CONCRETIZER: MODEL INVERSION ATTACK VIA OCCUPANCY CLASSIFICATION AND DISPERSION CONTROL FOR 3D POINT CLOUD RESTORATION

ABSTRACT

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

A VOXELIZATION EFFECT

To address the challenging issue of restoring voxel-based 3D features to a 3D point scene, we utilize the Voxel Single-Point (VSP) hypothesis. This hypothesis asserts that a single point within a voxel is sufficient to restore the 3D point scene. We validate the VSP hypothesis by analyzing 3D scenes from the KITTI dataset using representative voxelization-based extractors (Zhou & Tuzel, 2018; Yan et al., 2018) commonly used in autonomous vehicle applications.

As shown in Figure 1 (left), 99.988% of the total voxels contain either no points or only a single point. Figure 1 (right) illustrates that voxels with multiple points, which are extremely rare, are mostly located near LiDAR sensors, contrasting with the broader distribution of single-point voxels. This is due to the inherent characteristic of LiDAR sensors, where the density of points decreases as the distance from the sensor increases. Figure 2 visualizes regions in the KITTI dataset where multi-point voxels exist, comparing the original point cloud with its voxelized result. This comparison highlights that even in areas close to the LiDAR sensor, there are negligible differences between the original and voxelized point clouds. This demonstrates that a single point per voxel is sufficient to preserve the integrity of the scene.

B DISPERSION OF VOI

The Voxels-of-Interest (VoI) experiences dispersion during feature extraction and restoration. In Figure 3 of main paper, 'Data stattistics' show the grid density at each layer throughout the feature extraction and restoration process. It is evident that the density increases as the data passes through the downsampling (3_{rd}, 6_{th}, 9_{th}, and 12_{th}) and upsampling (13_{th}, 16_{th}, 19_{th}, and 22_{th}) layers. This phenomenon is attributed to the characteristics of convolution and transposed convolution layer, which inherently spread values to the surrounding regions. Conversely, the density remains unchanged in other layers owing to the characteristics of submanifold convolution (Graham & Van der Maaten, 2017). Submanifold convolution effectively tackles memory consumption and computational overhead by preserving the spatial shape of the data during feature extraction, thereby maintaining unchanged density. Given the inherent characteristics of operations, *ConcreTizer* maximizes benefits of additional supervision by partitioning based on the downsampling layer where VoI dispersion manifests.

C METRICS

The mathematical expressions of the metrics used in evaluation part are as follows. In the following equations, Let P and Q denote the two point cloud sets and $||x||_2$ denote the Euclidean norm of vector x. (Implementations are based on Density-aware Chamfer distance code (Wu et al., 2021).)

- Chamfer distance (CD): The CD metric is computed by performing minimum-distance matching between two point cloud sets and then averaging the distances.

$$CD(P,Q) = \frac{1}{2}\left(\frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}||p-q||_2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}||p-q||_2\right).$$

- Hausdorff distance (HD): The HD metric is calculated by performing minimum-distance matching between two point cloud sets and then taking the maximum distance among the matched pairs.

$$HD(P,Q) = \max\left(\max_{p\in P}\min_{q\in Q}||p-q||_2, \max_{q\in Q}\min_{p\in P}||p-q||_2\right).$$

- F1 score: The F1 score can be obtained as a harmonic mean of precision and recall. The correctness of restored point is judged by whether it falls within a specified threshold radius from a GT point. During the evaluation on the KITTI and Waymo datasets, we set the threshold of F1 score as 15 cm and 30 cm, respectively.

$$F1score = 2 \times \frac{recall \times precision}{recall + precision}.$$

Each of the aforementioned metrics has its own strengths and limitations in fully evaluating restoration performance. Therefore, in the main paper, we introduce a variety of metrics and use visual aids to provide a more insightful understanding.

D Implementation Details

Training. The training process employs an RTX 3090 GPU with 24GB of memory. Initially, feature extractors are pre-trained separately on the KITTI (Geiger et al., 2012) and Waymo (Sun et al., 2020) datasets, and then frozen during the training of inversion attack models. The KITTI dataset consists of 3,712 training and 3,769 evaluation data, while the Waymo dataset comprises 15,809 training and 3,999 evaluation data (1/10 sampling ratio).

The point cloud range and voxel size for the 3D feature extractor are configured according to the 3D object detection benchmarks of each dataset. For KITTI, with a range of x: [0, 70.4] m, y: [-40, 40] m, and z: [-3, 1] m of range, the voxel size is set to (5 cm, 5 cm, 10 cm), resulting in a grid size of (1408, 1600, 40). For Waymo, with a range of x: [-75.2, 75.2] m, y: [-75.2, 75.2] m, and z: [-2, 4] m of range, the voxel size is set to (10 cm, 10 cm, 15 cm), resulting in a grid size of (1504, 1504, 40). Notably, during the training of our inversion attack models, we crop these regions to approximately 1/16 of the total range to accommodate GPU memory constraints. (For KITTI, x: [0, 17.6] m, y: [-10, 10] m, and z: [-3, 1] m, resulting in (352, 400, 40) grid. For Waymo, x: [0, 40] m, y: [-20, 20] m, and z: [-2, 4] m, resulting in (400, 400, 40) grid.) The region near the origin of the LiDAR sensor is selected because severe distortion is more likely to occur there due to the feature extractor. During evaluation, the range is extended back to the full object detection range. (For visualization, captured images from the close range are used.)

The training process uses the Adam optimizer with a learning rate of 0.0001. For the KITTI dataset, models are trained for 150 epochs with a batch size of 4, while 30 epochs with a batch size of 2 for the Waymo dataset. When employing SF loss (VOC and *ConcreTizer*), the γ value is set to 2. Tables 1, 2 present the α values used in the experiments. For *ConcreTizer*, the number of blocks increases alongside the number of downsampling layers; consequently, the α value for each block is denoted as an ordered pair.

