IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 August 2024, accepted 17 September 2024, date of publication 30 September 2024, date of current version 9 October 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3470971*

RESEARCH ARTICLE

# Masked Structural Point Cloud Modeling to Learning 3D Representation

**RYOSUKE YAMADA[1,2], RYU TADOKORO[2], YUE QIU[2], HIROKATSU KATAOKA[2], AND YUTAKA SATOH[1,2]**

[1]Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8577, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba 305-8560, Japan

Corresponding author: Ryosuke Yamada (s2230162@s.tsukuba.ac.jp)

**ABSTRACT** Pre-training for 3D object recognition typically requires a large-scale 3D dataset to learn effective 3D geometric representations. However, constructing such datasets is costly due to the extensive 3D data collection and human annotation required. This paper explores a synthetic pre-training approach that learns 3D geometric representations by reconstructing structural point clouds without relying on real data or human annotation. We propose the Point Cloud Perlin Noise (PCPN) dataset, which is an automatically generated point cloud dataset that uses Perlin noise to simulate natural 3D structures found in the real world. The proposed method enables the rapid generation of diverse 3D geometric patterns using a simple Perlin noise-based formula, significantly reducing the human effort typically involved in creating conventional 3D datasets. We applied PointMAE to the PCPN dataset for pre-training, demonstrating improved performance in downstream tasks such as 3D shape classification and part segmentation. Our experiments showed that the proposed pre-trained model outperformed a model trained from scratch on ModelNet40 by 1.4%. In addition, our pre-training strategy proves effective for 3D object recognition without requiring real data or supervised labels. This study highlights that Perlin noise can capture 3D structural properties and that the diversity of geometric patterns is crucial for learning effective 3D geometric representations.

**INDEX TERMS** Deep learning, computer vision, 3D object recognition, point cloud, transfer learning, self-supervised learning, formula-driven supervised learning.

## I. INTRODUCTION

3D object recognition using point clouds is gaining attention in computer vision due to its potential in applications such as autonomous driving, industrial robotics, virtual reality, and augmented reality. The advantage of point clouds is that their data structure is invariant to camera viewpoint differences and can express fine-grained geometric shapes. Thus, 3D object recognition using point clouds offers a more robust and detailed representation of 3D geometric structures in the real world compared to images. In addition, 3D sensors (*e.g.,* LiDAR) have gradually become more affordable and can capture data from real-world environments. Consequently, there is a growing demand for 3D object recognition models based on point clouds.

The associate editor coordinating the review of this manuscript and approving it for publication was Mingbo Zhao.

However, from the perspective of application, one of the main challenges is the cost of constructing a large-scale point cloud dataset. This challenge can be primarily divided into two aspects: (i) point cloud collection and (ii) human labeling. First, point cloud collection requires direct manual 3D scanning from real environments, the creation of 3D computer-aided design (CAD), or reconstruction from multi-view images. All of these approaches require an enormous amount of time to collect tens of thousands of point clouds. Moreover, except for CAD-generated objects, point clouds are relatively noisy and often include occlusions and missing data. Second, human labeling requires annotators to consider the position and orientation in 3D space for each point cloud. While images only require labeling on a 2D plane, annotating point clouds is generally more time-consuming because the positions of objects in 3D space must be considered. Thus, constructing a point cloud dataset requires considerably more

human effort and time compared to constructing an image dataset. In addition, missing point clouds adversely affect the learning of 3D geometric representations.

To address the aforementioned challenges in constructing point cloud datasets, self-supervised learning enables pre-training by automatically assigning pseudo labels to unlabeled point clouds [1], [2], [3], [4], [5], [6]. The pre-trained model using self-supervised learning has been reported to improve performance in various 3D vision tasks, such as 3D shape classification and part segmentation [6], [7], [8]. One of the latest self-supervised learning methods is the masked autoencoder (MAE)-based approach, which uses masked modeling inspired by bidirectional encoder representations from transformers (BERT) [7]. For example, PointMAE learns 3D geometric representations by reconstructing masked embeddings from input point clouds [7]. Thus, self-supervised learning is a promising approach to alleviating the challenges of point cloud dataset construction in 3D vision.

However, self-supervised learning is thought to be reaching a plateau due to the limitations of existing pre-training point cloud datasets in terms of the quantity and variation of training data. Currently, self-supervised learning relies on a 3D CAD object dataset called ShapeNet [9], which consists of 51,300 objects across 55 categories, as a pre-training dataset. ShapeNet is extremely small in scale compared to datasets like ImageNet [10] and Pictures without humAns for Self-Supervised Pretraining (PASS) [11], which are used for self-supervised learning in image recognition. Given that point cloud and annotations are manually collected, expanding the size of point cloud datasets in the future will remain just as costly.

Thus, it seems ideal to develop a pre-training approach without the costs of point cloud collection and human labeling. A recent approach that is gaining importance is the artificial generation of synthetic data with automatic label assignment. For example, the research strategy in [12] generates synthetic images according to specific rules and employs self-supervised learning to extract visual feature representations from these images. Researchers have conducted studies using simple shapes with the 'dead leaves' model and natural images generated by StyleGAN [13], successfully acquiring superior feature representations compared to those learned from random parameters. We believe this concept can also facilitate the acquisition of feature representations for 3D object recognition, not just for images. Thus, this paper hypothesizes that the concept will be effective not only for image recognition but also for learning 3D geometric representations.

One of the frameworks used in the video recognition domain is based on video Perlin noise. Kataoka et al. [14] automatically generated simple motions using Perlin noise [15] and achieved effective initialization for training a 3D spatiotemporal convolutional network. In this approach, the spatial domains of $x, y$ and the temporal domain of $t$ are considered to have different properties, and the simple motions generated by Perlin noise may not have been sufficient as pre-training data for action recognition. Therefore, this paper investigates whether a structured point cloud in the $x, y, z$ domains can be used to acquire 3D recognition features, replacing the task of video recognition with point cloud recognition. We generated Point Cloud Perlin Noise (PCPN) and explored the learning of geometric feature representations for 3D object recognition.

This paper proposes a pre-training strategy that involves the automatic construction of a point cloud dataset called PCPN, utilizing self-supervised learning, as shown in Figure 1. The PCPN is a point cloud dataset for pre-training, which automatically generates structured point clouds based on Perlin noise. The PCPN can be constructed more easily than real-world data. In addition, we utilized PointMAE [7] as a pre-training strategy because it does not suffer from missing parts in the 3D structure, unlike captured real-world point clouds. PointMAE is inspired by BERT [16], a pretext task proposed for natural language processing. BERT masks words in sentences and restores the original text, whereas PointMAE deliberately creates masked tokens from input point clouds and reconstructs the original data. Therefore, we build the PCPN pre-trained model for 3D object recognition using PCPN masked modeling.

The proposed method was compared to ModelNet40 [17], ScanObjectNN [18], and ShapeNetParts [19], as baseline frameworks for 3D shape classification and part segmentation, which are fundamental 3D vision tasks. As a result, the proposed PCPN pre-trained model performs similarly to or better than the previous baselines in 3D shape classification and part segmentation tasks. In particular, the proposed method outperforms learning from scratch on ModelNet40 by $+1.4\%$. The feature representations of the pre-trained models were compared using centered kernel alignment (CKA) similarity [20]. Visualization of CKA similarity revealed that the PCPN pre-trained model learns a feature representation similar to that of the ShapeNet pre-trained model. The PCPN is generated based on mathematical formulas and, at first glance, appears to be 3D data that differs from real-world data. Our experimental results suggest that PCPN is a useful pre-training dataset for 3D object recognition. Furthermore, the proposed method does not require manual data collection or annotation. The contributions of this paper are as follows.

- In this paper, point clouds were automatically generated based on Perlin noise, and PCPN was constructed as the pre-training dataset for PointMAE. PCPN can flexibly control geometric shapes by adjusting the parameter set.
- PCPN's point clouds were intentionally masked and reconstructed to the original point clouds using PointMAE to learn 3D geometric feature representations. The PCPN pre-trained model improves recognition performance in 3D shape classification, part segmentation, and 3D object detection, without requiring human effort.
- Visualization of CKA similarity metrics revealed that the feature representations of the PCPN pre-trained model are similar to those of the ShapeNet pre-trained model.

