TREND: Unsupervised 3D Representation Learning via Temporal Forecasting for LiDAR Perception

Runjian Chen[1] Hyoungseob Park[2] Bo Zhang[3] Wenqi Shao[3] Ping Luo[1*] Alex Wong[2*]

[1]The University of Hong Kong [2]Yale University [3]Shanghai AI Laboratory

{rjchen, pluo}@cs.hku.hk {hyoungseob.park, alex.wong}@yale.edu

{zhangbo, shaowenqi}@pjlab.org.cn

Abstract

*Labeling LiDAR point clouds is notoriously time-and-energy-consuming, which spurs recent unsupervised 3D representation learning methods to alleviate the labeling burden in LiDAR perception via pretrained weights. Almost all existing work focus on a single frame of LiDAR point cloud and neglect the temporal LiDAR sequence, which naturally accounts for object motion (and their semantics). Instead, we propose TREND, namely **T**emporal **RE**ndering with **N**eural fiel**D**, to learn 3D representation via forecasting the future observation in an unsupervised manner. Unlike existing work that follows conventional contrastive learning or masked auto encoding paradigms, TREND integrates forecasting for 3D pre-training through a Recurrent Embedding scheme to generate 3D embedding across time and a Temporal Neural Field to represent the 3D scene, through which we compute the loss using differentiable rendering. To our best knowledge, TREND is the first work on temporal forecasting for unsupervised 3D representation learning. We evaluate TREND on downstream 3D object detection tasks on popular datasets, including NuScenes, Once and Waymo. Experiment results show that TREND brings up to 90% more improvement as compared to previous SOTA unsupervised 3D pre-training methods and generally improve different downstream models across datasets, demonstrating that indeed temporal forecasting brings improvement for LiDAR perception. Codes and models will be released.*

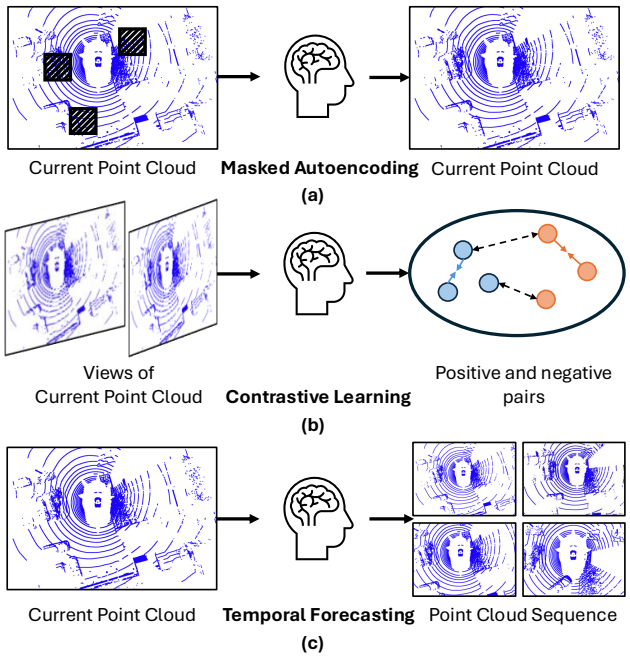

Figure 1. Different schemes for unsupervised 3D representation learning. (a) Masked Autoencoding first applies random masked on current LiDAR point cloud and then pre-train 3D backbones with a reconstruction objective. (b) Contrastive-based methods build up different views of current point cloud and pre-train the networks by pulling together positive pairs and pushing away negative pairs. (c) Our proposed TREND explores object motion and semantic information in LiDAR sequence and introduces temporal forecasting for unsupervised 3D pre-training.

1. Introduction

Light-Detection-And-Ranging (LiDAR) is widely used in autonomous driving. By emitting laser rays into the surrounding environment, it provides an accurate estimation of the distance along each ray with time-of-flight principle. There has been strong research interest on LiDAR-based perception like 3D object detection [2, 9, 25, 33, 35, 52, 56] and semantic segmentation [12, 64]. However, labeling for LiDAR point clouds is notoriously time-and-energy-consuming. According to [44], it costs an expertise labeler at least 10 minutes to label one frame of LiDAR point

*Corresponding authors.

cloud at a coarse-level and more at finer granularity. Assuming sensor frequency at $20Hz$, it could cost more than 1000 days of a human expert to annotate a one-hour sequence of LiDAR point clouds. To alleviate the labeling burden, unsupervised 3D representation learning [7, 13–15, 19, 22, 48, 49, 53, 54, 65] pre-trains 3D backbone to initialize downstream models for performance improvement with the same number of labels for downstream task.

Previous literature on unsupervised 3D representation learning for LiDAR perception can be divided into two streams, as shown in Figure 1 (a) and (b). (a) Masked-autoencoder-based methods [14, 49, 53, 54, 65] randomly mask LiDAR point clouds and the pre-training entails reconstructing the masked areas. (b) Contrastive-based methods [7, 19] construct two views from one frame of LiDAR point cloud and maximize the similarity among positive pairs while minimizing the similarity of negative pairs. Both approaches assume a predefined set of nuisance variability. In (a), it is occlusions, which naturally is induced by motion; in (b) it is the handcrafted set of transformations used in contrastive learning. While the procedures are unsupervised, they implicitly select the set of invariants, which benefits the downstream tasks. Unlike them, we subscribe to allowing the data to determine nuisances by simply observing and predicting scene dynamics. This leads to a novel unsupervised 3D representation learning approach based on forecasting LiDAR point clouds (Figure 1 (c)). Naturally, points belonging to the same object instance, within a point cloud, tend to move together. By observing current point cloud and predicting future observation, our pretraining scheme implicitly encodes semantics and biases of object interactions over time.

However, leveraging forecasting as unsupervised 3D representation is nontrivial as scene dynamics are often complex and nonlinear. There are two main challenges: 1) How to generate 3D embeddings at different timestamps from current 3D embeddings? 2) How to represent the 3D scene with embeddings and optimize the network via forecasting the future observation? In this paper, we delve into these two challenges and propose TREND, namely **T**emporal **RE**ndering with **N**eural fiel**D**, for unsupervised 3D pre-training via temporal forecasting.

First of all, there exists tangential work in occupancy prediction field [1, 18, 59] that generates 3D features at different timestamps via directly use 3D/2D convolution [1, 18] or a deep diffusion-based decoder with frozen 3D encoder [59]. The former way does not take the action of the ego-vehicle into account, which reflects the interaction between ego-vehicle and other traffic participants. The latter one fixes 3D encoder when training to forecast future, making the 3D encoder unaware of the temporal information. In order to solve the problems above, we propose a Recurrent Embedding scheme and generate 3D embeddings along time axis with the action of the ego-vehicle and a shallow 3D convolution.