The training process uses the Adam optimizer with a learning rate of 0.0001. For the KITTI dataset, models are trained for 150 epochs with a batch size of 4, while for the Waymo dataset, 30 epochs are used with a batch size of 2. When employing SF loss (for VOC and ConcreTizer), the gamma value is set to 2. Tables 1 and 2 present the α values used in the experiments. For ConcreTizer, the number of blocks increases with the number of downsampling layers; thus, the α value for each block is represented as an ordered pair.

License. The licenses of the datasets we used in the experiment are the custom (non-commercial) for KITTI dataset and the CC BY-NC-SA 3.0 for Waymo dataset, respectively. In the case of the 3D feature extractor, it was created based on the OpenPCDet (Team, 2020) project corresponding to the license of the Apache License 2.0.

E MODEL ARCHITECTURE

3D feature extractor. Our training process employs two feature extractors: VoxelBackBone and VoxelResBackBone. Their structures are provided in Tables 3, 4, respectively. Both extractors consist of four downsampling layers, each preceded by submanifold convolution layers. VoxelResBackBone incorporates two submanifold convolutional layers and a skip connection, forming a residual block, rather than a single submanifold convolutional layer. Our inversion attack model employs an identical structure for both feature extractors (i.e., symmetric with VoxelBackBone), considering the absence of spatial dispersion in submanifold convolutions.

Inversion attack model. Table 5 shows the structure of the point regression (PR) model. Basically, it is symmetrical to the VoxelBackBone feature extractor but output with three-dimensional channel because it predicts the x, y, and z coordinates excluding the intensity value. Conversely, the voxel occupancy classification (VOC), as shown in Table 6, outputs a one-dimensional channel because the occupancy of the voxel unit is classified in the final layer. Regarding *ConcreTizer* in Table 7, since it undergoes block-wise training through dispersion-controlled supervision (DCS), a classification layer is appended to each block. In this case, both classification and regression are performed together except for last block, as the intermediate layer's feature necessitates not only occupancy but also channel values.

F SUPPLEMENTARY EVALUATION

In this section, while the main paper already effectively conveys our message through its results, we aim to provide more detailed experimental outcomes and settings. This additional information offers deeper insights and a more comprehensive understanding of our research methodology and findings.

F.1 FURTHER DETAILS OF RESTORATION PERFORMANCE

To understand where the performance of *ConcreTizer* manifests, we delve into a detailed examination of the impact of VOC and DCS, the key components of *ConcreTizer*. Figures 3 and 4 illustrates the comparative performance of Point Regression, VOC, and *ConcreTizer* (VOC+DCS) across different depths of the feature extractor's layers. These results show that using only VOC significantly improves performance in all cases compared to conventional Point Regression. Incorporating DCS ensures sustained performance even with increased layer depth, particularly evident in metrics like CD and F1 score, where variance is reduced. This effectiveness stems from DCS's ability to efficiently mitigate the dispersion of VoI that arises with deeper layer configurations.

In an extension to the main paper, Table 8 provides a quantitative evaluation result for VoxelResBackBone. Figures 5, 6, 7, and 8 serve as visual aids to demonstrate the performance for VoxelBackBone and VoxelResBackBone on a wider array of example scenes from the KITTI and Waymo datasets, respectively. Each figure displays the restoration results from the final (12_{th}) layer. *ConcreTizer* demonstrates superior performance in restoring the overall shape when compared to VOC's restoration, which tends to be excessively clustered.

F.2 FURTHER DETAILS OF DCS INSTANCES

Section 5.5 covers an ablation study on the number of DCS instances. Figure 9 illustrates the restoration results for different numbers of DCS instances. An increasing number of DCS instances leads to a gradual accumulation of errors in partitions, resulting in a significant deterioration in restoration quality for ten instances. In contrast, *ConcreTizer*'s downsampling-based partitioning performs better by preventing VoI dispersion, which outweighs the cumulative error effect.

F.3 DCS OPTIMAL SPLIT POSITION

When employing DCS, a trade-off occurs at the split point: while DCS helps mitigate dispersion effects with additional supervision, it also risks accumulating restoration errors in the next block. Section 5.5 analyzes performance with respect to the number of DCS blocks, showing high performance with 2, 3, or 4 blocks. Here, we explore performance with different split positions for the 2-block and 4-block configurations.

Figure 10 shows the performance with two DCS blocks. Notably, split option 0 exhibited a significant performance drop compared to options 1 or 2. In less dispersed blocks (f12 ~f'9), the splitting effect is minimal, while in highly dispersed blocks (f'9 ~f'2), restoration without splitting proved to be challenging. The best performance was observed with split option 2, because supervision was effectively placed where dispersion effects were similar in blocks (f12 ~f'5) and (f'5 ~f'2). Our *ConcreTizer* (option 1) achieved balanced performance by evenly splitting based on downsampling layers. Further, in Figure 11, with four DCS blocks, options 0 and 1 displayed inferior performance due to uneven dispersion splitting. In contrast, our *ConcreTizer* (option 2) and options 3 and 4 achieved better results by appropriately distributing dispersion effects.

This suggests potential research avenues for finding optimal split positions. The randomization effects in 3D voxel data can be divided into two types: value randomization due to convolution filters and spatial randomization caused by downsampling layers. These effects may change depending on the dimension and sparsity of the input data. Therefore, modeling the randomization effects for each layer could enable future investigations into optimal split positions.

