Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling

Xumin Yu*,1, Lulu Tang*,1,2, Yongming Rao*,1, Tiejun Huang2,3, Jie Zhou1, Jiwen Lu†,1,2
1Tsinghua University 2BAAI 3Peking University

Abstract

We present Point-BERT, a new paradigm for learning Transformers to generalize the concept of BERT [8] to 3D point cloud. Inspired by BERT, we devise a Masked Point Modeling (MPM) task to pre-train point cloud Transformers. Specifically, we first divide a point cloud into several local point patches, and a point cloud Tokenizer with a discrete Variational AutoEncoder (dVAE) is designed to generate discrete point tokens containing meaningful local information. Then, we randomly mask out some patches of input point clouds and feed them into the backbone Transformers. The pre-training objective is to recover the original point tokens at the masked locations under the supervision of point tokens obtained by the Tokenizer. Extensive experiments demonstrate that the proposed BERT-style pre-training strategy significantly improves the performance of standard point cloud Transformers. Equipped with our pre-training strategy, we show that a pure Transformer architecture attains 93.8% accuracy on ModelNet40 and 83.1% accuracy on the hardest setting of ScanObjectNN, surpassing carefully designed point cloud models with much fewer hand-made designs. We also demonstrate that the representations learned by Point-BERT transfer well to new tasks and domains, where our models largely advance the state-of-the-art of few-shot point cloud classification task. The code and pre-trained models are available at https://github.com/lulutang0608/Point-BERT.

1. Introduction

Compared to conventional hand-crafted feature extraction methods, Convolutional Neural Networks (CNN) [20] is dependent on much less prior knowledge. Transformers [49] have pushed this trend further as a step towards no inductive bias with minimal man-made assumptions, such as translation equivalence or locality in CNNs. Recently, the structural superiority and versatility of standard Transformers are proved in both language [3, 8, 18, 25, 34] and image tasks [2, 6, 9, 45, 55, 66], and the capability of diminishing the inductive biases is also justified by enabling more parameters, more data [9], and longer training schedules. While Transformers produce astounding results in Natural Language Processing (NLP) and image processing, it is not well studied in the 3D community. Existing Transformer-based point cloud models [11, 63] bring in certain inevitable inductive biases from local feature aggregation [63] and neighbor embedding [11], making them deviate from the mainstream of standard Transformers. To this end, we aim to apply standard Transformers on point cloud directly with minimal inductive bias, as a stepping stone to a neat and unified model for 3D representation learning.

Figure 1. Illustration of our main idea. Point-BERT is designed for pre-training of standard point cloud Transformers. By training a dVAE via point cloud reconstruction, we can convert a point cloud into a sequence of discrete point tokens. Then we are able to pre-train the Transformers with a Mask Point Modeling (MPM) task by predicting the masked tokens.

Apparently, the straightforward adoption of Transformers does not achieve satisfactory performance on point cloud tasks (see Figure 5). This discouraging result is partially attributed to the limited annotated 3D data since pure Transformers with no inductive bias need massive training data. For example, ViT [9] uses ImageNet [20] (14M images) and JFT [41] (303M images) to train vision Transformers. In contrast, accurate annotated point clouds are relatively insufficient. Despite the 3D data acquisition is getting easy with the recent proliferation of modern scanning devices, labeling point clouds is still time-consuming, error-prone, and even infeasible in some extreme real-world scenarios. The difficulty motivates a flux of research into learning from unlabelled 3D data. Self-supervised pre-
training thereby becomes a viable technique to unleash the scalability and generalization of Transformers for 3D point cloud representation learning.

Among all the Transformer-based pre-training models, BERT [8] achieved state-of-the-art performance at its released time, setting a milestone in the NLP community. Inspired by BERT [8], we seek to exploit the BERT-style pre-training for 3D point cloud understanding. However, it is challenging to directly employ BERT on point clouds due to a lack of pre-existing vocabulary. In contrast, the language vocabulary has been well-defined (e.g., WordPiece in [8]) and off-the-shelf for model pre-training. In terms of point cloud Transformers, there is no pre-defined vocabulary for point clouds. A naive idea is to treat every point as a ‘word’ and mimic BERT [8] to predict the coordinates of masked points. Such a point-wise regression task surges computational cost quadratically as the number of tokens increases. Moreover, a word in a sentence contains basic contextual semantic information, while a single point in a point cloud barely entails semantic meaning.

Nevertheless, a local patch partitioned from a holistic point cloud contains plentiful geometric information and can be treated as a component unit. What if we build a vocabulary where different tokens represent different geometric patterns of the input units? At this point, we can represent a point cloud as a sequence of such tokens. Now, we can favorably adopt BERT and its efficient implementations almost out of the box. We hypothesize that bridging this gap is a key to extending the successful Transformers and BERT to the 3D vision domain.

