

Common Corruption Robustness of Point Cloud Detectors: Benchmark and Enhancement

Shuangzhi Li, Zhijie Wang, Felix Juefei-Xu, Qing Guo*, Xingyu Li, and Lei Ma

Abstract—Object detection through LiDAR-based point cloud has recently been important in autonomous driving. Although achieving high accuracy on public benchmarks, the state-of-the-art detectors may still go wrong and cause a heavy loss due to the widespread corruptions in the real world like rain, snow, sensor noise, *etc.* Nevertheless, there is a lack of a large-scale dataset covering diverse scenes and realistic corruption types with different severities to develop practical and robust point cloud detectors, which is challenging due to the heavy collection costs. To alleviate the challenge and start the first step for robust point cloud detection, we propose the physical-aware simulation methods to generate degraded point clouds under different real-world common corruptions. Then, for the first attempt, we construct a benchmark based on the physical-aware common corruptions for point cloud detectors, which contains a total of 1,122,150 examples covering 7,481 scenes, 25 common corruption types, and 6 severities. With such a novel benchmark, we conduct extensive empirical studies on 8 state-of-the-art detectors that contain 6 different detection frameworks. Thus we get several insight observations revealing the vulnerabilities of the detectors and indicating the enhancement directions. Moreover, we further study the effectiveness of existing robustness enhancement methods based on data augmentation and data denoising. The benchmark can potentially be a new platform for evaluating point cloud detectors, opening a door for developing novel robustness enhancement methods.

Index Terms—Point cloud, Object Detection, Benchmark, Robustness

I. INTRODUCTION

Object detection via LiDAR-based point cloud [1], [2], as a crucial task in 3D computer vision, has been widely used in applications like autonomous driving [3]. Recently, the data-driven methods (*i.e.*, deep neural networks) have significantly improved the performance of 3D point cloud detectors [4], [5], [2] on various public benchmarks, *e.g.*, KITTI [6], NuScenes [7], and Waymo [8]. However, the scenarios covered by these public benchmarks are usually limited. For instance, there is a lack of natural fog effects in these datasets, while fog could affect the reflection of laser beams and corrupt point cloud data with false reflections by droplets [9], [10]. Apart from the external scenarios, the internal noise of sensors can also increase the deviation and variance of ranging

measurements [11] and result in corrupted data and detector performance degradation. Given that LiDAR-based point cloud detection is usually used in safety-critical applications (*e.g.*, autonomous driving) and these external and internal corruptions could potentially affect detectors’ robustness [12], [13], [11], it is critical to comprehensively evaluate an object detector under those corruptions before deploying it in real-world environments.

There are some works constructing datasets while considering extreme weather like CADC [14], Boreas [15], SeeThrough-Fog (STF) [10]. Nevertheless, the constructed datasets only consider limited situations in the real world due to the heavy collection costs, which are far from a comprehensive evaluation. For instance, Boreas only covers 4 rainy scenes and 5 snowy scenes. STF only contains foggy point clouds at severity levels of “dense” and “light”. Hence, there is an increasing demand for extending existing benchmarks to conduct a comprehensive evaluation through covering diverse corruptions in the real world. A straightforward way is to synthesize the corrupted point clouds given the success of similar solutions in the image-based tasks [16], [17] and 3D object recognition [11], [18]. However, there is no accessible dataset for the robustness evaluation of point cloud detectors. Note that, the robustness datasets (*e.g.*, Modelnet40-C [11]) for 3D object recognition cannot be used to evaluate the point cloud detectors, directly: (1) the example in the recognition dataset only contains the points of an object and cannot be adopted for object detection task that aims to localize and classify objects in 3D scene. (2) The latest Modelnet-C [18] and Modelnet40-C [11] only consider 7 corruptions and 15 corruptions, respectively, which is still limited for a comprehensive evaluation in safety-critical environments such as autonomous driving.

The main challenge for building a dataset for the robustness evaluation of point cloud detection stems from the huge amount of diverse corruption types with different physical imaging principles. For example, flawed sensors and different object characteristics could lead to noise-like corruptions and affect spherical and Cartesian coordinates of points, respectively. Different weathers like rain and fog might lead to false reflections. These corruptions have different imaging principles and need careful designs of the respective simulation methods.

In this work, for the first attempt, we construct a benchmark to evaluate the robustness of point cloud object detectors based on LiDAR under diverse common corruptions and discuss the effectiveness of existing robustness enhancement methods. Regarding the benchmark construction, we first design physical-aware simulation methods for 25 corruptions according to their physical models, respectively. Then, we borrow 7,481

*Qing Guo is the corresponding author.

Shuangzhi Li, Zhijie Wang, Xingyu Li, and Lei Ma are with the University of Alberta, AB, Canada. Zhijie Wang and Lei Ma are also with the Alberta Machine Intelligence Institute, AB, Canada. Lei Ma is also with Kyushu University, Japan. (e-mail: {shuangzh, zhijie.wang, xingyu}@ualberta.ca, ma.lei@acm.org)

Qing Guo is with the Nanyang Technological University, Singapore. (e-mail: tsingguo@ieec.org)

Felix Juefei-Xu is with New York University, New York, NY 10012, USA. (e-mail: juefei.xu@nyu.edu)

TABLE I: Summary of datasets used for LiDAR-based point cloud object detection

Dataset	Year	Real/Simulated	Frames	BBoxes	Classes	Corruptions	Corruption Severities	Robustness Metric
KITTI [6]	2012	real	15K	200K	8	cutout, noise	2	-
NuScenes [7]	2019	real	400K	1.4M	23	rain, sun, clouds, cutout, various vehicle types, noise	2	-
Waymo [8]	2019	real	200K	12M	4	rain, fog, cutout, dust, various vehicle types, noise	2	-
Boreas [15]	2022	real	7.1K	320K	4	snow, rain, sun, clouds, cutout, noise	2	-
STF [10]	2020	real	13.5K	100K	4	fog, rain, snow, cutout, noise	3	-
CADC [14]	2020	real	7K	334K	10	snow, bright light, cutout, noise	5	-
ModelNet40-C [11]	2022	real+simulated	185K	-	40	occlusion, LiDAR, local_density_inc/dec, cutout, uniform, Gaussian, impulse, upsampling, background, rotation, shear, FFD, RBF, inv_RBF	6	✓
ModelNet-C [18]	2022	real+simulated	185K	-	40	scale, rotate, jitter, drop_global/local, add_global/local	6	✓
Argoverse [19]	2019	real	468K	993K	15	rain, cutout, dust, noise	2	-
Lyft Level 5 [20]	2020	real	30K	1.3M	9	rain, cutout, noise	2	-
Ours	2022	real+simulated	1.1M	15M	8	Scene: rain, snow, fog, uniform_rad, gaussian_rad, impulse_rad, upsample, background, cutout, beam_del, local_dec/inc, layer_del; Object: uniform, gaussian, impulse, upsample, cutout, local_dec/inc, shear, scale, rotation, FFD, translation	6	✓

raw 3D scenes (*i.e.*, clean point clouds) from [6] and build large-scale corrupted datasets by adding 25 corruptions with 6 different severity levels to each clean point cloud. Finally, we obtain a total of 1,122,150 examples covering 7,481 scenes, 25 common corruption types, and 6 severity levels. Compared with real-world data benchmark (see Table I), the proposed benchmark synthesized more examples for benchmarking robustness. Compared with other synthesized benchmark (see Table I), our benchmark provides more types of corruption patterns to specifically support benchmarking object detection. Note that, we conduct extensive experiments to quantitatively validate the effectiveness of simulation methods by evaluating the naturalness of synthesized data.

With such a novel benchmark, we investigate the robustness of current point cloud detectors by conducting extensive empirical studies on 8 existing detectors, covering 3 different representations and 2 different proposal architectures. In particular, we study the following four research questions to identify the challenges and potential opportunities for building robust point cloud detectors:

- **How do the common corruption patterns affect the point cloud detector’s performance?** Given overall common corruptions, an accuracy drop of 11.01% (on average) on all detectors anticipates a noticeable accuracy drop of detectors against diverse corruption patterns.
- **How does the design of a point cloud detector affect its robustness against corruption patterns?** Compared with two-stage detectors, one-stage detectors perform more robust against a majority of corruptions. Compared with point-based detectors, voxel-involving detectors perform more robust against the most of corruptions.
- **What kind of detection bugs exist in point cloud detectors against common corruption patterns?** Followed by the decrease in the rate of true detection, common corruptions widely trigger a number of false detections on all point cloud detectors.
- **How do the robustness enhancement techniques im-**

prove point cloud detectors against common corruption patterns? Even with the help of data augmentation and denoising, common corruptions still cause a severe accuracy drop of over 10% on detection.

In summary, this work makes the following contributions:

- We design physical-aware simulation methods covering 25 common corruptions related to natural weather, noise disturbance, density change, and object transformations at the object and scene level.
- We create the first robustness benchmark of point cloud detection against common corruptions.
- Based on the benchmark, we conduct extensive empirical studies to evaluate the robustness of 8 existing detectors to reveal the vulnerabilities of the detectors under common corruptions.
- We study the existing data augmentation (DA) method and denoising method’s performance on robustness enhancement for point cloud detection and further discuss their limitations.

II. RELATED WORK

A. LiDAR Perception

LiDAR perception is sensitive to both internal and external factors that could result in different corruptions. Adversarial weather [9] (*e.g.*, snow, rain, and fog) can dim or even block transmissions of lasers by dense liquid or solid droplets. Regarding noise characteristics of point clouds, strong illumination [22] affects the signal transmission by lowering Signal-to-Noise Ratio (SNR), increasing the noise level of LiDAR ranging [23]. Besides, the intrinsically inaccurately ranging and the sensor vibration [24], [25] potentially trigger noisy observations during LiDAR scanning. Environmental floating particles (*e.g.*, dust [26]) could perturb point cloud with the background noise. Density distribution of LiDAR-based point clouds can also easily affect autonomous driving. For instance, common object-object occlusions block LiDAR scanning on objects in the scene [13]. Besides, the dark-color cover and rough surface

TABLE II: Taxonomy of collected common corruption patterns

Scene-level			Object-level		
Corruption Category	Corruption	Potential Reasons	Corruption Category	Corruption	Potential Reasons
Weather	<i>rain</i>	Environment: natural weather [9];	Noise	<i>uniform</i>	Object surface: coarse surface [21] and dark-color cover [21];
	<i>snow</i>			<i>gaussian</i>	
	<i>fog</i>			<i>impulse</i>	
Noise	<i>uniform_rad</i>	Environment: strong illumination [22]; Sensor: low ranging accuracy [23] and sensor vibration [24], [25];	Density	<i>upsample</i>	Object surface: object or self-occlusions [13], dark-color cover [21] and transparent components;
	<i>gaussian_rad</i>			<i>cutout</i>	
	<i>impulse_rad</i>			<i>local_dec</i>	
	<i>upsample</i>			<i>local_inc</i>	
Density	<i>background</i>	Environment: floating particles [26]; Sensor: different scanning layers, object occlusion [13], and randomly laser beam [13] or layer (rotary laser) malfunction;	Transformation	<i>translation</i>	Object: different locations and heading directions [27]; Object deformation: bending or moving pedestrians [28], different styles of vehicles [29].
	<i>cutout</i>			<i>rotation</i>	
	<i>local_dec</i>			<i>shear</i>	
	<i>local_inc</i>			<i>FFD</i>	
	<i>beam_del</i>			<i>scale</i>	
	<i>layer_del</i>				

[21] could affect LiDAR’s reflection and thus reduce local point density when sensing such objects. Moreover, the malfunction of (fixed or rotary) lasers [13] globally loses points or layers of points in point clouds. For 3D tasks, various shapes [28], [29], locations and poses [27] of objects can also influence the context perception in the scene.

Apart from these natural corruptions, LiDAR perception is also sensitive to adversarial attack. Adversarial attacks [30] pose significant security issues and vulnerability on 3D point cloud tasks (*e.g.*, classification [31], detection [32], and segmentation [33]).

B. Point Cloud Detectors

Based on the different representations acquired from point clouds, point cloud detectors can be categorized into **2D-view-based** detectors (*e.g.*, VeloFCN [34] and PIXOR [35]), **voxel-based detectors** (*e.g.*, SECOND [36] and VoTr [37]), **point-based detectors** (*e.g.*, PointRCNN [38] and 3D-SSD [39]), and **point-voxel-based detectors** (*e.g.*, PVRCNN [40] and SA-SSD [41]). On the other hand, based on the different proposal architectures, point cloud detectors can also be divided into **one-stage detectors** (*e.g.*, 3D-SSD [39] and SA-SSD [41]) and **two-stage detectors** (*e.g.*, PointRCNN [38] and PVRCNN [40]). In this paper, we select 8 representative methods covering all these categories.

