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

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing use of 3D point cloud data in autonomous vehicles (AVs) has raised serious privacy concerns, particularly due to the sensitive information that can be extracted from 3D data. While model inversion attacks have been widely studied in the context of 2D data, their application to 3D point clouds remains largely unexplored. To fill this gap, we present the first in-depth study of model inversion attacks aimed at restoring 3D point cloud scenes. Our analysis reveals the unique challenges, the inherent sparsity of 3D point clouds and the ambiguity between empty and non-empty voxels after voxelization, which are further exacerbated by the dispersion of non-empty voxels across feature extractor layers. To address these challenges, we introduce *ConcreTizer*, a simple yet effective model inversion attack designed specifically for voxel-based 3D point cloud data. *ConcreTizer* incorporates Voxel Occupancy Classification to distinguish between empty and non-empty voxels and Dispersion-Controlled Supervision to mitigate non-empty voxel dispersion. Extensive experiments on widely used 3D feature extractors and benchmark datasets, such as KITTI and Waymo, demonstrate that *ConcreTizer* concretely restores the original 3D point cloud scene from disrupted 3D feature data. Our findings highlight both the vulnerability of 3D data to inversion attacks and the urgent need for robust defense strategies.

1 INTRODUCTION

Recent advancements in Autonomous Vehicles (AVs) have underscored the importance of continuous vision data collection and sharing. At the same time, AI and big data have amplified privacy concerns, prompting increased research on this issue (Guo et al., 2017; Stahl & Wright, 2018). Consequently, AV's data collection faces strict regulations that requires data de-identification (Mulder & Vellinga, 2021). For example, the EU's General Data Protection Regulation (GDPR) (EU, 2016) mandates businesses to adopt stringent data protection protocols.

Beyond these regulations, the need for privacy preservation is rapidly increasing, particularly in 3D point cloud data. This is because various types of privacy-related information can be revealed through rich 3D shape information. For instance, personal information can be identified through facial recognition (Zhang et al., 2019) and person re-identification (Cheng & Liu, 2021). Furthermore, behavioral patterns can be exposed through human pose estimation (Zhou et al., 2020) and activity recognition (Singh et al., 2019b). Additionally, by using methods like Simultaneous Localization and Mapping (SLAM) (Kim et al., 2018), location information can also be inferred. Moreover, the fact that 2D images can be reconstructed from sparse 3D data (Pittaluga et al., 2019; Song et al., 2020) emphasizes the importance of securing raw 3D point data from the outset.

However, research on privacy in 3D point cloud data remains significantly underexplored compared to advancements in the 2D image domain. A prominent research area in 2D image privacy is inversion attacks. While earlier studies (Gupta & Raskar, 2018; Vepakomma et al., 2018; Singh et al., 2019a) suggested that 2D images could be anonymized by extracting features, inversion attacks have demonstrated that these features can be used to restore the original 2D images. In contrast, while there have been a few prior studies on privacy of 3D data (Wang et al., 2024), inversion attacks for 3D data remain largely unexplored. This research gap allowed a recent study (Hwang et al., 2023) to operate under the assumption that disseminating 3D features inherently prevents the restoration of the original dataset. In the absence of existing inversion attack methods for 3D data, the authors developed a

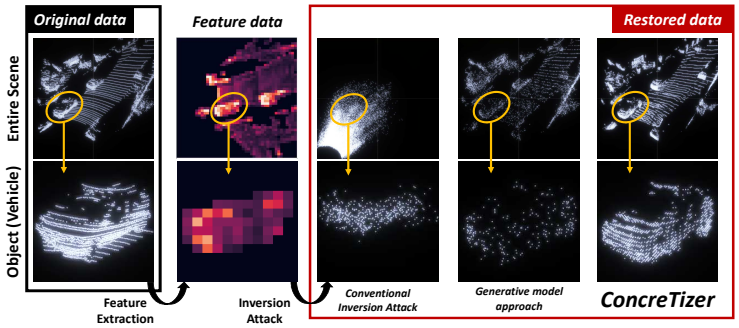


Figure 1: **Restoration result of a 3D Point Clouds**. Feature data is extracted from original point cloud through a 3D feature extractor (Yan et al., 2018). *ConcreTizer* (right) enables restoration with simple modifications to conventional approach (left), and even achieves more concrete restoration than generative model approach (middle) (Xiong et al., 2023).

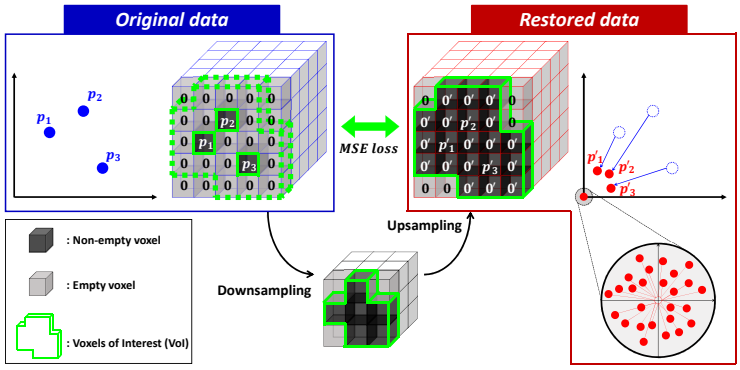


Figure 2: **Restoration through conventional inversion attack method.** Voxelization introduces zero-padding to non-empty voxels. During downsampling in feature extraction, these voxels spread to neighboring areas, expanding the VoI (green region). Within the VoI, voxel-wise channel regression generates additional points in zero-padded regions, leading to clustering near the origin.

point regression method to invert voxel-based backbones, aiming to demonstrate that restoring the original 3D scene from its extracted features is infeasible. As in Figure 1, the conventional point regression in (Hwang et al., 2023) fails to restore 3D point cloud data from intermediate features.

However, we argue that this failure is not due to an inherent safety of 3D features but rather a lack of careful design that considers the characteristics of 3D backbones. To address this issue, Figure 2 examines the phenomena arising when the point regression method inverts voxel-based feature extractors, which are dominant architectures in autonomous driving applications. The point regression approach attempts to directly restore point coordinates within each voxel by minimizing mean squared error (MSE). However, the method relies on a naive determination of point existence in each voxel, which introduces critical issues. Specifically, during the voxelization process, empty voxels are zero-padded. If a voxel's regression output is exactly (0, 0, 0), it indicates that the zero-padded voxel remains unaffected throughout the forward and inversion processes. These voxels are treated as empty regions and excluded from the MSE calculation. Conversely, if a voxel's output deviates from (0, 0, 0), it is included in the MSE calculation (referred to as a Voxel of Interest (VoI)) and is used to generate a valid point in the restored scene.

The problem is as follows: The sparsity of 3D point cloud data results in a large number of zero-padded voxels. During the forward and inversion processes, particularly in deeper feature extractors, non-empty voxels (VoIs) spread into these empty voxels. This dispersion leads to a proliferation of false VoIs (originally empty voxels), causing point regression to erroneously generate points in regions that were initially void. Moreover, these false VoIs disproportionately impact the MSE loss, prompting the point regression model to bias the restoration by concentrating most points near the origin (0, 0, 0) to minimize estimation errors for the false VoIs. This bias significantly degrades localization performance for the relatively smaller number of true VoIs (originally non-empty voxels).

The analysis reveals that the key to a successful inversion attack is not restoring the representation of the voxel (i.e., point coordinates) but accurately determining whether a voxel was originally

empty or non-empty. Once this classification is achieved, localizing points within non-empty voxels becomes more straightforward, as the error is constrained by the voxel size, which is typically small. Based on this insight, we transform the conventional point regression problem into a more explicit Voxel Occupancy Classification (VOC) problem. In addition, the spread of VoI should be suppressed during restoration to minimize the negative impact of false VoIs. To address this, our model incorporates Dispersion-Controlled Supervision (DCS), which segments the feature extractor based on downsampling layers and trains each segment individually, proactively controlling the dispersion of VoI. Thanks to its tailored design, our model, called *ConcreTizer*, outperforms conditioned generation methods that use generative models (see Figure 1, the generative model approach (Xiong et al., 2023)).

To demonstrate the general applicability of *ConcreTizer*, we deployed it on two representative 3D feature extractors (Yan et al., 2018; Lu et al., 2022), which are essential components in various applications including 3D object detection (Yan et al., 2018; Shi et al., 2020), 3D semantic segmentation (Wu et al., 2019; Thomas et al., 2019), and tracking (Yin et al., 2021). Our experiments conducted on prominent datasets, KITTI (Geiger et al., 2012) and Waymo (Sun et al., 2020), confirm that *ConcreTizer* consistently outperforms across various datasets and 3D feature extractors. We showcase the superior performance of *ConcreTizer* through a range of quantitative and qualitative evaluations, employing point cloud similarity metrics, visual aids, specific-task (3D object detection) performance using restored scenes, and the performance of potential defense mechanisms.