Driven by the above analysis, we present Point-BERT, a new scheme for learning point cloud Transformers. Two essential components are conceived: 1) Point Tokenization: A point cloud Tokenizer is devised via a dVAE-based [37] point cloud reconstruction, where a point cloud can be converted into discrete point tokens according to the learned vocabulary. We expect that point tokens should imply local geometric patterns, and the learned vocabulary should cover diverse geometric patterns, such that a sequence of such tokens can represent any point cloud (even never seen before). 2) Masked Point Modeling: A ‘masked point modeling’ (MPM) task is performed to pre-train Transformers, which masks a portion of input point cloud and learns to reconstruct the missing point tokens at the masked regions. We hope that our model enables reasoning the geometric relations among different patches of the point cloud, capturing meaningful geometric features for point cloud understanding.

Both two designs are implemented and justified in our experiments. We visualize the reconstruction results both on the synthetic (ShapeNet [5]) and real-world (ScanObjectNN [47]) datasets in Figure 2. We observe that Point-BERT correctly predicts the masked tokens and infers diverse, holistic reconstructions through our dVAE decoder. The results suggest that the proposed model has learned inherent and generic knowledge of 3D point clouds, i.e, geometric patterns or semantics. More significantly, our model is trained on ShapeNet, the masked point predictions on ScanObjectNN reflect its superior performance on challenging scenarios with both unseen objects and domain gaps.

Our Point-BERT with a pure Transformer architecture and BERT-style pre-training technique achieves 93.8% accuracy on ModelNet40 and 83.1% accuracy on the complicated setting of ScanObjectNN, surpassing carefully de-
signed point cloud models with much fewer human pri-
or. We also show that the representations learned by
Point-BERT transfer well to new tasks and domains, where
our models largely advance the state-of-the-art of few-shot
point cloud classification task. We hope a neat and unified
Transformer architecture across images and point clouds
could facilitate both domains since it enables joint modeling
of 2D and 3D visual signals.

2. Related Work

Self-supervised Learning (SSL). SSL is a type of unsup-
vised learning, where the supervision signals can be gener-
a from the data itself [15]. The core idea of SSL is to de-
define a pretext task, such as jigsaw puzzles [29], colorization
[21], and optical-flow [27] in images. More recently, several
studies suggested using SSL techniques for point cloud un-
derstanding [10,14,22,30,36,38,39,42,50,54,57]. Example
3D pretext tasks includes orientation estimation [31], de-
formation reconstruction [1], geometric structural cues [43]
and spatial cues [28,40]. Inspired by the jigsaw puzzles in
images [29], [39] proposes to reconstruct point clouds from
the randomly rearranged parts. A contrastive learning
framework is proposed by DepthContrast [62] to learn rep-
resentations from depth scans. More recently, OcCo [50]
describes an encoder-decoder mechanism to reconstruct the
occluded point clouds. Different from these studies, we at-
tempt to explore a point cloud SSL model following the suc-
cessful Transformers [49].

Transformers. Transformers [49] have become the domi-
nant framework in NLP [3, 8, 18, 25, 34] due to its salient
benefits, including massively parallel computing, long-
distance characteristics, and minimal inductive bias. It has
intrigued various vision tasks [12, 19], such as object clas-
cification [6,9], detection [4,66] and segmentation [51,64].
Nevertheless, its applications on point clouds remain lim-
ited. Some preliminary explorations have been imple-
mented [11,59,63]. For instance, [63] applies the vectorized
self-attention mechanism to construct a point Transformer
layer for 3D point cloud learning. [11] uses a more typical
Transformer architecture with neighbor embedding to learn
point clouds. Nevertheless, prior efforts for Transformer-
based point cloud models more or less involve some in-
ductive biases, making them out of the line with standard
Transformers. In this work, we seek to continue the suc-
sess of standard Transformers and extend it to point cloud
learning with minimal inductive bias.

BERT-style Pre-training. The main architecture of BERT
[8] is built upon a multi-layer Transformer encoder, which is
first designed to pre-train bidirectional representations from
the unlabeled text in a self-supervised scheme. The primary
ingredient that helps BERT stand out and achieve impres-
sive performance is the pretext of Masked Language Mod-
eling (MLM), which first randomly masks and then recov-
ers a sequence of input tokens. The MLM strategy has also
inspired a lot of pre-training tasks [2, 7, 18, 25, 46]. Take
BEiT [2] for example, it first tokenizes the input image
into discrete visual tokens. After that, it randomly masks
some image patches and feeds the corrupted images into
the Transformer backbone. The model is trained to recover
the visual tokens of the masked patches. More recently,
MAE [13] presents a masked autoencoder strategy for im-
age representation learning. It first masks random patches
of the input image and then encourages the model to recon-
struct those missing pixels. Our work is greatly inspired
by BEiT [2], which encodes the image into discrete visual
tokens so that a Transformer backbone can be directly ap-
plied to these visual tokens. However, it is more challeng-
ing to acquire tokens for point clouds due to the unstruc-
tured nature of point clouds, which subsequently hinders
the straightforward use of BERT on point clouds.

3. Point-BERT

The overall objective of this work is to extend the BERT-
style pre-training strategy to point cloud Transformers. To
achieve this goal, we first learn a Tokenizer to obtain dis-
crete point tokens for each input point cloud. Mimicking
the ‘MLM’ strategy in BERT [8], we devise a ‘masked point
modeling’ (MPM) task to pre-train Transformers with the
help of those discrete point tokens. The overall idea of our
approach is illustrated in Figure 3.

