Self-Supervised Point Cloud Representation Learning with Occlusion Auto-Encoder

Junsheng Zhou*, Xin Wen*, Baorui Ma, Yu-Shen Liu, Yue Gao, Yi Fang, Zhizhong Han

Abstract—Learning representations for point clouds is an important task in 3D computer vision, especially without manually annotated supervision. Previous methods usually take the common aid from auto-encoders to establish the self-supervision by reconstructing the input itself. However, most of the existing self-reconstruction based auto-encoders merely focus on the global shapes, and ignore the hierarchical context between the local and global geometries, which is a crucial supervision for 3D representation learning. To resolve this issue, we present a novel self-supervised point cloud representation learning framework, named 3D Occlusion Auto-Encoder (3D-OAE). Our key idea is to randomly occlude some local patches of the input point cloud and establish the supervision via recovering the occluded patches using the remaining visible ones. Specifically, we design an encoder for learning the features of visible local patches, and a decoder for leveraging these features to predict the occluded patches. In contrast to previous methods, our 3D-OAE can remove a large proportion of patches and predict them only with a small number of visible patches, which enable us to significantly accelerate training and yield a nontrivial self-supervisory performance. The trained encoder can be further transferred to various downstream tasks. We demonstrate our superior performances over the state-of-the-art methods in different discriminant and generative applications under widely used benchmarks.

Index Terms—3D representation learning, self-supervised learning, point cloud completion, transformer.

I. INTRODUCTION

POINT clouds play a crucial role in 3D computer vision applications [1]–[3] due to its flexibility to represent arbitrary geometries and memory-efficiency. In this paper, we specifically focus on the task of learning representations of point clouds without manually annotated supervision. As 2D images, learning representations for 3D point clouds has been comprehensively studied for many years, and the research line along the 2D and 3D representation learning shares a lot of common practices, such as the auto-encoder based framework and the self-reconstruction based supervision. The recent development in both NLP and 2D computer vision fields has also driven several improvements in 3D representation learning, such as PCT [4], Point-BERT [5] and STRL [6]. However, the different data characteristics between the 2D and 3D domains limit the direct applications of many 2D improvements into 3D scenarios, e.g. the differences between ordered 2D grids and unordered 3D points.

One of major challenges is to learn the hierarchical context between the global structure and local geometries. In 3D scenarios, this is more difficult than the learning of 2D images due to the discrete nature of 3D points. In most previous methods, the 3D auto-encoder usually relies on the self-reconstruction as the supervision to focus on the global structures and the local parts. However, the simple self-reconstruction based framework usually does not explicitly distinguish the local parts and the global structures apart. As a result, both of them are only revealed by the shape matching constraints (e.g. Chamfer Distance) as a whole, while more detailed self-supervision to reveal the local to global hierarchy in 3D point clouds is merely discussed.

The recent improvements of mask-based 2D auto-encoders [7] have proved that masked auto-encoders are effective in image representation learning through the inference of the overall image information based on the visible local patches. It provides a new perspective to establish the self-supervision between the local and global information. However, due to the discrete nature of point clouds, it’s difficult to directly use a 2D mask-based auto-encoder to learn 3D representations. Driven by the above analysis, we present 3D-OAE, a novel Transformer-based self-supervised learning framework with Occlusion Auto-Encoder. As shown in Fig. 1, we separate the unlabelled point cloud into local point patches and centralize them to their corresponding seed point. After that, we occlude a large proportion of the patches but remain the seed points, and learn to recover occluded patches from seed points and the visible patches. The seed points serve as global hints to guide the shape generation and the model will be forced to focus on learning the local geometries details. Specifically, we design an encoder to learn features only on the visible subset of patches, and a decoder to leverage the features of visible patches to predict the local features of the occluded ones, and finally reconstruct the occluded patches with seed points as the global hints. After self-supervised learning without any manual annotation, we can transfer the trained encoder to different downstream tasks. We demonstrate our superior performances by comparing our method under widely used benchmarks.

Our main contributions can be summarized as follows:

- We proposed a novel self-supervised learning framework
named 3D Occlusion Auto-Encoder. Unlike previous 3D auto-encoders, 3D-OAE designs an asymmetrical encoder-decoder Transformer architecture to learn the patterns from the visible local patches and leverage them to control the local geometry generation of the occluded patches. After self-supervised learning, the trained encoder can be transferred to new downstream tasks.

- Our 3D-OAE can remove a large proportion (e.g. 75%) of point cloud patches before training and only encodes a small number of visible patches. This enable us to accelerate training for 3-4 times and makes it possible to do self-supervised learning in large scale unlabelled data efficiently.
- We achieved the state-of-the-art performances in six different downstream applications compared with previous self-supervised methods.

II. RELATED WORK

1) Representation Learning on Point Clouds: Different from structured data like images, point clouds are unordered sets of vectors, which brings great challenges to the learning of representations. The deep learning based 3D point cloud processing techniques has achieved very promising results in different tasks [8]–[15]. Qi et al. [16] pioneered point cloud learning by proposing PointNet, which directly input the raw point cloud into point-wised MLPs and use max-pooling to solve the permutation invariant of point cloud. Further more, PointNet++ [17] applies query ball grouping and hierarchical spatial structure to increase the sensitivity to local geometries. Some works followed this idea and developed different grouping strategies [18]–[20]. A number of approaches build graphs to connect points and aggregate information through graph edges [21]–[25]. DGСN [21] propose to use graph convolutions on KNN graph nodes, and GACNet [25] apply attention mechanism in graph convolutions. Some other methods propose to use continuous convolutions on point clouds [18], [26]–[31]. PointCNN [30] applies convolution neural network on point set after reordering points with special operators. Different from the idea of using convolution-based structure, Point2Sequence [32] propose to use a recurrent model (i.e. RNN) to capture the fine-grained sequence information of features in local patches. Point2SpatialCapsule [33] introduces a capsule network for modeling the spatial relationships of local regions by aggregating the local features into a set of learnable cluster centers.

