Common Corruption Robustness of Point Cloud Detectors: Benchmark and Enhancement

Shuangzhi Li, *Student Member, IEEE,* Zhijie Wang, *Student Member, IEEE,* Felix Juefei-Xu[†], *Member, IEEE,* Qing Guo[*], *Member, IEEE,* Xingyu Li, *Member, IEEE,* and Lei Ma, *Member, IEEE*

***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 12 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, data denoising, test-time adaptation. The benchmark can potentially be a new platform for evaluating point cloud detectors, opening a door for developing novel robustness enhancement methods. We make this benchmark publicly available on https://github.com/Castiel-Lee/robustness_pc_detector.**

***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.

[*] 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 The University of Tokyo, Japan. (e-mail: {shuangzh, zhijie.wang, xingyu}@ualberta.ca, ma.lei@acm.org)

Qing Guo is with the Institute of High Performance Computing (IHPC) and Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR). (e-mail: tsingqguo@ieee.org)

Felix Juefei-Xu is with Meta AI, New York, NY 10001, USA. (e-mail: felixu@meta.com)

[†] This work is done prior to joining Meta.

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	-
ONCE [19]	2021	real	1M(15K labeled)	417K	5	rain, clouds, cutout, noise	2	-
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 [20]	2019	real	468K	993K	15	rain, cutout, dust, noise	2	-
Lyft Level 5 [21]	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	✓

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 collect existing simulation methods for common corruptions and improve them based on their affecting ranges and physical mechanisms of formation Then, we borrow 7,481 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 12 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 10.94% (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 overall corruptions. Compared with point-based detectors, voxel-involving detectors perform more robust against a majority 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 improve point cloud detectors against common corruption patterns?** Even with the help of data augmentation, denoising, and test-time adaptation, common corruptions still cause a severe accuracy drop of over 10%.

In summary, this work makes the following contributions:

- We design the first robustness benchmark of point cloud detection covering 25 common corruptions related to natural weather, noise disturbance, density change, and object transformations at the object and scene level.
- Based on the benchmark, we conduct extensive empirical studies to evaluate the robustness of 12 existing detectors to reveal the vulnerabilities of the detectors under common corruptions.
- We study the existing methods of data augmentation, denoising, and test-time adaptation and explore their 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 [23] affects the signal transmission by lowering Signal-to-Noise Ratio (SNR), increasing the noise level of LiDAR ranging [24]. Besides, the intrinsically inaccurately ranging and the sensor vibration [25], [26] potentially trigger noisy observations during LiDAR scanning. Environmental floating particles (*e.g.*, dust

TABLE II: Taxonomy of collected common corruption patterns

Scene-level			Object-level		
Corruption Category	**Corruption**	**Potential Reasons**	**Corruption Category**	**Corruption**	**Potential Reasons**
Weather	*rain*	**Environment**: natural weather [9];	**Noise**	*uniform*	**Object surface**: coarse surface [22] and dark-color cover [22];
	snow			*gaussian*	
	fog			*impulse*	
Noise	*uniform_rad*	**Environment**: strong illumination [23]; **Sensor**: low ranging accuracy [24] and sensor vibration [25], [26];		*upsample*	
	gaussian_rad		**Density**	*cutout*	**Object surface**: object or self-occlusions [13], dark-color cover [22] and transparent components;
	impulse_rad			*local_dec*	
	upsample			*local_inc*	
	background	**Environment**: floating particles [27];	**Transformation**	*translation*	**Object**: different locations and heading directions [28];
Density	*cutout*	**Sensor**: different scanning layers, object occlusion [13], and randomly laser beam [13] or layer (rotary laser) malfunction;		*rotation*	
	local_dec			*shear*	**Object deformation**: bending or moving pedestrians [29], different styles of vehicles [30].
	local_inc			*FFD*	
	beam_del			*scale*	
	layer_del				

[27]) 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 [22] 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 [29], [30], locations and poses [28] 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 [31] pose significant security issues and vulnerability on 3D point cloud tasks (*e.g.*, classification [32], detection [33], and segmentation [34]).