3.1. Point Tokenization

Point Embeddings. A naive approach treats per point as
one token. However, such a point-wise reconstruction task
tends to unbearable computational cost due to the quadratic
complexity of self-attention in Transformers. Inspired by
the patch embedding strategy in Vision Transformers [9],
we present a simple yet efficient implementation that groups
each point cloud into several local patches (sub-clouds).
Specifically, given an input point cloud \( p \in \mathbb{R}^{N \times 3} \), we first sample \( g \) center points from the holistic point cloud
\( p \) via farthest point sampling (FPS). The k-nearest neighbor
(kNN) algorithm is then used to select the \( n \) nearest neighbor
points for each center point, grouping \( g \) local patches
(sub-clouds) \( \{ p_i \}_{i=1}^{g} \). We then make these local patches un-
biased by subtracting their center coordinates, disentangling
the structure patterns and spatial coordinates of the local
patches. These unbiased sub-clouds can be treated as words
in NLP or image patches in the vision domain. We further
adopt a mini-PointNet [32] to project those sub-clouds into
point embeddings. Following the practice of Transformers
in NLP and 2D vision tasks, we represent a point cloud as
a sequence of point embeddings \( \{ f_i \}_{i=1}^{g} \), which can be re-
Figure 3. The pipeline of Point-BERT. We first partition the input point cloud into several point patches (sub-clouds). A mini-PointNet [32] is then used to obtain a sequence of point embeddings. Before pre-training, a Tokenizer is learned through dVAE-based point cloud reconstruction (as shown in the right part of the figure), where a point cloud can be converted into a sequence of discrete point tokens; During pre-training, we mask some portions of point embeddings and replace them with a mask token. The masked point embeddings are then fed into the Transformers. The model is trained to recover the original point tokens, under the supervision of point tokens obtained by the Tokenizer. We also add an auxiliary contrastive learning task to help the Transformers to capture high-level semantic knowledge.

**Point Tokenizer.** Point Tokenizer takes point embeddings \{f_i\}_{i=1}^{N} as the inputs and converts them into discrete point tokens. Specifically, the Tokenizer Q_ϕ(z|f) maps point embeddings \{f_i\}_{i=1}^{N} into discrete point tokens \(z = [z_1, z_2, ..., z_p] \in \mathcal{V}^1\), where \(\mathcal{V}\) is the learned vocabulary with total length of \(N\). In this step, the sub-clouds \{p_i\}_{i=1}^{P} can be tokenized into point tokens \{z_i\}_{i=1}^{P}, relating to effective local geometric patterns. In our experiments, DGCNN [52] is employed as our Tokenizer network.

**Point Cloud Reconstruction.** The decoder \(P_{\phi}(p|z)\) of dVAE receives point tokens \{z_i\}_{i=1}^{P} as the inputs and learns to reconstruct the corresponding sub-clouds \{p_i\}_{i=1}^{P}. Since the local geometry structure is too complex to be represented by the limited \(N\) situations. We adopt a DGCNN [52] to build the relationship with neighboring point tokens, which can enhance the representation ability of discrete point tokens for diverse local structures. After that, a FoldingNet [57] is used to reconstruct the sub-clouds.

The overall reconstruction objective can be written as \(\mathbb{E}_{z \sim Q_\phi(z|p)} \log P_\phi(p|z)\), and the reconstruction procedure can be viewed as maximizing the evidence lower bound (ELB) of the log-likelihood \(P_\theta(p|\tilde{p})\) [35]:

\[
\sum_{(p_i, \tilde{p}_i) \in \mathcal{D}} \log P_\theta(p_i|\tilde{p}_i) \geq \sum_{(p_i, \tilde{p}_i) \in \mathcal{D}} \left( \mathbb{E}_{z \sim Q_\phi(z|p)} \log P_\phi(p_i|z) \right) - D_{KL}[Q_\phi(z|p_i), P_\phi(z|\tilde{p}_i)],
\]

where \(p\) denotes the original point cloud, \(\tilde{p}\) denotes the reconstructed point cloud. Since the latent point tokens are discrete, we cannot apply the reparameterization gradient to train the dVAE. Following [35], we use the Gumbel-softmax relaxation [17] and a uniform prior during dVAE training. Details about dVAE architecture and its implementation can be found in the supplementary.

3.2. Transformer Backbone

We adopt the standard Transformers [49] in our experiments, consisting of multi-headed self-attention layers and FFN blocks. For each input point cloud, we first divide it into \(g\) local patches with center points \{c_i\}_{i=1}^{G}. Those local patches are then projected into point embeddings \{f_i\}_{i=1}^{G} via a mini-PointNet [32], which consists of only MLP layers and the global maxpool operation. We further obtain the positional embeddings \{pos_i\}_{i=1}^{G} of each patch by applying an MLP on its center point \(c_i\). Formally, we define the input embeddings as \(x_i\), which is the combination of point embeddings \(f_i\) and positional embeddings \(pos_i\). Then, we send the input embeddings \(x_i\) into the Transformer. Following [8], we append a class token \(E[s]\) to the input sequences. Thus, the input sequence of Transformer can be expressed as \(H^0 = \{E[s], x_1, x_2, \cdots, x_g\}\). There are \(L\) layers of Transformer block, and the output of the last layer \(H^L = \{h_1^L, h_2^L, \cdots, h_N^L\}\) represents the global feature, along with the encoded representation of the input sub-clouds.