Some recent studies focus on improving the extraction of local features. RS-CNN [34] proposes a shape-aware convolution network to learn local features by exploring the relationship of local points. ShellNet [28] designs a permutation invariant convolution operation for learning and aggregating the geometric features of local regions with different scales. GS-Net [35] proposes to use the Eigen-Graph to learn the geometric relationships between local point cloud regions. A more recent work Point-MLP [36] introduces a MLP-based network with a learnable geometric affine operation to adaptively transform the point features in local regions.

Inspired by the great success achieved by Transformers in both NLP [37]–[39] and 2D vision [40]–[43], some recently works try to apply Transformers in 3D point cloud representation learning [4], [44], [45]. Zhao et al. [44] propose to use vectorized self-attention mechanism in Transformer-layer, and apply a hierarchical structure with local feature aggregation. Guo et al. [4] propose to use neighbour embedding to enhance the representation learning ability. However, previous Transformer-based methods on point cloud represen-
tation learning bring in inevitable inductive biases and manual assumptions, the standard Transformer with no inductive bias is proved to perform poorly [5] due to the limited scale of point cloud data. In this work, we aim to extend the success of standard Transformer to 3D point cloud representation learning.

2) Self-supervised Learning on Point Clouds: Self-supervised learning (SSL) is to learn the representation from unlabelled data, where the supervision signals are built from the data itself. Since the annotation of point clouds is often time-consuming and error-prone, the performance of supervised approaches is difficult to be further improved. Therefore, SSL on point clouds becomes more and more important. Recently, several works propose to use SSL techniques for point cloud representation learning [46]–[55]. Sauder et al. [50] propose to rearrange shape parts and reconstruct the original shapes. PointContrast [47] design a SSL scheme by contrastive learning on different views of point clouds. DepthContrast [55] propose to conduct contrastive learning information from the depth scans. CrossPoint [56] introduces a cross-modality contrastive learning strategy by exploring self-supervised signals from the semantic differences between point clouds and their rendering images. Inspired by BERT, Point-BERT [5] achieves great performance by pre-training a standard Transformer in a BERT-style SSL scheme. However, Point-BERT only focuses on distinguishing tokens of different patches, makes it difficult for them to transfer to downstream generative tasks.

Auto-encoder. An auto-encoder architecture consists of two parts: an encoder and a decoder. A number of approaches [46], [48], [54], [57], [58] apply auto-encoder architecture to learn meaningful representations from unlabelled point clouds. The proposal of point cloud auto-encoder is to learn the presentation from the input shape and then reconstruct the shape from the learned low-dimension latent code. FoldingNet [46] designs a point cloud auto-encoding with a folding-based decoder. IAE [59] proposes to use an implicit auto-encoder to learn more detailed geometric information from the additional supervision of the occupancy values and the signed distance values, which prevents IAE from self-supervised learning directly using the point clouds. OcCo [48] proposes to complete view-occluded point cloud with a standard point cloud completion network. However, these methods only focus on the generation ability of the whole shape, thus mixing the local and global geometry features together, making it hard to transfer the knowledge to downstream tasks. Recently, in 2D vision, He et al. [7] propose a new form of auto-encoders named MAE by masking regular patches of images and learning to recover the masked parts. Partly inspired by MAE, we design a new self-supervised learning framework to recover the complete shapes from the highly occluded shapes.

III. OCCLUSION AUTO-ENCODER

The overall architecture of 3D-OAE is shown in Fig. 3. Like other point cloud auto-encoders, 3D-OAE consists of an encoder which learns the representation from the input shape and a decoder to reconstruct the original shape from the learned representation. Unlike other point cloud auto-encoders which operates on the whole shape, 3D-OAE divides the complete shape into groups of patches, highly occludes them, and learns to recover the missing patches of shapes. To achieve this, an asymmetrical encoder-decoder architecture is designed with an encoder only operates on the visible subset of patches, and a decoder to predict local features of occluded patches from the visible ones. After that, we combine the predicted local features of occluded patches and their corresponding seed points which serve as global hints to infer the missing geometries that semantically match the input 3D shape. After self-supervised learning, we can leverage the encoder in different downstream tasks as illustrated in Fig. 1. Specifically, we first operate average pooling to aggregate all local features extracted from the trained encoder into a global feature for representing the whole shape, and then fed it into the special decoders of different downstream tasks.

A. Grouping and Occluding

Previous Transformer-based methods treat each single point in the original shape as a minimum operation unit like words in sentences. However, it brings huge computational complexity and large demand for memory due to the large scale of point
cloud data (we don’t expect a sentence to have thousands of words). Inspired by previous works [5], [42], we choose to use patches of point clouds as the minimum unit. To achieve this, we first use Furthest Point Sampling (FPS) to sample seed points. Inspired by previous works [5], [42], we choose to use cloud data (we don’t expect a sentence to have thousands of

where we compute a loss function with the ground truth. In addition to the output patches and their corresponding seed points to regain their spatial locations and further merge the local patches into a complete shape, we compute a loss function with the ground truth.

We apply a straightforward occluding strategy: we randomly select a subset of seed points \( \{ s_i \}_{i=1}^R \), and then remove their corresponding patches \( \{ g_i \}_{i=1}^G \). After that, we project each of the remain visible patches \( \{ g_i \}_{i=1}^{G-R} \) into a patch embedding as shown in Fig. 3 using a simple PointNet as:

\[
E_i = \text{Max}(x_i) \in \mathbb{R}^{1 \times C}, \quad \text{where} \quad x_i = \phi(g_i|\theta) \in \mathbb{R}^{K \times C},
\]

where \( \phi \) and \( \theta \) are the MLP layers and the weights, \( C \) is the channel of patch embeddings and \( \text{Max} \) denotes Max-Pooling operation. The patch embeddings \( \{ E_i \}_{i=1}^{G-R} \) will serve as the inputs to the encoder \( f \).

