Point-McBert: A Multi-choice Self-supervised Framework for Point Cloud Pre-training

Kexue Fu$^{12*}$, Mingzhi Yuan$^{12*}$, Manning Wang$^{12†}$

1Digital Medical Research Center, School of Basic Medical Science, Fudan University, Shanghai, China
2Shanghai Key Laboratory of Medical Image Computing and Computer Assisted Intervention, Shanghai, China
kxfu18@fudan.edu.cn, mzyuan20@fudan.edu.cn, mnwang@fudan.edu.cn

Abstract

Masked language modeling (MLM) has become one of the most successful self-supervised pre-training task. Inspired by its success, Point-Bert, as a pioneer work in point cloud, proposed masked point modeling (MPM) to pre-train point transformer on large scale unannotated dataset. Despite its great performance, we find inherent difference between language and point cloud tends to cause ambiguous tokenization for point cloud. For point cloud, there doesn’t exist a gold standard for point cloud tokenization. Although Point-Bert introduce a discrete Variational AutoEncoder (dVAE) as tokenizer to allocate token ids to local patches, it tends to generate ambiguous token ids for local patches. We find this imperfect tokenizer might generate different token ids for semantically-similar patches and same token ids for semantically-dissimilar patches. To tackle above problem, we propose our Point-McBert, a pre-training framework with eased and refined supervision signals. Specifically, we ease the previous single-choice constraint on patches, and provide multi-choice token ids for each patch as supervision. Moreover, we utilize the high-level semantics learned by transformer to further refine our supervision signals. Extensive experiments on point cloud classification, few-shot classification and part segmentation tasks demonstrate the superiority of our method, e.g., the pre-trained transformer achieves 94.1% accuracy on ModelNet40, 84.28% accuracy on the hardest setting of ScanObjectNN and new state-of-the-art performance on few-shot learning. We also demonstrate that our method not only improves the performance of Point-Bert on all downstream tasks, but also incurs almost no extra computational overhead.

Introduction

Self-supervised pre-training (Grill et al. 2020; Chen and He 2021; He et al. 2021; Yang, Zhang, and Cui 2022; Ohri and Kumar 2021; Zhao and Dong 2020; Larsson, Maire, and Shakhnarovich 2017; He et al. 2020) is attracting growing attentions as it can transfer knowledge learned from large scale unlabeled dataset to boost the performance on downstream tasks. Most self-supervised pre-training methods are based on specific proxy tasks such as permutation prediction (Zhao and Dong 2020), image colorization (Larsson, Maire, and Shakhnarovich 2017), instance-level discrimination (Grill et al. 2020; Chen and He 2021; He et al. 2020). Among them, the masked language modeling (MLM) task proposed in BERT (Devlin et al. 2018) is currently one of the most successful proxy tasks and has been migrated to many other domains (Bao, Dong, and Wei 2021; Li et al. 2022; Yu et al. 2021b). Point-Bert (Yu et al. 2021b), as a pioneer work in point cloud learning, proposed a variant of MLM called masked point modeling (MPM) to pre-train point cloud transformers. Specifically, it first divides a point cloud into several local point patches and assigns a token id to each patch to convert the point cloud into multiply discrete tokens. Then it masks a proportion of tokens and pre-trains the model by recovering the masked tokens based on the transformer encoding results of the corrupted point cloud. Since there is not well-defined vocabulary to generate token ids for local point cloud patches, Point-Bert utilizes a pre-trained discrete Variational AutoEncoder (dVAE) (Rolfe 2016) as the tokenizer. However, we find such discrete tokenization in MPM hinders the framework to achieve better performance due to the difference between point cloud and language.

Languages are naturally composed of discrete words, which are strictly bijective to the token ids. However, there is no such gold standard for point clouds discretization as for language, making it inevitable to introduce noise to MPM. Specifically, dVAE-based tokenization tends to cause the following two kinds of ambiguities. (1) Semantically-dissimilar patches have the same token ids. Due to the lack of well-defined vocabulary, MPM adopts a pre-trained dVAE as tokenizer to generate token ids for local patches. However, the tokenizer focuses on local patches’ geometry but barely considers their semantics, resulting in some wrong token ids. For example, two patches shown in red in Figure 1 consist of similar geometric structure but have different semantics (landing gear and aero-engine). However, they are allocated with the same token ids (#3776) due to their similar geometric structure. (2) Semantically-similar patches have different token ids. As shown in Figure 2, semantically-similar patches of the aero-engine of the airplane are allocated with many different token ids (#599, #1274). It’s reasonable for them to have the same token ids because of their similar semantics and geometry to each other. However, the tokenizer neglects their relations and al-

*These authors contributed equally.
†Corresponding Author.
Inspired by Mc-Beit (Li et al. 2022), we propose an improved Bert-style point cloud pre-training framework called Point-McBert with eased and refined masked prediction targets to tackle the above problems. Specifically, since semantically-similar patches might have different token ids, we ease the previous strict single-choice constraint on patches. For each local patch, we utilize a probability distribution vector of token ids as supervision signal rather than previous single token id, which means each patch has Multi-Choice token ids. Moreover, we believe high-level perceptions produced by point cloud transformer can provide extra semantic supervision signals, which benefit our pre-training. Therefore, we refine the above supervision signals, i.e., the probability distribution vectors, using inter-patch semantic similarities, which come from our point cloud transformer. By considering high-level similarities, the ambiguities caused by local geometric similarity can be mitigated.