F.4 FURTHER DETAILS OF POINT CLOUD AUGMENTATION

In Section 5.6, we analyzed the trade-off between utility (3D object detection) and privacy (defense against inversion attacks) by applying point cloud augmentation techniques (Wang et al., 2024b; Li et al., 2021; Wang et al., 2024a). The results indicate that both rotation and scaling methods caused a sharp decline in utility. This is attributed to the distinct characteristics of the labels used in classification and object detection tasks.

For classification tasks, the label corresponds to the entire point cloud and represents the object's category. Transformations such as rotation or scaling do not alter the label, as they preserve the object's overall shape and category. Therefore, these augmentations can enhance the models' robustness and improve performance.

For object detection tasks, the labels not only include the object category but also precise location details, such as position, size, and orientation within a complex scene. When transformations like rotation or scaling are applied, the ground-truth location information in the labels must also be updated to reflect these changes. During training, this adjustment is feasible since the labels are available, enabling effective data augmentation. However, during test-time defenses against inversion attacks, label information is unavailable, and data perturbations occur without corresponding label updates. This mismatch between transformed data and unchanged labels leads to significant performance degradation. This effect is particularly pronounced for distant objects, where small augmentations like rotation and scaling can result in large distortions. In contrast, random sampling does not require label adjustments, as it does not alter the location-based information in the labels. Consequently, its impact on performance is relatively minor.

F.5 Further Details of Noise Effect

Section 5.6 explores the impact of various types of noise on feature restoration using the SECOND object detection model (Yan et al., 2018) and assesses object detection performance with these noise-added features. Additionally, Figure 12 illustrates the restoration results as noise levels vary, highlighting how the sparse nature of the 3D features leads to different impacts on restoration performance depending on the noise's location within the feature.

F.6 Potential Defense Strategies for Inversion Attack

To counter inversion attacks, several defense mechanisms have been proposed, each with unique strengths and limitations.

Differential privacy (DP) protects against privacy leakages by adding noise, such as Gaussian or Laplacian, based on a mathematically defined privacy budget (Abadi et al., 2016; Zhao & Chen, 2022). This approach provides robust protection against worst-case scenarios but excessively sacrifices utility. Recent advancements have integrated DP with generative models (Chen et al., 2021; Xue et al., 2023), achieving better privacy-utility trade-offs. However, these methods rely on separate generative models, introducing latency that makes them unsuitable for real-time applications.

Adversarial training improves model robustness by training alongside an attack model. Early methods (Raval et al., 2019; Wu et al., 2018) used generative models for obfuscation, but this led to inference overhead, which is impractical for latency-sensitive environments. More recent approaches (Liu et al., 2019) have proposed adversarial training without generative models, relying solely on the utility model. While this reduces inference overhead, it still requires retraining the feature extractor for a given attack model.

Feature obfuscation, among other approaches, reduces mutual information between raw data and feature data through loss functions (Zhang et al., 2022), offering a good balance between privacy and utility. However, this approach also requires managing both a utility model and an independent model, adding complexity to system management and maintenance.

Future research should focus on developing defense techniques that offer optimal privacy-utility trade-offs without sacrificing real-time performance, especially for latency-sensitive applications like autonomous driving.

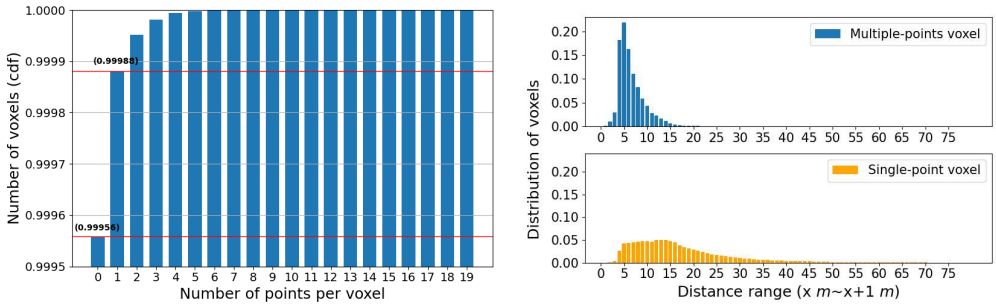

Figure 1: **(Left) Voxel distribution by point count:** The voxel distribution based on the number of points inside each voxel using a cumulative distribution function. **(Right) Non-empty voxel distribution:** The distribution of multiple-points and single-point voxels within the range of x to x+1 meters.

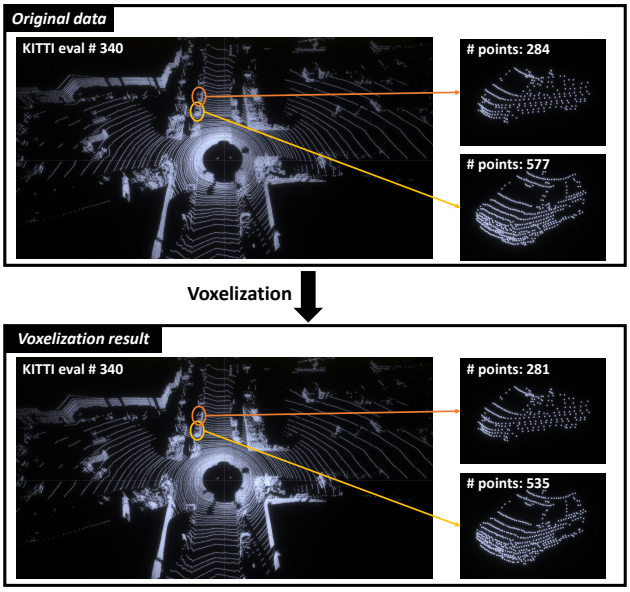

Figure 2: **Effect of voxelization process on point cloud.** The voxel size is 5 cm x 5 cm x 10 cm, and the maximum number of points per voxel is set as 5. Then points in each voxel are averaged to get a single representative value for each channel.

