tag{14}$$

where τ denotes the temperature coefficient of 0.1 in default, R denotes the number of dimensions of the prediction feature S.

4. Experiments and Results

A. Datasets

To validate the effectiveness and transferability of our method, three benchmarks (ModelNet40[31], ShapeNet part[32], and S3DIS[33]) are used in the experiments. In the pre-training stage, ModelNet40 is used for all experiments, ShapeNet55 is additionally used for linear evaluation comparison. For downstream tasks, we use ModelNet40 benchmark for shape classification, ShapeNet Part benchmark for shape part segmentation, and S3DIS benchmark for scene semantic segmentation.

ModelNet40. ModelNet40 includes 12,311 synthesized 3D objects (divided into 9,843 training samples and 2,468 testing samples) from 40 categories. We downsample each object to 2,048 points whose XYZ coordinates normalized into a unit sphere following the pre-processing method from PointNet[19].

ShapeNet Part. ShapeNet55[34] contains 57,748 synthetic 3D shapes from 55 categories. ShapeNet Part benchmark includes 16,881 shapes of 16 categories selected from ShapeNet55. Each sample is annotated with 2 to 5 parts, part labels for all categories amounted to 50. Intersection of Union (IoU) is widely used for segmentation evaluation that measures the ratio between point-wise ground truth and prediction. For the part segmentation task, we compute category mIoU by averaging IoUs over parts of the same object category, instance mIoU is obtained by averaging over all test shapes.

S3DIS. Stanford 3D Indoor Spaces (S3DIS) dataset contains 3D scans of 6 different places including 271 rooms, which cover over $6,000m^2$. Each point is represented by a 9-dimensional vector consisting of XYZ coordinates, RGB color values, and normalized location, individual point is labeled with 13 semantic categories. We use the same pre-processing procedures as the original work, each room is split into blocks with $1m \times 1m$ area, and each block contains 4,096 points sampled. To evaluate semantic segmentation performance, mIoU is computed by averaging IoUs over all points.

B. Implementation Details

Architecture Parameters. For a fair comparison with previous methods, DGCNN backbone is used as the default encoder network which outputs features with 1,024 dimensions. All projectors and predictors are designed with the same architecture. Specifically, each projection MLP head consists of 3 fully connected layers with dimensions of [512,256,256], each prediction MLP head consists of 2 fully connected layers with dimensions of [512,256], each layer has batch normalization applied, and LeakyReLU activation with a negative slope of 0.2 is used except for the final output layer. For jointly learning instance-level similarity, point-level similarity, and feature separability, our ULD-Net optimizes the total loss $L = \lambda_1 L_{instance} + \lambda_2 L_{point} + \lambda_3 L_{separability}$, to balance the significance of all tasks, we choose $\lambda_1 = \lambda_2 = 100$ and $\lambda_3 = 10$ based on the numbers of each loss to keep them in the same order of magnitude.

Pre-training Setup. We follow the settings of STRL in unsupervised pre-training experiments. We implemented our work with the deep learning library PyTorch using a single TITAN RTX GPU for all experiments. Specifically, the Adam optimizer is used in our model with an initial learning rate of 0.001, the learning rate is decayed by 0.7 every 20 epochs, and the batch size is 24 by default. We pre-train ULD-Net for 200 epochs on ModelNet40.

Fine-tuning Setup. As an end goal in URL, we verify the effectiveness of the pre-trained features transferred to new tasks in a fully-supervised fashion. For 3D shape classification on ModelNet40, we use a batch size of 24 for training and testing with 250 epochs, the SGD optimizer is used with an initial learning rate of 0.1, momentum 0.9, and weight decay 0.0001, and the learning rate is decayed with a cosine annealing scheduler. Slightly different from the above settings for the classification task, the batch size used for 3D part segmentation on ShapeNet Part is 16 and we train the network for 100 epochs with Adam optimizer for 3D semantic segmentation on S3DIS.

C. Downstream Results

1. Linear evaluation for Shape Classification

For 3D shapes classification, we train a linear SVM (Support Vector Machine) on the target dataset ModelNet40 to evaluate the effectiveness of the learned instance-level features following the common protocol in prior URL works[8, 10, 11]. For the SVM classifier, the input features are obtained after the pre-trained encoder network with the following pooling layer, and the weights of the feature extractor are frozen during evaluation. Following the settings in DGCNN classification network, the pooling layer outputs concatenated features after max-pooling and average-pooling operations. The classification results compared with the state-of-the-art are shown in Table 1, all methods tabulated are implemented with DGCNN backbone as a feature extractor for a fair comparison. As shown in the table, the proposed method achieves 91.9% and 92.0% overall accuracy after pre-trained on ShapeNet55 and ModelNet40 dataset, which outperforms existing unsupervised method STRL[10] by 1.0% and OcCo[8] by 2.8%. These results suggest that the features attained by our pre-training method are discriminative that can easily achieve competitive performance even with little effort of training on SVM.