To verify the effectiveness of our framework, we pre-train a point cloud transformer on ShapeNet (Chang et al. 2015) and conduct extensive fine-tuning experiments on downstream tasks including point cloud classification, few-shot classification and part segmentation. Our framework not only improve the performance of previous Point-Bert on all downstream tasks, but also incurs almost no extra computational overhead during pre-training. Our Point-McBert achieves 94.1% accuracy on ModelNet40 (Wu et al. 2015) and 84.28% accuracy on the complicated setting of ScanObjectNN (Uy et al. 2019), outperforming a series of state-of-the-art methods. Our method also achieves the best results on many point cloud analysis tasks. In this work, we focus on the setting of Point-Bert and also utilize a standard transformer as our backbone. The proposed method differs from Point-Bert in applying a new tokenization mechanism.

Self-supervised pre-training

Pre-training on large-scale data and then fine-tuning on target tasks has been proved to be an effective paradigm to boost the performance of model on downstream tasks (He, Girshick, and Dollar 2019). However, although many efficient annotation tools (Wu et al. 2021; Girardeau-Montaut 2016) have been proposed, labeling a large-scale dataset is still costly especially for point cloud data. Therefore, self-supervised pre-training which pre-trains models without annotations has more potential. In the past few years, many self-supervised pre-training works had been proposed in both natural language processing (Devlin et al. 2018) and computer vision (Grill et al. 2020; Chen and He 2021; He et al. 2021; Yang, Zhang, and Cui 2022; Obri and Kumar 2021; Zhao and Dong 2020; Larsson, Maire, and Shakhnarovich 2017; He et al. 2020), which motivated works in point cloud (Yu et al. 2021b; Poursaeed et al. 2020; Huang et al. 2021; Xie et al. 2020; Wang et al. 2021). The core of self-supervised methods is designing a proxy task. For example, Poursaeed et al. (Poursaeed et al. 2020) pre-trained networks by predicting point cloud’s rotation. Inspired by the success of a series of contrastive learning-based self-supervised method (Grill et al. 2020; Chen and He 2021; He et al. 2021; He et al. 2020), PointContrast (Xie et al. 2020) and STRL (Huang et al. 2020) implemented a contrastive learning paradigm to learn deep representations from depth scan. OcCo (Wang et al. 2021) pre-trained their encoder by reconstructing occluded point clouds. Recently, Point-Bert (Yu et al. 2021b) which designed a proxy task called masked point modeling (MPM) to pre-train point cloud transformer, performed better than previous self-supervised pre-training methods for point cloud. However, we find improper tokenization in Point-Bert tends to result in ambiguous supervision, which hinder it to achieve better performance. In this work, we improve the problem of improper tokenizer in Point-Bert and devise a new framework to achieve better performance.

Related Works

Point cloud learning

Point cloud is an important type of geometric data structure, which is widely used in many applications, such as remote sensing (Liu et al. 2022; Han and Sánchez-Azofeifa 2022), autonomous driving (Li et al. 2019), robotics (Yang, Shi, and Carlone 2020) and medical image analysis (Shen et al. 2021). Since point cloud is irregular, researchers focused on converting point cloud into regular data such as 3D voxel and projected image in the past. Many voxel-based (Maturana and Scherer 2015; Wu et al. 2015) or view-based (Qi et al. 2016; Su et al. 2015) methods had been proposed, but didn't perform well. A pioneering work, PointNet (Qi et al. 2017a) utilized shared multi-layer perceptron with pooling to achieve permutation-invariant learning on point cloud, strongly outperformed previous work. Motivated by PointNet, a series of PointNet-style works (Qi et al. 2017b; Wang et al. 2019; Thomas et al. 2019) were proposed subsequently. Recently, inspired by success of vision transformer (VIT) (Dosovitskiy et al. 2020), many works (Zhao et al. 2021; Park et al. 2021; Guo et al. 2021) investigated the application of transformer (Vaswani et al. 2017) to point cloud learning. Due to the permutation-invariance of self-attention, PointTransformer (Zhao et al. 2021) designed a self-attention-based point transformer layer for 3D scene understanding. Yu et al. (Yu et al. 2021b) believed that the above transformer-based point cloud models deviate from the mainstream of standard transformer. Therefore, they used a standard transformer with minimal inductive bias in their Point-Bert (Yu et al. 2021b) and achieved new state-of-the-art on many point cloud analysis tasks. In this work, we focus on the setting of Point-Bert and also utilize a standard transformer as our backbone. The proposed method differs from Point-Bert in applying a new tokenization mechanism.

Preliminaries

Similar to Point-Bert (Yu et al. 2021b), we also adopt a Masked Point Modeling (MPM) paradigm as our proxy task to pre-train our model. Specifically, given a point cloud P, we first sample g center points \( \{c_i\}_{i=1}^g \) using farthest point sampling (Lang, Manor, and Avidan 2020) and then select their k-nearest neighbor points to build g local patches \( \{p_i\}_{i=1}^g \). The points in local patches are normalized by sub-
tracting their center points to further mitigate their coordinate bias. Then a mini-PointNet (Qi et al. 2017a) is used to map the normalized patches to a sequence of point embeddings \( \{ f_i \}_{i=1}^{g} \). After that, a tokenizer will take embeddings as inputs and generate token ids \( \{ \hat{z}_{i,k} \in \mathbb{R}^{V} \}_{i=1}^{g} \) for patches, where \( V \) denotes the length of the vocabulary \( V \). As done in other Bert-style works, we randomly mask a proportion of tokens and then send corrupted point embeddings into a backbone implemented by transformer encoder. The backbone will learn \( l_2 \)-normalized representations \( \{ h_i \}_{i=1}^{g} \) for both masked and unmasked tokens. Finally, we pre-train the backbone by predicting the masked tokens based on these representations. The objective of MPM can be formulated as follow:

\[
\mathcal{L}_{MPM} = \mathbb{E}_{i \in M} \left[ - \sum_{k=1}^{V} \hat{z}_{i,k} \log q(\hat{z}_{i,k} | h_i) \right]
\]