Secondly, TREND takes the inspiration from [14, 27, 43, 54, 65] and applies neural-field-decoder to render LiDAR point clouds at current and future timestamps. However, directly using the neural field in [14, 54, 65] to represent the 3D scene at different timestamps yields little to no improvement. The main reason is that the network needs to learn to understand the concept of "time" with the 3D convolution, which could be very difficult. On the contrary, we propose a Temporal Neural Field in TREND, which explicit takes timestamps as inputs, and a differentiable rendering process to reconstruct and forecast LiDAR point clouds for optimizing the network.

We demonstrate TREND on three benchmark datasets (NuScenes [5], Once [24] and Waymo [36]) for the downstream 3D object detection task, where TREND improves over previous SOTA pre-training methods by 90% on NuScenes, and by up to 1.77 points in mAP over training-from-scratch for Once.

2. Related Work

Pre-training for Point Cloud. Since annotating 3D point clouds requires significant effort and time, there has aroused great interest on improving label efficiency for point cloud perception via 3D pre-training. Starting from CAD-model point clouds, [20, 29, 31, 42, 50, 55, 57, 60] propose various pre-training method ranging from masked auto-encoder to reconstruction and point cloud completion, where downstream tasks are normally cad model point cloud classification and segmentation. For indoor scene point cloud, PointContrast [48] is a pioneering work to first reconstruct the whole scene and use contrastive learning for pre-training, followed by P4Contrast [22] and Contrastive-Scene-Context [13]. For outdoor scene LiDAR point clouds, research can be divided into two branches depending on whether labels are required during the pre-training stage. Embraced by AD-PT [58] and SPOT [51], the first branch is semi-supervised 3D pre-training that utilizes a few labels during pre-training and the pre-training tasks include object detection (AD-PT [51]), occupancy prediction (SPOT [51]) and so on. The second branch is unsupervised 3D representation learning where no label is required during pre-training. 1) Contrastive-based methods [7, 15, 19, 28] build adequate views for outdoor scene LiDAR point cloud and conduct contrastive learning to improve the performance in downstream LiDAR perception task. 2) Mask-Autoencoder-based methods [14, 49, 53, 54, 65] first mask the input LiDAR point clouds and reconstruct the masked part to pre-train 3D backbones. Among the works above, only STRL [15] and SPOT [51] utilize temporal information during pre-training. STRL [15] is initially proposed on 3D pre-training for static indoor scenes and use point cloud

at different timestamps as different views for contrastive learning. However, outdoor scenes are generally dynamic and this makes it hard to find correct correspondence for contrastive learning, which results in inferior performance in downstream task. SPOT [51] generates pre-training labels with multiple frames of LiDAR point clouds and labels but only use the labels at current frame for pre-training. A concurrent work called T-MAE [47] proposes to use the adjacent previous frame of LiDAR point clouds for masked autoencoding pre-training, where temporal information is limited to two frames (less than 0.5 second) and only history information is used. Additionally, action embedding of the ego-vehicle is not utilized in T-MAE [47]. This makes the pre-training lacks of information about the interaction of ego-vehicle and other traffic participants. Furthermore, the decoder in [47] is simply Multi-layer Perceptron on occupied 3D space but understandings about empty parts of the environments also benefits downstream tasks. On the contrary, we propose TREND and use temporal forecasting as the pre-training goal. TREND utilizes a Sequential Embedding scheme for temporal forecasting and a Temporal Neural Field as decoder, which makes it able to incorporate longer length of point cloud sequence and gain full understanding about the 3D scenes.

LiDAR-based Neural Field. Neural Field plays an important role in 3D scene representation [27, 43]. Recently, researchers working on LiDAR sensor introduce neural field into scene reconstruction with LiDAR point clouds as inputs and propose Neural LiDAR Field [16, 37, 62], which takes second-return properties of LiDAR sensor into consideration and reconstruct intensity. IAE [50] and Ponder [14] are pioneering work to introduce neural field into 3D pre-training and both of them use reconstruction as pre-training task. In our paper, we use time-dependent neural field as part of our pre-training decoder and pre-training task is to forecast future LiDAR point clouds.

3D Scene Flow and LiDAR Point Cloud Forecasting. 3D scene flow has long been investigated [21, 26, 40, 41, 46, 61]. The inputs are current and future point clouds and the goal is to estimate per-point translation for the current point clouds, which means that without future point clouds as inputs, it is difficult to forecast the future sensor observation. Recently, there arouses great interest for research in LiDAR point cloud forecasting, where the inputs are past observations and prediction goal are the future LiDAR observations. Representative works include 4DOCC [18], Copilot4D [59] and Uno [1]. 4DOCC [18] uses a U-Net convolutional architecture and conduct differentiable rendering on the bev feature map to predict the LiDAR observation in the future. Copilot4D [59] first trains a tokenizer/encoder for LiDAR point cloud with masked-and-reconstruction task and then freeze the encoder to train a diffusion-based decoder for LiDAR forecasting. Uno [1] proposes to use occupancy field as the scene representation for point cloud forecasting. The forecasting training stage in Copilot4D [59] does not envolve the 3D encoder for LiDAR point cloud and only focuses on training the diffusion-based decoder, which actually does not introduce temporal information into the 3D encoder. 4DOCC [18] and Uno [1] train the 3D encoder for forecasting but do not take the action of the autonomous vehicle into consideration. However, the interaction between the autonomous vehicle and the traffic participants is important for the prediction. In this paper, TREND adapt point cloud forecasting for unsupervised 3D representation learning and takes the action of the autonomous vehicle as inputs for forecasting.

LiDAR-based 3D Object Detection. LiDAR 3D object detectors aims to take the raw LiDAR point clouds as input and predict boundary boxes for different object categories in the scene. Existing literature on LiDAR-based 3D object detection can be divided into three main streams based on the 3D encoder of the detector. 1) Point-based methods [32, 34] apply point-level embedding to detect objects in the 3D space. 2) Embraced by [2, 9, 52, 56], voxel-based methods apply voxelization to the raw point clouds and use sparse 3D convolution to encode the 3D voxels, with which the detection head is able to localize and identify 3D objects. 3) Point-voxel-combination methods [33, 35] combine the point-level and voxel-level features from 1) and 2). In this paper, LiDAR-based 3D object detection is used as downstream task to evaluate the effectiveness of TREND.