The contributions of this paper are as follows:

- This work is the first to conduct an in-depth study of model inversion attacks aimed at restoring voxel-based 3D point cloud scenes. Our analysis identifies unique challenges for inversion attacks, which arise from interaction between sparse point cloud data and voxel-based feature extractors.
- To address these unique challenges, we propose *ConcreTizer*, tailored for inverting 3D backbone networks, incorporating Voxel Occupancy Classification and Dispersion-Controlled Supervision.
- Through extensive experiments with representative 3D feature extractors and well-established open-source datasets, we demonstrate the quantitative and qualitative effectiveness of *ConcreTizer*.

2 Related Work

3D Point Clouds Feature Extraction. Feature extractors for 3D point cloud data encompass set, graph, and grid-based approaches, each distinguished by its representation format. The computational complexity of set and graph-based methods (Qi et al., 2017; Kipf & Welling, 2016) scales significantly with the number of points, limiting their use in real-time applications like autonomous driving. Conversely, grid-based methods (Zhou & Tuzel, 2018; Yan et al., 2018; Shi et al., 2020; Lang et al., 2019; Sun et al., 2022) organize the 3D space into a grid and apply sparse convolution (Liu et al., 2015; Graham & Van der Maaten, 2017) for efficient feature extraction from sparse voxel data. These methods are particularly advantageous for autonomous driving applications due to their efficiency with sparse data. Considering these characteristics, we explore inversion attacks tailored to scenarios using grid-based feature extractors.

Model Inversion. Model inversion was initially studied from the perspective of interpretability in deep learning models. Traditional approaches generate saliency maps to understand how models produce outputs (Du et al., 2018). Other methods (Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016b;a) reconstruct the input from intermediate features to analyze the information flow through model layers. Recently, with growing concerns about data privacy, model inversion has gained attention as a privacy attack. Early studies attempted to restore input face images from confidence scores (Yang et al., 2019b). Subsequent studies (Zhang et al., 2020; Zhao et al., 2021) leverage additional information for more sophisticated restoration. Based on these studies, corresponding defense techniques (Liu et al., 2019; Xue et al., 2023; Dusmanu et al., 2021; Ng et al., 2022; Zhang et al., 2022) have also been investigated, enriching the exploration of data privacy. However, previous work has primarily focused on 2D image data. There is a clear need for an inversion attack technique that accounts for the unique characteristics of 3D point cloud data in autonomous driving. To the best of our knowledge, this research is the first to study inversion attacks for 3D data.

Point Cloud Generation. Generative models, with their ability to produce plausible raw data, are widely used for various tasks. In the 3D point cloud domain, diverse generative models are being researched. Unconditional generation tasks create plausible shapes from random inputs like

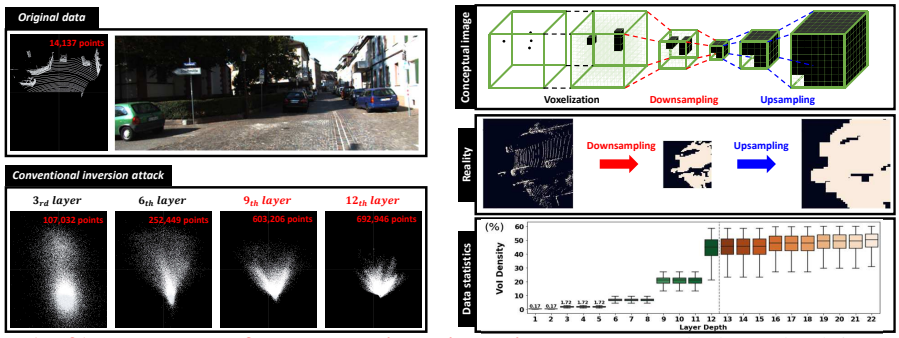


Figure 3: **(Left) The results of the conventional inversion attack:** As the layer depth increases, the number of restored points increases rapidly, and the concentration of points near the origin becomes more noticeable. **(Right) The VoI (Voxel of Interest) dispersion effect:** The non-empty voxels spread as they pass through the feature extractor and inversion attack model.

noise (Achlioptas et al., 2018; Valsesia et al., 2018; Shu et al., 2019; Yang et al., 2019a; Luo & Hu, 2021). Conditional generation tasks involve generating the remaining part from a partial point cloud (Yu et al., 2021; Huang et al., 2020; Wen et al., 2020) or generating a 3D point cloud from a 2D image (Mandikal et al., 2018; Mandikal & Radhakrishnan, 2019; Melas-Kyriazi et al., 2023). However, most existing research focuses only on dense point cloud data for single objects (e.g., Chang et al. (2015)). In contrast, there is little research on handling scene-level sparse point clouds captured from autonomous vehicles. Only a few studies (Caccia et al., 2019; Zyrianov et al., 2022) deal with scene-level sparse point clouds. However, even these studies require specific representation formats and do not support using 3D grid-shaped features, given in our inversion attack scenario, as conditions. To our knowledge, the only scene-level sparse point cloud generation model based on 3D grid representations is Xiong et al. (2023). We also conducted performance comparisons with conditional generation approach using this generative model.

3 Preliminary: Limitations of Conventional Inversion Attack

The only known attempt at an inversion attack on 3D point cloud data is by Hwang et al. (2023). Even this research does not directly focus on inversion attacks but rather seeks to assess the privacy protection effectiveness of 3D features by developing a sample inversion attack based on point regression. Before designing our method, we explore why the conventional approach can not effectively restore 3d point cloud scenes (Figure 3, left).

Firstly, we identified an issue in voxel-based models related to the voxelization process. During voxelization, regions without points are zero-padded. However, the conventional regression method does not consider point existence but focus solely on point localization, mistakenly interpreting zero-padded representations as valid points located at (0, 0, 0). As a result, points are created even for empty voxels, leading to an overgeneration of points compared to the original data. Secondly, the inherently sparse nature of point clouds results in a large number of zero-padded voxels, far exceeding those containing valid points. Point regression-based inversion attacks naturally prioritize minimizing estimation errors across all voxels, including these zero-padded regions. Consequently, this bias towards zero-padded voxels causes an over-concentration of points near the origin in the restored scene. Moreover, it significantly increases localization errors for the relatively smaller number of valid points, as these errors are considered negligible within the overall regression error.

Lastly, we observed that as the feature extractor layers deepen, existing attack methods are increasingly hindered by the negative impact of zero-padded voxels: (1) an excessive number of restored points and (2) an intensified concentration of points near the origin. Specifically, if voxels with a value of (0, 0, 0) persist in the final restored state, they are excluded from the regression targets and do not directly affect the regression loss. However, due to the nature of convolution operations, the values of non-empty voxels—defined as Voxels of Interest (VoI)—gradually disperse into the surrounding empty voxels. As the layers deepen, more originally zero-padded voxels become non-empty during the feature extraction and inversion processes. Consequently, an increasing number of these zero-padded voxels are included in the regression targets. Our experiments revealed that the density of VoI spikes significantly at downsampling layers (Figure 3, right), further amplifying the influence of zero-padded voxels on the final restoration results.

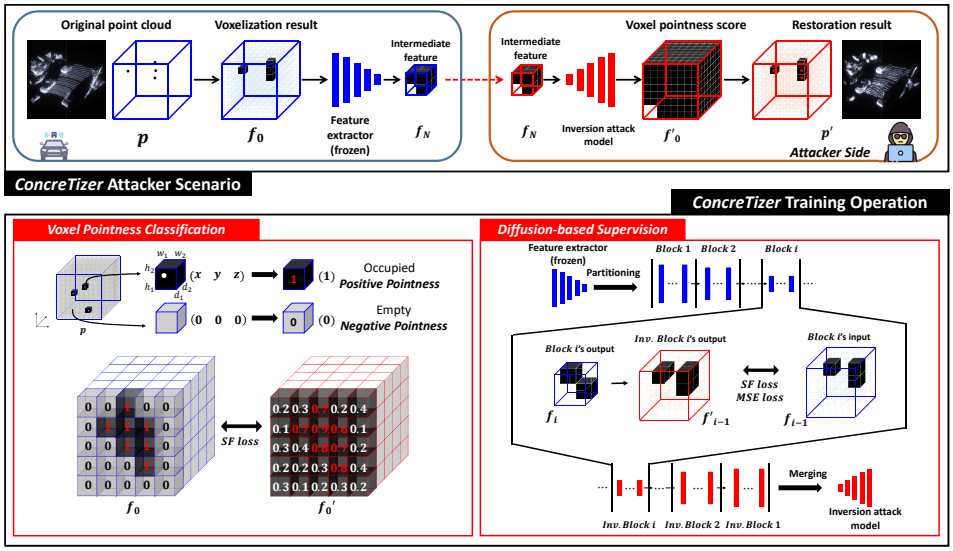


Figure 4: *ConcreTizer* **framework.** Original point cloud and feature data are represented by p and f_i, while their restored counterparts are denoted by p' and f'_i, respectively. The value i means that it is obtained from the i-th downsampling layer. *ConcreTizer* restores data by classifying the occupancy of f_0 and then placing points at the center of each restored voxel. For deeper layers, it employs partitioning at the downsampling layer and restores f_{i-1} from f_i for each block.