Table 1: α **values of SF loss for VoxelBackBone with KITTI and Waymo dataset.** *ConcreTizer* employs multiple blocks, each with a distinct α value.

# of Downsampling (LayerDepth)		**1 (3rd)**	**2 (6th)**	**3 (9th)**	**4 (12th)**
KITTI	VOC	0.7	0.75	0.8	0.825
	ConcreTizer	0.7	(0.7, 0.75)	(0.7, 0.75, 0.75)	(0.7, 0.75, 0.75, 0.75)
Waymo	VOC	0.6	0.6	0.72	0.75
	ConcreTizer	0.6	(0.7, 0.7)	(0.9, 0.7, 0.8)	(0.9, 0.85, 0.95, 0.95)

Table 2: α **values of SF loss for VoxelResBackBone with KITTI and Waymo dataset.** *ConcreTizer* employs multiple blocks, each with a distinct α value.

# of Downsampling (LayerDepth)		**1 (3rd)**	**2 (6th)**	**3 (9th)**	**4 (12th)**
KITTI	VOC	0.7	0.75	0.8	0.8
	ConcreTizer	0.7	(0.7, 0.8)	(0.7, 0.8, 0.75)	(0.7, 0.8, 0.75, 0.7)
Waymo	VOC	0.6	0.5	0.68	0.75
	ConcreTizer	0.6	(0.4, 0.4)	(0.5, 0.5, 0.6)	(0.65, 0.7, 0.8, 0.7)

Table 3: **Baseline 3D feature extractor (VoxelBackBone).**

Blocks	Layers	Output size (KITTI)	Output size (Waymo)
Input	Voxelization result	4×41×1600×1408	4×41×1504×1504
Down block 1	4×3×3×3, 16 16×3×3×3, 16 16×3×3×3, 32, stride 2,2,2, padding 1,1,1	32×21×800×704	32×21×752×752
Down block 2	32×3×3×3, 32 32×3×3×3, 32 32×3×3×3, 64, stride 2,2,2, padding 1,1,1	64×11×400×352	64×11×376×376
Down block 3	64×3×3×3, 64 64×3×3×3, 64 64×3×3×3, 64, stride 2,2,2, padding 0,1,1	64×5×200×176	64×5×188×188
Down block 4	64×3×3×3, 64 64×3×3×3, 64 64×3×1×1, 128, stride 2,1,1	128×2×200×176	128×2×188×188

Table 4: **3D feature extractor with residual blocks (VoxelResBackBone).**

Blocks	Layers	Output size (KITTI)	Output size (Waymo)
Input	Voxelization result	4×41×1600×1408	4×41×1504×1504
Down block 1	4×3×3×3, 16 [16×3×3×3, 16 16×3×3×3, 16] ×2 16×3×3×3, 32, stride 2,2,2, padding 1,1,1	32×21×800×704	32×21×752×752
Down block 2	[32×3×3×3, 32 32×3×3×3, 32] ×2 32×3×3×3, 64, stride 2,2,2, padding 1,1,1	64×11×400×352	64×11×376×376
Down block 3	[64×3×3×3, 64 64×3×3×3, 64] ×2 64×3×3×3, 128, stride 2,2,2, padding 0,1,1	128×5×200×176	128×5×188×188
Down block 4	[128×3×3×3, 128 128×3×3×3, 128] ×2 128×3×1×1, 128, stride 2,1,1	128×2×200×176	128×2×188×188

Table 5: **Inversion attack model with point regression (PR).**

Blocks	Layers	Output size (KITTI)	Output size (Waymo)
Input	Down block 4 result	128×2×50×44	128×2×50×50
Up block 4	128×3×1×1, 64, stride 2,1,1 64×3×3×3, 64 64×3×3×3, 64	64×5×50×44	64×5×50×50
Up block 3	64×3×2×2, 64, stride 2,2,2 64×3×3×3, 64 64×3×3×3, 64	64×11×100×88	64×11×100×100
Up block 2	64×2×2×2, 32, stride 2,2,2 32×3×3×3, 32 32×3×3×3, 32	32×21×200×176	32×21×200×200
Up block 1	32×2×2×2, 16, stride 2,2,2	16×41×400×352	16×41×400×400
Regression	16×3×3×3, 3	3×41×400×352	3×41×400×400

Table 6: **Inversion attack model with VOC.**

Blocks	Layers	Output size (KITTI)	Output size (Waymo)
Input	Down block 4 result	128×2×50×44	128×2×50×50
Up block 4	128×3×1×1, 64, stride 2,1,1 64×3×3×3, 64 64×3×3×3, 64	64×5×50×44	64×5×50×50
Up block 3	64×3×2×2, 64, stride 2,2,2 64×3×3×3, 64 64×3×3×3, 64	64×11×100×88	64×11×100×100
Up block 2	64×2×2×2, 32, stride 2,2,2 32×3×3×3, 32 32×3×3×3, 32	32×21×200×176	32×21×200×200
Up block 1	32×2×2×2, 16, stride 2,2,2	16×41×400×352	16×41×400×400
Classification	16×3×3×3, 1	1×41×400×352	1×41×400×400