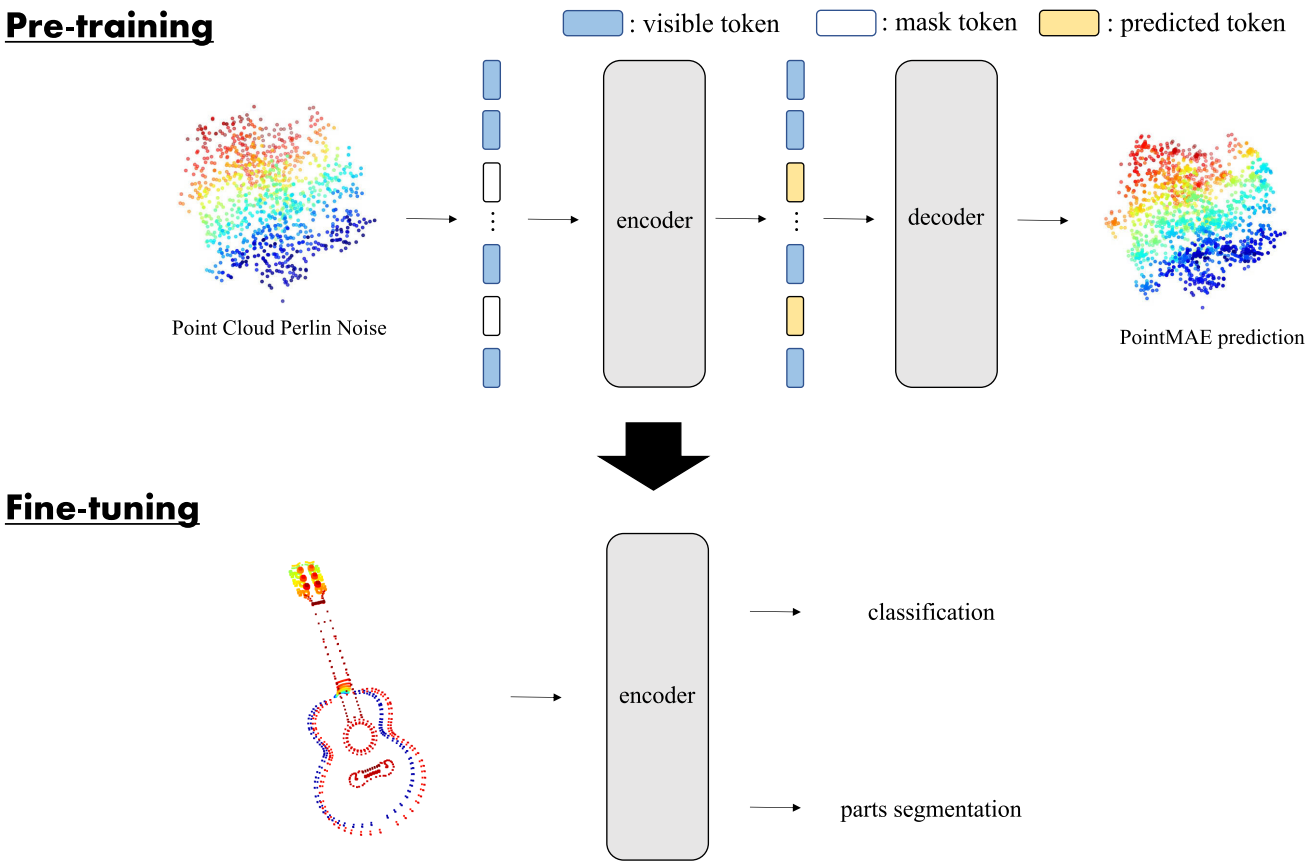


**FIGURE 1. Overview of Point Cloud Perlin Noise pre-training process. The pre-training converts the input point clouds with a random embedding (e.g., 60%) as mask tokens, and other embeddings as visible tokens. Encoder processes only visible tokens. Mask tokens are added to input sequence of decoder of prediction token to reconstruct point clouds. Parameters learned in pre-training are set as initial parameters for fine-tuning the downstream tasks.**

- 3D synthetic data generated from mathematical formulas are effective as a pre-training dataset for self-supervised learning, just like real or hand-crafted data such as 3D CAD objects.

The rest of this paper is organized as follows. Section II reviews the related work. Section III explains the construction of the PCPN and the pre-training strategy. The experimental settings, verification, additional experimental validation, and analysis are presented in Sections IV, V, VI, and VII respectively. Based on the experimental results, Section VIII discusses the proposed method based on the experimental results. Finally, Section IX summarizes the main contributions of this paper.

## II. RELATED WORK

### A. POINT CLOUD DATASETS

Most 3D datasets have been constructed to evaluate the performance of 3D object recognition models in various tasks. Existing 3D datasets can be broadly categorized into two types: (1) object-level 3D model datasets and (2) scene-level 3D scan datasets. While some urban-scale aerial 3D scene datasets also exist, this paper focuses primarily on object-level and scene-level datasets without describing them in detail. The following is an introduction to the point cloud datasets in each category.

(1) Object-level 3D model datasets. These datasets are primarily used for 3D recognition tasks such as 3D shape classification and 3D part segmentation. Object-level 3D model data can be categorized into two types: (i) synthetic and (ii) real-world. (i) Synthetic datasets [9], [17] are generated from 3D CAD objects without occlusions or backgrounds. Creating 3D CAD objects requires specialized knowledge of CAD software, making it inaccessible to everyone. (ii) Real-world datasets [18] are captured using 3D sensors, such as depth cameras, and include occlusions, backgrounds, and noise. Real-world datasets tend to be smaller in size due to the significant human resources required compared to synthetic datasets.

(2) Scene-level 3D scan datasets. These datasets are used for 3D recognition tasks such as 3D object detection, semantic segmentation, and tracking. Scene-level 3D scan data can be categorized into two types: (i) indoor and (ii) outdoor datasets. (i) Indoor datasets [21], [22] are designed for 3D scene understanding in environments such as houses, schools, and universities, where each furniture object is assigned a semantic label, and objects must be recognized

based on their context within the 3D scene. In contrast, (ii) outdoor datasets [23], [24] are developed for autonomous driving, where objects are sparse and spatially well-separated. In most cases, outdoor data are point clouds captured by cars equipped with LiDAR sensors. 3D sensors used to collect data for indoor and outdoor scenes often differ, and each requires specialized post-processing. Furthermore, compared to a single 3D object, annotating 3D bounding boxes and labeling each point cloud with semantic tags is a complex and labor-intensive process.

These factors represent significant bottlenecks for point cloud datasets compared to images. We believe that one key direction for advancing 3D object recognition lies in constructing point cloud datasets without the need for manual data collection and human labeling.

### B. 3D OBJECT RECOGNITION NETWORKS WITH POINT CLOUDS

Compared to data such as images, videos, and voxels, point clouds present significant challenges when designing neural networks for learning 3D geometric representations. This is because point clouds are an irregular data format consisting of 3D coordinates, and neural network design for point clouds must account for two key properties: permutation invariance and shift equivariance. As a result, neural networks that can directly process point clouds have lagged behind those designed for other data types. In 2016, the first neural network to directly process point clouds, PointNet [25], was introduced. The introduction of PointNet marked a significant paradigm shift in 3D object recognition. PointNet addressed the aforementioned challenges, leading to the rapid development of 3D object recognition using point clouds in deep learning. It introduced an end-to-end network that directly processes point clouds, utilizing multi-layer perceptrons and max pooling to extract global information.

Since the introduction of PointNet, 3D object recognition models have been able to directly process point clouds, leading to significant progress in various 3D recognition tasks, such as 3D shape classification [26], [27], [28], [29], [30], [31], [32]. Recently, graph-based methods like dynamic graph CNN (DGCNN) [30] and graph attention convolution [31], which represent point clouds as graphs where each point is a node and relationships with other points form edges, have gained popularity. These methods are effective in capturing local geometric structures. In 2023, PointTransformer [33], recognized as a state-of-the-art 3D recognition model, was applied in various domains, from natural language processing to point clouds. Compared to conventional 3D object recognition models, PointTransformer can learn dependencies among features over long distances in the feature space.

However, transformers require a considerable amount of training data due to their lower inductive bias compared to other networks. For example, in image recognition, the Vision Transformer (ViT) [34] achieved state-of-the-art performance on ImageNet by pre-training on JFT-300M [35] in 2020. It should be noted that JFT-300M is a dataset containing millions of images that is not publicly available. In contrast, for 3D object recognition using point clouds, there are no large-scale 3D pre-training datasets comparable to JFT-300M. This paper focuses on PointTransformer and investigates whether synthetic 3D data generated from mathematical formulas can provide effective pre-training for transformer-based networks. By demonstrating that synthetic pre-training is possible without manually generating point clouds, this paper presents a new direction for synthetic pre-training in 3D object recognition.

### C. SELF-SUPERVISED LEARNING

Self-supervised learning enables 3D geometric representation learning by assigning pseudo labels to unlabeled 3D data [6], [7], [8], [36], [37], [38], [39], [40]. Its primary advantage is that it facilitates 3D geometric representation learning without the need for manual labeling, achieving performance comparable to supervised pre-training. From an application perspective, self-supervised learning is ideal for enhancing 3D object recognition performance when working with limited 3D data.

Recent self-supervised learning on point clouds can be categorized into two main types: (i) self-supervised learning on a single object and (ii) self-supervised learning on a 3D scene. (i) In self-supervised learning for a single 3D object, the most common approach involves performing a reconstruction task [7], [8], [40], [41]. For example, PointMAE [7] uses a pretext task that randomly masks regions in point clouds and reconstructs them to their original form. PointMAE achieved 93.8 (ii) For self-supervised learning in 3D scenes, the mainstream approach is to learn point cloud correspondences using contrastive learning [1], [37], [38], [39], [42]. For example, PointContrast [1] applies contrastive learning at the point cloud level to analyze 3D scenes from different viewpoints. On datasets like SUN RGB-D [22] and ScanNet, PointContrast was the first to demonstrate pre-training effectiveness in the context of self-supervised learning in 2020.