4 Proposed Method

4.1 AV Scenario

We focus on autonomous vehicle (AV) scenarios because they carry a high risk of exposure to inversion attacks. In AV contexts, there is a need to share feature data for purposes such as computation offloading (Xiao et al., 2022; Hanyao et al., 2021), model enhancement (Hwang et al., 2023), and cooperative perception (Wang et al., 2020; Xu et al., 2022; Yu et al., 2022). Specifically, we selected voxel-based feature extractors as the target for inversion attacks. Voxel-based feature extractors are commonly used in autonomous vehicles due to their suitability for real-time applications like 3D object detection (Yan et al., 2018; Lang et al., 2019; Shi et al., 2019; 2020; Shi & Rajkumar, 2020), semantic segmentation (Wu et al., 2019; Thomas et al., 2019), and tracking (Yin et al., 2021). In this scenario, an attacker with access to the same feature extractor would need to prepare 3D point cloud data for training the inversion attack model. Since the restoration task doesn't require separate labeling, training can be conducted using open-source datasets or self-collected data.

4.2 Problem Definition

The goal of an inversion attack is to discover the inverse process of a given feature extractor. For voxel-based feature extractors, the initial step involves a voxelization process that transforms point cloud data into a grid format.

Voxelization converts a 3D point cloud $p \in \mathbb{R}^{k\times 3}$, where k is the number of points, into a voxel grid $f_0 \in \mathbb{R}^{3\times H\times W\times D}$, where H, W, D represent the spatial dimensions of the grid. Here x, y, and z coordinate information is channelized and voxels without points are zero-padded, resulting in channel values of (0, 0, 0). In particular, during the downsampling process, the spatial dimensions shrink while the channel size increases, producing features $f_N \in \mathbb{R}^{C_N\times h_N\times w_N\times d_N}$, where N is the number of downsampling layers, $C_N > 3$, and h_N, w_N, d_N are smaller than H, W, D. Consequently, our inversion attack aims to restore the original voxel grid f_0 from the downsampled features f_N.

4.3 ConcreTizer Framework

Figure 4 depicts the overall *ConcreTizer* framework incorporating the scenario and attacker-side training operations. In designing the structure of the inversion attack model, we chose a symmetrical design with the feature extractor, following previous studies (Yang et al., 2019b; Zhang et al., 2020; Zhao et al., 2021). In other words, original shape can be restored by performing upsampling at the positions where downsampling occurs (detailed structure in the supplementary material). Building upon symmetric structure, *ConcreTizer* applies Voxel Occupancy Classification (VOC) and

Dispersion-Controlled Supervision (DCS) to overcome the limitations of traditional inversion attack. VOC transforms the regression problem into a classification problem to address the issue of point clustering near the origin. DCS prevents the dispersion of VoI by splitting the feature extractor, mitigating the degradation of restoration performance as the network deepens.

4.3.1 VOXEL OCCUPANCY CLASSIFICATION

In traditional inversion attack methods, the original data is directly restored through regression on channel values. In our scenario, since the x, y, and z coordinates are channelized during the voxelization process, performing regression would restore coordinate values. However, since voxelization of sparse point clouds produces a large number of zero-padded voxels with (0, 0, 0) channel value, many unnecessary points cluster near the origin in the inversion attack results (Figure 3, left). To address this issue, we transform the regression problem into a classification problem to resolve the semantic ambiguity of zero-padded voxels—whether they represent empty voxels or valid points at coordinates (0, 0, 0). This can be achieved through simple binary encoding, where each voxel is labeled as 0 (*negative occupancy*) or 1 (*positive occupancy*), making the meaning of zero-padding clear. Using the VOC method, the inversion attack model outputs binary classification scores in the form of $\mathbb{R}^{1\times H\times W\times D}$, rather than continuous coordinate values in the form of $\mathbb{R}^{3\times H\times W\times D}$. If a voxel is determined to contain a point, the corresponding coordinate can be restored easily. This is because the range of coordinate values is bounded by the spatial location of the voxel, and the voxel size is typically small enough. As a result, by using the center coordinates of the voxel, we achieve effective restoration within an error range constrained by the voxel size.

Additionally, due to the sparsity of original point cloud data, the binary-encoded labels have a higher ratio of 0s compared to 1s. This phenomenon is particularly exacerbated as the depth of the layers increases. Let f_0 be the original voxelized point cloud and f_0' be the restored one by the inversion attack. The number of positive labels is fixed as $|\text{VoI of } f_0|$, while the number of negative labels, $|\text{VoI of } f_0'| - |\text{VoI of } f_0|$, increases exponentially as the depth of the layer increases. To account for this imbalance, we apply the Sigmoid Focal (SF) loss (Lin et al., 2017), a variant of the conventional cross-entropy loss. The mathematical representation of the SF loss is given by:

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t), \tag{1}$$

where p_t denotes the model's predicted probability for the target class. The factor α_t is employed to adjust the importance given to the positive and negative classes.

4.3.2 DISPERSION-CONTROLLED SUPERVISION

While applying SF loss in VOC can partially address the label imbalance issue, it cannot prevent the problem of VoI dispersion. The original data is sparse with many empty voxels. However, as we observed earlier, the VoI density exponentially increases during the downsampling process (Figure 3, right). Therefore, after VoI spreads too much at the deep layer, it becomes difficult to restore the data back to the original sparse state.

Our proposed DCS is a more fundamental solution to address VoI dispersion. It divides the feature extractor into multiple blocks and performs restoration progressively. First, feature extractor can be partitioned based on the downsampling layer where the VoI spread occurs. Through this, the VoI dispersion in each block can be effectively controlled. Then, in the inversion attack model, an inversion block corresponding to each block can be created to train the restoration in block units. Note that at the original voxel level, the channel values represent point coordinates, eliminating the need for regression (if classification result is positive, channel value is estimated as the center coordinate of the voxel). However, at the intermediate feature level, normalization is applied, so both classification and regression on the channel values are required.

For example, if the input of the $(i+1)$-th block is $f_i \in \mathbb{R}^{C_i\times h_i\times w_i\times d_i}$ and the output is $f_{i+1} \in \mathbb{R}^{C_{i+1}\times h_{i+1}\times w_{i+1}\times d_{i+1}}$, then the $(i+1)$-th inversion block in the inversion attack model takes f_{i+1} as input and produces $f_i' \in \mathbb{R}^{C_i\times h_i\times w_i\times d_i}$, which is the result of restoring f_i. Specifically, $m_i' \in \mathbb{R}^{1\times h_i\times w_i\times d_i}$ (spatial occupancy scores found by applying SF loss) and $c_i' \in \mathbb{R}^{C_i\times h_i\times w_i\times d_i}$ (channel values found by applying L2 loss) are derived from f_{i+1}. Then, c_i' is masked by using m_i' to generate f_i'. In the masking process, unnecessary VoI values are erased, so the dispersion of VoI can be suppressed. Note that in the first inversion block, which is the final stage of the inversion attack model, only classification is performed, with no additional regression. The loss function for each inversion block is:

Table 1: **Inversion attack result 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 scene sets 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	1.3868	23.5855	0.3543	1.2879	34.2395	0.3904	3.1229	54.0173	0.2110	4.1439	56.9811	0.1298
	UltraLiDAR	0.0744	8.2269	0.9122	0.0818	8.0974	0.8905	0.0836	7.9561	0.8869	0.1012	**7.9185**	0.8152
	ConcreTizer	**0.0321**	**7.5603**	**0.9918**	**0.0373**	**7.5249**	**0.9914**	**0.0507**	**7.8453**	**0.9793**	**0.0776**	8.1193	**0.9160**
Waymo	Point Regression	1.4979	55.6589	0.7644	2.7733	66.7899	0.6489	4.1053	70.6608	0.5524	4.9340	71.9608	0.4355
	UltraLiDAR	0.0810	10.9582	0.9742	0.0898	11.3360	0.9623	0.1017	11.4987	0.9503	0.1378	12.0259	0.8849
	ConcreTizer	**0.0374**	**10.2544**	**0.9984**	**0.0466**	**10.2326**	**0.9979**	**0.0712**	**10.5724**	**0.9781**	**0.1087**	**11.3399**	**0.9251**

$$Loss(\text{inversion block } i+1) = \begin{cases} L_{\text{cls}} & \text{if } i = 0, \\ L_{\text{cls}} + \beta \cdot L_{\text{reg}} & \text{if } i \geq 1. \end{cases}$$

$$L_{\text{cls}} = \sum \text{SF loss}(m_i, m_i') \quad \text{and} \quad L_{\text{reg}} = \sum \text{L2 loss}(c_i, c_i')$$

where the summation is taken over VoI of the output.

$$\begin{pmatrix} m_i & : \text{Ground Truth spatial occupancy mask} \\ m_i' & : \text{Predicted spatial occupancy scores} \\ c_i & : \text{Ground Truth channel values} \\ c_i' & : \text{Predicted channel values} \end{pmatrix}$$

The final result of passing through all inversion blocks is a binary classification scores in the form of $\mathbb{R}^{1 \times H \times W \times D}$, and the restoration is completed by generating a point at the center of the voxel corresponding to the positive occupancy.

5 Experiments

5.1 Experimental Setup

3D Feature Extractor. We employ a voxelization-based 3D feature extractor as the target of our inversion attack. Based on the OpenPCDet (Team, 2020) project, we utilize pre-trained Voxel-Backbone (Yan et al., 2018) and VoxelResBackbone (Lu et al., 2022), extensively used in a crucial application area for 3D point cloud data. The VoxelBackBone structure contains four downsampling layers (i.e., $N = 4$), each preceded by two convolutional layers, while the VoxelResBackbone incorporates additional convolutional layers and skip connections.

