The Devil is in the Details: Simple Remedies for Image-to-LiDAR Representation Learning
— Supplementary Material —

Wonjun Jo[1], Kwon Byung-Ki[2], Kim Ji-Yeon[3], Hawook Jeong[4], Kyungdon Joo[5], and Tae-Hyun Oh[1,2,6]

[1] Department of Electrical Engineering, POSTECH, South Korea
[2] Graduate School of AI, POSTECH, South Korea
[3] Department of Convergence IT Engineering, POSTECH, South Korea
[4] RideFlux Inc., South Korea
[5] Artificial Intelligence Graduate School, UNIST, South Korea
[6] Institute for Convergence Research and Education in Advanced Technology, Yonsei University, South Korea
{jo1jun,byungki.kwon,jiyeon.kim,taehyun}@postech.ac.kr, hawook@rideflux.com, kyungdon@unist.ac.kr

In this supplementary material, we provide the additional qualitative results (Sec. 1), additional experiments (Sec. 2), pseudo-code of the overall pipeline (Sec. 3), and implementation details (Sec. 4), which are not presented in the main paper due to space limitations.

1 Additional Qualitative Results

Cosine Similarity. We present the 3D feature cosine similarity maps (See Fig. S1). After extracting 3D point-wise features through a pre-trained 3D model, cosine similarity is computed between the query point (red dot) feature and all the other point features. Then, visualization is performed by projection to the corresponding image. The projected points' colors go from violet to yellow for low and high similarity, respectively. The results show that a pre-trained model with treatments learns a more coherent 3D representation of the same objects.

2 Additional Experiments

2.1 Keyframe Only

As shown in Table 2 of the main paper, our method utilizes the unsynced inter-frame LiDAR point clouds from the nuScenes dataset [3]. To ensure a fair comparison with previous image-to-LiDAR distillation methods that only use synced keyframe data, we report the results of our method using only the keyframe data. The Ours-Keyframe method matches keyframe images with keyframe LiDAR from different timestamps rather than matching inter-frame LiDAR with keyframe images to compose unsynced data (See Table S1b). Although relying solely on keyframes can increase the misalignment between points and pixels, our method still outperforms the previous methods (See Table S1a).

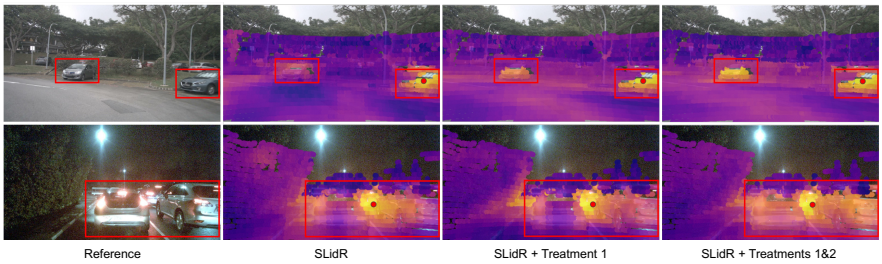
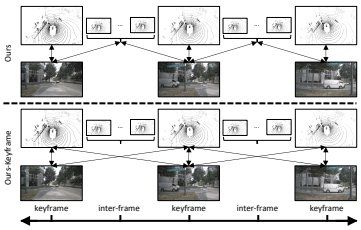


Fig. S1. Qualitative results of cosine similarity. The cosine similarity between the query point (red dot) and the 3D point feature learned with SLidR, SLidR with treatment1, and SLidR with treatments 1 and 2. The projected points' colors go from violet to yellow for low and high similarity, respectively. We show these results in the validation set of nuScenes [3]. This result shows that SLidR with treatments learns a more coherent 3D representation of the same objects.

Table S1. 3D semantic segmentation results on nuScenes and SemanticKITTI validation sets. We compare our method using only the keyframe with the existing 3D representation learning methods using the nuScenes and SemanticKITTI datasets. Our method using only the keyframe surpasses the existing methods across all metrics.