However, while self-supervised learning can reduce annotation costs, the scale of training data remains highly dependent on existing 3D datasets. As a result, even if a pretext task is designed to learn better 3D geometric representations, the learnable data distribution is still limited. Unlike image recognition and natural language processing, where large amounts of data can be easily collected from the web, the current method of constructing 3D datasets does not scale as easily, as described in Section II-A.

Thus, we believe that the baseline of self-supervised learning can be improved by constructing 3D datasets without human effort. This paper hypothesizes that 3D data generated from specific rules, such as Perlin noise, may be more suitable for PointMAE than 3D objects acquired from real-world or CAD data, as these generated 3D objects have fewer missing point clouds.

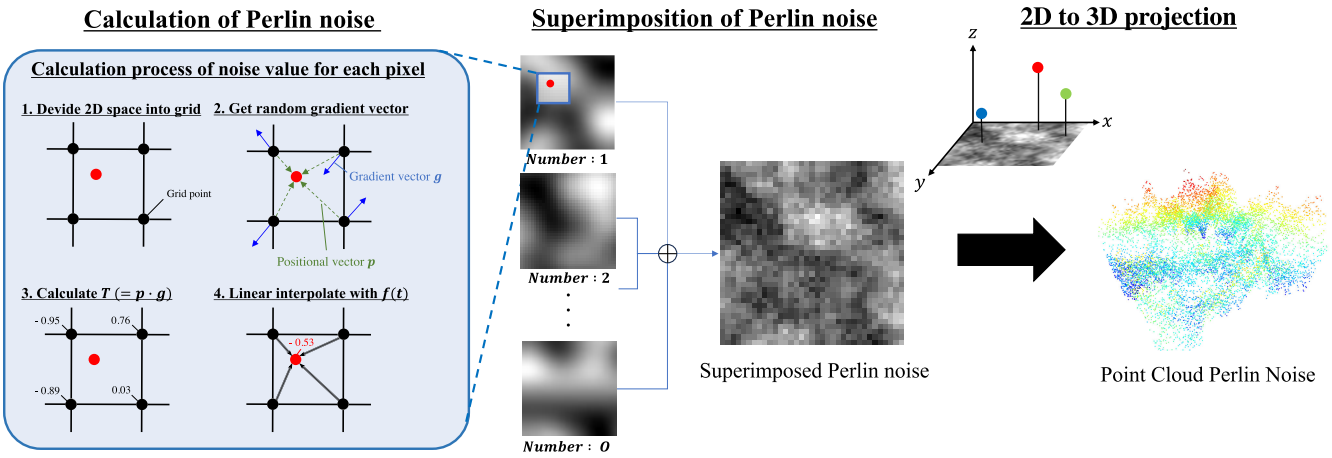


**FIGURE 2.** **Overview of the generation process of PCPN. Generation of PCPN consists of three steps: (i) calculation of Perlin noise, (ii) superimposition of Perlin noise (iii) 2D to 3D projection. The blue box on the left shows how to determine the noise value for each pixel. The generation of Perlin noise in (i) is repeated *O* times for the number of superimpositions in (ii). In (ii), *O* noises are superimposed according to the algorithm in Algorithm. 1. Superimposed Perlin noise is projected from 2D to 3D by transforming each noise value as a coordinate along the z-axis.**

## III. POINT CLOUD PERLIN NOISE

This section introduces the construction of a PCPN. Section III-A outlines the design concept of PCPN. Section III-B explains the generation of Perlin noise in point clouds. Finally, Section III-C describes the pre-training strategy for applying the PCPN to PointMAE.

### A. OVERVIEW

This paper proposes a PCPN, which is a 3D pre-training dataset for self-supervised learning. A PCPN is constructed to automatically generate a variety of point clouds from formulas based on Perlin noise, which mimics real-world patterns. Perlin noise is a type of noise commonly used to simulate natural phenomena in computer graphics and was proposed by Ken Perlin in 1985. Perlin noise exhibits four critical properties: (i) mimicking natural phenomena, (ii) scale invariance, (iii) local continuity, and (iv) controllability.

(i) The mimicry of natural phenomena refers to Perlin noise's ability to emulate and generate various patterns in the real-world. This results in the generation of point clouds that represent pseudo-real-world patterns. (ii) Scale invariance stems from the fractal nature of the Perlin noise, enabling it to exhibit self-similar patterns observable at different scales. This facilitated the generation of consistent point clouds. (iii) Local continuity refers to the property of Perlin noise, which generates locally consistent continuous values. This allows the generated point clouds to maintain diversity and consistency, which is challenging to achieve with other random noise methods such as white noise. (iv) Controllability refers to the adjustability of the generation of Perlin noise by modifying parameters such as noise frequency and amplitude. This allows the fine-tuning of the properties of the generated point clouds to align with specific tasks or objectives. Based on these properties, we propose a PCPN based on Perlin noise for a 3D pre-training dataset.

### B. GENERATION OF POINT CLOUD PERLIN NOISE

This section describes how the PCPN is generated, as shown in Figure 2. We use Perlin noise [15], [43], [44], [45] as the generation rule for a point cloud dataset. The point cloud dataset is defined in the PCPN as $D = \{p_i\}_N$, where $p_i$ is a single point cloud and $N$ is the total number of data points. $p_i$ is expressed as $p_i = \{(x_{il}, y_{il}, z_{il})\}_n$, where $n$ denotes the number of point clouds per instance. The point clouds $p_i$ are synthesized using the following steps. The PCPN generation consists of three steps: (1) calculation of the Perlin noise, (2) superimposition of the Perlin noise, and (3) 2D to 3D projection as shown in Figure 2. Details of each step are explained as described in the following sections.

#### 1) CALCULATION OF PERLIN NOISE

In the calculation of the Perlin noise step, the single noise value $s'_{(x,y)}$ at coordinates $c = (x, y) \in \mathbb{R}^2$ is determined on a 2D plane $H$. The single noise value $s'_{(x,y)}$ is computed multiple times and superimposed in step (ii) to compute the final noise value $s_{(x,y)}$. Space $H$ is divided into a regular square grid, and $L = \{(x_q, y_q)\}_{q=4}$ are assumed to be four grid points in the neighborhood of $c$. The set of coordinates $L$ is expressed as follows:

$$L = \{(|x|, |y|), (|x| + 1, |y|), (|x|, |y| + 1), (|x| + 1, |y| + 1)\}. \quad (1)$$

For each of the neighboring grid points $L$, we find the pseudo-random gradient vector $\mathbf{g}_{(x',y')}$ which is unique to the coordinates. Because the same gradient vector must be assigned identical coordinates, a hash function is set to derive the vector. Hash function $h$ consists of table $P$, which is a list of values from 0 to 255 in random order.

Hash Table $P$

| Index | Value |
|---|---|
| 0 | 157 |
| 1 | 72 |
| 2 | 238 |
| 3 | 55 |
| ... | ... |
| ... | ... |
| 254 | 201 |
| 255 | 8 |

Gradient Vector Table $G$

| Index | Vectors |
|---|---|
| 0 | (1,0) |
| 1 | (1,-1) |
| 2 | (-1,1) |
| 3 | (1,1) |
| 4 | (-1,-1) |
| 5 | (0,1) |
| 6 | (-1,0) |
| 7 | (0,-1) |

**FIGURE 3. Hash table *H* and gradient vector table *G*. We use hash table *P* and gradient vector table *G* to compute pseudo-random gradient vectors on each grid in the 2*D* plane, which are necessary for calculating Perlin noise at each pixel. In the hash table *P*, each index from 0 to 255 corresponds to a random value from 0 to 255. In gradient vector table *G*, vectors on the two-dimensional plane are assigned to the indices from 0 to 8.**

A unique value is computed for each coordinate using the hash function and the gradient vector is selected from table $G$, which contains the unit-length gradient vectors based on the calculated value. $P$ and $G$ are shown in Figure 3. The $j$-th element in $P$ and $G$ are denoted as $P[j]$ and $G[j]$, respectively. Using the modulo operation with the value of each coordinate as the dividend and 255 as the divisor, the gradient vector $\mathbf{g}_{(x',y')}$ unique to the coordinates is obtained as follows:

$$h_{(x',y')} = P[P[(x' mod\ 255)] + (y' mod\ 255)], \tag{2}$$

$$\mathbf{g}_{(x',y')} = G[h_{(x',y')} mod\ 8]. \tag{3}$$

At each grid point, the dot product of the position vector $\$\boldsymbol{p}_{(x',y')} = (x' - x, y' - y)\$$ and the gradient vector $\boldsymbol{g}_{(x',y')}$ is calculated, and this value is defined as $T$ as follows:

$$T(x', y') = \mathbf{p}_{(x',y')} \cdot \mathbf{g}_{(x',y')} \tag{4}$$

The single noise value $s'_{(x,y)}$ is derived by linearly interpolating $T$ at the surrounding grid points. The coefficient of linear interpolation is defined as $f(t)$. The noise value at $(x, y)$ is then derived using the following equation:

$$f(t) = 6t^5 - 15t^4 + 10t^3, \tag{5}$$

$$s'_{(x,y)} = Lerp\Big(Lerp(T_0, T_1, f(x)), Lerp(T_2, T_3, f(x)), f(y)\Big). \tag{6}$$

$T_q$ is equivalent to $T_{(x_q,y_q)}$ at grid point $(x_q, y_q)$ in $L$. $Lerp$ is a function for linear interpolation, $Lerp(u, v, w) = u+(v-u)w$.