C. Robustness Benchmarks against Common Corruptions

Several attempts have been made to benchmark robustness for different data domains. Based on ImageNet [42], ImageNet-C simulates real-world corruptions to test image classifiers’ robustness. ObjectNet [17] illustrates the performance degradation of 2D recognition models considering object backgrounds, rotations, and imaging viewpoints. Inspired by 2D works, several benchmarks were built for 3D tasks. Modelnet40-C

[11] corrupts ModelNet40 [43] with 15 simulated common corruptions affecting point clouds’ noise, density, and transformations, to evaluate the robustness of point cloud recognition. Targeting 7 fundamental corruptions (*i.e.*, “Jitter”, “Drop Global/Local”, “Add Global/Local”, “Scale”, and “Rotate”), ModelNet-C reveals the vulnerability of different components of 9 existing point cloud classifiers. Regarding point cloud detection, NuScenes, Waymo, and STF collect LiDAR scans under adversarial rainy, snowy, and foggy conditions, where the accuracy of 3D detectors is tested [7], [10], [8]. However, to the best of our knowledge, a lack of benchmark of point cloud detection’s robustness comprehensively against various common corruptions is still remaining.

D. Robustness Enhancement for Point Cloud Detection

Recently, improving the robustness of point cloud detection has also received significant concerns. Zhang *et al.* propose PointCutMix [44] as a single way to generate new training data by replacing the points in one sample with their optimal assigned pairs in another sample. Lee *et al.* [45] propose a rigid subset mix (RSMix) augmentation to get a virtual mixed sample by replacing part of the sample with shape-preserved subsets from another sample. Specifically for 3D object detection, there are several ways to improve detectors’ robustness. Choi *et al.* [46] propose a part-aware data augmentation that stochastically augments the partitions of objects by 5 basic augmentation methods. LiDAR-Aug [47] presents a rendering-based LiDAR augmentation framework to improve the robustness of 3D object detectors. LiDAR light scattering augmentation [12] and LiDAR fog stimulation [48] utilize physics-based simulators to generate data corrupted by fog/snow/rain and then augment object detectors. Self-supervised pre-training [49], [50] can also endow the model with resistance to augmentation-related transformations. Besides, denoising methods [51], [52], [53] can remove the outliers in point clouds and thus potentially

improve detectors’ robustness. Regarding module design, there are also some detectors specialized for resisting corruptions, *e.g.* BtcDet [13] with the occupancy estimator for estimating occluded regions and Centerpoint [54] with key-point detector for a flexible orientation regression. In this paper, we evaluate part-aware data augmentation and K-nearest-neighbors-based filtering methods for improving point cloud detectors against diverse common corruption patterns.

III. BACKGROUND

A. Point Cloud Detection

Point clouds detectors aim to detect objects of interest in point clouds in the format of *bounding boxes* (BBoxes). Suppose a frame of point cloud data \mathbf{P} is a set of point $\mathbf{p} = [x^p, y^p, z^p, r^p]$, where (x^p, y^p, z^p) denotes its 3D location and r^p denotes reflective intensity. Thus we can formulate the point cloud detection as:

$$\begin{aligned} \text{Det}(\mathbf{P}) &= \{\mathbf{b}_i\}^N \\ \mathbf{b}_i &= [x_i, y_i, z_i, w_i, h_i, l_i, \theta_i, c_i, s_i] \end{aligned} \quad (1)$$

where $\text{Det}(\cdot)$ represents the detector; N is the number of detected BBoxes in \mathbf{P} ; \mathbf{b}_i denotes i_{th} detected BBox in \mathbf{P} , where $i = 1, 2, \dots, N$; (x_i, y_i, z_i) is the Cartesian coordinate of the center of \mathbf{b}_i , (w_i, h_i, l_i) is its dimensions, θ_i is its heading angle, c_i is its classification label, and s_i is its prediction confidence score.

Point cloud feature representation. Representation for features used in point cloud detection includes 2D-view images, voxels, and raw points. By projecting point clouds into a 2D bird’s eye view or front view, 2D-view-based 3D detectors can intuitively fit into a 2D image detection pipeline [34], [35]. However, 2D-view images could lose depth information [2], where the localization accuracy of the detector is affected. To efficiently acquire 3D spatial knowledge in large-scale point clouds, the “voxelization” operation is leveraged to partition unordered points into spatially and evenly distributed voxels [36], [55]. After pooling interior features, those voxels are fed into a sparse 3D convolution backbone [36] for feature abstraction. Given an appropriate voxelization scale, voxel-based representation is computationally efficient, but the quantization loss by voxelization is also inevitable [2]. Different from the above methods, PointNet [56] and PointNet++ [57] directly extract abstract features from raw points, which keeps the integrity of spatial context in point clouds. However, the point-based detectors are not cost-efficient for large-scale data [2]. As a trade-off between the voxel-based and point-based methods, Point-voxel-based representations [40], [41] possess the potential of fusing the high-efficient voxels and accurate-abstract points in feature abstraction.

Proposal architecture. One-stage detectors [36], [50] directly generate candidate BBoxes from the abstracted features. To improve candidate BBoxes’ precision, two-stage detectors [40], [13] refine those BBoxes by region proposal network (RPN) and tailor them into unified size by region of interest (RoI) pooling before predicting output BBoxes. Compared with one-stage detectors, two-stage ones [2] usually present more accurate

localization but intuitively, are more computationally time-consuming.

B. Robustness Enhancement Solutions

Several attempts have been made to enhance the robustness of point cloud detectors. In this paper, we select data augmentation and denoising methods to study their effects on improving point cloud detectors’ robustness against common corruptions. Data augmentation [58] is an effective way of increasing the amount of relevant data by slightly modifying existing data or newly creating synthetic data from existing data. Data augmentation on the point cloud [46], [59] provides detectors with a way to be trained with a larger dataset and thus potentially obtain more robust detectors. Different from data augmentation, denoising [51], [52] serves as a pre-process to detect and remove spatial outliers in point clouds, which can reduce the effects of noisy point cloud data.

IV. PHYSICAL-AWARE ROBUSTNESS BENCHMARK FOR POINT CLOUD DETECTION

We propose the first robustness benchmark of point cloud detectors against common corruption patterns. We first introduce different corruption patterns collected for this benchmark and dataset in Section IV-A. Then we propose the evaluation metrics used in our benchmark in Section IV-B. Finally, we introduce the subject-object detection methods and robustness enhancement methods selected for this benchmark in Section IV-C.

A. Physical-aware Corrupted Dataset Construction

After the literature investigation in Section II-A, we summarize 25 corruption patterns in Table II and categorize them into 4 categories based on presentations of common corruptions: *weather*, *noise*, *density*, and *transformation*. On the other hand, we also divide common corruption patterns into the *scene-level* and the *object-level*. As an initial effort, the dataset covers representative but not all corruptions, and we encourage continuous work with more diverse corruptions considered in the future.

The simulation of corruptions implemented in the paper mainly operates on the spatial locations and the reflection intensity of points in the point cloud. Those point-targeting operations are equivalent to the perturbations of the real-world corruptions on the LiDAR point cloud and have been widely utilized in the simulation-related studies, as in noise-related [24], [23], [18], [11], [46], density-related [18], [11], [46], [60], [13], and transformation-related [18], [11], [29], [27], [60]. Next, We briefly introduce each corruption pattern in the following (refer to Appendix C for detailed implementations and visualizations).

Weather corruptions: LiDAR is sensitive to adversarial weather conditions, such as rainy, snowy, and foggy [9]. Dense droplets of liquid or solid water dim the reflection intensity and reduce the signal-to-noise ratio (SNR) of received lights. Floating droplets can also reflect and fool sensors with false alarms. Both effects, in some cases, can significantly affect the detectors. To simulate three weather corruptions: $\{rain,$

TABLE III: Classification on real and simulated data

Corruption	Training			Testing		
	dataset	size	val accuracy	dataset	size	test accuracy
<i>snow</i>	Boreas	24292	99.11%	KITTI	14962	97.13%
<i>fog</i>	STF	1787	80.00%	KITTI	14962	92.60%

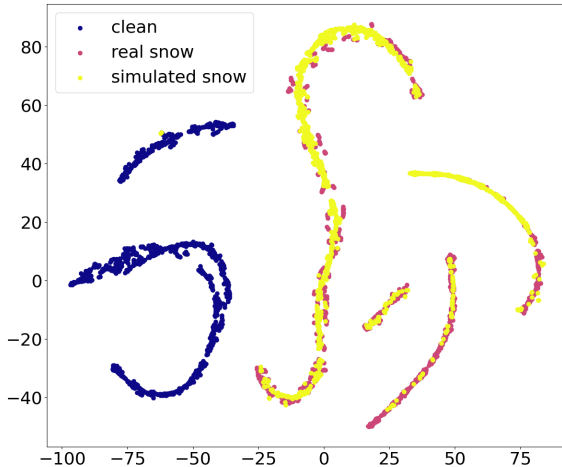
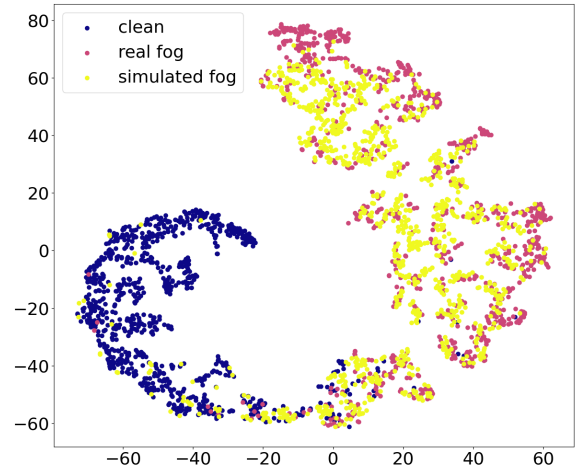
TABLE IV: MMD distances among clean, real corrupted, and simulated corrupted 2D features transformed by T-SNE

	Real vs Simulated	Simulated vs Clean	Real vs Clean
<i>snow</i>	0.0549	0.1446	0.1445
<i>fog</i>	0.0302	0.1212	0.1295

snow, fog}, we adopt LiDAR light scattering augmentation (LISA) [12] as a simulator for rain and snow and LiDAR fog stimulation (LFS) [48] as a fog simulator.

To verify the naturalness of weather simulation, we train weather-oriented PointNet-based classifiers with datasets collected in real snowy and foggy weather. Then, we leverage the classification accuracy of those trained classifiers testing on simulated data to measure the similarity of simulated data to real data. As shown in Table III, the testing accuracy 97.13% and 92.60% of trained weather classifiers on simulated snow data and fog data show that the simulated snow data and fog data are highly similar to the real data (refer to Appendix B-A for detailed experiment settings).

We further analyze the similarity of distribution of real and simulated corrupted data. Specifically, we extract the high-level features from the trained classifier. Then, we utilize T-SNE [61] to reduce the dimensionality of acquired features to 2 and visualize these 2D features. As shown in Figure 1 and 2, the distributions of the real and simulated corruptions are significantly similar. We further quantitatively measure the distance between the feature distribution of clean data, simulated *snow/fog*, and real *snow/fog*, as shown in Table IV. The maximum mean discrepancy (MMD) [62] results reveal that the simulated *snow/fog* is close to the real *snow/fog*, respectively, while not close to the clean data.

**Fig. 1:** Feature visualization of the snow classification by T-SNE**Fig. 2:** Feature visualization of the fog classification by T-SNE

Regarding *rain* corruption, we find the effects of rain droplets on point clouds are too subtle to be caught by classifiers, as shown in Figure 3. Alternatively, we visualize simulated and real point clouds and qualitatively verify the high similarity between simulated and real point clouds (see more comparisons in Appendix B-A).

Noise corruptions: Noise commonly exists in point cloud signals [53], [51]. Scene-level factors (*e.g.*, strong illumination [22], limited ranging accuracy of sensors, and sensor vibration [24], [25]) could increase the variance of ranging or extend the positioning bias. Floating particles, *e.g.*, dust [26], could cause the background noise in point clouds. Hence, we collect 5 scene-level noise corruptions: $\{uniform_rad, gaussian_rad, impulse_rad\}$ add uniform, Gaussian, impulse noise on the spherical coordinates of points in point clouds; $\{upsample\}$ randomly upsamples points nearby original points in point clouds; $\{background\}$ uniformly randomly samples points within the spatial range of point clouds. Besides scene-level effects, object-related factors could cause noise in LiDAR points, *e.g.*, dark color [21] and coarse surface. Thus, we formulate 4 object-level corruptions: $\{uniform, gaussian, impulse\}$ add uniform, Gaussian, impulse noise on the Cartesian coordinates of points of objects; $\{upsample\}$ upsamples points nearby original points of objects.