3. Method

In this section, we introduce TREND for unsupervised 3D representation learning on LiDAR perception via temporal forecasting. As shown in Fig. 2, TREND pre-trains the 3D encoder with (a) Recurrent Embedding scheme that accounts for the effect of autonomous vehicle's action to generate 3D embeddings at different timestamps, (b) Temporal Neural Field, which represents the 3D scene with signed distance value predicted by a geometry feature extraction network f^{geo} and a signed distance network f^{SDF}. (c) Rendering current and future point clouds to compute loss and optimize the network. We first introduce problem formulation and overall pipeline in Section 3.1. Then we describe the Recurrent Embedding scheme and the Temporal Neural Field in details respectively in Section 3.2 and 3.3. Finally in Section 3.4, we discuss the differentiable rendering process and loss computation.

3.1. Problem Formulation and Pipeline

Notations. To start with, LiDAR point clouds are denoted as $\mathbf{P} = [\mathbf{L}, \mathbf{F}] \in \mathbb{R}^{N\times(3+d)}$, the concatenation of the xyz-location $\mathbf{L} \in \mathbb{R}^{N\times 3}$ and point features $\mathbf{F} \in \mathbb{R}^{N\times d}$. Here N

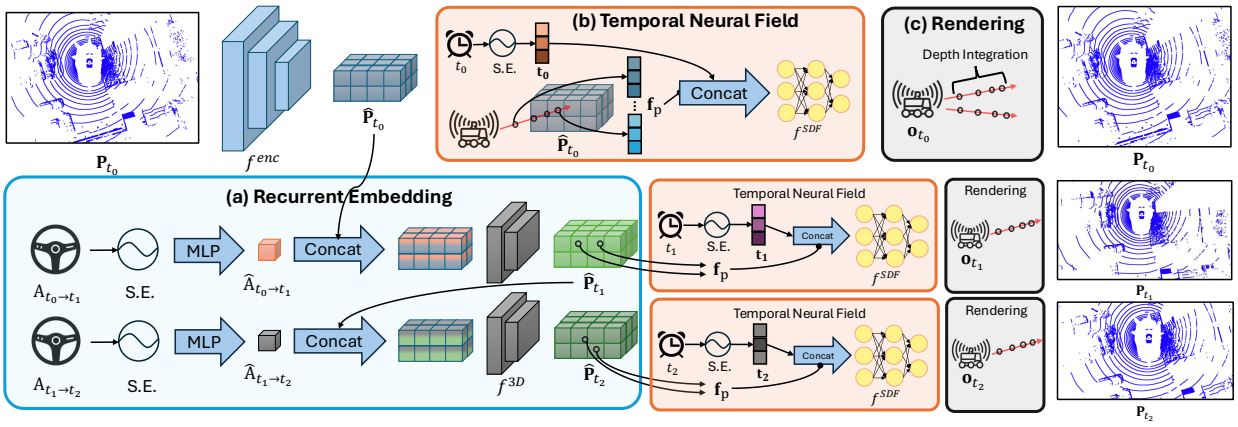

Figure 2. The pipeline of TREND. "S.E." means sinusoidal encoding [17, 39]. To pre-train the encoder f^{enc} via temporal forecasting in an unsupervised manner, TREND first generate 3D embeddings at different timestamps with a recurrent embedding scheme as shown in part (a). Action embeddings are computed with sinusoidal encoding and projected by an Multi-layer Perceptron. Then the action embeddings are repeated and concatenated with embeddings from previous timestamp, followed by a shared shallow 3D convolution f^{3D} to generate 3D embeddings for timestamp t_1, t_2, ... Then as described in part (b), a Temporal Neural Field is utilized to represent the 3D scene at different timestamps. We query features of the sampled points along LiDAR rays and concatenate them with sinusoidal embeddings of timestamps as well as the position of the sampled points to feed into a signed distance function [6, 23, 43] f^{SDF} for signed distance value prediction. Next, we conduct differentiable rendering to aggregate the sampled points along each ray and predict the ranges in the direction of the ray, that is reconstructing and forecasting the LiDAR point clouds at different timestamps. Finally we compute the pre-training loss with the predicted LiDAR point clouds and the actual LiDAR sequence.

means the number of points in the point clouds and d is the number of feature channels. For instance, $d = 1$ in Once [24] representing intensity and for Waymo [36], $d = 2$ are intensity and elongation. To indicate point clouds at different timestamps, we use subscripts and $\mathbf{P}_t = [\mathbf{L}_t, \mathbf{F}_t] \in \mathbb{R}^{N_t \times (3+d)}$ is point cloud at time $t \in \{t_0, t_1, t_2, ..., t_k\}$, where t_0 indicates current timestamp and $t_1, t_2, ...t_k$ are future timestamps. At each timestamp t_n, we also have the action $\mathbf{A}_{t_n \to t_{n+1}} = [\Delta_x, \Delta_y, \Delta_\theta] \in \mathbb{R}^3$ of the autonomous vehicle and it is described with the relative translation on x-y plane (Δ_x, Δ_y) and orientation with respect to z-axis (Δ_θ) between timestamp t_n and t_{n+1}.

Pipeline. Our goal is to pre-train the 3D encoder f^{enc} in an unsupervised manner via forecasting to leverage temporal information. Firstly, $\mathbf{P}_{t_0}$ are embedded with the 3D encoder f^{enc} to obtain the 3D representations

$$\hat{\mathbf{P}}_{t_0} = f^{\text{enc}}(\mathbf{P}_{t_0}), \tag{1}$$

where $\hat{\mathbf{P}}_{t_0} \in \mathbb{R}^{D \times H \times W \times \hat{d}}$ indicates the embedded 3D features with spatial resolution of $D \times H \times W$ and $\hat{d}$ feature channels. Then with $\hat{\mathbf{P}}_{t_0}$ and action at different timestamps $\mathbf{A}_{t_n \to t_{n+1}}$ as inputs, we apply the recurrent embedding scheme f^{rec} and get the 3D embedding at different timestamps

$$\hat{\mathbf{P}}_{t_{n+1}} = f^{\text{rec}}(\mathbf{A}_{t_n \to t_{n+1}}, \hat{\mathbf{P}}_{t_n}), \tag{2}$$

where $n = 0, 1, ...$ Finally, to guide the training of 3D encoder in an unsupervised manner, we use a Temporal Neural Field to reconstruct and forecast LiDAR point clouds $\tilde{\mathbf{P}}_{t_n}$

$$\tilde{\mathbf{P}}_{t_n} = f^{\text{render}}(\hat{\mathbf{P}}_{t_n}), \tag{3}$$

and compute the loss against the raw observation $\mathbf{P}_{t_n}$ for optimization. Note that all the LiDAR point clouds are transformed into the coordinate of t_0 for consistency.