Table 1. Classification accuracy results (%) with linear SVM in URL methods on ModelNet40 ("OA" denotes overall accuracy.)

Pre-training Dataset	Method	OA
ShapeNet	FoldingNet [4]	88.4
	Du et al. [16]	89.6
	Jigsaw3D [11]	90.6
	Rotation3D [12]	90.8
	STRL [10]	90.9
	Ours	**91.9**
ModelNet40	FoldingNet [4]	84.4
	Jigsaw3D [11]	87.8
	MAP-VAE [5]	90.2
	OcCo [8]	89.2
	Ours	**92.0**

Towards a better understanding of the capability of our method proposed, we visualize the learned features on the test dataset of the ModelNet10 as illustrated in Fig.3, which is compared with features from a randomly initialized encoder network. Using T-SNE (t-distributed stochastic neighbor embedding)[35] to project the instance-level high-dimensional features in 2D space, we observe that the learned features from instances of different categories are separable, except dressers and nightstands, which are difficult to distinguish even by a human. Compared with the projected features from random initialization, our pre-trained are dragged to further distances between features of distinct categories. Since the random initialized features can be regarded as the prior of the encoder network, the comparison proves our pre-training method can learn knowledge of 3D shapes without supervision.

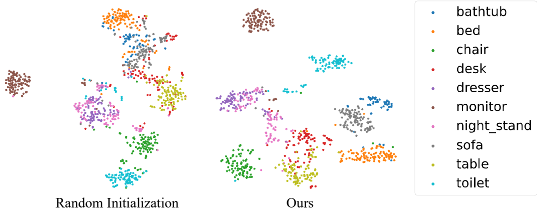

Fig. 3. Visualization of pre-trained instance-level features.

2. Supervised Fine-tuning for 3D Shape classification

Further fine-tunes the encoder network on ModelNet40 without freezing, resulting in better classification accuracy. Following a common fine-tuning pipeline in URL methods, after an unsupervised pre-training stage aimed at maximizing dense similarities, we take the pre-trained encoder network parameters as initialization for the encoder network used in transfer learning, then optimize the network by specific objective for classification task in a supervised fashion. To produce predictions for the classification task, we train a classification MLP head during fine-tuning along with the encoder network, the classification head takes instance-level features after pooling as input and output with classification scores for each object towards supervised validation on ModelNet40. Comparisons of fine-tuned classification results are illustrated in Table 2.

Table 2 . Comparisons of our fine-tuned classification accuracy (%) result against other methods on ModelNet40 ("Sup." denotes supervised.)

Method	Sup.	OA
PointNet [19]	✓	89.2
RGCNN [23]	✓	90.5
PointNet++ [20]	✓	90.7
KD-Net [24]	✓	91.8
PointCNN [21]	✓	92.2
DGCNN [22]	✓	92.2
Point Cloud Transformer [26]	✓	93.2
PointTransformer [25]	✓	**93.7**
Jigsaw3D [11]	×	92.4
Info3D [15]	×	93.0
OcCo [8]	×	93.0
FoldingNet [4]	×	93.1
STRL [10]	×	93.1
Ours	×	**93.4**

As shown in Table 2, after fine-tuning from our pre-trained model, the proposed method achieves an additional 1.0% accuracy gain over the original DGCNN trained from randomly initialized parameters (93.2% vs. 92.2%), which suggests our pre-training method can boost the ability of the feature extractor. Our method outperforms unsupervised methods OcCo and STRL by 0.2% and 0.1% in terms of overall accuracy and achieves the best fine-tuned performance on ModelNet40. The results indicate that our ULD-Net can attain a comparable performance with the state-of-the-art fully supervised methods.

Our method accelerates the convergence of the encoder framework during the fine-tuning stage. As shown in Fig. 4, compared with the random initialization, the loss number of our method keeps lower than random initialization at about 0.2 during training and convergence after fewer epochs.