Inversion Model Training. We train the inversion attack model on the real-world KITTI (Geiger et al., 2012) and Waymo (Sun et al., 2020) datasets. When using the SF loss function in VOC, only the alpha value in the hyperparameters is adjusted, and in DCS, the weight on regression loss, β, is set to 1.

Metrics. In our analysis of 3D scene restoration performance, we utilize various metrics. First, for qualitative analysis, we visualize the 3D point cloud using KITTI viewer web tool. Next, for quantitative analysis, we use point cloud similarity metrics such as Chamfer Distance (CD) (Borgefors, 1984), Hausdorff Distance (HsD) (Huttenlocher et al., 1993), and F1 Score (Goutte & Gaussier, 2005). Furthermore, to verify the utility of the restored data, we also check the 3D object detection accuracy with pre-trained detection models.

5.2 Restoration Performance

Comparison Schemes. We conducted comparisons in various ways to confirm the superiority of *ConcreTizer*. We start with Point Regression (Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016a;b), a traditional inversion attack method. In Point Regression, post-processing is applied to eliminate points that fell outside the defined point cloud range or were excessively concentrated near the origin. Next we compare *ConcreTizer* with a generative model approach. To perform an inversion attack using a generative model, a conditional generation with feature data as a condition must be employed. Among the existing LiDAR point cloud generation models, UltraLiDAR (Xiong et al., 2023) is the only model based on voxel representation, similar to our feature extractor. We modified the encoder part of UltraLiDAR to accept voxel features as an input and then trained it with the KITTI and Waymo datasets.

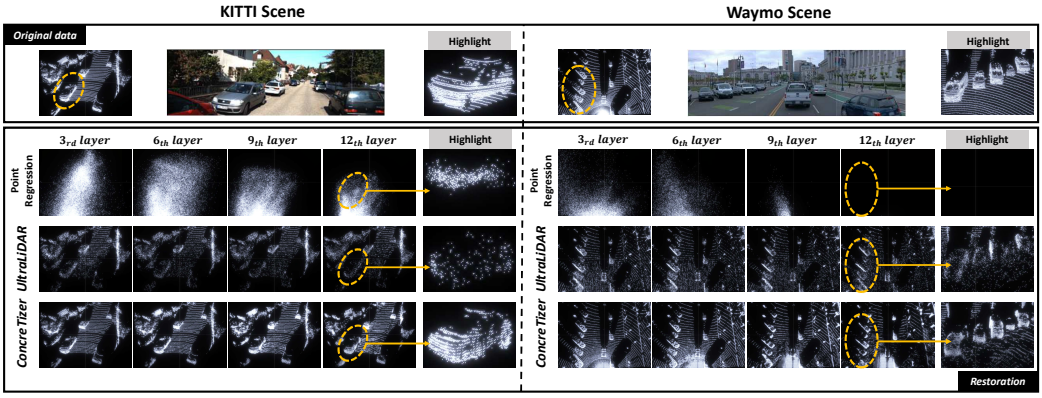
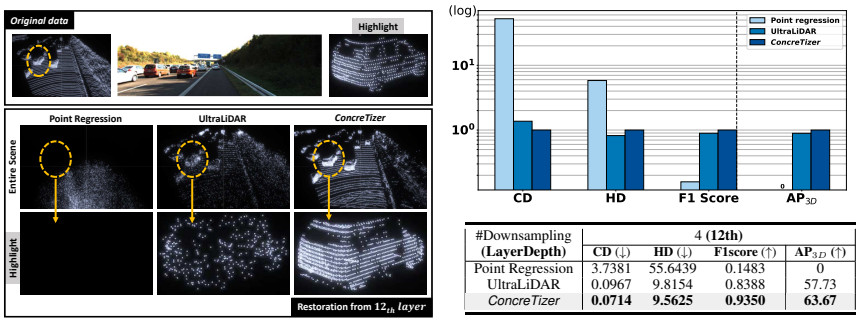


Figure 5: **Qualitative results for KITTI (scene 73) and Waymo (scene 79).** Top shows the original point cloud, 2D image, and highlighted region. Below, restoration performance of three techniques is displayed, progressing left to right by layer depth.



#Downsampling (LayerDepth)	4 (12th)			
	CD (↓)	HD (↓)	F1score (↑)	AP_{3D} (↑)
Point Regression	3.7381	55.6439	0.1483	0
UltraLiDAR	0.0967	9.8154	0.8388	57.73
ConcreTizer	**0.0714**	**9.5625**	**0.9350**	**63.67**

Figure 6: **Restoration result on VoxelResBackbone with KITTI dataset.** At the left, the last layer's restoration performance for three techniques is shown. At the right, average performance across the KITTI dataset is presented. A bar graph depicts relative performance, and a table details raw values.

Result Analysis. Table 1 presents the point scene restoration performance at different depths of VoxelBackBone (Yan et al., 2018), utilizing point cloud similarity metrics, while Figure 5 illustrates the corresponding restored point cloud scenes. It is evident that *ConcreTizer* consistently demonstrates exceptional performance across all cases within both the KITTI and Waymo datasets.

It can be observed that traditional Point Regression methods are not feasible for performing inversion attacks on 3D features. In particular, the results show that numerous points cluster near the origin, and this phenomenon intensifies as the depth of the layers increases. This can be understood as a failure to take into account the characteristics inherent to 3D sparse features.

By employing a conditional generation approach, UltraLiDAR can restore the overall scene in a coarse-grained manner with less significant performance degradation in terms of the HD metric relative to layer depth. This rough recovery can be attributed to the conversion of 3D sparse features into 2D dense features, aligning with the 2D VQ-VAE design. In the context of 2D dense features, problems like VoI dispersion are no longer present, which contributes to better robustness. However, this transition results in the loss of 3D sparse characteristics, leading to less accurate restoration of finer details.

In contrast, our *ConcreTizer* effectively suppresses VoI dispersion through DCS while preserving the sparse characteristics of 3D features. As a result, despite its simple design, *ConcreTizer* achieves a more concrete restoration than the generative model approach. At the deepest layer, *ConcreTizer* outperforms the generative approach by 23.4% and 12.4% on KITTI, while on Waymo, it shows improvements of 21.1% and 4.5% in CD and F1 score, respectively.

In Figure 6, we also tested *ConcreTizer* to the VoxelResBackbone (Lu et al., 2022). In this case, when analyzing the representative results from the deepest layer, *ConcreTizer* exhibits the highest of performance. A persistent issue with UltraLiDAR is the lack of detailed shape in the restored scenes. Detailed experimental results, including those conducted with the VoxelResBackbone and the Waymo dataset, are provided in the supplementary materials.

Table 2: **3D object detection results with KITTI and Waymo datasets.** The reported metric for the KITTI dataset is Average Precision (AP) at hard difficulty, while for the Waymo dataset, Average Precision weighted by Heading (APH) is reported at LEVEL2 difficulty.

	Detection Model	PointPillar	PVRCNN	VoxelRCNN	PointRCNN
KITTI	Original Data	76.11	78.82	78.78	78.25
	Point Regression	0	0	0	0
	UltraLiDAR	58.32	56.08	59.00	54.19
	ConcreTizer	**65.97**	**59.42**	**64.46**	**63.62**

	Detection Model	PointPillar	PVRCNN	VoxelRCNN	CenterPoint
Waymo	Original Data	0.5604	0.6534	0.6554	0.6239
	Point Regression	0	0	0	0
	UltraLiDAR	0.2328	0.1602	0.2179	0.1944
	ConcreTizer	**0.4245**	**0.4369**	**0.4100**	**0.4107**

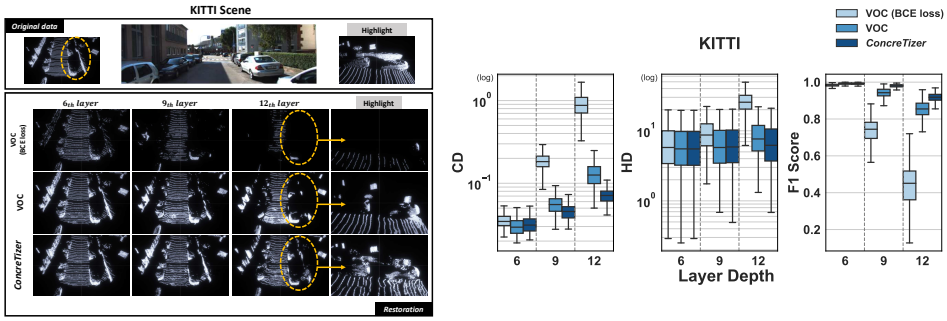


Figure 7: **Ablation study on VoxelBackbone with KITTI dataset.** At the left, the restoration performance for three cases is shown. At the right, average performance across the KITTI dataset is presented wih boxplot.

