GD-MAE: Generative Decoder for MAE Pre-training on LiDAR Point Clouds

Honghui Yang[1,3*]   Tong He[3*]   Jiaheng Liu[3,4]   Hua Chen[5]   Boxi Wu[2]

Binbin Lin[2†]   Xiaofei He[1]   Wanli Ouyang[3]

[1]State Key Lab of CAD&CG, Zhejiang University

[2]School of Software Technology, Zhejiang University

[3]Shanghai AI Laboratory   [4]Beihang University

[5]COMAC Beijing Aircraft Technology Research Institute

Abstract

*Despite the tremendous progress of Masked Autoencoders (MAE) in developing vision tasks such as image and video, exploring MAE in large-scale 3D point clouds remains challenging due to the inherent irregularity. In contrast to previous 3D MAE frameworks, which either design a complex decoder to infer masked information from maintained regions or adopt sophisticated masking strategies, we instead propose a much simpler paradigm. The core idea is to apply a **G**enerative **D**ecoder for MAE (GD-MAE) to automatically merges the surrounding context to restore the masked geometric knowledge in a hierarchical fusion manner. In doing so, our approach is free from introducing the heuristic design of decoders and enjoys the flexibility of exploring various masking strategies. The corresponding part costs less than **12%** latency compared with conventional methods, while achieving better performance. We demonstrate the efficacy of the proposed method on several large-scale benchmarks: Waymo, KITTI, and ONCE. Consistent improvement on downstream detection tasks illustrates strong robustness and generalization capability. Not only our method reveals state-of-the-art results, but remarkably, we achieve comparable accuracy even with **20%** of the labeled data on the Waymo dataset. Code will be released.*

1. Introduction

We have witnessed great success in 3D object detection [44, 47, 64, 68, 71, 78], due to the numerous applications in autonomous driving, robotics, and navigation. Despite the impressive performance, most methods count on large amounts of carefully labeled 3D data, which is often

---

*Equal contribution. This work was done when Honghui was an intern at Shanghai Artificial Intelligence Laboratory.

†Corresponding author
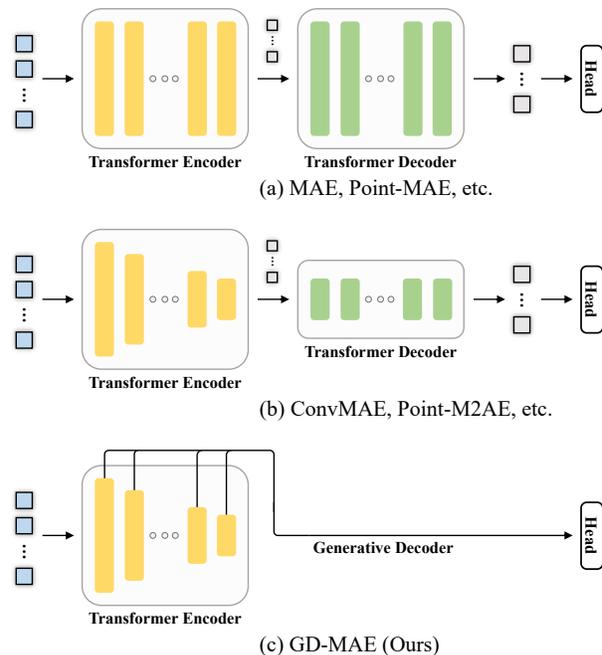
Figure 1. Comparisons. Previous MAE-style pre-training architectures of (a) single-scale [18, 19, 38] and (b) multi-scale [12, 73] take as inputs the visible tokens and learnable tokens for decoders. In contrast, (c) the proposed framework avoids such a process.

of high cost and time-consuming. Such a fully supervised manner hinders the possibility of using massive unlabeled data and can be vulnerable when applied in different scenes. Mask Autoencoder (MAE) [18], serving as one of the effective ways for pre-training, has demonstrated great potential in learning holistic representations. This is achieved by encouraging the method to learn a semantically consistent understanding of the input beyond low-level statistics. Although MAE-based methods have shown effectiveness in 2D image [18] and video [52], how to apply it in large-scale point clouds remains an open problem.

Due to the large variation of the visible extent of objects, learning hierarchical representation is of great signif-

icance in 3D supervised learning [40, 46, 62]. To enable MAE-style pre-training on the hierarchical structure, previous approaches [12, 73] introduce either complex decoders or elaborate masking strategies to learn robust latent representations. For example, ConvMAE [12] adopts a block-wise masking strategy that first obtains a mask for the late stage of the encoder and then progressively upsamples the mask to larger resolutions in early stages to maintain masking consistency. Point-M2AE [73] proposes a hierarchical decoder to gradually incorporate low-level features into learnable tokens for reconstruction. Meanwhile, it needs a multi-scale masking strategy that backtracks unmasked positions to all preceding scales to ensure coherent visible regions and avoid information leakage. The minimum size of masking granularity is highly correlated to output tokens of the last stage, which inevitably poses new challenges, especially to objects with small sizes, e.g., pedestrians.

To alleviate the issue, we present a much simpler paradigm dubbed GD-MAE for pre-training, as shown in Figure 1. The key is to use a generative decoder to automatically expand the visible regions to the underlying masked area. In doing so, it eliminates the need for designing complex decoders, in which masked regions are presented as learnable tokens. It also allows for the unification of multi-scale features into the same scale, thus enabling flexible masking strategies, e.g., point- and patch-wise masking, while avoiding intricate operations such as backtracking in [12, 73] to keep masking consistency. Specifically, it consists of the following components:

Firstly, we propose the Sparse Pyramid Transformer (SPT) as the multi-scale encoder. Following [9, 22, 43], SPT takes pillars as input due to the compact and regular representation. Unlike PointPillars [22] that uses traditional convolutions for feature extraction, we use the sparse convolution [62] to downsample the tokens and the sparse transformer [9] to enlarge the receptive field of the visible tokens when deploying extensive masking.

Secondly, we introduce the Generative Decoder (GD) to simplify MAE-style pre-training on multi-scale backbones. GD consists of a series of transposed convolutions used to upsample multi-scale features and a convolution utilized to expand the visible area, as shown in Figure 2. The expanded features are then directly indexed according to the coordinates of the masked tokens for the geometric reconstruction.

Extensive experiments have been conducted on Waymo Open Dataset [49], KITTI [13], and ONCE [33] to verify the efficacy. On the Waymo dataset, GD-MAE sets new state-of-the-art detection results compared to previously published methods.