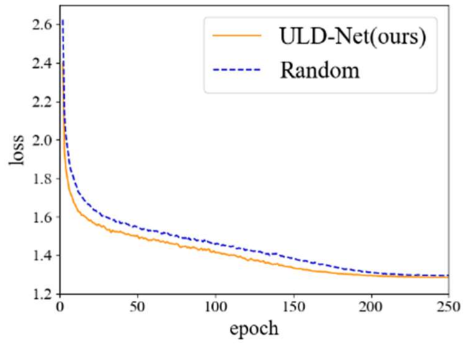

Fig. 4. Convergence curves during fine-tuning.

3. *Semi-supervised Fine-tuning for 3D Shape Classification*

We further evaluate our pre-trained model on the shape classification task under a semi-supervised setting. We use the same setting as STRL and report the overall accuracy on ModelNet40 as shown in Table 3. Specifically, we reduce the annotated input shapes to 1%, 5%, 10%, and 20% of the training data, and at least one shape is selected for each category. Then we evaluate the model fine-tuned by the reduced training data on the full test dataset. The results show that our model surpasses the randomly initialized model by 2.2% and 1.7% when 1% and 20% of training shapes were sampled, and our ULD-Net slightly outperforms STRL when the sampling ratio of 1%, 10%, and 20%, indicates our pre-training method improves annotation efficiency.

Table 3. Fine-tuned results (%) under a semi-supervised setting

Method	1%	5%	10%	20%
DGCNN [22]	58.4	80.7	85.2	88.1
STRL [10]	60.5	**82.7**	86.5	89.7
Ours	**60.6**	82.5	**86.8**	**89.8**

4. *Supervised Fine-tuning for 3D Shape part segmentation*

To validate the effectiveness of fine-grained point-level features gained from our method, we fine-tune the pre-trained network for the part segmentation task. Different from classification fine-tuning only transfers parameters from the encoder network, segmentation fine-tuning uses parameters from the pre-trained encoder and its attached point-level projector. We fine-tune them on ShapeNet Part dataset to verify the performance of our ULD-Net on the part segmentation task. The quantitative results compared with the state-of-the-art URL and supervised methods are shown in Table 4. It shows that our ULD-Net shows the best performance (85.7% instance mIoU) among other URL approaches and achieves top performance in 6 categories such as Aeroplane, Car, and Knife. Since ShapeNet Part is a long-tailed dataset, the instance mIoU is mostly decided by shapes of large amounts (Aeroplanes, Chairs, Lamps, Tables, etc.), which leads to the unbalance performance on different categories of shapes. Compared with supervised methods, we also achieve comparable results.

The segmentation results of all shapes are qualitatively illustrated in Fig.5. These visualization results show our method can segment one shape to clear parts close to the ground truths.

Fig. 5. Qualitative results on ShapeNet Part dataset.

Furthermore, we compare our ULD-Net with STRL and OcCo on shapes including aeroplanes, bags, and cars as illustrated in Fig.6, which shows ULD-Net captures more local details than STRL and OcCo. In confusing regions annotated with blue bounding boxes, containing points in the intersection of the main body and other parts of different categories, such as the tail of aeroplanes demonstrated in Fig.6 (a), the handle of bags in Fig.6 (b) and the roof of cars in Fig.6 (c), shows that our method distinguishes such regions better.

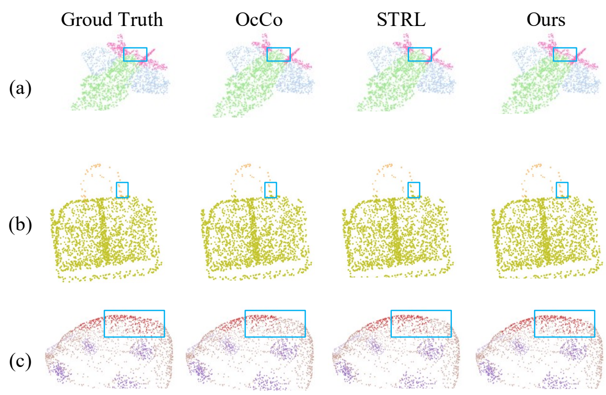

Fig. 6. Visual comparison of part segmentation on ShapeNet Part.

5. *Supervised Fine-tuning for 3D Semantic Segmentation*

Transferring features pre-trained on synthetic CAD object models to real-world segmentation tasks is considered more challenging than tasks on synthetic shapes. To elucidate this problem, we also test our method for the indoor semantic segmentation task on S3DIS dataset to validate the cross-domain generalizability of our pre-trained features to a real-world dataset.

