Boosting Point-BERT by Multi-Choice Tokens

Kexue Fu, Mingzhi Yuan, Shaolei Liu, and Manning Wang, Member, IEEE

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 the inherent difference between language and point cloud tends to cause ambiguous tokenization for point cloud, and no gold standard is available for point cloud tokenization. Point-BERT uses a discrete Variational AutoEncoder (dVAE) as tokenizer, but it might generate different token ids for semantically-similar patches and the same token ids for semantically-dissimilar patches. To tackle the above problems, we propose our McP-BERT, a pre-training framework with multi-choice tokens. Specifically, we ease the previous single-choice constraint on patch token ids in Point-BERT, 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 learning and new state-of-the-art performance on few-shot learning. Our method improves the performance of Point-BERT on all downstream tasks without extra computational overhead.

Index Terms—Point cloud, mask patch modeling, pretraining, multi-choice.

I. INTRODUCTION

SELF-SUPERVISED pre-training [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] 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 [6], image colorization [7], instance-level discrimination [1], [2], [8]. Among them, the masked language modeling (MLM) task proposed in BERT [13] is currently one of the most successful proxy tasks and has been migrated to many other domains [14], [15], [16]. As a pioneer work in point cloud learning, Point-BERT [16] 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 no well-defined vocabulary to generate token ids for local point cloud patches, Point-BERT employs a pre-trained discrete Variational AutoEncoder (dVAE) [17] 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 1, semantically-similar patches of the aero-engine of the airplane are allocated with many different token ids (#509, #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 allocates them with different token ids due to the interference of imperfect discretization and acquisition noise.

Inspired by mc-BEiT [15], we propose an improved BERT-style point cloud pre-training framework called McP-BERT with multi-choice tokens to tackle the above problems. Specifically, since semantically-similar patches might have different token ids, we ease the previous strict
Fig. 1. Visualization of improper token ids. To better visualize the improper tokenization, we provide two different views of the same point cloud. Different colors represent the different token ids, and the circles indicate the range of local patches. As shown in above two figures, semantically-dissimilar patches (landing gear and aero-engine, colored in red) have the same token ids (#3776), and semantically-similar adjacent patches (aero-engine, colored in purple and brown) have different token ids (#1274, #599).

single-choice constraint on patch token ids. 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, high-level perceptions produced by point cloud transformer can provide extra semantic supervision signals, which is beneficial to our pre-training process. Therefore, we refine the above supervision signals, i.e., the probability distribution vectors, using inter-patch semantic similarities from 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 [18] and conduct extensive fine-tuning experiments on downstream tasks including point cloud classification, few-shot classification and part segmentation. Our proposed framework not only improves the performance of previous Point-BERT on all downstream tasks, but also incurs almost no extra computational overhead during pre-training. Our McP-BERT achieves 94.1% accuracy on ModelNet40 [19] and 84.28% accuracy on the complicated setting of ScanObjectNN [20], outperforming a series of state-of-the-art methods. Our method also achieves new state-of-the-art on point cloud classification, few-shot classification and part segmentation.

pre-training methods which pre-train models without annotations have more potential. In the past few years, many self-supervised pre-training methods for point cloud learning have been proposed and the key point lies in designing a proper proxy task. For example, Poursaeed et al. [27] pretrained networks by predicting point cloud’s rotation. Inspired by the success of a series of contrastive learning-based self-supervised methods [1], [2], [8], [28], [29], [30], Point-Contrast [31] and STRL [32] implemented a contrastive learning paradigm to learn deep representations from depth scan. OcCo [33] pre-trained their encoder by reconstructing occluded point clouds. Recently, inspired great success of BERT [13], Point-BERT [16] designed a proxy task called masked point modeling (MPM), following a mask-and-then-predict paradigm to pre-train point cloud transformer. Since there is no vocabulary for officially breaking raw point clouds into small predefined chunks, Point-BERT incorporates a dVAE as tokenizer. It first divides a complete point cloud into several patches and then allocates token ids for patches based on dVAE’s latent features. However, we find that as we analyzed above, this improper tokenizer tends to produce ambiguous token ids for each patch. Therefore, we improve Point-BERT by introducing eased and refined token ids in this work.

