Equivariant Spatio-Temporal Self-Supervision for LiDAR Object Detection

Deepti Hegde[1], Suhas Lohit[2], Kuan-Chuan Peng[2], Michael J. Jones[2], and Vishal M. Patel[1]

[1] Johns Hopkins University, Baltimore, MD 21218, USA
[2] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA
{dhegde1,vpatel36}@jhu.edu, {slohit,kpeng,mjones}@merl.com

Abstract. Popular representation learning methods encourage feature invariance under transformations applied at the input. However, in 3D perception tasks like object localization and segmentation, outputs are naturally equivariant to some transformations, such as rotation. Using pre-training loss functions that encourage *equivariance* of features under certain transformations provides a strong self-supervision signal while also retaining information of geometric relationships between transformed feature representations. This can enable improved performance in downstream tasks that are equivariant to such transformations. In this paper, we propose a spatio-temporal equivariant learning framework by considering both spatial and temporal augmentations jointly. Our experiments show that the best performance arises with a pre-training approach that encourages equivariance to translation, scaling, and flip, rotation and scene flow. For spatial augmentations, we find that depending on the transformation, either a contrastive objective or an equivariance-by-classification objective yields best results. To leverage real-world object deformations and motion, we consider sequential LiDAR scene pairs and develop a novel 3D scene flow-based equivariance objective that leads to improved performance overall. We show that our pre-training method for 3D object detection outperforms existing equivariant and invariant approaches in many settings.

Keywords: LiDAR · 3D object detection · Self-supervised learning

1 Introduction

Relying on fully-supervised training paradigms can be limiting as manual annotation is expensive. Interest in autonomous navigation and lowering cost of sensing hardware has enabled access to large amounts of LiDAR data [19, 26], a crucial source of depth information useful for perception tasks. However, the annotation of outdoor LiDAR point clouds is challenging due to their irregularities and sparsity. Self-supervised learning (SSL) enables learning of generic visual representations of unlabelled data by completing tasks designed based on human intuition about what information can be inferred from its inherent properties, without the need for explicit supervision. The availability of large amounts of

unlabelled LiDAR data thus makes SSL pre-training methods a natural choice for improving performance of perception tasks in limited label scenarios.

Many pretext tasks in state-of-the-art (SOTA) representation learning approaches encourage feature invariance under different views, transformations, and across time by training under a contrastive learning objective [3, 14, 31, 39]. These frameworks show which transformations can train the network to learn rich visual representations useful for downstream tasks [3, 30]. Methods like STRL [15] use a BYOL-like [11] framework to encourage **invariance** of the features over time. However, in tasks like object detection and semantic segmentation, input orientation is important information that should be retained in the feature representation. That is, if the LiDAR scene is rotated, all the object bounding boxes should also rotate by the same amount. Encouraging invariance to rotations conflicts with this task. In PointContrast [32], networks are trained to be equivariant to transformations such as rotation, scaling, and translation, variations over time are not considerd. In contrast, we propose a training scheme, which encourages learning features that transform in an **equivariant** fashion over spatial augmentations as well as over time. We also show improved performance for rotation equivariance via equivariance-via-classification strategy.

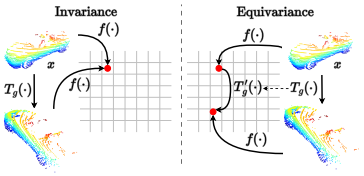

Fig. 1: An illustration of invariance (left) vs. equivariance (right), as described in Sec. 1. Invariance of f to group of transformations G means that the output representation does not change with the applied input transformation, whereas equivariance of f to $\mathcal{G}$ means the output representation changes by T'_g for some applied input transformation T_g. In this visualization, T_g is 3D rotation. Figure inspired by [7].

We can define invariance and equivariance to group action more formally. Let the inputs be denoted by $x \in \mathcal{X}$, $f(\cdot)$ be the encoder, and the output be $f(x)$. A group G is a set along with a binary operation $\circ$ that respects closure $(\forall g, g' \in G, g \circ g' \in G)$, associativity $(\forall g, g', g^* \in G, g \circ (g' \circ g^*) = (g \circ g') \circ g^*)$, has a unique identity element $e \in G$ and an inverse g^{-1} exists for each element $g \in G$ such that $g \circ g^{-1} = g^{-1} \circ g = e$. If G is a group of transformations, for $g \in G$, let $T_g(x)$ and $T'_g(x)$ be the group action on x and $f(x)$, respectively. The invariance of f to G means that the output representation does not change with the applied transformation, $\forall x,\ \forall g,\ f(T_g(x)) = f(x)$. By the equivariance of f to G, we mean that $\forall x,\ \forall g,\ f(T_g(x)) = T'_g(f(x))$, where T'_g is the same transformation acting on the features, see Fig. 1.

Dangovski *et al.* [5] show that certain transformations discarded for training for invariance can instead be leveraged to train for equivariance, but the downstream tasks they consider include only image classification, which is invariant to the transformations at the input. We are interested in better understanding the effect of, and improving methods for equivariant pre-training for LiDAR point clouds for object detection that inherently has an equivariant component

– bounding box regression should be equivariant to geometric transformations applied at the input.

Data augmentation methods are an important component for self-supervised learning and influence the nature of learned visual representations [30]. Existing augmentations for LiDAR scenes do not include realistic transformations that describe things such as ego-motion and geometric object deformations over time. For perceiving moving objects, relative motion becomes a useful property to localize objects. We expand the study of equivariant feature learning to more natural transformations by considering temporal sequences of LiDAR frames. We model the point-level transformation over time as a scene flow matrix. Scene flow naturally captures local transformations of objects through their motion. 3D scene understanding tasks should be equivariant to these local transformations. We thus include 3D scene flow as an additional transformation under which to train the network to be equivariant.