Table 4. Fine-tuning part segmentation mIoU results (%) on ShapeNet Part dataset ("Ins." denotes instance mIoU.)

Shapes	Supervised Method			Unsupervised Method					
	DGCNN [22]	RSCNN [36]	PCT [26]	LGAN [6]	Method in [16]	Jigsaw3D [11]	OcCo [8]	STRL [10]	Ours
Ins.	85.2	86.2	**86.4**	57.0	82.3	85.3	85.5	85.1	**85.7**
Aero	84.0	83.5	**85.0**	54.1	82.1	84.1	84.4	83.7	**84.7**
Bag	83.4	**84.8**	82.4	48.7	74.5	**84.0**	77.5	80.3	82.8
Cap	86.7	88.8	**89.0**	62.6	83.6	85.8	83.4	**87.6**	83.8
Car	77.8	79.6	**81.2**	43.2	74.9	77.0	77.9	77.7	**78.3**
Chair	90.6	91.2	**91.9**	68.4	87.9	90.9	**91.0**	90.9	90.9
Earphone	74.7	**81.1**	71.5	58.3	72.4	**80.0**	75.2	78.0	77.0
Guitar	91.2	**91.6**	91.3	74.3	89.9	91.5	**91.6**	91.4	91.3
Knife	87.5	**88.4**	88.1	68.4	85.4	87.0	**88.2**	87.7	**88.2**
Lamp	82.8	86.0	**86.3**	53.4	79.1	83.2	83.5	83.7	**83.8**
Laptop	95.7	**96.0**	95.8	82.6	95.2	95.8	**96.1**	**96.1**	95.6
Motor	66.3	**73.7**	64.6	18.6	67.3	**71.6**	65.5	66.7	68.6
Mug	94.9	94.1	**95.8**	75.1	93.3	94.0	94.4	**95.0**	94.3
Pistol	81.1	83.4	**83.6**	54.7	81.0	**82.6**	79.6	81.2	80.6
Rocket	**63.5**	60.5	62.2	37.2	58.2	60.0	58.0	58.2	**61.9**
Skateboard	74.5	**77.7**	77.6	46.7	74.0	**77.9**	76.2	75.3	75.1
Table	82.6	83.6	**83.7**	66.4	79.2	81.8	82.8	82.1	**83.4**

Using the pipeline similar to part segmentation, we transfer the parameters of the encoder and point-level projector to supervised fine-tuning for the semantic segmentation task.We test our model under 6-fold cross-validation over the 6 areas as in the original work[33]. As the quantitative results summarized in Table 5, our ULD-Net achieves the best segmentation result with 85.5% overall accuracy and 59.2% mIoU, which surpasses the state-of-the-art method OcCo by 0.4% overall accuracy and 0.7 mIoU. Compared with existing URL methods, these results demonstrate better transferability of our ULD-Net from synthetic shapes to real-world scene datasets. It is observed that our results even surpass the supervised PointNet, PointNet++, and DGCNN, and also achieve competitive performance with other supervised models.

Table 5. Semantic segmentation results (%) on S3DIS dataset.

Method	Sup.	OA	mIoU
PointNet [19]	✓	78.6	47.6
PointNet++ [20]	✓	81.0	54.5
PointCNN [21]	✓	**88.1**	**65.4**
DGCNN [22]	✓	84.1	56.1
Jigsaw [11]	×	84.4	56.6
OcCo [8]	×	85.1	58.5
Ours	×	**85.5**	**59.2**

We show qualitative results of S3DIS indoor semantic segmentation by visualizing selected rooms in Fig. 7. Empirically, we observe that our network is able to understand and classify semantic objects in a real-world scene, and our segmentation results are close to the ground truth.

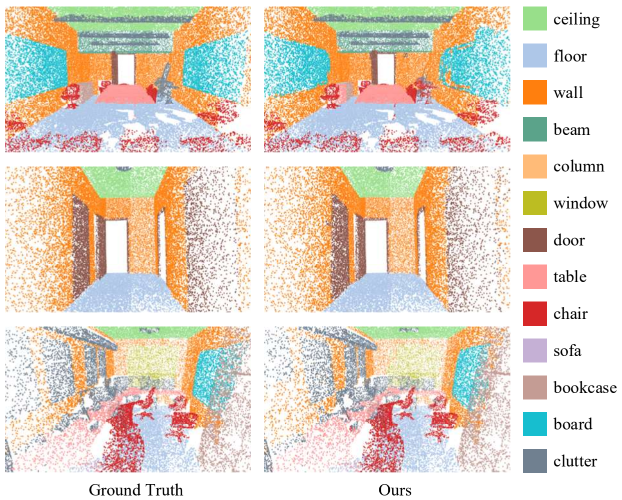

Fig. 7. Visualization of semantic segmentation results on S3DIS Dataset.