#### 2) SUPERIMPOSITION OF PERLIN NOISE

As shown in the center of Figure 4, multiple layers of Perlin noise are derived by varying the frequency and amplitude and then combining them. This process result in Perlin noise that is more natural and complex. Here, in this paper, we selected three essential parameters—Octaves ($O$), Persistence ($V$), and Lacunarity ($U$)—as generation parameters in the Perlin noise procedure. The number of waves to be superimposed is referred to as Octaves ($O$); the scaling factor for the amplitude of the superimposed waves is termed Persistence ($V$); and the scaling factor for the frequency is called Lacunarity ($U$). These elements work together to increase the variation in the generated noise. The algorithm for this Perlin noise superposition is illustrated in Algorithm 1. Specifically, we set Octaves ($O$) to [1.0, 10.0], Persistence ($V$) to [0.01, 1.00], and Lacunarity ($U$) to [1.0, 5.0]. We determined these ranges experimentally, considering that if the parameters were set too low, the diversity of the Perlin noise would decrease, while if set too high, the noise could become overly chaotic.

The single Perlin noise $s'_{(x,y)}$ is superimposed through an iterative process, repeated for $O$ times. In each iteration, the final noise value $s_{(x,y)}$ is calculated by adding the weights as the amplitude, which corresponds to the weight of the resulting noise, and the frequencies of the coordinates are input into the Perlin function, as shown in Eq. 6. The final noise value at coordinates $c$ is computed using the function. Figure 4 illustrates the results of the PCPN when three parameters of Octave $O$, Persistence $V$, and Lacunarity $U$ are varied. By altering each parameter, it becomes possible to generate a more diverse array of geometrical shapes.

#### 3) 2D TO 3D PROJECTION

The noise value $s_{(x,y)}$ is calculated for each of $n$ coordinates in $\{(x_k, y_k)\}_n$ on the 2D plane $H$, as described previously. Each calculated $noise(x_k, y_k)$ is treated as a new $z$-axis coordinate $z_k$ and set $(x_k, y_k, z_k)$ are set as one of the point clouds in $p_k$. This process is repeated $N$ times to derive $D\{p_1, p_2, \ldots, p_k\}_N$. Using the above process of projection to 3D, complex and natural geometrical shapes can be generated, as shown in Figure. 4.

### C. PRE-TRAINING WITH PointMAE

In this paper, we develop a pre-trained model by applying PointMAE [7] to PCPN. PointMAE has the highest reported 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 suitable for reconstruction tasks such as PointMAE. The PointMAE pre-training method involves three steps: (1) dividing the 2D space into patches, (2) masking and patch embedding, and (3) autoencoder pre-training.

#### 1) PROJECTING INTO A 3D SPACE

We downsample point clouds using farthest-point sampling and determine $m$ center points. For each center point cloud, we apply the $k$-nearest neighbor algorithm and group the selected points 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 cloud.

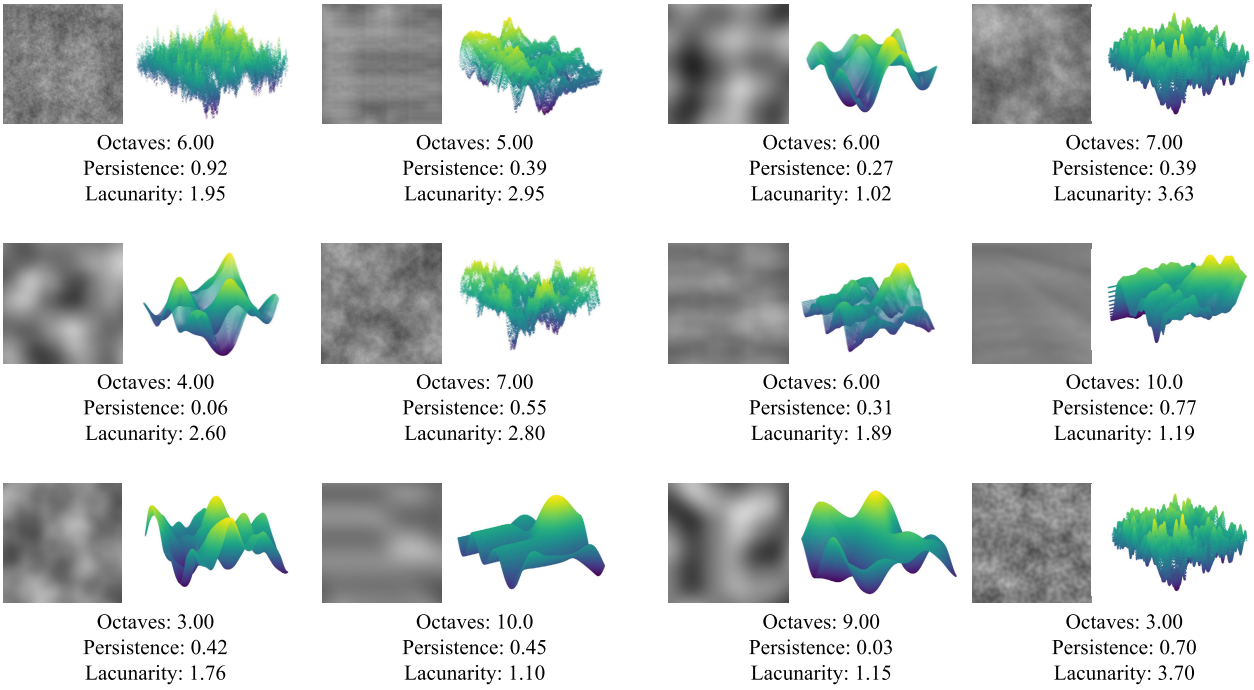


**FIGURE 4. Geometrical diversity of the superimposed Perlin noise. We superimpose multiple Perlin noises by the algorithm shown in Algorithm. 1 to generate more natural and complex Perlin noise. There are three parameters for the superposition of Perlin noise: Octaves, Persistence, and Lacunarity. The PCPN becomes more diverse in geometric structure by changing the value of each parameter.**

**Algorithm 1** Superimposition of Perlin Noise

0: **function** OctavePerlin ($x$, $y$, octaves, persistence)
0: total $\leftarrow$ 0
0: frequency $\leftarrow$ 1
0: amplitude $\leftarrow$ 1
0: maxValue $\leftarrow$ 0
0: **for** $i = 0$*to*octaves $-$ 1
0: total $\leftarrow$ total+perlin($x \times$ frequency, $y \times$ frequency) $\times$ amplitude
0: maxValue $\leftarrow$ maxValue + amplitude
0: amplitude $\leftarrow$ amplitude $\times$ persistence
0: frequency $\leftarrow$ frequency $\times$ 2
0: **end for**
0: finalNoise $s_{(x,y)} \leftarrow$ total/maxValue
0: **return** finalNoise $s_{(x,y)}$
0: **end function**=0

#### 2) 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 $PT_{unmasked} \in \mathbb{R}^{(r-n)\times k\times 3}$, we use the lightweight PointNet to convert them into $E_{unmasked} \in \mathbb{R}^{(r-n)\times C}$, where C is the number of dimensions for the tokens. The mask token is the shared weight token $TK_{mask} \in \mathbb{R}^{(r-n)\times C}$.

#### 3) AUTOENCODER PRE-TRAINING

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 $TK_{unmasked} \in \mathbb{R}^{(r-n)\times C}$ is the output. Furthermore, in the decoder, the output of the encoder $TK_{unmasked}$ and 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, the minimization of the loss function is given by

$$L = \frac{1}{|PT_{pred}|}\sum_{a\in PT_{pred}}\min_{b\in PT_{mask}}\|a-b\|_2^2 + \frac{1}{|PT_{mask}|}\sum_{b\in PT_{mask}}\min_{a\in PT_{pred}}\|a-b\|_2^2. \quad (7)$$

Figure 5 illustrates the input and output of PointMAE. It can be observed that PointMAE can predict the masked input shapes for both the PCPN and ShapeNet datasets. This paper aims to acquire pre-trained parameters that are beneficial for downstream tasks by performing self-supervised pre-training through such reconstruction tasks.

## IV. EXPERIMENTAL SETTING

This section describes the verification of the effectiveness of PCPN pre-training for 3D object recognition through comparative experiments. Section IV-A describes the pre-training setup. In Section V describes the fine-tuning setup for each downstream task.

