MV-JAR: Masked Voxel Jigsaw and Reconstruction for LiDAR-Based Self-Supervised Pre-Training

Runsen Xu[1,2] Tai Wang[1,2] Wenwei Zhang[3,2] Runjian Chen[4] Jinkun Cao[5]
Jiangmiao Pang[2]✉ Dahua Lin[1,2]
[1]The Chinese University of Hong Kong [2]Shanghai AI Laboratory [3]S-Lab, NTU
[4]The University of Hong Kong [5]Carnegie Mellon University
{runsenxu,wt019,dhlin}@ie.cuhk.edu.hk, wenwei001@ntu.edu.sg, rjchen@connect.hku.hk,
jinkunc@andrew.cmu.edu, pangjiangmiao@gmail.com

Abstract

This paper introduces the Masked Voxel Jigsaw and Reconstruction (MV-JAR) method for LiDAR-based self-supervised pre-training and a carefully designed data-efficient 3D object detection benchmark on the Waymo dataset. Inspired by the scene-voxel-point hierarchy in downstream 3D object detectors, we design masking and reconstruction strategies accounting for voxel distributions in the scene and local point distributions within the voxel. We employ a Reversed-Furthest-Voxel-Sampling strategy to address the uneven distribution of LiDAR points and propose MV-JAR, which combines two techniques for modeling the aforementioned distributions, resulting in superior performance. Our experiments reveal limitations in previous data-efficient experiments, which uniformly sample fine-tuning splits with varying data proportions from each LiDAR sequence, leading to similar data diversity across splits. To address this, we propose a new benchmark that samples scene sequences for diverse fine-tuning splits, ensuring adequate model convergence and providing a more accurate evaluation of pre-training methods. Experiments on our Waymo benchmark and the KITTI dataset demonstrate that MV-JAR consistently and significantly improves 3D detection performance across various data scales, achieving up to a 6.3% increase in mAPH compared to training from scratch. Codes and the benchmark are available at https://github.com/SmartBot-PJLab/MV-JAR.

1. Introduction

Self-supervised pre-training has gained considerable attention, owing to its exceptional performance in visual representation learning. Recent advancements in contrastive learning [4, 6, 8, 16, 45] and masked autoencoders [1, 7, 15, 36, 47] for images have sparked interest among researchers and facilitated progress in modalities such as point clouds.

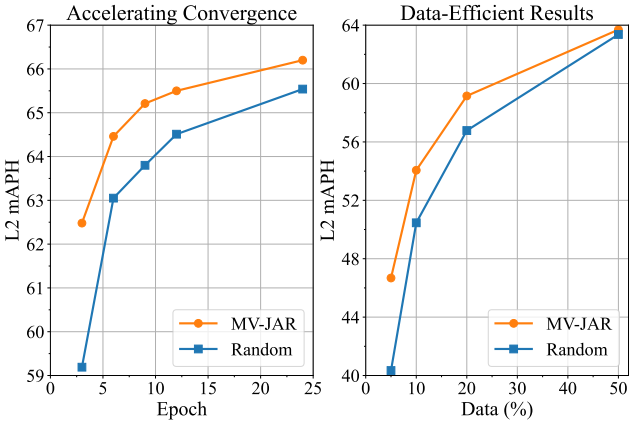


Figure 1. 3D object detection results on the Waymo dataset. Our MV-JAR pre-training accelerates model convergence and greatly improves the performance with limited fine-tuning data.

However, LiDAR point clouds differ from images and dense point clouds obtained by reconstruction as they are naturally sparse, unorganized, and irregularly distributed. Developing effective self-supervised proxy tasks for these unique properties remains an open challenge. Constructing matching pairs for contrastive learning in geometry-dominant scenes is more difficult [20, 40], as points or regions with similar geometry may be assigned as negative samples, leading to ambiguity during training. To address this, our study explores masked voxel modeling paradigms for effective LiDAR-based self-supervised pre-training.

Downstream LiDAR-based 3D object detectors [12, 19, 33, 38, 41, 50] typically quantize the 3D space into voxels and encode point features within them. Unlike pixels, which are represented by RGB values, the 3D space presents a scene-voxel-point hierarchy, introducing new

✉Corresponding author.

challenges for masked modeling. Inspired by this, we design masking and reconstruction strategies that consider voxel distributions in the scene and local point distributions in the voxel. Our proposed method, Masked Voxel Jigsaw And Reconstruction (MV-JAR), harnesses the strengths of both voxel and point distributions to improve performance.

To account for the uneven distribution of LiDAR points, we first employ a Reversed-Furthest-Voxel-Sampling (R-FVS) strategy that samples voxels to mask based on their sparseness. This approach prevents masking the furthest distributed voxels, thereby avoiding information loss in regions with sparse points. To model voxel distributions, we propose Masked Voxel Jigsaw (MVJ), which masks the voxel coordinates while preserving the local shape of each voxel, enabling scene reconstruction akin to solving a jigsaw puzzle. For modeling local point distributions, we introduce Masked Voxel Reconstruction (MVR), which masks all coordinates of points within the voxel but retains one point as a hint for reconstruction. Combining these two methods enhances masked voxel modeling.