3.3. Masked Point Modeling

Motivated by BERT [8] and BEiT [2], we extend the masked modeling strategy to point cloud learning and devise a masked point modeling (MPM) task for Point-BERT.

**Masked Sequence Generation.** Different from the ran-
and then find its $M \in \{ \text{masked positions as directly apply such a block-wise masking strategy like [2]} \}$ region to generate the masked point cloud. In practice, we mask out all local patches in this region to generate the masked point cloud. In practice, we directly apply such a block-wise masking strategy like [2] to the inputs of the Transformer. Formally, we mark the masked positions as $M \in \{ 1, \ldots, g \}^{\lceil \rho g \rceil}$, where $r$ is the mask ratio. Next, we replace all the masked point embeddings with a same learnable pre-defined mask embeddings $E[M]$ while keeping its positional embeddings unchanged. Finally, the corrupted input embeddings $X^M = \{ x_i : i \notin M \}_{i=1}^g \cup \{ E[M] + \text{pos}_i : i \in M \}_{i=1}^g$ are fed into the Transformer encoder.

**Pretext Task Definition.** The goal of our MPM task is to enable the model to infer the geometric structure of missing parts based on the remaining ones. The pre-trained dVAE (see section 3.1) encodes each local patch into discrete point tokens, representing the geometric patterns. Thus, we can directly apply those informative tokens as our surrogate supervision signal to pre-train the Transformer.

**Point Patch Mixing.** Inspired by the CutMix [60,61] technique, we additionally devise a neat mixed token prediction task as an auxiliary pretext task to increase the difficulty of pre-training in our Point-BERT, termed as ‘Point Patch Mixing’. Since the information of the absolute position of each sub-cloud has been excluded by normalization, we can create new virtual samples by simply mixing two groups of sub-clouds without any cumbersome alignment techniques between different patches, such as optimal transport [61]. During pre-training, we also force the virtual sample to predict the corresponding tokens generated by the original sub-cloud to perform the MPM task. In our implementation, we generate the same number of virtual samples as the real ones to make the pre-training task more challenging, which is helpful to improve the training of Transformers with limited data as observed in [45].

**Optimization Objective.** The goal of MPM task is to recover the point tokens that are corresponding to the masked locations. The pre-training objective can be formalized as maximizing the log-likelihood of the correct point tokens $z_i$ given the masked input embeddings $X^M$:

$$
\max \sum_{X \in D} \mathbb{E}_M \left[ \sum_{i \in M} \log p \left( z_i | X^M \right) \right]. \quad (2)
$$

MPM task encourages the model to predict the masked geometric structure of the point clouds. Training the Transformer only with MPM task leads to an unsatisfactory understanding on high-level semantics of the point clouds, which is also pointed out by the recent work in 2D domain [65]. So we adopt the widely used contrastive learning method MoCo [14] as a tool to help the Transformers to better learn high-level semantics. With our point patch mixing technique, the optimization of contrastive loss encourages the model to pay attention to the high-level semantics of point clouds by making features of the virtual samples as closely as possible to the corresponding features from the original samples. Let $q_i$ be the feature of a mixed sample that comes from two other samples, whose features are $k_i^1$ and $k_i^2$ ($\{k_i\}$ is extracted by the momentum feature encoder [14]). Assuming the mixing ratio is $r$, the contrastive loss can be written as:

$$
L_q = -r \log \frac{\exp(qk_i^1 / \tau)}{\sum_{i=0}^K \exp(qk_i / \tau)} - (1-r) \log \frac{\exp(qk_i^2 / \tau)}{\sum_{i=0}^K \exp(qk_i / \tau)}, \quad (3)
$$

where $\tau$ is the temperature and $K$ is the size of memory bank. Coupling MPM objective and contrastive loss enables our Point-BERT to simultaneously capture the local geometric structures and high-level semantic patterns, which are crucial in point cloud representation learning.

### 4. Experiments

In this section, we first introduce the setups of our pre-training scheme. Then we evaluate the proposed model with various downstream tasks, including object classification, part segmentation, few-shot learning and transfer learning. We also conduct an ablation study for our Point-BERT.

#### 4.1. Pre-training Setups

**Data Setups.** ShapeNet [5] is used as our pre-training dataset, which covers over 50,000 unique 3D models from 55 common object categories. We sample 1024 points from each 3D model and divide them into 64 point patches (sub-clouds). Each sub-cloud contains 32 points. A lightweight PointNet [32] containing two-layer MLPs is adopted to project each sub-cloud into 64 point embeddings, which are used as input both for dVAE and Transformer.