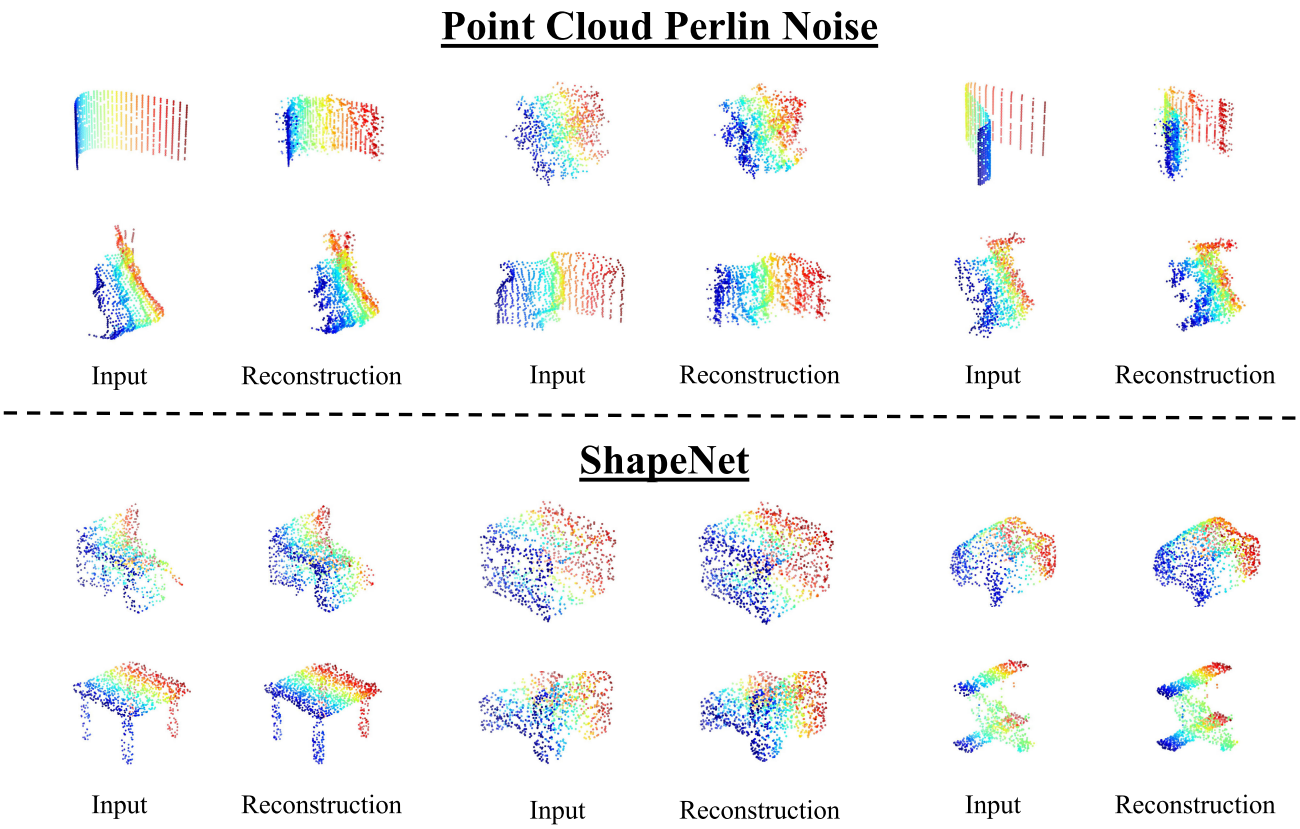


**FIGURE 5.** The input point clouds and reconstructed results of PointMAE pre-training. This visualization result shows the input point clouds and output results of pre-training PCPN and ShapeNet using PointMAE, respectively. This visualization result reveals that PointMAE can train PCPN and ShapeNet to restore the original point clouds.

### A. PRE-TRAINING SETUP

In this experiment, the PointMAE approach [7] was employed to pre-training the proposed PCPN using the PointCloud Transformer architecture [33]. All experiments were performed on a system equipped with four NVIDIA TITAN V100 GPUs, and PyTorch 1.8.0 was used as a deep learning framework. During the pre-training phase, a data augmentation strategy was applied that scaled and translated each point within a point cloud. The scaling factor and translation value were randomly chosen from a uniform distribution, with a scaling range of 2/3 to 3/2 and a translation range of 0.2.

To evaluate the performance of the proposed model after fine-tuning, three PCPN datasets are constructed at different scales: 1,000, 10,000, and 100,000, named PCPN-1k, PCPN-10k, and PCPN-100k, respectively. In addition, to assess the effectiveness of pre-training with the PCPN, we used the ShapeNet dataset for comparison. ShapeNet is a standard dataset commonly used in self-supervised learning tasks with single-object focus, such as PointMAE [17]. It consisted of 55 object categories and 51,300 3D CAD models.

To pre-train the PointCloud transformer, we employed AdamW as the optimizer and applied cosine learning rate decay to minimize the specific loss function during the training process. The selection of hyperparameters, including an initial learning rate of 0.001, a weight decay of 0.05, and a batch size of 256, was guided by the settings recommended in the original PointMAE paper [7]. The pre-training was performed for 300 epochs.

### B. FINE-TUNING SETUP ON EACH DOWNSTREAM TASK

In the downstream task, we evaluated the PCPN pre-trained model in comparison with ShapeNet pre-trained PointMAE and previous 3D object recognition methods in 3D shape classification, few-shot learning, and part segmentation.

#### 1) 3D SHAPE CLASSIFICATION

For 3D shape classification, we utilized two benchmark datasets: ModelNet40 [17] and ScanObjectNN [18]. ModelNet40, widely used for 3D object classification tasks, is a single-object dataset comprising 40 categories, including 9,843 samples for training and 2,468 samples for testing. On the other hand, ScanObjectNN is a real-world dataset design to handle the complexity and variety of real-world 3D objects, consisting of 15 categories, 2,312 samples for training, and 581 samples for testing. The performance of the proposed method was evaluated using ModelNet40

and three subsets of ScanObjectNN {OBJ-BG, OBJ-ONLY, and PB-T50-RS}. OBJ-BG includes the surroundings of an object, whereas OBJ-ONLY comprises a 3D object without any background. PB-T50-RS is a more challenging subset in which all 3D objects undergo translation, rotation (about the gravity axis), and scaling to simulate real-world scenarios in which objects may not be in standard orientations or scales.

In the fine-tuning phase, we used AdamW [46] as the optimizer with an initial learning rate of 0.005, which was then adjusted according to the cosine learning rate decay schedule across 300 epochs. Each pre-trained model was fine-tuned on point clouds with 1,024 points for ModelNet40 and 2,048 points for ScanObjectNN. In addition, to evaluate the performance of the 3D shape classification, we used the overall accuracy as an evaluation metric. In particular, we compared the highest accuracy achieved within 300 epochs as the performance metrics. We set up these experimental conditions and evaluation metrics in accordance with the PointMAE in order to make a fair comparison with the previous studies.

#### 2) FEW-SHOT LEARNING

In the few-shot learning experiments, we randomly selected $K$ classes from ModelNet40 and sampled $N + 20$ objects for each class. These $K$ classes were used to form the $K$-way, $N$-shot subsets used for training. Each pre-trained model was then evaluated on the remaining 20 objects per class. In this paper, we prepared 10 subsets by varying $K$ and $N$ with values of $\{5, 10\}$ and $\{10, 20\}$, respectively.

During the fine-tuning phase, we employed the AdamW optimizer, starting with a learning rate of 0.0005, which was then modulated following a cosine decay schedule over 150 epochs. ModelNet40 was fine-tuned using point clouds consisting of 1024 points. The performance was evaluated based on the mean and standard deviation of the highest accuracy achieved across each subset. We set up these experimental conditions and evaluation metrics in accordance with PointMAE to ensure a fair comparison with previous studies.

#### 3) PART SEGMENTATION

Part segmentation, which involves identifying finer class labels for all points of a 3D model, presents a significant challenge. For the evaluation, we used the ShapeNetPart [9] dataset, a benchmark specifically designed for part segmentation. It comprises 16,881 3D models spanning 16 diverse categories.

We evaluated performance using the mean intersection over union (mIoU) for all instances and IoU for each category. This paper reports two key metrics that together provide a comprehensive overview of the model's performance: mIoUC and mIoUI. The mIoUC measures the average overlap between the predicted and ground truth segmentations for each category, while the mIoUI averages the IoU across all instances, regardless of their category. We set up these experimental conditions and evaluation metrics in accordance with PointMAE to ensure a fair comparison with previous studies.