We choose to occlude a very large regions (75%) of the original shape, more numerical comparison of occlusion ratios can be found in Table IX. Removing a high ratio of patches largely increases the difficulty of auto-encoding reconstruction, thus force model to learn a powerful representation to generate more detailed local geometries. More importantly, the design of highly occlusion strategy makes it possible for efficient self-supervised learning on large scale unlabelled point cloud data.

B. Transformer

We will simply review the standard transformer block [37] which serves as the unit architecture of our proposed 3D-OAE in this section. A transformer block consists of a self-attention layer with multi-heads and a feed-forward network which is implemented as a single hidden layer. Given a set of patch embeddings \( \{ E_i \}_{i=1}^{G-R} \) as the input, we first map them into queries, keys and values which are formulated as matrices \( Q, K, \) and \( V \) using MLPs. The dot-product self-attention is then computed by:

\[
A(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}}V),
\]

where \( d \) means the channel dimension. A parallel multi-head strategy is further applied for allowing the transformer to attend to the information from different subspace jointly,

\[
M(Q, K, V) = \text{Concat}(a_1, a_2, ..., a_h)W,
\]

\[
\text{where} \quad a_i = A(QW_i^Q, KW_i^K, VW_i^V)
\]

and \( W, W_i^Q, W_i^K, W_i^V \) are projection matrices with learnable parameters.

C. Auto-encoder Architecture

1) 3D-OAE Encoder: We adopt the 3D point cloud standard Transformers with multi-headed self-attention layers and FFN blocks as detailed above as the unified backbone of our architecture. Specifically, our encoder is a standard Transformer but applies only on visible patches. For the input visible patches, we first extract their patch embeddings as described in Eq. (1).
points \( \{ s_i \}_{i=1}^{G-R} \) and add them to their corresponding patch embeddings as:

\[
E_i \leftarrow \gamma(s_i|\theta) + E_i,
\]

After that, a series of Transformer blocks is applied to these patch embeddings to learn representations.

\[
E' = \text{Linear}(f_\omega(E,H_e)),
\]

where \( f_\theta \) indicates the Transformer encoder, \( H_e \) represents the number of Transformer blocks in \( f_\theta \) and \( E = [E_1,E_2,\ldots,E_{G-R}] \) is the set of patch embeddings. A linear projection layer is further applied for dimension mapping.

Since we use a very high occlusion ratio, the encoder operates only on a small subset (e.g. 25%) of patches, which makes it possible to do self-supervised learning in very large scale unlabelled data with a relatively huge encoder.

2) 3D-OAE Decoder: The input of 3D-OAE decoder is a full set of patch embeddings consisting of the encoded visible patch embeddings and the occlusion tokens \( \{ T_i \}_{i=1}^R \), formulated as:

\[
U = \text{Concat}(E', T),
\]

where \( E' \in \mathbb{R}^{(G-R) \times K \times C} \), \( T \in \mathbb{R}^{R \times K \times C} \) and \( U \in \mathbb{R}^{G \times K \times C} \).

Each occlusion token is a shared, learnable vector which aim at learning to reconstruct one occluded patch. We further add the position embeddings \( \{ \gamma(s_i|\theta) \}_{i=1}^G \) to the full set of patch embeddings for providing the location information to the occlusion tokens. Then a series of light-weighted Transformer blocks are further applied to learn the occlusion tokens from features of visible patches via self-attention mechanism:

\[
U' = f_\omega(U + \{ \gamma(s_i|\theta) \}, H_d),
\]

where \( f_\omega \) and \( H_d \) are the Transformer decoder and the number of blocks of it.

Since we calculate the attention map of each patch embedding to all of the others, the model will have no sensitivity about the ordering of patches, which indicates that 3D-OAE is suitable for the unordered point cloud data.

The decoder \( g \) is only used during self-supervised learning to recover the occluded parts of the original shape, only the learned encoder \( f \) is used when transferring to downstream tasks (e.g. object classification, point cloud segmentation, point cloud completion), which means we don’t care much about the learning ability of the decoder. Therefore, we design a light-weighted decoder with only about 20% computation of the encoder. And the training process is largely accelerated since the full set of patch embeddings is only processed by the light-weighted decoder.

D. Optimization Objective

During training, the goal of 3D-OAE is to reconstruct the complete shape from seed points and visible point patches. After encoding and decoding, 3D-OAD outputs patch-wised vectors where each vector contains the local geometry information of a single patch. The feature channel of the Transformer decoder \( f_\omega \) is set to be the product of point dimensions and patch point numbers, thus each vector can be directly reshaped to the size of a local patch. Finally, the seed points are added to their corresponding patches to reconstruct the complete shape. We choose Chamfer Distance described by Eq. (8) as our loss function. Similar to BERT [38] and MAE [7], we don’t pay much attention to the reconstruction ability of visible parts, and only compute loss between the points of predicted patches \( P^o \) and the ground truth point cloud of these occluded patches denoted as \( P^t \).

\[
L_{\text{CD}}(P^o, P^t) = \frac{1}{|P^o|} \sum_{p^o \in P^o} \min_{p^t \in P^t} \| p^o - p^t \|_2^2 + \frac{1}{|P^t|} \sum_{p^t \in P^t} \min_{p^o \in P^o} \| p^t - p^o \|_2^2.
\]

We also try to use Earth Mover’s Distance as the loss function but find it unhelpful, please see the numerical comparison in Table VIII.