We present a study into **E**quivariance for **S**elf-**S**upervised **L**earning for **3D** perception tasks on LiDAR point clouds and propose the framework **E-SSL^{3D}**. We consider LiDAR scenes applied with a series of spatial and temporal augmentations to train a 3D feature encoder under a joint equivariant contrastive learning and flow equivariance objective. To encourage spatial equivariance, transformed views of a scene are contrasted at the point level as well as passed to a classification head to predict the applied geometric transformation. To encourage temporal equivariance, the network is trained to minimize the distance between sequential pairs of LiDAR frames in the voxel feature space, where the feature map of the first frame is warped to the second frame. Extensive experiments on 3D object detection in low-data regime show effectiness of **E-SSL^{3D}**. We summarize our contributions below:

1. We propose **E-SSL^{3D}**, a self-supervised pre-training method for LiDAR scenes that trains a network to learn spatio-temporal equivariance through a joint loss objective. We are the first to leverage 3D scene flow to encourage equivariance to temporal augmentations in LiDAR scenes.
2. We show that our pre-training strategy is effective in improving performance on downstream tasks, particularly in low-data scenarios. An object detector pre-trained using **E-SSL^{3D}** and fine-tuned on just 20% of data can achieve comparable performance to a network trained from scratch on 100% data.
3. We show improved performance for rotation equivariance over standard contrastive approaches through an equivariance-via-classification strategy.

2 Related work

2.1 Equivariant self-supervision

In recent years, there has been a growing interest in exploring the role of equivariance in learning visual representations in a self-supervised manner [1, 5, 7, 9, 12, 31]. Dangovski *et al.* generalize the standard contrastive SSL framework to a equivariant SSL framework and improve the existing performance of purely

invariant SSL methods on the tasks such as image classification and regression problems in photonics. In this work, we follow the intuition of probing for complementary augmentations, but apply equivariant SSL to more complex downstream tasks that are inherently equivariant, such as object detection. In [9], Garrido *et al*. propose a benchmark for evaluating equivariance on both inherently equivariant tasks such as 3D rotation prediction, as well as invariant ones such as classification, and present a method of splitting representations into invariant and equivariant parts. CARE [12] learns to translate augmentations such as cropping into linear transformations in a spherical feature space. The above approaches evaluate their frameworks on the tasks that generally benefit from invariance, such as classification or on the benchmarks specifically designed to evaluate learned equivariance. However, little investigation has been made into the pretext tasks that encourage equivariance for improving downstream tasks for 3D scene understanding. Xiong *et al*. propose FlowE [33], an SSL framework for image segmentation and object detection that is a variation of BYOL [11], where they introduce a flow equivariance objective by applying the flow transformation to a reference video frame, thus covering natural deformations, but this method is only applied to image video sequences. To the best of our knowledge, we are the first to explore the role of equivariance to both spatial and temporal, synthetic and natural transformations for 3D scene understanding with LiDAR.

2.2 Self-supervision for point cloud scenes

Self-supervised pre-training shows promise for learning useful representations from unlabeled point cloud scenes, both indoors and outdoors. Xie *et al*. propose PointContrast [32], which uses point-level contrastive training across partial transformed views of indoor scenes. This objective encourages equivariance to rigid transformations. In this work, we explore more effective ways of learning equivariant features under the more natural augmentation of 3D scene flow, as well as including an equivariance-via-classification strategy. Several follow-up works such as SegContrast [20] and DepthContrast [40] contrast point-level and segment-level features of transformed point clouds. The methods specifically tailored to outdoor LiDAR point clouds leverage contrastive learning, occupancy prediction, and point cloud completion to learn meaningful representations. TARL [21] learns temporally consistent feature representations by associating objects across time and maximizing their feature similarity. This approach depends on indexing and spatial clustering to compute object correspondence, which can be a limiting factor when considering LiDAR sequences captured with a high frame rate. ALSO [2] employs surface reconstruction as an auxiliary task to improve downstream performance in object detection and semantic segmentation. This pre-training strategy is specific to the downstream task and network, and thus lacks general applicability. Our pre-training framework is unified for all downstream networks that share a 3D feature backbone. ProposalContrast [39] considers region-level features obtained through a spatial clustering approach to enforce feature similarities between transformed objects.

However, the success of their method depends on the quality of unsupervised region clustering, which becomes difficult for sparse LiDAR scenes.

The above approaches perform self-supervised learning using convolutional feature backbones. Recent approaches propose the pretext tasks for representation learning to improve the performance of transformer-based 3D object detection networks. Yang *et al*. propose GD-MAE [37], a generative approach based on the masked autoencoder (MAE) [13], which uses hierarchical fusion to infer information from masked voxels. MV-JAR [34] employs both masked reconstruction and voxel position estimation in the form of a classification objective to perform self-supervised learning on LiDAR scenes. These approaches are effective as pre-training strategies for transformer-based detection networks such as SST [8], however cannot be applied to sparse convolutional backbones. We provide a more general solution for representation learning for LiDAR scenes, and focus on feature extractor backbones that are sparse convolution based, as the same backbone may then be used for a large number of detection [6, 17, 22, 24, 25, 36] and segmentation [4, 27, 41] networks.

2.3 3D object detection

The neural networks that perform 3D object detection on LiDAR scenes process point clouds as points [24, 38], as a set of 3D grids known as voxels [6, 17, 36], or a combination of the two [22, 23]. Single-stage networks [36, 38] directly estimate bounding box dimensions and predict category labels whereas two-stage networks include an additional bounding box refinement head [6, 24]. The popular detection network SECOND [36] is a single-stage detector consisting of a 3D sparse convolution backbone and a 2D convolutional layer following a Bird's-Eye View (BEV) compression step. The two-stage detector VoxelRCNN [6] shares a similar architecture but has an additional region refinement head and proposes a novel region pooling approach. Despite differences in architectures many sparse-convolutional 3D detectors share a common 3D feature extractor, which is advantageous for pre-training. Recently, attempts have been made to move away from the sparse convolutional approaches that operate on voxels and move towards transformer-based backbones [8], but these methods have high memory requirements. In this work, we focus on the detectors with convolutional backbones due to the broader applicability to downstream tasks. A single pre-trained 3D backbone is applicable to multiple detectors.

