Upsampling Autoencoder for Self-Supervised Point Cloud Learning

Jian Shi  
School of Media and Design, Hangzhou Dianzi University  
Hangzhou, China  
sand2sand@hdu.edu.cn

Zizhao Wu*  
School of Media and Design, Hangzhou Dianzi University  
Hangzhou, China

*Corresponding author: wuzizhao@hdu.edu.cn

Abstract—While a large number of supervised learning methods have been proposed to handle the unordered point clouds and achieved remarkable success, their performance is limited to the costly data annotation. In this work, we propose a novel self-supervised pre-training model for point cloud learning without human annotations, which relies solely on upsampling operation to perform feature learning of point cloud in an effective manner. The key observation of our approach is that upsampling operation encourages the network to capture both high-level semantic information and low-level geometric information of the point cloud. As a result, the downstream tasks such as classification and segmentation will benefit from the pre-trained model. And our method outperforms previous methods in shape classification, part segmentation.

Keywords-component: Computer Vision, Point Cloud Learning, Self-Supervised Learning, Point Cloud Upsampling

I. INTRODUCTION

In view of the great success of deep learning on other computer vision tasks, many endeavors have been made to adapt the deep learning technologies to the analysis of 3D point clouds [4], [8], [14], [17], [22], [23], [30], [31], [36], [38], [44] including PointNet [22], VoxNet [45], etc. However, most of the existing works are supervised and rely on large-scale and accurate annotated dataset, which hinder their applicability.

Depending on the models they used, existing unsupervised approaches for learning on point clouds can be roughly classified into two categories: reconstruction-based and generation-based methods. The generation-based methods [1], [9] typically employed the Generative Adversarial Networks (GANs) [1] or Variational Auto-Encoders (VAEs) [9] to learn feature representations in an unsupervised framework. The reconstruction-based models [25] usually adopted a framework that trains an encoder to learn shape representations by reconstructing the input data via a decoder. Though these models have been demonstrated to be effective in certain applications, they usually fail to acquire high-level structure information [2].

Recently, contrastive learning [2], [5], [7], [11], [12], [24], [26], [35], [42], a class of unsupervised pre-training methods, is gaining certain interest in the point cloud learning. By adopting these pre-trained models for point cloud learning, these approaches are capable of predicting high-level structure information and with good performance. However, most contrastive learning approaches are not efficient enough, as they need a careful treatment of negative pairs by either relying on large batch sizes [35],[42], memory banks [12], or customized mining strategies [5], [12] to retrieve the negative pairs. Furthermore, their performance critically depends on complicated 3D data augmentations, e.g., cuboid [42], shape disorganizing [3] and shearing [6].

In this work, we propose a simple but effective self-supervised pre-training model, namely UAE, as illustrated in Figure 1, which achieves higher performance than sota self-supervised methods without using negative pairs or data augmentations. Specifically, our method first conducts the random subsampling from the input point cloud at a low proportion. Then, we feed them into an asymmetric encoder-decoder architecture, where the encoder is devised to operate only on the subsampled points, along with an upsampling decoder is adopted to reconstruct the original point cloud based on the learned features. Finally, we leverage a joint loss function to enforce the upsampled points to be similar with the original points and uniformly distributed on the surface.

Our contributions can be summarized as follows:

- presents a novel self-supervised pre-training method for point cloud learning based on pre-trained upsampling model.
investigates a novel upsampling architecture, where a joint loss function is adopted to enforce the upsampled points to be similar to the original point cloud.

• achieves significant performance in shape classification, part segmentation and point cloud upsampling tasks.

II. METHOD

Suppose that the original point cloud with \( N \) points is denoted by \( \mathcal{X} \in \mathbb{R}^{N \times C} \). In the simplest setting of \( C = 3 \), each point contains 3D coordinates. We aim to train a model that is capable of unsupervised learning the point cloud representations. Toward this goal, we propose upsampling autoencoder (UAE), a simple self-supervised learning approach that reconstructs the original point cloud from a small number of subsampled points. Unlike classical self-supervised methods, our UAE is suggested to capture high-level semantic information without any data augmentation and negative pairs. The overall structure of UAE is shown in Figure 2.

In what follows, we begin by describing the strategy that makes up our subsampled points. Then we detail how to learn a encoder \( \phi(\cdot) \) and an upsampling decoder \( \phi(\cdot) \).