IV. EXPERIMENTS AND APPLICATIONS

In this section, we first introduce the self-supervised learning setting of 3D-OAE in Sec. IV-A. Next, we evaluate the proposed model with various downstream tasks in Sec. IV-B to Sec. IV-F, including shape understanding, few-shot learning, part segmentation and transfer learning to generative tasks and real world tasks. We show the efficiency and learning curves in IV-G and IV-H. Finally, We conduct ablation studies on framework designs and occlusion ratios in Sec. IV-I.

A. Self-supervised Learning

Dataset. We learn the self-supervised representation model from the ShapeNet [60] dataset which contains 57,448 synthetic models from 55 categories. We sample 1024 points from each 3D model and divide them into 64 point cloud patches using Furthest Point Sample (FPS) and K-Nearest Neighbor (KNN), where each patch contains 32 points. During training, we apply the same data augmentations as PointNet++ [17].

Training setups. In the self-supervised learning stage, we set the Transformer depth of both encoder and decoder to 12 and the number of Transformer heads both to 6. The feature channel dimension of encoder and decoder Transformers are set to 384 and 96, and the occlusion ratio is set to 75%. We adopt an AdamW [61] optimizer, using an initial learning rate of 0.0005 and a weight decay of 0.05. And we train our model for 300 epochs with a batch size of 256 on one 2080Ti.

Visualization. In Fig. 4, we present more self-reconstruction results of 3D-OAE as a supplement of Fig.2. We provide the visualization of both the reconstructed shapes and the complete input shapes which also indicate the target shapes. We show the complete input shapes for visual comparison but still notice that 3D-OAE only operates on a small subset patches of the input shapes.

B. Shape Understanding

We follow prior works [46], [57], [62] to evaluate the shape understanding capability of our self-supervision model using the ModelNet40 [63] benchmark. It contains 12,311 synthesized models from 40 categories and is split into 9843/2468
which proves that the representation domain learned by 3D-OAE from scratch (e.g. PointNet++ (90.5\%)) has reached the accuracy of train a classification network only achieves 91.2\% of 92.3\%.

Our proposed 3D-OAE achieves state-of-the-art performance. The comparison of classification results is shown in Table I.

Clouds is down-sampled to 2048 for both training and testing. After training our models in ShapeNet [60] as detailed in Sec. IV-A, we evaluate the learned models for training and testing. After training our models in the following experiments.

1) Linear SVM: In this experiment, we train a linear Support Vector Machine (SVM) classifier using the representation from our trained encoder Transformer. The number of point clouds is down-sampled to 2048 for both training and testing. The comparison of classification results is shown in Table I. Our proposed 3D-OAE achieves state-of-the-art performance of 92.3\% accuracy on test sets, while the runner-up method only achieves 91.2\% accuracy. It’s worth noting that this result has reached the accuracy of train a classification network from scratch (e.g. PointNet++ (90.5\%), DGCNN (92.2\%)), which proves that the representation domain learned by 3D-OAE is highly decoupled. Since our model is learned on the ShapeNet dataset, we believe that this result also shows the strong transfer ability of our model.

2) Supervised Fine-tuning: In this experiment, we explore the ability of our model to transfer to downstream classification tasks. The supervised models are trained from scratch and the self-supervised models use the trained weights from self-supervised learning as the initial weights for fine-tuning. All the self-supervised methods use the standard Transformer (STransformer) as backbone architecture. For a fair comparison with OcCo, we follow the details illustrated in [48] and use the standard Transformer encoder and Transformer-based decoder PointTr [45] to reproduce the completion task in ShapeNet. In comparison, our 3D-OAE brings 2.0\% (91.4\%/93.4\%) accuracy improvement over training from scratch. And our method also outperforms PCT [4], which is a variety of standard Transformer. The result proves that using our self-supervised learning scheme, a standard Transformer with no inductive bias could also learn a powerful representation.

3) Embedding Visualizations: We visualize the feature distributions using t-SNE [68]. Fig. 5 (b) shows the features learned by 3D-OAE after self-supervised training on the ShapeNet dataset. It’s clear that the feature space of different categories which are mixed together in random initialization (Fig. 5 (a)) can be well separated into different regions by 3D-OAE. We also achieve comparable performance with PointBERT (Fig. 5 (c)). As shown in Fig. 5 (e), the feature space are almost separated completely independent after fine-funing on ModelNet40 train sets, and are more clearly disentangled than training from scratch (Fig. 5 (d)). It proves that 3D-OAE can guide Transformers to learn powerful representations from unlabelled data, even from different dataset.

### C. Few-shot Learning

We further evaluate our model by conducting few-shot learning experiments on ModelNet40. A common used setting is “K-way N-shot”, where K classes are first random selected, and then (N+20) samples are sampled from each class. The model is trained on K × N samples, and evaluated on K × 20 samples. Following previous work [5], [69], we choose 4 different few-shot learning settings: “5 way, 10 shot”, “5 way, 20 shot”, “5 way, 10 shot” and “10 way, 20 shot”. For fair comparison, we use the data processed by Point-BERT [5].

| Category | Method | Accuracy |
|----------|--------|----------|
| Supervised | PointNet [16] | 89.2\% |
| | PointNet++ [17] | 90.5\% |
| | PointCNN [30] | 92.2\% |
| | DGCNN [21] | 92.2\% |
| | PCT [4] | 93.2\% |
| | STransformer | 91.4\% |
| Self-supervised | STransformer + OcCo [48] | 92.1\% |
| | STransformer + Point-BERT [5] | 93.2\% |
| | 3D-OAE (Ours) | 93.4\% |
Fig. 5. **Visualization of feature distributions.** We visualize the features of test sets in ModelNet40 using t-SNE. (a) random initialization, (b) 3D-OAE pre-trained on ShapeNet, (c) Point-BERT pre-trained on ShapeNet, (d) train an randomly initialized encoder on ModelNet40, (e) fine-tuning learned encoder of 3D-OAE on ModelNet40.