Method	nuScenes		SemanticKITTI
	Lin. Prob.	1%	1%
Random	8.1	30.3	39.5
PointContrast [14]	21.9	32.5 (+2.2)	41.1 (+1.6)
DepthContrast [15]	22.1	31.7 (+1.4)	41.5 (+2.0)
PPKT [9]	36.4	37.8 (+7.5)	43.9 (+4.4)
SLidR [13]	38.8	38.2 (+7.9)	44.6 (+5.1)
ST-SLidR [10]	40.4	40.7 (+10.4)	44.7 (+5.2)
TriCC [11]	38.0	41.2 (+10.9)	45.9 (+6.4)
Ours-Keyframe	**46.3**	**41.6 (11.3)**	**50.3 (10.8)**

(a)



(b)

2.2 Various Voxel Sizes

We report how changing the voxel size for each coordinate affects linear probing performance on the nuScenes dataset. We denote the voxel size for cylindrical coordinate as the values of $\delta\rho$ and δz.

In cylindrical coordinate, a voxel size of $10cm$, and in Cartesian coordinate, a voxel size of $5cm$, are found to be best. Regardless of the coordinate system, excessively reducing the voxel size to minimize quantization error can lead to a significant decrease in performance (See Table S2). While over-reducing the voxel size decreases quantization error and increases the number of preserved raw points, it can result in sparse data, with most voxels being empty. This sparsity can pose challenges for 3D networks in effectively learning and recognizing patterns.

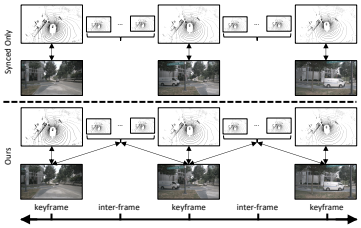
Table S2. Impact of coordinate and various voxel sizes. We report that changing the voxel size for each coordinate affects linear probing performance on the nuScenes dataset, with optimal sizes being $10cm$ in cylindrical coordinates and $5cm$ in Cartesian coordinates. However, excessively reducing the voxel size results in data sparsity, which poses challenges for 3D network pattern recognition.

Coordinate \ Voxel Size (cm)	1	5	10	20
Cylindrical	33.0	38.0	38.8	37.5
Cartesian	31.3	41.2	40.8	40.8

Table S3. Ablation study on the number of sampling inter-frame LiDAR. Performance improvements are marginal whether sampling once or twice. This indicates the importance of using unsynced data itself to achieve better results.

# inter-frame	nuScenes Lin. Prob. (100%)
Synced Only	41.2
1	45.2
2	45.3

(a)



(b)

2.3 The Number of Sampling Inter-frame LiDAR

We report the performance variations based on the number of LiDAR samplings in inter-frame data when utilizing unsynced data. The difference in performance between sampling once and sampling twice is negligible (See Table S3). This indicates that the utilization of unsynced data itself is crucial.

2.4 Dynamic Point Cloud Accumulation

We compare PPM's point cloud accumulation performance with that of existing methods, WsRSF [6] and PCAccumulation [7]. PPM is based on an unsupervised method, WsRSF on a weakly supervised method, and PCAccumulation on a supervised method to point cloud accumulation. PPM demonstrates good performance on static parts but has room for improvement in handling dynamic parts, which are a primary cause of misalignment on unsynced data (See Table S4). As shown in Table 2 of the main paper, correcting misalignments with PPM can enhance the performance of image-to-LiDAR distillation. Following this trend, we expect that replacing PPM with a more effective point cloud accumulation method could further improve the performance of image-to-LiDAR distillation. However, LiDAR 3D scene flow needs to be trained whenever the data domain is changed; its model designs often cover a limited range and point cloud. Therefore, we propose the PPM, whose design motivation is its unsupervised manner and versatility with respect to LiDAR characteristics and environments.

Table S4. Dynamic Point Cloud Accumulation results on nuScenes. Point cloud accumulation performances show that PPM, an unsupervised method, excels in static parts but has room for improvement with dynamic parts, in contrast to WsRSF, a weakly supervised method, and PCAccumulation, a supervised method.