3 Proposed method: E-SSL^{3D}

Our goal is to train a network to be equivariant to certain spatial and temporal transformations in order to learn meaningful geometric representations that aid the network in performing dense scene prediction tasks. **E-SSL^{3D}** trains a feature encoder to learn dense voxel-level features that reflect the natural deformations that arise from object motion and that are equivariant to rigid transformations as well as naturally occurring temporal augmentations. This is

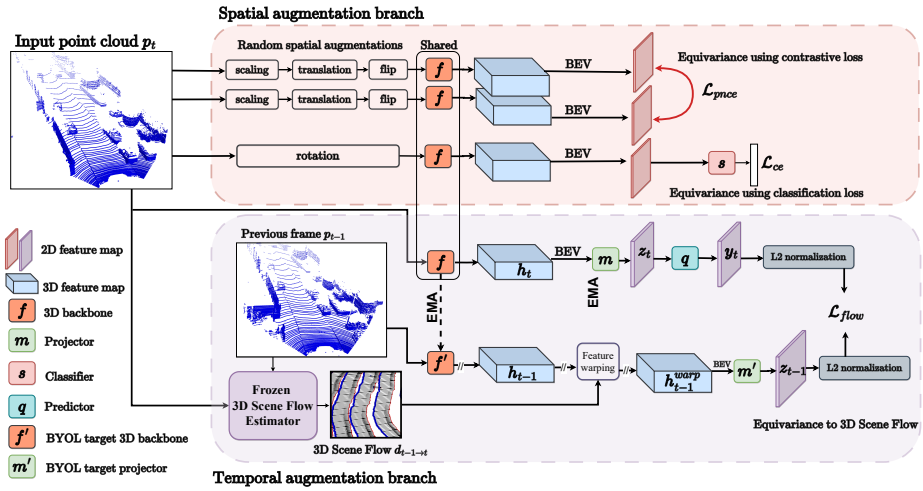

Fig. 2: E-SSL^{3D}: A LiDAR point cloud undergoes spatial and temporal augmentations before being input to the network that consists of a 3D feature extraction backbone f, a projector network m, a predictor network q, and a classifier s. (f, m, q) form the online branch and the copies (f', m') form the target network that is only updated through an exponential moving average (EMA) of the weights of the online network.

done by a joint training procedure that consists of learning equivariant features for both spatial augmentations and temporal changes via computing 3D scene flow. Our overall framework is summarized in Fig. 2.

Consider an input point cloud p from a sequence of LiDAR scenes. The network that is being trained using self-supervision can be divided into four parts – (a) a 3D feature encoder f that is a point-based or a voxel-based neural network that maps the input point cloud to a set of 3D features which are then mapped into 2D BEV features, (b) a projector m that maps the BEV features into a lower feature dimension, (c) a predictor q that matches the output of the projector to a target feature map, and (d) an augmentation classifier s.

3.1 Equivariance to global rigid spatial augmentations

We focus on common global rigid geometric transformations that are invertible – rotations, translations, scaling, and flips. We train the network to be equivariant to a group of spatial transformations in two ways, through a contrastive training objective and through an augmentation prediction objective.

Let the set of rigid transformations be $\mathcal{A}$. During training, for each iteration and each input point cloud, two instances from $\mathcal{A}$ are sampled randomly and applied to the input point cloud. We apply the contrastive loss PointInfoNCE [32] at the point level, which encourages invariance to the applied rigid transformations through a point-level contrastive objective between matched points in a

scene. Considering the two augmented views, Let $\mathcal{P}^+$ be the set of all positive matched points from both views. For a matched pair $(i, j) \in \mathcal{P}^+$:

$$\mathcal{L}_{pnce} = -\sum_{(i,j)\in\mathcal{P}_+} \log \frac{\exp((\mathbf{x}_i \cdot \mathbf{x}_j)/\tau)}{\sum_{(\cdot,k)\in\mathcal{P}_+} \exp((\mathbf{x}_i \cdot \mathbf{x}_k)/\tau)}, \tag{1}$$

where $(\mathbf{x}_i, \mathbf{x}_j)$ are the point features for each scene and τ is the scaling parameter.

Inspired by [5], we consider a uniform discrete subset of rigid transformations to further encourage equivariance through a classification objective. For example, for planar rotations, we consider 10 randomly sampled rotations. The features extracted for these augmentations are passed through an additional classifier layer s that is trained to predict the transformation applied at the input. The parameters of f, g, and s are trained by minimizing the cross-entropy loss $\mathcal{L}_{ce}$ between the input transformation and the predicted transformation. Thus, the network is trained to retain the information of the transformation that was applied at the input, *i.e.*, the features are equivariant to the input transformation. Note that the cross-entropy loss is applied in addition to the contrastive loss.

3.2 Equivariance to temporal changes via 3D scene flow estimation and feature warping

While spatial augmentations help in improving the final downstream performance, the augmentations are not realistic, in general. In addition to spatial augmentations, we exploit real frames that are captured sequentially to provide additional self-supervision. The key insight for temporal self-supervision is that the features of the network should evolve in a manner equivariant to how the points move in the real world. We term this 3D scene flow equivariance, where the 3D scene flow is the vector field that describes the motion of points in a point cloud frame at some time instance to locations in a different time instance. The 3D scene flow is therefore a locally varying transformation computed between two real frames and the features learned by the network are trained to respect the 3D scene flow constraint.