A. Subsampling

Given a point cloud \( \mathcal{X} \in \mathbb{R}^{N \times C} \), we first subsample \( \mathcal{X} \) into a lower resolution subset \( \mathcal{P} \in \mathbb{R}^{rN \times 3} \), where \( r \) denotes that subsampling ratio. As shown in Figure 5, there exists a variety of sampling methods: 1) Random Sampling, where the probability of sampling each point in the point cloud follows a uniform distribution; 2) Farthest Point Sampling, where each point to be sampled is as far away as possible from points in the sampled set; 3) Local Sampling, where the points are sampled from a local part of the point cloud. In this paper, we propose to randomly sample a subset of points at a low proportion uniformly (e.g., \( r = 0.125 \)), which largely eliminates redundancy, thus creating a rather challenging task that cannot be easily handled by extrapolation from subsampled neighboring points [10]. This highly sparse point cloud is more conducive for us to design an effective encoder to capture high-level semantic information.

B. Encoder

The encoder \( \phi(\cdot) \) takes the coordinates of the subsampled point clouds \( \mathcal{P} \) as input and outputs high-dimensional features. In contrast to previous self-supervised methods, where the global shape representation is learned by the encoder, we perform point-wise feature extraction on subsampled point cloud \( \mathcal{P} \), where each point can learn its representation by upsampling decoding the original point cloud to reveal the local structure around it.

Generally, any deep learning-based network that takes point clouds as the input and outputs high-dimensional features can be adopted as the encoder \( \phi(\cdot) \) of UAE. However, as 3D points have some special properties such as irregularity and permutation invariance, we cannot directly leverage the convolutional neural network (CNN). Therefore, we deploy the dynamic graph CNN (DGCNN[31]) to overcome this limitation. In particular, we adopt the EdgeConv layer in DGCNN as our basic feature extraction block. By performing EdgeConv, our encoder can aggregate the features of neighbor points to the center point and update the feature of the center point.

Figure 3. The overview of feature up (top) and feature down (bottom) blocks, where \( r \) is the subsampling ratio and \( D \) is the dimension of point feature. We construct our upsampling decoder based on these blocks.
Specifically, given an input subsampled points \( \mathcal{P} = \{ p_j \}_{j=1}^{\mathcal{N}} \subseteq \mathbb{R}^3 \), we first apply K-Nearest Neighbor (KNN) algorithm to construct a local graph in the feature space. The KNN algorithm based on the feature space can efficiently and effectively find two points with the most similar semantics, i.e., the points on the two wings of the airplane, which have similar semantics. Then, the EdgeConv layer performs feature transformation on the \( r \mathcal{N} \) points and output the updated point features of \( p_i \). The output feature of a point \( p_i \) is

\[
f_i = \max_{p_j \in \mathcal{N}(p_i)} \text{ReLU} \left( h_\theta(p_i, p_i - p_j) \right)
\]

where \( p_j \in \mathcal{N}(p_i) \) denotes that point \( p_j \) belongs to the neighborhood of point \( p_i \), \( h_\theta \) denotes that the learnable parameters of MLP.

C. Decoder

The decoder \( \phi(\cdot) \) takes the point-wise features as input and outputs the upsampled point cloud \( \hat{\mathcal{P}} \). To upsample a point cloud, PU-Net [40] has been proposed to duplicate the point features and then separate MLPs have been employed to process each copy independently. However, the expanded features would be too similar to the inputs, so affecting the upsampling quality. In recent years, MPU [32] has been proposed to break a \( 16 \times 16 \times 16 \) upsampling network into four successive \( 2 \times 2 \times 2 \) steps to progressively upsample points in multiple steps. MPU preserves superior upsampling details but usually requires a more complex training process.

Inspired by the PU-GAN [15], we design a novel upsampling decoder to expand the point features, which is mainly composed of feature up and feature down blocks. As shown in Figure 2, we first upsample the point features \( F_{in} \) (after a MLP layer) by \( \frac{1}{r} \) to generate \( F_{up} \) and downsample it to generate \( F_{down} \); then, instead of directly constructing the original point cloud \( \mathcal{X} \), we adopt residual learning to regress the per-point feature offset by calculating the difference between \( F_{up} \) and \( F_{down} \); Ultimately, we feed them into a feature up block and a MLP layer to restore the original point cloud \( \mathcal{X} \). Such a strategy that utilizes feature offset to self-correct the expanded features has two advantages: firstly, it facilitates the production of fine-grained features while avoiding tedious multi-step training; secondly, the features of the same object can be completely different with respect to rigid transformations. Therefore, relative offsets are generally more robust. In the following, we detail the design choices of feature up and feature down blocks.