(1)

where \( M \) denotes the set of masked patches, \( q \) denotes a MLP head to predict the token id \( \hat{z} \) of masked patch \( p_i \) based on representation \( h_i \). For Point-Bert, the token id \( \hat{z} \) is a one-hot vector, which is derived from latent feature \( z_i \) in dVAE:

\[
\hat{z}_{i,k} = \begin{cases} 
1 & \text{if } k = \arg \max_j z_{i,j} \\
0 & \text{else}
\end{cases}
\]

(2)

where \( z_i \in \mathbb{R}^{V} \) denotes latent feature in the dVAE’s encoder. However, as mentioned above, this imperfect tokenizer will make some semantically-similar patches have different tokens and some semantically-dissimilar patches have the same tokens. To tackle these problems, we ease the token id \( \hat{z} \) to a soft vector satisfying \( \sum_k \hat{z}_{i,k} = 1 \), allowing patches correspond to multiply token ids. Moreover, we re-build \( \hat{z} \) based on the high-level semantic relationship for more accurate supervision. The details are presented in the next section.

**Method**

Our framework is shown in Figure 2 and we will introduce it in detail in this section. To verify the effectiveness of our proposed idea, we set Point-Bert as our baseline and modify it as little as possible.

Before pre-training, we train a dVAE (Rolfe 2016) as our tokenizer in the same way as in Point-Bert. During pre-training, transformer (Vaswani et al. 2017) encoder in our framework is trained on unlabeled data over MPM. For fine-tuning, the transformer encoder is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from downstream tasks. We first briefly introduce the implements of the tokenizer and the transformer encoder in section and , respectively. Then we provide details about our multi-choice strategy in section , which plays an important role in improving the tokenization.

**Tokenizer**

Our tokenizer is trained through dVAE-based point cloud reconstruction. As shown in Figure 2 the dVAE adopts an encoder-decoder architecture. The encoder is our tokenizer \( Q_\phi(z \mid p) \), which consists of a DGCNN (Wang et al. 2019) and Gumbel-softmax (Jang, Gu, and Poole 2016).
Local Centers

Mini-CLS Token

Normalize

P

likelihood optimizing the evidence lower bound (ELBO) of the log-and Welling 2013), we achieve reconstruction objective by DGCNN’s outputs. As with other VAE-style works (Kingma and Welling 2013), we achieve reconstruction objective by optimizing the evidence lower bound (ELBO) of the log-likelihood $P_{\theta}(p | \hat{p})$:

$$
\sum_{(p_i, \tilde{p}_i) \in D} \log P_{\theta}(p_i | \tilde{p}_i) \geq \sum_{(p_i, \tilde{p}_i) \in D} \left( E_{z_i^o \sim Q_{\phi}(z^o | p_i)} \log P_{\theta}(p_i | z_i^o) \right) - KL \left[ Q_{\phi}(z^o | p_i) \| P_{\theta}(z^o | \tilde{p}_i) \right]
$$

where $\tilde{p}$ denotes the reconstructed patches, $D$ denotes the training corpus, $P_{\theta}(z^o | \tilde{p}_i)$ follows a uniform distribution.

The former of ELBO represents the reconstruction loss and the latter represents the distribution loss. We follow (Yu et al. 2021a) to calculate the reconstruction loss and follow (Ramesh et al. 2021) to optimize the distribution difference between one-hot vectors and prior. Our tokenizer has the same architecture as in Point-Bert (Yu et al. 2021b) but outputs the latent features $\{z_i^g\}_{i=1}$ rather than the one-hot vectors $\{z_i^o\}_{i=1}$ during pre-training. More details about the architecture of tokenizer can refer to Point-Bert.

**Point transformer**

We utilize a standard transformer (Vaswani et al. 2017) following Point-Bert’s settings as our backbone. The transformer has 12 blocks, each of which consists of a multi-head self-attention layer and a feed forward layer. As shown in Figure 2, the transformer takes point embeddings $\{f_{cls}\} \cup \{f_i\}_{i=1}$ and center points $\{c_i\}_{i=1}$ as inputs, where $f_{cls}$ is a learnable classification token appended to the sequence of point embeddings $\{f_i\}_{i=1}$ produced by mini-PointNet (Qi et al. 2017a). Center points are used to generate positional encodings $\{pos_{cls}\} \cup \{pos_i\}_{i=1}$ through a MLP, where $pos_{cls}$ denotes the positional encoding for classification token, which is a learnable parameter. During pre-training, we mask a proportion of local point patches. We replace their corresponding point embeddings with same learnable mask embeddings $E[M]$ while keeping their positional embeddings unchanged. The corrupted embedding sequences are fed into transformer and the transformer outputs representations $\{h_{cls}\} \cup \{h_i\}_{i=1}$ for both masked and unmasked patches, where $h_{cls}$ represents the representation for classification, as part of the input of classification head in the downstream task.