**dVAE Setups.** We use a four-layer DGCNN [52] to learn the inter-patch relationships, modeling the internal structures of input point clouds. During dVAE training, we set the vocabulary size $N$ to 8192. Our decoder is also a DGCNN architecture followed by a FoldingNet [57]. It is worth noting that the performance of dVAE is susceptible to hyper-parameters, which makes that the configurations of image-based dVAE [35] cannot be directly used in our scenarios. The commonly used $\ell_1$-style Chamfer Distance loss is employed during the reconstruction procedure. Since the value of this $\ell_1$ loss is numerically small, the weight of KLD loss in Eq.1 must be smaller than that in the image tasks. We set the weight of KLD loss to 0 in the first 10,000 steps and gradually increased to 0.1 in the following 100,000 steps. The learning rate is set to 0.0005 with a cosine learning schedule with 60,000 steps warming up. We decay the temperature in Gumble-softmax function from 1
In this subsection, we report the experimental results on downstream tasks. Besides the widely used benchmarks, including classification and segmentation, we also study the model’s capacity on few-shot learning and transfer learning.

Object Classification. We conduct classification experiments on ModelNet40 [53]. In the classification task, a two-layer MLP with a dropout of 0.5 is used as our classification head. We use AdamW with a weight decay of 0.05 and a learning rate of 0.0005 under a cosine schedule to optimize the model. The batch size is set to 32.

The results are presented in Table 1. We denote our baseline model as “Transformer”, which is trained on ModelNet40 with random initialization. Several Transformer-based models are illustrated, where [ST] represents a standard Transformer architecture, and [T] denotes the Transformer model with some special designs or inductive biases. Although we mainly focus on pre-training for standard Transformers in this work, our MPM pre-training strategy is also suitable for other Transformer-based point cloud models [11, 63]. Additionally, we compare with a recent pre-training strategy OcCo [50] as a strong baseline of our pre-training method. For fair comparisons, we follow the details illustrated in [50] and use the Transformer-based decoder PoinTr [59] to perform their pretext task. Combining our Transformer encoder and PoinTr’s decoder, we conduct the completion task on ShapeNet, following the idea of OcCo. We term this model as “Transformer+OcCo”.

We see pre-training Transformer with OcCo improves 0.7%/1.0% over the baseline using 1024/4096 inputs. In comparison, our Point-BERT brings 1.8%/2.2% gains over that of training from scratch. We also observe that adding more points will not significantly improve the Transformer model without pre-training while Point-BERT models can be consistently improved by increasing the number of points. When we increase the density of inputs (4096), our Point-BERT achieves significantly better performance (93.4%) than that with the baseline (91.2%) and OcCo (92.2%). Given more input points (8192), our method can be further boosted to 93.8% accuracy on ModelNet40.

Table 1. Comparisons of Point-BERT with state-of-the-art models on ModelNet40. We report the classification accuracy (%) and the number of points in the input. [ST] and [T] represent the standard Transformers models and Transformer-based models with some special designs and more inductive biases, respectively.

| Method             | #point | Acc.  |
|--------------------|--------|-------|
| PointNet [32]      | 1k     | 89.2  |
| PointNet++ [33]    | 1k     | 90.5  |
| SO-Net [22]        | 1k     | 92.5  |
| PointCNN [23]      | 1k     | 92.2  |
| DGCNN [52]         | 1k     | 92.9  |
| DensePoint [24]    | 1k     | 92.8  |
| RSCNN [36]         | 1k     | 92.9  |
| KPConv [44]        | ~6.8k  | 92.9  |
| [T] PCT [11]       | 1k     | 93.2  |
| [T] PointTransformer [63]| –  | 93.7  |
| [ST] NPTC [11]     | 1k     | 91.0  |
| [ST] Transformer   | 1k     | 91.4  |
| [ST] Transformer + OcCo [50] | 1k | 92.1 |
| [ST] Point-BERT    | 1k     | 93.2  |
| [ST] Transformer   | 4k     | 91.2  |
| [ST] Transformer + OcCo [50] | 4k | 92.2 |
| [ST] Point-BERT    | 4k     | 93.4  |
| [ST] Point-BERT    | 8k     | 93.8  |

to 0.0625 in 100,000 steps following [35]. We train dVAE for a total of 150,000 steps with a batch size of 64.

MPM Setups. In our experiments, we set the depth for the Transformer to 12, the feature dimension to 384, and the number of heads to 6. The stochastic depth [16] with a 0.1 rate is applied in our transformer encoder. During MPM pre-training, we fix the weights of Tokenizer learned by dVAE. 25% ~ 45% input point embeddings are randomly masked out. The model is then trained to infer the expected point tokens at those masked locations. In terms of MoCo, we set the memory bank size to 16,384, temperature to 0.07, and weight momentum to 0.999. We employ an AdamW [26] optimizer, using an initial learning rate of 0.0005 and a weight decay of 0.05. The model is trained for 300 epochs with a batch size of 128.

