UpCycling: Semi-supervised 3D Object Detection without Sharing Raw-level Unlabeled Scenes

Sunwook Hwang† Youngseok Kim† Seongwon Kim§ Saewoong Bahk† ∗ Hyung-Sin Kim‡ ∗
†Department of Electrical and Computer Engineering, Seoul National University, ∗Corresponding author
§SK Telecom, Seoul, Korea, ‡Graduate School of Data Science, Seoul National University
{swhwang, yskim}@netlab.snu.ac.kr, s1kim@sk.com, {sbahk, hyungkim}@snu.ac.kr

Abstract

This is a supplementary material which provides additional details for the paper.

A. Implementation Details

A.1. Experiment settings

Training. For the pre-training stage, we train on 4 RTX 3090 GPUs with a batch size of 16 and 8 for SECOND-IoU and PV-RCNN, respectively. Then, following the original model training settings, we use epochs 80, and 30 for KITTI dataset and Waymo dataset, respectively. Especially, pre-training on small amounts of KITTI labeled data 2%, we lengthen the epoch to 120 for the model to converge. For the semi-supervised learning stage, we train with a batch size of 32 (16 labeled + 16 unlabeled, 4 GPUs) and 16 (8 labeled + 8 unlabeled, 4 GPUs) for SECOND-IoU and PV-RCNN, respectively. We set the ratio of unlabeled data to twice that of labeled data in domain adaptation experiments. The learning rate is initialized as the value of the original model usage and updated by cosine annealing strategy.

Table 1: Waymo [7], KITTI [3], and Lyft [6] dataset overview. † and ∗ indicate obtaining information from [10] and [12], respectively.

	Waymo	KITTI	Lyft
LiDAR Type	64-beam	64-beam	64-beam
Beam Angles †	[-18.0°, 2.0°]	[-23.6°, 3.2°]	[-29.0°, 5.0°]
Points per Scene ∗	160,139	118,624	69,175
Training Frames	158,081	3,712	18,900
Validation Frames	39,987	3,769	3,780
Night / Rainy	Yes / Yes	No / No	No / No
Location	USA	Germany	USA

Dataset and Source Code License. We implement our UpCycling based on OpenPCDet [8] (v0.5.1) which is licensed under the Apache License 2.0. According to https://paperswithcode.com/datasets, the license of Waymo dataset [7] and KITTI dataset [3] is the custom (non-commercial) and the CC BY-NC-SA 3.0, respectively, and the license of Lyft dataset [6] is unknown. The details of each datasets are in Table 1.

A.2. Architecture details – 3D backbone network

In this paper, the 3D backbone network of SECOND [11] (see Table 2) is used for generating the grid-type feature data in PV-RCNN and SECOND-IoU experiments. Voxel Feature Extractor (VFE) converts the point cloud data into voxel format covering the entire point cloud range. After that, the output of VFE goes through the SparseConv layers [4] where each Conv layer contains both batch normalization and ReLU, which is a non-linear function. Lastly, the output of SparseConv layers becomes the grid-type feature data which UpCycling utilizes. On the other hand, the 3D backbone network of PV-RCNN additionally generates the set-type features from Voxel Set Abstraction (VSA) (see Table 3). In this process, PV-RCNN samples a fixed number of keypoints from raw points following the Farthest-first rule. After that, set abstraction modules create voxel-wise features from each layer in VFE corresponding to keypoint positions. Finally, to generate the final form of set-type features, VSA Point Feature Fusion module concatenates the features from the set abstraction modules to the accurate keypoint positions.

A.3. Implementation Details for SECOND-IoU based 3DIoUMatch

We basically follow and reuse the official codes from the SOTA schemes for comparison except for 3DIoUMatch [9]. 3DIoUMatch method uses IoU-guided NMS modules for filters pseudo labels. However, the authors did not implement 3DIoUMatch in SECOND-IoU, we have implemented 3DIoUMatch on SECOND-IoU to analyze its effectiveness compared with UpCycling.

According to 3DIoUMatch, among the pseudo labels filtered according to the module in IoU, only terms that help improve box regression are selectively included in the loss. In the first attempt, the experiment † case in Table 4 is con-

Table 2: 3D backbone network architecture generating grid-type feature data.