Figure 2: **Overview of our proposed method, Point-McBert.** We improve the Point-Bert by incorporating eased and refined masked supervision signals. We utilize a softmax layer to ease the hard token ids into soft probability distribution vectors of token ids and use the encoded features generated by transformer encoder to refine the soft probability distribution vectors. During pre-training, a proportion of tokens are randomly masked and fed into transformer encoder. The transformer encoder is optimized by predicting the masked eased and refined token signals in the form of a soft-label cross-entropy loss.
Multi-choice tokenization

As mentioned above, there doesn’t exist a gold standard for point cloud tokenization. Therefore, it’s inevitable for tokenizer to produce improper supervision, including generating the same token ids for semantically-dissimilar patches and generating different token ids for semantically-similar patches. We observe that given a local patch, there may exist multiply suitable token ids as candidates as illustrated in Figure 1. Inspired by Mc-Beit (Li et al. 2022), we attempt to release the strict single-choice constraint on patches. Given a local patch \( p_i \), we don’t use unique token id as supervision any more. Instead, we predict the probability distribution vector \( p(z_i) \in \mathbb{R}^{|V|} \) of the token ids. Specifically, the probability distribution vector \( p(z_i) \in \mathbb{R}^{|V|} \) is generated by a softmax operation:

\[
    p(z_i)_k = \frac{\exp(z_{i,k}/\tau)}{\sum_{j=1}^{V} \exp(z_{i,j}/\tau)}
\]

where \( z_i \in \mathbb{R}^{|V|} \) denotes the latent feature in tokenizer, \( \tau \) is a temperature coefficient, which controls the smoothness of probability distribution. When \( \tau \) is small, \( p(z_i) \) tend to be a one-hot vector. When \( \tau \) is large, it is equivalent to incorporates more choices.

Moreover, to further tackle the ambiguity of token ids as shown in Figure 1, more high-level semantics should be incorporated. As analyzed in introduction, tokenizer focuses on encoding local geometry of local patches while ignoring the associations between patches, making semantically-dissimilar patches have the same token ids. To alleviate this problem, we use the representations learned by transformer to refine the probability distribution vectors as done in Li et al. (2022). Specifically, we use the cosine similarity between patch features learned by transformer to re-weight the probability distribution matrix \( p(z) \in \mathbb{R}^{g \times |V|} \). The similarity matrix \( W \in \mathbb{R}^{g \times g} \) is calculated as follows:

\[
    W_{i,k} = \frac{\exp < h_i, h_k >}{\sum_{j=1}^{g} \exp < h_i, h_j >}
\]

where \( h_j \) denotes the \( f \)-normalized representations learned by transformer, \( < \cdot > \) denotes the inner product between two vectors. The re-weighted probability distribution matrix \( W p(z) \) considers the inter-patch associations, making semantically-similar patches have more similar probability distributions and semantically-dissimilar patches have more discriminable probability distributions.

The final targets \( \hat{z} \) for prediction is a weighted sum of the reweighted probability distribution matrix \( p(z) \) and the refined probability distribution matrix \( W p(z) \):

\[
    \hat{z} = \omega p(z) + (1 - \omega) W p(z)
\]

where \( \omega \) is a coefficient balancing the low-level semantics from tokenizer and high-level semantics from inter-patch similarity. Slightly different from Mc-Beit (Li et al. 2022), we adopt a gradually decreasing \( \omega \) rather than a constant one. Specifically, at the beginning of pre-training, we set \( \omega = 1 \) due to inadequate training of transformer. After 30 epochs, transformer can well encode both the local geometry of local patches and dependencies between patches. Then the coefficient \( \omega \) begins to decrease in a cosine schedule to boost pre-training. We set the lower bound of \( \omega \) as 0.8. Our experiments also show this gradually decreasing paradigm plays an essential role in the pre-training. When we set \( \omega \) to a constant at the beginning, the transformer is easy to collapse due to the noisy semantics from initial transformer.

Experiment

In this section, we first introduce the setup of pre-training (section ). Then we conduct experiments on downstream tasks including point cloud classification (section ), few-shot learning (section ) and part segmentation (section ). We also present various ablations to analyze the effectiveness of our method (section ) and provide visualizations of our learned representations (section ). Furthermore, we also compare the computational overhead of our Point-McBert with PointBert (section ). Our code will be publicly available.

Pre-training setup

Dataset: For all experiments, we use ShapeNet (Chang et al. 2015) as the pre-training dataset, which originally contains over 50,000 CAD models from 55 common object categories. We randomly sample 1024 points from each CAD model and divide them into 64 local patches. Each patch contains 32 points. For the Bert-style pre-training, we randomly mask 25 ~ 45% patches for MPM in a blocked manner (Yu et al. 2021b).

Architecture: The mini-PointNet (Qi et al. 2017a) in our framework is a MLP followed by a global pooling layer. Our backbone network is a standard transformer (Vaswani et al. 2017) with 12 layers. We set the number of heads of each layer to 6 and dimension of features to 384. Our tokenizer is the encoder of dVAE, whose vocabulary size \( |V| \) is 8192.

Hyperparameters: We set the temperature coefficient \( \tau \) to 0.005 and set \( \omega \) to 1.0 as initial. After pre-training 30 epochs, we gradually decrease \( \omega \) in a cosine manner, whose lower bound is 0.8. We follow Point-Bert (Yu et al. 2021b) to train our dVAE. We use an AdamW (Loshchilov and Hutter 2017) optimizer with 0.0005 learning rate and 0.05 weight decay during pre-training. Our transformer is pre-trained for 300 epochs with a batch size of 128.

Point cloud classification

We perform our classification experiments on both the synthetic dataset ModelNet40 (Wu et al. 2015) and real-world dataset ScanObjectNN (Uy et al. 2019).

Experiment on synthetic dataset Dataset: ModelNet40 (Wu et al. 2015) is the most popular 3D dataset for point cloud classification, which contains 12311 CAD models from 40 object categories. We randomly sample 8k points from each CAD model for training and test. We follow previous setting (Qi et al. 2017a; Wu et al. 2021) to split the dataset into 9843/2468 for training and test.