III. PRELIMINARIES

Similar to Point-BERT [16], 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 \( \{c_i\}_{i=1}^{\gamma} \) using farthest point sampling [34] and then select their \( k \)-nearest neighbor points to build \( g \) local patches \( \{p_i\}_{i=1}^{\gamma} \). The points in local patches are normalized by subtracting their center points to further mitigate their coordinate bias. Then a mini-PointNet [35] is used to map the normalized patches to a sequence of patch embeddings \( \{f_i\}_{i=1}^{\gamma} \). The above process is shown in Figure 2(a). After that, a tokenizer will take embeddings as inputs and generate token ids \( \{\hat{z}_i \in \mathbb{R}^{|V|}\}_{i=1}^{\gamma} \) for corresponding 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 patch embeddings into a backbone implemented by transformer encoder. The backbone will learn \( \ell_2 \)-normalized representations \( \{h_i\}_{i=1}^{\gamma} \) for both masked and unmasked tokens, as shown in Figure 2(b). Finally, we pre-train the backbone by predicting the masked tokens based on these representations. The objective of MPM can be formulated as follow:

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

where \( M \) denotes the set of masked patches, \( q \) denotes a MLP head to predict the token id \( \hat{z}_i \) of masked patch \( p_i \), based on representation \( h_l \), \( \hat{z}_{i,k} \) denotes the \( k \)-th channel of \( \hat{z}_i \). For Point-BERT, the token id \( \hat{z}_i \) 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, $z_{i,j}$ denotes the $j$-th channel of $z_i$. Details regarding the architecture and training of the dVAE tokenizer can be found in Section IV-A. However, as mentioned above and shown in Figure 1, 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 $\hat{z}_i$ to a soft vector satisfying $\sum_k \hat{z}_{i,k} = 1$, allowing one patch corresponds to multiply token ids. Moreover, we re-build $\hat{z}_i$ based on the high-level semantic relationship for more accurate supervision. The details are presented in the next section.

IV. METHOD

The proposed framework is shown in Figure 2, and the details are introduced in the following part. 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 [17] as our tokenizer in the same way as in Point-BERT. During pre-training, transformer [36] 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. Then we provide details about our multi-choice strategy, which plays an important role in improving the tokenization.

A. Tokenizer

Our tokenizer is trained through dVAE-based point cloud reconstruction. As shown in Figure 2(d), the dVAE adopts an encoder-decoder architecture. The encoder is our tokenizer $Q_\phi(z^o \mid p)$, which consists of a DGCNN [37] and Gumbel-softmax [38]. During training dVAE, DGCNN $Q_\phi(z \mid p)$ takes a sequence of patch embeddings as input and outputs latent features $\{z_i\}_{i=1}^g$. Then the latent features $\{z_i\}_{i=1}^g$ is discretized into one-hot vectors $\{z_{i,k}^o\}_{k=1}^8$ through a Gumbel-softmax. The decoder $P_h(p \mid z^o)$ consists of a DGCNN followed by a FoldingNet [39]. DGCNN takes one-hot vectors $\{z_{i,k}^o\}_{k=1}^8$ as input and take full advantage of their neighborhood relationship in the feature space to enhance the representation of these discrete tokens. The following FoldingNet reconstructs the original point cloud according to DGCNN’s outputs. As with other VAE-style works [40], we achieve reconstruction optimization by optimizing the evidence lower bound (ELBO) of the log-likelihood $P_\theta(p \mid \tilde{p})$:

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

(3)

where $\tilde{p}$ denotes the reconstructed patches, $D$ denotes the training corpus, $P_\theta(z^o \mid \tilde{p}_i)$ follows a uniform distribution. The former of ELBO represents the reconstruction loss and the latter represents the distribution loss. We follow [41] to calculate the reconstruction loss and follow [42] to optimize the distribution difference between one-hot vectors and prior. Our tokenizer has the same architecture as in Point-BERT [16] but outputs the latent features $\{z_{i,k}^o\}_{i=1}^g$ rather than the one-hot vectors $\{z_{i,k}^o\}_{i=1}^g$ during pre-training. More details about the architecture of tokenizer can refer to Point-BERT.