5.3 Attack Performance in the Context of 3D Object Detection

To assess the effectiveness of the restored scenes in compromising privacy, we measured the 3D object detection accuracy using restored point cloud scenes with separately trained object detection models. Table 2 displays the benchmark results for the KITTI and Waymo datasets. Point Regression failed to restore meaningful data, producing entirely unusable results. In addition, UltraLiDAR performed relatively well on KITTI but showed very poor performance on Waymo, which involves a broader range and higher scene complexity. This indicates that while generative models can restore overall shapes, they struggle to restore detailed features. Only *ConcreTizer* demonstrates a consistent performance across both datasets, achieving 75.4 to 86.7% and 62.6 to 75.7% of the detection performance based on original scenes in KITTI and Waymo, respectively.

5.4 Ablation Study: Component-wise Analysis

To understand the performance of *ConcreTizer*, we examine the impact of each component. Figure 7 illustrates the comparative performance of VOC (BCE loss), VOC, and *ConcreTizer* (VOC+DCS). Firstly, VOC (BCE loss) reveals that shifting from regression to classification, thus clarifying the meaning of zero-padded voxels, made restoration possible (6th layer result). However, the BCE loss struggles to handle the significant label imbalance as the layer depth increases. By comparing VOC (BCE loss) with VOC, it becomes evident that using SF loss helps mitigate the label imbalance issue to some extent. Nonetheless, in the results of VOC, the restored points tend to cluster in specific areas, resulting in biased restoration (12th layer result). Only *ConcreTizer* successfully restores points with a distribution closely matching the original point cloud. This effectiveness stems from DCS's ability to efficiently mitigate the dispersion of VoI that arises with deeper layer.

5.5 Partitioning Policy in Dispersion-Controlled Supervision

We conduct experiments to determine the effective strategy for applying DCS in *ConcreTizer*, given a specific 3D feature extractor. Restoration performance is evaluated on the KITTI dataset by varying the number of DCS instances (i.e., the number of inversion blocks). In each case, partitioning is applied at positions that aim to achieve an even division of the total number of layers. As shown in Figure 8, applying 10 DCS instances leads to significantly worse performance than not using DCS at all (i.e., DCS 1). This is because the restoration error of each block accumulates as it goes through multiple blocks. The best performance is achieved with 2, 3, or 4 DCS instances, where each partitioned block contains at least one downsampling layer. This is attributed to the additional supervision effectively suppressing VoI dispersion that occurs during the downsampling process. Therefore, to maximize the benefits of supervision, partitioning should align with the downsampling layers, where VoI dispersion manifests. Qualitative results for different DCS instances and further discussion on the optimal DCS split position are provided in the supplementary materials.

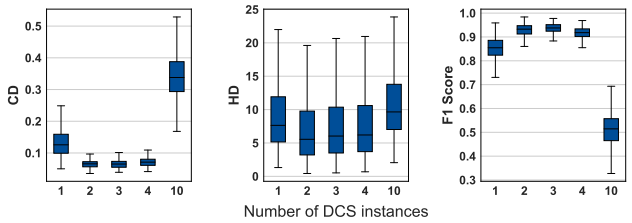


Figure 8: **Effect of the number of DCS instances.** DCS 1 is the end-to-end approach without partitioning. DCS 2, 3, and 4 use downsampling-based partitioning with 2 or 1 downsampling layers per block. DCS 10 partitions at every layer.

Table 3: **Effect of point cloud augmentation.** Measured: AP_{3D} (detection accuracy) of the SECOND model and CD (restoration error) of *ConcreTizer*.

Rotation (°)	0	1	2	3	4	5
AP	81.77	38.38	17.08	12.10	6.10	2.78
CD	0.0776	0.1142	0.1728	0.2310	0.2848	0.3344

Scaling (%)	0	2	4	6	8	10
AP	81.77	54.12	24.09	11.47	8.88	5.26
CD	0.0776	0.1468	0.2253	0.2780	0.3179	0.3516

Sampling (%)	100	25	20	15	10	5
AP	81.77	63.35	58.32	52.71	40.31	24.59
CD	0.0776	0.1368	0.1516	0.1717	0.2034	0.2789

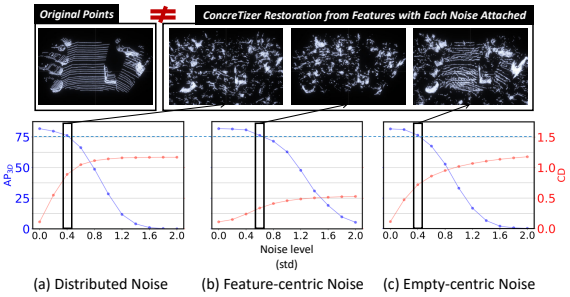


Figure 9: **Effect of Gaussian noise.** Measured: AP_{3D} (detection accuracy) of the SECOND model and CD (restoration error) of *ConcreTizer*.

5.6 Tradeoff Between Privacy and Utility

To understand the trade-off between utility (3D object detection accuracy) and privacy protection (restoration error), we explored various data perturbation techniques as potential defense mechanisms against our *ConcreTizer* inversion attack. We investigated two types of perturbations: **point cloud augmentations** and **Gaussian noise addition**. For point cloud augmentations, we applied *random rotations* within a range of $-$a to $+$a degrees (a=1, 2, 3, 4, 5), *random scaling* within $-$s% to $+$s% (s=2, 4, 6, 8, 10), and *random sampling* where r% of points were randomly selected (r=25, 20, 15, 10, 5). For Gaussian noise addition, we introduced noise at the feature data level with three region-specific configurations: *distributed noise*, which is uniformly applied across all feature data regions; *feature-centric noise*, applied only to VoI containing information; and *empty-centric noise*, exclusively targeting empty region.

The results, as shown in Table 3 and Figure 9, indicate that while these data perturbation techniques can reduce the restoration capability (defense), they also lead to a degradation in object detection performance (target task). This trade-off highlights the significant challenge of mitigating *ConcreTizer* attacks without severely compromising the system's utility. Notably, in Figure 9 we observed that the sparse nature of 3D feature data causes noise to impact different regions unevenly. This finding underscores the importance of considering such spatial characteristics when designing future defense mechanisms. More visualization results are provided in the supplementary materials. We also discuss more about potential defense mechanisms in the supplementary materials.

6 Conclusion

This paper presents the first comprehensive study on model inversion for 3D point cloud restoration. In the context of autonomous driving, we focus on the most dominant voxel-based feature extractors and examine the challenges arising from their interaction with 3D point cloud characteristics. Based on this, we introduced *ConcreTizer*, a simple yet effective inversion technique tailored for restoring 3D point data from features, which incorporates Voxel Occupancy Classification and Dispersion-Controlled Supervision. Through rigorous evaluations using prominent open-source datasets such as KITTI and Waymo, along with representative 3D feature extractors, we not only demonstrated the superiority of *ConcreTizer* but also analyzed each of its components in detail for valuable insights. Our research reveals the vulnerability of 3D point cloud data to inversion attacks, emphasizing the urgent need to devise extensive defense strategies. While this work focuses on voxel-based representations, we see inversions attacks for more diverse representations of 3D data, such as point set, graph, and projection, as valuable future work.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer vision, graphics, and image processing*, 27(3):321–345, 1984.

Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5034–5040. IEEE, 2019.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Yuwei Cheng and Yimin Liu. Person reidentification based on automotive radar point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2021.

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 2016a.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016b. URL `https://proceedings.neurips.cc/paper/2016/file/371bce7dc83817b7893bcdeed13799b5-Paper.pdf`.

Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. Towards explanation of dnn-based prediction with guided feature inversion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1358–1367, 2018.

Mihai Dusmanu, Johannes L Schonberger, Sudipta N Sinha, and Marc Pollefeys. Privacy-preserving image features via adversarial affine subspace embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14267–14277, 2021.

GDPR EU. General data protection regulation, 2016.

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.

Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In David E. Losada and Juan M. Fernández-Luna (eds.), *Advances in Information Retrieval*, pp. 345–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31865-1.

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

Longhua Guo, Mianxiong Dong, Kaoru Ota, Qiang Li, Tianpeng Ye, Jun Wu, and Jianhua Li. A secure mechanism for big data collection in large scale internet of vehicle. *IEEE Internet of Things Journal*, 4(2): 601–610, 2017.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Mengxi Hanyao, Yibo Jin, Zhuzhong Qian, Sheng Zhang, and Sanglu Lu. Edge-assisted online on-device object detection for real-time video analytics. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2021.

Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7662–7670, 2020.

D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993. doi: 10.1109/34.232073.

Sunwook Hwang, Youngseok Kim, Seongwon Kim, Saewoong Bahk, and Hyung-Sin Kim. Upcycling: Semi-supervised 3d object detection without sharing raw-level unlabeled scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23351–23361, October 2023.

Pileun Kim, Jingdao Chen, and Yong K Cho. Slam-driven robotic mapping and registration of 3d point clouds. *Automation in Construction*, 89:38–48, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12697–12705, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015.

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.

Yuhui Lu, Zhongxi Chen, and Mingbo Zhao. 3d objective detection for autonomous driving based on two-stage approach. In *2022 IEEE International Symposium on Product Compliance Engineering-Asia (ISPCE-ASIA)*, pp. 1–5. IEEE, 2022.

Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2837–2845, 2021.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Priyanka Mandikal and Venkatesh Babu Radhakrishnan. Dense 3d point cloud reconstruction using a deep pyramid network. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1052–1060. IEEE, 2019.