Density corruptions: The density-related corruptions refer to the corruption patterns that change the global or local density distribution of LiDAR point clouds. For instance, the global static density of points in LiDAR varies due to different amounts of scanning layer (*e.g.*, 32 or 64). Besides, inter-object occlusion and random signal loss [13] could remove points randomly. We hence propose 5 corruptions: $\{cutout\}$ cuts out the sets of locally gathering points; $\{local_dec, local_inc\}$ locally decrease or increase the density of points; $beam_del, layer_del$ randomly delete points or layers of points in point clouds. In terms of object-level factors, dark-color cover [21] and transparent materials (*e.g.*, glasses and plastics) of objects can affect the point density of objects. Hence, at the object level, we also propose a set of corruptions: $\{cutout, local_dec, local_inc\}$, affecting the point density of objects.

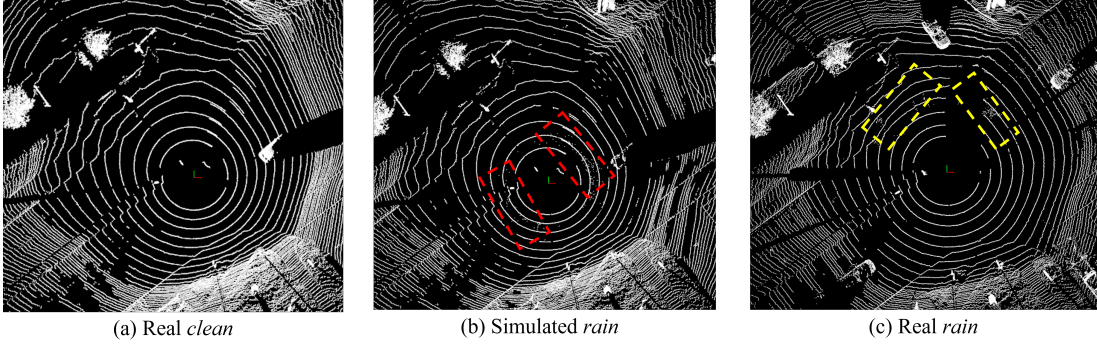


Fig. 3: Comparison between real *rain* and simulated *rain* (red and yellow boxes contain the false points in the simulated and real *rain*, respectively; the data of real *clean* and real *rain* from Boreas were sampled at the same location; the simulated *rain* data was augmented on the basis of the real *clean* data)

Transformation corruptions: In the scenario of autonomous driving, shapes of objects within one class could be various (e.g., flat sports cars and round vintage cars [29], bending and walking pedestrians [28]). Those long-tail data could potentially be recognized wrong. Besides, dynamic changes in heading directions and locations of objects [27] could potentially affect the positioning accuracy of detectors. Hence, we formulate 5 corruptions: $\{translation, rotation\}$ change locations and heading directions of objects to a milder degree, *i.e.*, $< 1m$ and $< 10^\circ$; $\{shear\}$ [63] and $\{scale\}$, as linear deformations, slant and scale points of objects; $\{FFD\}$ adopts free-form deformation (FFD) [64] to distort the point shape of an object in a nonlinear manner.

Dataset selection. As one of the most popular benchmarks in autonomous driving, KITTI [6] contains 7481 training samples covering 8 object classes. Unlike other datasets in Table I, the data in KITTI are mostly collected under clean conditions and also have a relatively simple annotation format, which makes it a good option for conducting comparative experiments. We also encourage the future extension to other real or synthesized datasets. To simulate various levels of severity in the real world, we set 6 severity levels for each corruption (considering “clean” as level 0).

B. Evaluation Metrics

To quantify the robustness performance of detectors, we design the following evaluation metrics from two perspectives: (1) detection accuracy and (2) number of bugs triggered.

Overall accuracy. For each test, we use the overall accuracy (OA), by taking the average of APs (*average precision*) at three difficulty levels (*i.e.*, “Easy”, “Moderate”, and “Hard”). And we follow the common settings of IoU thresholds {Car: 0.7, Pedestrian: 0.5, Cyclist: 0.5} to search for the true positive detections in AP and recall calculation.

For every corruption, we calculate corruption error (CE) to measure performance degradation according to OA by:

$$CE_{c,s}^m = OA_{clean}^m - OA_{c,s}^m \quad (2)$$

where $OA_{c,s}^m$ is the overall accuracy of detector m under corruption c of severity level s (exclude “clean”, *i.e.*, severity

level 0) and *clean* represent the clean data. For detection m , we can calculate the mean CE (mCE) for each detector by:

$$mCE^m = \frac{\sum_{s=1}^5 \sum_{c=1}^{25} CE_{c,s}^m}{5C} \quad (3)$$

Detection bug. There are various bugs existing in the pipeline of point cloud detection, such as annotation errors, run-time errors, detection bugs. In this paper, we focus on the bugs in detection results. Specifically, we’re interested in false detection, false classification, and missed detection:

- False detection (FD) on detection BBoxes: maximum IoU > 0 with correct classification w.r.t. ground-truth BBoxes;
- False classification (FC) on detection BBoxes: maximum IoU > 0 with false classification w.r.t. ground-truth BBoxes;
- Missed detection (MD) on detection BBoxes: maximum IoU $= 0$ w.r.t. ground-truth BBoxes.

Correspondingly, the bug rates (BRs) are calculated by:

$$BR_* = \frac{N_*}{N_{det}} \quad (4)$$

where $*$ stands for FD, FC, and MD; N_* is the number of objects of $*$; N_{det} is the number of detected objects.

To measure the increase of BR after being affected by common corruptions, we calculate corruption risk (CR) and the mean CR (mCR) for detector m by

$$CR_{*,c,s}^m = BR_{*,c,s}^m - BR_{*,clean}^m \quad (5)$$

$$mCR_*^m = \frac{\sum_{s=1}^5 \sum_{c=1}^C CR_{*,c,s}^m}{5C} \quad (6)$$

where $BR_{*,c,s}^m$ is the BR_* of detector m under corruption c of severity level s .

C. Benchmark Subjects

Point cloud detectors. For benchmarking point cloud detection, we select 8 representative detectors: SECOND [36], PointRCNN [38], PVRCNN [40], BtcDet [13], VoTr-SSD, VoTr-TSD [37], Centerpoint [54], and SE-SSD [65] to cover different kinds of feature representations and proposal architectures. We show the detailed taxonomy in Table VII.

Data augmentation and denoising methods. In this paper, we study the effectiveness of data augmentation and denoising

TABLE V: AP(%) of all detectors under clean observations (at the severity level of 0)

	PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SESSD	Centerpoint
Car	86.77	82.82	83.67	87.32	81.04	86.39	86.44	82.14
Pedestrian	60.61	52.34	52.15	-	-	-	-	49.32
Cyclist	76.42	77.60	68.51	-	-	-	-	68.58

TABLE VI: CE_{AP} (%) of different detectors under different corruptions on *Car* detection (the green cell stands for the lowest CE_{AP} among detectors given a certain corruption and the yellow cell for the average mCE_{AP})

Corruption			Point-voxel		Point		Voxel				Average	
			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint		
Scene-level	Weather	<i>rain</i>	25.11	23.31	21.81	31.07	28.17	26.77	29.51	25.83	26.45	
		<i>snow</i>	44.23	37.74	34.84	54.07	54.10	52.18	49.19	38.74	45.64	
		<i>fog</i>	1.59	3.52	1.60	1.81	1.77	2.02	1.59	1.11	1.88	
	Noise	<i>uniform_rad</i>	10.19	8.32	9.51	9.13	3.79	4.11	9.34	8.15	7.82	
		<i>gaussian_rad</i>	13.02	9.98	12.13	10.83	4.84	5.18	11.02	10.17	9.65	
		<i>impulse_rad</i>	2.20	3.86	2.23	2.50	2.25	3.57	1.18	1.86	2.46	
		<i>background</i>	2.93	6.49	2.41	1.82	4.59	3.68	2.14	1.86	2.46	
		<i>upsample</i>	0.81	1.84	0.31	0.95	0.37	0.71	0.55	0.46	0.75	
	Density	<i>cutout</i>	3.75	3.97	4.27	3.99	4.51	3.59	4.26	4.11	4.06	
		<i>local_dec</i>	14.04	-	13.88	14.55	14.44	12.50	17.04	14.64	14.44	
		<i>local_inc</i>	1.40	3.34	1.33	2.20	1.66	1.69	0.90	0.95	1.68	
		<i>beam_del</i>	0.58	0.79	0.73	0.88	0.80	0.53	1.07	0.47	0.73	
		<i>layer_del</i>	2.94	3.46	3.10	3.39	3.29	3.16	3.37	2.67	3.17	
	Object-level	Noise	<i>uniform</i>	15.44	12.95	9.48	12.60	2.76	4.81	6.99	6.51	8.94
			<i>gaussian</i>	20.48	17.62	12.98	17.05	4.72	7.46	9.56	9.49	12.42
<i>impulse</i>			3.30	4.70	2.53	4.07	2.88	4.29	2.20	2.11	3.26	
<i>upsample</i>			1.12	1.95	0.67	1.33	0.08	0.40	0.22	0.16	0.74	
Density		<i>cutout</i>	15.81	15.62	14.99	15.62	15.07	16.09	16.51	14.06	15.47	
		<i>local_dec</i>	14.38	14.16	13.23	14.26	12.66	14.41	15.08	12.52	13.84	
		<i>local_inc</i>	13.93	14.19	13.74	13.56	11.34	13.05	11.03	11.64	12.81	
Transformation		<i>shear</i>	37.27	40.96	40.35	41.37	39.52	37.85	40.35	40.00	39.71	
		<i>FFD</i>	32.42	38.88	33.15	36.77	33.14	34.26	37.96	32.86	34.93	
		<i>rotation</i>	0.60	0.47	0.31	0.97	0.39	0.75	0.27	0.38	0.52	
		<i>scale</i>	5.78	8.13	6.96	5.81	8.53	6.50	6.53	7.50	6.97	
		<i>translation</i>	3.82	3.03	3.24	4.58	4.88	5.34	1.37	3.91	3.77	
mCE			11.49	11.64	10.39	12.21	10.42	10.60	11.17	10.09	11.01	

TABLE VII: Subject point cloud detectors.

Detectors	Representations			Proposal Architectures	
	point	voxel	point-voxel	one-stage	two-stage
SECOND		✓		✓	
PointRCNN	✓				✓
PVRCNN			✓		✓
BtcDet		✓			✓
VoTr-SSD		✓		✓	
VoTr-TSD		✓			✓
SE-SSD		✓		✓	
Centerpoint		✓			✓

methods for improving detectors' robustness against corruption. For data augmentation, we choose the part-aware data augmentation (PA-DA) method [46]. For denoising, we adopt K-nearest-neighbors-based outlier removing (KNN-OR) [53] to remove the outliers out with 3 times the standard deviation of distance distribution within the cluster of 50 points.

V. EXPERIMENTS AND ANALYSIS

A. Experimental Set-ups

For a fair comparison, each detector in Table VII is trained with the clean training set of KITTI, following the training strategy in each paper, and evaluated with corrupted validation sets of KITTI. All detectors are executed based on the open-source codes released on GitHub, as shown in Table XIV

in the Appendix A. The configuration files and pre-trained checkpoints can be found in Table XIV.

The training and evaluation are all executed on the NVIDIA RTX A6000 GPU with a memory of 48GB. The batch size of each detector is optimized to reach the limit of GPU memory. In the experiments of robustness enhancement, data augmentation is adopted to augment the clean *train* dataset before training and the denoising directly processes the *val* data during the testing stage. Note that, since only detection of "Car" is available for all detectors, as shown in Table V, the following evaluation will mainly focus on detected results in the "Car" category. We encourage readers refer to the Appendix A for complete evaluation results, *e.g.*, about "Pedestrian".

B. Effects of Common Corruptions to Point Cloud Detectors

How do different corruptions affect detectors' overall accuracy? As shown in the yellow cell in Table VI, the average mCE_{AP} of 11.01% anticipates a noticeable accuracy drop of detectors against diverse corruption patterns. These results suggest that there is an urgent need of addressing the point cloud detector's robustness issue.