Estimating 3D scene flow. Given the point cloud p, we consider its natural temporal augmentation by taking the previous frame in the sequence. For the purpose of illustrating scene flow, let the previous frame be denoted at p_{t-1} and p_t be the current frame. We model the temporal transformation between scenes as 3D scene flow, represented as per-point displacement, $d_{t-1\to t} \in \mathbb{R}^3$ for each point in the point clouds. We denote the forward time transformation operation as $\mathcal{F}_{t-1\to t}$. Thus, we can represent the relationship between p_t and p_{t-1} as: $p_t = \mathcal{F}_{t-1\to t}(p_{t-1})$. $d_{t-1\to t}$ is estimated from a frozen 3D scene flow estimation network based on PV-RAFT [29] trained on synthetic data and adapted to real-world LiDAR data in an unsupervised manner through a student-teacher framework [16].

Learning flow equivariance. We build the flow equivariance component on BYOL [11]. Under this SSL framework, a set of two networks – online and target – are trained to minimize the distance between predicted feature maps. Given

an augmented view, the online network is trained to predict the representation of the scene from the target network under a different augmentation. We choose this framework as an alternative to the contrastive learning approach since in the case of natural augmentations such as object motion across time, there is no useful discriminative feature to be learned by considering "negative" samples. This strategy for training for flow equivariance has seen success for images [33]. The information of the architectures of the online and target networks is in Sec. 4.1. An illustration of the online-target architecture can be seen in Fig. 2.

The 3D feature backbone f, the projector m, and the feature predictor q are considered as the online network, while the copies f' and m' are considered as the target network. Note that f is shared with the spatial augmentations branch. The role of m is to create more general representations for ease of adaptation of f to downstream tasks. The feature predictor matches the representation from the online network to that of the target network. The online network is updated with the standard training process while the weights of the target network are updated only through an exponential moving average (EMA) of the online network. This prevents representation collapse during the prediction step. Let the parameters of the online network as a whole be denoted as Θ, and that of the target network be Ψ. The EMA weight update step can be written as $\Psi \leftarrow \gamma\Psi + (1-\gamma)\Theta$, where $\gamma \in [0,1]$ is the target decay rate. γ is updated at every training step using the formula $\gamma \triangleq 1 - (1-\gamma_{\text{base}}) \cdot (\cos(\pi k/K) + 1)/2$, where k is the current training step and K is the maximum number of training steps, with γ_{base} being the initial target decay rate.

The point cloud scene pair $(p_{t-1},\ p_t)$ is fed to the target network and online network, respectively. We get the 3D voxel feature representations h_{t-1}, h_t by

$$h_t = f(p_t),\ h_{t-1} = f'(p_{t-1}). \tag{2}$$

h_{t-1} is warped to the future feature frame h_{t-1}^{warp} using the 3D scene flow estimate in a process detailed in the next section. After BEV compression, the features $(h_{t-1}^{warp},\ h_t)$ are passed to the projectors m' and m respectively to give $(z_{t-1},\ z_t)$. The projected feature map z_t is the input to feature predictor q to give y_t which is matched to the output of the target network z_{t-1}.

Warping with 3D scene flow. At the input, the point cloud p_{t-1} can be transformed to the next scene, point cloud p_t, by simply adding the per-point displacement from the scene-flow matrix, assuming index correspondence is maintained, which holds for the datasets used [21]. Let this warped estimate at the input be p_t^{warp}. Performing the same transformation in the feature space is not as straightforward, as we deal with the 3D sparse spatial feature maps $(h_{t-1},\ h_t)$ instead of sets of points. The 3D feature maps are represented as sparse tensors, which consist of voxel features and their corresponding 3D coordinates denoting their position in the feature volume. From the flow estimate $p_{1\to 2}$, we estimate the voxel coordinates of the points of the estimated future frame by applying the standard input voxelization process on p_t^{warp}. These estimated future voxel coordinates are then used to sample voxel features from h_{t-1} to get the warped feature estimate h_{t-1}^{warp}.

Loss function. The loss function minimizes of the $L2$ distance between the normalized z_{t-1} and y_t averaged over the spatial dimensions, defined as:

$$\mathcal{L}_{flow} = \frac{1}{HW} \left\| \hat{z}_{t-1} - \hat{y}_t \right\|_2^2, \tag{3}$$

where HW is the spatial dimension. $(\hat{z}_{t-1},\ \hat{y}_t)$ are the normalized feature maps.

The final loss function used to train the network is the sum of the contrastive loss and $L2$ distance loss, both of which are brought to similar scales through loss coefficients λ_{pnce}, λ_{ce} and λ_{flow} (*i.e.*, we set the loss coefficients such that all the loss terms have comparable ranges). The final loss is written as:

$$\mathcal{L} = \lambda_{pnce}\mathcal{L}_{pnce} + \lambda_{ce}\mathcal{L}_{ce} + \lambda_{flow}\mathcal{L}_{flow}. \tag{4}$$

4 Experiments

4.1 Pre-training

Datasets. We use the KITTI-360 [18] and the Waymo Open Dataset (WOD) [26] datasets for pre-training. KITTI-360 consists of 100k LiDAR scenes from 11 sequences captured in urban roads. The WOD consists of 230k LiDAR scenes from 1150 scenes, of which we use 100k for pre-training. For pre-training, we remove the validation sequences. To mitigate the distribution gap between the pre-training and fine-tuning datasets, we consider the front field-of-view (FFOV) scenes during pre-training.

