Structural Point Cloud Data Recovery to Learning 3D Feature Representation

Ryosuke Yamada[1,2], Ryu Tadokoro[2], Yue Qiu[2], Hirokatsu Kataoka[2], and Yutaka Satoh[1,2]

[1] University of Tsukuba, Japan
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan
{ryosuke.yamada, ryu.tadokoro, qiu.yue hirokatsu.kataoka, yu.satou}@aist.go.jp

Abstract. Can we obtain 3D feature representations by reconstructing structural point clouds without real data? This paper aims to develop Point Cloud Perlin Noise (PCPN) for pre-training for 3D object recognition. PCPN is automatically generated based on the natural 3D structure in the real world. Using a simple formula that is based on Perlin noise, our proposed method can automatically generate more 3D patterns than conventional 3D datasets. In addition, we apply Point-MAE [1] to PCPN in order to construct a pre-trained model for improving the performance of downstream tasks. Through experiments, we demonstrate that our proposed method improves performance by 1.5% for ModelNet40 compared to the conventional 3D datasets used for pre-training. Our proposed pre-training strategy has also revealed the ability to achieve effective pre-training in 3D object recognition without real data and supervised labels.

Keywords: 3D object recognition, Point cloud, Self-supervised learning

1 Introduction

3D object recognition using point clouds has been extensively studied for autonomous driving and robotics applications. Point clouds can better represent real-world environments than 2D images, as it is not dependent on the camera's viewpoint. We can capture rich 3D data using high-precision sensors such as LiDAR systems, and therefore, 3D object recognition has received considerable attention [2,3,4].

However, one challenge associated with the construction of 3D point cloud datasets is the considerable amount of human effort required for the collection of 3D data and for annotation. Collecting 3D data requires 3D scanning or reconstruction from 2D images and annotation considering position and orientation in 3D space. Thus, constructing a 3D dataset demands more human effort than 2D datasets. In addition, missing 3D data collected from the real world often adversely affects the learning of 3D feature representation. Previous research has used 3D Computer Aided Design (CAD) models to solve these issues for

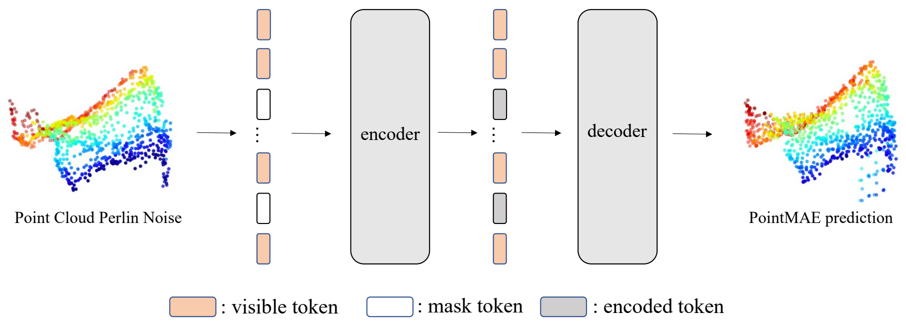

Fig. 1. Overview of the point cloud Perlin noise pre-training process. The pre-training converts the input point clouds with a random embedding (e.g., 60%) as mask tokens and other embeddings as visible tokens. The encoder processes only visible tokens. The mask tokens are added to the input sequence of the decoder to the predict token and reconstruct the point clouds. The parameters learned in the pre-training are set as initial parameters for finetuning the downstream tasks.

training [5]. The advantage of this approach is the annotation is automatic and does not require data collection. However, we expect that increasing the size of a dataset in the future will be pretty costly.

Recently, increasing attention has been given to the use of artificially generated images and automatically generated labels for learning feature representations [7]. For example, the research strategy in "Learning to See by Looking at Noise" involves generating images according to specific rules and using self-supervised learning to learn visual feature representations from the images. Researchers have conducted studies using simple shapes with "Dead Leaves" and natural images generated with StyleGAN [6] and have successfully acquired superior feature representations compared to learning from random parameters. This framework can also help acquire feature representations in 3D object recognition and not just for 2D images.