Finetuning: We follow the setting in (Yu et al. 2021b) and employ a two-layer MLP with a dropout of 0.5 as the classification head. Specifically, we fine-tune the pre-trained backbone and the classification head using a AdamW (Loshchilov and Hutter 2017) optimizer with a weight decay
Table 1: Comparisons of the classification on ModelNet40 (Wu et al. 2015).

| Method                  | Acc (%) |
|-------------------------|---------|
| PointNet (Qi et al. 2017a) | 89.2    |
| PointNet++ (Qi et al. 2017b) | 90.5    |
| PointCNN (Li et al. 2018) | 92.2    |
| DGCNN (Wang et al. 2019) | 92.2    |
| DensePoint (Liu et al. 2019a) | 92.8    |
| KPConv (Thomas et al. 2019) | 92.9    |
| PTC (Guo et al. 2021) | 93.2    |
| PointTransformer (Zhao et al. 2021) | 93.7    |
| GLR (Rao, Lu, and Zhou 2020) | 92.9    |
| SRTL (Huang et al. 2021) | 93.1    |
| Baseline | 91.4    |
| OcCo (Wang et al. 2021) | 92.2    |
| Point-Bert (Yu et al. 2021b) | 93.8    |
| Ours | **94.1** |

0.05 and learning rate of 0.0005 under a cosine schedule. We set the batch size to 32.

**Competitors:** We compare our method with supervised methods, i.e., PointNet (Qi et al. 2017a), PointNet++ (Qi et al. 2017b), PointCNN (Li et al. 2018), DGCNN (Wang et al. 2019), DensePoint (Liu et al. 2019a), KPConv (Thomas et al. 2019), PTC (Guo et al. 2021), PointTransformer (Zhao et al. 2021) and recently published self-supervised pre-training methods (Yu et al. 2021b; Huang et al. 2020).

Moreover, to illustrate the effectiveness of pre-training, we also set a baseline model, which use the same backbone model as ours but trained from scratch.

**Results:** We adopt the classification accuracy as the evaluation metric and the experiment results are listed in Table 1. As observed from the results, our method obtain 94.1% accuracy on ModelNet40, outperforming the competitive methods and achieving new state-of-the-art performance. We also provide a convergence curve in Figure 3. We can see our method improves the baseline by 2.7% and converges to a high performance faster. Our method also outperforms Point-Bert by 0.3% with negligible extra computational overhead, which indicates the effectiveness of our multi-choices strategy.

**Experiment on real-world dataset**

**Dataset:** To test our method’s generalization to real scenes, we also conduct an experiment on a real-world dataset. ScanObjectNN (Uy et al. 2019) is a dataset modified from scene mesh datasets SceneNN (Hua et al. 2016) and ScanNet (Dai et al. 2017). It contains 2902 point clouds from 15 categories. We follow previous works (Yu et al. 2021b; Uy et al. 2019) to carry out experiments on three variants: OBJ-BG, OBJ-ONLY and PB-T50-RS, which denote the version with background, the version without background and the version with random perturbations, respectively. More details can be found in (Uy et al. 2019).

**Fine-tuning:** We implement the same setting as on the synthetic dataset.

Table 2: Comparisons of the classification on ScanObjectNN (Uy et al. 2019). We report the accuracy (%) of three different settings (OBJ-BG, OBJ-ONLY, PB-T50-LS).

| Method      | OBJ-BG | OBJ-ONLY | PB-T50-LS |
|-------------|--------|----------|-----------|
| PointNet    | 73.3   | 79.2     | 68.0      |
| SpiderCNN   | 77.1   | 79.5     | 73.7      |
| PointNet++  | 82.3   | 84.3     | 77.9      |
| PointCNN    | 86.1   | 85.5     | 78.5      |
| DGCNN       | 82.8   | 86.2     | 78.1      |
| BGA-DGCNN   | -      | -        | 79.7      |
| BGA-PointNet++ | -     | -        | 80.2      |
| Baseline    | 79.86  | 80.55    | 77.24     |
| OcCo        | 84.85  | 85.54    | 78.79     |
| Point-Bert  | 87.43  | 88.12    | 83.07     |
| Ours        | **88.98** | **90.02** | **84.28** |
Table 3: Few-shot classification results on ModelNet40 (Wu et al. 2015). We list the average accuracy (%) and the standard deviation over 10 independent experiments.

|                | 5-way | 10-way |
|----------------|-------|--------|
|                | 10-shot | 20-shot | 10-shot | 20-shot |
| 3D GAN         | 55.8±3.4 | 75.8±3.1 | 40.3±2.1 | 48.4±1.8 |
| FoldingNet     | 33.4±4.1 | 35.8±5.8 | 18.6±1.8 | 15.4±2.2 |
| L-GAN          | 41.6±5.3 | 46.2±6.2 | 32.9±2.9 | 25.5±3.2 |
| 3D-Caps        | 42.3±5.5 | 53.0±5.9 | 38.0±4.5 | 27.2±4.7 |
| PointNet       | 52.0±3.8 | 57.8±4.9 | 46.6±4.3 | 35.2±4.8 |
| PointNet++     | 38.5±4.4 | 42.4±4.5 | 23.1±2.2 | 18.8±1.7 |
| PointCNN       | 65.4±2.8 | 68.6±2.2 | 46.6±1.5 | 50.0±2.3 |
| RSCNN          | 65.4±8.9 | 68.6±7.0 | 46.6±4.8 | 50.0±7.2 |
| DGCNN          | 31.6±2.8 | 40.8±4.6 | 19.9±2.1 | 16.9±1.5 |
| Baseline       | 87.8±5.2 | 93.3±4.3 | 84.6±5.5 | 89.4±6.3 |
| 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 |
| Ours           | **97.1±1.8** | **98.3±1.2** | **92.4±4.3** | **94.9±3.7** |

significantly improves the baseline by 9.12%, 9.47%, and 7.04% on the three variants, which strongly confirms the generalization of our method. The above results also indicate that pre-training can transfer useful knowledge to downstream task and plays important roles in downstream task especially when downstream task is challenging.