Specifically, $\{rain, snow\}$ and $\{shear, FFD\}$ corruptions have the AP loss of more than 20% (last column in Table VI), which presents a serious degradation of detection accuracy. By contrast, some corruption patterns (*e.g.* scene-level and

TABLE VIII: $CE_{recall}(\%)$ of different detectors under different corruptions on *Car* detection (the green cell with CE of over 15%)

Corruption			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint	Average	
Scene-level	Weather	<i>rain</i>	24.50	23.67	20.92	29.69	23.46	24.98	27.10	24.84	24.90	
		<i>snow</i>	36.23	32.72	29.19	43.32	37.11	40.26	38.32	32.80	36.24	
		<i>fog</i>	4.22	5.30	3.14	2.36	2.45	2.43	2.28	3.47	3.21	
	Noise	<i>uniform_rad</i>	14.23	12.91	13.67	9.20	3.70	5.12	9.33	11.31	9.93	
		<i>gaussian_rad</i>	17.74	15.67	17.46	11.30	4.81	6.39	11.29	14.35	12.38	
		<i>impulse_rad</i>	4.17	4.47	4.32	3.93	3.86	5.02	2.11	4.05	3.99	
		<i>background</i>	3.26	8.66	2.84	2.21	4.64	4.73	2.59	2.42	3.92	
		<i>upsample</i>	0.94	2.58	1.20	0.93	0.94	0.68	0.80	0.76	1.10	
	Density	<i>cutout</i>	4.87	4.61	5.55	4.05	4.37	3.55	4.45	5.55	4.62	
		<i>local_dec</i>	15.56	-	14.76	13.04	11.78	11.10	14.06	16.19	13.78	
		<i>local_inc</i>	1.58	2.62	1.59	1.66	1.70	1.48	1.21	1.56	1.68	
		<i>beam_del</i>	0.92	0.77	1.13	0.95	0.94	0.63	1.34	1.11	0.97	
		<i>layer_del</i>	3.48	3.37	3.59	3.12	3.13	2.74	3.37	3.78	3.32	
	Object-level	Noise	<i>uniform</i>	9.60	10.19	6.74	9.22	2.65	3.67	7.05	5.82	6.87
			<i>gaussian</i>	12.69	13.02	9.16	12.05	3.84	5.18	9.06	7.78	9.10
<i>impulse</i>			2.11	3.14	1.88	2.03	1.99	1.94	1.95	1.86	2.11	
<i>upsample</i>			0.77	1.71	0.96	1.00	0.44	0.26	0.44	0.57	0.77	
Density		<i>cutout</i>	22.21	20.90	21.61	17.61	16.27	17.44	17.67	21.21	19.36	
		<i>local_dec</i>	19.99	18.73	19.04	15.97	14.11	15.52	16.26	18.96	17.32	
		<i>local_inc</i>	10.58	10.95	10.78	9.30	7.63	8.11	7.89	9.33	9.32	
Transformation		<i>shear</i>	22.13	25.19	25.06	23.26	22.14	20.61	21.70	23.95	23.00	
		<i>FFD</i>	17.53	21.51	18.41	19.43	16.74	17.26	18.70	18.43	18.50	
		<i>rotation</i>	0.49	0.42	0.4	0.63	0.4	0.42	0.53	0.26	0.44	
		<i>scaling</i>	5.1	5.95	5.95	4.56	6.09	4.77	4.82	5.86	5.39	
		<i>translation</i>	3.79	3.42	3.78	4.69	5.32	4.66	2.39	4.26	4.04	
mCE			10.35	10.52	9.73	9.82	8.02	8.36	9.07	9.62	9.45	

TABLE IX: $CE_{AP}(\%)$ under different severity levels of different common corruptions on *Car* detection (yellow cells for the CE_{AP} under *rain*)

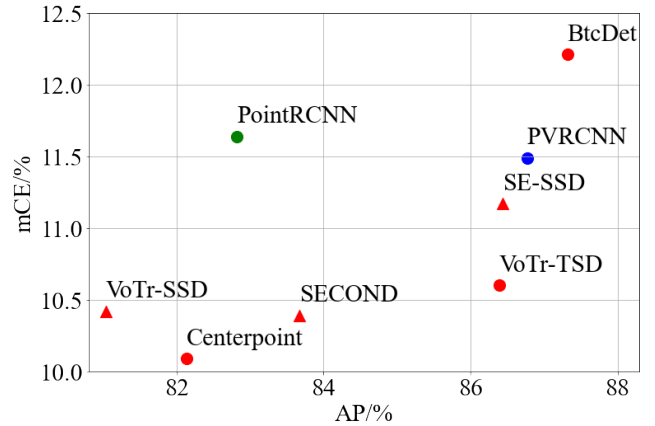
Corruption			1	2	3	4	5	Average	
Scene-level	Weather	<i>rain</i>	27.11	26.80	25.44	25.80	27.08	26.45	
		<i>snow</i>	26.86	30.93	45.82	57.09	67.48	45.64	
		<i>fog</i>	0.05	0.49	1.22	2.68	4.94	1.88	
	Noise	<i>uniform_rad</i>	0.46	2.48	6.27	11.64	18.22	7.81	
		<i>gaussian_rad</i>	1.65	4.36	8.81	13.67	19.74	9.65	
		<i>impulse_rad</i>	1.05	1.53	1.98	2.74	4.97	2.45	
		<i>background</i>	2.13	2.49	2.90	3.28	5.48	3.26	
		<i>upsample</i>	0.30	0.31	0.53	0.75	1.85	0.75	
	Density	<i>cutout</i>	1.86	2.31	3.90	5.04	7.17	4.06	
		<i>local_dec</i>	5.20	6.71	9.44	15.05	35.82	14.44	
		<i>local_inc</i>	0.82	1.05	1.53	2.10	2.92	1.68	
		<i>beam_del</i>	0.05	0.10	0.41	0.93	2.15	0.73	
		<i>layer_del</i>	0.39	2.23	2.82	4.53	5.89	3.17	
	Object-level	Noise	<i>uniform</i>	0.62	2.04	5.78	12.57	23.71	8.94
			<i>gaussian</i>	1.54	4.26	9.09	17.54	29.66	12.42
<i>impulse</i>			1.86	2.42	3.08	3.64	5.31	3.26	
<i>upsample</i>			0.37	0.54	0.57	0.79	1.45	0.74	
Density		<i>cutout</i>	5.97	11.45	16.08	20.29	23.56	15.47	
		<i>local_dec</i>	2.10	10.35	15.00	19.02	22.70	13.83	
		<i>local_inc</i>	8.04	11.87	13.82	14.85	15.46	12.81	
Transformation		<i>shear</i>	3.99	15.49	37.54	63.85	77.67	39.71	
		<i>FFD</i>	2.39	14.04	34.85	55.59	67.78	34.93	
		<i>rotation</i>	0.05	0.19	0.26	0.84	1.25	0.52	
		<i>scale</i>	0.46	2.34	5.23	9.65	17.16	6.97	
		<i>translation</i>	0.98	3.85	5.06	4.22	4.75	3.77	
Average			3.85	6.43	10.30	14.73	19.77	11.01	

object-level *upsample*, scene-level *beam_del*, and object-level *rotation*) show less effects on detectors (CE_{AP} less than 1%). It demonstrates that upsampling noise, sparse beam loss, and slight rotation don't affect detectors' accuracy.

Besides, as shown in Table VIII, the recall metric performs similarly to AP, as the serious recall loss of over 20% happens to {*rain*, *snow*, *shear*}. In addition, object-level {*cutout*, *local_dec*, *FFD*} present an unignorable drop of recall within [15%, 20%].

How do corruption severity levels affect detectors' overall accuracy? We find almost all common corruptions have a predictable trend, *i.e.*, each corruption's CE_{AP} increases as

its severity level increases (see Table IX for detailed results). The only exception is *rain*, CE_{AP} of which remain around 26% regardless of the severity level. There are two plausible explanations: (1) noise points reflected by rain droplets are too sparse to affect detection (see Figure 3), and (2) a vast amount of points with zero-value reflection intensity in KITTI are not affected by *rain* corruptions at 1-5 severity levels, which cause a fixed accuracy drop on point cloud detection.


Fig. 4: mCE_{AP} of detectors with different representations on *Car* detection (red, green, blue for {voxel-based, point-based, voxel-point-based} detectors and {circle, triangle} for {two-stage, one-stage} ones)

C. Reacts of Detector Designing to Common Corruptions

How do different representations affect detectors? As shown in Figure 4, voxel-based Centerpoint and BtcDet record the lowest and highest CE_{AP} . For voxel-involving detection (*i.e.*, except for PointRCNN), mCE_{AP} approximately increases as AP increases. It indicates a potential trend that more

TABLE X: CE_{AP} (%) of detectors with different proposal architectures on Car detection (green cell for the lower mean CE between one-stage and two-stage detector under a certain corruption)

		Corruption	one-stage	two-stage
Scene-level	Weather	<i>rain</i>	26.50	26.42
		<i>snow</i>	46.04	45.39
		<i>fog</i>	2.01	
	Noise	<i>uniform_rad</i>	7.55	7.98
		<i>gaussian_rad</i>	9.33	9.84
		<i>impulse_rad</i>	1.89	2.80
		<i>background</i>	3.05	3.38
		<i>upsample</i>	0.41	0.95
	Density	<i>cutout</i>	4.35	3.88
		<i>local_dec</i>	15.12	13.93
		<i>local_inc</i>	1.30	1.92
		<i>beam_del</i>	0.87	0.65
<i>layer_del</i>		3.25	3.12	
Object-level	Noise	<i>uniform</i>	6.41	10.46
		<i>gaussian</i>	9.09	14.42
		<i>impulse</i>	2.54	3.69
		<i>upsample</i>	0.32	0.99
	Density	<i>cutout</i>	15.52	15.44
		<i>local_dec</i>	13.66	13.95
		<i>local_inc</i>	12.04	13.27
	Transformation	<i>shear</i>	40.07	39.49
		<i>FFD</i>	34.75	35.04
		<i>rotation</i>	0.32	0.63
		<i>scale</i>	7.34	6.74
		<i>translation</i>	3.16	4.14
mCE			10.66	11.21

accurate models trend to become less robust against common corruptions.

We also find that, for the most of corruptions (except $\{shear, FFD, scale\}$), voxel-based methods are generally more robust against corruption patterns (as shown in Table VI). One plausible explanation is that the spatial quantization of a group of neighbor points by voxelization mitigates the local randomness and the absence of points caused by Noise and Density corruptions. Specifically, for severe corruptions (e.g., *shear*; *FFD* in the Transformation), the point-voxel-based method PVRCNN is more robust. The point-based PointRCNN doesn't have the most robust performance against any corruption, suggesting potential limitations.

How do different proposal architectures affect detectors?

As shown in Figure 4, two-stage detectors perform less robust against common corruptions compared to one-stage detectors, showing a lower mCE_{AP} . One possible cause is that corrupted data could affect the proposal generation of stage 1 (for two-stage detectors and one-stage ones), and the low-quality proposals significantly affect the BBox regression of stage 2 (only for two-stage detectors).

As shown in Table X, two-stage detectors present more accurate detection under the scene-level $\{cutout, local_dec, beam_del, layer_del\}$ and object-level $\{cutout, shear, scale\}$, displaying a lower average CE_{AP} , while one-stage detectors present more accurate under the rest of common corruptions. In summary, one-stage detectors perform more robust against corruptions of scene-level Noise and object-level Noise and Density, while two-stage detectors are mainly more robust

TABLE XI: Bug rates (%) of true detection, false classification, false detection, and missing detection of detectors under different corruptions

		Corruption	TD	FC	FD	MD
		<i>Clean</i>	43.81	0.37	9.81	46.01
Scene-level	Weather	<i>rain</i>	42.93	1.12	17.43	38.52
		<i>snow</i>	35.11	1.34	19.96	43.58
		<i>fog</i>	41.99	0.46	9.91	47.64
	Noise	<i>uniform_rad</i>	42.86	0.95	13.79	42.40
		<i>gaussian_rad</i>	42.35	1.16	14.52	41.97
		<i>impulse_rad</i>	43.37	0.46	11.70	44.48
		<i>background</i>	36.99	0.30	9.22	53.49
		<i>upsample</i>	42.46	0.35	10.00	47.19
	Density	<i>cutout</i>	41.79	0.50	10.75	46.97
		<i>local_dec</i>	39.63	0.74	13.00	46.63
		<i>local_inc</i>	42.94	0.40	10.30	46.36
		<i>beam_del</i>	43.90	0.41	10.24	45.44
<i>layer_del</i>		42.64	0.47	10.77	46.13	
Object-level	Noise	<i>uniform</i>	41.15	0.55	12.13	46.16
		<i>gaussian</i>	40.08	0.61	12.97	46.33
		<i>impulse</i>	42.75	0.38	10.89	45.98
		<i>upsample</i>	43.59	0.40	10.20	45.81
	Density	<i>cutout</i>	36.82	0.66	11.16	51.36
		<i>local_dec</i>	37.70	0.58	10.69	51.02
		<i>local_inc</i>	39.09	0.45	14.01	46.45
	Transformation	<i>shear</i>	29.11	0.47	24.44	45.99
		<i>FFD</i>	31.77	0.44	21.64	46.15
		<i>rotation</i>	43.61	0.37	10.05	45.96
		<i>scaling</i>	40.44	0.38	13.09	46.08
		<i>translation</i>	42.08	0.47	11.45	46.00

TABLE XII: Bug rates (%) of true detection, false classification, false detection, and missing detection of different detectors under corruptions (testing results on *clean* data are in parentheses)

Detector	TD	FC	FD	MD
PVRCNN	32.63 (36.27)	1.16 (0.72)	11.1 (8.28)	55.11 (54.73)
PointRCNN	47.11 (50.20)	0.68 (0.40)	16.04 (12.47)	36.17 (36.93)
SECOND	20.61 (23.57)	0.81 (0.49)	8.42 (6.51)	70.16 (69.43)
BtcDet	65.24 (68.19)	0.03 (0.01)	15.19 (11.13)	19.54 (20.67)
VoTr-SSD	27.95 (32.17)	0.62 (0.55)	13.52 (10.43)	57.91 (56.85)
VoTr-TSD	42.75 (47.20)	0.28 (0.20)	14.67 (10.72)	42.3 (41.88)
SE-SSD	64.56 (68.18)	0.05 (0.03)	15.72 (11.45)	19.67 (20.34)
Centerpoint	21.74 (24.70)	0.98 (0.54)	9.26 (7.53)	68.02 (67.23)

against Weather and scene-level Density. As for Transformation corruptions, one-stage detectors present better robustness on $\{FFD, rotation, translation\}$ and two-stage detectors work better under corruptions of $\{shear, scale\}$.