Priyanka Mandikal, KL Navaneet, Mayank Agarwal, and R Venkatesh Babu. 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image. *arXiv preprint arXiv:1807.07796*, 2018.

Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12923–12932, 2023.

Trix Mulder and Nynke E Vellinga. Exploring data protection challenges of automated driving. *Computer Law & Security Review*, 40:105530, 2021.

Tony Ng, Hyo Jin Kim, Vincent T Lee, Daniel DeTone, Tsun-Yi Yang, Tianwei Shen, Eddy Ilg, Vassileios Balntas, Krystian Mikolajczyk, and Chris Sweeney. Ninjadesc: content-concealing visual descriptors via adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12797–12807, 2022.

Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 145–154, 2019.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–779, 2019.

Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10529–10538, 2020.

Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1711–1719, 2020.

Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3859–3868, 2019.

Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019a.

Akash Deep Singh, Sandeep Singh Sandha, Luis Garcia, and Mani Srivastava. Radhar: Human activity recognition from point clouds generated through a millimeter-wave radar. In *Proceedings of the 3rd ACM Workshop on Millimeter-wave Networks and Sensing Systems*, pp. 51–56, 2019b.

Zhenbo Song, Wayne Chen, Dylan Campbell, and Hongdong Li. Deep novel view synthesis from colored 3d point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pp. 1–17. Springer, 2020.

Bernd Carsten Stahl and David Wright. Ethics and privacy in ai and big data: Implementing responsible research and innovation. *IEEE Security & Privacy*, 16(3):26–33, 2018.

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.

Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *European Conference on Computer Vision*, pp. 426–442. Springer, 2022.

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

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6411–6420, 2019.

Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. In *International conference on learning representations*, 2018.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 605–621. Springer, 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*, 2024.

Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1939–1948, 2020.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 9621–9630, 2019.

Zhu Xiao, Jinmei Shu, Hongbo Jiang, Geyong Min, Hongyang Chen, and Zhu Han. Perception task offloading with collaborative computation for autonomous driving. *IEEE Journal on Selected Areas in Communications*, 41(2):457–473, 2022.

Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1074–1083, 2023.

Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *European conference on computer vision*, pp. 107–124. Springer, 2022.

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.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4541–4550, 2019a.

Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 225–240, 2019b.

Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11784–11793, 2021.

Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, et al. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21361–21370, 2022.

Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12498–12507, 2021.

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

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 253–261, 2020.

Ziyu Zhang, Feipeng Da, and Yi Yu. Data-free point cloud network for 3d face recognition. *arXiv preprint arXiv:1911.04731*, 2019.

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)*, pp. 682–692, October 2021.

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.

Yufan Zhou, Haiwei Dong, and Abdulmotaleb El Saddik. Learning to estimate 3d human pose from point cloud. *IEEE Sensors Journal*, 20(20):12334–12342, 2020.

Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision*, pp. 17–35. Springer, 2022.

Appendix

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

A VOXELIZATION EFFECT

To address the challenging issue of restoring 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 for restoring a 3D point scene. We analyze 3D scenes in the KITTI dataset using representative voxelization-based extractors (Zhou & Tuzel, 2018; Yan et al., 2018) used in AV applications to prove the validity of the VSP hypothesis.

As shown in Figure 10 (left), the proportion of voxels with no points at all or containing only a single point consists of 99.988% of the total voxels. Figure 10 (right) indicates that voxels with multiple points, which are extremely rare, are mostly 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 11 visualizes regions of the KITTI dataset's where multi-point voxels exist, by using the original point cloud and its voxelized result (one point per voxel) using a 3D feature extractor. This comparison highlights that even close area from the LiDAR sensor shows negligible differences between the original and voxelized point clouds. This demonstrates that a single point per voxel sufficiently preserves the scene's information integrity.

B DISPERSION OF VOI

The Voxels-of-Interest (VoI) experiences dispersion during feature extraction and restoration, as described in Section 3. In Figure 2 of main paper, 'Data stattistics' shows the grid density statistics at each layer throughout the feature extraction and restoration process. It is evident that the density increases as the data passes through the downsampling (3rd, 6th, 9th, and 12th) and upsampling (13th, 16th, 19th, and 22th) layers. This phenomenon is attributed to the inherent nature of convolution and transposed convolution layer, which inherently spread values to the surrounding regions. Conversely, the density does not change 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 (**?**).)

- 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 aforementioned metric has its own advantages and disadvantages in fully assessing the accuracy from a restoration points perspective. Therefore, in the main paper, we have introduced various metrics and used visual aids to facilitate a more insightful understanding of the restoration performance.

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 each dataset's 3D object detection benchmark. For KITTI, under 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 (1408, 1600, 40) grid. For Waymo, under 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 (1504, 1504, 40) grid. 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 the evaluation process, aside from visualization, the range was extended back to the object detection range. (For visualization, captured images from the close range are used.)

The training utilizes 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 gamma value is set to 2. Tables 4, 5 present the alpha values used in the experiments. For *ConcreTizer*, the number of blocks increases alongside the number of downsampling layers; consequently, the alpha value for each block is denoted 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 6, 7, 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 8 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 9, exhibits a one-dimensional channel because the occupancy of the voxel unit is classified in the final layer. Regarding *ConcreTizer* in Table 10, 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 sufficiently conveys our message through its results, we aim to share more granular experimental outcomes and settings details. This additional information provides deeper insights and a fuller 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 12 and 13 illustrates the comparative performance of Point Regression, VOC, and *ConcreTizer* (VOC+DCS) across different depths of the feature extractor's layers. These results indicate that using only VOC initially 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 11 shows a quantitative evaluation result for VoxelResBackBone. Figures 14, 15, 16, and 17 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 shows the restoration results from the final (12th) layer. *ConcreTizer* demonstrates better performance in restoring the overall shape when compared to VOC's restoration, which is more excessively clustered.

F.2 FURTHER DETAILS OF DCS INSTANCES

Section 5.4 covers an ablation study about the number of DCS instances. Figure 18 shows the results restored for different 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. On the other hand, the downsampling-based partitioning of *ConcreTizer* performs better by preventing VoI dispersion, which is greater than the cumulative effect of error.

F.3 DCS OPTIMAL SPLIT POSITION

When employing DCS, a trade-off between two effects occurs at the split point: while DCS can mitigate dispersion effects with additional supervision, it also risks accumulating reconstruction errors into the next block. Section 5.5 analyzes performance about the number of DCS blocks, revealing high performance with either 2 or 4 blocks. Here, we investigate performance with different split positions for 2 and 4 blocks configurations.

The Figure 19 shows the performance when there are 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), splitting effect is minimal, while in highly dispersed blocks (f'9 ~f'2), restoration without splitting is challenging. The best performance was observed at 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 20 with four DCS blocks, options 0 and 1 displayed inferior performance due to uneven dispersion splitting. Conversely, 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. Consequently, if we can model the randomization effects for each layer, we can investigate optimal split positions in future research.

F.4 FURTHER DETAILS OF NOISE EFFECT

Section 5.5 demonstrates the impact of various noise types on feature restoration from the SECOND object detection model (Yan et al., 2018) and assesses object detection performance with these noise-added features. Additionally, Figure 21 illustrates the restoration results as noise levels vary, highlighting how the sparse nature of the 3D features results in different impacts on restoration performance depending on the noise's location within the feature.

F.5 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 (**??**). This approach provides robust protection against worst-case scenarios but excessively sacrifices utility. Recent advancements have integrated DP with generative models (**?**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 (**??**) 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.

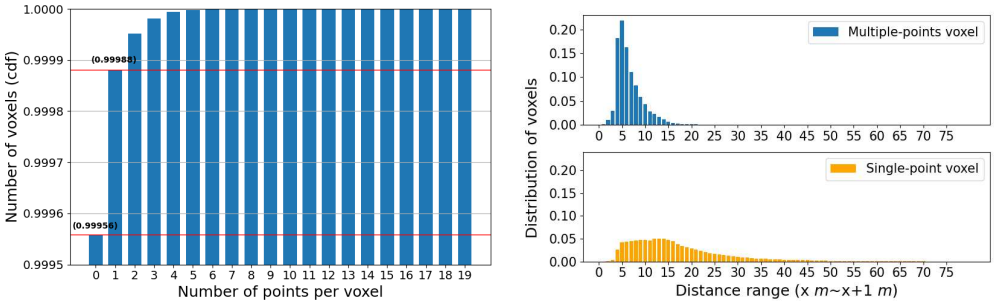


Figure 10: (Left) The voxel distribution based on the number of points inside each voxel using a cumulative distribution function. (Right) The distribution of voxels within the range of x to x+1 meters that are not single-point voxels.