B. Transformer Encoder

We utilize a standard transformer [36] following Point-BERT’s settings as our backbone, as shown in Figure 2(b). During pre-training, we randomly mask a proportion of patch embeddings and fed them including masked and unmasked patch embeddings into the backbone. Then the backbone outputs the deep representations $\{h_{cls}\} \cup \{h_i\}_{i=1}^g$, where $h_i$ represents the representations for both masked and unmasked patches and $h_{cls}$ represents the representation for classification, as part of the input of classification head in the downstream task. The deep representations extracted by transformer consider the associations between patches and contain more semantic information, which is suitable for many downstream tasks and refining the supervision signals. We will go over how to use these representations to refine the supervision signals in the following section.

C. Multi-Choice Tokenization

As mentioned above, there is no 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.

Inspired by mc-BERT [15], we attempt to release the strict single-choice constraint on token ids. As shown in Figure 2(c), given a local patch $p_i$, we don’t use discrete 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:

$$
P(z_i)_k = \frac{\exp(z_{i,k}/\tau)}{\sum_{j=1}^{|V|} \exp(z_{i,j}/\tau)}
$$

(4)

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, $P(z_i)$ tends 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 the 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 re-weight

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
patches. Then the coefficient local geometry of local patches and dependencies between
After 30 epochs, transformer can well encode both the warm-up strategy, i.e. gradually decreasing
similarity. Slightly different from mc-BEiT [15], we adopt a from tokenizer and high-level semantics from inter-patch
cosine schedule to boost pre-training. We set the lower bound
ω
where
P
the ease probability distribution matrix
ω
begins to decrease in a
discriminable probability distributions.

\[ W_{i,k} = \frac{\exp < h_i, h_k >}{\sum_{j=1}^8 \exp < h_i, h_j >} \]  \quad (5)

where \( h_i \) denotes the \( l_2 \)-normalized representations learned by transformer, \( < \cdot, \cdot > \) denotes the inner product between two vectors. The refined probability distribution matrix \( WP(z) \) 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 \( \tilde{z} \) for prediction is a weighted sum of the eased probability distribution matrix \( P(z) \) and the refined probability distribution matrix \( WP(z) \):

\[ \tilde{z} = \omega P(z) + (1 - \omega)WP(z) \]  \quad (6)

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 [15], we adopt a warm-up strategy, i.e. 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.

V. Experiment

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

Our code will be publicly available.

A. Pre-Training

For all experiments, we use ShapeNet [18] 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 block masking manner [16].

The mini-PointNet [35] in our framework is a MLP followed by a global pooling layer. Our backbone network is a standard transformer [36] 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. 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 [16] to train our dVAE. We use an AdamW [43] 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.
B. Point Cloud Classification

We perform our classification experiments on both the synthetic dataset ModelNet40 [19] and real-world dataset ScanObjectNN [20]. We follow the setting in [16] and employ a two-layer MLP with a dropout of 0.5 as the classification head. Specifically, we fine-tune the pretrained backbone and the classification head using a AdamW [43] optimizer with a weight decay 0.05 and learning rate of 0.0005 under a cosine schedule. We set the batch size to 32.

1) Experiment on Synthetic Dataset: ModelNet40 [19] 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 [25], [35] to split the dataset into 9843/2468 for training and test.

a) Competitors: We compare our method with supervised methods, i.e., PointNet [35], PointNet++ [44], PointCNN [45], DGCNN [37], DensePoint [46], PSNet [48], KPConv [47], PCT [49], PointTransformer [50] and recently published self-supervised pre-training methods [16], [32], [33], [51]. 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.

b) Results: We adopt the classification accuracy as the evaluation metric and the experiment results are listed in Table I. 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.

2) Experiment on Real-World Dataset: To test our method's generalization to real scenes, we also conduct an experiment on a real-world dataset. ScanObjectNN [20] is a dataset modified from scene mesh datasets SceneNN [52] and ScanNet [53]. It contains 2902 point clouds from 15 categories. We follow previous works [16], [20] 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 [20].

a) Competitors: We compare our method with supervised methods, i.e., PointNet [35], SpiderCNN [54], PointNet++ [44], PointCNN [45], DGCNN [37], models equipped with background-aware [20] (BGA) module and some state-of-the-art self-supervised pre-training methods [16], [33]. In this experiment, we also set a baseline as we do on ModelNet40.