**TABLE 1. Comparison of PCPN and ShapeNet on object classification. This table shows the Accuracy (%) of the downstream tasks of object classification using ModelNet40 and ScanObjectNN. We evaluated ScanObjectNN under three conditions, following the method of Pang et.al. [7]. We also cite and compare the score of conventional methods (PointNet, PointNet++, and PointCNN).**

| Pre-training | ModelNet40 | ScanObjectNN | | |
|---|---|---|---|---|
| | | OBJ-BG | OBJ-ONLY | PB-T50-RS |
| PointNet [25] | 89.2 | 73.3 | 73.3 | 68.0 |
| SpiderCNN [47] | – | 77.1 | 79.5 | 73.7 |
| PointNet++ [48] | 90.7 | 82.3 | 84.3 | 77.9 |
| DGCNN [30] | 92.9 | 82.8 | 86.2 | 78.1 |
| PointCNN [26] | 92.5 | 86.1 | 85.5 | 78.5 |
| BGA-DGCNN [18] | – | – | – | 79.7 |
| BGA-PN++ [18] | – | – | – | 80.2 |
| GBNet [49] | – | – | – | 80.5 |
| PRANet [50] | – | – | – | 81.0 |
| From scratch | 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 |

## V. COMPARISON WITH PREVIOUS RESULTS

This section explains the experimental results for downstream tasks. The proposed method was evaluated by verifying it using benchmark datasets for 3D shape classification, few-shot learning, and part segmentation.

### A. 3D SHAPE CLASSIFICATION

Table 1 shows that the proposed PCPN-1k pre-trained model consistently outperformed learning from scratch regardless of the real or CAD data. This result indicates that the pre-trained framework is effective for 3D shape classification. Moreover, the PCPN-1k pre-trained model achieved higher performance than the previous baseline ShapeNet despite the 1/50 dataset size except for ScanObjectNN (PB-T50-RS). Specifically, for ModelNet40, the PCPN-1k pre-trained model outperformed the ShapeNet pre-trained model with a performance gain of +0.7%. For ScanObjectNN, the PCPN-1k pre-trained model also outperformed the ShapeNet pre-trained model with a gain of +0.7% for the OBJ-BG subset and an increase of +0.3% for the OBJ-ONLY subset.

The stable and consistent improvements, independent of the data domain, prove the effectiveness of the proposed PCPN construction and pre-training using PointMAE. More importantly, the PCPN pre-training framework consumes much less human effort for 3D dataset construction and computational resources for pre-training than ShapeNet pre-trained PointMAE, further validating the efficiency and effectiveness of the proposed method.

In contrast, for the PB-T50-RS subset of the ScanObjectNN, the proposed method falls below the baseline. PB-T50-RS is a relatively challenging subset of ScanObjectNN that is subject to noise, occlusions, rotational extensions, etc. We suspect that the PCPN-1k pre-trained model

**TABLE 2. Comparison of PCPN and ShapeNet on few-shot learning. We evaluated the Accuracy(%) of ModelNet40 under four conditions of few-shot learning. In each experiment, we conducted experiments ten times and calculated the mean and standard deviation. We also cite and compare the score of conventional methods (DGCNN-rand, and DGCNN-OcCo).**

| Methods | 5-way, 10-shot | 5-way, 20-shot | 10-way, 10-shot | 10-way, 20-shot |
|---|---|---|---|---|
| DGCNN-rand [30] | 31.6 ± 2.8 | 40.8 ± 4.6 | 19.9 ± 2.1 | 16.9 ± 1.5 |
| DGCNN-OcCo [30] | 90.6 ± 2.8 | 92.5 ± 1.9 | 82.9 ± 1.3 | 86.5 ± 2.2 |
| Transformer-OcCo [30] | 94.0 ± 3.6 | 95.9 ± 2.3 | 89.4 ± 5.1 | 92.4 ± 4.6 |
| PointBERT [6] | 94.6 ± 3.1 | 96.3 ± 2.7 | 91.0 ± 5.4 | 92.7 ± 5.1 |
| From scratch | 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-1k | 95.0 ± 3.0 | 95.3 ± 5.6 | 89.6 ± 4.1 | 92.6 ± 2.8 |

performs poorly due to the fact that it is pre-trained based on mathematical formulas that generate 3D data with no missing data. Therefore, we speculated the performance could be improved by generating point clouds with intentional misses or occlusions in the PCPN generation algorithm process.

### B. FEW-SHOT LEARNING

To investigate the effects of the proposed PCPN pre-trained model on few-shot learning for 3D shape classification, experiments were conducted to compare the effectiveness of PCPN-1k pre-training frameworks on various 5, 10-way and 10, 20-shots setting on ModelNet40. A pre-training model that achieves good performance with a limited sample of training data is ideal for 3D vision tasks because the collection and manual labeling of 3D data is expensive. Therefore, it is important to validate the effect of pre-training on learning using a few shots.

As shown in Table 2, we validated the effectiveness of the proposed PCPN-1k pre-trained model on ten ModelNet40 subsets of each few-shot learning experiment using the Point-MAE experimental setting. Table 2 shows that the PCPN-1k pre-trained model becomes much more efficient than learning from scratch, with significant performance gains. In particular, it was compared with learning from scratch in four subsets, and it confirmed performance improvements of 7.2%, 2.0%, 5.0%, and 3.2%. In particular, a significant improvement in performance was observed in the 10-shot case, where the amount of data was small. 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 using any real data or manual annotation.

However, an approximately 2.0% lower performance was observed in all subsets compared to the ShapeNet pre-trained model. The domain similarity between the evaluation dataset of ModelNet40 and the pre-training dataset of the baseline of ShapeNet is considered responsible for this result. Both ModelNet40 and ShapeNet are 3D datasets of CAD models. The categories of the two 3D datasets are very similar. Therefore, the shape classification accuracy was believed to be higher with fewer samples than with the PCPN.

In the context of image recognition with synthetic pre-training, the difference in image domains between natural and synthetic images is a major challenge for pre-training with limited data. However, although our results fall short of pre-training with ShapeNet, the observation that the effects of PCPN-1k pre-training can be confirmed even with a small number of samples on 3D datasets with different domains is of great value in academia and the industry.

### C. PART SEGMENTATION

In this experiment, we evaluated the pre-trained model using ShapeNetPart, which contains 16,881 objects from 16 categories. As shown in Table 3, the PCPN-1k pre-trained model achieved a mIoUI of 85.8%, outperforming from-scratch training by 0.7% mIoU. Furthermore, the performance was equivalent to that of the ShapeNet pre-trained model. Through part segmentation, the PCPN pre-trained model demonstrates an understanding of 3D objects through the accurate identification and segmentation of individual parts. The ability to segment 3D objects into their constituent parts allows for a more detailed understanding of their 3D geometry. This demonstrates the effectiveness of the PCPN pre-trained model in tasks requiring nuanced 3D object recognition.

## VI. ABLATION EXPERIMENTS

In this section, we describe exploratory experiments conducted based on the properties of PCPNs. Specifically, we investigated the masking ratio for PCPNs during pre-training and PCPN dataset size.

### A. EFFECT OF MASK RATIO OF PointMAE

In this experiment, we investigated the optimal mask ratio for PCPN pre-training using PointMAE. The effective mask ratio for fine-tuning performance can vary depending on the pre-training characteristics. PointMAE used a mask ratio of 60% on ShapeNet. Therefore, we aimed to determine the optimal mask ratio for PCPN in terms of fine-tuning performance.

As shown in Table 4, the mask ratio of 60% outperforms the ratios of 20%, 40%, and 80%. This result may be due to the fact that the pre-training task becomes too easy when

**TABLE 3. Part segmentation results on ShapeNetPart dataset. We report the mIoU across all part categories (mIoUC %), the mIoU across all instances (mIoUI %), and the IoU (%) for each category. We also cite and compare the scores of conventional methods (PointNet, PointNet++, and DGCNN).**

| Methods | mIoUC | mIoUI | aero<br>lamp | bag<br>laptop | cap<br>motor | car<br>mug | chair<br>pistol | earphone<br>rocket | guitar<br>skateboard | knife<br>table |
|---|---|---|---|---|---|---|---|---|---|---|
| PointNet [25] | - | 83.7 | 83.4<br>80.8 | 78.7<br>95.3 | 82.5<br>65.2 | 74.9<br>93.0 | 89.6<br>81.2 | 73.0<br>57.9 | 91.58<br>72.8 | 85.9<br>80.6 |
| PointNet++ [48] | - | 85.1 | 82.4<br>83.7 | 79.0<br>95.3 | 87.7<br>71.6 | 77.3<br>94.1 | 90.8<br>81.3 | 71.8<br>58.7 | 91.0<br>76.4 | 85.9<br>**82.6** |
| DGCNN [30] | - | 85.2 | 84.0<br>82.8 | 83.4<br>95.7 | 86.7<br>66.3 | 77.8<br>94.9 | 90.6<br>81.1 | 74.7<br>63.5 | 91.2<br>74.5 | 87.5<br>**82.6** |
| From scratch | 83.4 | 85.1 | 82.9<br>85.3 | **85.4**<br>95.6 | 87.7<br>73.9 | 78.8<br>94.9 | 90.5<br>83.5 | **80.8**<br>61.2 | 91.1<br>74.9 | 87.7<br>80.6 |
| ShapeNet | **84.4** | **86.1** | **85.2**<br>**86.1** | 83.8<br>**96.0** | **89.2**<br>**77.2** | **80.7**<br>**95.0** | **91.6**<br>**85.4** | 75.7<br>**65.5** | **92.0**<br>**77.4** | **88.0**<br>81.8 |
| PCPN-1k | 83.6 | 85.8 | 84.1<br>85.7 | 83.2<br>95.9 | 87.7<br>73.8 | 79.9<br>94.6 | 91.2<br>83.8 | 77.8<br>61.6 | 91.4<br>76.6 | 87.0<br>82.0 |