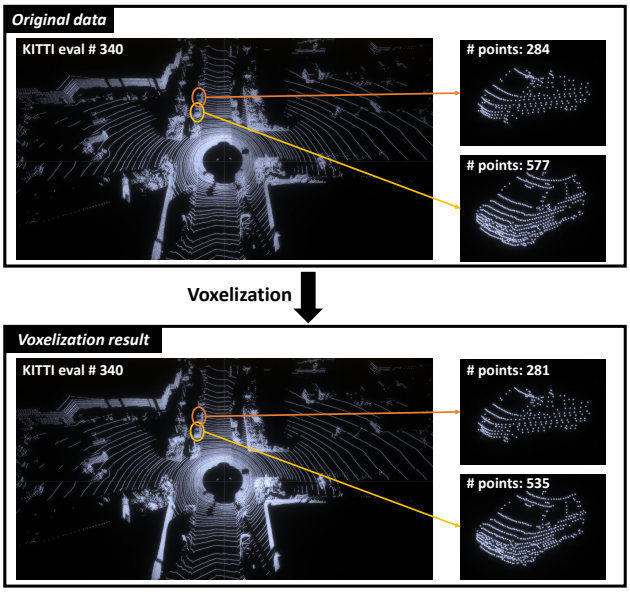


Figure 11: **Effect of voxelization process on point cloud.** The voxel size is 5 cm x 5 cm x 10 cm, and the maximum number of points per voxel is set as 5. Then points in each voxel are averaged to get a single representative value for each channel.

Table 4: **Alpha values of SF loss for VoxelBackBone with KITTI and Waymo dataset.** *ConcreTizer* employs multiple blocks, each with a distinct alpha 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 5: **Alpha values of SF loss for VoxelResBackBone with KITTI and Waymo dataset.** *ConcreTizer* employs multiple blocks, each with a distinct alpha 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 6: **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 7: **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 8: **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 9: **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 10: **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 11: **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 scene sets 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**

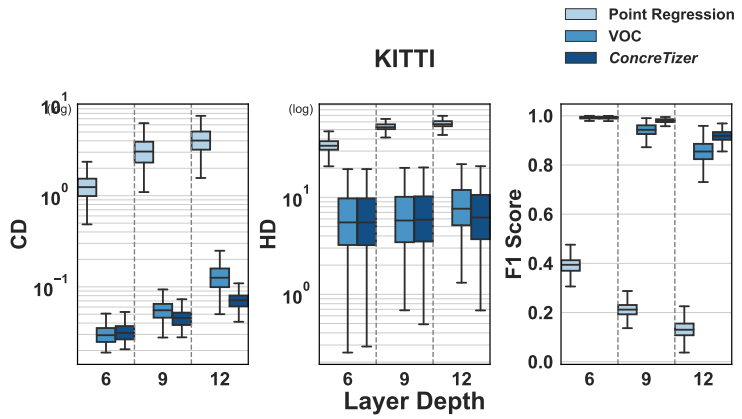


Figure 12: **Component-wise comparison with KITTI dataset.**

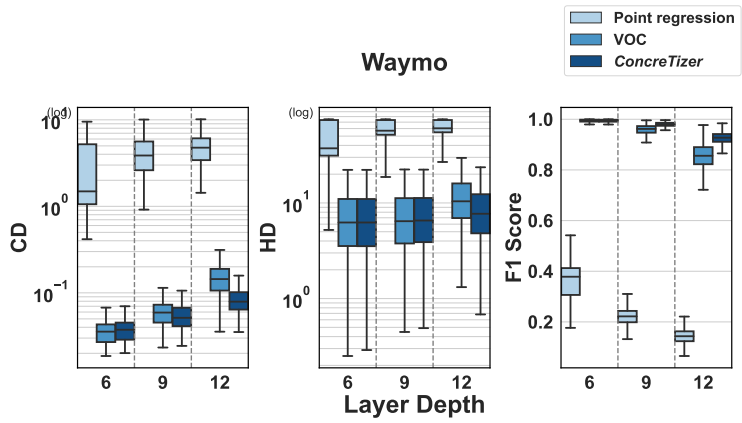


Figure 13: **Component-wise comparison with Waymo dataset.**

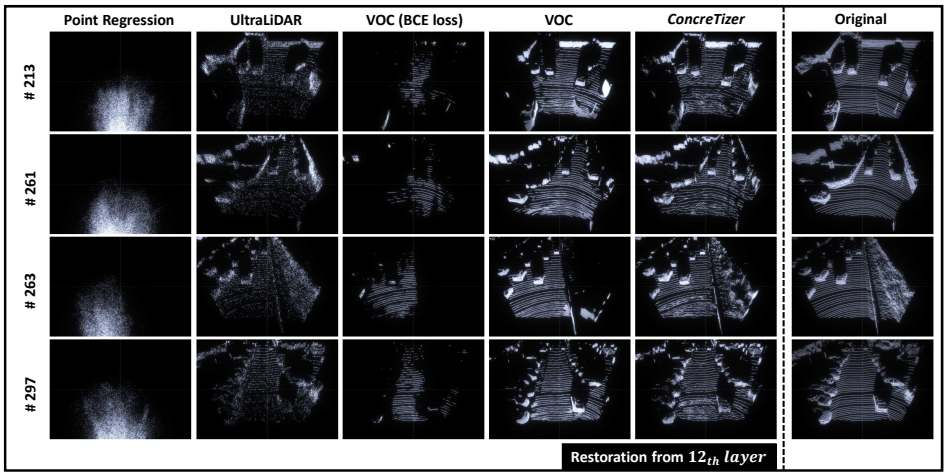


Figure 14: **Additional qualitative results for VoxelBackBone with KITTI dataset.** Each row presents the restoration result and corresponding original data for a specific KITTI validation scene. The input is the 12th (the final) layer.

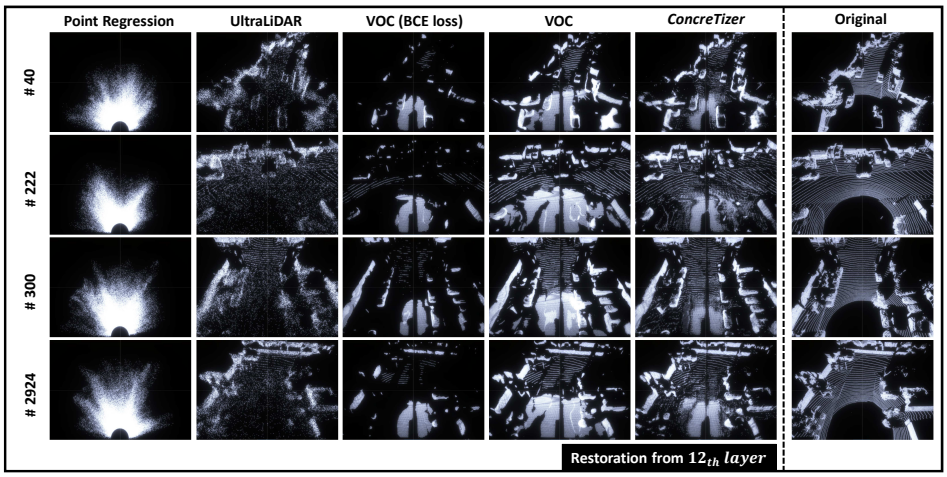


Figure 15: **Additional qualitative results for VoxelBackBone with Waymo dataset.** Each row presents the restoration result and corresponding original data for a specific Waymo validation scene. The input is the 12th (the final) layer.

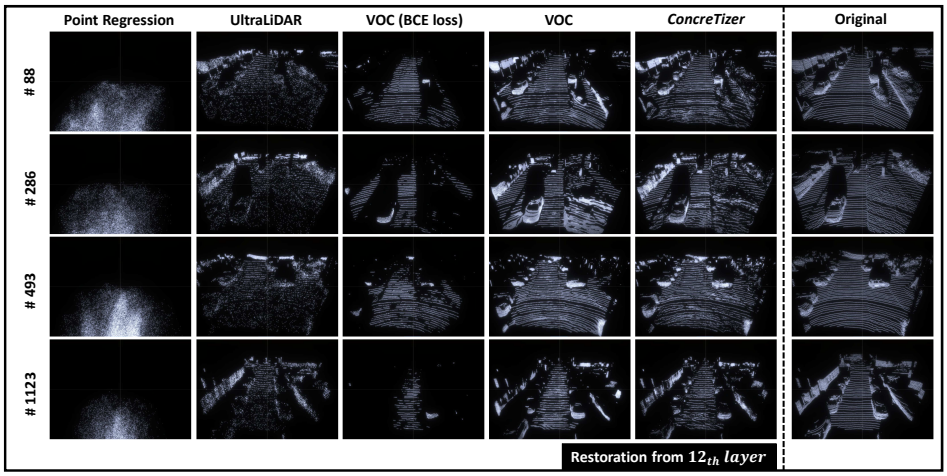


Figure 16: **Additional qualitative results for VoxelResBackBone with KITTI dataset.** Each row presents the restoration result and corresponding original data for a specific KITTI validation scene. The input is the 12th (the final) layer.