4.2. Downstream Tasks

Few-shot Learning. We follow previous work [40] to evaluate our model under the few-shot learning setting. A typical setting is “K-way N-shot”, where $K$ classes are first randomly selected, and then ($N+20$) objects are sampled for each class [40]. The model is trained on $K \times N$ samples (support set), and evaluated on the remaining $20K$ samples (query set). We compare Point-BERT with OcCo [50], which achieves state-of-the-art performance on this task. In our experiments, we test the performance under “5way 10shot”, “5way 20shot”, “10way 10shot” and “10way 20shot”. We conduct 10 independent experiments under each setting and report the average performance as

Table 2. Few-shot classification results on ModelNet40. We report the average accuracy (%) as well as the standard deviation over 10 independent experiments.

| Method             | 5-way | 10-way |
|--------------------|-------|--------|
|                    | 10-shot | 20-shot | 10-shot | 20-shot |
| DGCNN-rand [50]    | 31.6 ± 2.8 | 40.8 ± 4.6 | 19.9 ± 2.1 | 16.9 ± 1.5 |
| DGCNN-OcCo [50]    | 90.6 ± 2.8 | 92.5 ± 1.9 | 82.9 ± 1.3 | 86.5 ± 2.2 |
| DGCNN-rand*        | 91.8 ± 3.7 | 93.4 ± 3.2 | 86.3 ± 6.2 | 90.9 ± 5.1 |
| DGCNN-OcCo*       | 91.9 ± 3.3 | 93.9 ± 3.1 | 86.4 ± 5.4 | 91.3 ± 4.6 |
| Transformer-rand   | 87.8 ± 5.2 | 93.3 ± 4.3 | 84.6 ± 5.5 | 89.4 ± 6.3 |
| Transformer-OcCo   | 94.0 ± 3.6 | 95.9 ± 2.3 | 89.4 ± 5.1 | 92.4 ± 4.6 |
| Point-BERT         | 94.6 ± 3.1 | 96.3 ± 2.7 | 91.0 ± 5.4 | 92.7 ± 5.1 |
well as the standard deviation over the 10 runs. We also reproduce DGCNN-rand and DGCNN-OcCo under the same condition for a fair comparison.

As shown in the Table 2, Point-BERT achieves the best in the few-shot learning. It obtains an absolute improvement of 6.8%, 3.0%, 6.4%, 3.3% over the baseline and 0.6%, 0.4%, 1.6%, 0.3% over the OcCo-based method on the four settings. The strong results indicate that Point-BERT learns more generic knowledge that can be quickly transferred to new tasks with limited data.

**Part Segmentation.** Object part segmentation is a challenging task aiming to predict a more fine-grained class label for every point. We evaluate the effectiveness of Point-BERT on ShapeNetPart [58], which contains 16,881 models from 16 categories. Following PointNet [32], we sample 2048 points from each model and increase the group number $g$ from 64 to 128 in the segmentation tasks. We design a segmentation head to propagate the group features to each point hierarchically. Specifically, features of $4^{th}$, $8^{th}$ and the last layer of Transformer are selected, denoted as $\{H^4 \} = \{h^4_i\}_{i=1}^g$, $H^8 = \{h^8_i\}_{i=1}^g$, $H^{12} = \{h^{12}_i\}_i^g$. Then we downsample the origin point cloud to 512 and 256 points via FPS, phrased as $P^{4} = \{p^4_{i} \}_{i=1}^{512}$ and $P^{8} = \{p^{8}_{i} \}_{i=1}^{256}$. We follow PointNet++ [33] to perform feature propagation between $H^4$ and $P^4$, $H^8$ and $P^8$. Here, we can obtain the upsampled feature map $\hat{H}^4$ and $\hat{H}^8$, which represent the features for the points in $P^4$ and $P^8$. Then, we can propagate the feature from $H^{12}$ to $H^4$ and finally to every point.

### Table 3. Part segmentation results on the ShapeNetPart dataset.
We report the mean IoU across all part categories mIoU$_C$ (%) and the mean IoU across all instance mIoU (%) as well as the IoU (%) for each categories.

| Methods          | mIoU$_C$ (%) | mIoU (%) | aero | bag  | cap  | car  | chair | earphone | guitar | knife | lamp | laptop | motor | mug   | pistol | rocket | skateboard | table |
|------------------|--------------|---------|------|------|------|------|-------|----------|--------|-------|------|--------|-------|-------|--------|--------|------------|-------|
| PointNet [32]    | 80.39        | 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++ [33]  | 81.85        | 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 [52]       | 82.33        | 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  |
| Transformer      | 83.42        | 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  |
| Transformer-OcCo | 83.42        | 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  |
| Point-BERT       | **84.11**    | **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  |

### Table 4. Classification results on the ScanObjectNN dataset.
We report the accuracy (%) of three different settings.

| Methods          | OBJ-BG (%) | OBJ-ONLY (%) | PB-T50-RS (%) |
|------------------|------------|--------------|---------------|
| PointNet [32]    | 73.3       | 79.2         | 68.0          |
| SpiderCNN [56]   | 77.1       | 79.5         | 73.7          |
| PointNet++ [33]  | 82.3       | 84.3         | 77.9          |
| PointCNN [23]    | 86.1       | 85.5         | 78.5          |
| DGCNN [52]       | 82.8       | 86.2         | 78.1          |
| BGA-DGCNN [47]   | –          | –            | 79.7          |
| BGA-PN++ [47]    | –          | –            | 80.2          |
| Transformer      | 79.86      | 80.55        | 77.24         |
| Transformer-OcCo | 84.85      | 85.54        | 78.79         |
| Point-BERT       | **87.43**  | **88.12**    | **83.07**     |