3.2. Recurrent Embedding Scheme

In order to introduce temporal information into 3D pre-training for f^{enc}, we first embed current 3D representation $\mathbf{P}_{t_0}$ into future 3D representation ($\mathbf{P}_{t_1}$, $\mathbf{P}_{t_2}$ ...). To achieve this, previous literature [1, 18] directly apply learnable 3D/2D decoders but neglect the effect of autonomous vehicle's action $\mathbf{A}_{t_n \to t_{n+1}}$. However, the action of the autonomous vehicle is a part of the interaction between the autonomous vehicle and other traffic participants and may influence the motion of pedestrians and other vehicles on the road. For example, if the autonomous vehicle does not move for some time, other traffic participants might move faster and vice versa. Thus, we propose to take $\mathbf{A}_{t_n \to t_{n+1}}$ into account and use a recurrent embedding scheme.

To begin, sinusoidal encoding [17, 39] are used to encode the relative translation part $[\Delta_x, \Delta_y]$ in raw action $\mathbf{A}_{t_n \to t_{n+1}}$ with sinusoidal functions of different frequencies. The resulting translation feature $\mathbf{f}_{\text{tl}} \in \mathbb{R}^{d_{\text{sin}}}$ contains d_{sin} bounded scalars. Then we use $\mathbf{f}_{\text{rot}} = [\sin \Delta_\theta, \cos \Delta_\theta] \in \mathbb{R}^2$ to represent the rotation part in $\mathbf{A}_{t_n \to t_{n+1}}$ and concatenate both features to generate an initial embedding

$\tilde{\mathbf{A}}_{t_n \to t_{n+1}} = [\mathbf{f}_{\text{tl}}, \mathbf{f}_{\text{rot}}] \in \mathbb{R}^{d_{\text{sin}}+2}$ for $\mathbf{A}_{t_n \to t_{n+1}}$ without any learnable parameter. To further learn to embed $\tilde{\mathbf{A}}_{t_n \to t_{n+1}}$, we apply a shared shallow multi-layer perceptron (MLP) f^{act} and project it to $\hat{\mathbf{A}}_{t_n \to t_{n+1}} \in \mathbb{R}^{d_{\text{act}}}$

$$\hat{\mathbf{A}}_{t_n \to t_{n+1}} = f^{\text{act}}(\tilde{\mathbf{A}}_{t_n \to t_{n+1}}). \tag{4}$$

With 3D embeddings at current timestamp $\hat{\mathbf{P}}_{t_0}$ and action embeddings at different timestamps $\hat{\mathbf{A}}_{t_n \to t_{n+1}}$, we repeat $\hat{\mathbf{A}}_{t_n \to t_{n+1}}$ $D \times H \times W$ times and concatenate it with $\hat{\mathbf{P}}_{t_n}$ on feature dimension, followed by a shared shallow 3D dense convolution f^{3D} to get the embedding at different timestamps $\hat{\mathbf{P}}_{t_{n+1}} \in \mathbb{R}^{D \times H \times W \times \hat{d}}$.

$$\hat{\mathbf{P}}_{t_{n+1}} = f^{\text{3D}}([\mathbf{A}_{t_n \to t_{n+1}}, \hat{\mathbf{P}}_{t_n}]),\ n = 0, 1, ... \tag{5}$$

3.3. Temporal Neural Field

Inspired by [16, 27, 37, 43, 62], we propose the Temporal Neural Field to represent the 3D scene around the autonomous vehicle at different timestamp t, which is the basis for LiDAR point clouds rendering. As shown in Fig. 2, the goal of Temporal Neural Field is to inference the signed distance value [6, 23] for a point $\mathbf{p}$ in 3D space at timestamp t. Given the location of a specific point $\mathbf{p} = [x, y, z] \in \mathbb{R}^3$ at timestamp t, we first query the feature $\mathbf{f}_{\text{p}} \in \mathbb{R}^{\hat{d}}$ at $\mathbf{p}$ with $\hat{\mathbf{P}}_t$ by trilinear interpolation f^{tri} implemented by Pytorch [30]:

$$\mathbf{f}_{\text{p}} = f^{\text{tri}}(\mathbf{p}, \hat{\mathbf{P}}_t). \tag{6}$$

Similar to initial action embedding in Section 3.2, we apply sinusoidal encoding [17, 39] to encode timestamp t to $\mathbf{f}_{\text{t}} \in \mathbb{R}^{d_{\text{sin}}}$. Taking the concatenation of location $\mathbf{p}$, $\mathbf{f}_{\text{t}}$ and the queried feature $\mathbf{f}_{\text{p}}$ as inputs, we predicts the signed distance value $s \in \mathbb{R}$ [6, 23] with f^{SDF}, which is parameterized by Multi-layer Perceptron:

$$s = f^{\text{SDF}}([\mathbf{p}, \mathbf{t}, \mathbf{f}]). \tag{7}$$

3.4. Point Cloud Rendering

Each LiDAR point $\mathbf{p}$ can described by the sensor origin $\mathbf{o} \in \mathbb{R}^3$, normalized direction $\mathbf{d} \in \mathbb{R}^3$ and the range $r \in \mathbb{R}$, that is $\mathbf{p} = o + r\mathbf{d}$. Similar to [16, 27, 37, 43, 62], we first sample N_{render} rays at the sensor position $\mathbf{o}$, each of which is described by its normalized direction $\mathbf{d}$, and apply differentiable rendering to predict the depth of rays at different timestamp $t \in \{t_0, t_1, t_2, ...\}$ with Temporal Neural Field.

Sampling of N_{render}. Generally, background LiDAR points contain much less information compared to foreground ones. As TREND aims for an unsupervised 3D representation learning, we do not have labels for foreground or background objects. Instead, LiDAR points on the ground are often background points and we filter out ground points by setting a threshold z_{thd} for z values of the point position. z_{thd} is determined by sensor height provided in the datasets. After filtering of ground points, we uniformly sample N_{render} at timestamp t_n to conduct depth rendering and loss computation.

