GeoMAE: Masked Geometric Target Prediction for Self-supervised Point Cloud Pre-Training

Xiaoyu Tian\textsuperscript{1}  Haoxi Ran\textsuperscript{2}  Yue Wang\textsuperscript{3}  Hang Zhao\textsuperscript{1}\textsuperscript{*}

\textsuperscript{1}IIS, Tsinghua University  \textsuperscript{2}CMU  \textsuperscript{3}NVIDIA

Abstract

This paper tries to address a fundamental question in point cloud self-supervised learning: what is a good signal we should leverage to learn features from point clouds without annotations? To answer that, we introduce a point cloud representation learning framework, based on geometric feature reconstruction. In contrast to recent papers that directly adopt masked autoencoder (MAE) and only predict original coordinates or occupancy from masked point clouds, our method revisits differences between images and point clouds and identifies three self-supervised learning objectives peculiar to point clouds, namely centroid prediction, normal estimation, and curvature prediction. Combined, these three objectives yield an nontrivial self-supervised learning task and mutually facilitate models to better reason fine-grained geometry of point clouds. Our pipeline is conceptually simple and it consists of two major steps: first, it randomly masks out groups of points, followed by a Transformer-based point cloud encoder; second, a lightweight Transformer decoder predicts centroid, normal, and curvature for points in each voxel. We transfer the pre-trained Transformer encoder to a downstream perception model. On the nuScene Datset, our model achieves 3.38 mAP improvement for object detection, 2.1 mIoU gain for segmentation, and 1.7 AMOTA gain for multi-object tracking. We also conduct experiments on the Waymo Open Dataset and achieve significant performance improvements over baselines as well.\textsuperscript{1}

1. Introduction

While object detection and segmentation from LiDAR point clouds have achieved significant progress, these models usually demand a large amount of 3D annotations that are hard to acquire. To alleviate this issue, recent works explore learning representations from unlabeled point clouds, such as contrastive learning \cite{20, 46, 53}, and mask modeling \cite{32, 51}. Similar to image-based representation learning settings, these representations are transferred to downstream tasks for weight initialization. However, the existing self-supervised pretext tasks do not bring adequate improvements to the downstream tasks as expected.

Contrastive learning based methods typically encode different ‘views’ (potentially with data augmentation) of point clouds into feature space. They bring features of the same point cloud closer and make features of different point clouds ‘repel’ each other. Other recent works use masked modeling to learn point cloud features through self-reconstruction \cite{32, 51}. That is, randomly sparsified point clouds are encoded by point cloud feature extractors, followed by a reconstruction module to predict original point clouds. These methods, when applied to point clouds, ignore the fundamental difference of point clouds from images – point clouds provide scene geometry while images provide brightness. As shown in Figure 1, this modality disparity hampers direct use of methods developed in the image domain for point cloud domain, and thus calls for novel self-supervised objectives dedicated to point clouds.

Inspired by modeling and computational techniques in geometry processing, we introduce a self-supervised learning framework dedicated to point clouds. Most importantly, we design a series of prediction targets which describe the fine-grained geometric features of the point clouds. These

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{figure1.pdf}
\caption{Pixel value regression has been proved effective in masked autoencoder pre-training for images. We find this practice ineffective in point cloud pre-training and propose a set of geometry aware prediction targets.}
\end{figure}

*Corresponding to: hangzhao@mail.tsinghua.edu.cn
\textsuperscript{1}Our code is available at https://github.com/Tsinghua-MARS-Lab/GeoMAE.
geometric feature prediction tasks jointly drive models to recognize different shapes and areas of scenes. Concretely, our method starts with a point cloud voxelizer, followed by a feature encoder to transform each voxel into a feature token. These feature tokens are randomly dropped based on a pre-defined mask ratio. Similar to the original MAE work [18], visible tokens are encoded by a Transformer encoder. Then a Transformer decoder reconstructs the features of the original voxelized point clouds. Finally, our model predicts point statistics and surface properties in parallel branches.