Augmentation. As established, we choose the spatial augmentations that are invertible, namely global rotation about the vertical axis with an angle in the range $(\frac{-\pi}{2},\ \frac{\pi}{2})$, global translation in the $(x,\ y,\ z)$ axes with the displacement range $(0m,\ 0.2m)$, global scaling with magnitudes falling in the range $(0.95,\ 1.05)$, and random vertical flip with a probability of 50%. We choose to train the network to be equivariant to rotations, based on the experiments in Sec. 4.3. For ease of prediction, we sample from a discrete set of 10 rotation angles. For the temporal augmentation, we sample the previous frame from the sequence of LiDAR frames. KITTI-360 consists of LiDAR sequences that capture around 1.2 frames for every meter, with a 10 meter overlap between consecutive frames. SemanticKITTI consists of sequences that capture 10 frames per second.

Network architecture. In this section, we detail the network architectures of each module of the the proposed framework.

3D feature encoder (f): We perform pre-training on the sparse convolutional feature backbone, SparseVoxel, popular among recent 3D object detection networks [6, 17, 36].

Projector (m): We perform feature projection on the 2D BEV compressions of 3D volumetric feature maps. The BEV feature maps are obtained by max-pooling the densified sparse tensor along the height dimension. The feature projector is a 3-layer 2D convolutional network with batch normalization and ReLU layers. It maintains the spatial dimensions of the BEV feature map while reducing the channel dimension from 256 to 128.

Classifier (s): The classification branch of the network predicts the applied n-fold transformation. This is a simple 3-layer fully connected network with 2 batch normalization layers. In practice, we choose $n = 10$.
Predictor (q): The predictor network of the online branch consists of a single 1×1 convolutional layer to maintain the spatial dimensions.
Implementation details. We use the AdamW optimizer with a cyclic learning rate schedule, with the maximum learning rate 10^{-4}, a weight decay of 0.01, and momentum 0.9. The network is trained for 80 epochs with a batch size of 56 split over 8 NVIDIA A6000 GPUs. We use the codebase OpenPCDet [28] for the implementation of the 3D encoder and BEV projection modules. We follow [11] for the hyperparameters and EMA update rules of the online-target networks, with an initial $\gamma_{base} = 0.999$. We use the implementation of PointContrast for LiDAR scenes from the 3DTrans codebase [35]. Here, point features are sampled from the multi-scale 3D features and the compressed BEV feature map. We modify the computation of the point features to include the output after projection. We sample 2048 points for both the positive and negative samples. We set $\lambda_{pnce} = 0.01$, $\lambda_{flow} = 300$, $\lambda_{ce} = 1$.

4.2 3D Object detection

We demonstrate the effectiveness of our pre-training strategy for 3D object detection using the two detectors SECOND [36] and VoxelRCNN [6]. We fine-tune VoxelRCNN on the KITTI object detection dataset [10] and the Waymo Open Dataset [26] under the standard training and validation splits, and perform evaluation using the official metrics. In the case of fine-tuning on KITTI, we perform fine-tuning in 3 data availability scenarios, 5%, 20%, and 100% of data. In the case of fine-tuning on Waymo, we demonstrate the performance on 5 % of data. To account for class bias, we perform subset sampling thrice for each split and report the average performance. We fine-tune the SECOND detector on the KITTI dataset. See the supplement for results on SECOND and the performance of VoxelRCNN pre-trained and fine-tuned on the Waymo Open Dataset.
Network architectures. SECOND [36] is a single-stage detector that consists of a sparse convolution 3D encoder, a BEV encoder, and a region proposal network. VoxelRCNN [6] is a two-stage network that shares a similar 3D backbone and 2D encoder but includes an additional proposal refinement head.
Datasets and metrics. We fine-tune the object detection networks on the KITTI object detection dataset [10], which consists of 3712 training samples and 3769 validation samples. We perform evaluation under the standard protocol detailed in [10] on three difficulty categories and report performance on the "Car," "Pedestrian," and "Cyclist" categories as well as the mean average precision. Precision is calculated under 40 recall positions, as is followed in [2]. We use the standard division of objects into their respective difficulties based on their truncation, occlusion, and distance from the camera [10, 26]. Additionally, we fine-tune VoxelRCNN on the Waymo object detection dataset, and demonstrate results on the "Cyclist" category.

Table 1: 3D object detection with VoxelRCNN [6] pre-trained on the Waymo Open Dataset [26] and fine-tuned on KITTI [10] under different data splits. Each result is an average over 3 fixed subsets of the dataset. We report 3D average precision for 3 categories as well as the mean average precision over 40 recall positions. The best and second best performance is marked in **bold** and underline, respectively.

Split	Method	average precision (AP) (%)									mAP (%)
		Car			Pedestrian			Cyclist			
		easy	moderate	hard	easy	moderate	hard	easy	moderate	hard	
5%	No pre-training	88.89	**79.21**	**75.55**	**57.50**	**49.84**	**44.27**	78.92	59.73	55.97	65.54
	PointContrast	88.25	76.30	71.65	51.90	44.37	40.01	80.67	60.60	56.54	63.37
	STRL	**89.15**	77.29	73.73	56.04	49.13	43.59	83.55	**63.81**	**59.61**	**66.21**
	E-SSL^{3D}	89.13	77.33	73.84	**56.06**	48.87	43.70	**83.57**	63.28	59.12	66.10
20%	No pre-training	**91.99**	**82.10**	**79.40**	56.09	49.29	44.26	85.24	**67.55**	**63.13**	68.78
	PointContrast	91.74	80.47	77.35	**59.30**	51.05	45.90	85.97	65.70	61.25	68.75
	STRL	91.95	81.04	77.89	58.25	50.53	45.37	85.36	66.24	62.00	68.74
	E-SSL^{3D}	91.74	80.46	77.27	59.26	**51.82**	**46.65**	**86.51**	67.44	62.86	**69.33**
100%	No pre-training	**92.45**	**83.00**	**80.20**	**62.41**	**55.89**	**50.31**	88.40	68.81	64.42	**71.77**
	PointContrast	91.61	82.26	79.76	55.47	48.06	43.28	**89.68**	**71.90**	**67.57**	69.95
	STRL	91.86	82.29	79.80	59.65	51.82	46.23	87.28	70.49	65.79	70.58
	E-SSL^{3D}	92.16	82.16	79.77	59.14	50.45	45.04	88.68	71.17	66.44	70.56