Table 7: **Inversion attack model with VOC and DCS (*ConcreTizer*).**

Blocks	Layers	Output size (KITTI)	Output size (Waymo)
Input	Down block 4 result	128×2×50×44	128×2×50×50
Up block 4	128×3×1×1, 64, stride 2,1,1 64×3×3×3, 64 64×3×3×3, 64	64×5×50×44	64×5×50×50
Classification 4	64×3×3×3, 1	1×6×50×44	1×6×50×50
Up block 3	64×3×2×2, 64, stride 2,2,2 64×3×3×3, 64 64×3×3×3, 64	64×11×100×88	64×11×100×100
Classification 3	64×3×3×3, 1	1×11×100×88	1×11×100×100
Up block 2	64×2×2×2, 32, stride 2,2,2 32×3×3×3, 32 32×3×3×3, 32	32×21×200×176	32×21×200×200
Classification 2	32×3×3×3, 1	1×21×200×176	1×21×200×200
Up block 1	32×2×2×2, 16, stride 2,2,2	16×41×400×352	16×41×400×400
Classification 1	16×3×3×3, 1	1×41×400×352	1×41×400×400

Table 8: **Inversion attack result for VoxelResBackBone with KITTI and Waymo dataset.** Average CD and HD values in centimeters, and F1 scores with 15 cm and 30 cm thresholds for KITTI and Waymo datasets. Metrics evaluate over two datasets with 3769 and 3999 scenes, respectively.

	#Downsampling (LayerDepth)	1 (3rd)			2 (6th)			3 (9th)			4 (12th)		
		CD (↓)	HD (↓)	F1score (↑)	CD (↓)	HD (↓)	F1score (↑)	CD (↓)	HD (↓)	F1score (↑)	CD (↓)	HD (↓)	F1score (↑)
KITTI	Point Regression	1.2115	21.7866	0.3752	1.1540	31.3538	0.4101	2.9976	52.9479	0.2421	3.7381	55.6439	0.1483
	UltraLiDAR	0.0766	8.2591	0.9040	0.0773	8.0839	0.9054	0.0811	7.8901	0.8945	0.0977	**7.8521**	0.8329
	VOC (BCE loss)	**0.0318**	7.5409	**0.9918**	0.0368	7.5395	0.9907	0.1217	8.4344	0.8122	0.6315	23.0653	0.6012
	VOC	0.0319	**7.5384**	**0.9918**	**0.0349**	**7.5336**	**0.9917**	0.0490	**7.5900**	0.9645	0.1261	10.9786	0.8726
	ConcreTizer	0.0319	**7.5384**	**0.9918**	0.0367	**7.5336**	0.9913	**0.0478**	7.7806	**0.9801**	**0.0714**	9.5625	**0.9350**
Waymo	Point Regression	1.4991	54.7718	0.7556	2.0036	60.7505	0.7194	3.8474	70.1183	0.5761	4.5276	71.9906	0.4951
	UltraLiDAR	0.0840	11.0301	0.9735	0.0890	11.6088	0.9635	0.1009	11.6971	0.9503	0.1243	11.9076	0.9128
	VOC (BCE loss)	**0.0380**	10.2578	0.9981	**0.0445**	10.2615	**0.9981**	0.1038	11.9206	0.9150	0.5445	25.7258	0.6273
	VOC	**0.0380**	**10.2366**	**0.9983**	**0.0445**	10.2678	0.9980	0.0658	**10.5032**	0.9758	0.1384	14.6677	0.8946
	ConcreTizer	**0.0380**	**10.2366**	**0.9983**	0.0466	**10.2431**	**0.9981**	**0.0629**	10.6323	**0.9922**	**0.0946**	**11.6200**	**0.9479**