We conduct experiments on a diverse set of large-scale point cloud datasets including nuScenes [4] and Waymo [39]. Our setting consists of a self-supervised pre-training stage and a downstream task stage (3D detection, 3D tracking, segmentation), where they share the same point cloud backbone. Our results show that even without additional unlabeled point clouds, self-supervised pre-training with objectives proposed by this paper can significantly boost the performance of 3D object detection. To summarize, our contributions are:

- We introduce geometry aware self-supervised objectives for point clouds pre-training. Our method leverages fine-grained point statistics and surface properties to enable effective representation learning.

- With our novel learning objectives, we achieve state-of-the-art performance compared to previous 3D self-supervised learning methods on a variety of downstream tasks including 3D object detection, 3D/BEV segmentation, and 3D multi-object tracking.

- We conduct comprehensive ablation studies to understand the effectiveness of each module and learning objective in our approach.

2. Related Work

2.1. Self-Supervised Learning for Point Clouds

Self-supervised learning for point cloud [14, 20, 23, 32, 37, 38, 42, 46, 51, 53] has drawn considerable attention due to the expensive cost of labeling the 3D point cloud. Some are based on contrastive paradigms [20, 46, 53]. PointContrast [46] learns from correspondences between different point cloud views with a contrastive loss. DepthContrast [53] considers different depth map as an instance and discriminating between them to learn the representation. STRL [20] learns the invariant representation from two augmented temporally-correlated frames from a 3D point cloud sequence. Others [14, 37, 38] utilize a pretext task to promote self-supervised representation learning. [38] phrases the pretext task as a part segmentation task by displacing the part of the parts of the point cloud and then predicting their ordering labels. [14] squeezes learned representations through an implicitly defined parametric discrete generative model bottleneck. [37] introduces a bidirectional reasoning between local and global to capture the underlying semantic knowledge. Motivated by the huge success of 2D masked image modeling, masked point modeling methods [32, 51] have been proposed recently. Point-BERT [51] adopts a BERT-style pre-training strategy by predicting discrete tokens of masked input point parts. Point-MAE [32] simply predicts the original coordinates of the masked point patches tokens.

2.2. Masked Image Modeling

Motivated by the success of BERT [12] for masked language modeling, Masked Image Modeling (MIM) [1, 3, 6, 13, 18, 24, 44, 47, 54] becomes a popular pretext task for self-supervised visual representation learning. BEiT [3] first introduces the pre-training pattern of BERT into the computer vision filed by masking out the random image patches and predicts discrete tokens. MAE [18] and SimMIM [47] both propose to predict the raw pixels of the masked patches. Compared with SimMIM, MAE is more pre-training efficient because it only takes the visible token as the input of the encoder and passes all tokens through a light-weight decoder. Many following works use such asymmetric architecture but explore different prediction targets. MaskFeat [44] uses low-level local features HOG [10] as the prediction target. A²MIM [24] introduce to learn the frequency component of the masked patch features. PeCo [13] uses an offline visual perceptual codebook to guide the training.

2.3. Geometry Learning in Point Cloud

In computer graphics, previous works [19, 31, 40, 45, 52, 59] propose various methods for calculation of differential properties of 3D discrete geometry. Curvature and normal are two of these most important properties. Taubin algorithm [40] proposes to estimate the curvature of a surface at each point of a polyhedral approximation. CAN [52] introduces Local Fitting for normal curvatures by employing chord, neighbor normal vector and osculating circle. As for surface normal estimation, Hoppe et al. [19] first suggests to fit a least square plane to k nearest neighbors of each point to estimate its normal. Mitra et al. [31] analyzes the methods of least square with noise added and provides theoretical bound.

In deep learning filed, methods [9, 30, 34, 35, 41, 43, 56] are commonly based on some assumptions of implicit local geometry. Point-based methods [25, 26, 34–36, 41, 43] usually adopt set abstraction to capture local points features in region-wise. Voxel-based methods [9, 22, 55, 56] project the point clouds to 3D voxel grids and encode features of points inside the same voxel by voxelization.
3. Method

3.1. Architecture Overview

We propose a simple yet effective method for self-supervised point cloud representation learning, named GeoMAE. GeoMAE predicts both point statistics and surface geometric properties from point clouds. The overall pipeline is illustrated in Figure 2. First, we voxelize the original point cloud and transform it into voxel patch tokens. We randomly mask out voxel tokens based on predefined ratios for a challenging pre-text task. We define a set of learnable tokens for masked tokens. These visible tokens (corresponding to masked tokens) are fed into a sparse transformer encoder. Conditioned on features of visible tokens, learnable masked tokens are processed by separated decoders to predict both point statistics (centroid and occupancy) and surface properties (normal and curvature). Next, we will elaborate on each step.