Feature up block. To upsample the point features \( \frac{1}{r} \) times, we adopt the commonly-used variation expansion operator [15], [16] by duplicating \( F_{in} \) with \( \frac{1}{r} \) copies and concatenating with a regular 2D grid. However, such operator may introduce redundant information or extra noise [16]. To rectify these problems, we propose to make use of the offset-attention [8] as the global refinement unit by considering the overall shape structure. The reason behind is that, compared with the widely-used self-attention unit [15], [16], [41], the offset-attention is generally more robust because it works by replacing the attention feature with the offset between the input of self-attention module and attention feature. The pipeline of this block is illustrated in Figure 3 (top).

Feature down block. To downsample the expanded features, we propose a novel GCN-based feature down block, illustrated in Figure 3 (bottom). Given the expanded features \( F_{up} \in \mathbb{R}^{r \mathcal{N} \times D} \), our method works in two steps. First, we reshape \( F_{up} \) of shape \( N \times D \) to \( r \mathcal{N} \times \frac{D}{r} \) and use one layer of EdgeConv to downsample \( F_{up} \) to shape \( r \mathcal{N} \times D \) using learnable parameters. Second, we feed them into a set of MLPs to regress the point features \( F_{down} \).

In contrast to previous works [15], [32], [40], our feature down block leverages GCNs, which are common modules for feature extraction. To the best of our knowledge, we are the first to introduce a GCN-based feature downsampling block. Our GCN design choice stems from the fact that GCNs enable our feature down block to encode spatial information from point neighborhoods and learn new features from the latent space rather than simply using Convs.

D. Joint Loss Function

Reconstruction loss. Our UAE reconstructs the original point cloud \( \mathcal{X} \) by predicting coordinate for each unsampled point. Each element in the decoder's output is a vector of coordinate values representing a point's spatial location. We formulate our objective function to encourage the geometric consistency between \( \mathcal{P} \) and \( \mathcal{X} \) in the point space:

\[
\mathcal{L}_{CD}(\hat{\mathcal{P}}, \mathcal{X}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \min_{x \in \mathcal{X}} \| \hat{p} - x \|_2^2 + \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \min_{p \in \mathcal{P}} \| x - \hat{p} \|_2^2
\]

where \( \mathcal{L}_{CD}(\cdot) \) means the Chamfer Distance (CD) to measure the average closest point distance between two point sets, \( |\cdot|_2 \) denotes the L2 distance between two points.

Repulsion loss. Since the reconstruction loss alone cannot ensure that the upsampled points over the underlying object surfaces will be uniformly distributed, which is important for capturing high-level shape information. To solve this issue, we use the repulsion loss [40] as the constraint term of uniformity distribution, which is represented as:

\[
\mathcal{L}_{rep} = \sum_{i=0}^{N} \sum_{\mathcal{P} \in \mathcal{N}(\hat{p}_i)} \delta \left( \| \hat{p}_i - \hat{p}_j \| \right) \omega \left( \| \hat{p}_i - \hat{p}_j \| \right)
\]

where \( \mathcal{N}(\hat{p}_i) \) is the point set of the k-nearest neighbors of point \( \hat{p}_i \), \( N \) is the number of upsampled points and \( |\cdot|_2 \) is the L2-norm. \( \delta(m) = -m \) is the repulsion term, which is a decreasing function to penalize \( \hat{p}_i \) if \( \hat{p}_i \) is located too close to other points in \( \mathcal{N}(\hat{p}_i) \). We further use the fast-decaying weight function \( \omega(m) = e^{-m^2/k^2} \) as the finite support radius [19]) to penalize \( \hat{p}_i \) only when it is too close to its neighboring points.

Overall loss function. The overall loss function can be calculated as the weighted sum of all the terms described above:

\[
\mathcal{L}_{total} = \alpha \mathcal{L}_{CD} + \beta \mathcal{L}_{rep}
\]

where \( \alpha \) and \( \beta \) are the weighting factors for the loss functions of reconstruction and repulsion, respectively. Following [40], we set \( \alpha = 1 \) and \( \beta = 0.01 \).
III. EXPERIMENTS

A. Setups

Pre-training Dataset. We train our UAE on ShapeNet dataset [46], which consists of 57,448 synthetic 3D CAD models organized into 55 categories with a further 203 subcategories, organized according to WordNet synsets. For pre-training we use the normalized version of ShapeNet, where all shapes are consistently aligned and normalized to fit inside a unit cube.