D. Detection Bugs in Detectors under Common Corruptions

How do different corrupted inputs trigger bugs in detectors?

We find that the rate of false classification (FC) against common corruption patterns is relatively small, where the largest CR_{FC} is only 0.97% (refer to Table XV in Appendix A-C). By contrast, the increase of false detection (FD) rate is relatively obvious, by the average CR_{FD} of 3.17% and the largest CR_{FD} of 14.61% (refer to Table XVI in Appendix A-C). Regarding missed detection (MD) (refer to Table XVII in Appendix A-C), scene-level $\{background\}$ and object-level $\{cutout, local_dec\}$ result in an increase of MD rate of more than 5%.

Surprisingly, according to Table XI, $\{rain\}$ and scene-level $\{uniform_rad, gaussian_rad\}$ even reduce the rate of missing objects. One plausible explanation for this observation is that

TABLE XIII: Average CE_{AP} (%) of detectors given different common corruptions on *Car* detection with DA and/or denoising (the differences between enhancement methods and **Origin** are in parentheses)

Corruption		Origin	PA-DA	KNN-RO	PA-DA + KNN-RO	
Scene-level	Weather	<i>rain</i>	26.45	27.51 (+1.06)	32.03 (+5.58)	32.79 (+6.34)
		<i>snow</i>	45.64	45.68 (+0.04)	47.76 (+2.12)	47.83 (+2.19)
		<i>fog</i>	1.88	2.0 (+0.12)	5.18 (+3.3)	5.09 (+3.21)
	Noise	<i>uniform_rad</i>	7.82	7.85 (+0.03)	10.79 (+2.97)	10.79 (+2.97)
		<i>gaussian_rad</i>	9.65	9.59 (-0.06)	12.65 (+3.0)	12.47 (+2.82)
		<i>impulse_rad</i>	2.46	2.01 (-0.45)	4.99 (+2.53)	4.53 (+2.07)
		<i>background</i>	3.25	3.08 (-0.17)	2.36 (-0.89)	2.13 (-1.12)
		<i>upsample</i>	0.75	0.65 (-0.1)	3.61 (+2.86)	3.4 (+2.65)
	Density	<i>cutout</i>	4.06	3.97 (-0.09)	6.71 (+2.65)	6.52 (+2.46)
		<i>local_dec</i>	14.44	14.83 (+0.39)	16.73 (+2.29)	17.0 (+2.56)
		<i>local_inc</i>	1.68	1.49 (-0.19)	4.39 (+2.71)	4.21 (+2.53)
		<i>beam_del</i>	0.73	0.77 (+0.04)	3.45 (+2.72)	3.39 (+2.66)
<i>layer_del</i>		3.17	3.19 (+0.02)	6.23 (+3.06)	6.2 (+3.03)	
Object-level	Noise	<i>uniform</i>	8.94	7.95 (-0.99)	11.55 (+2.61)	10.66 (+1.72)
		<i>gaussian</i>	12.42	11.46 (-0.96)	14.99 (+2.57)	14.05 (+1.63)
		<i>impulse</i>	3.26	2.96 (-0.3)	6.07 (+2.81)	5.76 (+2.5)
		<i>upsample</i>	0.74	0.57 (-0.17)	2.77 (+2.03)	2.4 (+1.66)
	Density	<i>cutout</i>	15.47	15.1 (-0.37)	17.15 (+1.68)	16.8 (+1.33)
		<i>local_dec</i>	13.84	13.62 (-0.22)	16.18 (+2.34)	15.92 (+2.08)
		<i>local_inc</i>	12.81	11.8 (-1.01)	15.01 (+2.2)	14.14 (+1.33)
	Transformation	<i>shear</i>	39.71	39.72 (+0.01)	40.06 (+0.35)	39.97 (+0.26)
		<i>FFD</i>	34.93	34.34 (-0.59)	40.55 (+5.62)	40.02 (+5.09)
		<i>rotation</i>	0.52	0.52 (+0.0)	3.19 (+2.67)	3.19 (+2.67)
		<i>scaling</i>	6.97	7.01 (+0.04)	9.4 (+2.43)	9.42 (+2.45)
		<i>translation</i>	3.77	3.48 (-0.29)	6.22 (+2.45)	5.76 (+1.99)
Average		11.01	10.85 (-0.16)	13.6 (+2.59)	13.38 (+2.37)	

milder noise points offer a better knowledge of the shape of some objects to detectors, but positioning on those objects is not accurate since the rate of false detection increases (more details in Table XVI and XVII in Appendix A-C).

Also, we find that, as shown in Figure 5, compared to clean observations, TD rates under corrupted observations are always lower at any distance of objects to LiDAR.

How do corrupted inputs trigger bugs in different detectors? In general, as shown in Table XII, most of the detectors perform relatively stable in terms of false classification rates and missed detection rates against common corruptions. In contrast, affected by corruptions, all detectors have increasing false detection rates (refer to Table XII), revealing a serious bias in BBox localization of point cloud detection. Among all detectors, BtcDet and SE-SSD records a serious FD increase

of over 4%.

E. Robustness Enhancement by Data Augmentation and Denoising

How do PA-DA and KNN-based outlier-removing affect detectors' robustness against different corruptions? Shown by Table XIII, the average CE_{AP} with PA-DA slightly decreased to 10.85% compared to the average CE_{AP} without PA-DA, which still poses serious robustness issues for point cloud detectors.

Regarding denoising strategy, the average CE_{AP} after adopting KNN-RO increases to 13.60% without PA-DA and 13.38% with PA-DA (refer to Table XIII). These results indicate that KNN-RO might not be capable of enhancing point cloud detectors' robustness against most of the corruptions in *Car* detection. However, we find that KNN-RO slightly improves the robustness of *Pedestrian* detection by decreasing the CE_{AP} by 0.14% without PA-DA and 1.19% with PA-DA (Table XVIII in Appendix A-D).

How do PA-DA and KNN-based outlier-removing affect different detectors' robustness against corruptions? Except for PVRCNN, SE-SSD, and Centerpoint, all the other detectors perform more robust against corruption patterns after adopting PA-DA (refer to Figure 6). Moreover, PointRCNN and VoTr-SSD increase their AP by 1.16% and 2.46% after adopting PA-DA, respectively.

According to Figure 6, KNN-RO degrade AP metric for all detectors, presenting no improvement on the robustness of any detector on *Car* detection. However, adopting KNN-RO slightly improves the AP by 0.14% without PA-DA and 1.19% with PA-DA on *Pedestrian* detection, respectively (refer to Table XIX in Appendix A-D). It illustrates that compared with effects on *Car* objects, KNN-RO are more effective in

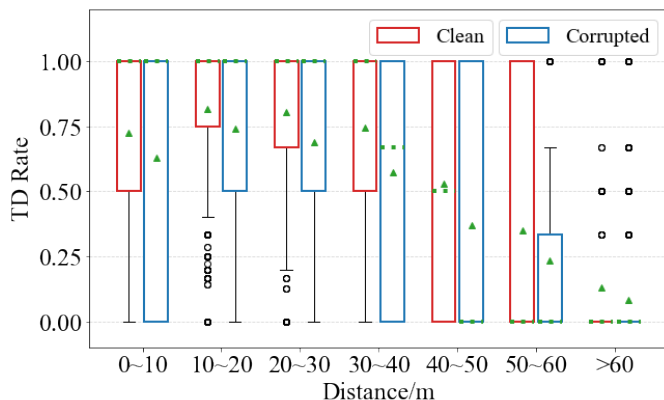


Fig. 5: Box-plot of TD rate of all frames w.r.t. different distances of objects to the LiDAR sensor (green dotted lines for the median and green triangles for the mean)

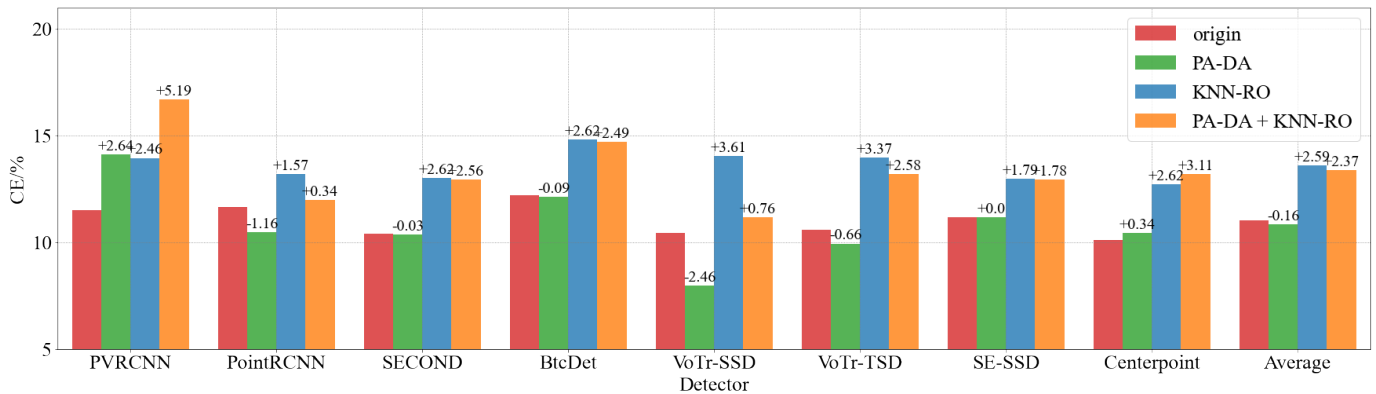


Fig. 6: Average $CE_{AP}(\%)$ of different detectors on *Car* detection given common corruptions

removing perturbations caused by corruptions for *Pedestrian* objects.

VI. CONCLUSION

In this paper, we propose the first physical-aware robustness benchmark of point cloud detection against common corruption patterns, which contains a total of 1,122,150 examples covering 25 common corruption types and 6 severity levels. Based on the benchmark, we conduct extensive empirical studies on 8 detectors covering 6 different detection frameworks and reveal the vulnerabilities of the detectors. Moreover, we further study the effectiveness of existing data augmentation and denoising methods and find them limited, calling for more research on robustness enhancement. We hope this benchmark and empirical study results can guide future research toward building more robust and reliable point cloud detectors.