### Table 5. Ablation study.
We investigate the effects of different designs and report the classification accuracy (%) after fine-tuning on ModelNet40. All models are trained with 1024 points.

| Pretext tasks | MPM | Point Patch Mixing | Moco | Acc.  |
|---------------|-----|--------------------|------|-------|
| Model A       |     |                    |      | 91.41 |
| Model B       | ✓   |                    |      | 92.58 |
| Model C       | ✓   | ✓                  |      | 92.91 |
| Model D       | ✓   | ✓                  | ✓    | 93.24 |

| Augmentation   | mask type | mask ratio | replace | Acc.  |
|----------------|-----------|------------|---------|-------|
| Model B        | block mask | [0.25, 0.45] | No      | 92.58 |
| Model B        | block mask | [0.25, 0.45] | Yes     | 91.81 |
| Model B        | rand mask  | [0.25, 0.45]  | No      | 92.34 |
| Model B        | block mask | [0.55, 0.85]  | No      | 92.52 |
| Model B        | block mask | [0.25, 0.45]  | Yes     | 92.58 |
| Model B        | rand mask  | [0.25, 0.45]  | No      | 92.91 |
| Model B        | block mask | [0.55, 0.85]  | No      | 92.59 |

Two types of mIoU are reported in Table 3. It is clear that our Point-BERT outperforms PointNet, PointNet++, and DGCNN. Moreover, Point-BERT improves 0.69% and 0.5% mIoU over vanilla Transformers, while OcCo fails to improve baseline performance in part segmentation task.

### Transfer to Real-World Dataset.
We evaluate the generalization ability of the learned representation by pre-training the model on ShapeNet and fine-tuning it on ScanObjectNN [47], which contains 2902 point clouds from 15 categories. It is a more challenging dataset sampled from real-world scans containing background and occlusions. We follow previous works to conduct experiments on three main variants: OBJ-BG, OBJ-ONLY, and PB-T50-RS. The experimental results are reported in Table 4. As we can see, Point-BERT improves the vanilla Transformers by about 7.57%, 7.57%, and 5.83% on three variants.

Comparing the classification results on ModelNet40 (Table 1) and ScanObjectNN (Table 2), we observe that DGCNN outperforms PointNet++ (+2.4%) on the ModelNet40. While the superiority is degraded on the real-world dataset ScanObjectNN. As for Point-BERT, it achieves SOTA performance on both datasets, which strongly confirms the effectiveness of our method.
4.3. Ablation Study

Pretext Task. We denote model A as our baseline, which is the Transformer training from scratch. Model B presents pre-training Transformer with MPM pretext task. Model C is trained with more samples coming from ‘point patch mixing’ technique. Model D (the proposed method) is trained under the setting of MPM, point patch mixing, and MoCo. As can be seen in the upper part of Table 5, Model B with MPM improves the performance about 1.17%. By adopting point patch mixing strategy, Model C gets an improvement of 0.33%. With the help of MoCo [14], Model D further brings an improvement of 0.33%.

Masking Strategy. We visualize the point token prediction task in Figure 2. Our Transformer encoder can reasonably infer the point tokens of the missing patches. In practice, we reconstruct the local patches through the decoder of dVAE, based on the point tokens predicted by the Transformer encoder. Two masking strategies are explored: block-wise masking (block-mask) and random masking (rand-mask). The masking strategy determines the difficulty of the pretext task, influencing reconstruction quality and representations. We further investigate the effects of different masking strategies and provide the results in Table 5. We see that Model D with block-mask works better at the ratio of 25% ~ 45%. Unlike images, which can be split into regular non-overlapping patches, sub-clouds partitioned from the original point cloud often involve overlaps. Thus, rand-mask makes the task easier than block-mask, and further degrades the reconstruction performance. We also consider another type of augmentations: randomly replace some input embeddings with those from other samples.

4.4. Visualization

We visualize the learned features of two datasets via t-SNE [48] in Figure 4. In figure (a) and (b), the visualized features are from our Point-BERT (a) before fine-tuning and (b) after fine-tuning on ModelNet40. As can be seen, features from different categories can be well separated by our method even before fine-tuning. We also visualize the feature maps on the PB-T50-RS of ScanObjectNN in (c).

5. Conclusion and Discussions

We present a new paradigm for 3D point cloud Transformers through a BERT-style pre-training to learn both low-level structural information and high-level semantic feature. We observe a significant improvement for the Transformer on learning and generalization by comprehensive experiments on several 3D point cloud tasks. We show the potential of standard Transformers in 3D scenarios with appropriate pre-training strategy and look forward to further study on standard Transformers in the 3D domain.