### TABLE III

Few-shot classification results on ModelNet40

| Method               | 5 way |          | 10 way |          |
|----------------------|-------|----------|--------|----------|
|                      | 10-shot | 20-shot | 10-shot | 20-shot |
| DGCNN-rand [21]       | 91.8 ± 3.7 | 93.4 ± 3.2 | 86.3 ± 6.2 | 90.9 ± 5.1 |
| DGCNN-OcCo [48]       | 91.9 ± 3.3 | 93.9 ± 3.1 | 86.4 ± 5.4 | 91.3 ± 4.6 |
| STransformer-rand     | 87.8 ± 5.2 | 93.3 ± 4.3 | 84.6 ± 5.5 | 89.4 ± 6.3 |
| STransformer-OcCo [48] | 94.0 ± 3.6 | 95.9 ± 2.3 | 89.4 ± 5.1 | 92.4 ± 4.6 |
| STransformer-Point-BERT [5] | 94.6 ± 3.1 | 96.3 ± 2.7 | 91.0 ± 5.4 | 92.7 ± 5.1 |
| 3D-OAE               | 96.3 ± 2.5 | 98.2 ± 1.5 | 92.0 ± 5.3 | 94.6 ± 3.6 |

We compare our model with currently state-of-the-art methods OcCo [48] and Point-BERT [5]. As shown in Table III, using standard Transformer as backbone, our proposed 3D-OAE achieves a significant improvement of 8.5%, 4.9%, 7.4%, 5.2% over baseline, and 1.7%, 1.9%, 1.0%, 1.9% over the runner-up method Point-BERT in 4 different sets of few-shot classification. We even achieve 98.2% accuracy on the “5 way 20 shot” with a standard deviation of only 1.5. The outstanding performance on few-shot learning proves the strong ability of 3D-OAE to transfer to downstream tasks using very limited data.

### D. Object Part Segmentation

Object part segmentation is a challenging task which aims to predict the part label for each point of the model. The ShapeNetPart [70] dataset consists of 16,800 models from 16 categories and is split into 14006/2874 for training and testing. The number of parts for each category is between 2 and 6, and there are 50 different parts in total. We sample 2048 points from each model follow PointNet [16], and apply a segmentation head achieved by Point-BERT [5] to propagates the group features to each point hierarchically.

As shown in Table IV, our model achieves 0.6% improvement over training a standard Transformer from scratch, while OcCo fails to improve performance. 3D-OAE also outperforms PointNet, PointNet++ and DGCNN. A visualization of our segmentation results is shown in Fig. 6. It’s clear that our model is able to make the right predictions at most points, and there are only small visual differences between our predictions and the ground truth.

### E. Transfer to Generative Tasks

Since most of the previous self-supervised learning methods only focus on the discriminant ability of the representation learned by their model and verify it by transferring the model to classification tasks. They fail to transfer their model to downstream generative tasks (e.g. point cloud completion, point cloud up-sampling). In this section, we show the transfer learning ability of 3D-OAE to downstream generative tasks by conducting point cloud completion experiments.

**Dataset briefs and evaluation metric.** The PCN [71] dataset is one of the most widely used benchmark datasets in point cloud completion task. It contains 30,974 models from 8 categories. For each 3D object, 16,384 points are sampled from the shape surface, and 8 partial point clouds are generated by back-projecting 2.5D depth images from 8 views into 3D. For a fair comparison, we use the same train/test split settings of PCN [71] and follow previous works to adopt the L1 version of Chamfer distance for evaluation. We use a standard Transformer encoder and a Transformer-based decoder proposed in PoinTr [45] as our backbone, and OcCo is trained using the same architecture. We finetune our model and OcCo under PCN dataset on a single GTX 3090Ti GPU, and it takes 300 epochs to converge.

**Quantitative comparison.** The results of our proposed 3D-OAE and other completion methods are shown on Table V, where 3D-OAE achieves the state-of-the-art performance over all counterparts compared with both supervised and self-supervised methods. We found that 3D-OAE reduces the average CD by 0.4 compared with training a standard Transformer from scratch. Especially, 3D-OAE with only a standard Transformer-based model reduces the average CD by 0.24 compared with the state-of-the-art supervised method SnowflakeNet [78] which proposed a carefully designed model
Fig. 6. Visualization of our part segmentation results. Different colors indicate different parts. Top row shows results predicted by our model; bottom row shows the corresponding ground truth.