Voxel Token Embedding and Masking. We follow recent 3D perception architectures [48] and transform sparse input point clouds into regular voxel grids. Then, these voxels are processed by 3D convolutional neural networks or transformer-based networks. We adopt the widely-used dynamic voxelization [55] to perform voxelization: First, the input scene is divided into equally spaced voxels as shown in Figure 2. Each point \( p_i \) will be assigned to a voxel \( v_j \) where the point resides. Then, we pass non-empty voxels through VFE [56] layers to obtain per-voxel features/tokens \( T_v \). Based on evidences from 2D masked modeling methods [18], we choose a high mask ratio (70%) when removing tokens. Our method predicts target properties per learnable masked token.

Sparse Encoder. After random masking, only visible voxel tokens are fed into an encoder. Due to the sparse and long-range nature of the input scene, we choose a sparse transformer proposed in SST [15] as our encoder. Similar to Swin-Transformer [27], self-attention is only calculated among non-empty voxels within the same region in SST. The output token of the encoder is \( T_e \), together with the learnable masked token \( T_m \) to form the input \( T_d \) of the decoder.

Decoders. We use two separate decoders to decode point statistics and surface properties, respectively. Each decoder consists of two sparse transformer blocks. These two decoders take the same input \( T_d \) and generate two output features \( T_{point} \) and \( T_{surface} \). Empirically, we found such separate design better facilitated models to learn point statistics and surface properties than a single shared decoder did. Finally, we use separate prediction heads with lightweight MLPs to predict each target \( P \in \mathbb{R}^{N \times K} \) based on features produced by previous decoders.

Prediction Targets. The prediction targets include the point statistics and surface properties of a point cloud region. The point statistics contain two objectives: pyramid centroid \( T_{cent} \) and occupancy \( T_{occ} \). There are also two objectives for surface properties: surface normal \( T_{norm} \) and surface curvature \( T_{curv} \). The details of each prediction target will be discussed in Section 3.2 and Section 3.3.

We train our network to learn the point statistics and surface properties of uneven point clouds by supervising those prediction targets:

\[
L_{point} = L_{cent}(P_{cent}, T_{cent}) + L_{occ}(P_{occ}, T_{occ}),
\]
\[
L_{surface} = L_{curv}(P_{curv}, T_{curv}) + L_{nor}(P_{nor}, T_{nor}),
\]

For centroid, curvature and normal prediction, we use MSE loss, and for occupancy prediction we use Cross-Entropy loss. The overall loss function of our framework is defined as:

\[
L = L_{point} + L_{surface}
\]
3.2. Point Statistics Prediction

Different from 2D images and 3D indoor point clouds, outdoor point clouds are sparse and occluded. Point density varies much in a point cloud, which prevents models directly predicting original point coordinates. The pilot study in Figure 1 also shows that such a prediction target is not available. To deal with non-uniform points, we opt to predict centroid of points in each voxel. In addition, to incorporate multi-scale information, we aim to predict these statistics in different scales by building a voxel pyramid. As shown in Figure 3, we break each masked voxel into three sub-voxel levels (top, middle, and bottom) and compute the voxel occupancy and centroid at each level.

**Centroid and Occupancy.** Let \( G_l^i = \{ G_l^i_i \} \) be the set of non-empty grids in the \( l \)-th pyramid level where \( I_{G_l^i} \) is the grid index, and \( N_l \) is the number of non-empty grids. Points that are within the same grid \( G_l^i \) are grouped together into a set \( \mathcal{N}(G_l^i) \) by calculating their belonging grid index \( I_{G_l^i} \) from their spatial coordinates. The point centroid of each grid \( G_l^i \) is then calculated as:

\[
c_{G_l^i} = \frac{1}{|\mathcal{N}(G_l^i)|} \sum_{x_{p_j} \in \mathcal{N}(G_l^i)} x_{p_j}
\]