Our experiments indicate that existing data-efficient experiments [20, 40] inadequately evaluate the effectiveness of various pre-training methods. The current benchmarks, which uniformly sample frames from each data sequence to create diverse fine-tuning splits, exhibit similar data diversity due to the proximity of neighboring frames in a sequence [3, 14, 30]. Moreover, these experiments train models for the same number of epochs across different fine-tuning splits, potentially leading to incomplete convergence. As a result, the benefits of pre-trained representations become indistinguishable across splits once the object detector is sufficiently trained on the fine-tuning data. To address these shortcomings, we propose sampling scene sequences to form diverse fine-tuning splits and establish a new data-efficient 3D object detection benchmark on the Waymo [30] dataset, ensuring sufficient model convergence for a more accurate evaluation.

We employ the Transformer-based SST [12] as our detector and pre-train its backbone for downstream detection tasks. Comprehensive experiments on the Waymo and KITTI [14] datasets demonstrate that our pre-training method significantly enhances the model's performance and convergence speed in downstream tasks. Notably, it improves detection performance by 6.3% mAPH when using only 5% of scenes for fine-tuning and reduces training time by half when utilizing the entire dataset (Fig. 1). With the representation pre-trained by MV-JAR, the 3D object detectors pre-trained on Waymo also exhibit generalizability when transferred to KITTI.

2. Related Work

Self-supervised learning for point clouds. Annotating point clouds demands significant effort, necessitating self-supervised pre-training methods. Prior approaches primarily focus on object CAD models [21, 26, 29, 39, 42, 44] and indoor scenes [17, 35, 46]. Point-BERT [42] applies BERT-like paradigms for point cloud recognition, while Point-MAE [26] reconstructs point patches without the tokenizer. To acquire scene-level representation, PointContrast [35] introduces a contrastive learning approach that compares points in two static partial views of a reconstructed indoor scene. Its successor, SceneContrast [17], incorporates spatial information into the contrastive learning framework. However, LiDAR point clouds are irregular and dynamic, with mainstream LiDAR perception models often exhibiting distinct architectures, which obstructs the direct adaptation of object-level and indoor scene pre-training methods. STRL [18] employs contrastive learning with two temporally-correlated frames for spatiotemporal representation. GCC-3D [20] presents a framework that combines geometry-aware contrast and pseudo-instance clustering harmonization. ProposalContrast [40] targets region-level contrastive learning to improve 3D detection. CO3 [5] leverages infrastructure-vehicle-cooperation point clouds to construct effective contrastive views. In contrast, our work explores masked voxel modeling, diverging from the prevailing contrastive learning paradigm.

Masked autoencoders for self-supervised pre-training. Masked language modeling plays a pivotal role in self-supervised pre-training for Transformer-based networks in natural language processing [2, 10, 28]. This approach typically masks portions of the input and pre-trains networks to predict the original information. With the successful integration of Transformers into computer vision [11], researchers have increasingly focused on masked image modeling [1, 7, 15, 36, 43, 47] to mitigate the data-intensive issue of ViT [11]. BEiT [1] employs a tokenizer to generate discrete tokens for image patches and utilizes a BERT-like framework to pre-train ViT. MAE [15] introduces an asymmetric autoencoder for reconstructing RGB pixels of original images, eliminating the need for an extra tokenizer. SimMiM [36] encodes masked raw images with Transformers and employs a lightweight prediction head for recovery. Demonstrating considerable potential in image recognition, masked autoencoders have been applied in video understanding [13, 32], medical image analysis [48], and audio processing [37]. In this work, we investigate the application of this powerful pre-training technique to LiDAR point clouds using voxel representation.

Transformer-based 3D object detection. The recent success of Vision Transformers [11] has inspired extensive research into the application of Transformer-based architectures for 3D object detection [12, 23–25, 31]. 3DETR [24] presents an end-to-end object detection framework with modified Transformer blocks for 3D point clouds. Pointformer [25] employs hierarchical Transformer blocks to

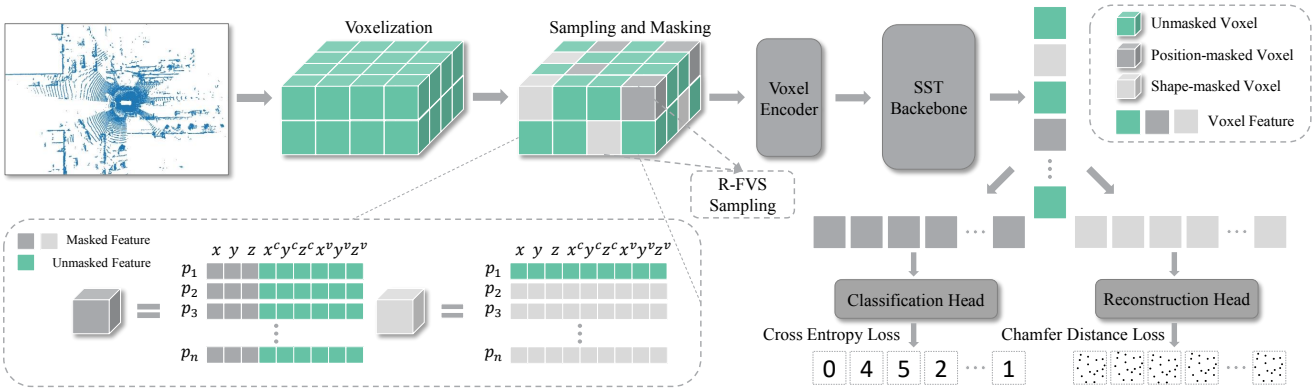


Figure 2. Overview of Masked Voxel Jigsaw and Reconstruction (MV-JAR). Our R-FVS sampling method selects a specific ratio of non-empty voxels to mask. For position-masked voxels, we mask the absolute coordinates of all points within the voxel while maintaining the relative coordinates. For shape-masked voxels, we mask both absolute and relative coordinates of all points, preserving only one point. Features are extracted from all non-empty voxels, and masked features are decoded by a dedicated head to recover corresponding targets.

extract point features for 3D detection. Voxel Transformer [23] develops a Transformer-based 3D backbone to establish long-range relationships between voxels. Recently, SST [12] introduced a single-stride sparse Transformer as a replacement for the PointPillars [19] backbone, abandoning the multi-resolution strategy to enhance the detection of small objects. Emulating the Swin-Transformer [22], SST partitions the space into windows and calculates attention only within each window for efficiency. This approach demonstrates impressive detection results, and we adopt it for our experiments.