Method	Strategy	Static part				Dynamic foreground				
		EPE avg.↓	AccS↑	AccR↑	ROutlier↓	EPE avg.↓	EPE med.↓	AccS↑	AccR↑	ROutliers↓
N/A	-	1.452	18.0	19.5	74.1	1.903	1.017	2.4	6.5	79.7
WsRSF [6]	Weakly	0.195	57.4	82.6	4.8	0.539	0.204	17.9	37.4	32.0
PCAccumulation [7]	Supervised	0.111	65.4	88.6	**1.1**	**0.301**	**0.146**	**26.6**	**53.4**	**12.1**
PPM	Unsupervised	**0.102**	**83.0**	**89.2**	5.0	0.992	0.409	13.3	26.3	49.5

Table S5. Additional Verification of Treatment 1. We report the performance of SLidR using different 3D voxel-based backbones, divided at the midline, with both Cylindrical and Cartesian coordinates. The results are presented for three different runs. Specifically, VoxelNet is pre-trained for 20 epochs. Consistent improvements are observed with the use of treatment1 across multiple runs and different backbones.

Method	Coordinate	3D backbone	nuScenes (100%)		
			Lin. Prob. (run 1)	Lin. Prob. (run 2)	Lin. Prob. (run 3)
SLidR	Cylindrical	MinkUNet	38.8	38.4	38.9
SLidR	**Cartesian**	MinkUNet	**41.6**	**41.2**	**41.8**
SLidR	Cylindrical	VoxelNet	25.0	25.5	25.3
SLidR	**Cartesian**	VoxelNet	**27.3**	**26.8**	**26.3**

2.5 Additional Verification of Treatment1

We verify the usefulness of treatment1 by adapting different voxel-based networks and conducting multiple runs. Table S5 shows that voxel-based networks with treatment1 consistently outperform those without it, even after multiple runs.

2.6 Complexity and Extra Memories for Treatments

Using PPM, the whole nuScenes dataset can be processed in 5 hours. As a data preprocessing method, PPM only needs to be performed once. Table S6 shows per GPU memory consumption, per epoch training time, and linear probing performance (LP) when applying Treatment 1&2. While our treatments slightly increase memory consumption and training time, our performance improvement is notable compared to (E) SLidR with similar resource

2.7 Different 2D Backbone for Distillation

To verify that our treatments are consistently applicable to different pre-trained 2D backbones, we replace the pre-trained 2D backbone with a ViT-S/8 model trained with DINO and report the performance. Table S7 shows consistent performance improvements, demonstrating that our treatments are effective across various 2D backbones.

Table S6. Resources required for Treatment 1&2. We report the per GPU memory consumption, per epoch training time, and linear probing performance (LP) when applying Treatment 1&2. While our treatments slightly increase memory consumption and training time, performance improvement is notable compared to SLidR, which has similar resources.

	Method	Epoch	Batch size	Memory [MB]	Time [hour]	LP
(A)	SLidR (Cylindrical)	50	16	10.4	0.5	38.8
(B)	+ Treatment 1 (Cartesian)	50	16	12.8	0.7	41.2
(C)	+ Treatment 2 (PPM)	50	32	18.4	0.9	41.2
(D)	+ Treatment 1&2 (ours)	50	32	21.6	1.2	45.2
(E)	+ Treatment 1&2 (ours)	20	16	13.0	0.5	44.7

Table S7. Linear Probing results of different 2D backbones. We report the performance of various methods using different pre-trained 2D backbones, specifically comparing MoCov2 and DINOv1. The methods are pre-trained on the nuScenes dataset. The results demonstrate that our treatments show the highest improvement in performance across both backbones, indicating consistent applicability of our treatments to various 2D backbones.