Layers		**BACKBONE Network**	Output size
VFE		Mean VFE	4×41×1600×1408
SparseConv Layers	conv_input	4×3×3×3, 16, padding 1,1,1	16×41×1600×1408
	conv_1	16×3×3×3, 16	16×41×1600×1408
	conv_2	16×3×3×3, 32, stride 2,2,2, padding 1,1,1 32×3×3×3, 32 32×3×3×3, 32	32×21×800×704
	conv_3	32×3×3×3, 64, stride 2,2,2, padding 1,1,1 64×3×3×3, 64 64×3×3×3, 64	64×11×400×352
	conv_4	64×3×3×3, 64, stride 2,2,2, padding 0,1,1 64×3×3×3, 64 64×3×3×3, 64	64×5×200×176
	conv_out	64×3×1×1, 128, stride 2,1,1	128×2×200×176

Table 3: 3D backbone network architecture generating set-type feature data.

Layers		**BACKBONE Network**		Output size
Key Point Sampling		Farthest Point Sampling		4×2048
VoxelSetAbstraction (VSA) Layers	SA_raw	radius 0.4 4×1×1, 16 16×1×1, 16	radius 0.8 4×1×1, 16 16×1×1, 16	32×2048
	SA_pv1	radius 0.4 19×1×1, 16 16×1×1, 16	radius 0.8 19×1×1, 16 16×1×1, 16	32×2048
	SA_pv2	radius 0.8 35×1×1, 32 32×1×1, 32	radius 1.2 35×1×1, 32 32×1×1, 32	64×2048
	SA_pv3	radius 1.2 67×1×1, 64 64×1×1, 64	radius 2.4 67×1×1, 64 64×1×1, 64	128×2048
	SA_pv4	radius 2.4 67×1×1, 64 64×1×1, 64	radius 4.8 67×1×1, 64 64×1×1, 64	128×2048
	SA_BEV	Bilinear Interpolation		256×2048
VSA Point Feature Fusion	Concat	$[\ f^{raw}, f^{pv1}, f^{pv2}, f^{pv3}, f^{pv4}, f^{BEV}\ ]$		640×2048
	Linear Layer	640, 128		128×2048

Table 4: Partial-label scenario results with 2% of labeled data in the KITTI dataset. 3DIoUMatch † indicates the first attempt experiment of not applying selective supervision of box regression loss term.

AP_{3D}		2%		
		Easy	Mod	Hard
SECOND-IoU	Baseline	56.69	44.11	37.19
	3DIoUMatch †	29.12	23.03	20.33
	improved (%)	-48.64	-47.78	-45.32
	3DIoUMatch	63.57	49.58	43.00
	improved (%)	12.13	12.39	15.62

ducted, including both the box regression and cls loss value from the RPN module among the pseudo labels extracted from SECOND-IoU. The performance, however, is severely degraded compared with the baseline model's performance. Thus, as following implementation of the 3DIoUMatch concept, we select only the loss useful for box regression among RPN module loss terms. It could be confirmed through the results of 3DIoUMatch from Table 4 that the baseline model performance is well improved by the correct loss selection. Through this, we could judge that the implementation of SECOND-IoU based 3DIoUMatch is reasonable.

A.4. Implementation of Inversion Attack

The research on inversion attacks that aim to restore original data from feature data has mainly focused on 2D images. Several studies, such as those referenced in [1, 2, 13], have proposed different inversion attack models based on convolutional neural networks (CNNs) and have shown improvements in performance by using prediction results and explanations. Additionally, an inversion attack model that utilizes a GAN generator and 1x1 convolution has been proposed in [5]. However, to the best of our knowledge, research on inversion attacks for 3D point clouds remains limited. For this purpose, we employ the inversion attack model utilizing the decoder method [2], which is commonly used to assess the invertibility of a model composed of convolutional layers [1, 13].

We have developed the inversion attack models that re-

Table 5: 3D reconstructor network architecture from xconv_1.

Layers		**RECONSTRUCTOR Network**	Output size
INPUT		xconv_1	16×41×400×352
Conv3d Layers	conv_1	16×3×3×3, 16, padding 1,1,1	16×41×400×352
	conv_2	16×3×3×3, 16, padding 1,1,1	16×41×400×352
	conv_3	16×3×3×3, 16, padding 1,1,1	16×41×400×352
ConvTranspose3d Layers	upconv_1	16×3×3×3, 4, padding 1,1,1	4×41×400×352

Table 6: 3D reconstructor network architecture from xconv_3.