**Few-shot learning**

Few-shot learning aims to tackle new tasks containing limited labeled training examples using prior knowledge. Here we conduct experiments on ModelNet40 (Wu et al. 2015) to evaluate our method. All our experiments follow K-way, m-shot setting (Sharma and Kaul 2020). Specifically, we randomly select K classes and sample m+20 samples for each class. The model is trained on K*m samples (support set), and evaluated on K*20 samples (query set). Here, we employ a deeper classification head with three layers and adopt fine-tuning settings identical to former classification experiments. We compare our method with other competitors under “5 way, 10 shot”, “5 way, 20 shot”, “10 way, 10 shot”, “10 way, 20 shot” settings and report the mean and standard deviation over 10 runs. Our competitors can be roughly divided into three categories: (1) unsupervised methods; (2) supervised methods; (3) self-supervised pre-training methods. For the unsupervised methods (Yang et al. 2018; Wu et al. 2016; Achariappas et al. 2018; Zhao et al. 2019), we train a linear SVM based on their extracted unsupervised representations. For supervised methods (Qi et al. 2017a,b; Wang et al. 2019, Li et al. 2018; Liu et al. 2019b) including our baseline, we train the model from scratch. For self-supervised pre-training methods (Yu et al. 2021b; Wang et al. 2021) including our Point-McBert, we fine-tune the model with pre-trained weight as initializations.

The results are shown in Table 3. We can see that when the labeled training data is insufficient, our method can still perform well. Our method has the highest average accuracy under four different settings. It’s noticeable that compared with other methods, our method commonly has a smaller standard deviation, which indicates our method is more stable. Our method also significantly improves the baseline by 9.3%, 5.2%, 7.8%, and 5.5%, which demonstrates the strong generalization ability of our method.

**Point cloud part segmentation**

Point cloud part segmentation is a challenging task aimed at predicting point-wise label for point cloud. Here, we evaluate our method on the widely used ShapeNetPart (Yi et al. 2016) dataset.

**Dataset:** ShapeNetPart contains 16881 CAD models from 16 object categories, annotated with 50 parts. We follow the setting in (Qi et al. 2017a) to randomly sample 2048 points from each CAD model for training and test. We also double the number of local patches to 128 in the pre-training for part segmentation.

**Fusion:** We follow Point-Bert (Yu et al. 2021b) to adopt a upsampling-based segmentation head for fine-tuning. We set the batch size to 16 and other settings are the same as for the classification tasks.

**Competitors:** We compare our method with some widely used methods (Qi et al. 2017a,b; Wang et al. 2019) and recently published self-supervised pre-training methods (Yu et al. 2021b; Wang et al. 2021). We also adopt a standard transformer trained from scratch as our baseline.

**Results:** We evaluate all the methods on two types of mIoU, i.e. mIoU_{C} and mIoU_{J}. The former denotes the mean mIoU across all part categories and the later denotes the mean mIoU across all instances. As shown in Table 4, our method achieves the best performance on both metrics. We also list the IoU for each category. As observed from these results, our method outperform the other methods on most categories. Our method also boosts baseline’s performance while OcCo fails to do it. Moreover, we visualize the segmentation ground truths and our predictions in Figure 4. Although the cases are challenging, our predictions are quite close to the ground truth.

**Ablation study**

In this section, we conduct extensive experiments on ModelNet40 (Wu et al. 2015) to study the effect of hyperparameters.

The **temperature coefficient** $\tau$ The hyperparameter $\tau$ controls the smoothness of supervision signals. When $\tau$ is small, we will get a sharp probability distribution of token ids. And conversely, when $\tau$ is large, the probability distribution tends to be a uniform distribution. To study the effect of $\tau$, we conduct an ablation study on $\tau$ and the result is shown on Table 5. We also add a single-choice version i.e., the same as in Point-Bert as competitor, which equals to $\tau \to 0$. Observed from the result, we find our multi-choice strategy performs best when temperature coefficient is set to 0.005 empirically and improves previous Bert-style pre-training.

The **weight coefficient** $\omega$ The hyperparameter $\omega$ balances the low-level semantics from tokenizer and high-level semantics from transformer encoder. We conduct an ablation
Table 4: Part segmentation results on the ShapeNetPart (Yi et al. 2016). We report the mean IoU across all part categories $mIoU_C$ (%) and the mean IoU across all instances $mIoU_I$ (%), as well as the IoU (%) for each categories.