We also introduce an occupancy prediction target to judge whether a grid is empty or not:

\[
o_{G_l^i} = \begin{cases} 1, & \text{at least one point in the grid} \\ 0, & \text{otherwise} \end{cases}
\]

3.3. Surface Properties Prediction

LiDAR point clouds naturally preserve geometric information. Although point statistics provide a rough estimation of shapes, they cannot describe the fine-grained geometric information that are usually critical to recognition tasks. Therefore, in addition to point statistics prediction, we further leverage 3D shape geometry of point clouds for self-supervised learning. Our desiderata include: these geometric features should be easy to compute and accurately approximate local shape geometry; we can estimate these features from local point groups. Therefore, our choices are surface curvature and surface normals which can be computed in closed-form from local points. To obtain a more stable geometric representation, we incorporate points from 8 neighboring voxels in addition to inside points in each voxel.

**Surface Normal and Curvature.** Inspired by surface feature estimation (i.e., curvature estimation [52] and normal estimation [31]) in geometry processing, we adopt local least square fitting to handle noisy LiDAR point clouds. Given a set of \( K \) gathered points \( p_i (1 \leq i \leq K) \), we compute a covariance matrix

\[
M = \frac{1}{K} \sum_{i=1}^{K} p_i p_i^T - pp^T,
\]
where $M$ is a $3 \times 3$ symmetric matrix, $\bar{p}$ is the centroid of this point cluster. After the eigen-decomposition of $M$, we obtain eigenvalues $\lambda_1, \lambda_2, \lambda_3$ ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) and their corresponding eigenvectors and $n_1, n_2, n_3$. In fact, we use singular value decomposition. Following the aforementioned work [31], the normal vector for each voxel is $n_3$ (the corresponding eigenvector of the least eigenvalue). Moreover, we compute three pseudo curvature vectors $c_m$ for each point:

$$c_m = \frac{\lambda_m}{\sum_{i=1}^{3} \lambda_i}, m \in \{1, 2, 3\}. \tag{7}$$

Therefore, surface properties prediction targets for each masked token $v_j$ can be formalized as:

$$P^{j}_{\text{nor}} = n^j_3, \quad P^{j}_{\text{curv}} = \{c^j_0, c^j_1, c^j_2\} \tag{8}$$

### 4. Experiments

In this section, we evaluate our proposed GeoMAE on two widely used benchmarks: Waymo Open Dataset [39] and nuScenes Dataset [4]. We first elaborate the experiment setting in Section 4.1. In Section 4.2 we compare our method with previous self-supervised point cloud representation learning methods. In Section 4.3 we show the generalization of our method on different downstream tasks. In Section 4.4, we conduct various ablation studies to evaluate the effectiveness of our approach.

#### 4.1. Experimental Setup

**Waymo Open Dataset.** Waymo Open Dataset [39] consists of 798 training sequences and 202 validation sequences. The point cloud scene is collected by a 64-beam LiDAR with around 158k point cloud samples in the training split and 40k point cloud samples in the validation split. For 3D detection, the official evaluation metric includes standard 3D mean Average Precision (mAP) and mAP weighted by heading accuracy (mAPH). These metrics are based on an IoU threshold of 0.7 for vehicles and 0.5 for other categories. These metrics are further broken down into two difficulty levels: L1 for boxes with more than five LiDAR points and L2 for boxes with at least one LiDAR point.

**nuScenes Dataset.** The nuScenes Dataset [4] is a large-scale autonomous driving dataset that contains 700, 150, 150 sequences for training, validation, and testing, respectively. For 3D detection, the major official metrics are mean Average Precision (mAP) and nuScenes detection score (NDS). The mAP uses a bird-eye-view center distance threshold (0.5m, 1m, 2m, 4m) instead of bounding box IoU. NDS is a weighted average of mAP and other attribute metrics, including translation, scale, orientation, velocity, and other box attributes. For 3D tracking, nuScenes mainly uses AMOTA, which penalizes ID switches, false positive, and false negatives and is averaged among various recall thresholds.