Our contributions are summarized as follows:

- We introduce a simpler MAE framework that avoids complex decoders and thus simplifies pre-training.
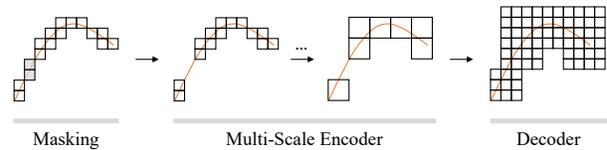
Figure 2. Illustration of area expansion. The input point cloud (i.e., the orange curve) is voxelized and fed into the multi-scale encoder. The generative decoder can automatically expand visible features to potentially masked areas.

- The proposed decoder enables flexible masking strategies on LiDAR point clouds, while costing less than 12% latency compared with conventional methods.

- Extensive experiments are conducted to verify the effectiveness of the proposed model.

2. Related Work

3D Object Detection from Point Clouds. With the release of several large-scale LiDAR datasets, there have been many recent networks proposed for 3D object detection [6, 17, 56, 59, 63, 77, 78]. VoxelNet [80] leverages Point-Net [41] to generate a voxel-wise representation and applies standard convolutions for object detection. SECOND [62] exploits sparse 3D convolutions to accelerate VoxelNet. Point2Seq [61] reformulates the 3D object detection task as decoding words from 3D scenes in an auto-regressive manner. Due to the quantization errors of voxelization, some methods [3, 46–48, 55, 67, 74] directly operate on raw point clouds for detection. 3DSSD [66] extends VoteNet [40] and proposes a hybrid sampling strategy by utilizing both feature and geometry distance for better classification performance. Sampling and grouping points are generally time-consuming. Thus, a number of approaches [32, 35, 64, 69] take advantage of the efficiency of 3D sparse convolutions while preserving accurate point positions. FSD [10] builds a fully sparse 3D object detector to enable efficient long-range detection. Graph R-CNN [64] speeds up the RoI pooling in PointRCNN [46] and introduces a graph-based refinement to achieve better performance.

Transformer in Point Cloud Analysis. Transformer [53] has become a competitive feature learning module in many computer vision tasks [2, 82], which also inspires recent studies [7, 15, 16, 27, 29, 30, 34, 36, 37, 42, 65, 68] for point cloud analysis. Point Transformer [76] employs vector attention to better extract local features. Point Transformer V2 [58] enhances Point Transformer and presents a more powerful and efficient model. Fast Point Transformer [39] proposes a lightweight self-attention layer and a voxel hashing-based architecture to boost computational efficiency. Stratified Transformer [21] enlarges the effective receptive field at a low computational cost by sampling nearby points densely and distant points sparsely in a strat-

ified way. Object DGCNN [57] models 3D object detection as message passing on a dynamic graph and removes the necessity of non-maximum suppression.

Self-supervised Learning for Point Clouds. Point cloud representation learning without labels has been widely studied in recent years [1, 8, 23, 25, 70, 72]. OcCo [54] occludes point clouds based on different viewpoints and learns to complete them. PointContrast [60] contrasts point-level features from two transformed views to learn discriminative 3D representations. DepthContrast [75] learns features by considering voxels and point clouds of the same 3D scene as data augmentations. 4DContrast [5] leverages 4D signals in unsupervised pre-training to imbue 4D object priors into learned 3D representations. Inspired by the promising results achieved by MAE [18] in 2D vision, some works extend it into point clouds. Point-MAE [38] divides the point cloud into irregular point patches and aims to reconstruct the masked patches. MaskPoint [26] represents the point cloud as discrete occupancy values and designs the decoder to discriminate masked real points and sampled fake points. Differently, we explore MAE in the challenging outdoor point clouds, which have not yet been fully investigated.

3. Methodology

In this section, we first review previous works in Sec. 3.1. Then, the designed sparse pyramid transformer and masked autoencoder are elaborated in Sec. 3.2.

3.1. Preliminaries

In contrast to conventional voxel-based detectors, pillar-based methods discretize the input point cloud with a grid of fixed size in the x-y plane, resulting in pillars rather than cubic voxels. This compact representation makes it achieve a good balance of efficiency and accuracy. In this section, we revisit the pillar-based representation and the extension of the sparse transformer on top of it.

Pillar-based Representation. PointPillars [22] is the pioneering pillar-based detector with 2D CNNs. The 3D space is divided into equally distributed pillars which are voxels of infinite height. The points are assigned to pillars to generate a feature vector. Subsequently, the obtained pillar features are scattered back to their corresponding horizontal locations in the scene to form a dense 2D pseudo-image. The pseudo-image is then processed by a feature pyramid network, which extracts multi-scale features using convolution layers with strides of $1\times$, $2\times$, and $4\times$. We refer the readers to [22] for more details.

Sparse Transformer. SST [9] is a transformer-based 3D detector operating on non-empty pillars. Similar to Swin

Transformer [28], SST divides the space into a list of non-overlapping windows with a fixed size. The self-attention is adopted among pillars within the same window. Owing to its single-stride property, SST achieves impressive results for small object detection.

Analysis. Due to the self-occlusion of 3D objects, most of the points are sparsely distributed over the surface of the objects. Spatial disconnection [4] of sparse points can be exacerbated when extensive masking is applied. For the visible points, it will be challenging to use a traditional convolution backbone like PointPillars to contain enough receptive fields. To address this issue, inspired by SST, we introduce a simple yet effective transformer-based pyramid structure to achieve a large spatial scope.

3.2. Masked Autoencoder

Inspired by the success of MAE [18] in 2D images, we develop the masked autoencoder for self-supervised learning on LiDAR point clouds, as shown in Figure 4. The core idea is to use the encoder to create multi-scale representations from partial observations of the input. The decoder is thereafter applied to unify the multi-scale features to a determined scale and expand the visible features to the underlying masked area. Finally, the features of masked parts are processed by a head to reconstruct corresponding input point clouds. After pre-training, the parameters of the encoder are used to warm up the backbone of the detection task. Details are described below.

Multi-Scale Encoder. Unlike previous approaches [19, 26, 38] that use a standard transformer encoder with a constant resolution for feature extraction, we exploit a hierarchical transformer architecture to better capture features from sparse LiDAR point clouds. We present the overview of the Sparse Pyramid Transformer (SPT) in Figure 3.