REFERENCES

- [1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.
- [2] R. Qian, X. Lai, and X. Li, "3d object detection for autonomous driving: a survey," *arXiv preprint arXiv:2106.10823*, 2021.
- [3] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.
- [4] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.
- [5] D. Fernandes, A. Silva, R. Névoa, C. Simoes, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, "Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy," *Information Fusion*, vol. 68, pp. 161–191, 2021.
- [6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nusenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.
- [8] P. Sun, H. Kretzschmar, X. Dotiwala, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. 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*, 2020, pp. 2446–2454.
- [9] R. H. Rasshofer, M. Spies, and H. Spies, "Influences of weather phenomena on automotive laser radar systems," *Advances in radio science*, vol. 9, no. B. 2, pp. 49–60, 2011.
- [10] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide, "Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [11] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, and Z. M. Mao, "Benchmarking robustness of 3d point cloud recognition against common corruptions," *arXiv preprint arXiv:2201.12296*, 2022.
- [12] V. Kilic, D. Hegde, V. Sindagi, A. B. Cooper, M. A. Foster, and V. M. Patel, "Lidar light scattering augmentation (lisa): Physics-based simulation of adverse weather conditions for 3d object detection," *arXiv preprint arXiv:2107.07004*, 2021.
- [13] Q. Xu, Y. Zhong, and U. Neumann, "Behind the curtain: Learning occluded shapes for 3d object detection," *arXiv preprint arXiv:2112.02205*, 2021.
- [14] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, "Canadian adverse driving conditions dataset," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 681–690, 2021.
- [15] K. Burnett, D. J. Yoon, Y. Wu, A. Z. Li, H. Zhang, S. Lu, J. Qian, W.-K. Tseng, A. Lambert, K. Y. Leung *et al.*, "Boreas: A multi-season autonomous driving dataset," *arXiv preprint arXiv:2203.10168*, 2022.
- [16] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.
- [17] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," *Advances in neural information processing systems*, vol. 32, 2019.
- [18] J. Ren, L. Pan, and Z. Liu, "Benchmarking and analyzing point cloud classification under corruptions," *arXiv preprint arXiv:2202.03377*, 2022.
- [19] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.
- [20] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Level 5 perception dataset 2020," <https://level-5.global/level5/data/>, 2019.
- [21] D. Bolkas and A. Martinez, "Effect of target color and scanning geometry on terrestrial lidar point-cloud noise and plane fitting," *Journal of applied geodesy*, vol. 12, no. 1, pp. 109–127, 2018.
- [22] F. Villa, F. Severini, F. Madonini, and F. Zappa, "Spads and sipms arrays for long-range high-speed light detection and ranging (lidar)," *Sensors*, vol. 21, no. 11, p. 3839, 2021.
- [23] T. Instruments, "Lidar pulsed time of flight reference design," 2016.
- [24] H. Ma and J. Wu, "Analysis of positioning errors caused by platform vibration of airborne lidar system," in *2012 8th IEEE International Symposium on Instrumentation and Control Technology (ISICT) Proceedings*. IEEE, 2012, pp. 257–261.
- [25] R. Wang, B. Wang, M. Xiang, C. Li, S. Wang, and C. Song, "Simultaneous time-varying vibration and nonlinearity compensation for one-period

- triangular-fmcw lidar signal,” *Remote Sensing*, vol. 13, no. 9, p. 1731, 2021.
- [26] L. Mona, Z. Liu, D. Müller, A. Omar, A. Papayannis, G. Pappalardo, N. Sugimoto, and M. Vaughan, “Lidar measurements for desert dust characterization: an overview,” *Advances in Meteorology*, vol. 2012, 2012.
- [27] X. Morin-Duchesne and M. S. Langer, “Simulated lidar repositioning: a novel point cloud data augmentation method,” *arXiv preprint arXiv:2111.10650*, 2021.
- [28] C.-C. Wong and C.-M. Vong, “Efficient outdoor 3d point cloud semantic segmentation for critical road objects and distributed contexts,” in *European Conference on Computer Vision*. Springer, 2020, pp. 499–514.
- [29] Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, “Train in germany, test in the usa: Making 3d object detectors generalize,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 713–11 723.
- [30] C. Xiang, C. R. Qi, and B. Li, “Generating 3d adversarial point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9136–9144.
- [31] D. Liu, R. Yu, and H. Su, “Extending adversarial attacks and defenses to deep 3d point cloud classifiers,” in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 2279–2283.
- [32] M. Abdelfattah, K. Yuan, Z. J. Wang, and R. Ward, “Adversarial attacks on camera-lidar models for 3d car detection,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2189–2194.
- [33] Y. Zhu, C. Miao, F. Hajiaghajani, M. Huai, L. Su, and C. Qiao, “Adversarial attacks against lidar semantic segmentation in autonomous driving,” in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 329–342.
- [34] B. Li, T. Zhang, and T. Xia, “Vehicle detection from 3d lidar using fully convolutional network,” *arXiv preprint arXiv:1608.07916*, 2016.
- [35] B. Yang, W. Luo, and R. Urtasun, “Pixor: Real-time 3d object detection from point clouds,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.
- [36] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, p. 3337, 2018.
- [37] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, “Voxel transformer for 3d object detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3164–3173.
- [38] S. Shi, X. Wang, and H. Li, “Pointcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.
- [39] Z. Yang, Y. Sun, S. Liu, and J. Jia, “3dssd: Point-based 3d single stage object detector,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.
- [40] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. 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*, 2020, pp. 10 529–10 538.
- [41] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, “Structure aware single-stage 3d object detection from point cloud,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 873–11 882.
- [42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [43] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.
- [44] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu, “Pointcutmix: Regularization strategy for point cloud classification,” *arXiv preprint arXiv:2101.01461*, 2021.
- [45] D. Lee, J. Lee, J. Lee, H. Lee, M. Lee, S. Woo, and S. Lee, “Regularization strategy for point cloud via rigidly mixed sample,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 900–15 909.
- [46] J. Choi, Y. Song, and N. Kwak, “Part-aware data augmentation for 3d object detection in point cloud,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3391–3397.
- [47] J. Fang, X. Zuo, D. Zhou, S. Jin, S. Wang, and L. Zhang, “Lidar-aug: A general rendering-based augmentation framework for 3d object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4710–4720.
- [48] M. Hahner, C. Sakaridis, D. Dai, and L. Van Gool, “Fog simulation on real lidar point clouds for 3d object detection in adverse weather,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 283–15 292.
- [49] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, “Point-bert: Pre-training 3d point cloud transformers with masked point modeling,” *arXiv preprint arXiv:2111.14819*, 2021.
- [50] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra, “Self-supervised pretraining of 3d features on any point-cloud,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 252–10 263.
- [51] Y. Duan, C. Yang, H. Chen, W. Yan, and H. Li, “Low-complexity point cloud denoising for lidar by pca-based dimension reduction,” *Optics Communications*, vol. 482, p. 126567, 2021.
- [52] X. Ning, F. Li, G. Tian, and Y. Wang, “An efficient outlier removal method for scattered point cloud data,” *PLoS one*, vol. 13, no. 8, p. e0201280, 2018.
- [53] A. Carrilho, M. Galo, and R. Santos, “Statistical outlier detection method for airborne lidar data,” *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2018.
- [54] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3d object detection and tracking,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.
- [55] M. Ye, S. Xu, and T. Cao, “Hvnet: Hybrid voxel network for lidar based 3d object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1631–1640.
- [56] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [57] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *Advances in neural information processing systems*, vol. 30, 2017.
- [58] D. A. Van Dyk and X.-L. Meng, “The art of data augmentation,” *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.
- [59] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. Snoek, “Pointmixup: Augmentation for point clouds,” in *European Conference on Computer Vision*. Springer, 2020, pp. 330–345.
- [60] S. Cheng, Z. Leng, E. D. Cubuk, B. Zoph, C. Bai, J. Ngiam, Y. Song, B. Caine, V. Vasudevan, C. Li *et al.*, “Improving 3d object detection through progressive population based augmentation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 279–294.
- [61] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [62] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, “A kernel method for the two-sample-problem,” *Advances in neural information processing systems*, vol. 19, 2006.
- [63] B. Chen and A. Kaufman, “3d volume rotation using shear transformations,” *Graphical Models*, vol. 62, no. 4, pp. 308–322, 2000.
- [64] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 151–160.
- [65] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, “Se-ssd: Self-ensembling single-stage object detector from point cloud,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 494–14 503.
- [66] O. D. Team, “Openpcdet: An open-source toolbox for 3d object detection from point clouds,” <https://github.com/open-mmlab/OpenPCDet>, 2020.
- [67] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, “Class-balanced grouping and sampling for point cloud 3d object detection,” *arXiv preprint arXiv:1908.09492*, 2019.
- [68] Wikipedia contributors, “Rain — Wikipedia, the free encyclopedia,” <https://en.wikipedia.org/w/index.php?title=Rain&oldid=1102188829>, 2022, [Online; accessed 5-August-2022].

APPENDIX A
DETAILS OF EMPIRICAL STUDY

Due to the page limit of supplementary materials, we present some tables and figures on our supplementary website <https://sites.google.com/ualberta.ca/robustness1pc2detector/>. We also encourage readers to refer to this **supplementary** website for additional details.

TABLE XIV: Code sources of detectors

Detector	Platform	URL(http://)
SECOND	Openpcdet [66]	github.com/open-mmlab/OpenPCDet
PointRCNN	Openpcdet	github.com/open-mmlab/OpenPCDet
PVRCNN	Openpcdet	github.com/open-mmlab/OpenPCDet
BtcDet	Openpcdet	github.com/xharlie/btcdet
VoTr-SSD	Openpcdet	github.com/PointsCoder/VOTR
VoTr-TSD	Openpcdet	github.com/PointsCoder/VOTR
SE-SSD	Det3D [67]	github.com/Vegeta2020/SE-SSD
Centerpoint	Openpcdet	github.com/tianweiy/CenterPoint-KITTI

A. Effects of Common Corruptions to Point Cloud Detectors

As shown in Table S1 in the **supplementary**, the average mCE_{AP} of 8.18% still anticipates a noticeable accuracy drop of *Pedestrian* detectors against diverse corruption patterns. Specifically, scene-level $\{uniform_rad, gaussian_rad, local_dec\}$ and object-level $\{cutout\}$ corruptions have the AP loss of more than 20%, which presents a serious degradation of detection accuracy. By contrast, some corruption patterns (e.g. scene-level $\{background, beam_del\}$, object-level $\{upsample, rotation, translation\}$) show less effects on detectors (absolute value of CE_{AP} less than 1.25%), demonstrating that background and locally upsampling noise, sparse beam loss, and slight rotation and translation don't affect *Pedestrian* detectors' accuracy. Surprisingly, compared to *Car* detection, *Pedestrian* detection are much less affected by *rain* and *snow*, presented by the average CE_{AP} of 2.42% and 2.58%. After the investigation of point clouds, we found it is because the proportion (58.94%)

of points with zero-value reflection intensity on *Car* objects is much higher than that (10.01%) of pedestrians, and those points with zero-value reflection intensity are easy to be blocked by dense rain and snow droplets.

Table S2 in the **supplementary** shows the CE_{AP} under different severity levels of corruptions on *Pedestrian* detection. According to Table S2, albeit with some minor exceptions, CE_{AP} of each corruption increases as the severity level increases, which especially rigorously applies to those relatively severe corruptions with the average CE_{AP} of more than 5%.

B. Reacts of Detector Designing to Common Corruptions

Figure S1 in the **supplementary** depicts the relationship between of AP and mCE_{AP} of *Pedestrian* detectors. As shown in Figure S1, similar to *Car* detection, the mCE_{AP} of *Pedestrian* detection increases as its AP increases.

C. Detection Bugs in Detectors under Common Corruptions

More details of the increase of bug rates (i.e., CR) of different detectors under different corruptions are recorded in Table XV, Table XVI, and Table XVII.

According to Figure 5, the TD rate of *Car* detection under clean and corrupted (at the severity level of 3) observations w.r.t. the distance range of [0, 30]m remains approximately still. However, beyond this distance range (i.e., distance > 30m), the TD rate decreases as the distance increases. On the other hand, compared to the clean observation, the TD rates under corrupted observations are always lower at any distance. More details of each corruption are shown in Figure S2 and S3 in the **supplementary**.