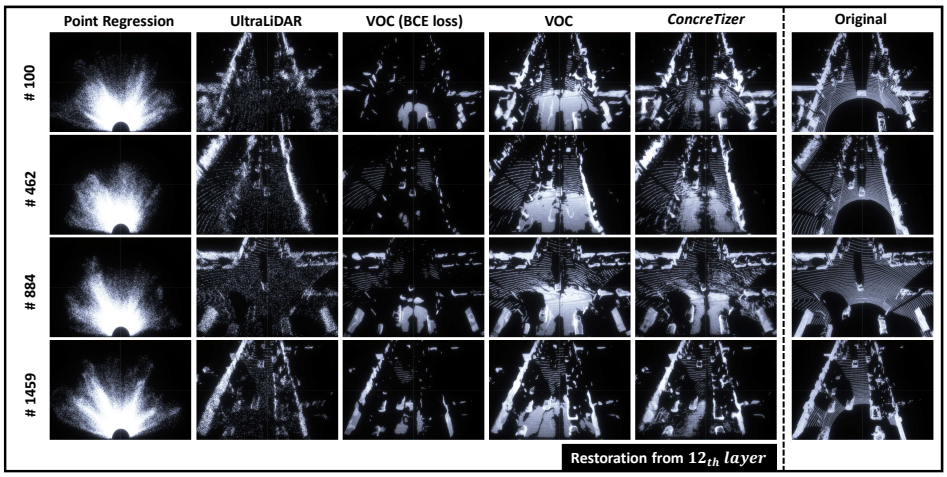


Figure 17: **Additional qualitative results for VoxelResBackBone with Waymo dataset.** Each row presents the restoration result and corresponding original data for a specific Waymo validation scene. The input is the 12th (the final) layer.

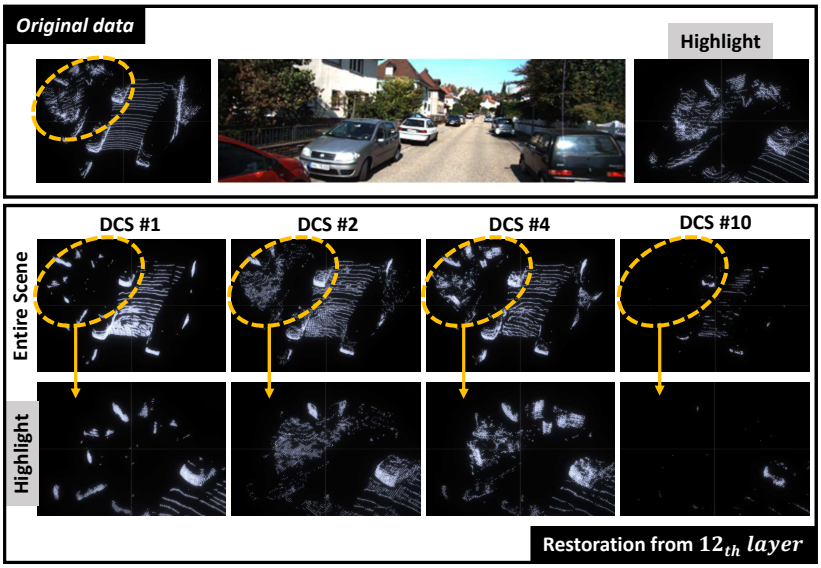


Figure 18: **Qualitative result for different DCS instances with** *ConcreTizer* **model.**

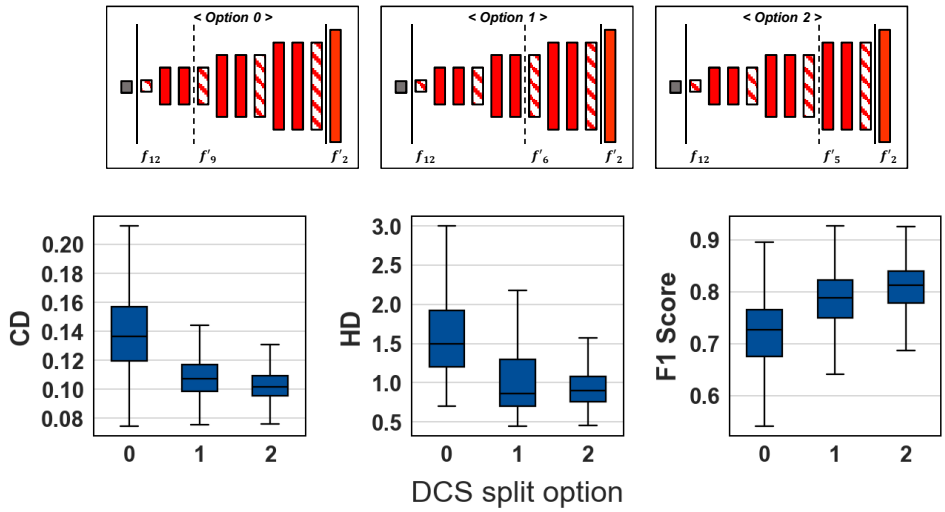


Figure 19: **DCS #2 split option 0 to 2.**

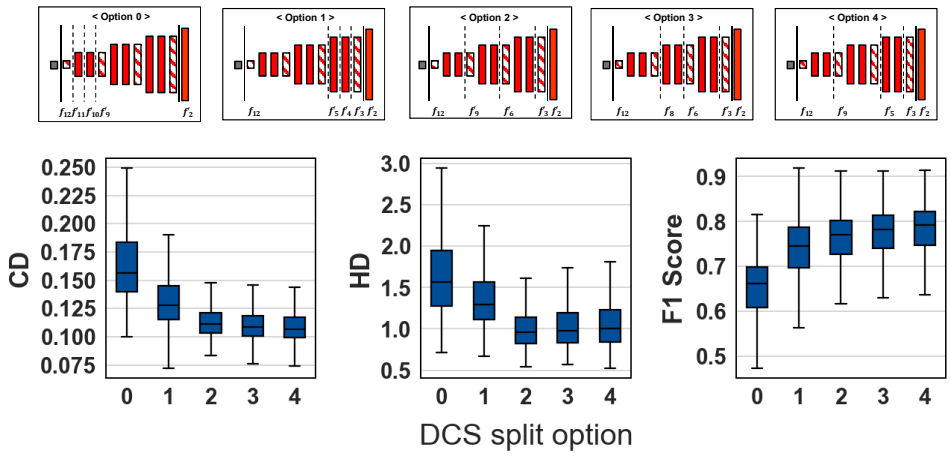


Figure 20: **DCS #4 split option 0 to 4.**

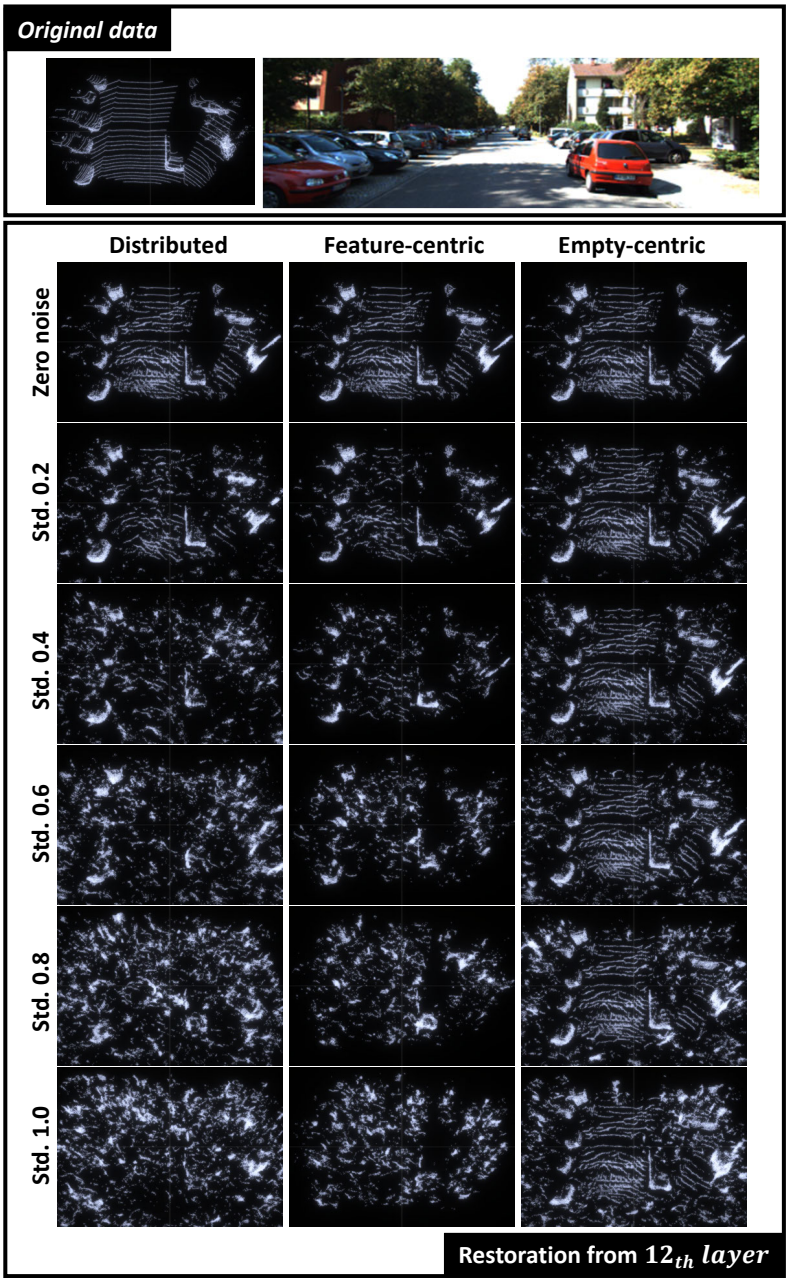


Figure 21: **Qualitative results for different noise levels.**