**Model.** Our proposed GeoMAE use a standard SST [15] as the encoder, which contains two consecutive blocks. Each attention module in the encoder has two heads, 128 input channels, and 256 hidden channels. We use two parallel decoders, and each decoder has two transformer blocks. Given the masked voxels $V_m$ with the voxel grid size $g_x, g_y, g_z$ along the x, y, and z axes of the 3D space, respectively, we sub-divide each voxel’s spatial space into three pyramid levels (as shown in Figure 3): top, middle, and bottom. The grid in the top level has the same grid size as the original voxel grid. The sub-grids in the middle level have a grid size of $g_x/2, g_y/2, g_z/2$, while the grid size in the bottom level is $g_x/4, g_y/4, g_z/4$. The number of grids in each level, from top to bottom, is 1, 16, and 128, respectively.

**Training Details.** On the Waymo Open Dataset, we use all the samples for pre-training and uniformly sample 20% of frames for finetuning following common practice. On the nuScenes Dataset, we use all the frames (including the unlabeled sweeps) for pre-training and the labeled samples for finetuning. For pre-training, all the self-supervised methods are trained for 72 epochs (denoted as 6x) with the AdamW optimizer [21]. The initial learning rate is 1e-5. For finetuning on downstream tasks, we follow the original training

<table>
<thead>
<tr>
<th>Method</th>
<th>Waymo</th>
<th>nuScenes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L1 AP/APH</td>
<td>L2 AP/APH</td>
</tr>
<tr>
<td>Scratch</td>
<td>70.68/66.39</td>
<td>64.28/60.42</td>
</tr>
<tr>
<td>BYOL [16]</td>
<td>70.15/65.72</td>
<td>63.71/59.94</td>
</tr>
<tr>
<td>PointContrast [46]</td>
<td>71.73/67.28</td>
<td>65.34/61.45</td>
</tr>
<tr>
<td>SwAV [5]</td>
<td>71.85/67.43</td>
<td>65.41/61.63</td>
</tr>
<tr>
<td>STRL [20]</td>
<td>71.91/67.64</td>
<td>65.52/61.77</td>
</tr>
<tr>
<td>GeoMAE (Ours)</td>
<td><strong>73.71/70.24</strong></td>
<td><strong>67.30/63.97</strong></td>
</tr>
</tbody>
</table>

Table 1. Performances of 3D object detection on the Waymo Open Dataset and nuScenes Dataset validation split.
### 3.3. Comparison on other Downstream Tasks

We further evaluate the effectiveness and generalization of GeoMAE in different 3D downstream tasks (including detection, segmentation, and tracking) with different model architectures (backbones and heads). For each model in a downstream task, we evaluate three variants with different 3D self-supervised methods.

#### 3.3.1 3D Object Detection

**Settings.** We comprehensively evaluate GeoMAE on both the anchor-based detector SECOND and a center-based detector CenterPoint [50]. For the Waymo Open Dataset, the detection point cloud range is set to [-74.88m, 74.88m] for X- and Y-axes, [-2m, 4m] for Z-axes, and the voxel size is set to (0.32m, 0.32m, 6m). For nuScenes Dataset, the detection range is set to [-51.2m, 51.2m] for X- and Y-axes, [-5m, 3m] for Z-axes, and the voxel size is set to (0.256m, 0.256m, 8m).

**Results.** As shown in Table 2, both anchor-based and center-based detectors with pre-trained SST by our GeoMAE achieve better performance than the baselines. For anchor-based detector, our GeoMAE outperforms the baseline by 3.03 L1 AP/APH on the Waymo Open Dataset and 3.38 mAP on the nuScenes Dataset. Compared with other self-supervised methods, GeoMAE outperforms the second best method STRL [20] by a significant margin, 1.78/2.20 L1 AP/APH on the Waymo Open Dataset and 2.05 mAP on the nuScenes Dataset, demonstrating the effectiveness of our model and prediction target designs.
3.02 mAP on the nuScenes Dataset. All the results verify the efficacy of our proposed method. In Table 3, we also test our method on the nuScenes test split and achieve a new state-of-the-art result.