Similar to PointPillars [22], the input points $\mathcal{P} = \{p_i\}_{i=0}^{N-1}$ are converted to a grid of 2D pillars on bird's eye view by the pillar feature encoding (PFE) module. Specifically, the pillar index of each point p_i is first calculated as $v_i = (\lfloor \frac{x_i}{V_x} \rfloor, \lfloor \frac{y_i}{V_y} \rfloor)$, where x_i and y_i are coordinates of p_i in the x-y plane, and V_x and V_y are the corresponding pillar size. According to the pillar index, each point can be assigned to evenly divided pillar grids. Since multiple points can potentially fall into the same pillar, a stack of Point-Net [41] is used to aggregate features from points to get pillar-wise features. Finally, we take the pillars' features $\mathcal{F} \in \mathbb{R}^{M \times C}$ and the pillars' coordinates $\mathcal{C} \in \mathbb{R}^{M \times 2}$ as non-empty tokens (i.e., tokens involving at least one point).

The tokens are fed into three stages to generate feature maps of different scales. In the first stage, non-empty tokens are taken as input and processed directly by a transformer encoder with a constant resolution, while in the other stages,
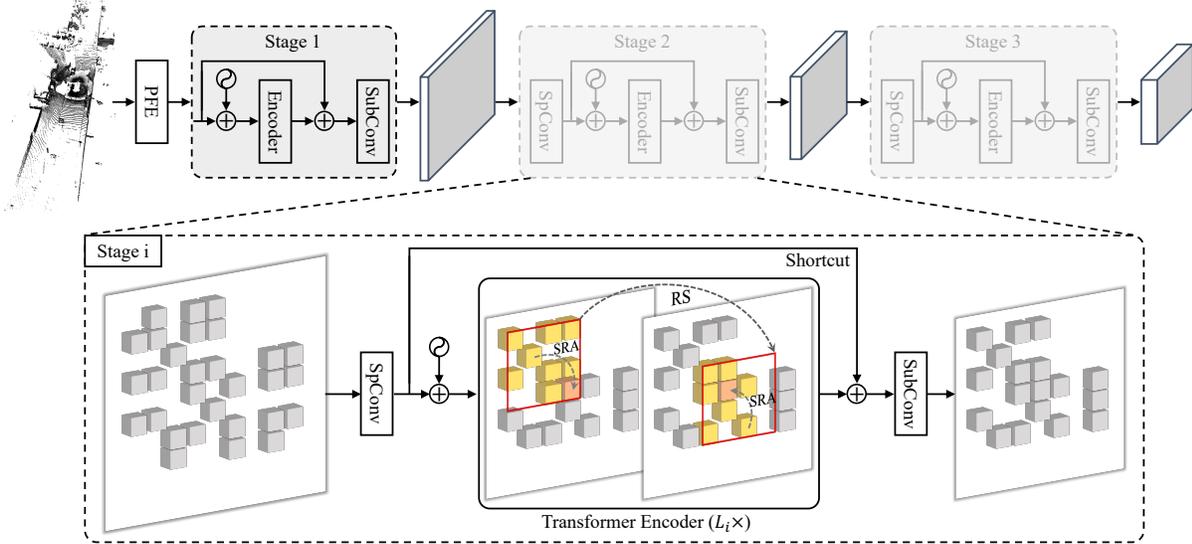
Figure 3. Architecture of sparse pyramid transformer (SPT). The point clouds are fed into pillar feature encoding (PFE) to obtain a grid of pillars. The features and coordinates of the pillars constitute the tokens, which are then processed by three stages. The three stages have a similar structure, except that the second and third stages have a sparse convolution (SpConv) with a stride of 2 for downsampling. In each stage, there are L_i transformer encoders and a shortcut connection, followed by a submanifold convolution (SubConv).

tokens are first downsampled by a sparse convolution (Sp-Conv) with a stride of 2 and then passed through a encoder.

To construct the transformer encoder, we borrow the idea from recent works [9, 28]. In the stage i, it has L_i encoder layers, each of which is composed of two sparse regional attention (SRA) and one region shift (RS). To be specific, two SRA are applied to perform self-attention on the tokens that fall in the same region, accompanied by the positional embedding based on the positions of tokens in each region. Between them, one RS is employed by adding offsets of half of the region size to expand the receptive field of the tokens to capture useful contexts. The entire process used to update tokens' features can be formulated as:

$$\mathcal{F} = \mathrm{SRA}\left(\mathrm{SRA}\left(\mathcal{F}, \mathcal{C}\right), \mathrm{RS}\left(\mathcal{C}\right)\right), \qquad (1)$$

where $\mathcal{F}$ is tokens' features, and $\mathcal{C}$ is tokens' coordinates. After the transformer encoder, the perceptual fields of tokens are broadened, and the long-range contexts are aggregated. However, local features are still important to obtain local geometric details, especially for small objects such as pedestrians. Motivated by this, we add a shortcut to fuse previous features and then use a submanifold convolution (SubConv) to realize adaptive fusion.

Masking. Directly applying the original masking strategy [12, 18, 38, 73] to the last stage of the multi-scale encoder would make the pretext task too difficult, especially for small objects. Because the masking granularity of the output tokens of the last stage is too large, making it hard to recover from unmasked parts. To study the impact, we design three masking strategies with different granularities,

which make the training task from difficult to easy. We use the example of pedestrians to better understand the difficulty level of these three masking strategies.

Block-wise Masking masks a portion of non-empty tokens from downsampled feature maps, e.g., stage 3, and tries to recover them, as shown in Figure 4(a). Different from MAE [18], the multi-scale structure requires the backtracing [12, 73] to make the masked regions consistent across scales to avoid information leakage from previous stages. Thus, inspired by [12, 73], we first upsample the masking map to its original scale, then index the corresponding pixels according to the coordinates of the input tokens to determine whether they are masked or not, and finally feed only the unmasked tokens into the encoder. In the case of a pedestrian, it can be seen that the whole body is masked out and needs to be recovered from the arms or elsewhere, making the task hard.

Patch-wise Masking adopts a smaller masking granularity than block-wise masking by masking some of the upsampled tokens, as shown in Figure 4(b). Since the granularity of the upsampled tokens is the same as that of the input tokens, masking consistency is naturally maintained. It can be regarded as an easier task as only some parts of the pedestrian's body need to be restored.