Depth Rendering. For a specific timestamp t, we sample N_{ray} points following [43] along each ray and construct the point set $\{\mathbf{p}_n = \mathbf{o} + r_n\mathbf{d}\}_{n=1}^{N_{\text{ray}}}$. For each point in the point set, we estimate the signed distance value s_n as described in Section 3.3. Then we predict the occupancy value α_n

$$\alpha_n = \max\left(\frac{\Phi_z(s_n) - \Phi_z(s_{n+1})}{\Phi_z(s_n)}, 0\right), \tag{8}$$

where $\Phi_z(x) = (1 + e^{-zx})^{-1}$ is the sigmoid function with a learnable scalar z. With α_n, we estimate the accumulated transmittance $\mathcal{T}_n$ [43] by

$$\mathcal{T}_n = \prod_{i=1}^{n-1}(1 - \alpha_i). \tag{9}$$

With $\mathcal{T}_n$ and α_n we follow a similar way proposed in [43] to compute an occlusion-aware and unbiased weight

$$w_n = \mathcal{T}_n \alpha_n. \tag{10}$$

Finally, differentiable rendering is conducted by integrating all the sampled points along the ray and the predicted range $\tilde{r}$ for this ray is computed,

$$\tilde{r} = \sum_{n=1}^{N_{\text{ray}}} w_n * r_n. \tag{11}$$

Loss Function. For each sampled ray, we have the observed range r^i and the predicted range $\tilde{r}^i$. We use a L-1 loss function to compute the loss at timestamp t_n,

$$\mathcal{L}_{t_n} = \frac{1}{N_{\text{render}}} \sum_{i=1}^{N_{\text{render}}} |r^i - \tilde{r}^i|. \tag{12}$$

3.5. Curriculum Learning for Forecasting Length

It is difficult for a randomly initialized network to directly learn to forecast several frames of LiDAR point clouds. Thus we propose to borrow the idea of curriculum learning [4, 45] and gradually increase the forecasting length. Specifically, we optimize the network with N_{curri}^l curriculum learning epochs for $\{\mathbf{P}_{t_n}\}_{n=0}^{l}$, where $l = 1, 2, ...$ Because the observation nearer to current timestamp introduce more information about the current stage, we always reconstruct the current LiDAR point clouds and apply a decay weights $p(m)$ $(m = 1, 2, ..., l)$ to sample a future timestamp, where $p(m) > p(m+1)$ always holds. The final loss is computed as,

$$\mathcal{L} = \mathcal{L}_{t_0} + \mathcal{L}_{t_m},\ \ m \sim p(m). \tag{13}$$

Init.	mAP	NDS	Car	Truck	Bus	Barrier	Mot.	Bic.	Ped.	T.C.
Rand.*	21.09	27.59	57.66	16.28	14.42	34.42	8.99	0.43	46.58	29.90
TREND*	24.93 $^{+3.84}$	29.12 $^{+1.53}$	67.13	20.92	21.79	32.20	13.13	2.72	57.85	30.59
Rand.	31.06	44.75	69.18	28.73	34.57	42.31	13.72	8.72	69.18	41.14
4DOCC [18]	26.99	40.97	67.44	25.40	29.37	35.58	9.53	5.16	65.26	29.47
T-MAE [47]	30.53	44.55	68.63	26.02	34.66	43.98	13.21	7.26	68.78	39.82
UniPAD [54]	32.16 $^{+1.10}$	45.50 $^{+0.75}$	69.82	29.54	35.73	46.79	13.65	7.98	70.45	42.73
TREND	**33.17** $^{+2.11}$	**46.21** $^{+1.46}$	71.24	30.08	39.57	45.42	16.65	9.33	71.84	43.70

Table 1. Results for few shot fine-tuning on NuScenes [5] dataset. We randomly sample 175 frames of labeled point clouds in the training set and use Transfusion [2] as the downstream model for all the experiments here. Results of overall performance (mAP) and different categories (APs) are provided. "Init." indicates the initialization methods. "Rand*" means training from scratch with the original number of training iterations in OpenPCDet [38]. "Rand" indicates the results where we gradually increase training iterations for train-from-scratch model until convergence is observed. "TREND*" indicates pre-training with TREND and fine-tuning with the original iteration number in OpenPCDet [38]. Mot., Bic., Ped. and T.C. are abbreviations for Motorcycle, Bicycle, Pedestrian and Traffic Cone. We use green color to highlight the performance improvement brought by different initialization methods and bold fonts for best performance in mAP and NDS. All the results are in %.

4. Experiments

Unsupervised 3D representation learning aims to pre-train 3D backbones and use the pre-trained weights to initialize downstream models for performance improvement. In this section, we design experiments to demonstrate the effectiveness of the proposed method TREND as compared to previous methods. We start with introducing experiment settings in Section 4.1. Then main results are provided in Section 4.2. Finally, additional experiment results and ablation study are discussed in Section 4.3 and 4.4.

4.1. Experiment Settings

Datasets. We conduct experiments on three popular autonomous driving datasets including NuScenes [5], Once [24] and Waymo [36]. NuScenes uses a 32-beam LiDAR to collect 1000 scenes in Boston and Singapore, where 850 of them are used for training and the other 150 ones for validation. We use the whole training set without label for all the pre-training methods and few-shot fine-tuning conducted. We evaluate all the models on the whole validation set of NuScenes. Once utilizes a 40-beam LiDAR to collect 144-hour data with 1 million LiDAR point cloud frames and labels 15k of them. Due to the computation resource limitation, we conduct pre-training with TREND on the small split of the unlabeled data (100k frames) and fine-tune the pre-trained backbone with the labeled training set. Waymo equips the autonomous vehicle with one top 64-beam LiDAR and 4 corner LiDARs to collect point clouds in San Francisco, Phoenix, and Mountain View. We use Waymo for evaluating the transferring ability of TREND. We initialize model with weights pre-trained on Once and train it on Waymo in a few-shot setting to see whether pre-training with TREND on Once could bring improvement for downstream task on Waymo.

Downstream Detectors and Evaluation Metrics. We follow the implementations in the popular code repository for LiDAR-based 3D object detection called OpenPCDet [38] and select the SOTA detectors on different datasets. For NuScenes [5], we use Transfusion [2] as the downstream model. Average precisions for different categories (APs), mean average precision (mAP) and NuScenes Detection Score (NDS) [5] are used as evaluation metrics. For Once [24] and Waymo [36], we select CenterPoint [56] as the downstream detector. APs for different categories and mAP are used for evaluation in Once. As for Waymo, APs are computed at two difficulty levels (Level-1 and Level-2) and average precisions with heading (APHs) are utilized for evaluation. The main goal of unsupervised 3D pre-training is to improve ***sample efficiency instead of accelerating convergence***, which has been discussed in previous literature [11, 49]. Sample efficiency means the best performance we can achieve with the same model trained by the same number of labeled data. Thus, we first gradually increase the training iterations for randomly initialized models until convergence is observed. Here convergence means increasing number of training iterations does not further improve the performance. Then we fix the training iterations and use the same schedule for fine-tuning experiments with different pre-training methods.