D. Ablation Study

To investigate the effectiveness of our key components in ULD-Net, we study the impact of adopting different combinations of losses and transformations during the pre-training stage by validating the downstream SVM classification results using their pre-trained features on ModelNet40.

1. Transformations

We analyze the effectiveness of different transformations in Inv-Aug and Cropping for view generation used in the pre-training stage. We remove certain transformations to produce augmented views when pre-training and validate the implication with SVM. As summarized in Table 6, our full model A_1 uses all transformations and achieves the best result of 92.0%. Without any transformations (model B_1), the network inputs of the two branches are exactly the same, which makes the network overfits pre-training samples due to too many task-irrelevant detailed features captured, hence the classification result degenerates to 88.0%. The result reduces when one transformation is removed, proving that each adopted transformation schedule boosts the performance of pre-trained features. Among transformations, removing the Cropping transformation C (model C_1) affects the performance the most by a 2.4% descent (92.0% vs. 89.6%) compared with the model A_1. Removing each transformation in Inv-Aug including Rotation (model D_1), Translation (model E_1), Jittering (model F_1), and Scaling (model G_1), the performance degenerates to 91.0%, 91.0%, 91.2%, and 91.5% respectively, which indicates the importance of each transformation is decreasing by the above order.

Table 6. Results (%) from pre-trained features with different transformations. ("Rot." Denotes Rotation, "Trans." Denotes Translation, "Scal." Denotes Scaling, "Jit." Denotes Jittering)

Model	C	Rot.	Trans.	Scal.	Jit.	OA
A_1	✓	✓	✓	✓	✓	**92.0**
B_1	×	×	×	×	×	88.0
C_1	×	✓	✓	✓	✓	89.6
D_1	✓	×	✓	✓	✓	91.0
E_1	✓	✓	×	✓	✓	91.0
F_1	✓	✓	✓	×	✓	91.0
G_1	✓	✓	✓	✓	×	91.0

2. Losses

We further study how the training objectives affect the performance of pre-trained features. The results are shown in Table 7, the baseline model A_2 is trained by instance-level similarity loss which closes the distance between the instance and its local parts in embedding space and gets a classification accuracy of 91.3%. Combined with one of the point-level similarity loss (model B_2) or feature separability loss (model C_2), we observed 0.4% and 0.3% improvement respectively. Our full model joint learns with three objectives (model D_2) achieves a notable 92.0% on ModelNet40.

Table 7. Ablation study results (%) of different pre-training objectives.

Model	$L_{instance}$	L_{point}	$L_{separability}$	OA
A_2	✓			91.3
B_2	✓	✓		91.7
C_2	✓		✓	91.6
D_2	✓	✓	✓	**92.0**

E. Robustness

To test the robustness of our method to random noise, we randomly jitter the XYZ coordinates of points with Gaussian noises in linear evaluation on ModelNet40 during test time. Each point cloud is jittered with randomly sampled Gaussian noises with zero mean and standard deviation $\sigma \in \{0.025, 0.05, 0.075, 0.1\}$. As shown in Fig. 8, we compare our ULD-Net with OcCo and STRL under different noise levels. We can see that our ULD-Net keeps robust with 83.9% accuracy even when noise is at a high level with a 0.1 standard deviation. It can also be observed that our ULD-Net gets competitive results with existing URL methods OcCo and STRL.

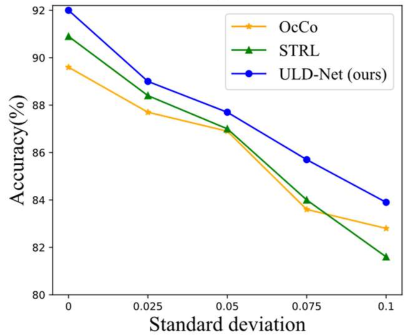

Figure 8. Results with Gaussian noise.