Point-wise Masking directly masks out a number of the input point clouds and reconstructs the masked points inside tokens, as shown in Figure 4(c). In contrast to the two strategies discussed above, it is trivial to train a point-wise decoder to predict the coordinates of the masked points because the positional encoding would leak the information [26]. Thus, we adopt a patch-wise decoder considering
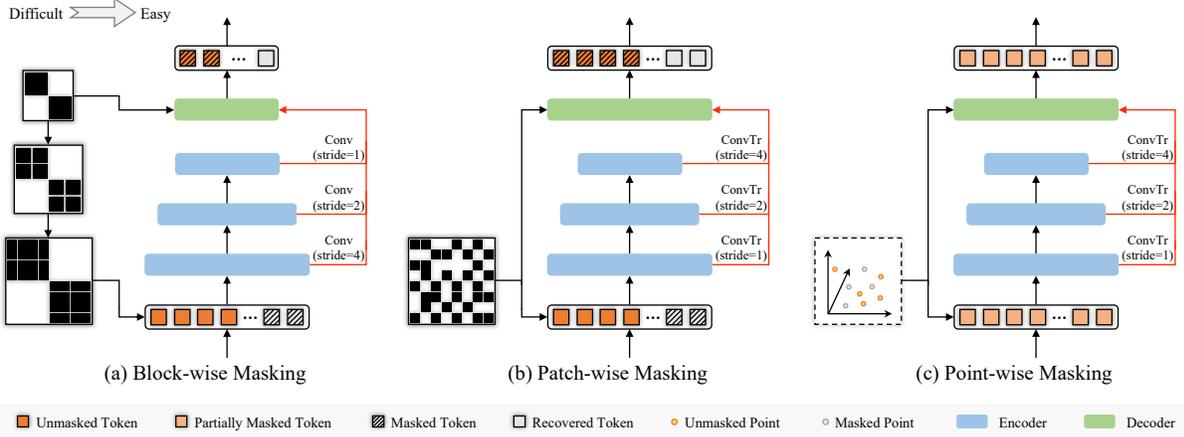
Figure 4. Architecture of the masked autoencoder (MAE). The visible tokens are fed into the multi-scale encoder to create hierarchical features. Then, the generative decoder takes as input the multi-scale feature map and the masking map to unify the multi-scale features to a specific scale and recover the features of masked tokens. Finally, the recovered tokens are used for geometric reconstruction.

a patch as the smallest granularity, and each token needs to reconstruct the masked points inside the token. It results in the simplest pre-training task since the entire structure of the body of a pedestrian is preserved, but some details require to be reconstructed.

Generative Decoder. To enable varying masking granularity and unleash multi-scale representations for downstream tasks, we propose the Generative Decoder (GD) to fuse hierarchical features for reconstruction. GD takes as input the visible tokens E_1, E_2, and E_3 from the multi-scale encoder to capture high-level semantic features and low-level geometric features, where E_j denotes the tokens from stage j. We then unify these tokens to the same scale, which is determined by the granularity of the corresponding masking strategy. Specifically, we first transform the sparse tokens into a dense 2D feature map by scattering back tokens' features according to their corresponding coordinates and then performing a series of standard convolutions:

$$D = \text{Conv}\left([F_1(S(E_1)), F_2(S(E_2)), F_3(S(E_3))]\right), \quad (2)$$

where $[\cdot, \cdot]$ is the concatenation function, F is the convolution or the transposed convolution determined by the scale, S is used to scatter back the sparse features, and Conv is a convolution with a kernel size of 3 for multi-scale feature fusion and area expansion. Finally, we obtain the features E_{mask} by indexing the feature vector on D in terms of the coordinates of masked tokens, as shown in Figure 5(b):

$$E = G\left(D\right), \quad (3)$$

where G is adopted to index features.

To set a baseline with the transformer decoder, we follow MAE [18] and some existing works [12,18,19,38,73] to design the pipeline, as shown in Figure 5(a). For a fair comparison, we also adopt Eq. (2) to utilize multi-scale features
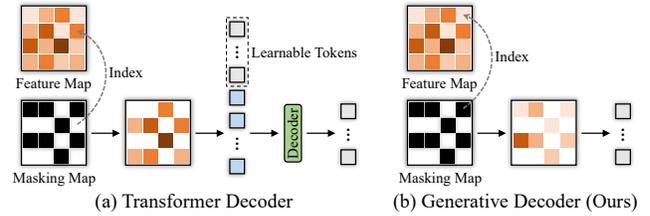
Figure 5. Illustration of (a) the transformer decoder and (b) the proposed generative decoder. We show the invisible areas in black.

for reconstruction, with the difference that the visible area is not expanded by setting the kernel size of the last convolution to 1. Then, we follow Eq. (3) to update the features of visible tokens according to the coordinates by indexing the feature map. The transformer decoder accepts as input visible tokens and masked tokens with shared learnable embeddings. Several flat transformer blocks [9] are then applied to recover the features of masked tokens E_{mask}.

Reconstruction Target. For each masked token, the target is to recover the point cloud that falls within the corresponding token. As different tokens contain a varying number of points, we randomly sample at most K points as the target for reconstruction. To stabilize the training, we normalize the point cloud to obtain $P_{\text{mask}} \in \mathbb{R}^{T \times K \times 3}$ by transforming it into local coordinates relative to the tokens, where T is the number of masked tokens. Given the features of masked tokens $E_{\text{mask}} \in \mathbb{R}^{T \times d}$, we project them using a linear function, followed by a reshape operation:

$$\hat{P}_{\text{mask}} = \text{Reshape}(\text{Linear}(E_{\text{mask}})). \quad (4)$$

Finally, the reconstruction loss is computed by l2 Chamfer Distance, which is formulated as:

$$\mathcal{L}_{\text{CD}} = \text{ChamferDistance}(\hat{P}_{\text{mask}}, P_{\text{mask}}). \quad (5)$$

Table 1. Performance comparisons on the Waymo validation set. 0.2: using 20% labeled data. †: we follow [20, 43, 77, 81] to use an extra IoU prediction head. ‡: the refinement network of [64] is adopted to construct a two-stage detector. The results achieved by our GD-MAE are shown in bold, while top-performed results are shown in underline.