TABLE IV

| Methods                  | mIoU | aero | bag | cap | car | chair | car | guitar | knife | lamp | lap | motor | mug | pistol | rock | skate | table |
|--------------------------|------|------|-----|-----|-----|-------|-----|--------|-------|------|-----|-------|-----|--------|------|-------|-------|
| PointNet [16]            | 83.7 | 83.4 | 78.7 | 82.5 | 74.9 | 89.6  | 73.0 | 91.5   | 85.9  | 80.8 | 95.3 | 65.2  | 93    | 81.2   | 57.9 | 72.8  | 80.6  |
| PointNet++ [17]          | 85.1 | 82.4 | 79   | 87.7 | 77.3 | 90.8  | 71.8 | 91     | 85.9  | 83.7 | 95.3 | 71.6  | 94.1  | 81.3   | 58.7 | 76.4  | 82.6  |
| DGCNN [21]               | 85.2 | 84   | 83.4 | 86.7 | 77.8 | 90.6  | 74.7 | 91.2   | 87.5  | 82.8 | 95.7 | 66.3  | 94.9  | 81.1   | 63.5 | 74.5  | 82.6  |
| STTransformer            | 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  |
| STTransformer-OcCo [48]  | 85.1 | 83.3 | 85.2 | 88.3 | 79.9 | 90.7  | 74.1 | 91.9   | 87.6  | 84.7 | 95.4 | 75.5  | 94.4  | 84.1   | 63.1 | 75.7  | 80.8  |
| STTransformer-Point-Bert [5] | 85.6 | 84.3 | 84.8 | 88.0 | 79.8 | 91.0  | 81.7 | 91.6   | 87.9  | 85.2 | 95.6 | 75.6  | 94.7  | 84.3   | 63.4 | 76.3  | 81.5  |
| 3D-OAE                   | 85.7 | 83.4 | 85.0 | 83.8 | 79.3 | 80.1  | 80.1 | 91.9   | 87.2  | 82.5 | 95.3 | 76.0  | 95.1  | 85.6   | 63.5 | 80.5  | 83.6  |

TABLE V

| Methods                    | Average | Plane | Cabinet | Car | Chair | Lamp | Couch | Table | Boat |
|-----------------------------|---------|-------|---------|-----|-------|------|-------|-------|------|
| FoldingNet [46]             | 14.31   | 9.49  | 15.80   | 12.61| 15.55 | 16.41| 15.97 | 13.65 | 14.99 |
| TopNet [72]                 | 12.15   | 7.61  | 13.31   | 10.90| 13.82 | 14.44| 14.78 | 11.22 | 11.12 |
| AtlasNet [73]               | 10.85   | 6.37  | 11.94   | 10.10| 12.06 | 12.37| 12.99 | 10.33 | 10.61 |
| PCN [71]                    | 9.64    | 5.50  | 22.70   | 10.63| 8.70  | 11.00| 11.34 | 11.68 | 8.59  |
| GRNet [74]                  | 8.83    | 6.45  | 10.37   | 9.45 | 9.41  | 7.96 | 10.51 | 8.44  | 8.04  |
| CDN [75]                    | 8.51    | 4.79  | 9.97    | 8.31 | 9.49  | 8.94 | 10.69 | 7.81  | 8.05  |
| PMP-Net [76]                | 8.73    | 5.65  | 11.24   | 9.64 | 9.51  | 6.95 | 10.83 | 8.72  | 7.25  |
| NSFA [77]                   | 8.06    | 4.76  | 10.18   | 8.63 | 8.53  | 7.03 | 10.53 | 7.35  | 7.48  |
| PoinTr [45]                 | 8.38    | 4.75  | 10.47   | 8.68 | 9.39  | 7.75 | 10.93 | 7.78  | 7.29  |
| SnowflakeNet [78]           | 7.21    | 4.29  | 9.16    | 8.08 | 7.89  | 6.07 | 9.23  | 6.55  | 6.40  |
| STTransformer-Scratch       | 7.37    | 4.28  | 9.45    | 8.21 | 7.99  | 6.35 | 9.38  | 6.77  | 6.50  |
| STTransformer-OcCo [48]     | 7.11    | 4.07  | 9.16    | 8.00 | 7.65  | 6.08 | 9.26  | 6.41  | 6.27  |
| 3D-OAE                     | 6.97    | 3.99  | 8.98    | 7.90 | 7.46  | 5.96 | 8.96  | 6.31  | 6.19  |

To improve the performance on point cloud completion task. And 3D-OAE also reduces the average CD by 0.14 compared with the state-of-the-art self-supervised method OcCo [48] which also takes a generation task to learn the self-supervised representation. These results prove that our generative self-supervised learning framework is able to learn a powerful representation which can bring significant improvement in downstream generative tasks, we may find a way to avoid spending lots of efforts on designing complex and heavy network structures. It's also worth noticing that we only use 1024 points during self-supervised learning, but transfer well to PCN dataset where the input has 2048 points. The visual comparisons of point cloud completion on PCN dataset is shown in Fig. 7.

F. Transfer to Real-World Data

To further evaluate the representation ability of our method, we use the encoder of 3D-OAE which is trained on the synthetic ShapeNet dataset to fine-tune on a real-world dataset ScanObjectNN, which contains 2902 scanned object instances from 15 categories. Due to the existence of background, occlusions and noise, this benchmark poses significant challenges to existing methods. We follow previous works to conduct...
Fig. 7. **Visual comparison of point cloud completion on PCN dataset.** The input and ground truth have 2048 and 16384 points, respectively. We show the completion comparison with current state-of-the-art completion methods like GRNet [74], PMP-Net [76] and SnowflakeNet [78] in (a), our 3D-OAE generates more smooth surfaces and more detailed structures. We also provide more completion results of our 3D-OAE in (b).

### TABLE VI
**CLASSIFICATION RESULTS ON THE SCANOBJECTNN DATASET.**

| Methods               | OBJ-BG | OBJ-ONLY | PB-T50-RS |
|-----------------------|--------|----------|-----------|
| PointNet [16]         | 73.3   | 79.2     | 68.0      |
| PointNet++ [17]       | 82.3   | 84.3     | 77.9      |
| SpiderCNN [27]        | 77.1   | 79.5     | 73.7      |
| PointCNN [30]         | 86.1   | 85.5     | 78.5      |
| DGCNN [21]            | 82.8   | 86.2     | 78.1      |
| BGA-DGCNN [79]        | -      | -        | 79.7      |
| BGA-PointNet++ [79]   | -      | -        | 80.2      |
| STransformer          | 79.86  | 80.55    | 77.24     |
| STransformer-OcCo [48]| 84.85  | 85.54    | 78.79     |
| STransformer-PBERT [5] | 87.43  | 88.12    | 83.07     |
| 3D-OAE                | **89.16** | **88.64** | **83.17** |

As shown in Table VI, our proposed 3D-OAE brings significant improvement of 9.03%, 8.09%, 5.93% over training a standard Transformer from scratch, and also outperforms the state-of-the-art methods OcCo and Point-BERT. The strong results show that using our self-supervised framework could learn meaningful information from artificial synthetic data and transfer it to real-world data, which could partly solve the domain gap between synthetic and scanned 3D data.