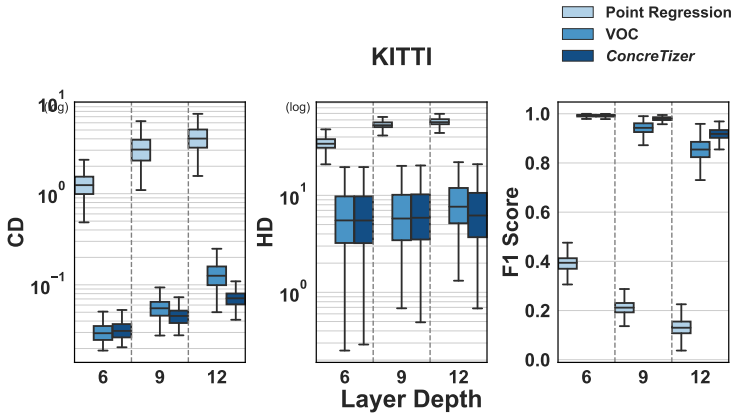

Figure 3: **Component-wise comparison with KITTI dataset.**

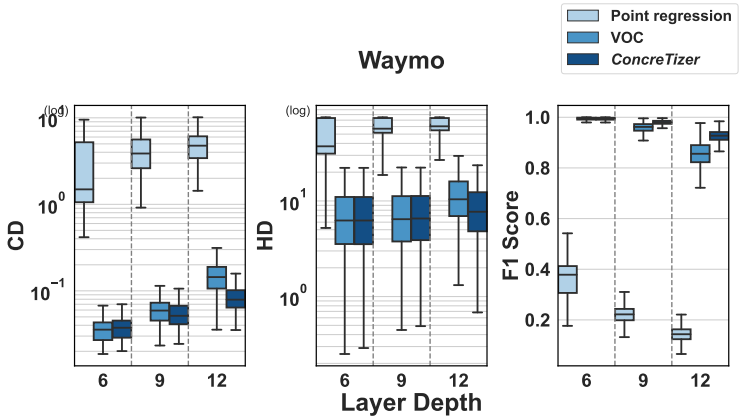

Figure 4: **Component-wise comparison with Waymo dataset.**

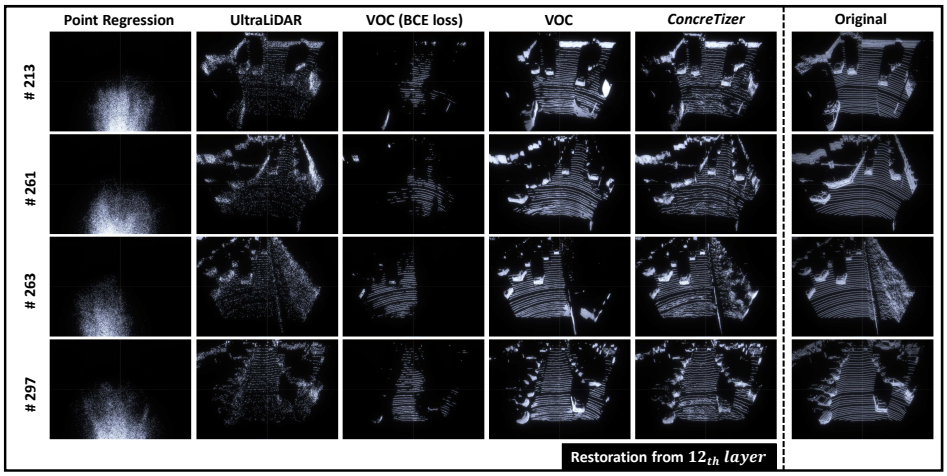

Figure 5: **Additional qualitative results for VoxelBackBone with KITTI dataset.** Each row presents the restoration result and corresponding original data for a specific KITTI validation scene. The input is the 12th (the final) layer.

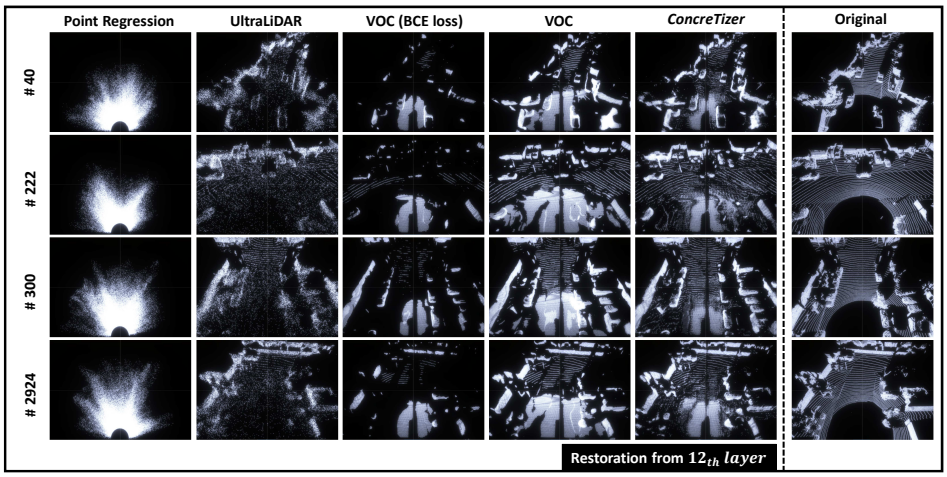

Figure 6: **Additional qualitative results for VoxelBackBone with Waymo dataset.** Each row presents the restoration result and corresponding original data for a specific Waymo validation scene. The input is the 12th (the final) layer.

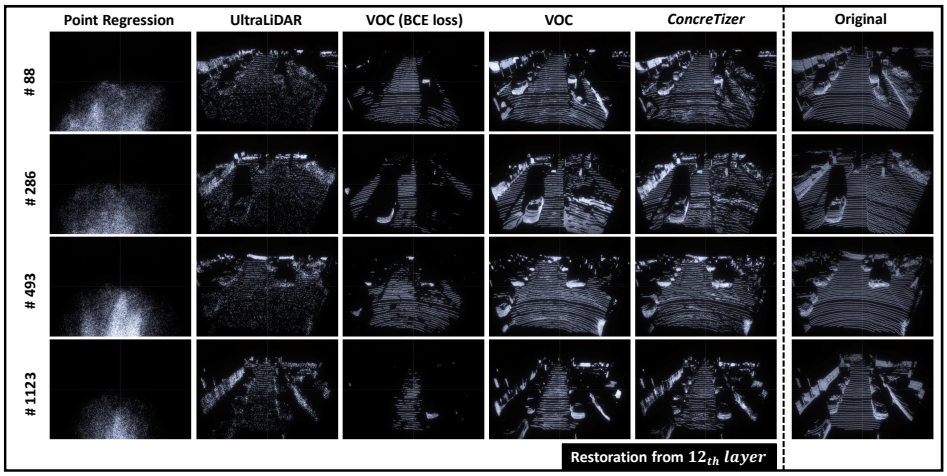

Figure 7: **Additional qualitative results for VoxelResBackBone with KITTI dataset.** Each row presents the restoration result and corresponding original data for a specific KITTI validation scene. The input is the 12th (the final) layer.