5. Discussion and Conclusion

In this paper, we propose a novel URL method for point cloud analysis. Our method extracts features by dense similarity learning, which is composed of instance-level and point-level similarity learning with the feature separability constraint. We also present the Equiv-Crop module to project point-level features from global to local scope to build correspondence across the transformed views. Without negative pairs, momentum encoder, or other complicated designs, ULD-Net pre-trains the network that extracts representations with the best results on linear SVM validation. After fine-tuning the pre-trained network on other downstream tasks including shape classification, shape part segmentation, and semantic segmentation, our ULD-Net also achieves competitive performances.

Though our ULD-Net can generalize representations across domains and achieve competitive results on real-world scene understanding tasks, there still exists a domain gap for transferring from synthetic to scene-level data due to the large point numbers and complicated structures. In the future, we will further explore how to extend our method to domain adaptive analysis of point clouds with the domain gap bridged. We hope the dense similarity learning, feature separability constraint, and Equiv-Crop module proposed could provide insights into future context-based discriminative URL methods.

Funding. National Natural Science Foundation of China (61701403); China Post-doctoral Science Foundation (2018M643719); Young Talent Support Program of the Shaanxi Association for Science and Technology (20190107).

Data availability. Data underlying the results presented in this paper are available in Ref. [31-33].

References

1. A. Oliver, S. Kang, B. C. Wünsche, and B. MacDonald, "Using the Kinect as a navigation sensor for mobile robotics," in *Proceedings of the 27th conference on image and vision computing New Zealand*, 2012), 509-514.
2. X. Lu, X. Mao, H. Liu, X. Meng, and L. Rai, "Event Camera Point Cloud Feature Analysis and Shadow Removal for Road Traffic Sensing," IEEE Sensors Journal **22**, 3358-3369 (2022).
3. C. Rausch, M. Nahangi, C. Haas, and J. West, "Kinematics chain based dimensional variation analysis of construction assemblies using building information models and 3D point clouds," Automation in Construction **75**, 33-44 (2017).

4. Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018), 206-215.
5. 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 *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (IEEE, 2019), 10441-10450.
6. P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*, (PMLR, 2018), 40-49.
7. R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019), 7203-7212.
8. H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021), 9782-9792.
9. S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," in *European conference on computer vision*, (Springer, 2020), 574-591.
10. S. Huang, Y. Xie, S.-C. Zhu, and Y. Zhu, "Spatio-temporal self-supervised representation learning for 3d point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021), 6535-6545.
11. J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," Advances in Neural Information Processing Systems **32**(2019).
12. O. Poursaeed, T. Jiang, H. Qiao, N. Xu, and V. G. Kim, "Self-supervised learning of point clouds via orientation estimation," in *2020 International Conference on 3D Vision (3DV)*, (IEEE, 2020), 1018-1028.
13. A. Thabet, H. Alwassel, and B. Ghanem, "Self-supervised learning of local features in 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020), 938-939.
14. C. Sun, Z. Zheng, X. Wang, M. Xu, and Y. Yang, "Point cloud pre-training by mixing and disentangling," arXiv preprint arXiv:2109.00452 (2021).
15. A. Sanghi, "Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning," in *European Conference on Computer Vision*, (Springer, 2020), 626-642.
16. B. a. Du, X. Gao, W. Hu, and X. Li, "Self-contrastive learning with hard negative sampling for self-supervised point cloud learning," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021), 3133-3142.
17. J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, and M. Gheshlaghi Azar, "Bootstrap your own latent-a new approach to self-supervised learning," Advances in neural information processing systems **33**, 21271-21284 (2020).
18. X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021), 15750-15758.
19. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017), 652-660.
20. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," Advances in neural information processing systems **30**(2017).
21. Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," Advances in neural information processing systems **31**(2018).
22. 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 Transactions On Graphics (tog) **38**, 1-12 (2019).
23. G. Te, W. Hu, A. Zheng, and Z. Guo, "Rgcnn: Regularized graph cnn for point cloud segmentation," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018), 746-754.
24. R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Proceedings of the IEEE international conference on computer vision*, 2017), 863-872.
25. H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021), 16259-16268.
26. 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 **7**, 187-199 (2021).
27. T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, (PMLR, 2020), 1597-1607.
28. K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020), 9729-9738.
29. M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," Advances in Neural Information Processing Systems **33**, 9912-9924 (2020).
30. A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748 (2018).
31. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, *3D ShapeNets: A deep representation for volumetric shapes* (2015), pp. 1912-1920.
32. 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 Trans. Graph. **35**(2016).
33. 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016), 1534-1543.
34. A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," (2015).
35. L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of machine learning research **9**(2008).
36. Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019), 8895-8904.