**TABLE 4. The effect of mask ratio on pre-training. We compare the performance on ModelNet40 and ScanObjectNN when the mask ratio on pre-training changed to {20%, 40%, 60%, 80%}.**

| Mask ratio | ModelNet40 | ScanObjectNN | | |
|---|---|---|---|---|
| | | OBJ-BG | OBJ-ONLY | PB-T50-RS |
| 20 % | 92.5 | 84.2 | 86.6 | 81.6 |
| 40 % | 92.8 | **86.2** | 84.7 | 81.0 |
| 60 % | **92.8** | 84.2 | **87.2** | **82.5** |
| 80 % | 92.7 | 85.2 | 84.2 | 80.5 |

**TABLE 5. The effect of dataset size on PCPN pre-training. We compare the pre-training performance on ModelNet40 and ScanObjectNN when the dataset size of PCPN is changed to {1k, 10k, 100k}.**

| Pre-training | ModelNet40 | ScanObjectNN | | |
|---|---|---|---|---|
| | | OBJ-BG | OBJ-ONLY | PB-T50-RS |
| 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 |

the mask ratio is low and too difficult when the mask ratio is high.

### B. SCALING PCPN

An important property of transformers is the scaling law, which suggests that as the dataset size increases, the performance of the transformer also improves. One notable property of the proposed PCPN is its extremely low cost for scaling the dataset size. The experimental results of varying PCPN dataset sizes are shown in Table 5, where PCPN-1k contains 1,000 samples, PCPN-10k contains 10,000 samples, and PCPN-100k contains 100,000 samples.

As shown in Table 5, PCPN-1k outperforms PCPN-10k and PCPN-100k. This result contradicts the scaling law typically observed in transformers. We speculate that this is due to the limited variety of PCPN data. Our proposed PCPN is generated using only three shape variation parameters. Consequently, PCPN-10k and PCPN-100k produce similar data, which may not increase the diversity of the dataset. Furthermore, we believe that pre-training on a large amount of similar data may have led to overfitting in PCPN. This is an important finding for future pre-training with synthetic 3D data, including PCPN. Based on these findings, we recognize the need to improve the dataset construction method moving forward.

## VII. ANALYSIS EXPERIMENTS

This section describes our evaluation of the feature representations obtained through training from scratch, training with the ShapeNet pre-trained model, and training with the PCPN pre-trained model. We visualized the feature space using T-distributed stochastic neighbor embedding (t-SNE) [51] and evaluated the obtained feature representations using centered kernel alignment (CKA) similarity [20].

### A. T-SNEs

The purpose of this experiment was to analyze how the PCPN pre-trained model differs from the ShapeNet pre-trained model in the feature space. Additionally, we aimed to verify the effectiveness of the PCPN pre-trained model by comparing it with models trained from scratch without pre-training.

In Figure 6, we visualize the feature spaces of the fine-tuned models on ModelNet40 using t-SNE. The feature space of the PCPN pre-trained model exhibited more compact clusters compared to the model trained from scratch. However, when compared with the ShapeNet pre-trained model, the ShapeNet model's feature space shows more distinct and compact clusters overall.

### B. CKA-SIMIRALITY

In this experiment, we evaluated the similarity of 3D feature representations among each pre-trained model using CKA similarity [20]. CKA similarity is a non-parametric method for measuring the correlation between two feature

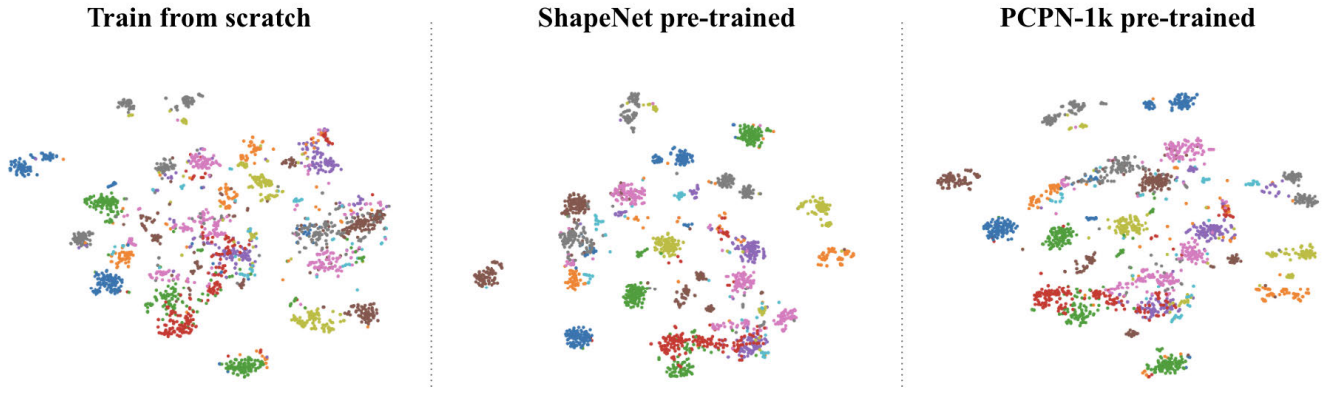


**FIGURE 6. Visualization of feature space on ModelNet. We utilized T-SNE [51] to visualize the distribution of feature space when Scratch, ShapeNet pre-trained model, and PCPN-1k pre-trained model are applied as pre-training parameters on ModelNet40. This figure reveals that the feature space is organized differently depending on the pre-trained model.**

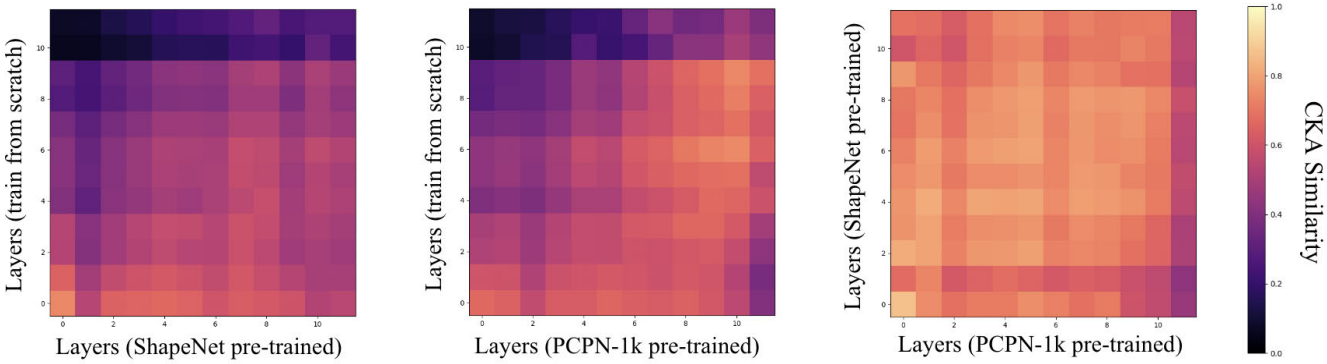


**FIGURE 7. Visualization of CKA similarity between different pre-trained models. This figure shows the comparisons of feature representations of three models, Scratch, ShapeNet pre-trained, and PCPN-1k pre-trained, on ModelNet40 using CKA similarity. The outputs of the 12 intermediate layers of the transformer model used are compared. The axes of each figure indicate the number of the compared layers.**

representations, providing robust results even for high-dimensional data. Specifically, we compared the 12 attention blocks of the PointCloud Transformer encoder across the models trained from scratch, ShapeNet pre-trained model, and PCPN pre-trained model. Fine-tuning was conducted on ModelNet40 using each model as the initial parameter, and a validation dataset was used to evaluate the similarity of the feature spaces.

Several important observations are presented in Figure. 7 Firstly, models trained from random initialization exhibit distinctly different feature representations compared to those that are pre-trained. This suggests that pre-training equips models with specific knowledge that guides them along a different learning path from models trained with random initialization. More intriguingly, a high similarity in feature representations is observed between the ShapeNet pre-trained model and our proposed PCPN pre-trained model. This result implies that the PCPN model can learn effective feature representations for 3D object recognition despite not requiring CAD models or real-world data. These results strongly suggest that the PCPN model is capable of acquiring useful knowledge through pre-training, forming powerful representations for 3D object recognition. This further substantiates the effectiveness of our proposed PCPN pre-trained model and introduces new possibilities for future research in 3D object recognition.