In this paper, we propose an automatic 3D dataset construction method called Point Cloud Perlin Noise (PCPN), as see Figure 1. Which pre-training tasks should be designed using the PCPN? In 2023, a self-supervised learning method called point-MAE [1] was proposed. point-MAE is based on BERT [8], which is a pre-text task proposed in natural language processing. BERT masks words in sentences and restores the original sentences, whereas point-MAE deliberately creates masked data from input 3D point clouds and reconstructs the original data. Using Perlin Noise, it is possible to generate structured 3D data based on predefined rules. Therefore, we can acquire general 3D feature representations more easily than those observed in the real world. Because 3D scans from the real world initially include missing parts in the 3D data, these could be better for reconstruction tasks, such as those that use Point-MAE. However, since the 3D data generated from equations do not have missing parts in the 3D structure, the pre-training strategy of point-MAE would be optimized for PCPN.

2 Related Work

2.1 3D object recognition with point cloud

The notion of 3D object recognition using point clouds has attracted attention since the emergence of PointNet [2], which made it possible to input point clouds into neural networks directly. Since then, a large number of 3D object recognition methods that used Graph Neural Networks (GNN) [11,12] and Convolutional Neural Networks (CNN) [9,10] have been proposed to improve recognition accuracy. Transformer models have been drawing attention in 2023; however, they require a larger amount of training data because of their lower inductive bias. In 2D image recognition, Vision Transformer [13] achieved state-of-the-art performance by pre-training on the JFT-300M [14], which is a dataset with 300 million images and is not available to the public. However, constructing a 3D dataset is costlier than building a 2D image dataset. For 3D object recognition using point clouds, there are no large-scale datasets similar to JFT-300M. In this paper, we propose a framework for generating a theoretically infinite amount of 3D data based on the regularity of natural phenomena. Our proposed method improves the pre-training effectiveness of the Transformer-based network. We believe it is also possible to construct a large-scale pre-training dataset that can become a de-facto standard.

2.2 Self-supervised learning

The construction of a 3D dataset requires a huge amount of human effort. Therefore, supervised labels and training data are expected to be limited when applied to applications. From the perspective of real-world applications, it would be ideal for improving the performance of 3D object recognition with limited data. Self-supervised learning learns pre-text tasks from unlabeled 3D data to learn 3D feature representations while the annotation cost is reduced. In particular, self-supervised learning with 3D point clouds can be classified into two main categories: reconstruction tasks based on generative models [15,16] and contrast learning based on the similarity of the input data [17,18]. In the case of reconstruction tasks, the network is trained to reconstruct the parts of 3D objects that have been deliberately masked using an Encoder-Decoder network. In the case of contrastive learning, utilizes different viewpoints of 3D scenes, learning to identify correspondences between two point clouds. A pre-trained model learn by reconstruction tasks is used for object classification or segmentation tasks. Contrariwise, the contrastive learning approach is used for 3D object detection and 3D scene segmentation in 3D scenes.

3 Point Cloud PerlinNoise

In this section, we explain the automatic construction method of PCPN that is based on Perlin noise in Section 3.1. We describe the pre-training strategy that uses point-MAE in Section 3.2.

3.1 Point Cloud Perlin Noise

In this paper, we propose Point Cloud Perlin Noise (PCPN) based on the simplex noise method [22]. To construct the PCPN, we take two steps: (i) generating Perlin noise and (ii) projecting into a 3D space.

Generating Perlin noise: To generate the Perlin noise map, we divide a 2D space into a grid of equilateral triangles with a side length of 1. For a given coordinate (x, y), we refer to the three vertices of the equilateral triangle grid that surround (x, y) as $P = \{(x_i, y_i)\}_{i=1}^{3}$. We use a hash function H to calculate the pseudo-random gradient vectors $(Gradx_k, Grady_k)$ for each vertex as follows:

$$(Gradx_k, Grady_k) = H(x_k, y_k) \tag{1}$$

We then define a function C that depends on the gradient at a coordinate (x_k, y_k) as follows:

$$f(x_k, y_k) = \max\left(0, \frac{1}{2}(1 - 2{x_k}^2 - 2{y_k}^2)\right)^4 \tag{2}$$

$$C(x_k, y_k) = (x_k Gradx_k + y_k Grady_k) f(x_k, y_k) \tag{3}$$

The noise N at a coordinate (x, y) in the 2D space is calculated as follows.

$$N(x, y) = \sum_{i=1,2,3} C(x_i, y_i) \tag{4}$$

Projecting into a 3D space: To project the Perlin noise map into a 3D space, we use the noise value at each coordinate on the 2D Perlin noise as the value on the z-axis and map the points in the 3D space. In this way, we can generate 3D point cloud based on Perlin noise.