Methods	Voxel Size	mAP/mAPH L2	Vehicle 3D AP/APH L1	L2	Pedestrian 3D AP/APH L1	L2	Cyclist 3D AP/APH L1	L2
Two-stage:								
RSN [50]	-	-/-	75.10/74.60	66.00/65.50	77.80/72.70	68.30/63.70	-/-	-/-
M3DETR [15]	[0.1, 0.1, 0.15]	-/-	75.71/75.08	66.58/66.02	-/-	-/-	-/-	-/-
Voxel RCNN [6]	[0.1, 0.1, 0.15]	-/-	75.59/-	66.59/-	-/-	-/-	-/-	-/-
Pyramid RCNN [32]	[0.1, 0.1, 0.15]	-/-	76.30/75.68	67.23/66.68	-/-	-/-	-/-	-/-
Part-A2-Net [47]	[0.1, 0.1, 0.15]	66.92/63.84	77.05/76.51	68.47/67.97	75.24/66.87	66.18/58.62	68.60/67.36	66.13/64.93
PV-RCNN [44]	[0.1, 0.1, 0.15]	66.80/63.33	77.51/76.89	68.98/68.41	75.01/65.65	66.04/57.61	67.81/66.35	65.39/63.98
PV-RCNN++ [45]	[0.1, 0.1, 0.15]	71.66/69.45	79.25/78.78	70.61/70.18	81.83/76.28	73.17/68.00	73.72/72.66	71.21/70.19
FSD [10]	[0.25, 0.25, 0.2]	72.90/70.80	79.20/78.80	70.50/70.10	82.60/<u>77.30</u>	73.90/69.10	77.10/76.00	74.40/73.30
Graph R-CNN [64]	[0.1, 0.1, 0.15]	73.17/70.87	<u>80.77</u>/<u>80.28</u>	<u>72.55</u>/<u>72.10</u>	82.35/76.64	74.44/69.02	75.28/74.21	72.52/71.49
LiDAR-RCNN [24]	[0.32, 0.32, 6]	64.63/60.10	73.50/73.00	64.70/64.20	71.20/58.70	63.10/51.70	68.60/66.90	66.10/64.40
SST_TS [9]	[0.32, 0.32, 6]	-/-	76.22/75.79	68.04/67.64	81.39/74.05	72.82/65.93	-/-	-/-
GD-MAE‡ (Ours)	**[0.32, 0.32, 6]**	**<u>74.11</u>/<u>71.60</u>**	**80.21/79.78**	**72.37/71.96**	**<u>83.10</u>/76.72**	**<u>75.53</u>/<u>69.43</u>**	**<u>77.22</u>/<u>76.18</u>**	**<u>74.43</u>/<u>73.42</u>**
One-stage:								
IA-SSD [74]	-	62.27/58.08	70.53/69.67	61.55/60.80	69.38/58.47	60.30/50.73	67.67/65.30	64.98/62.71
SECOND [62]	[0.1, 0.1, 0.15]	60.97/57.23	72.27/71.69	63.85/63.33	68.70/58.18	60.72/51.31	60.62/59.28	58.34/57.05
RangeDet [11]	-	64.96/63.20	72.90/72.30	64.00/63.60	75.90/71.90	67.60/63.90	65.70/64.40	63.30/62.10
CenterPoint-Voxel [71]	[0.1, 0.1, 0.15]	68.25/65.81	74.78/74.24	66.66/66.17	75.95/69.75	68.42/62.67	72.27/71.12	69.69/68.59
Point2Seq [61]	[0.1, 0.1, 0.15]	-/-	77.52/77.03	68.80/68.36	-/-	-/-	-/-	-/-
AFDetV2 [20]	[0.1, 0.1, 0.15]	70.96/68.76	77.64/77.14	69.68/69.22	80.19/74.62	72.16/66.95	73.72/72.74	71.06/70.12
CenterFormer [81]	[0.1, 0.1, 0.15]	71.20/68.93	75.20/74.70	70.20/69.70	78.60/73.00	73.60/68.30	72.30/71.30	69.80/68.80
PillarNet-34 [43]	[0.1, 0.1, 6]	70.97/68.43	79.09/78.59	70.92/70.46	80.59/74.01	72.28/66.17	72.29/71.21	69.72/68.67
MVF [79]	[0.32, 0.32, 6]	-/-	62.93/-	-/-	65.33/-	-/-	-/-	-/-
Pillar-OD [56]	[0.32, 0.32, 6]	-/-	69.80/-	-/-	72.51/-	-/-	-/-	-/-
PointPillars [22]	[0.32, 0.32, 6]	62.61/57.57	71.56/70.99	63.05/62.54	70.60/56.69	62.85/50.24	64.35/62.26	61.94/59.93
CenterPoint-Pillar [71]	[0.32, 0.32, 6]	65.98/62.21	73.37/72.86	65.09/64.62	75.35/65.11	67.61/58.25	67.76/66.22	65.25/63.77
SST [9]	[0.32, 0.32, 6]	-/-	74.22/73.77	65.47/65.07	78.71/69.55	70.02/61.67	-/-	-/-
VoxSeT [16]	[0.32, 0.32, 6]	69.13/66.22	74.50/74.03	65.99/65.56	80.03/72.42	72.45/65.39	71.56/70.29	68.95/67.73
GD-MAE_0.2 (Ours)	**[0.32, 0.32, 6]**	**70.24/67.14**	**76.24/75.74**	**67.67/67.22**	**80.50/72.29**	**73.18/65.50**	**72.63/71.42**	**69.87/68.71**
GD-MAE (Ours)	**[0.32, 0.32, 6]**	**70.62/67.64**	**77.26/76.78**	**68.72/68.29**	**80.26/72.36**	**72.84/65.47**	**73.12/71.94**	**70.30/69.16**
GD-MAE† (Ours)	**[0.32, 0.32, 6]**	**72.90/70.43**	**79.40/78.94**	**70.91/70.49**	**82.20/75.85**	**74.82/68.79**	**75.75/74.77**	**72.98/72.03**

Table 2. Performance comparisons on the Waymo leaderboard.

Methods	mAP/mAPH L2	Vehicle 3D AP/APH L1	L2	Pedestrian 3D AP/APH L1	L2
CenterPoint [71]	72.20/69.10	80.20/79.70	72.20/71.80	78.30/72.10	72.20/66.40
PV-RCNN [44]	72.31/69.22	80.60/80.15	72.81/72.39	78.16/72.01	71.81/66.05
PV-RCNN++ [45]	73.99/71.24	81.62/81.20	73.86/73.47	80.41/74.99	74.12/69.00
FSD [10]	75.17/72.66	82.70/82.33	74.40/74.06	82.90/<u>77.88</u>	75.93/71.26
Graph R-CNN [64]	75.82/73.05	83.55/83.12	<u>76.04</u>/<u>75.64</u>	81.91/76.49	75.59/70.45
GD-MAE (Ours)	**<u>76.47</u>/<u>73.37</u>**	**<u>83.56</u>/<u>83.16</u>**	**75.83/75.46**	**<u>83.16</u>/77.05**	**<u>77.10</u>/<u>71.28</u>**

4. Experiments

4.1. Datasets

Waymo Open Dataset [49] is currently the largest dataset with LiDAR point clouds for autonomous driving. There are total 798 training sequences and 202 validation sequences. The evaluation protocol consists of the average precision (AP) and average precision weighted by heading (APH). Also, it includes two difficulty levels: LEVEL_1 denotes objects containing more than 5 points, and LEVEL_2 denotes objects containing at least 1 point. By default, we use a subset of the training splits by sampling every 5 frames from the training sequence for ablation studies.