### G. Efficiency

**TABLE VII**
**EFFICIENCY COMPARISON RESULTS.**

| Methods               | OcCo [48] | Point-BERT [5] | 3D-OAE |
|-----------------------|-----------|----------------|--------|
| FLOPs (G)             | 8.7       | 9.79           | 0.65   |
| EpochTime (s)         | 1438      | 688            | 231    |

In Table VII, we show the efficiency of our 3D-OAE compared with other point cloud self-supervised learning methods. For a fair comparison, all the methods is trained using a single 2080Ti GPU. The results show that our proposed 3D-OAE has extremely low computational complexity of only 0.65G FLOPs, which is more than 10 times lower than OcCo and Point-BERT. And 3D-OAE also achieves about 6 times faster than OcCo and 3 times faster than Point-BERT. These
TABLE VIII
ABLATION STUDY ON FRAMEWORK DESIGN.

| Methods | Centralize Loss Occlusion | Linear Acc. | Fine-t. Acc. |
|---------|---------------------------|-------------|--------------|
| Solution A | X CD Rand | 20.8 | – |
| Solution B | ✓ EMD Rand | 88.4 | 92.4 |
| Solution C | ✓ CD Block | 91.5 | 92.7 |
| Solution D | ✓ CD Rand | 92.3 | 93.4 |

Outstanding results show the efficiency of 3D-OAE, due to the designed asymmetric encoder-decoder architecture of 3D-OAE. Using our 3D-OAE, it takes only less than one day to train on the full set of ShapeNet dataset for 300 epochs using a single 2080Ti. We can see the possibility of efficient pre-training on large-scale real scanned point cloud data using our framework.

Fig. 8. Visualization of learning curves in ModelNet40.

H. Learning Curves

In Fig. 8, we present the visualization of learning curves of both training a standard Transformer from scratch and using the trained encoder from 3D-OAE for fine-tuning. The results show that our proposed 3D-OAE can significantly accelerate convergence and improve accuracy. The training process of 3D-OAE is also more smooth and steady. These results demonstrate that 3D-OAE can learn powerful representation from unlabelled data and transfer well to downstream tasks.

I. Ablation study

We analyze the effectiveness of each design in 3D-OAE. For convenience, we conduct all experiments on the ShapeNet dataset, and report the classification accuracy of both Linear SVM and supervised fine-tuning in ModelNet40. By default, all the experiment settings remains the same as Sec. IV-A, except for the analyzed part.

1) Effect of each design in 3D-OAE: To evaluate the effectiveness of each design in 3D-OAE, we make comparisons between four different experimental solutions shown in Table VIII. Solution A is trained without centralizing point patches to seed points. Solution B is trained using Earth Mover's Distance as the loss function. Solution C is trained with a block occlusion strategy. And Solution D is our default setting. It’s clear that all the proposed designs in 3D-OAE can improve the performance of our method. And we find that using patch mix strategy [80], [81] fails to enhance the representation learning ability of 3D-OAE.

2) Occluding ratio: Table XI shows the numerical comparison of different occlusion ratios. With an occlusion ratio of 0, the auto-encoder fails to learn a powerful representation from the self-reconstruction task, which proves the effectiveness of our proposed occlusion strategy. Similar to MAE [7], we find that the occlusion ratio of 75% performs the best on both the linear accuracy and supervised fine-tuning accuracy. This is very different from BERT-style self-supervised learning works, where BERT masks only 15% of words and Point-BERT choose to occlude 25% to 45% of the point patches.

TABLE IX
ABLATION STUDY ON OCCLUSION RATIOS.

| Occlusion ratio | 0 | 0.5 | 0.65 | 0.75 | 0.85 |
|----------------|---|-----|------|------|------|
| Linear Acc.    | 59.2 | 90.9 | 91.1 | **92.3** | 90.7 |
| Fine-t. Acc.   | 92.1 | 93.1 | 92.7 | **93.4** | 93.0 |

TABLE X
ABLATION STUDY ON GROUP NUMBERS.

| Group | 16 | 32 | 64 |
|-------|----|----|----|
| Linear Acc. | 88.6 | 91.1 | **92.3** | 91.4 |
| Fine-t. Acc. | 92.6 | 92.5 | **93.4** | 92.6 |

TABLE XI
ABLATION STUDY ON THE PATCH SIZE.

| Patch size | 8 | 16 | 32 | 64 |
|------------|---|----|----|----|
| Linear Acc. | 75.6 | 90.4 | **92.3** | 91.5 |
| Fine-t. Acc. | 92.2 | 92.5 | **93.4** | 92.4 |

3) Grouping strategy: Table X and XI show the influence of the group numbers G and the patch size K. It’s clear that the chosen grouping strategy with a group number of 64 and a patch size of 32 achieves the best accuracy. For a fair comparison with other self-supervised methods, we adopt the same grouping strategy when reproducing them.

V. CONCLUSION AND FUTURE WORKS

In this paper, we present a novel point cloud self-supervised learning framework, named 3D Occlusion Auto-Encoder (3D-OAE). Our method shows powerful ability on transferring the learned representations to various downstream tasks, even to generative tasks and real-world tasks. Specifically, we conduct comprehensive experiments on six different downstream tasks. These results show that predicting complete shapes from highly occluded ones is an effective way for self-supervised representation learning for point clouds. Moreover, 3D-OAE also shows great
efficiency since our encoder only operates on 25% of the input point cloud patches.