Implementation details. We use the AdamW optimizer with a cyclic learning rate schedule, with the maximum learning rate 3×10^{-3}. We use a weight decay of 0.01, and momentum 0.9. We use the codebase OpenPCDet [28] for the implementation of the detection networks. Each network is fine-tuned for 80 epochs with a batch size of 8 over 2 NVIDIA A6000 GPUs. The temperature parameter τ in Eq. (1) is $\tau = 1$. The value of the target decay rate for the exponential moving averaging is $\gamma_{base} = 0.9996$.

Comparative methods. We compare the performance of **E-SSL^{3D}** with the recent SOTA SSL methods for LiDAR scenes:

- **PointContrast** [32] encourages point-level equivariance to different transformed views. We perform spatial augmentations to create view pairs and implement the adapted version for LiDAR scenes that samples point-level features from multi-scale 3D and BEV features. We sample 2048 points.
- **STRL** [15] encourages feature invariance across synthetically created temporal sequences of point cloud scenes by minimizing the $L2$ distance between samples passed through a BYOL-like online-target network. This becomes an invariant counterpart to our temporal equivariance component, and we re-implement this method by training the online-target networks with sequential LiDAR scene pairs.
- **ALSO** [2] uses occupancy prediction as a pretext task. We use the model pre-trained on KITTI-360 for the SECOND detector, and reproduce the fine-tuning result to the best of our ability. We note that our reproduction is slightly lower than reported. We note that this is a generative representation learning approach that differs fundamentally from our discriminative one.

Table 2: 3D object detection with VoxelRCNN [6] pre-trained on KITTI-360 [18] and fine-tuned on KITTI [10] under different data splits. Each result is an average over 3 fixed subsets of the dataset. We report 3D average precision for 3 categories as well as the mean average precision over 40 recall positions. The best and second best performance is marked in **bold** and underline, respectively.

Split	Method	average precision (AP) (%)									mAP (%)
		Car			Pedestrian			Cyclist			
		easy	moderate	hard	easy	moderate	hard	easy	moderate	hard	
5%	No pre-training	88.89	79.21	75.55	**57.50**	**49.84**	44.27	78.92	59.73	55.97	65.54
	PointContrast	**89.94**	79.21	**76.12**	56.13	48.13	43.01	77.98	58.92	55.20	64.96
	STRL	89.30	78.92	75.94	55.68	48.13	42.73	73.98	56.85	53.26	63.87
	ALSO	89.74	**79.37**	75.91	56.33	49.79	**44.77**	82.84	64.09	60.16	**67.00**
	E-SSL^{3D}	88.79	78.93	75.41	56.02	48.55	43.19	**82.85**	**64.40**	**60.53**	66.52
20%	No pre-training	91.99	82.10	79.40	56.09	49.29	44.26	85.24	67.55	63.13	68.78
	PointContrast	92.23	82.25	79.57	57.33	50.74	45.43	84.16	66.74	62.28	68.97
	STRL	91.97	82.07	79.41	57.40	50.85	45.38	86.36	68.64	64.23	69.59
	ALSO	92.46	**82.44**	79.77	60.57	53.21	48.61	86.22	69.88	65.40	70.95
	E-SSL^{3D}	**92.67**	82.42	**79.89**	**60.72**	**53.94**	**49.19**	**88.04**	**71.40**	**66.36**	**71.63**
100%	No pre-training	92.45	**83.00**	80.20	**62.41**	**55.89**	**50.31**	88.40	68.81	64.42	71.77
	PointContrast	91.73	82.41	79.89	59.82	54.14	48.54	87.28	69.15	63.54	70.72
	STRL	92.27	82.54	79.99	61.38	54.01	48.31	86.95	67.64	63.31	70.71
	ALSO	**92.57**	82.88	**80.24**	60.10	52.12	46.76	90.71	**73.94**	69.21	72.06
	E-SSL^{3D}	92.08	82.73	80.18	61.00	53.82	48.58	**91.15**	72.68	**69.32**	**72.41**

Quantitative results. We evaluate our pre-training framework for object detection on two networks SECOND [36] and VoxelRCNN [6]. Please see the supplement for quantitative results on SECOND. These detectors share a common sparse convolutional 3D feature extraction backbone and are initialized with the same model pre-trained on KITTI-360. We compare these fine-tuning results against PointContrast, STRL, and ALSO, as well as fine-tuning from a random weight initialization, denoted as "No pre-training." In Table 2, we demonstrate performance on the detector VoxelRCNN [6]. We perform best or second best in most categories. Overall, we outperform both PointContrast and STRL in general, showing that joint spatio-temporal equivariance is a good self-supervision signal for 3D object detection. We perform on-par with the recent state-of-the-art method ALSO. We note that for SECOND, ALSO's pre-training strategy trains both the 3D feature backbone as well as the 2D convolutional layers of the detection network, leaving only the classification and regression box prediction layers to be randomly initialized. On the other hand, we train only the 3D backbone and leave the rest of the network to be randomly initialized. Additionally, our method converges much more quickly than ALSO, which is trained for 75 epochs, whereas our approach converges at around 10-20 epochs. Importantly, that these two approaches use different types of self-supervised learning techniques – ALSO uses a generative strategy while **E-SSL^{3D}** uses specially designed loss functions for representation learning.