Method	Pretrain dataset	2D backbone	
		MoCov2	DINOv1
PPKT	Nuscenes	36.4	38.6
SLidR	Nuscenes	38.8	39.3
Ours	Nuscenes	**45.2**	**47.3**

Table S8. Linear Probing results of different frame gaps. We report the linear probing (LP) performance for different frame gaps, ranging from 1 to 40. The results indicate that performance increases as the frame gap increases up to 10 frames and then slightly decreases as the frame gap continues to increase.

Method	Frame Gap					
	1	5	10	20	30	40
Ours	45.6	47.3	47.3	47.1	47.2	46.5

2.8 Results of Different Frame Gaps

We experiment to see how performance changes according to different frame gaps from keyframe data. We evaluate our treatments using a ViT-S/8 2D backbone trained with DINO, measuring LP performance with frame gaps up to 40. Table S8 shows that performance increases as the frame gap increases up to 10 frames and then decreases as the frame gap continues to increase. The performance increase is likely due to data diversity, while the decrease is likely due to the expected increase in PPM errors.

3 Algorithm

In this section, we provide a pseudo-code of our overall pipeline in Algo.1. The *sampling* function is designed to alleviate data redundancy by more frequently sampling data from inter-frames that are farthest from the keyframe. Transformation Z is initialized to an identity matrix for all points, and the computed transformation is assigned only to points classified as moving. Within the *transformation* function, x_p_s is transformed to global coordinates, transformed by Z, and restored to LiDAR sensor coordinates. Within the *pixelPointMatching* function, x_p_t and x_p_s_trans are transformed from LiDAR sensor coordinates to camera sensor coordinates and projected to the 2D coordinate of x_i_t through the camera intrinsic matrix. The *pixelPointMatching* function creating a pixel-point matching index corresponds to function T, first mentioned in Sec. 3.2 of the main paper. For brevity, the algorithm does not include the matching of SLIC [1] based superpixels and corresponding point clouds matching.

4 Implementation Details

Positive Pair Mining (PPM). Figure S2 describes the overall scheme of PPM. The 11 number of consecutive point clouds $\{P^{t-5}, ..., P^t, ..., P^{t+5}\}$ are aggregated in the global coordinate through the relative poses readily obtained from GPS and IMU [5]. We first split the aggregated points into ground and non-ground points in sequence using an unsupervised ground removal method [8]. The non-ground points are converted to clustered points using HDBSCAN [4], and they pass through two consecutive steps: *Moving cluster tracking* and *Cluster-wise ICP*, as shown in Fig. S3. Moving cluster tracking identifies clusters that are in motion. We form each cluster's points in consecutive times and then calculate their center coordinates. If any l_1 distance between the center coordinates of consecutive times exceeds the threshold c, we categorize the points in the cluster as moving points and non-moving ones otherwise. We set c to 0.5 meter. The clusters of moving points are fed to the Cluster-wise ICP. In the Cluster-wise ICP, we apply an unsupervised point cloud matching [12] to each moving cluster by exploiting the keyframe as the reference frame. This mining process outputs the 3D transformation Z for each cluster that is combined with T to obtain the positive pixel-point matching index.

For ground removal, we utilize the patchwork++[7] [8]. Because the official implementation is designed for the SemanticKITTI dataset [2], to apply it to the nuScenes dataset [3], we modified to set the mean coordinates of the aggregated point clouds to zero by subtracting the mean coordinates. For the HDBSCAN clustering [4], we set a minimum number of clusters to 50, the number of clusters to 300, α to 1, distance metric to Euclidean, and the leaf size to 100. For the mean tracking in cluster tracking.