3.2 Pre-training with Point-MAE

In this paper, we develop a pre-trained model by applying Point-MAE to PCPN. The Point-MAE reported the highest accuracy in self-supervised learning for 3D object recognition. We consider that PCPN can theoretically generate data infinitely and has no missing data, making it suited for reconstruction tasks such as Point-MAE. In the following, we introduce PointMAE's pre-training method. The pre-training process for Point-MAE involves three steps: (i) dividing into patches, (ii) masking and patch embedding, and (iii) autoencoder pre-training.

projecting into a 3D space: We downsample point clouds by using farthest point sampling and determine m center points. For each center point, we select k nearest neighbor points using the k-nearest neighbor (KNN) algorithm and group them into patches $PT \in \mathbb{R}^{m \times k \times 3}$. The point clouds in each patch overlap, and the coordinates within each patch are normalized by the center point.

Masking and patch embedding: We randomly select n out of the m patches created during Stage (i) and create masked patches $PT_{mask} \in \mathbb{R}^{n \times k \times 3}$. For unmasked patches $PT_{unmasked} \in \mathbb{R}^{(m-n) \times k \times 3}$, we use the lightweight Point-Net to convert them into $E_{unmasked} \in \mathbb{R}^{(m-n) \times C}$. C is the number of dimensions of the tokens. The mask token is the shared weight token $TK_{mask} \in \mathbb{R}^{(m-n)\times C}$. The encoder and decoder are composed of standard transformer blocks. The positional embedding adopts the coordinates of the center points of each patch converted by a multi-layer perceptron (MLP), which is added to each transformer block in the encoder and decoder. In the encoder, $E_{unmasked}$ is the input, and the encoded token $TK_{unmasked} \in \mathbb{R}^{(m-n)\times C}$ is the output. Further, in the decoder, the output of the encoder $TK_{unmasked}$ and the masked token T_{mask} are combined as the input, and only T_{mask} is decoded to output the token $F_{mask} \in \mathbb{R}^{n\times C}$. By inputting F_{mask} into the MLP as the head, we obtain the reconstructed patch $PT_{pred} \in \mathbb{R}^{n\times k\times 3}$. During training, we minimize the loss function shown in the following equation:

$$Ł = \frac{1}{|PT_{pred}|}\sum\nolimits_{p\in PT_{pred}} \min_{q\in PT_{mask}} \|p-q\|_2^2 + \frac{1}{|PT_{mask}|}\sum\nolimits_{q\in PT_{mask}} \min_{p\in PT_{pred}} \|p-q\|_2^2 \tag{5}$$

4 Experiments

In this section, we verify the effectiveness of PCPN pre-training in 3D object recognition through comparative experiments. In Section 4.1, we explain the experimental setup. In Section 4.2, we describe the fine-tuning experiments for each downstream task.

4.1 Experiment setting

Pre-training setup. In this experiment, we follow Point-MAE and adopt the PointCloud Transformer [4] as the network. In this paper, we use PCPN and ShapeNet to pre-train Point-MAE. We divide PCPN into three subsets of data {1,000, 10,000, 100,000} and evaluate them for fine-tuning performance. ShapeNet is a commonly used dataset in conventional self-supervised learning and consists of 55 object categories and 51,300 3D CAD models. We use AdamW as the optimizer and the cosine learning rate decay during pre-training. We set the initial learning rate to 0.001, weight decay to 0.05, and batch size to 256 and conducted the training for 300 epochs.

Fine-tuning setup. In the downstream task, we evaluate the pre-trained model with regard to object classification, few-shot learning, and part segmentation. For object classification, we use ModelNet40 [20] and ScanObjectNN [21] as the evaluation datasets. ModelNet40 is a single-object dataset consisting of 40 categories, 9,843 samples for train, and 2,468 samples for test. ScanObjectNN is a real-world dataset consisting of 15 categories, 2,312 samples for train, and 581 samples for test. We verify our proposed method on three subsets {OBJ-BG, OBJ-ONLY, PB-T50-RS} of ScanObjectNN. The OBJ-BG contains the background around the object, while OBJ-ONLY is the 3D object without a background. The PB-T50-RS is a subset with translation, rotation (about the

Table 1. The comparison for PCPN and ShapeNet on the object classification.