Layers		**RECONSTRUCTOR Network**	Output size
INPUT		xconv_3	64×11×100×88
Conv3d Layers	conv_1	64×3×3×3, 64, padding 1,1,1	64×11×100×88
	conv_2	64×3×3×3, 64, padding 1,1,1	64×11×100×88
	conv_3	64×3×3×3, 64, padding 1,1,1	64×11×100×88
ConvTranspose3d Layers	upconv_1	64×3×3×3, 32, stride 2,2,2, padding 1,1,1/0,1,1	32×21×200×176
	upconv_2	32×3×3×3, 16, stride 2,2,2, padding 1,1,1/0,1,1	16×41×400×352
	upconv_3	16×3×3×3, 4, padding 1,1,1	4×41×400×352

Table 7: 3D reconstructor network architecture from xconv_out.

Layers		**RECONSTRUCTOR Network**	Output size
INPUT		xconv_out	128×2×50×44
Conv3d Layers	conv_1	128×3×3×3, 128, padding 1,1,1	128×2×50×44
	conv_2	128×3×3×3, 128, padding 1,1,1	128×2×50×44
	conv_3	128×3×3×3, 128, padding 1,1,1	128×2×50×44
ConvTranspose3d Layers	upconv_1	128×5×3×3, 64, stride 2,1,1, padding 1,1,1	64×5×50×44
	upconv_2	64×5×3×3, 64, stride 2,2,2, padding 1,1,1/0,1,1	64×11×100×88
	upconv_3	64×3×3×3, 32, stride 2,2,2, padding 1,1,1/0,1,1	32×21×200×176
	upconv_4	32×3×3×3, 16, stride 2,2,2, padding 1,1,1/0,1,1	16×41×400×352
	upconv_5	16×3×3×3, 4, padding 1,1,1	4×41×400×352

construct the raw point clouds from the intermediate features at the 1st, 3rd, and 5th convolution layers in 3D backbone network in Table 2, following the decoder method [2]. The inversion attack model structures for reconstructing features from the 1st, 3rd, and 5th layers are consistent with the structures presented in Tables 5, 6, and 7, respectively. The initial part of each inversion attack model consistently consists of three convolution layers. After that, the number of transposed convolution layers in the model corresponds to the count of layers that generate the input feature data.

To reconstruct the point clouds from input features for each dataset (KITTI, Waymo, and Lyft), we have developed independent inversion attack models for every dataset and followed the training settings in the decoder method [2].

B. Supplementary Evaluation

In this section, we provide additional supplementary experiment results that reinforce the arguments of this paper.

B.1. Analysis on 3D Scene Feature Augmentation

In the main paper, Section 4 displays the error bars for the case when compared with a grid-type feature. Further analysis was conducted on 3D scenes with set-type features as follows:

Comparing point set features directly as with grid-type features is not accurate, because raw-point augmentation influences point sampling. This means that feature augmentation is carried out based on point samples that differ from those used when raw-point augmentation is applied. To address this, we undertook additional experiments by comparing the nearest point features in pairs. As illustrated in Figure 4, the RMSE results are 1.605@FLIP, 1.297@ROT, and 0.906@GT, which confirms a trend similar to that observed with grid-type features.

B.2. Privacy Protection

B.2.1 Feature data produced from the 3D object detection network.

Figure 5 shows the grid-type feature's activation heatmaps and set-types feature's positions corresponding to GTs at the raw-point data. In UpCycling, the state of the feature-level data after passing the 3D Backbone networks is very coarse. UpCycling uses these de-identified feature-level data for SSL of 3D object detection. In order to extract identifying information from this de-identified data, inversion attacks must be employed. We will discuss the attempts to reconstruct data

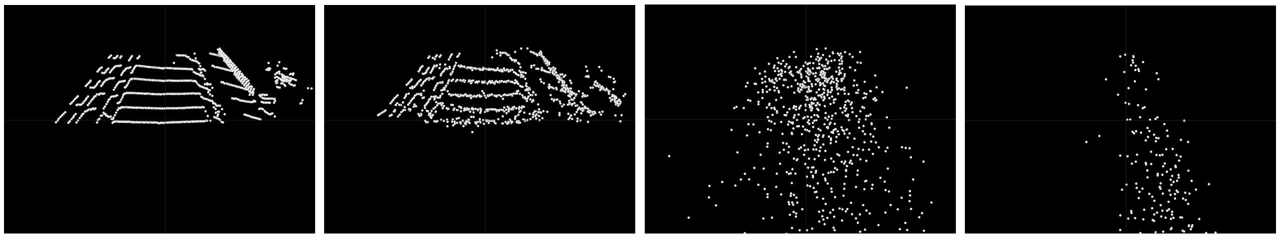
(a) Original raw-point scene (b) Restoration from the 1st layer (c) Restoration from the 3rd layer (d) Restoration from the 5th layer, same as UpCycling

Figure 1: Results of inversion attack for the 3D backbone model of SECOND-IoU and PV-RCNN. The example 3D point cloud scene is in **KITTI**.