3. Methodology

In this section, we first present an overview of a general masked voxel modeling framework for LiDAR point clouds and our Reversed-Furthest-Voxel-Sampling strategy to tackle the uneven distribution of LiDAR points (Sec. 3.1). Masked Voxel Jigsaw (MVJ) (Sec. 3.2) and Masked Voxel Reconstruction (MVR) (Sec. 3.3) are both instantiations of the general framework that learns the voxel distributions and local point distributions, respectively. Masked Voxel Jigsaw and Reconstruction (MV-JAR) consist of these two tasks to synergically learn the data distributions at two levels (Sec. 3.4).

3.1. Masked Voxel Modeling and Sampling

As illustrated in Fig. 2, a general masked voxel modeling framework converts LiDAR point clouds into voxels and samples a proportion of non-empty voxels using a specific sampling strategy and masking ratio. Various features within the voxel can be masked for a particular learning target in this general framework to instantiate different self-supervised learning tasks. Specifically, task-specific features of the points in the sampled voxels are replaced with a learnable vector. Subsequently, all non-empty voxels are encoded by a voxel encoder, processed by a backbone network to extract features, and followed by a lightweight decoder that reconstructs task-specific targets from each voxel feature. Each stage is described in the following sections. In this paper, we employ SST [12], which demonstrates superior performance in 3D detection. The pre-trained weights of the voxel encoder and the SST backbone are utilized as initialization for the downstream 3D detection task.

Voxelization. Let p denote a point with coordinates x, y, z. Given a point cloud $P = \{p_i = [x_i, y_i, z_i]^T \in \mathbb{R}^3\}_{i=1...n}$ containing n points and range W, H, D, we partition the space into a 3D grid using voxel sizes v_W, v_H, v_D along the three axes, grouping points in the same voxel together. As a common technique [12, 19, 49], we decorate each point with $x^c, y^c, z^c, x^v, y^v, z^v$, where the superscript c denotes distance to the clustering center of all points in each voxel and the superscript v denotes distance to the voxel center. Considering the sparse and irregular nature of LiDAR point clouds, we focus only on non-empty voxels. Assuming non-empty voxels contain at most T points and a total of N non-empty voxels, we obtain the voxel-based point cloud representation as $M = \{V_i\}_{i=1...N}$, where $V_i = \{p_j = [x_j, y_j, z_j, x_j^c, y_j^c, z_j^c, x_j^v, y_j^v, z_j^v]^T \in \mathbb{R}^9\}_{j=1...t_i}$ and $t_i \leq T$. We assume each non-empty voxel contains T points for simplicity. We also maintain each voxel's coordinate in the 3D grid with $C_i \in \mathbb{R}^3$.

Masked voxel sampling. A key component of masked autoencoders is the mask sampling strategy. Unlike 2D pixels that uniformly distribute on the image plane, voxels are unevenly distributed due to the sparse and irregular nature of LiDAR point clouds. Random sampling may result in information loss in regions with sparse points. Inspired by furthest point sampling [27] (FPS), which samples an evenly distributed point set while maintaining the key struc-

ture of the data, we propose a Reversed-Furthest-Voxel-Sampling (R-FVS) scheme. Given a masking ratio r, we apply FPS to select $\lfloor N(1-r) \rfloor$ voxels from all non-empty voxels according to each voxel's coordinate C_i. These sampled voxels are *kept* to preserve points in sparse regions and maintain key geometric structures in the data, while the remaining unsampled voxels are masked during training.

Masking voxels. We directly mask the raw inputs of voxels, providing a unified formulation for the masked voxel modeling framework. For each of the remaining unsampled $\lceil Nr \rceil$ voxels to be masked, we replace some of the original features of each point in the voxel with a shared learnable vector, denoted as a mask token m. The dimensions of m and the replaced point features vary for different tasks.

Voxel encoding and decoding. All the non-empty voxels are then encoded by a voxel encoder (VE) and the SST backbone. As the features of masked voxels are also extracted by the backbone, we only need a lightweight task-specific MLP head to decode and predict their original information [36]. The loss is computed only on masked voxels, following previous practices [15, 36].

3.2. Masked Voxel Jigsaw (MVJ)

Object detection in LiDAR point clouds necessitates a representation capable of capturing contextual information, such as the distribution and relationships among voxels. We propose a jigsaw puzzle that obscures position information, compelling the model to learn these relationships.

Masking position information. Given a position-masked voxel, we replace the absolute coordinates x_j, y_j, z_j of each point with a mask token $m_v \in \mathbb{R}^3$, while preserving the local shape by retaining the decorated coordinates of each point within the voxel (Fig. 2). The masked voxel is then represented as $V_{\text{masked}} = \{p_j = [m_v, x_j^c, y_j^c, z_j^c, x_j^v, y_j^v, z_j^v]^T \in \mathbb{R}^9\}_{j=1...T}$. All voxels are subsequently input to the voxel encoder and the SST backbone. Notably, positional embeddings are not added to the voxels before being fed to the backbone, as the model is tasked with predicting position information.