Evaluation Metrics. For the classification task on ModelNet10 and ModelNet40 datasets, we use the overall accuracy (OA) as the metric. On ShapeNet Part dataset, we evaluate our scheme with part classification accuracy and mean Intersection-over-Union (mIoU). For each sample, IoU is computed for each part that belongs to that object category. The mean of all part IoUs is regarded as the IoU for that sample.

B. Shape Classification

Dataset. We utilize ModelNet40 [34] and ModelNet10 [34] for shape classification task. We follow the same data split protocols of PointNet-based methods [31] for these two datasets. For ModelNet40, the train set has 9,840 models and the test set has 2,468 models, and the dataset consists of 40 categories. For ModelNet10, 3,991 models are for fine-tuning and 908 models for testing. It contains 10 categories. We follow the experimental configuration [22]: (1) we uniformly sample 1,024 points from the mesh faces for each model; (2) the point cloud is re-scaled to fit the unit sphere; and (3) the (x,y,z) coordinates of the sampled points are used in the experiment. During the training process, randomly scaling and perturbing the objects are adopted as the data augmentation strategy in our experiment.

Table 1. Shape Classification Results on ModelNet40 and ModelNet10. The quantitative results of SOTA unsupervised and supervised fine-tuning methods. “Unsupervised Transfer Learning” denotes the parameters of the pre-trained encoder are fixed on downstream tasks, and “Supervised Fine-tuning” denotes the pre-trained encoders are fine-tuned on target tasks.

| Model    | ModelNet40 | ModelNet10 |
|----------|------------|------------|
| Supervised Learning |            |            |
| PointNet [22] | 89.2       | -          |
| PointNet++ [23] | 91.9       | -          |
| PointConv [33] | 92.5       | -          |
| RGCNN [27] | 90.5       | -          |
| PointCNN [17] | 92.2       | -          |

Implementation Details. The global max pooling and average pooling layer are deployed in our classification head to acquire a 1,296-dimensional global feature vector. Three layers of linear projection with dropout ratio of 50% are used to get the final classification score. For unsupervised transfer learning, we fix the parameters of encoder and only train the classification head. During training, a random translation in [-0.2, 0.2], a random anisotropic scaling in [0.67, 1.5] and a random input dropout were applied to augment the input data. During testing, no data augmentation or voting methods were used. The batch sizes were 32, 200 training epochs were used and the initial learning rates were 10^{-3}, with a cosine annealing schedule to adjust the learning rate at every epoch. For supervised fine-tuning, we fully fine-tuned the pre-trained model on ModelNet40 and ModelNet10 datasets.

Results. The classification results are presented in Table 1. When utilizing DGCNN as the encoder, our method outperforms most of previous unsupervised counterparts and the results on ModelNet10 and ModelNet40 are comparable to certain fully-supervised models. Since the pre-training of the encoder is based on different datasets, the results demonstrate that our framework has a strong generalizability, which is regarded as a significant advantage of self-supervised representation learning. Notably, most of the classes are unseen

Table 2. We present the part segmentation evaluation results on ShapeNet Part, where mIoU refers to mIOU.

| Model               | mIoU  | Are | Ba | Ca | Car | Chai | Ear | Phon | Gui | Knif | Lam | Lapto | Moto | Mu | Pisto | Rocke | Skate | Tabl |
|---------------------|-------|-----|----|----|-----|------|-----|------|-----|------|-----|-------|------|----|-------|-------|-------|-----|
| Supervised Learning |       |     |    |    |     |      |     |      |     |      |     |       |      |    |       |       |       |     |
| KDNet [13]          | 82.3  | 80.1| 74.| 74.| 70. | 88.6 | 73.5| 90.2 | 87.2| 81.0 | 94.9| 57.4  | 86.  | 78.1| 51.8  | 69.9  | 80.3  |
| PointNet            | 83.7  | 83.4| 78.| 82.| 74. | 89.6 | 73.0| 91.5 | 85.9| 80.8 | 95.3| 65.2  | 93.  | 81.2| 57.9  | 72.8  | 80.6  |
| PointNet++ [23]     | 85.1  | 82.4| 79.| 87.| 77. | 90.8 | 71.8| 91.0 | 85.9| 83.7 | 95.3| 71.6  | 94.  | 81.3| 58.7  | 76.4  | 82.6  |
| P2Sequence[2]       | 85.1  | 82.6| 81.| 87.| 77. | 90.8 | 77.1| 91.1 | 86.9| 83.9 | 95.7| 70.8  | 94.  | 79.3| 58.1  | 75.2  | 82.8  |