D. Robustness Enhancement by Data Augmentation and Denoising

More details of CE_{AP} of different detectors on *Car* detection with PA-DA (data augmentation) and/or KNN-RO (denoising)

TABLE XV: CR_{FC} (%) of detectors on *Car* detection under common corruptions

Corruption		PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint	Average		
Scene-level	Weather	<i>rain</i>	1.90	1.27	0.61	-0.01	-0.09	0.06	-0.01	2.29	0.75	
		<i>snow</i>	2.17	2.01	0.71	0.03	0.17	0.12	-0.01	2.59	0.97	
		<i>fog</i>	0.03	0.02	0.10	0.02	0.21	0.16	0.04	0.16	0.09	
	Noise	<i>uniform_rad</i>	1.35	0.82	1.27	0.00	-0.06	0.00	-0.01	1.29	0.58	
		<i>gaussian_rad</i>	1.83	1.25	1.67	0.00	-0.06	0.01	0.00	1.62	0.79	
		<i>impulse_rad</i>	0.12	0.01	0.14	0.02	0.07	0.06	0.04	0.25	0.09	
		<i>background</i>	-0.18	0.20	-0.19	0.01	-0.22	-0.01	0.04	-0.20	-0.07	
		<i>upsample</i>	-0.01	0.03	0.03	0.01	-0.15	-0.02	0.01	-0.01	-0.01	
	Density	<i>cutout</i>	0.21	0.13	0.16	0.03	0.18	0.15	0.04	0.12	0.13	
		<i>local_dec</i>	0.70	-0.40	0.47	0.05	0.37	0.29	0.03	0.70	0.28	
		<i>local_inc</i>	0.04	0.02	0.05	0.01	0.00	0.05	0.05	0.04	0.03	
		<i>beam_del</i>	0.08	0.04	0.04	0.01	0.08	0.05	0.01	0.05	0.04	
		<i>layer_del</i>	0.13	0.07	0.09	0.03	0.17	0.17	0.04	0.09	0.10	
	Object-level	Noise	<i>uniform</i>	0.53	0.24	0.39	0.01	-0.01	0.00	0.00	0.32	0.19
			<i>gaussian</i>	0.71	0.32	0.50	0.01	-0.01	0.00	0.01	0.41	0.24
<i>impulse</i>			0.03	0.00	0.04	-0.01	0.00	0.02	0.00	0.00	0.01	
<i>upsample</i>			0.06	0.00	0.12	0.00	0.01	0.02	0.03	0.04	0.03	
Density		<i>cutout</i>	0.38	0.03	0.51	0.08	0.49	0.41	0.04	0.40	0.29	
		<i>local_dec</i>	0.26	-0.03	0.40	0.07	0.37	0.31	0.04	0.32	0.22	
		<i>local_inc</i>	0.23	0.12	0.29	0.00	-0.13	-0.02	0.01	0.17	0.08	
Transformation		<i>shear</i>	0.16	-0.01	0.25	0.01	0.17	0.06	0.01	0.16	0.10	
		<i>FFD</i>	0.08	0.02	0.18	0.01	0.08	0.06	0.03	0.11	0.07	
		<i>rotation</i>	-0.01	0.02	0.01	0.01	0.00	0.01	0.01	0.00	0.01	
		<i>scaling</i>	0.02	0.00	0.03	0.01	0.02	0.01	0.03	0.00	0.01	
		<i>translation</i>	0.14	0.09	0.14	0.03	0.10	0.09	0.05	0.15	0.10	
mCR		0.44	0.25	0.32	0.02	0.07	0.08	0.02	0.44	0.20		

TABLE XVI: CR_{FD} (%) of detectors on *Car* detection under common corruptions

Corruption		PVRCNN	PointRCNN	SECOND	BteDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint	Average	
Scene-level	Weather	<i>rain</i>	8.25	4.23	5.70	7.90	13.39	11.46	4.78	5.24	7.62
		<i>snow</i>	8.54	7.45	5.93	14.51	12.90	16.34	10.11	5.39	10.15
		<i>fog</i>	0.82	0.11	0.62	-0.30	-0.84	-0.49	0.10	0.71	0.09
	Noise	<i>uniform_rad</i>	4.08	5.94	3.30	3.81	2.61	4.01	5.52	2.50	3.97
		<i>gaussian_rad</i>	4.88	6.83	3.87	4.31	3.34	4.94	6.44	3.02	4.70
		<i>impulse_rad</i>	1.54	2.30	0.99	1.81	2.55	3.72	1.22	0.92	1.88
		<i>background</i>	-0.34	2.25	-2.25	0.34	-3.22	-0.14	0.78	-2.14	-0.59
		<i>upsample</i>	-0.04	0.82	-0.34	0.34	0.29	0.14	0.69	-0.44	0.18
	Density	<i>cutout</i>	0.73	0.57	0.58	1.11	1.05	1.17	1.87	0.41	0.94
		<i>local_dec</i>	3.79	-	2.77	3.40	3.44	4.29	4.43	2.83	3.56
		<i>local_inc</i>	0.39	0.95	0.23	0.43	0.53	0.55	0.48	0.30	0.48
		<i>beam_del</i>	0.39	0.06	0.39	0.54	0.52	0.48	0.75	0.30	0.43
<i>layer_del</i>		0.98	0.54	0.79	1.25	0.98	1.01	1.36	0.73	0.95	
Object-level	Noise	<i>uniform</i>	2.30	4.51	1.25	2.85	1.37	2.13	3.01	1.10	2.31
		<i>gaussian</i>	3.11	6.00	1.70	3.99	1.98	3.11	3.91	1.47	3.16
		<i>impulse</i>	0.89	1.47	0.53	1.40	1.06	1.32	1.43	0.53	1.08
		<i>upsample</i>	0.28	0.79	0.22	0.33	0.24	0.13	0.95	0.16	0.39
	Density	<i>cutout</i>	2.29	-0.92	1.64	0.39	2.75	3.27	0.54	0.77	1.34
		<i>local_dec</i>	1.83	-1.15	1.27	0.10	2.02	2.57	-0.13	0.48	0.87
		<i>local_inc</i>	3.54	5.44	2.54	4.81	3.43	4.74	6.77	2.31	4.20
	Transformation	<i>shear</i>	10.08	17.48	7.56	23.27	11.70	14.84	24.29	7.74	14.62
		<i>FFD</i>	8.06	14.85	5.59	18.56	8.82	12.28	20.52	5.95	11.83
		<i>rotation</i>	0.22	0.13	0.12	0.04	0.25	0.29	0.41	0.07	0.24
		<i>scaling</i>	2.31	3.82	1.81	4.40	3.24	3.38	5.37	1.90	3.28
		<i>translation</i>	1.58	1.09	0.87	1.55	2.78	3.24	1.09	0.88	1.63
mCR		2.82	3.56	1.91	4.06	3.09	3.95	4.27	1.73	3.17	

under different corruptions are shown in Table S3, Table S4, and Table S5 in the **supplementary**.

Table XVIII gives the details of average AP loss (*i.e.*, CE_{AP}) given different corruptions on *Pedestrian* detection with PA-DA (data augmentation) and/or KNN-RO (denoising). According to Table XVIII, PA-DA presents an AP increase of over 3% on *Pedestrian* detection under scene-level $\{uniform_rad, gaussian_rad\}$ and object-level $\{local_inc, shear\}$, and PA-DA + KNN-RO presents an AP increase of over 3% under scene-level $\{uniform_rad, gaussian_rad\}$ and object-level $\{local_inc, shear\}$.

Table XIX shows CE_{AP} of different detectors on *Pedestrian* detection with PA-DA (data augmentation) and/or KNN-RO (denoising). According to Table XIX, with only PA-DA,

an increase of average precision (AP) falls on PointRCNN, SECOND, and Centerpoint; with only KNN-RO, an increase of AP falls on PVRCNN and PointRCNN; with PA-DA + KNN-RO, an AP increase falls on PointRCNN, SECOND, and Centerpoint.

APPENDIX B

CORRUPTION SIMULATION NATURALNESS VALIDATION

A. Naturalness Validation of Weather Corruption Simulation

1) *Snow and Fog Validation*: To verify the naturalness of snow and fog simulation, we train weather-oriented PointNet-based classifiers via data collected in real snowy and foggy weather (from public datasets as in Table III).

TABLE XVII: CR_{MD} (%) of detectors on *Car* detection under common corruptions

Corruption		PVRCNN	PointRCNN	SECOND	BteDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint	Average	
Scene-level	Weather	<i>rain</i>	-9.33	-4.63	-3.42	-12.61	-8.37	-5.64	-11.01	-4.89	-7.49
		<i>snow</i>	-0.71	-3.47	0.62	-8.96	4.14	-0.61	-9.99	-0.40	-2.42
		<i>fog</i>	-0.60	-0.43	0.25	-0.54	6.77	6.53	2.70	-1.59	1.64
	Noise	<i>uniform_rad</i>	-2.10	-5.78	-2.98	-4.51	-2.79	-3.62	-4.63	-2.45	-3.61
		<i>gaussian_rad</i>	-2.32	-6.42	-3.03	-5.00	-3.37	-4.28	-5.17	-2.68	-4.03
		<i>impulse_rad</i>	0.26	-5.51	0.85	-2.23	-2.45	-2.89	-1.51	1.24	-1.53
		<i>background</i>	7.43	-7.01	13.13	-1.12	19.16	10.67	5.10	12.48	7.48
		<i>upsample</i>	2.07	-0.73	2.88	-0.62	0.79	1.20	1.07	2.80	1.18
	Density	<i>cutout</i>	1.74	1.83	1.59	-1.01	1.77	0.53	-1.03	2.26	0.96
		<i>local_dec</i>	-0.08	-	-0.09	-3.24	2.18	-0.38	-3.44	0.34	-0.67
		<i>local_inc</i>	0.39	-1.11	0.59	-0.37	1.05	1.16	0.81	0.29	0.35
		<i>beam_del</i>	-0.37	0.14	-0.48	-1.02	-0.37	-0.92	-1.27	-0.22	-0.56
<i>layer_del</i>		0.13	1.32	-0.27	-1.00	1.02	0.55	-0.99	0.20	0.12	
Object-level	Noise	<i>uniform</i>	0.90	-0.93	0.32	0.86	-0.15	-0.31	0.06	0.50	0.16
		<i>gaussian</i>	1.23	-1.00	0.49	1.33	-0.11	-0.32	0.27	0.69	0.32
		<i>impulse</i>	0.13	-0.74	0.10	0.06	0.01	-0.16	0.15	0.22	-0.03
		<i>upsample</i>	-0.02	-1.15	0.01	-0.22	-0.07	-0.11	-0.09	0.05	-0.20
	Density	<i>cutout</i>	5.05	10.70	3.62	5.00	3.76	4.81	5.15	4.74	5.35
		<i>local_dec</i>	4.76	9.90	3.36	4.70	3.57	4.52	4.89	4.43	5.02
		<i>local_inc</i>	0.52	0.05	0.30	0.86	0.24	0.23	0.85	0.45	0.44
	Transformation	<i>shear</i>	0.01	-0.60	0.05	0.40	-0.09	-0.18	0.05	0.22	-0.02
		<i>FFD</i>	0.08	-1.11	0.04	0.72	0.02	0.05	1.08	0.24	0.14
		<i>rotation</i>	0.03	-0.28	0.00	0.01	-0.06	-0.05	-0.02	0.03	-0.04
		<i>scaling</i>	0.08	0.08	0.01	0.10	-0.03	-0.04	0.31	0.09	0.07
		<i>translation</i>	0.29	-1.29	0.45	0.27	-0.07	-0.23	-0.02	0.60	0.00
mCR		0.38	-0.76	0.74	-1.13	1.06	0.42	-0.67	0.79	0.11	

TABLE XVIII: Average CE_{AP} (%) of detectors given different common corruptions on *Pedestrian* detection with DA and/or denoising

Corruption		Origin	PA-DA	KNN-RO	PA-DA + KNN-RO	
Scene-level	Weather	<i>rain</i>	2.42	1.67(-0.75)	1.6(-0.82)	1.39(-1.03)
		<i>snow</i>	2.58	3.99(+1.41)	2.0(-0.58)	3.24(+0.66)
		<i>fog</i>	6.27	4.82(-1.45)	6.17(-0.1)	4.22(-2.05)
	Noise	<i>uniform_rad</i>	30.14	25.13(-5.01)	29.79(-0.35)	24.93(-5.21)
		<i>gaussian_rad</i>	36.99	32.22(-4.77)	36.8(-0.19)	32.32(-4.67)
		<i>impulse_rad</i>	2.09	5.1(+3.01)	2.21(+0.12)	5.1(+3.01)
		<i>background</i>	0.62	0.09(-0.53)	-0.22(-0.84)	-0.29(-0.91)
		<i>upsample</i>	1.38	1.77(+0.39)	1.24(-0.14)	1.71(+0.33)
	Density	<i>cutout</i>	7.43	6.02(-1.41)	7.3(-0.13)	5.73(-1.7)
		<i>local_dec</i>	12.6	12.25(-0.35)	12.62(+0.02)	12.21(-0.39)
		<i>local_inc</i>	1.36	1.42(+0.06)	1.49(+0.13)	1.23(-0.13)
		<i>beam_del</i>	-0.02	0.23(+0.25)	-0.15(-0.13)	-0.13(-0.11)
<i>layer_del</i>		3.81	2.97(-0.84)	3.64(-0.17)	2.66(-1.15)	
Object-level	Noise	<i>uniform</i>	4.21	3.07(-1.14)	4.25(+0.04)	2.98(-1.23)
		<i>gaussian</i>	5.47	4.09(-1.38)	5.59(+0.12)	4.04(-1.43)
		<i>impulse</i>	1.85	1.74(-0.11)	2.05(+0.2)	1.79(-0.06)
		<i>upsample</i>	0.08	-0.21(-0.29)	-0.14(-0.22)	-0.37(-0.45)
	Density	<i>cutout</i>	20.34	18.15(-2.19)	20.07(-0.27)	18.04(-2.3)
		<i>local_dec</i>	17.05	15.71(-1.34)	17.09(+0.04)	15.71(-1.34)
		<i>local_inc</i>	8.21	5.09(-3.12)	8.03(-0.18)	5.09(-3.12)
	Transformation	<i>shear</i>	16.91	13.22(-3.69)	16.85(-0.06)	12.87(-4.04)
		<i>FFD</i>	11.74	9.5(-2.24)	11.7(-0.04)	9.34(-2.4)
		<i>rotation</i>	-0.03	0.23(+0.26)	-0.09(-0.06)	0.05(+0.08)
		<i>scaling</i>	3.73	3.98(+0.25)	3.65(-0.08)	3.67(-0.06)
		<i>translation</i>	-1.22	-1.42(-0.2)	-1.08(+0.14)	-1.34(-0.12)
Average		7.84	6.83(-1.01)	7.7(-0.14)	6.65(-1.19)	

TABLE XIX: Average CE_{AP} (%) of different detectors on *Pedestrian* detection with DA and/or denoising

Detector	Origin	PA-DA	KNN-RO	PA-DA + KNN-RO
PVRCNN	8.71	10.6(+1.89)	8.2(-0.51)	10.05(+1.34)
PointRCNN	7.08	3.73(-3.35)	7.05(-0.03)	3.57(-3.51)
SECOND	8.38	7.63(-0.75)	8.46(+0.08)	7.76(-0.62)
Centerpoint	7.19	5.37(-1.82)	7.91(+0.72)	5.21(-1.98)
Average	7.84	6.83(-1.01)	7.7(-0.14)	6.65(-1.19)

Dataset Split: As for snow or fog, we randomly divide collected real corrupted and clean data into training set D_{train} and validation set D_{val} at the size ratio of 9:1, and gather the original clean and simulated corrupted KITTI data as testing set D_{test} . Note that, D_{train} , D_{val} , and D_{test} all contain roughly 50% corrupted data and 50% clean data.