KITTI [13] includes 7481 LiDAR frames for training and 7518 LiDAR frames for testing. As a common practice, the training data are divided into a *train* set with 3712 samples and a *val* set with 3769 samples.

ONCE [33] contains one million point clouds in total, in which 5k, 3k, and 8k point clouds are labeled as the training, validation, and testing split, respectively. The remaining point clouds are kept unannotated, which are adopted by us for the pre-training. The official evaluation metric is mean Average Precision (mAP), and the detection results are divided into 0-30m, 30-50m, and 50m-Inf.

4.2. Implementation Details

Our implementation is based on the codebase of Open-PCDet[1]. For the Waymo dataset, the detection ranges are set as $(-74.88, 74.88)$, $(-74.88, 74.88)$, and $(-2, 4)$, and the voxel size is $(0.32m, 0.32m, 6m)$. For the KITTI dataset, the detection ranges are $(0, 69.12)$, $(-39.68, 39.68)$, and $(-3, 1)$, with a voxel size of $(0.32m, 0.32m, 4m)$. For the ONCE dataset, the detection ranges are $(-74.88, 74.88)$, $(-74.88, 74.88)$, and $(-5, 3)$, and the voxel size is set to $(0.32m, 0.32m, 8m)$. The pillar feature encoding module has two layers of MLPs with channel size of $[64, 128]$. The 3D backbone consists of three stages, each of which has two transformer encoders with the input dimensions of

---

[1] https://github.com/open-mmlab/OpenPCDet

Table 3. Performance comparisons on the ONCE validation split. †: reproduced by us.

Methods	Pre-trained	mAP	Vehicle				Pedestrian				Cyclist			
			Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
PV-RCNN [44]	✗	53.55	77.77	89.39	72.55	58.64	23.50	25.61	22.84	17.27	59.37	71.66	52.58	36.17
IA-SSD [74]	✗	57.43	70.30	83.01	62.84	47.01	39.82	47.45	32.75	18.99	62.17	73.78	56.31	39.53
CenterPoint-Pillar† [71]	✗	59.07	74.10	85.23	69.22	53.14	40.94	48.43	34.72	20.09	62.17	73.70	56.05	40.19
CenterPoint-Voxel [71]	✗	60.05	66.79	80.10	59.55	43.39	49.90	56.24	42.61	26.27	63.45	74.28	57.94	41.48
SECOND [62]	✗	51.89	71.19	84.04	63.02	47.25	26.44	29.33	24.05	18.05	58.04	69.96	52.43	34.61
w/ BYOL [14]	✓	51.63$^{\downarrow 0.26}$	71.32	83.59	64.89	50.27	25.02	27.06	22.96	17.04	58.56	70.18	52.74	36.32
w/ PointContrast [60]	✓	53.59$^{\uparrow 1.70}$	71.87	86.93	62.85	48.65	28.03	33.07	25.91	14.44	60.88	71.12	55.77	36.78
w/ DeepCluster [51]	✓	53.72$^{\uparrow 1.83}$	72.89	83.52	67.09	50.38	30.32	34.76	26.43	18.33	57.94	69.18	52.42	34.36
SPT (Ours)	✗	**62.62**	75.64	87.21	70.10	53.21	45.92	54.78	37.84	22.56	66.30	78.12	60.52	42.05
w/ GD-MAE (Ours)	✓	**64.92**$^{\uparrow 2.30}$	76.79	88.01	71.70	55.60	48.84	58.70	37.30	25.72	69.14	80.29	64.58	45.14

Table 4. Ablation study on the Waymo validation set.

Pyramid	Shortcut	GD-MAE	Vehicle	Pedestrian	Cyclist
			62.10/61.61	69.80/61.30	66.68/65.24
✓			64.20/63.70	71.01/62.97	68.00/66.77
✓	✓		66.02/65.55	71.82/63.76	67.91/66.75
✓	✓	✓	**67.00/66.54**	**72.51/64.93**	**68.94/67.75**

Table 5. Ablation study of the number of transformer encoders. Using 5% data for training.

# Layer	SST [9]				Ours
	4	6	8	10	6
Vehicle	57.25	56.56	56.71	56.49	**60.33**
Pedestrian	55.95	54.18	53.58	53.96	**57.28**

Table 6. Ablation study of different scales of labeled data. Using 100% data for pre-training.

w/ GD-MAE	5%	10%	20%	100%
	59.97	63.58	65.35	67.34
✓	**63.86**$^{\uparrow 3.89}$	65.62$^{\uparrow 2.04}$	67.14$^{\uparrow 1.79}$	67.64$^{\uparrow 0.30}$

Table 7. Ablation study of masked autoencoder on the KITTI *val* set with moderate AP calculated by 40 recall positions.

w/ GD-MAE	Car	Pedestrian	Cyclist
	81.46	46.52	65.59
✓	82.01$^{\uparrow 0.55}$	48.40$^{\uparrow 1.88}$	67.16$^{\uparrow 1.57}$

[128, 256, 256]. All of the transformer encoder layers have 8 heads and inner MLP ratio of 2.

During pre-training, we adopt several popular 3D data augmentation techniques: random flipping, scaling, and rotation. For the masking, the mask ratio is set to 0.75, and the K is 64 for point reconstruction. The generative decoder first converts the dimensionality of the output feature map of the three stages to 128 using three transposed convolutions and then transforms the concatenated features to 128 using a convolution. The model is trained for 30 epochs with the AdamW [31] optimizer using the one-cycle policy, with a max learning rate of $3e^{-3}$. During fine-tuning, in addition to the mentioned data augmentation, the copy-n-paste augmentation [62] is added to increase the number of training samples. Similar to PointPillars, the multi-scale features are upsampled with transposed convolutions and then concatenated for the detection head. We adopt a center-based head, and the training strategy and the target assignment strategy are the same as CenterPoint [71] in OpenPCDet.