| SpiderCNN [37]      | 92.4  |     |    |    |     |      |     |      |     |      |     |       |      |    |       |       |       |     |
| PointWeb [43]       | 92.3  |     |    |    |     |      |     |      |     |      |     |       |      |    |       |       |       |     |
| DGCNN [31]          | 92.9  |     |    |    |     |      |     |      |     |      |     |       |      |    |       |       |       |     |

Unsupervised Transfer Learning
- FoldingNet [39] 88.4 94.4
- MAP-VAE [9] 90.1 94.8
- MTD-FC [29] 90.3 -
- GSIR [2] 90.3 -
- RS-DGCNN [25] 90.6 94.5
- GLR-RSCNN [24] 91.3 94.2
- GraphTER [6] 92.0 -
- PointDis [20] 92.3 95.3
- SSC-RSCNN [3] 92.4 95.0
- Ours-DGCNN 92.9 95.6

Supervised Fine-tuning
- DepthContrast [42] 91.3
- MTD-FC 93.1
- GLR-RSCNN 92.2 94.8
- ParAE-DGCNN 92.9
- SSC-RSCNN 93.0 95.5
- Ours-DGCNN 93.2 95.7

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
by the model during ShapeNet pre-training. Thus, the superior performance further demonstrates that our model has a good ability to generalize to novel classes.

To the best of our knowledge, the most important application of self-supervised learning methods is to make full use of a large number of unlabeled data and boost the performance of supervised learning methods. Thus, we pre-train the encoder with our framework on ShapeNet and fine-tune the segmentation head to propagate these features to each point dimensional global feature vector. And we design a experiment.

Unsupervised Transfer Learning

| Method     | 85.2 | 84.0 | 83.4 | 86.7 | 77.8 | 90.6 | 74.7 | 91.2 | 87.5 | 82.8 | 95.7 | 66.3 | 94.9 | 81.1 | 63.5 | 74.5 | 82.6 |
|------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| DGCNN      | 57.0 | 54.1 | 48.7 | 62.2 | 43.7 | 68.4 | 58.3 | 74.3 | 68.4 | 53.4 | 82.6 | 18.6 | 75.1 | 54.7 | 37.2 | 46.7 | 66.4 |
| MAP-VAE    | 68.0 | 62.7 | 67.3 | 73.3 | 58.1 | 77.1 | 67.3 | 84.8 | 77.1 | 60.9 | 90.8 | 35.8 | 87.1 | 64.2 | 45.0 | 60.4 | 74.8 |
| Graph-TER[6] | 81.9 | 81.7 | 68.1 | 85.1 | 74.1 | 88.1 | 68.9 | 90.6 | 86.6 | 80.0 | 95.6 | 56.3 | 90.0 | 80.8 | 55.2 | 70.7 | 79.1 |
| MID-FC[29] | 84.2 | 80.4 | 82.7 | 89.8 | 80.9 | 89.9 | 80.7 | 90.5 | 85.7 | 77.8 | 95.9 | 73.4 | 90.0 | 81.1 | 56.7 | 81.8 | 82.4 |
| Ours-DGCNN | 85.0 | 83.5 | 82.9 | 86.6 | 77.7 | 90.4 | 75.6 | 91.0 | 86.9 | 81.0 | 95.1 | 68.9 | 94.9 | 81.4 | 62.5 | 73.1 | 82.7 |