### 4.3.2 3D Object Tracking

**Settings.** We also conduct experiments in a 3D multi-object tracking (MOT) task on the nuScenes Dataset by performing tracking-by-detection algorithms proposed by CenterPoint [50] and SimpleTrack [33]. The point cloud range and voxel size are the same as the 3D object detection settings.

**Results.** From Table 6, we can see that our GeoMAE outperforms the baseline (SST) by 1.7 AMOTA for Centerpoint and 1.1 AMOTA for SimpleTrack. These observations are consistent with those in 3D object detection.

![Table 6](image)

Table 4. Performances of 3D multi-object tracking on the nuScenes Dataset validation split.

3.02 mAP on the nuScenes Dataset. All the results verify the efficacy of our proposed method. In Table 3, we also test our method on the nuScenes test split and achieve a new state-of-the-art result.

### 4.3.2 3D Object Tracking

**Settings.** We also conduct experiments in a 3D multi-object tracking (MOT) task on the nuScenes Dataset by performing tracking-by-detection algorithms proposed by CenterPoint [50] and SimpleTrack [33]. The point cloud range and voxel size are the same as the 3D object detection settings.

**Results.** From Table 6, we can see that our GeoMAE outperforms the baseline (SST) by 1.7 AMOTA for Centerpoint and 1.1 AMOTA for SimpleTrack. These observations are consistent with those in 3D object detection.

![Table 6](image)

Table 6. Performances of 3D multi-object tracking on the nuScenes Dataset validation split. *: re-implemented by MMDe-

### 4.3.3 LiDAR Semantic Segmentation

**Settings.** To demonstrate the generalization capability, we evaluate our method on the nuScenes Dataset for the LiDAR segmentation task. We follow the official guidance to leverage mean intersection-over-union (mIoU) as the evaluation metric. We adopt the Cylinder3D [58] as our baseline architecture and replace the last stage of the backbone from sparse convolutions into SST. Other training settings are the same as in Cylinder3D.

**Results.** As reported in Table 4, the Cylinder3D obtains 0.4 performance gain by replacing the backbone from sparse convolutions with our modified SST. When applying our GeoMAE to pre-train the backbone, it achieves 2.1 mIoU gain than training from scratch.

### 4.3.4 BEV Map Segmentation

**Settings.** We further experiment our method in the BEV Map Segmentation task on the nuScenes Dataset. We perform the evaluation in the [-50m, 50m]×[-50m, 50m] region following the common practice in BEVFusion [28]. We develop the CenterPoint-SST and BEVFusion-SST by replacing the last two stages of the LiDAR backbone with SST.

**Results.** We report the BEV map segmentation results in Table 5. For the LiDAR-only model, our method surpass the SST baseline by 2.7 mIoU. In the multi-modality setting, GeoMAE further boosts the performance of BEVFusion-SST about 2.1 mIoU, which demonstrates the strong generalization capability of our method.

### 4.4 Ablation Study

We adopt standard SST [15] the default backbone in our ablation study. To get efficient validation and reduce experimental overhead, all the experiments are pre-trained for 72 (6x) epochs if not specified.

**Prediction Targets.** We present ablation studies in Table 7 to justify our design choices. Scratch means train-
Table 7. Ablation study on prediction targets. Detection results on the Waymo and nuScenes Dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>Waymo L1 AP</th>
<th>nuScenes mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td></td>
<td>70.68</td>
<td>50.39</td>
</tr>
<tr>
<td>+ Centroid</td>
<td>Point</td>
<td>71.60</td>
<td>51.25</td>
</tr>
<tr>
<td>+ Occupancy</td>
<td>Statistics</td>
<td>72.65</td>
<td>52.12</td>
</tr>
<tr>
<td>+ Surface Normal</td>
<td>Surface</td>
<td>73.37</td>
<td>52.94</td>
</tr>
<tr>
<td>+ Surface Curvature</td>
<td>Surface</td>
<td>73.71</td>
<td>53.31</td>
</tr>
</tbody>
</table>

Table 8. Ablation study on decoder design. Detection results on the Waymo and nuScenes Dataset.