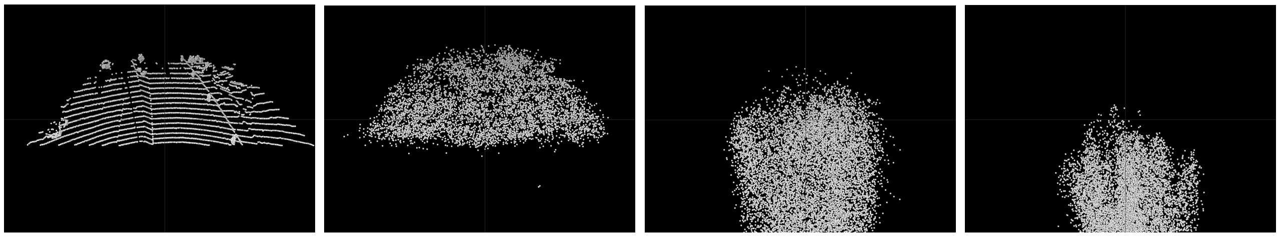
(a) Original raw-point scene (b) Restoration from the 1st layer (c) Restoration from the 3rd layer (d) Restoration from the 5th layer, same as UpCycling

Figure 2: Results of inversion attack for the 3D backbone model of SECOND-IoU and PV-RCNN. The example 3D point cloud scene is in **Waymo**.

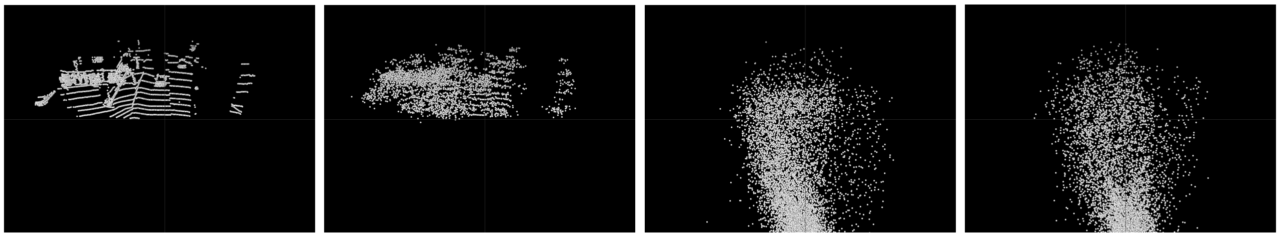
(a) Original raw-point scene (b) Restoration from the 1st layer (c) Restoration from the 3rd layer (d) Restoration from the 5th layer, same as UpCycling

Figure 3: Results of inversion attack for the 3D backbone model of SECOND-IoU and PV-RCNN. The example 3D point cloud scene is in **Lyft**.

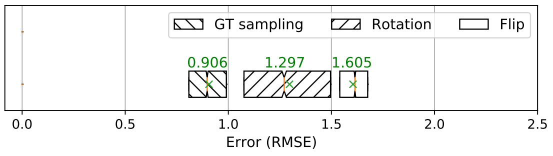

Figure 4: RMSE between raw- and set-type feature-level augmentations of the entire KITTI training dataset. Box range covers the first quartile to the third quartile and the mark '×' indicates the mean value.

via inversion attacks in the following section. Additionally, since a regular detection pipeline with the 3D scene naturally produces an unlabeled intermediate feature, UpCycling eliminates the need for extra AV-side computation (e.g., local training) or server-side annotation effort.

B.2.2 Restored point cloud scene using the inversion attack.