b) Results: The classification accuracy on ScanObjectNN is shown in Table II. As observed from the results, all the methods perform worse on real-world dataset than on synthetic dataset ModelNet40, which is caused by less data for fine-tuning and the interference of noise, background, occlusion, etc. However, our method still achieves best performances on all three variants. It's worth noting that our method significantly improves the baseline by 9.12%, 9.47%,

---

**Table I**

| Method | Acc (%) |
|--------|---------|
| PointNet [35] | 89.2 |
| PointNet++ [44] | 90.5 |
| PointCNN [45] | 92.2 |
| DGCNN [37] | 92.2 |
| DensePoint [46] | 92.8 |
| KPConv [47] | 92.9 |
| PSNet [48] | 93.3 |
| PTC [49] | 93.2 |
| PointTransformer [50] | 93.7 |
| GLR [51] | 92.9 |
| SRTL [32] | 93.1 |
| Transformer* | 91.4 |
| OcCo [33] | 92.2 |
| Point-BERT [16] | 93.8 |
| Ours | **94.1** |

**Table II**

| Method | OBJ-BG | OBJ-ONLY | PB-T50-RS |
|--------|--------|----------|-----------|
| 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-PN++ | - | - | 80.2 |
| Transformer* | **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** |

---

Fig. 3. **Convergence curve.** We compare the performance of transformers training from scratch (green) and pre-training with our method (red) in terms of validation accuracy (%) on ModelNet40. The red dotted line denotes the best performance of our method while the green dotted line denotes the best performance of the baseline. Our method reaches baseline's best performance in only 19 epochs.
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.

C. Few-Shot Learning

Few-shot learning aims to tackle new tasks containing limited labeled training examples using prior knowledge.

1) Dataset: Here we conduct experiments on ModelNet40 [19] to evaluate our method. All our experiments follow K-way, m-shot setting [55]. 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). 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.

2) Competitors: 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 [39], [56], [57], [58], we train a linear SVM based on their extracted unsupervised representations. For supervised methods [35], [37], [44], [45], [59] including our baseline, we train the model from scratch. For self-supervised pre-training methods [16], [33] including our McP-BERT, we fine-tune the model with pre-trained weight as initialization.

3) Results: The results are shown in Table III. 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.

D. 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 [60] dataset. We follow Point-Bert [16] 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.

1) Dataset: ShapeNetPart contains 16881 CAD models from 16 object categories, annotated with 50 parts. We follow the setting in [35] 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.

2) Competitors: We compare our method with some widely used methods [35], [37], [44] and recently published self-supervised pre-training methods [16], [33]. We also adopt a standard transformer trained from scratch as our baseline.

3) Results: We evaluate all the methods on two types of mIoU, i.e., mIoU and mIoU. The former denotes the mean IoU across all part categories and the later denotes the mean IoU across all instances. As shown in Table IV, our method achieves the best performance on both metrics. Our method also boosts baseline’s performance while OcCo fails to do so. 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. In addition, we also visualized the difference in segmentation results between McP-BERT and Point-BERT, as shown in Figure 5. From the Figure, it is obvious that Point-BERT cannot correctly segment different parts, such as the top of the table, the lampshade of the lamp, the hilt of the hilt and the cushion of the chair. Our method can effectively handle the above situations.

E. Ablation Study

In this section, we conduct extensive experiments on ModelNet40 [19] to study the effect of hyperparameters.

1) 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 V. We also add a single-choice version i.e., the former denotes the mean IoU across all part categories and the later denotes the mean IoU across all instances. As shown in Table IV, our method achieves the best performance on both metrics. Our method also boosts baseline’s performance while OcCo fails to do so. 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. In addition, we also visualized the difference in segmentation results between McP-BERT and Point-BERT, as shown in Figure 5. From the Figure, it is obvious that Point-BERT cannot correctly segment different parts, such as the top of the table, the lampshade of the lamp, the hilt of the hilt and the cushion of the chair. Our method can effectively handle the above situations.