<table>
<thead>
<tr>
<th>Decoder Design</th>
<th>Waymo L1 AP</th>
<th>nuScenes mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Decoder</td>
<td>72.45</td>
<td>52.04</td>
</tr>
<tr>
<td>Separate (Only Point Statistics)</td>
<td>72.65</td>
<td>52.28</td>
</tr>
<tr>
<td>Separate (Only Surface Properties)</td>
<td>72.03</td>
<td>51.87</td>
</tr>
<tr>
<td>Separate (Different Targets)</td>
<td><strong>73.71</strong></td>
<td><strong>53.31</strong></td>
</tr>
</tbody>
</table>

Table 9. Ablation on masking ratio. Detection results on the Waymo and nuScenes Dataset.

<table>
<thead>
<tr>
<th>Masking Ratio</th>
<th>Waymo L1 AP</th>
<th>nuScenes mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>72.27</td>
<td>52.47</td>
</tr>
<tr>
<td>60%</td>
<td>73.28</td>
<td>53.00</td>
</tr>
<tr>
<td>70%</td>
<td><strong>73.71</strong></td>
<td><strong>53.31</strong></td>
</tr>
<tr>
<td>80%</td>
<td>71.94</td>
<td>51.85</td>
</tr>
</tbody>
</table>

Table 10. Ablation on training schedule. Detection results on the Waymo and nuScenes Dataset.

<table>
<thead>
<tr>
<th>Training Epochs</th>
<th>Waymo L1 AP</th>
<th>nuScenes mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>72.67</td>
<td>52.64</td>
</tr>
<tr>
<td>48</td>
<td>73.32</td>
<td>52.97</td>
</tr>
<tr>
<td>72</td>
<td><strong>73.71</strong></td>
<td><strong>53.31</strong></td>
</tr>
<tr>
<td>96</td>
<td>73.71</td>
<td>53.30</td>
</tr>
</tbody>
</table>

Table 11. Impacts of different scale of the pre-training dataset. Detection results on the Waymo and nuScenes Dataset.

<table>
<thead>
<tr>
<th>Dataset Scale</th>
<th>Waymo L1 AP</th>
<th>nuScenes mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>70.68</td>
<td>50.39</td>
</tr>
<tr>
<td>20%</td>
<td>72.27</td>
<td>51.64</td>
</tr>
<tr>
<td>50%</td>
<td>72.88</td>
<td>52.74</td>
</tr>
<tr>
<td>80%</td>
<td>73.48</td>
<td>53.03</td>
</tr>
<tr>
<td>100%</td>
<td><strong>73.71</strong></td>
<td><strong>53.31</strong></td>
</tr>
</tbody>
</table>

5. Conclusion

We present GeoMAE, a geometry-aware self-supervised pre-training approach for point clouds. GeoMAE achieves strong performance on a variety of downstream tasks including 3D detection, segmentation, and tracking. GeoMAE leverages recent development in masked modeling. In addition to the commonly used occupancy prediction target, our method proposes three additional learning objectives, which jointly become a challenging and informative pretext task. Our key observation is that geometric features provide strong information for models to reason objects and scenes, therefore improve downstream recognition performance. Our results also suggest several venues for future inquiry. First, pre-training using GeoMAE on larger unlabeled dataset will further boost the performance (e.g., pre-training on DDAD dataset [17]). Besides, exploring other types of geometric features remains an open and intriguing question.

Acknowledgement  This work was supported by National Key R&D Program of China (2022ZD0161700).


Xiaopeng Zhang, Hongjun Li, Zhanglin Cheng, et al. Curve-
ture estimation of 3d point cloud surfaces through the fitting
of normal section curvatures. *Proceedings of ASIAGRAPH*,

Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan
Misra. Self-supervised pretraining of 3d features on any
point-cloud. In *Proceedings of the IEEE/CVF International

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang
Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training
2021.

Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang
Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Va-

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning
for point cloud based 3d object detection. In *Proceedings of
the IEEE conference on computer vision and pattern recog-

Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and
Gang Yu. Class-balanced grouping and sampling for point
2019.

Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin
Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical
and asymmetrical 3d convolution networks for lidar seg-
mentation. In *Proceedings of the IEEE/CVF conference on
computer vision and pattern recognition*, pages 9939–9948,
2021.

Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus
Gross. Pointshop 3d: An interactive system for point-based
surface editing. *ACM Transactions on Graphics (TOG)*,