## VIII. DISCUSSION AND LIMITATION

Our experimental results demonstrate that the proposed pre-training framework based on Perlin noise is useful and effective for synthetic pre-training in 3D object recognition. These results suggest that the method has the potential to support the development of applications requiring 3D object recognition with limited amounts of supervised 3D data, such as autonomous driving and robotic navigation.

However, the proposed method has certain limitations. First, we primarily verified the effect of PCPN pre-training on ModelNet40, ScanObjectNN, and ShapeNetParts, which limits the generalizability of our experimental results. Nevertheless, many studies in self-supervised learning [52], [53], [54] still rely on these datasets for evaluation due to the

limited availability of diverse and high-quality 3D datasets. By using these datasets in our experiments, we ensure that our proposed method can be compared directly and fairly with existing self-supervised learning methods. Additionally, a drawback of our method is that the data domain of the 3D shapes generated by our method differs from that of real-world or CAD data. This domain discrepancy could result in lower data efficiency during pre-training compared to competing methods.

Furthermore, Table 5 shows that scaling the size of the PCPN did not result in linear performance improvements. We speculate that one of the limitations of the PCPN generation process lies in the limited variation of 3D shape parameters. Therefore, it is important to increase the number of generation parameters to enhance the diversity of 3D shapes in the PCPN. In this paper, we focused on our pre-training strategy using PointMAE, the latest method in transformer-based self-supervised learning. However, the performance of PCPN with other self-supervised learning methods remains unknown. Thus, it is crucial to explore better representation learning methods for PCPN.

## IX. CONCLUSION

This paper presents a pre-training method for automatically constructing a point cloud dataset, named Point Cloud Perlin Noise (PCPN), without the need for real 3D data collection or human annotation. Large-scale point cloud datasets have long been a bottleneck in pre-training for 3D object recognition due to the high costs associated with data collection and labeling. Our proposed method addresses this challenge by generating a point cloud dataset using a mathematical formula based on Perlin noise, thereby eliminating the need for human intervention. Specifically, our method generates diverse 3D patterns by varying key parameters within the Perlin noise generation process. By training on the generated point cloud through a reconstruction task, we develop a robust pre-training model.

The experimental results demonstrate that the PCPN pre-trained model achieve performance comparable to those pre-trained models on conventional datasets. Moreover, our approach not only alleviates the shortage of 3D data but also enhances the diversity of 3D patterns within the dataset. These findings confirm the effectiveness of the PCPN pre-training strategy for 3D object recognition.

While this study introduces PCPN as an initial exploration into automatic dataset generation for pre-training, we anticipate that further refinement and the development of more advanced learning techniques tailored to PCPN will lead to even greater improvements in pre-trained model performance. Additionally, expanding the scope of testing to include more diverse 3D datasets represents a valuable direction for future research. We plan to explore this in future studies as more comprehensive 3D datasets become available.

## REFERENCES


[1] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "PointContrast: Unsupervised pre-training for 3D point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 574–591.

[2] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[3] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9726–9735.

[4] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15745–15753.

[5] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9630–9640.

[6] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-BERT: Pre-training 3D point cloud transformers with masked point modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 19291–19300.

[7] Y. Pang, W. Wang, F. E. H. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 604–621.

[8] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9762–9772.

[9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, and H. Su, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[11] Y. M. Asano, C. Rupprecht, A. Zisserman, and A. Vedaldi, "PASS: An ImageNet replacement for self-supervised pretraining without humans," in *Proc. NeurIPS Track Datasets Benchmarks*, 2021.

[12] M. Baradad, J. Wulff, T. Wang, P. Isola, and A. Torralba, "Learning to see by looking at noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 2556–2569.

[13] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4396–4405.

[14] H. Kataoka, K. Hara, R. Hayashi, E. Yamagata, and N. Inoue, "Spatiotemporal initialization for 3D CNNs with generated motion patterns," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 737–746.

[15] K. Perlin, "Making noise," *GDC Talk*, 1999.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[17] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[18] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597.

[19] 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.*, pp. 1–12, Nov. 2016.

[20] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3519–3529.

[21] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2432–2443.
[22] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 567–576.
[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.
[24] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2443–2451.
[25] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.
[26] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on $\chi$-transformed points," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 828–838.
[27] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.
[28] Y. Zhang and M. Rabbat, "A graph-CNN for 3D point cloud classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 6279–6283.
[29] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9613–9622.
[30] 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 Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.
[31] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10288–10297.
[32] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6410–6419.
[33] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16239–16248.
[34] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[35] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852.
[36] H. Liu, M. Cai, and Y. J. Lee, "Masked discrimination for self-supervised learning on point clouds," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, Oct. 2022, pp. 657–675.
[37] Y. Rao, B. Liu, Y. Wei, J. Lu, C.-J. Hsieh, and J. Zhou, "RandomRooms: Unsupervised pre-training from synthetic shapes and randomized layouts for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3263–3272.
[38] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra, "Self-supervised pretraining of 3D features on any point-cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10232–10243.
[39] J. Hou, B. Graham, M. Nießner, and S. Xie, "Exploring data-efficient 3D scene understanding with contrastive scene contexts," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15582–15592.
[40] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[41] B. Eckart, W. Yuan, C. Liu, and J. Kautz, "Self-supervised learning on 3D point clouds by learning discrete generative models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8244–8253.
[42] R. Yamada, H. Kataoka, N. Chiba, Y. Domae, and T. Ogata, "Point cloud pre-training with natural 3D structures," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 21251–21261.
[43] K. Perlin, "Advanced image synthesis," in *Proc. ACM SIGGRAPH Conf.*, 1984.
[44] K. Perlin and E. M. Hoffert, "Hypertexture," *ACM SIGGRAPH Comput. Graph.*, vol. 23, no. 3, pp. 253–262, Jul. 1989.
[45] K. Perlin, "Improving noise," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 681–682, Jul. 2002, doi: 10.1145/566654.566636.
[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[47] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.
[48] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.
[49] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Trans. Multimedia*, vol. 24, pp. 1943–1955, 2022.
[50] S. Cheng, X. Chen, X. He, Z. Liu, and X. Bai, "PRA-Net: Point relation-aware network for 3D point cloud analysis," *IEEE Trans. Image Process.*, vol. 30, pp. 4436–4448, 2021.
[51] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html
[52] G. Chen, M. Wang, Y. Yang, K. Yu, L. Yuan, and Y. Yue, "PointGPT: Auto-regressively generative pre-training from point clouds," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.
[53] Y. Zha, H. Ji, J. Li, R. Li, T. Dai, B. Chen, Z. Wang, and S.-T. Xia, "Towards compact 3D representations via point feature enhancement masked autoencoders," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 7, pp. 6962–6970.
[54] S. Yan, Y. Yang, Y. Guo, H. Pan, P.-S. Wang, X. Tong, Y. Liu, and Q. Huang, "3D feature prediction for Masked-AutoEncoder-Based point cloud pretraining," 2023, *arXiv:2304.06911*.

**RYOSUKE YAMADA** received the M.S. degree from Tokyo Denki University (TDU), in 2022. He is currently pursuing the Ph.D. degree with the University of Tsukuba. He is a Research Assistant with the Computer Vision Research Team, Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan. His research interests include computer vision, 3D object recognition, and synthetic pre-training.

**RYU TADOKORO** is currently pursuing the degree with the School of Medicine, Tohoku University. He is interning with the Computer Vision Research Team, Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan. His research interests include computer vision, with an emphasis on medical image analysis, 3D vision, and pre-training.

**YUE QIU** received the Ph.D. degree from the University of Tsukuba, in 2021. She is currently a Researcher with the Computer Vision Research Team, Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan. Her research interests include computer vision, 3D vision, vision-and-language models, and natural language processing. She serves as a Reviewer for top-tier conferences in computer vision and artificial intelligence, including the Computer Vision and Pattern Recognition (CVPR), the International Conference on Computer Vision (ICCV), the European Conference on Computer Vision (ECCV), and the Conference on Neural Information Processing Systems (NeurIPS).

**HIROKATSU KATAOKA** received the Ph.D. degree in engineering from Keio University, in 2014. He is currently a Chief Senior with the Computer Vision Research Team, Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan. His research experience includes as a Visiting Scientist with the Technical University of Munich (TUM) and a JSPS Fellow (PD) with The University of Tokyo. He also leads the cvpaper.challenge, which conducts comprehensive surveys and collaborative research in computer vision and related academic disciplines. He has contributed to the development of a strong baseline for video recognition using 3D ResNets and established a synthetic pre-training strategy in formula-driven supervised learning (FDSL). He has received several awards, including the ACCV 2020 Best Paper Honorable Mention Award, the AIST 2019 Best Paper Award, and the ECCV 2016 Workshop Brave New Idea Award, and has been featured in *MIT Technology Review*.

**YUTAKA SATOH** received the Ph.D. degree in engineering from Hokkaido University, Japan, in 2001. He is currently a Principal Researcher with the Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan. He is also a Professor with the Cooperative Graduate School Program, University of Tsukuba, Japan. His research interests include machine vision systems, mobile robots, and robust pattern recognition algorithms.