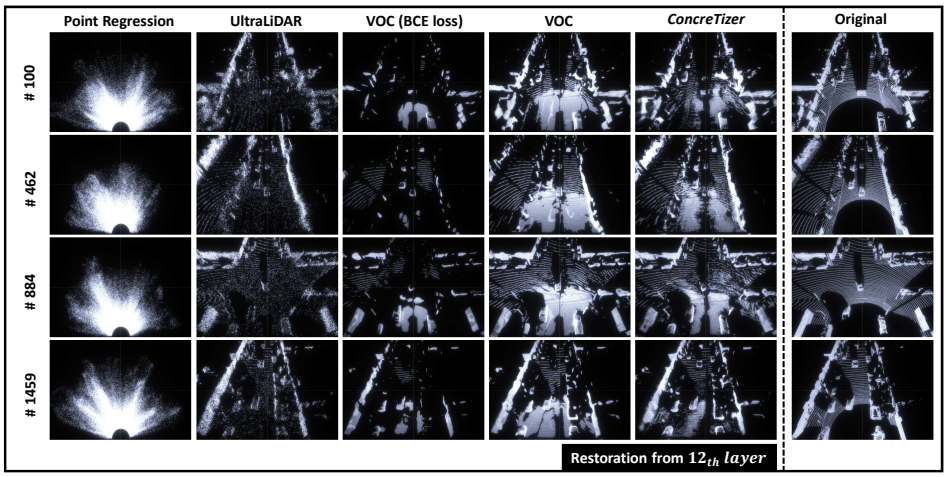

Figure 8: **Additional qualitative results for VoxelResBackBone with Waymo dataset.** Each row presents the restoration result and corresponding original data for a specific Waymo validation scene. The input is the 12th (the final) layer.

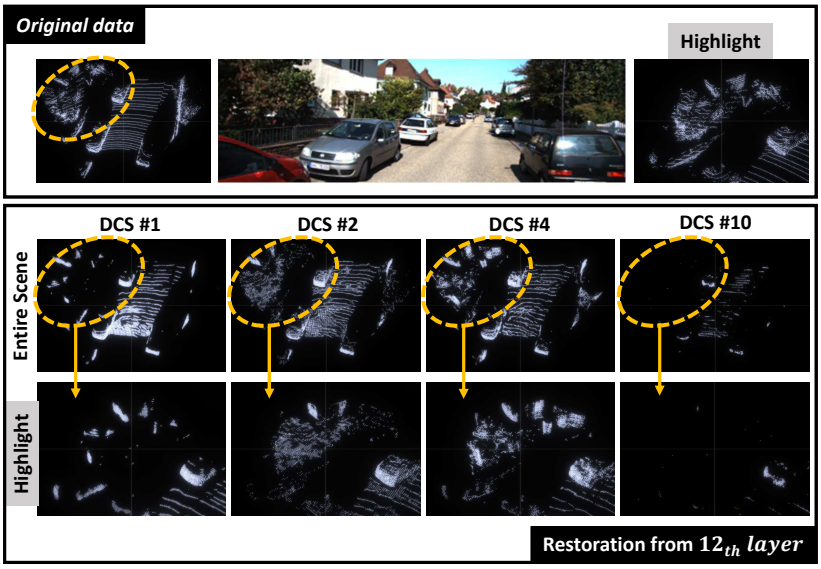

Figure 9: **Qualitative result for different DCS instances with** *ConcreTizer* **model.**