2) 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 study on $\omega$ and the result is shown on Table VI. In this ablation study, we not only compare our McP-BERT 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
Fig. 4. **Qualitative results for part segmentation.** We visualize the part segmentation results across all 16 object categories.

| TABLE IV |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| PART SEGMENTATION RESULTS ON THE SHAPENETPART [60]. 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 | 79.0 | 83.4 | 85.4 | 85.2 | 84.8 | 85.1 |
| cap | 82.5 | 87.7 | 86.7 | 87.7 | 88.3 | 88.0 | 88.4 |
| car | 74.9 | 77.3 | 77.8 | 78.3 | 79.9 | 79.8 | 80.6 |
| chair | 89.6 | 90.8 | 90.6 | 90.5 | 90.7 | 91.0 | 91.5 |
| earphone | 73.0 | 71.8 | 74.7 | 80.8 | 74.1 | 81.7 | 80.5 |
| guitar | 91.5 | 91.0 | 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 | 93.0 | 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 | 74.3 | 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 | 85.1 | 85.6 | 86.1 |

Fig. 5. **Visualization of part segmentation results differences between McP-BERT and Point-BERT.**

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.

3) **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
To further understand the effectiveness of our method, we visualize the learned features via t-SNE [62]. Figure 6(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 6(b) and (c) provide visualization of features fine-tuned on ModelNet40 [19] and ScanObjectNN [20]. As can be seen, features form multiply clusters that are far away from each other, indicating the effectiveness of our method.

G. 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 [16] on the same device. The pre-training is implemented on an Intel Xeon Platinum 8260 CPU and two RTX3090 GPUs. We select the time overhead of each pre-training epoch as our evaluation metric. As shown in Table VIII, our method only incurs 1% extra time consuming, which is almost negligible.

| 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    |

**Table VIII** Comparisons of the Time Overhead of Pre-Training

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

VI. Conclusion

In this paper, we propose McP-BERT, a BERT-style pre-training method for point cloud pre-training, which tackles the problem of improper tokenizer in previous work. We release the previous strict single-choice constraint on token ids 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 re-weight 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 McP-BERT. The results show that our McP-BERT 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.

**References**

[1] J.-B. Grill et al., “Bootstrap your own latent-a new approach to self-supervised learning,” in *Proc. Adv. Neural Info. Process. Syst.*, vol. 33, 2020, pp. 21271–21284.

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

[3] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” 2021, arXiv:2111.06377.

[4] X. Yang, Z. Zhang, and R. Cui, “TimeCLR: A self-supervised contrastive learning framework for univariate time series representation,” *Knowl.-Based Syst.*, vol. 245, Jun. 2022, Art. no. 108606.

[5] K. Ohri and M. Kumar, “Review on self-supervised image recognition using deep neural networks,” *Knowl.-Based Syst.*, vol. 224, Jul. 2021, Art. no. 107090.

[6] Q. Zhao and J. Dong, “Self-supervised representation learning by predicting visual permutations,” *Knowl.-Based Syst.*, vol. 210, Dec. 2020, Art. no. 106534.

[7] G. Larsson, M. Maire, and G. Shakhnarovich, “Colorization as a proxy task for visual understanding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 840–849.

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

[9] C. Liu, P. Liu, W. Zhao, and X. Tang, “Visual tracking by structurally optimizing pre-trained CNN,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 3153–3166, Sep. 2020.
Z. Du, X. Wang, and Q. Wang, “Self-supervised global spatio-temporal interaction pre-training for group activity recognition,” IEEE Trans. Circuits Syst. Video Technol., early access, Feb. 27, 2023, doi: 10.1109/TCSVT.2023.32349006.

J. Huang, Y. Huang, Q. Wang, W. Yang, and H. Meng, “Self-supervised representation learning for videos by segmenting via sampling rate order prediction,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 6, pp. 3475–3489, Jun. 2022.

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

H. Bao, L. Dong, S. Piao, and F. Wei, “BEiT: BERT-pretraining of image transformers,” 2021, arXiv:2106.08254.

X. Li, Y. Ge, K. Yi, Z. Hu, Y. Shan, and L.-Y. Duan, “McBEiT: Multi-choice discretization for image BERT pre-training,” 2022, arXiv:2203.15371.

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

J. Tyler Rolfe, “Discrete variational autoencoders,” 2016, arXiv:1609.02200.

A. X. Chang et al., “ShapeNet: An information-rich 3D model repository,” 2015, arXiv:1512.03012.

Z. Wu et al., “3D ShapeNets: A deep representation for volumetric shapes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1912–1920.