Baseline 3D Pre-training Methods. We select three baseline methods. The first one is UniPAD [54], the masked-and-reconstruction-based method with rendering decoder. The second one is a LiDAR point cloud forecasting method called 4DOCC [18]. We train 4DOCC [18] with the back-

Init.	F.T.	mAP	Vehicle			Pedestrian			Cyclist		
			0-30m	30-50m	50m-	0-30m	30-50m	50m-	0-30m	30-50m	50m-
Rand*	5%	20.48	58.03	25.22	12.98	11.62	9.75	6.97	21.55	6.83	3.11
TREND*		29.95 +9.47	63.54	33.05	18.94	18.47	14.51	9.44	41.82	20.55	8.53
Rand		46.07	76.71	51.15	31.84	37.53	20.12	9.84	62.00	42.61	24.18
TREND		47.84 +1.77	79.14	55.68	36.34	35.23	18.00	11.18	64.99	45.80	28.15
Rand*	20%	54.65	80.53	57.79	39.62	48.30	36.30	19.65	68.10	51.02	32.90
TREND*		55.97 +1.42	83.34	63.10	44.99	47.56	34.27	18.95	68.60	53.02	34.64
Rand		57.68	82.70	63.37	46.34	52.61	36.48	19.03	71.03	55.34	36.34
TREND		58.93 +1.25	84.08	65.80	50.51	50.31	33.37	19.42	72.54	56.31	39.26
Rand*	100%	64.00	86.21	70.20	58.20	57.80	41.18	23.55	75.95	61.45	45.80
TREND*		64.79 +0.79	87.90	72.38	60.29	57.21	40.60	24.97	77.26	61.92	46.14
Rand		65.03	88.18	74.23	61.75	57.32	38.90	21.96	78.07	64.32	48.16
TREND		65.66 +0.63	88.81	74.63	61.98	57.94	39.85	20.95	79.98	64.18	47.62

Table 2. Results for fine-tuning on Once [24] dataset. We use CenterPoint [56] as the downstream detector. "Init." indicates the initialization methods. "F.T." is the ratio of sampled training data for fine-tuning stage. We show mAP for the overall performance and APs for different categories within different ranges. "Rand*" means training randomly initialized model with the original iteration number in OpenPCDet [38]. "Rand" indicates that we increase the training iterations for train-from-scratch model until convergence is observed. "TREND*" indicates pre-training with TREND and fine-tuning with the original iteration number in OpenPCDet [38]. "TREND" uses the same training iterations as "Rand". Green color is used to highlight the performance improvement brought by TREND. All the results are in %.

bone used in our experiments and then migrates the pre-trained encoder to downstream task. The third one is a concurrent work called T-MAE [47], which utilizes previous adjacent frame of LiDAR point clouds for masked-and-reconstruction without considering action of the autonomous vehicle. All the pre-trainings for [18, 47, 54] are conducted with the official code released with the papers.

Implementation Details of TREND. For f^{enc}, we select the backbones used in [2, 56]. The feature channels for embedded 3D feaures $\hat{\mathbf{P}}_{t_n}$, sinusoidal encoding and action embeddings are respectively set to $\hat{d} = 128$, $d_{\text{sin}} = 32$ and $d_{\text{act}} = 16$. The sampled ray number for rendering is $N_{\text{render}} = 12288$ and number of sampled points along each ray is $N_{\text{ray}} = 48$. For curriculum learning on forecasting length, we set the curriculum learning epoch as $N^1_{\text{curri}} = 12$ and $N^2_{\text{curri}} = 36$. We set the pre-training learning rate as 0.0002 with a cosine learning schedule and use mask augmentation for TREND with a masking rate of 0.9.

4.2. Main Results

Results on NuScenes Dataset. Both TREND and baseline methods are pre-trained on the whole training set of NuScenes dataset [5]. We then randomly select 175 frames of labeled LiDAR point clouds in the training set and conduct few-shot fine-tuning experiments. Results are shown in Table 1. It can be found that directly incorporate encoders from the LiDAR forecasting method 4DOCC [18] even degrades the performance, which might stems from that 4DOCC neglects action embeddings and uses a simply convolution-based decoder for point cloud forecasting. Our proposed method TREND achieves 2.11% mAP and 1.46% NDS improvement over randomly initialization at convergence, which is 91% more improvement for mAP and 94% more improvement for NDS than the previous SOTA unsupervised 3D representation method UniPAD [54]. T-MAE [47] only achieves comparable performance to train-from-scratch model at convergence. If we look into detailed categories, TREND achieves general improvement on all the categories. Specifically, for Car, Barrier, Motorcycle, Pedestrian and Traffic Cone, the improvement are more than 2% AP. And for Bus, TREND introduce an improvement of 5% AP.

Results on Once Dataset. We pre-train TREND and baseline methods on the small split of unlabeled data in Once [24] and fine-tune the pre-trained backbone with three settings 5%, 20% and 100% of the labeled training set. The results are shown in Table 2. It can be found that TREND improve the mAP at convergence by 1.77, 1.25 and 0.70 respectively for 5%, 20% and 100% fine-tuning data, which demonstrates TREND is able to improve downstream sample efficiency. We also provide results where the original number of training iterations in [38] is used for down-

Init.	Vehicle				$\bar{\Delta}$	Pedestrian				$\bar{\Delta}$	Cyclist				$\bar{\Delta}$
	Level-1		Level-2			Level-1		Level-2			Level-1		Level-2		
	AP	APH	AP	APH		AP	APH	AP	APH		AP	APH	AP	APH	
Rand.	66.84	66.23	58.84	58.30	**+1.17**	68.21	61.22	60.17	53.88	-0.10	49.76	48.28	47.86	46.43	**+1.16**
TREND	68.04	67.39	60.02	59.44		68.12	60.99	60.20	53.77		50.89	49.52	48.94	47.62	

Table 3. Results for transferring experiments. We utilize the weights pre-trained on Once [24] dataset to initialize CenterPoint [56] and train it with 1% training data in Waymo [36]. "Init." indicates the initialization methods. "Rand" means that we increase the training iterations for train-from-scratch model on Waymo until convergence is observed. "TREND" uses the same training iterations as "Rand" for fine-tuning. All the results are AP and APH in %. We compute the performance of TREND minus that of "Rand" and then average within each category, which results in $\bar{\Delta}$.

stream task, highlighted with $*$. In this setting, TREND improves training-from-scratch by up to 9.47% mAP, which greatly accelerate convergence. As for converged results on different categories, TREND achieves up to 4% mAP improvement on Vehicle and Cyclist for 5% fine-tuning data and generally improve these two categories within different ranges. However, it can also be found that for Pedestrian class, TREND degrades the performance a little bit under 5% and 20% fine-tuning data settings. We think this is because LiDAR point clouds stand for geometry and pedestrians are always captured in LiDAR point clouds with a cylinder-like shape, which is less-distinguishable as compared to cyclists and vehicle. For example, trash bins or poles on the road also appear to be a cylinder-like shape in LiDAR point clouds. Thus learning to reconstruct and forecast such less-distinguishable geometry harms the ability of the pre-trained backbone to identify pedestrians among similar cylinder-like shapes especially when there are less labeled downstream data, leading to a little degradation for 5% and 20% settings.