Training and Testing: We first train the PointNet model with D_{train} to learn the model to classify the clean or corrupted condition of point clouds. By means of D_{val} , we validate the model across all epochs and settle its best checkpoint. The settled model is utilized to classify the clean or corrupted condition of point clouds in D_{test} .

Experiment Setting: As for the experiment setting, we set the epoch number to 80 and the batch size to 32 for the model training and validation. The training and testing are all executed on the NVIDIA RTX A6000 GPU with a memory of 48GB. We choose PointNet as the classification model due to its efficiency and effectiveness in recognizing the global point feature in point clouds [56], which fits into the global effects of weather corruptions on LiDAR scanning.

Analysis: According to the snow classification in III, the validation accuracy of 99.11% illustrates the snow classifier’s precise snow recognition on point clouds collected in the real world. Hence, the classifier’s testing accuracy of 97.13% on simulated data demonstrates the high naturalness of the snow simulation. Likewise, even though the fog classifier’s validation

accuracy is reduced by the small data size, the testing accuracy of 92.60% on simulated data still presents a high similarity of data corrupted by simulated fog to data affected by real fog.

2) *Rain Validation:* In Figure 3, data of real clean and real rain from Boreas were sampled at the same location but at different time-stamps; the data of simulated rain was augmented on the basis of the data of real clean. According to Figure 3, compared with the data of real clean, the point cloud under simulated rain has sparse “false points” (as in red boxes) surrounding the LiDAR sensor nearby and wipes out some points on the road. Both are similar to the effects of real-world rain, shown by the yellow boxes and missing points on the road. More comparisons between real rainy data and simulated rainy data are shown in Figure S4 in the **supplementary**.

APPENDIX C

IMPLEMENTATION OF CORRUPTION SIMULATION

Figures S5 to S30 in the **supplementary** display the point cloud examples under “clean” and simulated common corruptions (at severity level of 3). Those examples are based on the KITTI LiDAR example with ID = “000008”. Besides, we provide the ground-truth annotations of objects and detection results obtained by PVRCNN. Taking Figures S6 to S30 for examples, we next detail on the simulation implementation.

A. Scene-level Corruption

Rain and Snow. We adopt the rain and snow simulators of LISA [12] to simulate rain and snow corruption. For LISA, the parameters *rainfall rate* and *snow rate* can be regulated to simulate corruptions at different severity levels. After the investigation of the real-world rainfall rates [68], we set *rainfall rate* and *snow rate* to $\{0, 5.0, 15.0, 50.0, 150.0, 500.0\}$ mm/hr and $\{0, 0.5, 1.5, 5.0, 15.0, 50.0\}$ mm/hr as 6 severity levels (*i.e.*, 0 to 5) for rain and snow corruption, respectively. Figures S6 and S7 display point cloud examples under *rain* and *snow*. Note that, the severity level 0 stands for the original clean data, which also applies to the rest of the corruptions.

Fog. We adopt the fog simulator of LFS [48] to simulate fog corruption in point clouds. For LFS, the parameter α is set to regulate the corruption severity levels. Following the recommended setting in [48] (with slight modifications for the relatively wide range of severity), we set α to $\{0, 0.005, 0.01, 0.02, 0.05, 0.1\}$. Figure S8 displays the point cloud example under fog.

Uniform_rad and Gaussian_rad. To fit into the mechanism of LiDAR scanning, we first convert the coordinates of points at from the Cartesian system into the spherical system (*i.e.*, $[x, y, z]$ to $[r, \theta, \phi]$). Then, we add the uniform or Gaussian noise into r of every point. The upper and lower bounds of the range of *uniform_rad* are set to +/- $\{0, 0.04, 0.08, 0.12, 0.16, 0.2\}$ m and the standard deviation of *gaussian_rad* to $\{0, 0.04, 0.06, 0.08, 0.10, 0.12\}$ m. Figures S9 and S10 display the examples under *uniform_rad* and *gaussian_rad*.

Impulse_rad. Likewise, we first convert coordinates of points at from the Cartesian system to the spherical system, and then, we add deterministic perturbations of $+/- 0.2m$ to r of a certain portion of points. The portion is set to

$\{0, N/30, N/25, N/20, N/15, N/10\}$ for 6 severity levels, where N represents the number of points. Figure S11 displays the point cloud example under *impulse_rad*.

Background. For *background*, within the spatial range of the scene, background points are randomly sampled uniformly and concatenated to the original points. The number of background points is set to $\{0, N/45, N/40, N/35, N/30, N/20\}$ where N represents the number of original points. Figure S12 displays the point cloud example under *background*.

Upsample. For *upsample*, we spatially upsample points (with the random bias within $[-0.1, 0.1]$) nearby a certain portion of original points. The portion is set to $\{0, N/10, N/8, N/6, N/4, N/2\}$ for 6 severity levels, where N represents the number of original points. Figure S13 displays the point cloud example under *upsample*.

Cutout. For *cutout*, we first randomly select a certain portion of points as centers. By KNN, we erase the distance-related neighborhood of every centers. The portion of selected points and the number of neighbor points are set to $\{(0,0), (N/2000,100), (N/1500,100), (N/1000,100), (N/800,100), (N/600,100)\}$. Figure S14 displays the point cloud example under *cutout*.

Local_dec and local_inc. For *local_dec*, we randomly select a certain portion of points as center, and delete the 75% points of the spatial neighborhood of every center. For *local_inc*, within the neighborhood of every center, we utilize quadratic-polynomial fitting to upsample points as many as the neighbor points and concatenate them into the original points. The portion of selected points and the number of neighbor points are set to $\{(0, 0), (N/300,100), (N/250,100), (N/200,100), (N/150,100), (N/100,100)\}$ for *local_dec* and $\{(0, 0), (N/2000,100), (N/1500,100), (N/1000,100), (N/800,100), (N/600,100)\}$ for *local_inc*. Figures S15 and S16 display the point cloud examples under *local_dec* and *local_inc*.

Beam_del. For *beam_del*, we randomly delete a certain portion of points in the point cloud. The portion is set to $\{0, N/100, N/30, N/10, N/5, N/3\}$ for 6 severity levels. Figure S17 displays the point cloud example under *beam_del*.

Layer_del. First, we convert the coordinates of points from the Cartesian system into the spherical system and obtain the range of the polar angle θ of all points in the point cloud. Then, based on the layer number of LiDAR scanning (e.g., 64 for KITTI [6]), we divide the range of θ into 32 or 64 bins. Finally, we randomly select a certain number of θ bins and delete the corresponding points. For KITTI, the number of deleted bins is set to $\{0, 3, 7, 11, 15, 19\}$. Figure S18 displays the point cloud example under *layer_del*.

B. Object-level Corruption

Uniform_obj, Gaussian_obj, and Impulse_obj. For every annotated object (e.g., Car or Cyclist), we add the noise into the Cartesian coordinates of its points. The upper and lower bounds of the range of *uniform_obj* are set to $\pm \{0, 0.02, 0.04, 0.06, 0.08, 0.10\}m$. The the standard deviation of *gaussian_obj* is set to $\{0, 0.02, 0.03, 0.04, 0.05, 0.06\}m$. The number of points affected by *impulse_obj* with the bias of $\pm 0.1m$ is set to $\{0, N/30, N/25, N/20, N/15, N/10\}$. Figures S19, S20, and S21 display the point cloud examples under *uniform_obj*, *gaussian_obj*, and *impulse_obj*.

Upsample_obj. We upsampled points (with the spatial bias within $[-0.05, 0.05]$) nearby a certain portion of points of annotated objects. The portion is set to $\{0, N/5, N/4, N/3, N/2, N\}$, where N represents the number of original points of objects. Figure S22 displays the point cloud example under *upsample_obj*.

Cutout_obj. For *cutout_obj*, we erase the distance-related neighborhoods of selected points from every annotated object in the point cloud. For the individual object, the number of selected points and the number of neighbor points are set to $\{(0, 0), (1, 20), (2, 20), (3, 20), (4, 20), (5, 20)\}$. Figure S23 displays the point cloud example under *cutout_obj*.

Local_dec_obj and Local_inc_obj. For *local_dec_obj*, we randomly select a certain number of points of every annotated object as centers, and delete the 75% points of the spatial neighborhood of every center. For *local_inc_obj*, within the neighborhood of every centers on objects, we utilize linearly fitting to upsample points as many as the neighbor points. The number of selected points and the number of neighbor points are set to $\{(0, 0), (1, 30), (2, 30), (3, 30), (4, 30), (5, 30)\}$ for *local_dec_obj* and *local_inc_obj*. Figures S24 and S25 display the point cloud examples under *local_dec_obj* and *local_inc_obj*.

Shear. For *shear*, we slant points of objects on X and Y-axis by a transformation matrix $A = \begin{bmatrix} 1 & a & b \\ c & 1 & d \\ 0 & 0 & 1 \end{bmatrix}$ where a, b, c, d are $\pm 1 \times$ a float sampled on the uniform distribution. The upper and lower boundaries of the uniform distribution are set to $\{(0, 0), (0, 0.10), (0.05, 0.15), (0.10, 0.20), (0.15, 0.25), (0.20, 0.30)\}$. Figure S26 displays the point cloud example under *shear*.

FFD. We use the *FFD* tool of *pygem* to distort points of objects. With the prior setting of $5 \times 5 \times 5$ control points, the distortion ratio is sampled on the uniform distribution. The upper and lower boundaries of the uniform distribution are set to $\pm \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Figure S27 displays the point cloud example under *FFD*.

Scale. Along the randomly selected axis (height, length, or width), we scale up or down points of objects by a transformation matrix $A = \begin{bmatrix} xs & 0 & 0 \\ 0 & ys & 0 \\ 0 & 0 & zs \end{bmatrix}$. The scaling parameter randomly selected among xs, ys, zs are set to $1 \pm \{0, 0.04, 0.08, 0.12, 0.16, 0.20\}$. Note that for scaling on Z-axis, we correspondingly move the object to the ground. Also, the ground-truth labels of objects (specifically, dimensions and locations of BBoxes) are modified accordingly. Figure S28 displays the point cloud example under *scale*.

Rotation and Translation. We rotate or translate annotated objects to a milder degree. Specifically, objects are 1) moved forward or backward on the X and Y-axis at a distance sampled on the uniform distribution $U_{distance}$, and are 2) rotated in a clockwise or anticlockwise direction to a degree sampled on the uniform distribution U_{degree} . The lower and upper boundaries of U_{degree} are set to $\{(0, 0), (0, 2), (3, 4), (5, 6), (7, 8), (9, 10)\}$ degree and those of $U_{distance}$ to $\{(0, 0), (0.0, 0.2), (0.3, 0.4), (0.5, 0.6), (0.7, 0.8), (0.9, 0.1)\}m$. Note that the ground-truth labels of objects are modified accordingly. Figures S29 and S30 display the examples under *rotation* and *translation*.