Prediction target and loss function. Instead of directly predicting the absolute position of the masked voxels, we formulate a classification task to facilitate optimization. Owing to the extensive range of LiDAR point clouds, the SST backbone partitions the 3D voxelized grid into windows, akin to the Swin-transformer [22], to enable efficient attention calculation within windows. We employ the same window partitioning method, requiring the model to predict only the relative index of the masked voxel within its corresponding window. Assuming a 3D window contains N_x, N_y, N_z voxels along the x, y, z axes and the voxel coordinates of a masked voxel are X, Y, Z, the relative coordinates I_x, I_y, I_z of the masked voxel with respect to its window are calculated as $I_x = X \mod N_x, I_y = Y \mod N_y, I_z = Z \mod N_z$. The relative index is then given by $I = I_x + I_y N_x + I_z N_x N_y$. The prediction head outputs a classification vector $\hat{v} \in \mathbb{R}^{N_x N_y N_z}$ representing the probabilities of the masked voxel occupying each position. The prediction loss for all masked voxel is calculated using the Cross-Entropy loss as follows:

$$L_{MVJ} = \frac{1}{R_p} \sum_{i=1}^{R_p} \text{CrossEntropy}(\hat{v}_i, I_i), \tag{1}$$

where R_p denotes the number of position-masked voxels.

3.3. Masked Voxel Reconstruction (MVR)

To incorporate local shape information within each voxel, we introduce masked voxel reconstruction (MVR). MVR masks the absolute and relative coordinates of all points in shape-masked voxels to learn point distributions, preserving only one point to provide the voxel position.

Masking voxel shapes. In a given voxel, both the original and decorated coordinates of points represent the local shape of the voxel (Fig. 2). We employ a shared learnable token $m_p \in \mathbb{R}^9$ to replace all point features in every shape-masked voxel, except for one point. This particular point conveys the voxel's positional information without revealing the voxel shape, which would render the reconstruction task trivial. Consequently, the masked voxel becomes $V_{\text{masked}} = \{[x_1, y_1, z_1, x_1^c, y_1^c, z_1^c, x_1^v, y_1^v, z_1^v]\} \cup \{p_j = m_p\}_{j=2...T}$.

Reconstruction target and loss function. We reconstruct each masked voxel and use the $L2$ Chamfer Distance loss to measure the discrepancy between the reconstructed and target point distributions. This loss function is insensitive to point density [34], which is crucial since each voxel may contain varying numbers of points. Assuming each reconstructed voxel contains n points, the reconstruction head outputs a vector containing the coordinates of each point in the masked voxel, $\hat{v} \in \mathbb{R}^{3n}$. To pre-train the model more stably, we normalize the target point distribution using each point's distance to the voxel center as ground truth and scale the distance between 0 and 1. Supposing each reconstructed masked voxel is $\hat{V} = \{\hat{p}_j = [\hat{x}_j, \hat{y}_j, \hat{z}_j]\}_{j=1...n}$ and the ground truth is $V = \{p_j = [x_j^v, y_j^v, z_j^v]\}_{j=1...t_i}$, the total loss across all masked voxels is given by:

$$L_{MVR} = \frac{1}{R_s} \sum_{i=1}^{R_s} L_{CD}(\hat{V}_i, V_i), \tag{2}$$

where R_s represents the number of shape-masked voxels and L_{CD} is the $L2$ Chamfer Distance loss calculated as:

$$\begin{aligned} L_{CD} = &\frac{1}{\left|\hat{V}_i\right|} \sum_{\hat{p} \in \hat{V}_i} \min_{p \in V_i} \|\hat{p} - p\|_2^2 \\ &+ \frac{1}{|V_i|} \sum_{p \in V_i} \min_{\hat{p} \in \hat{V}_i} \|p - \hat{p}\|_2^2. \end{aligned} \tag{3}$$

3.4. Joint Pre-Training

We jointly pre-train the model using both MVJ and MVR tasks to enable the model to learn both point and voxel distributions. We use R-FVS to sample R_s voxels for MVR and R_p voxels for MVJ. As illustrated in Fig. 2, the voxel encoder and the backbone extract features from both types of masked voxels and unmasked voxels. Masked features are decoded to recover their corresponding targets, and Cross-Entropy loss and Chamfer Distance loss are utilized for supervision. The total loss for joint pre-training is expressed as follows:

$$L = \alpha L_{MVJ} + \beta L_{MVR} \tag{4}$$

where α and β are balancing coefficients.

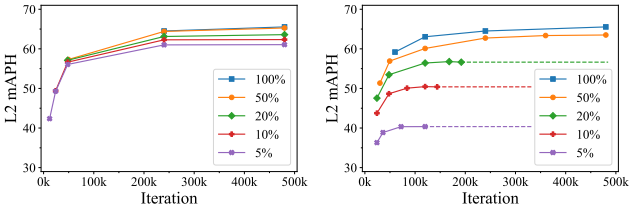
4. Data-Efficient Benchmark on Waymo

To evaluate the pre-trained representation, we transfer the weights of the voxel encoder and the backbone for object detection. Previous benchmarks fail to effectively reveal the effectiveness of pre-training strategies due to data diversity issues. We propose a new data-efficient benchmark to address these limitations.