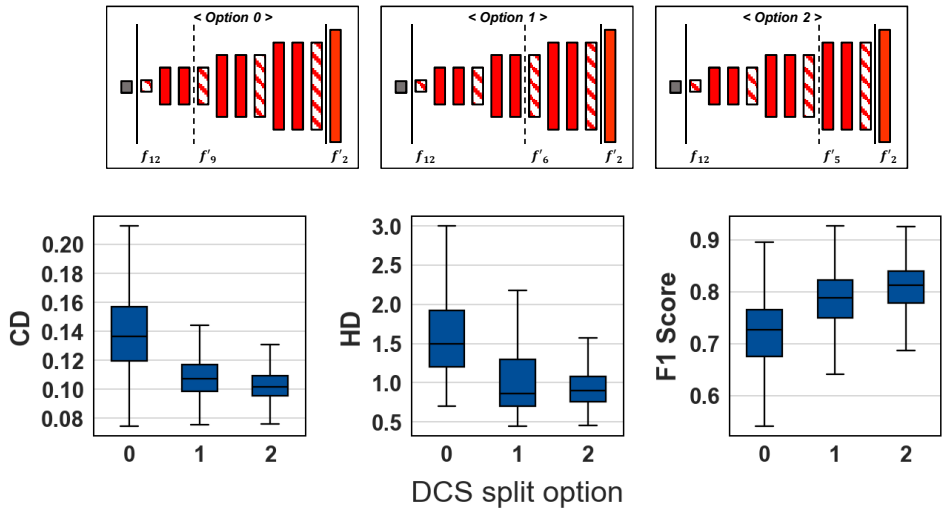

Figure 10: **DCS #2 split option 0 to 2.**

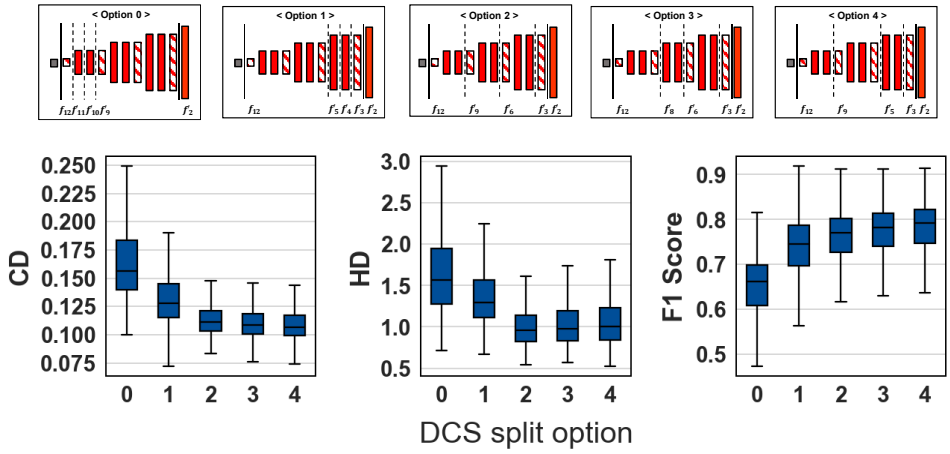

Figure 11: **DCS #4 split option 0 to 4.**

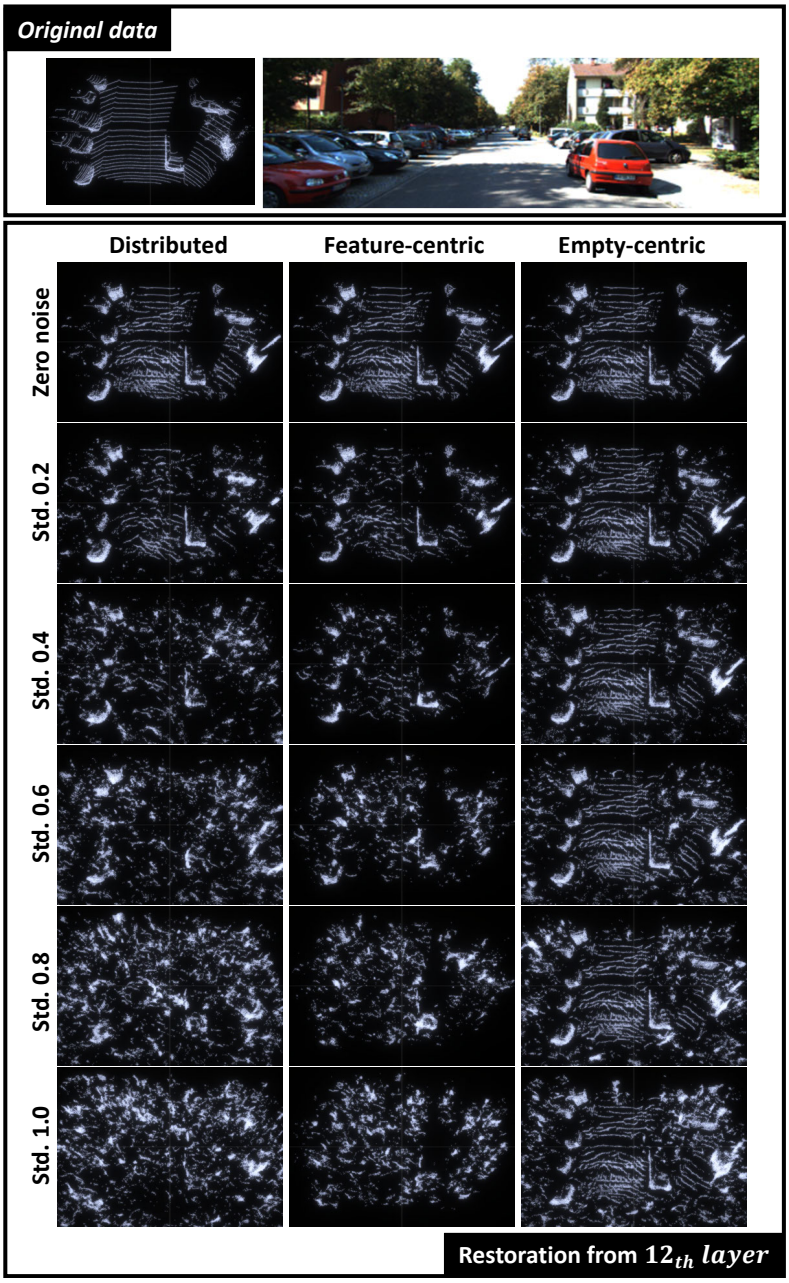

Figure 12: **Qualitative results for different noise levels.**

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Jia-Wei Chen, Li-Ju Chen, Chia-Mu Yu, and Chun-Shien Lu. Perceptual indistinguishability-net (pi-net): Facial image obfuscation with manipulable semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6478–6487, 2021.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361. IEEE, 2012.

Benjamin Graham and Laurens Van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

Xinke Li, Zhirui Chen, Yue Zhao, Zekun Tong, Yabang Zhao, Andrew Lim, and Joey Tianyi Zhou. Pointba: Towards backdoor attacks in 3d point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16492–16501, 2021.

Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong. Privacy adversarial network: representation learning for mobile data privacy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(4):1–18, 2019.

Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2019.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

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

Xianlong Wang, Minghui Li, Wei Liu, Hangtao Zhang, Shengshan Hu, Yechao Zhang, Ziqi Zhou, and Hai Jin. Unlearnable 3d point clouds: Class-wise transformation is all you need. In *In Advances in Neural Information Processing Systems (NeurIPS), 2024*, 2024a.

Xianlong Wang, Minghui Li, Peng Xu, Wei Liu, Leo Yu Zhang, Shengshan Hu, and Yanjun Zhang. Pointapa: Towards availability poisoning attacks in 3d point clouds. In *European Symposium on Research in Computer Security*, pp. 125–145. Springer, 2024b.

Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. In *In Advances in Neural Information Processing Systems (NeurIPS), 2021*, 2021.

Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 606–624, 2018.

Hanyu Xue, Bo Liu, Ming Ding, Tianqing Zhu, Dayong Ye, Li Song, and Wanlei Zhou. Dp-image: Differential privacy for image data in feature space, 2023.

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

Jiang Zhang, Lillian Clark, Matthew Clark, Konstantinos Psounis, and Peter Kairouz. Privacy-utility trades in crowdsourced signal map obfuscation. *Computer Networks*, 215:109187, 2022.

Ying Zhao and Jinjun Chen. A survey on differential privacy for unstructured data content. *ACM Computing Surveys (CSUR)*, 54(10s):1–28, 2022.

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4490–4499, 2018.