Supervised fine-tuning

| Method     | 85.2 | 84.0 | 83.4 | 86.7 | 77.8 | 90.6 | 74.7 | 91.2 | 87.5 | 82.8 | 95.7 | 66.3 | 94.9 | 81.1 | 63.5 | 74.5 | 82.6 |
|------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| SSC-RSCNN  | 85.3 | 84.1 | 84.7 | 90.9 | 80.0 | 91.5 | 87.0 | 83.2 | 95.8 | 71.6 | 94.0 | 82.6 | 60.0 | 77.9 | 81.8 | 0.0  | 0.0  |
| Self-Sup[25] | 85.3 | 84.1 | 84.7 | 90.9 | 80.0 | 91.5 | 87.0 | 83.2 | 95.8 | 71.6 | 94.0 | 82.6 | 60.0 | 77.9 | 81.8 | 0.0  | 0.0  |
| PointDis[20] | 85.3 | 84.4 | 83.7 | 91.0 | 75.2 | 91.6 | 88.2 | 83.5 | 96.1 | 65.5 | 94.0 | 79.6 | 58.0 | 76.2 | 82.8 | 0.0  | 0.0  |
| OcCo[28] | 85.5 | 84.4 | 83.7 | 91.0 | 75.2 | 91.6 | 88.2 | 83.5 | 96.1 | 65.5 | 94.0 | 79.6 | 58.0 | 76.2 | 82.8 | 0.0  | 0.0  |
| MID-FC[29] | 85.5 | 83.6 | 82.9 | 91.0 | 81.5 | 91.8 | 87.1 | 79.3 | 95.7 | 68.7 | 95.0 | 83.6 | 68.3 | 82.7 | 83.2 | 0.0  | 0.0  |
| Ours-DGCNN | 85.6 | 84.7 | 80.9 | 86.3 | 91.3 | 76.2 | 91.2 | 89.6 | 82.1 | 96.7 | 68.9 | 94.9 | 83.8 | 65.3 | 79.5 | 83.1 | 0.0  |

Results. As shown in Table 2, the proposed UAE outperforms all other unsupervised point cloud learning models by a large margin, which indicates that our pre-trained model captures more effective semantic information that can transfer well to downstream segmentation tasks. Some results are visualized in Figure 4. We also conduct the object part segmentation experiments under supervised fine-tuning strategy and make comparisons with previous excellent models, such as PointDis, MID-FC and OcCo in Table 2. As shown, our UAE model (Ours-DGCNN) fine-tuned on 100% labeled samples achieves state-of-the-art performance. Compared to the supervised learning framework DGCNN, our pre-trained model (Ours-DGCNN) achieves remarkable performance improvements, which demonstrates the advantage of our pre-training strategy.

C. Part Segmentation

Dataset. We use the large-scale 3D dataset ShapeNet Parts [18] as the experiment bed. ShapeNet Parts contains 16,880 models (14,006 models are used for training, and 2874 models are used for testing), each of which is labeled with two to six parts, and the entire dataset has 50 different part labels. We sample 2,048 points from each model as input, with a few points having six labeled parts. We directly adopt the same train–test split strategy similar to DGCNN [31] in our experiment.

Implementation Details. The global max pooling and average pooling layer are also deployed to acquire a 1,296-dimensional global feature vector. And we design a segmentation head to propagate these features to each point hierarchically. Like shape classification task, three layers of linear projection with dropout ratio of 50% are also used to get the final classification score of each point. We optimize our networks via SGD with batch size 32. The learning rate of unsupervised transfer learning setting starts from $10^{-2}$ and decreases to $10^{-4}$, and the learning rate of supervised fine-tuning setting decays from $10^{-1}$ to $10^{-3}$.

For unsupervised transfer learning, we fix the parameters of encoder and only train the segmentation head. For supervised fine-tuning, we fully fine-tuned the pre-trained model on ShapeNet Part dataset.

Effect of subsampling ratio. We conduct an ablation study to analyze the setting of subsampling ratio $r$. The results are shown in Table 3. The best performance is achieved when $r$ is set to 12.5%. When the ratio becomes smaller, the lack of key points will result in a performance decline. On the contrary, when the ratio increases, a lot of noises at the local boundary lead to the deformation of feature extraction ability, and then reduces the accuracy of the model. Meanwhile, the task with high subsampling ratio (25% or 50%) can be easily solved by our UAE, which is not conducive to capture high-level semantic information.

Impact of loss function. We also investigate the options of loss functions. The results are shown in Table 4. can better encourage the output points to be located close to the underlying object surfaces [40]. However, in Table 4, Furthermore, we can see that the performance of the model is significantly improved after adding repulsion loss, especially under the combination of “CD + RL,” where the results from the output points are marked semantically and distinguished in position to reduce noise.
Figure 4. We compare our part segmentation results with the ground truth. From top to bottom: ground truth results and the prediction results of our model.

Figure 5. Various subsampling strategies. We subsample 256 points from original 2048 points (a) by performing FPS (b), LS (c) and random sampling (d). Form this figure, we can see that random sampling retains less geometric details.