We perform an inversion attack on the 3D backbone network in SECOND-IoU and PV-RCNN. The 3D point cloud scenes in Figures 1-3(a) originate from the KITTI, Waymo, and Lyft datasets, respectively. Figures 1-3(b)-(d) present the restoration results for intermediate features at three different convolutional layers of the backbone network: 1st, 3rd, and 5th (last) layers, respectively. Although the point cloud restored from the first layer is relatively similar to the original scene, it becomes considerably different when applied to features from deeper layers in all cases. We confirm that intermediate feature data generated from the deepest layer utilized by UpCycling in all datasets, including KITTI, Waymo, and Lyft, makes it impossible to accurately recon-

Table 8: Effects of feature augmentation schemes in the domain adaptation scenario with the same settings as Sections 5.2 and 5.4

Dataset	Method	SECOND-IoU (Closed Gap[%])	
		AP_{BEV}	AP_{3D}
KITTI	Baseline	54.14 (0.00)	10.16 (0.00)
	Flip(w/ SN)	76.68 (62.23)	48.73 (53.67)
	Noise(w/ SN)	81.46 (75.43)	51.21 (57.12)
	RS(w/ SN)	78.59 (67.50)	46.52 (50.61)
	Rotation(w/ SN)	77.98 (65.83)	44.25 (47.44)
	UpCycling (w/ SN)	**84.12 (82.77)**	**67.65 (80.00)**
	Oracle	90.36 (100.0)	82.02 (100.0)

struct the original scene.

B.3. Effect of Feature Augmentation Schemes in Domain Adaptation

In Section 5.2, we investigate feature augmentation by evaluating the superiority of *F-GT*, which is utilized for UpCycling, to other augmentation schemes (*e.g*. Flip, Noise, RS, and Rotation) in a partial-label scenario. Further, we investigate the superiority of UpCycling to other feature augmentation schemes in the domain adaptation scenario with the same settings as Sections 5.2 and 5.4.

In this experiment, **Baseline** evaluates the baseline model pre-trained with Waymo dataset directly in target domain (KITTI) and **Oracle** adapts the model with fully supervised learning in the target domain, which provide the lower- and upper-bound performance, respectively. For feature-level augmentations, we adopts Flip, Noise, RS, and Rotation described in Section 5.2. We utilize SECOND-IoU and adopt **SN** option for adaptation to KITTI domain since object sizes in KITTI are different from those in Waymo.

Table 8 performs the same comparison in the domain adaptation scenario described in Section 5.4, showing each scheme's $\mathrm{AP_{BEV}}$, $\mathrm{AP_{3D}}$ performances and its relative position between Baseline (0) and Oracle (100). The results show that our UpCycling provides *the best performance* in all cases.

B.4. Other Class Detection Results

We report per-class average precision on other classes of the KITTI dataset in Table 9. We use the same settings as in Sections 5.5. The experiment using a 10% partial-label scenario on KITTI training data is essential to understand UpCycling's effectiveness to the Pedestrian class as well as the Car class. As shown in Table 9, UpCycling achieves to improve the detection accuracy in other classes significantly, regardless of the class, model, and task difficulty.

Table 9: The AP results for Car and Pedestrian classes in a partial-label scenario, utilizing 10% labeled data from the KITTI dataset.

$\mathrm{AP_{3D}}$(10%)		Car (@0.7 IoU)			Pedestrian (@0.5 IoU)		
		Easy	Mod	Hard	Easy	Mod	Hard
SECOND-IoU	Baseline	75.77	58.75	52.27	15.40	13.10	12.27
	UpCycling	**76.01**	**61.09**	**54.34**	**18.08**	**15.13**	**14.49**
	Improved (%)	0.32	3.98	3.96	17.40	15.50	18.10
PV-RCNN	Baseline	80.98	66.80	59.60	14.81	13.39	12.42
	UpCycling	**83.82**	**69.52**	**62.47**	**16.10**	**15.18**	**15.00**
	Improved (%)	3.51	4.07	4.82	8.71	13.37	20.77

$\mathrm{AP_{BEV}}$(10%)		Car (@0.7 IoU)			Pedestrian (@0.5 IoU)		
		Easy	Mod	Hard	Easy	Mod	Hard
SECOND-IoU	Baseline	82.59	73.63	65.80	22.02	18.30	17.48
	UpCycling	**86.81**	**75.87**	**67.28**	**23.59**	**19.67**	**18.95**
	Improved (%)	5.11	3.04	2.25	7.13	7.49	8.41
PV-RCNN	Baseline	89.22	80.95	73.30	16.87	15.26	15.01
	UpCycling	**91.33**	**83.25**	**75.85**	**20.03**	**18.27**	**18.10**
	Improved (%)	2.36	2.84	3.48	18.73	19.72	20.59

References

[1] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 2

[2] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 3

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1

[4] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 1

[5] Rudolf Herdt, Maximilian Schmidt, Daniel Otero Baguer, Jean Le'Clerc Arrastia, and Peter Maass. Model stitching and visualization how gan generators can invert networks in real-time, 2023. 2