Pre-train	ModelNet40	ScanObjectNN		
		OBJ-BG	OBJ-ONLY	PB-T50-RS
From scratch [16]	91.4	79.8	80.5	77.2
ShapeNet	92.1	83.5	86.9	87.4
PCPN-1k	92.8	84.2	87.2	82.5
PCPN-10k	92.9	80.2	81.9	82.4
PCPN-100k	92.5	80.7	81.4	82.2

Table 2. The comparison for PCPN and ShapeNet on few-shot learning.

Pre-train	5-way, 10-shots	5-way, 20-shots	10-way, 10-shots	10-way, 20-shots
From scratch [16]	87.8 ±5.2	93.3 ±4.3	84.6 ±5.5	89.4 ±6.3
ShapeNet	97.3 ±1.9	96.8 ±2.1	91.7 ±4.3	93.4 ±2.7
PCPN-100k	95.0 ±3.0	95.3 ±5.6	89.6 ±4.1	92.6 ±2.8

gravity axis), and scaling for all 3D objects. For few-shot learning, we randomly select K classes from ModelNet40 and sample $N + 20$ objects for each class. We train the model using the K-way $\times$ N-shots subset and evaluate it using the 20K samples. In this study, we prepare ten subsets with K={5, 10} and N={10, 20} and evaluate the performance by averaging the highest accuracy for each subset. Part segmentation is a difficult task identifying finer class labels for all points of 3D models. We evaluate it using ShapeNetPart [19], which includes 16,881 models from 16 categories. Following previous studies, we sample 2,048 points as the input for each object and generate 128-point patches. We evaluate the performance using the mean Intersection over Union (mIoU) for all instances and the IoU for each category.

4.2 Downstream tasks

In this section, we explain the experimental results of the downstream tasks. We evaluate our proposed method by verifying it using benchmark datasets for object classification, few-shot learning, and part segmentation.

Object Classification. We investigate and evaluate the transfer learning performance by setting ModelNet40 and ScanObjectNN as the downstream datasets. Table 1 shows the experimental results for object classification. As can be seen in the table, an increase in the amount of data in PCPN leads to improved classification accuracy for ModelNet40 and ScanObjectNN. Furthermore, when comparing the PCPN-10k pre-trained model with scratch training, the classification accuracy improves by +1.5% for ModelNet40. This result confirms the effect of pre-training with PCPN. The classification accuracy was also improved over the scratch in ScanObjectNN. Furthermore, the performance is comparable to that of the ShapeNet pre-trained model. The results show that our proposed pre-training model is applicable to both synthetic and real data in the object classification task.

Table 3. Part segmentation results on the ShapeNetPart dataset. We report the mean IoU across all part categories mIoUC (%) and the mean IoU across all instances mIoUI (%), as well as the IoU (%) for each category.

	mIoUC	mIoUI	aero	bag	cap	car	chair	earphone	guitar	knife
			lamp	laptop	motor	mug	pistol	rocket	skateboard	table
From scratch [16]	83.4	85.1	82.9	85.4	87.7	78.8	90.5	80.8	91.1	87.7
			85.3	95.6	73.9	94.9	83.5	61.2	74.9	80.6
ShpeNet	85.5	83.1	83.1	85.1	87.0	77.9	90.6	79.8	91.1	86.3
			85.6	94.8	71.8	93.7	83.4	61.8	72.9	82.0
PCPN-100k	83.6	85.8	84.1	83.2	87.7	79.9	91.2	77.8	91.4	87.0
			85.7	95.9	73.8	94.6	83.8	61.6	76.6	82.0

Few-shot Learning. In the experimental results for object classification, we evaluate the PCPN-100k pre-trained model with regard to few-shot learning. As shown in Table 2, the PCPN-100k pre-trained model also demonstrates effectiveness for few-shot learning. Specifically, we compare it to learning from scratch in four subsets and confirm performance improvements of 7.2%, 2.0%, 5.0%, and 3.2%, respectively. This result suggests that the PCPN model pre-trained with PointMAE acquires a universal 3D feature representation that can quickly adapt to new downstream tasks without the use of any real data or manual annotations. However, the results were inferior for ShapeNet pre-trained model. We speculate that the factor is that ShapeNet is created from a 3D CAD model, which is the same 3D data domain as ModelNet40 for fine-tuning.