| Category | PointNet | PointNet++ | DGCNN | Baseline | OcCo | Point-Bert | Ours |
|----------|----------|------------|-------|----------|------|-----------|------|
| aero     | 83.4     | 82.4       | 84    | 82.9     | 83.3 | 84.3      | **84.8** |
| bag      | 78.7     | 79         | 83.4  | **85.4** | 85.2 | 84.8      | 85.1 |
| cap      | 82.5     | 87.7       | 86.7  | 87.7     | 88.3 | **88.4** | **88.4** |
| car      | 74.9     | 77.3       | 77.8  | 78.8     | 79.9 | 79.8      | **80.6** |
| chair    | 89.6     | 90.8       | 90.6  | 90.5     | 90.7 | 91        | **91.5** |
| earphone | 73       | 71.8       | 74.7  | 80.8     | 74.1 | **81.7** | 80.5 |
| guitar   | 91.5     | 91         | 91.2  | 91.1     | **91.9** | 91.6 | 91.8 |
| knife    | 85.9     | 85.9       | 87.5  | 87.7     | 87.6 | **87.9** | 87.7 |
| lamp     | 80.8     | 83.7       | 82.8  | 85.3     | 84.7 | 85.2      | **85.5** |
| laptop   | 95.3     | 95.3       | 95.7  | 95.6     | 95.4 | 95.6      | **96.3** |
| motor    | 65.2     | 71.6       | 66.3  | 73.9     | 75.5 | 75.6      | **76.3** |
| mug      | 93       | 94.1       | 94.9  | 94.9     | 94.4 | **94.7** | 94.6 |
| pistol   | 81.2     | 81.3       | 81.1  | 83.5     | 84.1 | **84.3** | 83.7 |
| rocket   | 57.9     | 58.7       | **63.5** | 61.2     | 63.1 | 63.4      | 62.7 |
| skateboard | 72.8   | 76.4       | 74.5  | 74.9     | 75.7 | 76.3      | **78.4** |
| table    | 80.6     | 82.6       | **82.6** | 80.6     | 80.8 | 81.5      | 82.2 |

$mIoU_C$ | 80.4 | 81.9 | 82.3 | 83.4 | 83.4 | 84.1 | **84.4** |
$mIoU_I$ | 83.7 | 85.1 | 85.2 | 85.1 | 85.6 | 86.1 |

Ground Truth

Ours

Airplane   Bag   Cap   Car   Chair   Table   Mug   Motorbike

Ground Truth

Ours

Earphone   Guitar   Knife   Lamp   Laptop   Skateboard   Rocket   Pistol

Figure 4: Qualitative results for part segmentation. We visualize the part segmentation results across all 16 object categories.

Table 5: Ablation study on the temperature coefficient $\tau$. The ablation is conducted on point cloud classification downstream task on ModelNet40 (Wu et al. 2015).

|                | Single-choice (Point-Bert) | Multi-choice |
|----------------|----------------------------|--------------|
|                | 0.005 | 0.05 | 0.5 | 5 | 93.8 | 94.1 | 93.6 | 93.5 | 93.6

study on $\omega$ and the result is shown on Table 6. In this ablation study, we not only compare our Point-McBert under different $\omega$ settings, but also compare with a baseline, i.e., Point-Bert and a version without warm up strategy for $\omega$. In the version without warm up, we set $\omega = 0.8$ at the beginning of the pre-training rather than progressively decreasing $\omega$ after pre-training several epochs. Observed from the result, the version without warm up collapses during the pre-training, because the transformer is not well-trained enough to provides accurate semantics. Moreover, pre-training seems to perform better as the weight coefficient $\omega$ goes larger. But when $\omega = 1.0$, the supervision signals don’t incorporate high-level semantics, making the pre-training has the same performance as the baseline. As can be seen from the above, both the warm up strategy and the high-level semantics generated from transformer play important roles.

Masking strategy and masking ratio Bert-style pre-training follows a mask-and-then-predict paradigm, where masking strategy and masking ratio play important roles and determine the difficulty of the prediction. Here, we test two kinds of masking strategy: random masking and block-wise masking. The former strategy randomly selects a proportion of patches and then masks them. The latter strategy selects a proportion of adjacent patches and then masks them. We
In this paper, we propose Point-McBert, a Bert-style pre-training method for point cloud pre-training, which tackle the problem of improper tokenizer in previous work. We release the previous strict single-choice constraint on patches and utilize the probability distribution vector of token ids as supervision signals to avoid semantically-similar patches corresponding to different token ids. In addition, we use the high-level semantics generated by transformer to refine the probability distribution vector to further avoid semantically-dissimilar patches corresponding to the same token ids. Extensive experiments on different datasets and downstream tasks are conducted to evaluate the performance of our Point-McBert. The results show that our Point-McBert not only improves the performance of previous work on all downstream tasks with almost no extra computational overhead, but also achieve new state-of-the-art on point cloud classification and point cloud few-shot learning. The experiments also reveal that our pre-training method successfully transfer knowledge learned from unlabeled data to downstream tasks, which has great potential in point cloud learning.

Table 6: Ablation study on the weight coefficient $\omega$. The ablation is conducted on point cloud classification downstream task on ModelNet40 (Wu et al. 2015).

| $\omega$       | Acc (%) |
|----------------|---------|
| 0             | 93.5    |
| 0.2           | 93.4    |
| 0.4           | 93.6    |
| 0.6           | 93.6    |
| 0.8           | 94.1    |
| 1.0           | 93.8    |

Table 7: Ablation study on the masking strategy and masking ratio. The ablation is conducted on point cloud classification downstream task on ModelNet40 (Wu et al. 2015).

| Masking strategy | Masking Ratio (%) | Acc (%) |
|------------------|-------------------|---------|
| Block            | 15~25             | 93.8    |
| Block            | 25~45             | 94.1    |
| Block            | 55~75             | 93.7    |
| Random           | 15~25             | 93.6    |
| Random           | 25~45             | 93.5    |
| Random           | 55~75             | 93.9    |

Visualization

To further understand the effectiveness of our method, we visualize the learned features via t-SNE (Van der Maaten and Hinton 2008). Figure 5(a) shows our learned features before fine-tuning. As observed from it, features are well separated even trained without annotations, which are suitable for model initialization. Figure 5(b) and (c) provide visualization of features fine-tuned on ModelNet40 (Wu et al. 2015) and ScanObjectNN (Uy et al. 2019). As can be seen, features form multiply clusters that are far away from each other, indicating the effectiveness of our method.