Ablation study. We conduct an ablation study on the spatial and temporal equivariance constraints and evaluate on the task of training VoxelRCNN [6] on

5% of KITTI data. We show the results in Table 3 where we report the 3D mean average precision for all the three object categories with 40 recall positions. By spatial equivariance, we mean training the network to be equivariant to the n-fold rotations using the cross-entropy loss and equivariant to random flips, scaling, and translations using the contrastive objective. By temporal equivariance, we mean training the network to be only be equivariant to 3D scene flow. Table 3 shows that enforcing both the spatial and temporal equivariance constraints performs the best overall and that both equivariance constraints contribute to the performance. We find that for the pedestrian class, pre-training with both objectives is not beneficial, but overall proves to be the better pre-training strategy.

4.3 Choice of loss function for equivariant pre-training

We compare the efficacy of the learning equivariance through contrastive training versus equivariance-via-classification as a pretext task for 3D object detection on KITTI with SECOND [36]. We test the effect of replacing the contrastive objective for the classification objective for individual transformations in Fig. 3. Specifically, we use PointContrast [32] with a single "random flip" augmentation as the baseline contrastive pretext task. The baseline performance under PointContrast is indicated by the gray dotted line. We encourage the network to be equivariant to three types of rigid transformations using the two learning objectives. For each additional transformation, we train the network to either predict the transformation in addition to the baseline or only train the network under PointInfoNCE loss. For random rotation and scaling, prediction is a 10-class classification problem. For translation, we predict the translation along each axes using a multi-label multi-class loss objective. We observe that encouraging equivariance to global translation along each axis as well as to scaling purely through the contrastive objective improves the performance. On the other hand, training for equivariance to rotation using the classification objective boosts performance relative to using the PointInfoNCE loss. The right choice of loss function depends on the nature of the augmentation. The standard ranges for translation augmentation for LiDAR object detectors is $(0m, 0.2m)$ along each axis. Considering that the range of the KITTI dataset reaches $70m$, this is a difficult fine-grained prediction task. The range of the scaling transformation is similarly small, $(0.95, 1.05)$. For n-fold rotations, the scene is rotated along the vertical axis by an angle ranging from $(\frac{-\pi}{2}, \frac{\pi}{2})$, a much larger range resulting

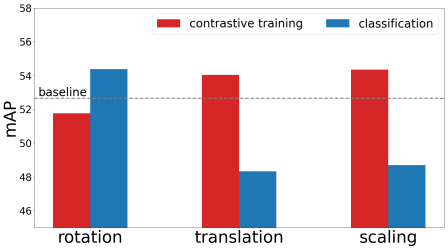

Fig. 3: 3D mean average precision of SECOND [36] pre-trained for equivariance for random spatial augmentations flip, rotation, translation, and scaling using the contrastive and classification objectives. The baseline network is pre-trained to be equivariant to "random flips" with contrastive learning.

Table 3: The ablation study of the spatial and temporal equivariance evaluated on the task of object detection with VoxelRCNN [6]. The reported numbers are 3D mean average precision (%) for the "Car," "Pedestrian," and "Cyclist" categories for the 3 difficulty levels and 40 recall positions.

Spatial equivariance	Temporal equivariace	average precision (AP) (%)									mAP(%)
		Car			Pedestrian			Cyclist			
		easy	moderate	hard	easy	moderate	hard	easy	moderate	hard	
✗	✗	88.68	78.85	74.36	56.30	49.13	43.33	76.48	58.62	54.79	64.50
✗	✓	**88.98**	77.80	73.81	56.53	49.73	44.61	81.50	61.74	57.67	65.82
✓	✗	87.12	77.34	74.63	**58.66**	**50.34**	**45.19**	81.09	61.71	58.00	66.01
✓	✓	88.79	**78.93**	**75.41**	56.02	48.55	43.19	**82.85**	**64.40**	**60.53**	**66.52**

in more distinct augmentations. We show that training the network to predict n-fold rotations while training the network under the point-level contrastive loss for the scaling, flip, and translation augmentations is a good strategy.

4.4 Limitations

We acknowledge certain limitations in our proposed self supervised learning framework. We observe that when fine-tuning on the KITTI dataset, self supervised pre-training does not always boost performance for the "Car" category. We believe this is due to the fact that this category is well represented in the dataset, and is a relatively "easier" object to detect. Additionally, **E-SSL$^{\mathbf{3D}}$** does not always outperform the SOTA approach ALSO , and performs second-best for certain categories. However, we point out that ALSO is a generative approach, which is not directly comparable to our method, which also may be integrated with our discriminative method, which we hope to explore in the future. We also observe that when plenty of annotated samples are available (*e.g.*, 100% of data in Table 2), pre-training does not have a large impact on performance, and we emphasize that **E-SSL$^{\mathbf{3D}}$** is most helpful in low-data scenarios, and it achieves close to full-data performance with just 20% of annotated training data in the case of KITTI object detection dataset.

5 Conclusion

In this work, we examine the role of equivariance in representation learning for large scale outdoor point clouds. We present **E-SSL$^{\mathbf{3D}}$**, a self supervised learning method for 3D object detection on LiDAR scenes that learns meaningful geometric representation by encouraging joint spatial and temporal equivariance. We developed a novel 3D scene flow equivariance objective to incorporate temporal information for improved representation learning. We showed that the choice of equivariance objective affects the final performance significantly depending on the type of augmentation applied. Our experiments demonstrate the usefulness of our learned representations and suggest that for certain transformations, it is helpful to encourage equivariance through augmentation classification.

Acknowledgements

For this work, Deepti Hegde, Suhas Lohit, Kuan-Chuan Peng, and Michael J. Jones were supported by Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA. Vishal M. Patel was partially supported by NSF CAREER award 2045489.