4.3. Transferring Experiments

We further use the backbone pre-trained on Once [24] to initialize CenterPoint [56] and fine-tune the detector with 1% training data of Waymo [36]. The converged results of both random initialization and pre-trained on Once are shown in Table 3. It can be found that for Vehicle and Cyclist, TREND brings an average improvement of 1.17 and 1.16 on APs and APHs, demonstrating that TREND is able to pre-train the backbone on one dataset and then transfer to another dataset for performance improvement. As for Pedestrian class, there exists similar phenomenon to that on Once fine-tuning where TREND only achieves comparable performance. The reason is similar to what we discuss in Section 4.2.

4.4. Ablation Study

We conduct ablation study to analyze the contribution of different parts of TREND. As shown in Table 4, it can be found that using neural field for reconstruction pre-training brings little improvement and even degrades the NDS score compared to training-from-scratch. Adding Recurrent Embedding scheme with neural field for reconstruction and forecasting improves the performance both on mAP and NDS, which demonstrates the effectiveness of Recurrent Embedding scheme to encode 3D features for different timestamps. Finally, with Temporal Neural Field, TREND achieves the best performance both on mAP and NDS, showing that Temporal Neural Field better utilizes the temporal information in LiDAR sequence for unsupervised 3D pre-training.

Rec. Emb.	N. F.	Temporal N. F.	mAP	NDS
✗	✗	✗	31.06	44.75
✗	✓	✗	32.16	45.26
✓	✓	✗	32.45	45.76
✓	✗	✓	33.17	46.21

Table 4. Results for ablation study. "Rec. Emb." is abbreviation for Recurrent Embedding. "N. F." and "Temporal N. F." are respectively for Neural Field and Temporal Neural Field. The first row is training-from-scratch. Then we add neural field for reconstruction pre-training, as shown in the second line. The third row is add recurrent embedding with original neural field and the last one for TREND.

5. Conclusion

In this paper, we propose TREND for unsupervised 3D representation learning via temporal forecasting. TREND is consisted of a Recurrent Embedding scheme to generate 3D embeddings for different timestamps and a Temporal Neural Field to represent the 3D scene across time, through which we conduct differentiable rendering for reconstructing and forecasting LiDAR point clouds. With extensive experiment on popular autonomous driving datasets, we demonstrate that TREND is superior in improving downstream performance compared to previous SOTA unsupervised 3D representation learning techniques. Additionally, TREND generally improves the performance on different downstream datasets with different 3D object detectors. We believe TREND will facilitate our understanding on 3D perception in autonomous driving.

References

[1] Ben Agro, Quinlan Sykora, Sergio Casas, Thomas Gilles, and Raquel Urtasun. Uno: Unsupervised occupancy fields for perception and forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14487–14496, 2024. 2, 3, 4

[2] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1090–1099, 2022. 1, 3, 6, 7, 2

[3] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009. 5

[5] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 2, 6, 7

[6] Tony Chan and Wei Zhu. Level set based shape prior segmentation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 1164–1170. IEEE, 2005. 4, 5

[7] Runjian Chen, Yao Mu, Runsen Xu, Wenqi Shao, Chenhan Jiang, Hang Xu, Zhenguo Li, and Ping Luo. Coˆ 3: Cooperative unsupervised 3d representation learning for autonomous driving. *arXiv preprint arXiv:2206.04028*, 2022. 2

[8] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 1, 2

[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. *arXiv preprint arXiv:2112.06375*, 2021. 1, 3

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 1, 2

[11] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4918–4927, 2019. 6

[12] Fangzhou Hong, Hui Zhou, Xinge Zhu, Hongsheng Li, and Ziwei Liu. Lidar-based panoptic segmentation via dynamic shifting network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13090–13099, 2021. 1

[13] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15587–15597, 2021. 2

[14] Di Huang, Sida Peng, Tong He, Honghui Yang, Xiaowei Zhou, and Wanli Ouyang. Ponder: Point cloud pre-training via neural rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16089–16098, 2023. 2, 3

[15] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6535–6545, 2021. 2

[16] Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18236–18246, 2023. 3, 5

[17] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021. 4, 5

[18] Tarasha Khurana, Peiyun Hu, David Held, and Deva Ramanan. Point cloud forecasting as a proxy for 4d occupancy forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1116–1124, 2023. 2, 3, 4, 6, 7

[19] 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/CVF International Conference on Computer Vision*, pages 3293–3302, 2021. 2

[20] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision*, pages 657–675. Springer, 2022. 2

[21] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 529–537, 2019. 3

[22] Yunze Liu, Li Yi, Shanghang Zhang, Qingnan Fan, Thomas Funkhouser, and Hao Dong. P4contrast: Contrastive learning with pairs of point-pixel pairs for rgb-d scene understanding. *arXiv preprint arXiv:2012.13089*, 2020. 2

[23] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995. 4, 5

[24] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, et al. One million scenes for autonomous driving: Once dataset. *arXiv preprint arXiv:2106.11037*, 2021. 2, 4, 6, 7, 8

[25] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8):1909–1963, 2023. 1

[26] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 3

[27] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3, 5

[28] Bo Pang, Hongchi Xia, and Cewu Lu. Unsupervised 3d point cloud representation learning by triangle constrained contrast for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5229–5239, 2023. 2

[29] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 2

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[31] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[32] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[33] 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, 3

[34] 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*, 43(8):2647–2664, 2020. 3

[35] 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. *arXiv preprint arXiv:2102.00463*, 2021. 1, 3

[36] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 2, 4, 6, 8

[37] Tang Tao, Longfei Gao, Guangrun Wang, Yixing Lao, Peng Chen, Hengshuang Zhao, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. *arXiv preprint arXiv:2304.10406*, 2023. 3, 5

[38] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 6, 7

[39] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 4, 5

[40] Sundar Vedula, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):475–480, 2005. 3

[41] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115:1–28, 2015. 3

[42] 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/CVF international conference on computer vision*, pages 9782–9792, 2021. 2