Observation: Incomplete model convergence. Prior works [20, 40] fine-tune the model on different splits with the same *epochs*, causing reduced iteration numbers when dataset sizes decrease. As illustrated in Fig. 3a, by training for more epochs (but the same *iterations*), the performance of SST significantly improves, suggesting that the models may not fully converge on smaller splits when training epochs remain constant. This entangled factor prevents accurate evaluation of pre-training strategies.

Issue: Fine-Tuning splits with similar data diversities. With the same training iterations, models trained on different splits display comparable performance, despite significant differences in data amounts (Fig. 3a). We find that scene diversity is a crucial factor. Previous works [20, 40] uniformly sample the entire Waymo training set to create different fine-tuning splits, for instance, sampling one frame from every two consecutive frames for a 50% split. However, neighboring frames in self-driving datasets are similar due to the short time difference between consecutive frames (e.g., only $0.1s$ in Waymo [30]). This results in similar scene diversity across splits, leading to comparable detection performance using the same training iterations.

We argue that such an experimental setting does not accurately represent a typical application scenario of self-supervised pre-training, where annotated fine-tuning data is less abundant and diverse compared to pre-training data. Furthermore, it fails to evaluate how pre-training benefits downstream tasks when varying amounts and diversities of labeled data are available. We provide an additional example in the supplementary material to further illustrate



(a) Existing (b) Ours

Figure 3. Comparison of data-efficient benchmarks. Previous uniformly sampled splits exhibit similar diversity, as evidenced by similar performance with the same training iterations. Our benchmark features varying data amounts and diversities, ensuring model convergence. Dashed lines show baseline performances.

these issues with previous benchmarks. Our proposed data-efficient benchmark addresses these issues, offering a more precise evaluation of pre-training strategies.

Solution: Sequence-Based data sampling. We sample fine-tuning data using different scene sequences rather than uniformly sampling varying numbers of frames from identical sequences. The Waymo training split comprises 798 distinct scene sequences, each lasting $20s$ with approximately 200 frames [30]. We randomly sample 5%, 10%, 20%, 50% of the scene sequences, respectively, using all frames within the sampled sequences for fine-tuning. Consequently, data diversities and amounts vary across each split. Each larger split also encompasses the smaller ones, enabling measurement of performance changes with additional fine-tuning data. Furthermore, we randomly sample three 5% and 10% splits and report average results to reduce variance on these smaller sets (see our supplementary material).

For different fine-tuning splits, we train the model until performance saturates (i.e., overfitting) and determine the number of epochs used for experiments accordingly. For example, for 5% data, we try {48, 60, 66, 72, 78, 84} epochs and ultimately select 72. The baseline performances across various training iterations are shown in Fig. 3b. With our new benchmark, baseline performance and convergence speed vary as anticipated for different splits.

5. Experiments

5.1. Experimental Settings

Datasets. The Waymo dataset [30] consists of 1150 self-driving scenes with 798 scenes as the training split, 202 scenes as the validation split, and 150 scenes as the testing split. We pre-train on the training split, which contains 158,240 annotated frames. For the downstream data-efficient 3D object detection task, we fine-tune the model using our sampled splits (as described in Sec. 4) and evaluate performance on the standard Waymo validation split with 3 classes (cars, pedestrians, and cyclists) and two difficulty levels (L1 and L2). We adopt L2 mean average preci-

Table 1. Data-efficient 3D object detection results of SST on the Waymo validation set. SST is pre-trained using various self-supervised methods on Waymo training split, with weights applied for the downstream detection task. "Random initialization" denotes training from scratch. The model performances are shown using different amounts and diversities of fine-tuning data.

Data amount	Initialization	Overall		Car		Pedestrian		Cyclist	
		L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH	L2 mAP	L2 mAPH
5%	Random	44.41	40.34	51.01	50.49	52.74	42.26	29.49	28.27
	PointContrast [35]	45.32	41.30	52.12	51.61	53.68	43.22	30.16	29.09
	ProposalContrast [40]	46.62	42.58	52.67	52.19	54.31	43.82	32.87	31.72
	MV-JAR (Ours)	**50.52**$^{+6.11}$	**46.68**$^{+6.34}$	56.47	56.01	57.65	47.69	37.44	36.33
10%	Random	54.31	50.46	54.84	54.37	60.55	50.71	47.55	46.29
	PointContrast [35]	53.69	49.94	54.76	54.30	59.75	50.12	46.57	45.39
	ProposalContrast [40]	53.89	50.13	55.18	54.71	60.01	50.39	46.48	45.28
	MV-JAR (Ours)	**57.44**$^{+3.13}$	**54.06**$^{+3.60}$	58.43	58.00	63.28	54.66	50.63	49.52
20%	Random	60.16	56.78	58.79	58.35	65.63	57.04	56.07	54.94
	PointContrast [35]	59.35	55.78	58.64	58.18	64.39	55.43	55.02	53.73
	ProposalContrast [40]	59.52	55.91	58.69	58.22	64.53	55.45	55.36	54.07
	MV-JAR (Ours)	**62.28**$^{+2.11}$	**59.15**$^{+2.37}$	61.88	61.45	66.98	59.02	57.98	57.00
50%	Random	66.43	63.36	63.81	63.38	70.78	63.05	64.71	63.66
	PointContrast [35]	65.51	62.21	62.66	62.23	69.82	61.53	64.04	62.86
	ProposalContrast [40]	65.76	62.49	62.93	62.50	70.09	61.86	64.26	63.11
	MV-JAR (Ours)	**66.70**$^{+0.27}$	**63.69**$^{+0.33}$	64.30	63.89	71.14	63.57	64.65	63.63
100%	Random	68.50	65.54	64.96	64.56	72.38	64.89	68.17	67.17
	PointContrast [35]	68.06	64.84	64.24	63.82	71.92	63.81	68.03	66.89
	ProposalContrast [40]	68.17	65.01	64.42	64.00	71.94	63.94	68.16	67.10
	MV-JAR (Ours)	**69.16**$^{+0.66}$	**66.20**$^{+0.66}$	65.52	65.12	72.77	65.28	69.19	68.20