Table 3. Supervised finetuning results (% and mIoU) on ModelNet40 and ShapeNet Part datasets with different subsampling rates

| Subsampling ratio (r) | ModelNet40 | ShapeNet |
|-----------------------|------------|----------|
| 5.0%                  | 91.5       | 83.9     |
| 12.5%                 | 93.2       | 85.6     |
| 25.0%                 | 92.6       | 85.2     |
| 50.0%                 | 92.5       | 84.9     |
| 100.0%                | 92.2       | 84.7     |

Table 4. Ablation study on various loss functions. CD: replace our joint loss function with chamfer distance loss in our architecture. EMD: replace the joint loss function with Earth’s moving distance loss in our architecture. EMD + RL: replace our joint loss function with Earth’s moving distance and the repulsion losses.

| Loss function | ModelNet40 | ShapeNet |
|---------------|------------|----------|
| CD            | 92.1       | 84.5     |
| EMD           | 92.2       | 84.7     |
| EMD+RL        | 93.1       | 85.5     |
| CD+RL         | 93.2       | 85.6     |

IV. CONCLUSION AND FUTURE WORK

In this work, we presented UAE, a framework for self-supervised point cloud learning. UAE learns high-level semantic information by upsampling a sparse point cloud uniformly within a simple and effective framework, without using negative pairs and data augmentations. We showed state-of-the-art results of our method on various downstream tasks including shape classification and object segmentation. In the future, it would be interesting to further exploit other applications, benefit the future point cloud works, and take one closer step to the harsh real-world setting, i.e., limited annotations. In addition, we will continue to study the point cloud pre-training methods on large-scale datasets, and focus on finding an efficient way to take advantage of the large-scale point data.

REFERENCE

[1] P Achlioptas, O Diamanti, I Mitliagkas, and L. Guibas. (2018) Learning representations and generative models for 3d point clouds. International Conference on Machine Learning.
[2] Haochan Chen, Shitong Luo, Xiang Gao, and Wei Hu. Unsupervised learning of geometric sampling invariant representations for 3d point clouds. In International Conference on Computer Vision, pages 893–903.
[3] Ye Chen, Jinxian Liu, Bingbing Ni, Hang Wang, Jiancheng Yang, Ning Liu, Teng Li, and Qi Tian. Shape self-correction for unsupervised point cloud understanding. In International Conference on Computer Vision, pages 8382–8391.
[4] Zhang Cheng, Haocheng Wan, Xinyi Shen, and Zizhao Wu. (2021) Patchformer: A versatile 3d transformer based on patch attention. CoRR, abs/2111.00207.
[5] Bi’an Du, Xiang Gao, Wei Hu, and Xin Li. (2021) Self-contrastive learning with hard negative sampling for self-supervised point cloud learning. In ACM Multimedia, pages 3133–3142.
[6] Xiang Gao, Wei Hu, and Guo-Jun Qi. (2020) Grahter: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 7163–7172.
[7] Jeffrey Gu and Serena Yeung. (2021) Staying in shape: Learning invariant shape representations using contrastive learning. Conference on Uncertainty in Artificial Intelligence.
[8] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R.Martin, and Shi-Min Hu. (2021) Pct: Point cloud transformer. Computational Visual Media, 7(2):187–199.
[9] Zhizhong Han, Xiuyang Wang, Yu-Shen Liu, and Matthias Zwicker. (2019) 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 International Conference on Computer Vision, pages 10441–10450.
[10] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. (2021) Masked autoencoders are scalable vision learners.
[11] Jincen Jiang, Xuequan Lu, Wanli Ouyang, and Meili Wang. (2021) Unsupervised representation learning for 3d point cloud data. arXiv preprint arXiv:2110.06632.
[12] Li Jiang, Shaoshuai Shi, Zhuoao Tian, Xin Lai, Sha Liu, Chi-Wing Fu, and Jiaya Jia. (2021) Guided point contrastive learning for semi-supervised point cloud semantic segmentation. In International Conference on Computer Vision, pages 6423–6432.
[13] Roman Klokov and Victor Lempitsky. (2017) Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. Proceedings of the IEEE International Conference on Computer Vision, pages 863–872.
[14] L. Landrieu and M. Boussaha. (2020) Point cloud oversegmentation with graph-structured deep metric learning. Conference on Computer Vision and Pattern Recognition (CVPR).