4.3. Comparison with State-of-the-Art Methods

We compare GD-MAE with other models on the Waymo dataset with a single frame LiDAR input. As shown in Table 1, GD-MAE achieves new state-of-the-art results among all single-stage detectors on the Waymo validation set: it has 1.5 mAPH/L2 higher than the prior best single-stage model CenterFormer [81]. Compared with the baseline, i.e., SST [9], GD-MAE improves the 3D APH at level 2 for vehicle and pedestrian by 5.42 and 7.12, respectively. Compared with one-stage methods under the same settings

of voxel size, GD-MAE outperforms them by a large margin (+4.93 APH/L2 for vehicle, +3.4 APH/L2 for pedestrian, and +4.3 APH/L2 for cyclist). GD-MAE even outperforms PV-RCNN++ [45] by 0.98 mAPH/L2. Equipped with a refinement network [64], GD-MAE surpasses all previously published methods. Table 2 shows that GD-MAE also achieves impressive results on the Waymo leaderboard.

We evaluate the performance of the proposed GD-MAE on the ONCE validation split in Table 3. GD-MAE achieves significantly better detection results than previous strong detectors. For example, the overall mAP of our approach is 64.92, which is 5.85 and 4.87 higher than the CenterPoint-Pillar and CenterPoint-Voxel, respectively.

4.4. Ablation Study

In this section, we conduct a series of ablation experiments to comprehend the roles of different components.

Sparse Pyramid Transformer. The first and second rows of Table 4 demonstrate that the hierarchical architecture can consistently improve the detection accuracy of each category. The impact on vehicle detection is greater than on other categories because vehicles are generally larger in size and therefore require a broader receptive field to obtain sufficient contextual information for 3D detection, such as size recognition. The second and third rows of Table 4 show that a simple shortcut can further improve performance, suggesting that local geometric features are also important. To further illustrate the necessity of the multi-scale structure, we scale up SST [9] by deepening the network depth in Table 5. We found that accuracy decreases with increasing

Table 8. Ablation study of different multi-scale encoders.

w/ GD-MAE	CNN	SpCNN	Transformer
	61.14	63.83	65.35
✓	**61.48**$^{\uparrow 0.34}$	**64.56**$^{\uparrow 0.73}$	**66.40**$^{\uparrow 1.05}$

Table 9. Ablation study of different masking granularities. Using 5% labeled data for fine-tuning.

Case	Block-wise [12,73]	Ours	
		Patch-wise	Point-wise
Vehicle	61.60	**62.65**	62.41
Pedestrian	59.25	**61.44**	60.60

depth, possibly due to overfitting.

Masked Autoencoder. The third and fourth rows of Table 4 show the effectiveness of the MAE, in which 20% of the data is used for pre-training and fine-tuning. It provides improvements of 0.99, 1.17, and 1.00 APH/L2 for vehicle, pedestrian, and cyclist, respectively. The third column of Table 6 shows that the performance can be further improved by 0.74 mAPH/L2 by using 100% of the unlabeled data for pre-training. To demonstrate the impact of pre-training on the data efficiency, Table 6 shows the performance with different proportions of annotated data. Without pre-training, the performance is increased by 1.99 mAPH/L2 when labeled data is added from 20% to 100%. The gap is reduced to 0.2 mAPH/L2 when the pre-training is applied. With only 5% of annotated data available, MAE can significantly improve accuracy by 3.89 mAPH. The impact of MAE is also verified on the ONCE and KITTI datasets in Tables 3 and 7, respectively, where consistent boosts can be observed.

Multi-Scale Encoders. We also verify the effect of GD-MAE on different multi-scale encoders such as convolution (CNN) and sparse convolution (SpCNN). Table 8 shows that GD-MAE consistently improves the accuracy of different multi-scale encoders. And we found that GD-MAE brings the highest gain for the proposed transformer encoder, which suggests that our simple decoder is particularly suitable for the transformer-based encoder.

Masking Granularity. We study the effect of different masking granularities on LiDAR point clouds. In Figure 4, for a fair comparison, all three masking strategies fuse hierarchical features to capture both high-level semantics and fine-grained patterns. The only difference is the different masking granularity, which leads to different levels of difficulty in the pretext task. From the last column of Table 5 and the first column of Table 9, we can see that pre-training

Table 10. Ablation study of the runtime of the decoder.

Case	Baseline	Ours
Runtime	27.1ms	**3.2ms**

Table 11. Ablation study of different decoders.

# Decoder	Baseline			Ours	
	1	2	3	1	2
Vehicle	66.24	66.20	66.23	**66.54**	66.40
Pedestrian	64.35	64.81	64.30	**64.93**	64.67

with block-wise masking can bring gains of 1.27APH/L2 and 1.97APH/L2 for the vehicle and pedestrian, respectively. By using the finer masking granularity to ease the task, the performance is consistently improved, especially for pedestrians, as shown in the second and third columns of Table 9. We can find that pre-training with the block-wise masking strategy brings less improvement than other methods, probably because the task is too difficult to learn effective features from sparse LiDAR point clouds. And the overly simple task is also suboptimal for efficient feature learning, while an appropriate difficulty level, i.e., patch-wise masking, yields the best result.

Generative Decoder. In Table 11, we study the effect of different decoders. The first three columns show the impact of different numbers of transformer decoders. For pedestrians, the performance is better with two decoders. If the number of decoders is increased or decreased, the results are adversely affected, indicating that performance is sensitive to the number of decoders. And in our case, i.e., the generative decoder, we achieve the best results for both pedestrians and vehicles with only a convolution. This shows that our framework could not only simplify MAE-style pre-training but also bring performance gains. Table 10 demonstrates that the proposed decoder reduces runtime to about $0.12\times$ compared to the transformer decoder.

5. Conclusion

In this paper, we present a much simpler paradigm dubbed GD-MAE for LiDAR point cloud pre-training in the MAE fashion. We first propose the Sparse Pyramid Transformer as the multi-scale encoder to increase the spatial scope of the visible tokens. Then, we introduce a Generative Decoder to simplify pre-training on hierarchical structures and enable fine-grained masking strategies for better feature learning. Extensive experiments are conducted on Waymo, KITTI, and ONCE to verify the efficacy.