sion (L2 mAP) and L2 mean average precision with heading (L2 mAPH) as main evaluation metrics. We also evaluate the model on the KITTI dataset [14] for the detection task. It contains 7481 training samples and 7518 test samples.

Implementation details. We employ the official implementation and training settings of SST [12] for pre-training and fine-tuning unless specified otherwise. The Waymo dataset's point cloud range is $W \times H \times D = 149.76m \times 149.76m \times 6m$, and the voxel size is $v_W \times v_H \times v_D = 0.32m \times 0.32m \times 6m$. For MVR, we predict 15 points per voxel with a masking ratio of 0.05. For MVJ, the window size is $N_x \times N_y \times N_z = 12 \times 12 \times 1$ and the masking ratio is 0.1. We set the loss weight $\alpha = \beta = 1$, pre-train for 6 epochs with an initial learning rate of $5e-6$. When fine-tuning, we initialize the voxel encoder and SST backbone with pre-trained weights, and other training hyper-parameters remain unchanged. We set KITTI's point cloud range to $69.12m \times 79.36m \times 4m$, voxel size to $0.32m \times 0.32m \times 4m$, and train for 160 epochs. We use the AdamW optimizer with a batch size of 8 and a cyclic learning rate scheduler with cosine annealing for all training.

5.2. Main Results on Waymo

Baselines. We fine-tune the model using data splits described in Sec. 4. For {5%, 10%, 20%, 50%, 100%} data, the model is trained for {72, 60, 42, 36, 24} epochs. We use the randomly initialized model's performances as our baselines. We also include experiments with models using convolutional backbones in the supplementary material.

Results with different initialization. Tab. 1 presents the SST model's performances with various initializations on our proposed data-efficient benchmark. Our pre-training method consistently improves SST baselines across all fine-tuning data amounts, especially when data is scarce. Our method achieves up to 15.7% relative improvement, from 40.34% L2 mAPH to 46.68% L2 mAPH, when using only 5% fine-tuning data. The performance gain exists across object categories, indicating effective general representation learning. As more fine-tuning data is introduced, the performance gain persists, and when fine-tuning with the entire Waymo training set, our pre-training enhances detection performance from 65.54% mAPH to 66.20% mAPH.

We note that with 50% and 100% fine-tuning data, the improvements from our pre-training are not as significant as with less data. We observe that the baseline result on 50% data (63.36% mAPH) is comparable to that on 100% data (65.54% mAPH), suggesting that training data is no longer the bottleneck. The original SST, with only 2.1M parameters, is a lightweight network. We scale up SST to 8.3M parameters by using larger hidden layer channels and observe larger benefits from pre-training on the 50% fine-tuning split, from 63.13% L2 mAPH to 64.24% L2 mAPH. Therefore, we argue that the SST model's capacity, rather than the fine-tuning data amount, becomes the bottleneck for detection performance.

Comparisons with contrastive learning methods. We

Table 2. Transferring results on the KITTI validation split. We pre-train SST with the Waymo training split and fine-tune with the KITTI training split.

Initialization	Overall		
	Easy	Mod.	Hard
Random	74.71	63.43	60.00
PointContrast [35]	73.35	62.53	59.01
ProposalContrast [40]	73.63	63.34	59.40
MV-JAR (Ours)	**75.22**	**63.80**	**60.35**

utilize the official implementation of the recently proposed ProposalContrast [40] and reimplement PointContrast [35] for comparison. As demonstrated in Tab. 1, both contrastive learning methods offer benefits only with 5% fine-tuning data, and their improvements diminish with higher baseline performances. This may be due to the difficulty in smoothly learning representations. A key challenge in LiDAR-based contrastive learning is constructing matched pairs. The point pairs used by PointContrast face the issue of spatially adjacent points with similar features being assigned as negative samples. We observe difficulty in convergence when sampling excessive pairs. ProposalContrast alleviates the issue by contrasting regions but does not entirely resolve it. Alternatively, our method offers a new avenue for developing self-supervised pre-training through masked voxel modeling on LiDAR point clouds.

5.3. Pre-training Accelerates Convergence

To investigate how MV-JAR pre-training accelerates convergence, we fine-tune the SST model on the entire Waymo training split. We train the SST model for different epochs and report the performance of both fine-tuning and training from scratch. As illustrated in Fig. 1, MV-JAR pre-training significantly enhances the convergence speed of SST. By fine-tuning for only 3 epochs, the SST model achieves 62.48% L2 mAPH. Remarkably, fine-tuning for just 12 epochs results in a performance comparable to training from scratch for 24 epochs, effectively halving the convergence time. As the number of fine-tuning epochs increases, the performance gain provided by MV-JAR does not wane. Pre-training can still improve the model's converged performance by 0.74% L2 mAPH, highlighting the superiority of the pre-trained representation.