[6] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. *CoRR*, abs/2006.14480, 2020. 1

[7] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[8] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020. 1

[9] He Wang, Yezhen Cong, Or Litany, Yue Gao, and Leonidas J Guibas. 3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection. In *Proceedings of the*

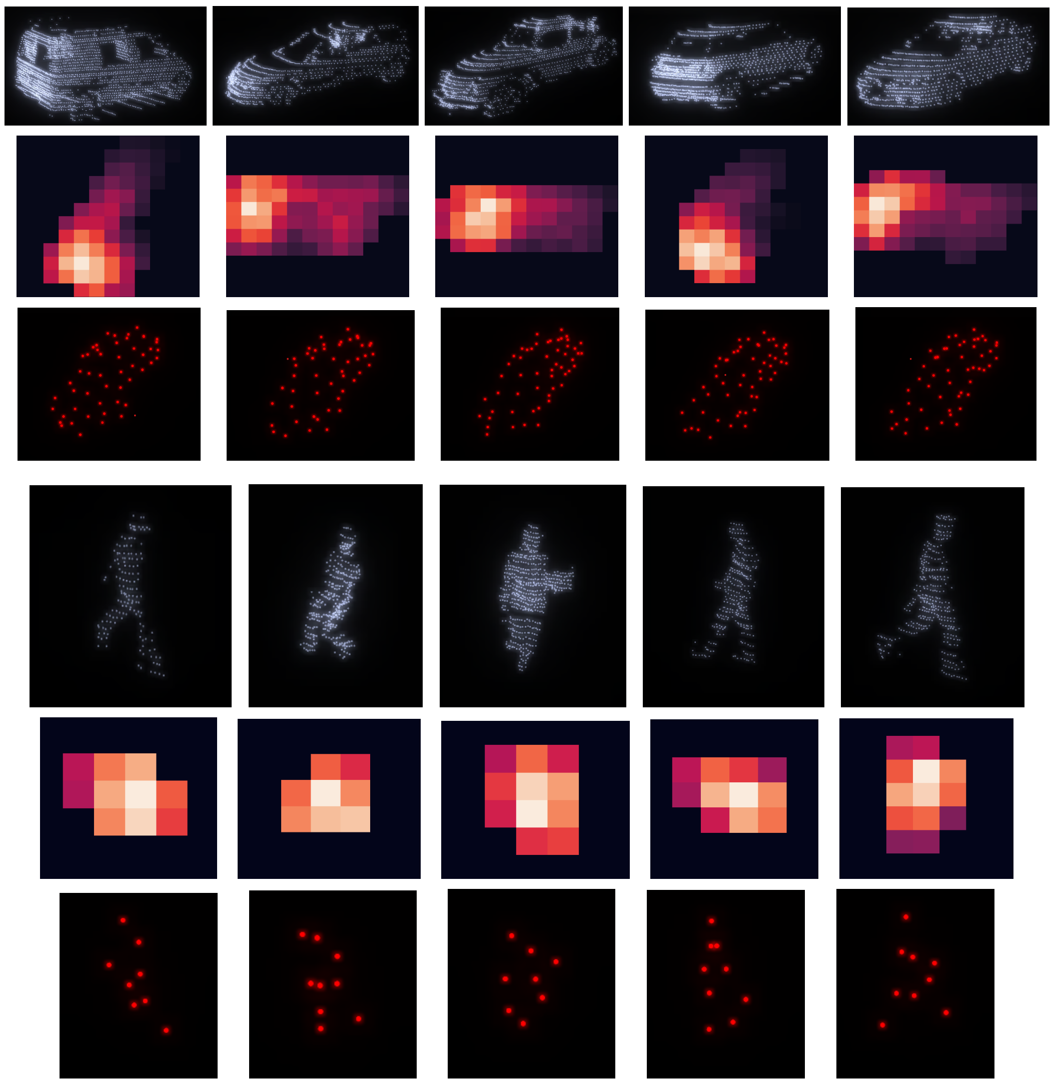

Figure 5: Figures for three classes in KITTI dataset: Cars (1st–3rd row), Pedestrians (4th–6th row). GT point clouds and corresponding grid-type feature's activation heatmaps and set-type feature's positions.

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14615–14624, 2021. 1

[10] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020. 1

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

[12] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2021. 1

[13] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. Exploiting explanations for model inversion attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 682–692, October 2021. 2