Part Segmentation. In this experiment, we evaluate the pre-trained model using the ShapeNetPart, which contains 16,881 objects from 16 categories. As shown in Table 3, the PCPN-100k pre-trained model achieved a mIoUI of 85.8%, outperforming the result from scratch training by 0.7% mIoU. Furthermore, the performance is equivalent to the ShapeNet pre-trained model. Based on these results, it can be inferred that the PCPN pre-trained model is also effective for more challenging tasks such as part segmentation.

5 Conclusion

This paper proposes a method for automatically constructing a 3D point cloud dataset (PCPN) without collecting 3D data or requiring human labeling. The experimental results show that the PCPN pre-trained model achieved a performance that was equivalent to the datasets used in conventional pre-training methods. Furthermore, our proposed method could improve the shortage of 3D data and the diversity of 3D datasets. Based on these results, we can confirm the effectiveness of the PCPN pre-training strategy. In this paper, we proposed PCPN to create a pre-trained model method as an initial consideration. However, we believe that designing the most effective learning method for PCPN will further affect pre-training in the future.

References

1. Pang, Yatian, et al., “Masked Autoencoders for Point Cloud Self-supervised Learning,” in *European Conference on Computer Vision (ECCV)*, 2022.
2. Qi, Charles R., et al., “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation,” in *Computer Vision and Pattern Recognition (CVPR)*, 2017.
3. Qi, Charles R., et al., “Deep Hough Voting for 3D Object Detection in Point Clouds,” in *International Conference on Computer Vision (ICCV)*, 2019.
4. Zhao, Hengshuang, et al., “Point Transformer,” in *International Conference on Computer Vision (ICCV)*, 2019.
5. Yongming Rao, et al., “RandomRooms: Unsupervised Pre-training from Synthetic Shapes and Randomized Layouts for 3D Object Detection,” in *International Conference on Computer Vision (ICCV)*, 2021.
6. Karras, Tero, et al., “A Style-Based Generator Architecture for Generative Adversarial Networks,” in *Computer Vision and Pattern Recognition (CVPR)*, 2019.
7. Baradad Jurjo, Manel, et al., “Learning to See by Looking at Noise,” in *Neural Information Processing Systems (NeurIPS)*, 2021.
8. Devlin, Jacob, et al., “Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *arXiv* , 2018.
9. Li, Yangyan, et al., “PointCNN: Convolution On X -Transformed Points,” in *Neural Information Processing Systems (NeurIPS)*, 2021.
10. Xu, Yifan, et al., “SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters,” in *European Conference on Computer Vision (ECCV)*, 2018.
11. Shi, Weijing, et al., “Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud,” in *Computer Vision and Pattern Recognition (CVPR)*, 2020.
12. Landrieu, Loic, et al., “Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs,” in *Computer Vision and Pattern Recognition (CVPR)*, 2018.
13. Dosovitskiy, Alexey, et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” in *International Conference on Learning Representations (ICLR)*, 2021.
14. Sun, Cen, et al., “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era,” in *International Conference on Computer Vision (ICCV)*, 2019.
15. Wang, Hanchen, et al., “Unsupervised Point Cloud Pre-training via Occlusion Completion,” in *International Conference on Computer Vision (ICCV)*, 2021.
16. Yu, Xumin, et al., “Point-bert: Pre-training 3D Point Cloud Transformers with Masked Point Modeling,” in *Computer Vision and Pattern Recognition (CVPR)*, 2022.
17. Xie, Saining, et al., “PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding,”in *European Conference on Computer Vision (ECCV)*, 2020.
18. Hou, Ji, et al., “Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts,” in *Computer Vision and Pattern Recognition (CVPR)*, 2021.
19. Chang, X., Angel, et al., “ShapeNet: An Information-Rich 3D Model Repository,” in *arXiv*, 2015.
20. Wu, Z., et al., “3D ShapeNets: A Deep Representation for Volumetric Shapes,” in *Computer Vision and Pattern Recognition (CVPR)*, 2015.
21. Angelina, Mikaela, Uy, et al., “Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data,” in *International Conference on Computer Vision (ICCV)*, 2019.
22. Perlin, K. 2002. Noise hardware. In Course Notes from Siggraph 2002, Course 36: Real-Time Shading Languages, SIGGRAPH 2002, ACM.