[43] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 2, 3, 4, 5

[44] Tai Wang, Conghui He, Zhe Wang, Jianping Shi, and Dahua Lin. Flava: Find, localize, adjust and verify to annotate lidar-based point clouds. In *Adjunct Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 31–33, 2020. 1

[45] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021. 5

[46] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 91–98, 2020. 3

[47] Weijie Wei, Fatemeh Karimi Nejadasl, Theo Gevers, and Martin R Oswald. T-mae: Temporal masked autoencoders for point cloud representation learning. *arXiv preprint arXiv:2312.10217*, 2023. 3, 6, 7

[48] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 574–591. Springer, 2020. 2

[49] Runsen Xu, Tai Wang, Wenwei Zhang, Runjian Chen, Jinkun Cao, Jiangmiao Pang, and Dahua Lin. Mv-jar: Masked voxel jigsaw and reconstruction for lidar-based self-supervised pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13445–13454, 2023. 2, 6

[50] Siming Yan, Zhenpei Yang, Haoxiang Li, Chen Song, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Implicit

autoencoder for point-cloud self-supervised representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14530–14542, 2023. 2, 3

[51] Xiangchao Yan, Runjian Chen, Bo Zhang, Jiakang Yuan, Xinyu Cai, Botian Shi, Wenqi Shao, Junchi Yan, Ping Luo, and Yu Qiao. Spot: Scalable 3d pre-training via occupancy prediction for autonomous driving. *arXiv preprint arXiv:2309.10527*, 2023. 2, 3

[52] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 3

[53] Honghui Yang, Tong He, Jiaheng Liu, Hua Chen, Boxi Wu, Binbin Lin, Xiaofei He, and Wanli Ouyang. Gd-mae: generative decoder for mae pre-training on lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9403–9414, 2023. 2

[54] Honghui Yang, Sha Zhang, Di Huang, Xiaoyang Wu, Haoyi Zhu, Tong He, Shixiang Tang, Hengshuang Zhao, Qibo Qiu, Binbin Lin, et al. Unipad: A universal pre-training paradigm for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15238–15250, 2024. 2, 6, 7

[55] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 2

[56] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 1, 3, 6, 7, 8

[57] 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/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022. 2

[58] Jiakang Yuan, Bo Zhang, Xiangchao Yan, Botian Shi, Tao Chen, Yikang Li, and Yu Qiao. Ad-pt: Autonomous driving pre-training with large-scale point cloud dataset. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[59] Lunjun Zhang, Yuwen Xiong, Ze Yang, Sergio Casas, Rui Hu, and Raquel Urtasun. Learning unsupervised world models for autonomous driving via discrete diffusion. *arXiv preprint arXiv:2311.01017*, 2023. 2, 3

[60] 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. *Advances in neural information processing systems*, 35:27061–27074, 2022. 2

[61] Ye Zhang and Chandra Kambhamettu. On 3d scene flow and structure estimation. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, pages II–II. IEEE, 2001. 3

[62] Zehan Zheng, Fan Lu, Weiyi Xue, Guang Chen, and Changjun Jiang. Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5145–5154, 2024. 3, 5

[63] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020. 1, 2

[64] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020. 1

[65] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chunhua Shen, Yu Qiao, et al. Ponderv2: Pave the way for 3d foundataion model with a universal pre-training paradigm. *arXiv preprint arXiv:2310.08586*, 2023. 2

TREND: Unsupervised 3D Representation Learning via Temporal Forecasting for LiDAR Perception

Supplementary Material

A. More Experiments on NuScenes

In this section, we conduct more fine-tuning experiments on NuScenes dataset. Specifically, we randomly sample 2.5% and 5% of NuScenes training set and train the randomly initialization model [2] until convergence is observed. Then we apply the pre-trained weight by TREND to initialize the model [2] and fine-tune it with the same training iterations. Results are shown in Table 5. It can be found that TREND consistently improve the performance in downstream 3D object detection task with different ratio of downstream training data.

B. LiDAR Segmentation

We further try to evaluate the effectiveness of TREND on LiDAR segmentation task. We use the pre-trained weights on Once to initialize Cylinder3D [63] and fine-tune it on SemanticKitti dataset [3, 10]. Note that in order to apply the pre-trained weights for Cylinder3D [63], we modify its encoder to match the pre-trained backbones and for other parts of the network, we utilize the implementation in MMDetection3D [8]. Mean Intersection over Union (mIoU) is used as the main evaluation metric, along with accuracy per category and overall accuracy. Results are shown in Table 6. It can be found that TREND is able to improve the performance by 2.89% in mIoU and 9.14% in overall accuracy, demonstrating the effectiveness of TREND on LiDAR semantic segmentation.

Init.	F.T.	mAP	NDS	CAR	Truck	Bus	Barrier	Mot.	Bic.	Ped.	T.C.
Rand.	2.5%	45.35	55.36	76.74	40.89	50.07	57.48	41.58	26.13	76.67	55.77
TREND		45.79$^{+0.64}$	56.23$^{+0.87}$	77.74	42.96	50.78	59.39	40.37	23.48	77.22	57.51
Rand	5%	51.56	60.24	80.22	48.56	58.69	63.42	50.84	36.59	79.29	60.30
TREND		52.02$^{+0.46}$	61.02$^{+0.78}$	80.54	48.15	57.93	63.57	52.59	36.92	79.99	60.94

Table 5. Results for few shot fine-tuning on NuScenes [5] dataset. We randomly sample 2.5% and 5% of labeled point clouds in the training set and use Transfusion [2] as the downstream model for all the experiments here. Results of overall performance (mAP) and different categories (APs) are provided. "Init." indicates the initialization methodshorthands. "Rand" indicates the results where we gradually increase training iterations for train-from-scratch model until convergence is observed. Mot., Bic., Ped. and T.C. are abbreviations for Motorcycle, Bicycle, Pedestrian and Traffic Cone. We use green color to highlight the performance improvement brought by different initialization methodshorthands and bold fonts for best performance in mAP and NDS. All the results are in %.

Init.	mIoU	Acc	Car	Bic.	Bus	Person	Building	Fence	Vegetation	Terrian
Rand.	28.23	70.68	79.01	27.40	9.34	15.57	42.89	13.32	56.26	57.78
TREND	31.12$^{+2.89}$	79.82$^{+9.14}$	89.75	35.49	11.62	19.66	73.62	17.70	78.65	53.35

Table 6. Fine-tuning experiments on Semantic Kitti [3, 10]. We modify the encoder part of Cylinder3D [63] and train it from scratch. Then we use the pre-trained weights on Once with TRENDto initialize the same network and fine-tune it. Training schedules are the same as that in [8] and we select the models with best mIoU performance.