We do not foresee any negative ethical/societal impacts at this moment. Although the proposed method can effectively improve the performance of standard Transformers on point clouds, the entire ‘pre-training + fine-tuning’ procedure is rather time-consuming, like other Transformers pre-training methods [2, 8, 13]. Improving the efficiency of the training process will be an interesting future direction.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 62152603, Grant U1813218, in part by a grant from the Beijing Academy of Artificial Intelligence (BAAI), and in part by a grant from the Institute for Guo Qiang, Tsinghua University.
References

[1] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In WACV, 2021. 3
[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254, 2021. 1, 3, 4, 5, 8
[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020. 1, 3
[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In ECCV, 2020. 3
[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Padmanabhan Ramakrishnan, Adrian Selli, and Shuran Song. A dataset of 3d mid-level scarce semantics from 10k scenes. In CVPR, 2021. 3
[6] Kai Han, Yunhe Wang, Hanting Chen, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image understanding. arXiv preprint arXiv:1810.04805, 2018. 1, 2, 3, 4, 5, 8
[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Justin Carion, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 tokens: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020. 1, 3
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. 1, 2, 3, 4, 5, 8
[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020. 1, 3
[10] Benjamin Eckart, Wentao Yuan, Chao Liu, and Ian Kautz. Self-supervised learning on 3d point clouds by learning discrete generative models. In CVPR, 2021. 3
[11] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud concrete generative models. In Self-supervised learning on 3d point clouds by learning discrete generative models. In CVPR, 2021. 3
[12] Kai Han, Yunhe Wang, Hanting Chen, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 tokens: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020. 1, 3
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 1, 3
[14] Kaiming He, Wing-Hung Khan, and Chen Change Loy. Deep residual learning for image recognition. In CVPR, 2016. 1, 3
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 1, 3
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 1, 3
[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016. 4
[18] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. TACL, 2020. 1, 3
[19] Salman Khan, Muszammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shabbaz Khan, and Mubarak Shah. Transformers in vision: A survey. arXiv preprint arXiv:2101.01169, 2021. 3
[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. NeurIPS, 2012. 1
[21] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In ECCV, 2016. 3
[22] Xiaojin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In CVPR, 2018. 3, 6
[23] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. NeurIPS, 2018. 6, 7
[24] Yongcheng Liu, Bin Fan, Gaoeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Denspoinet: Learning densely contextual representation for efficient point cloud processing. In ICCV, 2019. 6
[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019. 1, 3
[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In ICLR, 2018. 6
[27] Aravindh Mahendran, James Thevlis, and Andrea Vedaldi. Cross pixel optical-flow similarity for self-supervised learning. In ACCV, 2018. 3
[28] Benedikt Mersch, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Self-supervised point cloud prediction using 3d spatio-temporal convolutional networks. arXiv preprint arXiv:2110.04076, 2021. 3
[29] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In ECCV, 2016. 3
[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018. 3
[31] Omid Poursaeed, Tianxing Jiang, Hsin Chien, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. In 3DV, IEEE, 2020. 3
[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In NeurIPS, 2017. 6, 7
[33] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In NeurIPS, 2017. 6, 7
[34] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In NeurIPS, 2017. 6, 7
[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 1, 3

[35] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 4, 5, 6

[36] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *CVPR*, 2020. 3, 6

[37] Jason Tyler Rolfe. Discrete variational autoencoders. In *ICLR*, 2017. 2

[38] Aditya Sanghi. Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning. In *ECCV*, 2020. 3

[39] Jonathan Sauder and Bjarnie Sievers. Self-supervised deep learning on point clouds by reconstructing space. *NeurIPS*, 2019. 3

[40] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. *NeurIPS*, 2020. 3, 6

[41] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 1

[42] Ali Thabet, Humam Alwassel, and Bernard Ghanem. Self-supervised learning of local features in 3d point clouds. In *CVPRW*, 2020. 3

[43] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, 2019. 6

[44] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 5

[45] Tieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019. 3

[46] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 2, 7

[47] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008. 8

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Luca Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 3, 4

[49] Jianhao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Imagebert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021. 5

[50] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 4, 5, 6, 7

[51] Ali Thabet, Humam Alwassel, and Bernard Ghanem. Self-supervised pretraining for image embedding. In *arXiv preprint arXiv:2105.04553*, 2021. 1

[52] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021. 1

[53] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Fold- ingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 3, 4, 5

[54] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ToG*, 35(6):1–12, 2016. 7

[55] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointir: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, 2021. 3, 6

[56] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 5

[57] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujing Chen, Yanmei Meng, and Danfeng Wu. Pointcutmix: Regularization strategy for point cloud classification. *arXiv preprint arXiv:2101.01461*, 2021. 5

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

[59] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 1, 3, 6

[60] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 3

[61] Zhihong Wu, Shurong Song, Aditya Khosla, Fisher Yu, Lingguang Zhang, Xiaou Tang, and Jianxiong Xiao. 3d shapenet: A deep representation for volumetric shapes. In *CVPR*, 2015. 6

[62] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020. 3

[63] Yi Fan, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 7

[64] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 4, 5, 6, 7

[65] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021. 1

[66] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointir: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, 2021. 3, 6

[67] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 5

[68] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujing Chen, Yanmei Meng, and Danfeng Wu. Pointcutmix: Regularization strategy for point cloud classification. *arXiv preprint arXiv:2101.01461*, 2021. 5

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

[70] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 1, 3, 6

[71] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 3

[72] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021. 5

[73] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 1, 3