[7] https://github.com/url-kaist/patchwork-plusplus

Algorithm 1: PyTorch-style pseudo-code of ours overall pipeline.

```
# f_p, f_i: 3d and 2d network
# x_p_t, x_p_s: 3d point cloud at keyframe t and inter-frame s
# x_p_s_list: list of 3d point cloud at inter-frames
# x_i_t: 2d image at keyframe t
# aug_p, aug_i: Augmentations for point clouds and images
# quant: Quantization for point clouds

for x_p_t, x_p_s_list, x_i_t in loader:

   Z = ppm(x_p_t, x_p_s_list, x_i_t)# get transformation Z

   s_i = sampling(x_p_s_list)# get index at inter-frame s
   x_p_s, Z = x_p_s_list[s_i], Z[s_i]
   x_p_s_trans = transformation(x_p_s, Z)# get transformed points

   # get positive pixel-point matching index
   i_t_i, p_t_i = pixelPointMatching(x_i_t, x_p_t)
   i_s_i, p_s_i = pixelPointMatching(x_i_t, x_p_s_trans)

   # augment, quantize, and feed-forward
   F_p_t, F_p_s, F_i_t = f_p(quant(aug_p(x_p_t))),
    f_p(quant(aug_p(x_p_s))), f_i(aug_i(x_i_t))

   # pair point and pixel-wise feature
   F_p_t, F_p_s, F_i_t, F_i_s = F_p_t[p_t_i], F_p_s[p_s_i],
    F_i_t[i_t_i], F_i_t[i_s_i]

   loss = contrastDistill(cat(F_p_t, F_p_s), cat(F_i_t, F_i_s))
   loss.backward()
   update(f_p, f_i)

def ppm(x_p_t, x_p_s_list, x_i_t):

   Z = eye(4).reshape((1, 4, 4)).repeat(len(cat(x_p_t,
    x_p_s_list)), 1, 1)

   # aggregation in the global coordinate
   x_p_t, x_p_s_list = sensor2global(x_p_t),
    sensor2global(x_p_s_list)
   x_p = cat(x_p_t, x_p_s_list)

   g_i = groundRemoval(x_p)# get ground point index
   x_p_ng = x_p[∼ g_i]# get non-ground point

   c_i = clustering(x_p_ng)# get cluster index

   m_i = movingClusterTracking(x_p_ng, c_i)# get moving point index

   x_p_m, c_i = x_p_ng[m_i], c_i[m_i]# get moving point and moving
    cluster index
   Z[∼g_i][m_i] = clusterWiseICP(x_p_m, c_i)# assign computed transformation

   return Z[len(x_p_t):]# return transformation Z at inter-frames

def contrastDistill(F_p, F_i):

   logits = mm(norm(F_p), norm(F_i).T)
   loss = crossEntropyLoss(logits/τ , range(len(F_p)))

   return loss
```

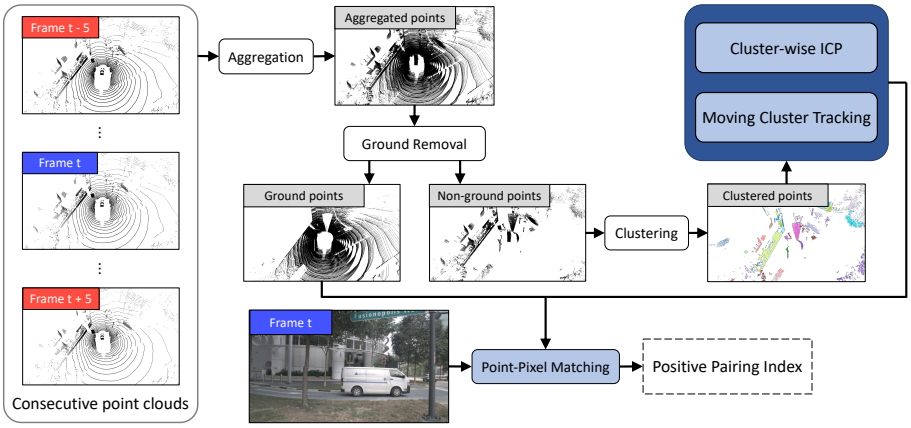


Fig. S2. The overview of the Positive Pair Mining (PPM) module. The Positive Pair Mining consists of four components, *i.e.*, aggregation, ground removal, clustering, moving cluster tracking, cluster-wise ICP, and point-pixel matching steps. The aggregation step aggregates consecutive point clouds in the global coordinate. The ground removal step separates aggregated points into ground and non-ground points. The moving cluster tracking and cluster-wise ICP transform all the moving points from the inter-frames into the nearest keyframe t. The point-pixel matching step constructs positive pairs of 3D-2D by projecting transformed 3D points to the 2D image at keyframe t.