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

K. He, R. Girshick, and P. Dollár, “Rethinking ImageNet pre-training,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2019, pp. 4917–4926.

Y. Wu, X. Hu, Y. Zhang, M. Gong, W. Ma, and Q. Miao, “SACF-Net: Skip-attention based correspondence filtering network for point cloud registration,” IEEE Trans. Circuits Syst. Video Technol., early access, Jan. 16, 2023, doi: 10.1109/TCSVT.2023.3237328.

Z. Song, L. Zhao, and J. Zhou, “Learning hybrid semantic affinity for point cloud segmentation,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 7, pp. 4599–4612, Jul. 2022.

D. Li, G. Shi, Y. Wu, Y. Yang, and M. Zhao, “Multi-scale neighborhood feature extraction and aggregation for point cloud segmentation,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 6, pp. 2175–2191, Jun. 2021.

T. Wu et al., “RedAL: Region-based and diversity-aware active learning for point cloud semantic segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 15490–15499.

D. Girardeau-Montaut, “Cloudcompare,” EDF R&D Telecom ParisTech, Paris, France, Tech. Rep. 11, 2016.

O. Poursaeed, T. Jiang, H. Qiao, N. Xu, and V. G. Kim, “Self-supervised learning of point clouds via orientation estimation,” in Proc. Int. Conf. 3D Vis. (3DV), Nov. 2020, pp. 1018–1028.

Z. Fu, Z. Mao, C. Yan, A. Liu, H. Xie, and Y. Zhang, “Self-supervised synthesis ranking for deep metric learning,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 7, pp. 4736–4750, Jul. 2022.

P. Xu, Z. Song, Q. Yin, Y. Song, and L. Wang, “Deep self-supervised representation learning for free-hand sketch,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 4, pp. 1503–1513, Apr. 2021.

L. Tao, X. Wang, and T. Yamashiki, “An improved inter-intra contrastive learning framework on self-supervised video representation,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 8, pp. 5266–5280, Aug. 2022.

S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, “Pointcontrast: Unsupervised pre-training for 3D point cloud understanding,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 574–591.

S. Huang, Y. Xie, S. Zhu, and Y. Zhu, “Spatio-temporal self-supervised representation learning for 3D point clouds,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 6515–6525.
[57] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning representations and generative models for 3D point clouds,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 40–49.
[58] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, “3D point capsule networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 1009–1018.
[59] Y. Liu, B. Fan, S. Xiang, and C. Pan, “Relation-shape convolutional neural network for point cloud analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 8887–8896.
[60] L. Yi et al., “A scalable active framework for region annotation in 3D shape collections,” ACM Trans. Graph., vol. 35, no. 6, pp. 1–12, Nov. 2016.
[61] Y. Wang, J. Wang, Z. Cao, and A. Barati Farimani, “Molecular contrasive learning of representations via graph neural networks,” Nature Mach. Intell., vol. 4, no. 3, pp. 279–287, Mar. 2022.
[62] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 11, pp. 1–27, 2008.

Kexue Fu received the bachelor’s degree in electronic and information engineering from the Wuhan University of Technology, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Basic Medical Sciences, Fudan University. His research interests include computer vision and image-guided intervention.

Mingzhi Yuan received the B.S. degree in communication engineering from the Harbin Institute of Technology (HIT) in 2020. He is currently pursuing the Ph.D. degree with the School of Basic Medical Sciences, Fudan University. His research interests include 3D vision and medical image processing.

Shaolei Liu received the B.S. degree in communication engineering from the Ocean University of China, Qingdao, China, in 2019. He is currently pursuing the Ph.D. degree in biomedical engineering with the School of Basic Medical Sciences, Fudan University. His research interests include medical image processing, image fusion-domain adaptation, and computer vision.

Manning Wang (Member, IEEE) received the B.S. and M.S. degrees in power electronics and power transmission from Shanghai Jiaotong University, Shanghai, China, in 1999 and 2002, respectively, and the Ph.D. degree in biomedical engineering from Fudan University, China, in 2011. He is currently a Professor of biomedical engineering with the School of Basic Medical Sciences, Fudan University. His research interests include medical image processing, image-guided intervention, and computer vision.