References

1. Bhardwaj, S., McClinton, W., Wang, T., Lajoie, G., Sun, C., Isola, P., Krishnan, D.: Steerable equivariant representation learning. arXiv preprint arXiv:2302.11349 (2023)
2. Boulch, A., Sautier, C., Michele, B., Puy, G., Marlet, R.: ALSO: Automotive lidar self-supervision by occupancy estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13455–13465 (2023)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
4. Choy, C., Gwak, J., Savarese, S.: 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3075–3084 (2019)
5. Dangovski, R., Jing, L., Loh, C., Han, S., Srivastava, A., Cheung, B., Agrawal, P., Soljacic, M.: Equivariant self-supervised learning: Encouraging equivariance in representations. In: International Conference on Learning Representations (2022)
6. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel R-CNN: Towards high performance voxel-based 3D object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1201–1209 (2021)
7. Devillers, A., Lefort, M.: EquiMod: An equivariance module to improve self-supervised learning. arXiv preprint arXiv:2211.01244 (2022)
8. Fan, L., Pang, Z., Zhang, T., Wang, Y.X., Zhao, H., Wang, F., Wang, N., Zhang, Z.: Embracing single stride 3D object detector with sparse transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8458–8468 (2022)
9. Garrido, Q., Najman, L., Lecun, Y.: Self-supervised learning of split invariant equivariant representations. In: The Fortieth International Conference on Machine Learning (2023)
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
11. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent: A new approach to self-supervised learning. Advances in neural information processing systems **33**, 21271–21284 (2020)
12. Gupta, S., Robinson, J., Lim, D., Villar, S., Jegelka, S.: Learning structured representations with equivariant contrastive learning. ICML Workshop on Topology, Algebra, and Geometry in Machine Learning (2023)
13. He, K., Chen, X., Xie, S., Li, Y., Doll'ar, P., Girshick, R.B.: Masked autoencoders are scalable vision learners. 2022 ieee. In: CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15979–15988 (2021)

14. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
15. Huang, S., Xie, Y., Zhu, S.C., Zhu, Y.: Spatio-temporal self-supervised representation learning for 3d point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6535–6545 (2021)
16. Jin, Z., Lei, Y., Akhtar, N., Li, H., Hayat, M.: Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7233–7243 (2022)
17. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: PointPillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
18. Liao, Y., Xie, J., Geiger, A.: KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. Pattern Analysis and Machine Intelligence (PAMI) (2022)
19. Mao, J., Niu, M., Jiang, C., Chen, J., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., Yu, J., et al.: One million scenes for autonomous driving: Once dataset. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1) (2021)
20. Nunes, L., Marcuzzi, R., Chen, X., Behley, J., Stachniss, C.: SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination. IEEE Robotics and Automation Letters **7**(2), 2116–2123 (2022)
21. Nunes, L., Wiesmann, L., Marcuzzi, R., Chen, X., Behley, J., Stachniss, C.: Temporal consistent 3D LiDAR representation learning for semantic perception in autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5217–5228 (2023)
22. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: PV-RCNN: Point-voxel feature set abstraction for 3D object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10529–10538 (2020)
23. Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., Li, H.: PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection. International Journal of Computer Vision **131**(2), 531–551 (2023)
24. Shi, S., Wang, X., Li, H.: PointRCNN: 3D object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019)
25. Shi, S., Wang, Z., Shi, J., Wang, X., Li, H.: 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**(08), 2647–2664 (2021)
26. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2446–2454 (2020)
27. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching efficient 3D architectures with sparse point-voxel convolution. In: European conference on computer vision. pp. 685–702. Springer (2020)
28. Team, O.D.: OpenPCDet: An open-source toolbox for 3D object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet` (2020)

29. Wei, Y., Wang, Z., Rao, Y., Lu, J., Zhou, J.: PV-RAFT: Point-voxel correlation fields for scene flow estimation of point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6954–6963 (2021)
30. Xiao, T., Wang, X., Efros, A.A., Darrell, T.: What should not be contrastive in contrastive learning. In: International Conference on Learning Representations (2021)
31. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves ImageNet classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10687–10698 (2020)
32. Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L., Litany, O.: 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. pp. 574–591. Springer (2020)
33. Xiong, Y., Ren, M., Zeng, W., Urtasun, R.: Self-supervised representation learning from flow equivariance. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10191–10200 (2021)
34. Xu, R., Wang, T., Zhang, W., Chen, R., Cao, J., Pang, J., Lin, D.: 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. pp. 13445–13454 (2023)
35. Yan, X., Chen, R., Zhang, B., Yuan, J., Cai, X., Shi, B., Shao, W., Yan, J., Luo, P., Qiao, Y.: SPOT: Scalable 3D pre-training via occupancy prediction for autonomous driving. arXiv preprint arXiv:2309.10527 (2023)
36. Yan, Y., Mao, Y., Li, B.: SECOND: Sparsely embedded convolutional detection. Sensors **18**(10) (2018). `https://doi.org/10.3390/s18103337`, `https://www.mdpi.com/1424-8220/18/10/3337`
37. Yang, H., He, T., Liu, J., Chen, H., Wu, B., Lin, B., He, X., Ouyang, W.: 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. pp. 9403–9414 (2023)
38. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3DSSD: Point-based 3D single stage object detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11040–11048 (2020)
39. Yin, J., Zhou, D., Zhang, L., Fang, J., Xu, C.Z., Shen, J., Wang, W.: ProposalContrast: Unsupervised pre-training for LiDAR-based 3D object detection. In: European Conference on Computer Vision. pp. 17–33. Springer (2022)
40. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3D features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10252–10263 (2021)
41. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9939–9948 (2021)