References

[1] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, 2020. 2

[3] Chen Chen, Zhe Chen, Jing Zhang, and Dacheng Tao. SASA: semantics-augmented set abstraction for point-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 2

[4] Yukang Chen, Jianhui Liu, Xiaojuan Qi, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Scaling up kernels in 3d cnns. *CoRR*, abs/2206.10555, 2022. 3

[5] Yujin Chen, Matthias Nießner, and Angela Dai. 4dcontrast: Contrastive learning with dynamic correspondences for 3d scene understanding. In *Proceedings of the European Conference on Computer Vision*, 2022. 3

[6] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2, 6

[7] Shengheng Deng, Zhihao Liang, Lin Sun, and Kui Jia. Vista: Boosting 3d object detection via dual cross-view spatial attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[8] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pinar Topçam, Maximilian Listl, Yilmaz Kaan Çayli, and Alois C. Knoll. 3d object detection with a self-supervised lidar scene flow backbone. In *Proceedings of the European Conference on Computer Vision*, 2022. 3

[9] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2, 3, 4, 5, 6, 7

[10] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fully sparse 3d object detection. In *Advances in Neural Information Processing Systems*, 2022. 2, 6

[11] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 6

[12] Peng Gao, Teli Ma, Hongsheng Li, Ziyi Lin, Jifeng Dai, and Yu Qiao. Convmae: Masked convolution meets masked autoencoders. *CoRR*, abs/2205.03892, 2022. 1, 2, 4, 5, 8

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 2, 6

[14] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020. 7

[15] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3DETR: multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Winter Conference on Applications of Computer Vision*, 2022. 2, 6

[16] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2, 6

[17] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 1, 3, 4, 5

[19] Georg Hess, Johan Jaxing, Elias Svensson, David Hagerman, Christoffer Petersson, and Lennart Svensson. Masked autoencoders for self-supervised learning on automotive point clouds. *CoRR*, abs/2207.00531, 2022. 1, 3, 5

[20] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 6

[21] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[22] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 3, 6

[23] Lanxiao Li and Michael Heizmann. A closer look at invariances in self-supervised pre-training for 3d vision. In *Proceedings of the European Conference on Computer Vision*, 2022. 3

[24] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 6

[25] Hanxue Liang, Chenhan Jiang, Dapeng Feng, Xin Chen, Hang Xu, Xiaodan Liang, Wei Zhang, Zhenguo Li, and Luc Van Gool. Exploring geometry-aware contrast and clustering harmonization for self-supervised 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3

[26] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *Proceedings of the European Conference on Computer Vision*, 2022. 3, 4

[27] Jiaheng Liu, Tong He, Honghui Yang, Rui Su, Jiayi Tian, Junran Wu, Hongcheng Guo, Ke Xu, and Wanli Ouyang. 3d-queryis: A query-based framework for 3d instance segmentation. *CoRR*, abs/2211.09375, 2022. 2

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3, 4

[29] Zili Liu, Guodong Xu, Honghui Yang, Minghao Chen, Kuoliang Wu, Zheng Yang, Haifeng Liu, and Deng Cai. Suppress-and-refine framework for end-to-end 3d object detection. *CoRR*, abs/2103.10042, 2021. 2

[30] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 7

[32] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2, 6

[33] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Chunjing Xu, and Hang Xu. One million scenes for autonomous driving: ONCE dataset. In *Advances in Neural Information Processing Systems*, 2021. 2, 6

[34] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[35] Zhenwei Miao, Jikai Chen, Hongyu Pan, Ruiwen Zhang, Kaixuan Liu, Peihan Hao, Jun Zhu, Yang Wang, and Xin Zhan. Pvgnet: A bottom-up one-stage 3d object detector with integrated multi-level features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[36] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[37] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[38] Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *Proceedings of the European Conference on Computer Vision*, 2022. 1, 3, 4, 5

[39] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[40] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[41] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3

[42] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[43] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *Proceedings of the European Conference on Computer Vision*, 2022. 2, 6

[44] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 6, 7

[45] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN++: point-voxel feature set abstraction with local vector representation for 3d object detection. *CoRR*, abs/2102.00463, 2021. 6, 7

[46] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[47] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1, 2, 6

[48] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[49] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 6

[50] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. RSN: range sparse net for efficient, accurate lidar 3d object detection. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, 2021. 6

[51] Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In Michelangelo Ceci, Jaakko Hollmén, Ljupco Todorovski, Celine Vens, and Saso Dzeroski, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference*, 2017. 7

[52] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, 2022. 1

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 2

[54] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3

[55] Haiyang Wang, Shaoshuai Shi, Ze Yang, Rongyao Fang, Qi Qian, Hongsheng Li, Bernt Schiele, and Liwei Wang. Rbgnet: Ray-based grouping for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[56] Yue Wang, Alireza Fathi, Abhijit Kundu, David A. Ross, Caroline Pantofaru, Thomas A. Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 6

[57] Yue Wang and Justin Solomon. Object DGCNN: 3d object detection using dynamic graphs. In *Advances in Neural Information Processing Systems*, 2021. 3

[58] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer V2: grouped vector attention and partition-based pooling. In *Advances in Neural Information Processing Systems*, 2022. 2

[59] Xiaopei Wu, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu, and Deng Cai. Sparse fuse dense: Towards high quality 3d detection with depth completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[60] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Proceedings of the European Conference on Computer Vision*, 2020. 3, 7

[61] Yujing Xue, Jiageng Mao, Minzhe Niu, Hang Xu, Michael Bi Mi, Wei Zhang, Xiaogang Wang, and Xinchao Wang. Point2seq: Detecting 3d objects as sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2, 6

[62] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. 2, 6, 7

[63] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceed-*

[64] Honghui Yang, Zili Liu, Xiaopei Wu, Wenxiao Wang, Wei Qian, Xiaofei He, and Deng Cai. Graph R-CNN: towards accurate 3d object detection with semantic-decorated local graph. In *Proceedings of the European Conference on Computer Vision*, 2022. 1, 2, 6, 7

[65] Zetong Yang, Li Jiang, Yanan Sun, Bernt Schiele, and Jiaya Jia. A unified query-based paradigm for point cloud understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[66] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[67] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[68] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3d-man: 3d multi-frame attention network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2

[69] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[70] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposal-contrast: Unsupervised pre-training for lidar-based 3d object detection. In *Proceedings of the European Conference on Computer Vision*, 2022. 3

[71] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 1, 6, 7

[72] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3

[73] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In *Advances in Neural Information Processing Systems*, 2022. 1, 2, 4, 5, 8

[74] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2, 6, 7

[75] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3

[76] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[77] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2, 6

[78] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2

[79] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, 2019. 6

[80] 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 Recognition*, 2018. 2

[81] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *Proceedings of the European Conference on Computer Vision*, 2022. 6, 7

[82] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. 2