5.4. Transferring Results on KITTI

To evaluate the transferability of the learned representation, we fine-tune various pre-trained SST models on the KITTI dataset. The original SST paper [12] did not conduct experiments on the KITTI dataset; thus, we follow the training schedule and setup of SST and PointPillars [19] within the MMDetection3D framework [9]. As demonstrated in Tab. 2, the performance gain from our MV-JAR pre-training persists when the representation is transferred to a different domain, indicating that the model learns a generic representation through pre-training. While the KITTI training samples account for approximately 5% of the entire Waymo training split, the relative improvement is much smaller than transferring to the 5% Waymo split. We hypothesize that this may be attributed to the domain gap between the Waymo and KITTI datasets.

5.5. Ablation Studies

In this section, we conduct ablation studies to investigate the effectiveness of our proposed Reversed-Furthest-Voxel-Sampling (R-FVS) strategy. Additionally, we explore the optimal masking ratio and the impact of MVR and MVJ pre-training. All pre-training experiments utilize the entire Waymo training split, and we fine-tune the model on one of our 5% data splits.

Mask sampling strategy. Our R-FVS mask sampling strategy masks voxels that are not sampled by FPS, aiming to avoid masking voxels located in sparse regions. We also examine random sampling and an opposing sampling strategy called FVS, which masks voxels sampled by FPS. With FVS sampling, regions with sparse points are more likely to be masked, leading to greater information loss compared to random sampling and R-FVS sampling. Tab. 3 presents the fine-tuning results of pre-training with the three sampling strategies. R-FVS sampling performs the best, while FVS performs the worst. These findings confirm that minimizing information loss benefits pre-training and validate the effectiveness of our R-FVS strategy.

Masking ratio. Tab. 4 illustrates the impact of different masking ratio combinations. Specifically, MVR with a 0.1 masking ratio and MVJ with a 0.05 masking ratio yield the best results. With this combination, the overall masking ratio is 0.15, which is significantly lower than the masking ratios in the image [15, 36] or video domain [13, 32]. A high masking ratio in masked autoencoders typically indicates information redundancy [13, 15, 32, 36]. We postulate that two factors contribute to our low masking ratio: 1) LiDAR point clouds represent a vast space where each meaningful object occupies only a small fraction. The information on each object is not highly redundant, especially when the objects are distant from the sensor. This contrasts with images in datasets like ImageNet, where objects often encompass a significant portion of the image. 2) The SST model calculates attention only within a partitioned window, which restricts the information used. We leave this question for future exploration.

Pre-training effects of MVR and MVJ. We investigate the individual pre-training effects of MVR and MVJ by optimizing the masking ratio for each task and reporting the fine-tuning performances in Tab. 5. Both MVR and MVJ pre-training enhance performance compared to random ini-

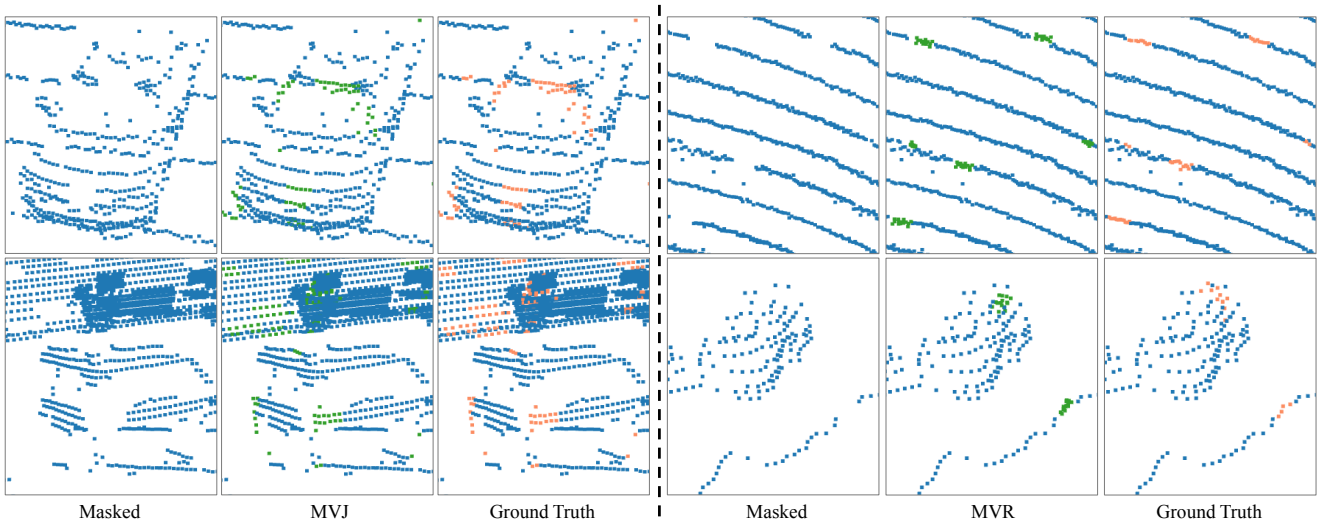


Figure 4. Visualization of the reconstruction results of MV-JAR on the Waymo validation set, compared with ground truths.

Table 3. Ablation study on mask sampling strategies.

Mask sampling	Overall	
	L2 mAP	L2 mAPH
FVS	49.73	45.88
Random	49.86	46.06
R-FVS	**50.52**	**46.68**

Table 4. Ablation study on masking ratios.