Although our proposed 3D-OAE can learn powerful representations efficiently from the unlabelled point clouds, it still has some limitations and can be further improved in the future. For example, 3D-OAE separates the point cloud into patches and uses a self-reconstruction framework to learn patch-level representations. But the supervision is still constructed globally, which means that some local details cannot be fully considered. We consider introducing more local region constraints, such as conducting optimization losses between each local patch and its corresponding target.

REFERENCES

[1] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao, “Deep learning for image and point cloud fusion in autonomous driving: A review,” IEEE Transactions on Intelligent Transportation Systems, 2021.

[2] E. Alexiou, E. Upenik, and T. Ebrahimi, “Towards subjective quality assessment of point cloud imaging in augmented reality,” in 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2017, pp. 1–6.

[3] F. Pomerleau, F. Colas, and R. Siegwart, “A review of point cloud registration algorithms for mobile robotics,” Foundations and Trends in Robotics, vol. 4, no. 1, pp. 1–104, 2015.

[4] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, “PCT: Point cloud transformer,” Computer Vision and Image, vol. 7, no. 2, pp. 187–199, 2021.

[5] 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 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

[6] S. Huang, Y. Xie, S.-C. Zhu, and Y. Zhi, “Spatial-temporal self-supervised representation learning for 3D point clouds,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6535–6545.

[7] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, and R. Girshick, “Masked autoencoders are scalable vision learners,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 16000–16009.

[8] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, “Neural-Pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces,” in Proceedings of the 38th International Conference on Machine Learning, vol. 139, 2021.

[9] C. Chen, Z. Han, Y.-S. Liu, and M. Zwicker, “Unsupervised learning of fine structure generation for 3D point clouds by 2d projections matching,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12466–12477.

[10] T. Li, X. Wen, Y.-S. Liu, H. Su, and Z. Han, “Learning deep implicit functions for 3d shapes with dynamic code clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12840–12850.

[11] X. Wen, J. Zhou, Y.-S. Liu, H. Su, Z. Dong, and Z. Han, “3D shape reconstruction from 2D images with disentangled attribute flow,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

[12] B. Ma, Y.-S. Liu, and Z. Han, “Reconstructing surfaces for sparse point clouds with on-surface priors,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6315–6325.

[13] B. Ma, Y.-S. Liu, M. Zwicker, and Z. Han, “Surface reconstruction from point clouds by learning predictive context priors,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6326–6337.

[14] X. Liu, X. Liu, Y.-S. Liu, and Z. Han, “SPU-Net: Self-supervised point cloud upsampling by coarse-to-fine reconstruction with self-projection optimization,” IEEE Transactions on Image Processing, pp. 1–1, 2022.

[15] X. Li, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, “PMP-Net++: Point cloud completion by transformer-enhanced multi-step point moving paths,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2022.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” Advances in neural information processing systems, vol. 30, 2017.

[18] W. Wu, Z. Qi, and L. Fu Xin, “PointConv: Deep convolutional networks on 3D point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9961–9960.

[19] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, “Modeling point clouds with self-attention and gumbel subset sampling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3323–3332.

[20] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “Randla-Net: Efficient semantic segmentation of large-scale point clouds,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11108–11117.

[21] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” ACM Transactions On Graphics (ToG), vol. 38, no. 5, pp. 1–12, 2019.

[22] M. Gadelha, R. Wang, and S. Maji, “Multiresolution tree networks for 3D point cloud processing,” in Proceedings of the European Conference on Computer Vision, 2018, pp. 103–118.

[23] N. Verma, E. Boyer, and J. Verbeek, “FeastNet: Feature-steered graph convolutions for 3D shape analysis,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2598–2606.

[24] Y. Shen, C. Feng, Y. Yang, and D. Tian, “Mining point cloud local structures by kernel correlation and graph pooling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4548–4557.

[25] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, “Graph attention convolution for point cloud semantic segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10296–10305.

[26] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, “Pointwise convolutional neural networks,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 984–993.

[27] L. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, “SpiderCNN: Deep learning on point sets with parameterized convolutional filters,” in Proceedings of the European Conference on Computer Vision, 2018, pp. 87–102.

[28] Z. Zhang, B.-S. Hua, and S.-K. Yeung, “ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1607–1616.

[29] H. Su, V. Janpani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, “SplatNet: Sparse lattice networks for point cloud processing,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2530–2539.

[30] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “PointCNN: Convolution on x-transformed points,” Advances in Neural Information Processing Systems, vol. 31, 2018.

[31] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “KPConv: Flexible and deformable convolution for point clouds,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6411–6420.

[32] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, “Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence model,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 8778–8785.

[33] X. Wen, Z. Han, X. Liu, and Y.-S. Liu, “Point2spatialCapsule: Aggregating features and spatial relationships of local regions on point clouds using spatial-aware capsules,” IEEE Transactions on Image Processing, vol. 29, pp. 8855–8869, 2020.

[34] Y. Liu, B. Fan, S. Xiang, and E. Pan, “Relation-shape convolutional neural network for point cloud analysis,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8895–8904.

[35] M. Xu, Z. Zhou, and Y. Qiao, “Geometry sharing network for 3D point cloud classification and segmentation,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, 2020, pp. 12500–12506.

[36] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, “Rethinking network design and local geometry in point cloud: A simple residual mlp framework,” in International Conference on Learning Representations, 2021.
sification model on real-world data,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1588–1597.

[80] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6023–6032.

[81] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu, “PointCutmix: Regularization strategy for point cloud classification,” arXiv preprint arXiv:2101.01461, 2021.