Computational overhead

To demonstrate that our method improves the performance while introducing almost no extra computational overhead, we compare to the computational overhead of Point-Bert (Yu et al. 2021b) on the same device. The pre-training is implemented on an Intel Xeon Platinum 8260 CPU and two RTX3090 GPU. We select the time overhead of each pre-training epoch as our evaluation metric. As shown in Table 8, our method only incurs 1% extra time consuming, which is almost negligible.

Table 8: Comparisons of the time overhead of pre-training.

|                | Point-Bert | Ours |
|----------------|------------|------|
| Running time (s/epoch) | 143.48 | 144.94 |

Conclusion

In this paper, we propose Point-McBert, a Bert-style pre-training method for point cloud pre-training, which tackle

References

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L. 2018. Learning representations and generative models for 3d point clouds. In International conference on machine learning, 40–49. PMLR.

Bao, H.; Dong, L.; and Wei, F. 2021. Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.

Chen, X.; and He, K. 2021. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15750–15758.

Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; and Nießner, M. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE conference on computer vision and pattern recognition, 5828–5839.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Girardeau-Montaut, D. 2016. CloudCompare. France: EDF R&D Telecom ParisTech.
Figure 5: Visualization of feature distributions. We utilize t-SNE (Van der Maaten and Hinton 2008) to visualize the features learned by our Point-McBert. Features from different categories are visualized in different colors. (a) Features after pre-training; (b) Features fine-tuned on ModelNet40; (c) Features fine-tuned on ScanObjectNN.
Poursaeed, O.; Jiang, T.; Qiao, H.; Xu, N.; and Kim, V. G. 2020. Self-supervised learning of point clouds via orientation estimation. In 2020 International Conference on 3D Vision (3DV), 1018–1028. IEEE.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, 652–660.

Qi, C. R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; and Guibas, L. J. 2016. Volumetric and multi-view cnns for object classification on 3d data. In Proceedings of the IEEE conference on computer vision and pattern recognition, 5648–5656.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems, 30.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In International Conference on Machine Learning, 8821–8831. PMLR.

Rao, Y.; Lu, J.; and Zhou, J. 2020. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5376–5385.

Rolle, J. T. 2016. Discrete variational autoencoders. arXiv preprint arXiv:1609.02200.

Sharma, C.; and Kaul, M. 2020. Self-supervised few-shot learning on point clouds. Advances in Neural Information Processing Systems, 33: 7212–7221.

Shen, Z.; Feydy, J.; Liu, P.; Curiale, A. H.; San Jose Estepar, R.; San Jose Estepar, R.; and Niethammer, M. 2021. Accurate Point Cloud Registration with Robust Optimal Transport. Advances in Neural Information Processing Systems, 34: 5373–5389.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-view convolutional neural networks for 3d shape recognition. In Proceedings of the IEEE international conference on computer vision, 945–953.

Thomas, H.; Qi, C. R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. Kpconv: Flexible and deformable convolution for point clouds. In Proceedings of the IEEE/CVF international conference on computer vision, 6411–6420.

Uy, M. A.; Pham, Q.-H.; Hua, B.-S.; Nguyen, T.; and Yeung, S.-K. 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In Proceedings of the IEEE/CVF international conference on computer vision, 1588–1597.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. Journal of machine learning research, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Wang, H.; Liu, Q.; Yue, X.; Lasenby, J.; and Kusner, M. J. 2021. Unsupervised point cloud pre-training via occlusion completion. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 9782–9792.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (toG), 38(5): 1–12.

Wang, Y.; Wang, J.; Cao, Z.; and Barati Faridmani, A. 2022. Molecular contrastive learning of representations via graph neural networks. Nature Machine Intelligence, 4(3): 279–287.

Wu, J.; Zhang, C.; Xue, T.; Freeman, B.; and Tenenbaum, J. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. Advances in neural information processing systems, 29.

Wu, T.-H.; Liu, Y.-C.; Huang, Y.-K.; Lee, H.-Y.; Su, H.-T.; Huang, P.-C.; and Hsu, W. H. 2021. ReDAL: Region-based and Diversity-aware Active Learning for Point Cloud Semantic Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 15510–15519.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, 1912–1920.

Xie, S.; Gu, J.; Guo, D.; Qi, C. R.; Guibas, L.; and Litany, O. 2020. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In European conference on computer vision, 574–591. Springer.

Xu, Y.; Fan, T.; Xu, M.; Zeng, L.; and Qiao, Y. 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In Proceedings of the European Conference on Computer Vision (ECCV), 87–102.

Yang, H.; Shi, J.; and Carlone, L. 2020. Teaser: Fast and certifiable point cloud registration. IEEE Transactions on Robotics, 37(2): 314–333.

Yang, X.; Zhang, Z.; and Cui, R. 2022. TimeCLR: A self-supervised contrastive learning framework for univariate time series representation. Knowledge-Based Systems, 245: 108606.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In Proceedings of the IEEE conference on computer vision and pattern recognition, 206–215.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG), 35(6): 1–12.

Yu, X.; Rao, Y.; Wang, Z.; Liu, Z.; Lu, J.; and Zhou, J. 2021a. Pointrr: Diverse point cloud completion with geometry-aware transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 12498–12507.
Yu, X.; Tang, L.; Rao, Y.; Huang, T.; Zhou, J.; and Lu, J. 2021b. Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. arXiv preprint arXiv:2111.14819.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021. Point transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 16259–16268.

Zhao, Q.; and Dong, J. 2020. Self-supervised representation learning by predicting visual permutations. Knowledge-Based Systems, 210: 106534.

Zhao, Y.; Birdal, T.; Deng, H.; and Tombari, F. 2019. 3D point capsule networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1009–1018.