Masking ratio		Overall	
MVJ	MVR	L2 mAP	L2 mAPH
0.3	0.1	49.74	46.05
0.05	0.1	49.67	45.92
0.1	0.1	49.85	45.97
0.1	**0.05**	**50.52**	**46.68**
0.1	0.3	50.25	46.35

Table 5. Ablation study on pre-training effects of MVR and MVJ.

Task	Overall	
	L2 mAP	L2 mAPH
MVR	46.57	42.73
MVJ	49.96	46.10
MV-JAR	**50.52**	**46.68**

tialization, with MVJ outperforming MVR. This finding suggests that capturing voxel distributions plays a more significant role in representation learning, which is consistent with the fact that LiDAR detectors process points by downsampling them into voxels. Our integrated MV-JAR method demonstrates further improvement over MVR and MVJ, validating the effectiveness of jointly capturing point and voxel distributions. Additionally, we offer an analysis of improvements across various distances in our supplementary material.

5.6. Visualization

Figure 4 displays the MV-JAR reconstruction results on the Waymo validation set, including MVJ and MVR outcomes. In our pre-training, MVJ achieves approximately 89% classification accuracy of masked voxels on the validation set, accurately reconstructing masked voxels in the correct positions most of the time. MVR can capture the point distributions within voxels, evidenced by reconstructed ground points aligning with ground circles. However, MVR struggles to capture detailed distributions, likely due to the discrete sampling of LiDAR points from continuous surfaces, resulting in considerable variability in the data and increased difficulty in capturing finer details.

6. Conclusions

In this paper, we introduce the Masked Voxel Jigsaw and Reconstruction (MV-JAR) pre-training method for LiDAR detectors. MV-JAR captures both point and voxel distributions of LiDAR point clouds, enabling models to learn effective and generic representations. We also develop a Reversed-Furthest-Voxel-Sampling strategy to address the uneven distribution of LiDAR points. Comprehensive experiments on the Waymo and KITTI datasets show that our method consistently and significantly enhances the detector's performance across different data scale regimes. MV-JAR offers a promising alternative for LiDAR-based self-supervised pre-training through Masked Voxel Modeling. Additionally, we establish a new data-efficient benchmark on the Waymo dataset, incorporating fine-tuning splits with diverse data variations. This benchmark effectively assesses the impact of pre-training on downstream tasks with varying amounts and diversities of labeled data.

Acknowledgements. This project is funded in part by the Shanghai AI Laboratory, CUHK Interdisciplinary AI Research Institute, and the Centre for Perceptual and Interactive Intelligence (CPII) Ltd. under the Innovation and Technology Commission (ITC)'s InnoHK.

References

[1] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv:2106.08254*, 2021. 1, 2

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv:1903.11027*, 2019. 2

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 1

[5] Runjian Chen, Yao Mu, Runsen Xu, Wenqi Shao, Chenhan Jiang, Hang Xu, Yu Qiao, Zhenguo Li, and Ping Luo. CO3: Cooperative unsupervised 3d representation learning for autonomous driving. In *ICLR*, 2023. 2

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1

[7] Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *arXiv:2202.03026*, 2022. 1, 2

[8] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 1

[9] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 7

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[12] 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 *CVPR*, 2022. 1, 2, 3, 6, 7

[13] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv:2205.09113*, 2022. 2, 7

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 6

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 2, 4, 7

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1

[17] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *CVPR*, 2021. 2

[18] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *CVPR*, 2021. 2

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 3, 7

[20] 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 *CVPR*, 2021. 1, 2, 5

[21] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *ECCV*, 2022. 2

[22] 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 *CVPR*, 2021. 3, 4

[23] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *CVPR*, 2021. 2, 3

[24] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *CVPR*, 2021. 2

[25] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *CVPR*, 2021. 2

[26] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022. 2

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 3

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 2

[29] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, 2019. 2

[30] 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 *CVPR*, 2020. 2, 5

[31] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *ECCV*, 2022. 2

[32] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022. 2, 7

[33] Tai Wang, Xinge Zhu, and Dahua Lin. Reconfigurable voxels: A new representation for lidar-based point clouds. In *CoRL*, 2020. 1

[34] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Balanced chamfer distance as a comprehensive metric for point cloud completion. In *NeurIPS*, 2021. 4

[35] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020. 2, 6, 7

[36] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *CVPR*, 2022. 1, 2, 4, 7

[37] Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, Christoph Feichtenhofer, et al. Masked autoencoders that listen. *arXiv:2207.06405*, 2022. 2

[38] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 1

[39] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 2

[40] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In *ECCV*, 2022. 1, 2, 5, 6, 7

[41] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 1

[42] 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 *CVPR*, 2022. 2

[43] Shuangfei Zhai, Navdeep Jaitly, Jason Ramapuram, Dan Busbridge, Tatiana Likhomanenko, Joseph Yitan Cheng, Walter Talbott, Chen Huang, Hanlin Goh, and Joshua Susskind. Position prediction as an effective pretraining strategy. In *ICML*, 2022. 2

[44] 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. *arXiv:2205.14401*, 2022. 2

[45] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. Dense siamese network for dense unsupervised learning. In *ECCV*, 2022. 1

[46] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *CVPR*, 2021. 2

[47] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv:2111.07832*, 2021. 1, 2

[48] Lei Zhou, Huidong Liu, Joseph Bae, Junjun He, Dimitris Samaras, and Prateek Prasanna. Self pre-training with masked autoencoders for medical image analysis. *arXiv:2203.05573*, 2022. 2

[49] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 3

[50] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *ECCV*, 2020. 1