[15] Ruihui Li, Jianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Peng-Ann Heng. (2019) PU-GAN: A point cloud upsampling adversarial network. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 7202–7211.

[16] Ruihui Li, Jianzhi Li, Peng-Ann Heng, and Chi-Wing Fu. (2021) Point cloud upsampling via disentangled refinement. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 344–353.

[17] Y. Li, R. Bu, M. Sun, and B. Chen. (2018) Pointcnn: Convolution on x-transformed points. Advances in Neural Information Processing Systems.

[18] Y. Li, et al. (2016) A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (TOG), 35(6):210.1–210.12.

[19] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. (2007) Parameterization-free projection for geometry reconstruction. In ACM SIGGRAPH, page 22–ex.

[20] Fayao Liu, Guosheng Lin, and Chaun-Sheng Foo. (2021) Point discriminative learning for unsupervised representation learning on 3d point clouds. CoRR, abs/2108.02104.

[21] X. Liu, Z. Han, Y. S. Liu, and M. Zwicker. (2018) Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. AAAI.

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. (2017) Pointnet: Deep learning on point sets for 3d classification and segmentation. Conference on Computer Vision and Pattern Recognition (CVPR), pages 652–660, 2017.

[23] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems.

[24] Yongmin Rao, Jiwen Lu, and Jie Zhou. (2020) Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 5376–5385.

[25] Jonathan Sauder and Bjørnar Sievers. (2019) Self-supervised deep learning on point clouds by reconstructing space. Advances in Neural Information Processing Systems, 32:12962–12972.

[26] Chao Sun, Zhedong Zheng, Xiaohan Wang, Mingliang Xu, and Yi Yang. (2021) Point cloud pre-training by mixing and disentangling. arXiv preprint arXiv:2109.00452.

[27] G. Te, W. Hu, Z. Guo, and A. Zheng. (2018) Rgcn: Regularized graph cnn for point cloud segmentation. In ACM Multimedia.

[28] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. (2021) Unsupervised point cloud pre-training via occlusion completion. In International Conference on Computer Vision, pages 9782–9792.

[29] Peng-Shuai Wang, Yu-Qi Yang, Qian-Fang Zou, Zhirong Wu, Yang Liu, and Xin Tong. (2020) Unsupervised 3d learning for shape analysis via multiresolution instance discrimination. ACM Trans. Graphic.

[30] Wenhai Wang, et al. (2021) Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. International Conference on Computer Vision.

[31] Y. Wang, et al. (2020) Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics, 38(5).

[32] Yifan Wang, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine Hornung. (2019) Patch-based progressive 3d point set upsampling. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 5958–5967.

[33] W. Wu, Z. Qi, and L. Fuxin. (2019) Pointconv: Deep convolutional networks on 3d point clouds. Conference on Computer Vision and Pattern Recognition (CVPR).

[34] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. (2015) 3d shapenets: A deep representation for volumetric shapes. Conference on Computer Vision and Pattern Recognition (CVPR).

[35] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. (2020) Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In European Conference on Computer Vision, pages 574–591. Springer.

[36] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. (2020) Grid-Fcn for fast and scalable point cloud learning. Conference on Computer Vision and Pattern Recognition (CVPR), pages 5561–5570.

[37] Y. Xu, T. Fan, M. Xu, Z. Long, and Q. Yu. (2018) Spidercnn: Deep learning on point sets with parameterized convolutional filters. European Conference on Computer Vision.

[38] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui. (2020) Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. Conference on Computer Vision and Pattern Recognition (CVPR).

[39] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. (2018) Foldingnet: Point cloud auto-encoder via deep grid deformation. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 206–215.

[40] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Peng-Ann Heng. (2018) Pu-net: Point cloud upsampling network. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 2790–2799.

[41] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. (2019) Selfattention generative adversarial networks. In International Conference on Machine Learning, pages 7354–7363.

[42] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. (2021) Self-supervised pretraining of 3d features on any point-cloud. arXiv preprint arXiv:2101.02691.

[43] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. (2019) Pointweb: Enhancing local neighborhood features for point cloud processing. Conference on Computer Vision and Pattern Recognition (CVPR), pages 5565–5573.

[44] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. (2021) Point transformer. International Conference on Computer Vision.

[45] Yin Zhou and Oncel Tuzel. Voxelnet: (2018) End-to-end learning for point cloud based 3d object detection. International Conference on Computer Vision, pages 4490–4499.

[46] Angel X Chang, et al. (2015) Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.