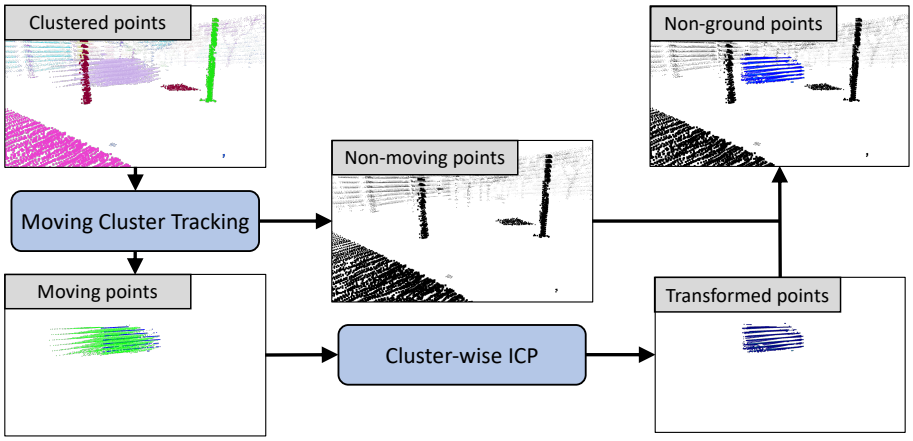


Fig. S3. The pipeline of moving cluster tracking and cluster-wise ICP. (moving cluster tracking) To distinguish the non-moving and moving points from the clustered points, we form each cluster's points in consecutive times and then measure their center coordinates. If any $L1$-distance is larger than the threshold c, we categorize the points in the cluster into moving points. (cluster-wise ICP) Using an unsupervised point cloud matching method, we obtain the 3D transformation Z to the keyframe for generating the positive pairing index.

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **34**(11), 2274–2282 (2012)
2. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: IEEE International Conference on Computer Vision (ICCV). pp. 9297–9307 (2019)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11621–11631 (2020)
4. Campello, R.J., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining. pp. 160–172. Springer (2013)
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3354–3361. IEEE (2012)
6. Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T.: Weakly supervised learning of rigid 3d scene flow. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5692–5703 (2021)
7. Huang, S., Gojcic, Z., Huang, J., Wieser, A., Schindler, K.: Dynamic 3d scene analysis by point cloud accumulation. In: European Conference on Computer Vision. pp. 674–690. Springer (2022)
8. Lee, S., Lim, H., Myung, H.: Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 13276–13283. IEEE (2022)
9. Liu, Y.C., Huang, Y.K., Chiang, H.Y., Su, H.T., Liu, Z.Y., Chen, C.T., Tseng, C.Y., Hsu, W.H.: Learning from 2d: Contrastive pixel-to-point knowledge transfer for 3d pretraining. arXiv preprint arXiv:2104.04687 (2021)
10. Mahmoud, A., Hu, J.S., Kuai, T., Harakeh, A., Paull, L., Waslander, S.L.: Self-supervised image-to-point distillation via semantically tolerant contrastive loss. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7102–7110 (2023)
11. Pang, B., Xia, H., Lu, C.: Unsupervised 3d point cloud representation learning by triangle constrained contrast for autonomous driving. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5229–5239 (2023)
12. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: Proceedings third international conference on 3-D digital imaging and modeling. pp. 145–152. IEEE (2001)
13. Sautier, C., Puy, G., Gidaris, S., Boulch, A., Bursuc, A., Marlet, R.: Image-to-lidar self-supervised distillation for autonomous driving data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9891–9901 (2022)
14. Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L., Litany, O.: Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In: European Conference on Computer Vision (ECCV). pp. 574–591. Springer (2020)
15. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. In: IEEE International Conference on Computer Vision (ICCV). pp. 10252–10263 (2021)