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 [35] and PIXOR [36]), **voxel-based detectors** (*e.g.*, SECOND [37] and VoTr [38]), **point-based** detectors (*e.g.*, PointRCNN [39] and 3D-SSD [40]), and **point-voxel-based** detectors (*e.g.*, PVRCNN [41] and SA-SSD [42]). 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 [40] and SA-SSD [42]) and **two-stage** detectors (*e.g.*, PointRCNN [39] and PVRCNN [41]). In this paper, we select 12 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 [43], 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 [44] 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 [7], Waymo [8], and STF [10]collect LiDAR scans under adversarial rainy, snowy, and foggy conditions, where the accuracy of 3D detectors is tested. 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 [45] 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.* [46] 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.* [47] propose a part-aware data augmentation that stochastically augments the partitions of objects by 5 basic augmentation methods. LiDAR-Aug [48] presents a rendering-based LiDAR augmentation framework to improve the robustness of 3D object detectors. LiDAR light scattering augmentation [12] and LiDAR fog stimulation [49] utilize physics-based simulators to generate data corrupted by fog/snow/rain and then augment object detectors. Lehner *et al.* [50] improve the generalization of 3D object detectors to bad-shape objects by means of adversarial vector fields. Self-supervised pre-training [51], [52] can also endow the model with resistance to augmentation-related transformations. Besides, denoising methods [53], [54], [55] can remove the outliers in point clouds and thus potentially improve detectors' robustness. Also, test-time BN [56] adapt the statistics of BN layers to models during testing for generalization to diverse test-time domains. 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 [57] with key-point detector for a flexible orientation regression. In this paper, we evaluate data augmentation, denoising, test-time adaptation 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:

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

where $\mathrm{Det}(\cdot)$ 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 $i = 1, 2, \cdots, N$; (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 [35], [36]. 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 [37], [58]. After pooling interior features, those voxels are fed into a sparse 3D convolution backbone [37] 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 [59] and PointNet++ [60] 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 [41], [42] possess the potential of fusing the high-efficient voxels and accurate-abstract points in feature abstraction.

Proposal architecture. One-stage detectors [37], [52] directly generate candidate BBoxes from the abstracted features. To improve candidate BBoxes' precision, two-stage detectors [41], [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 explore data augmentation, denoising, and test-time adaptation methods and study their effects on improving point cloud detectors' robustness against common corruptions. Data augmentation [61] effectively increases the amount of relevant data by slightly modifying existing data or newly creating synthetic data from existing data. Data augmentation on the point cloud [47], [62], [50] provides detectors with a way to be trained with a larger dataset and thus potentially obtain more robust detectors. Considering most detectors commonly adopt the global/scene-level augmentations (*e.g.*, {*random_world_flip, random_world_rotate, random_world_scale*}), we compensatively choose the local/object-level part-aware data augmentation [47] which mixes up 5 basic object-level augmentation methods (*i.e.*, {*dutout, swap, mix, sparse, noise*}) in the partitions of an object. We also explore Cutout [63] and CutMix [64] in the 3D point cloud detection, which have been widely applied as 2D image augmentation methods for robustness enhancement. As the data augmentation by extreme samples generated through adversarial attacks has drawn increasing attention, we selected the adversarial shape-deformation augmentation 3D-VField [50] and study its robustness enhancement on point cloud detection.

Different from data augmentation, denoising [53], [54] serves as a pre-process during testing stage to detect and remove spatial outliers in point clouds, which can reduce the effects of noisy point cloud data. Considering not severely degrading the efficiency of detector inference, we choose the common-K-nearest-neighbors-based outlier removing (KNN-OR) [55] of high efficiency (around 0.05s on each sample), which simply removes the outlier points of over 3 times the standard deviation of distance distribution within the cluster of 50 points. Besides, test-time adaptation methods [56], [65] try to tackle data distribution shifts between training and testing data by adapting models to testing samples during testing time. Considering the test-time adaptation is relatively unexplored in the 3D point cloud detection, we adapt the test-time batch normalization (TT-BN) [56] well-explored in the image domain to point cloud detection, which utilizes testing data to update the statistics of BN layers during testing time.

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:

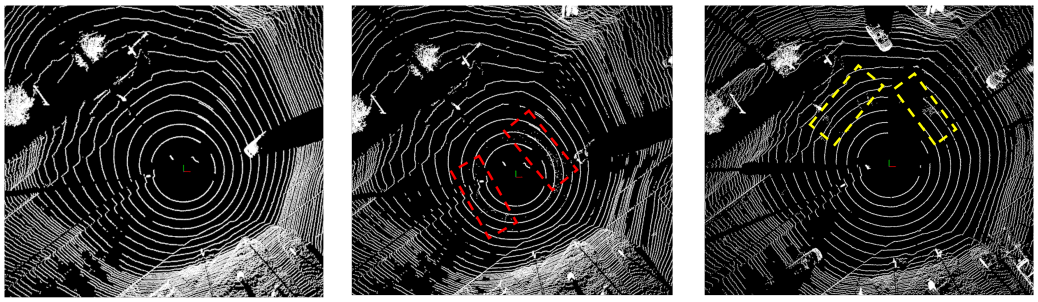


Fig. 1: Comparison between real *rain* and simulated *rain* (red and yellow boxes contain the false points in the simulated and real *rain*, respectively; the data of real *clean* and real *rain* from Boreas were sampled at the same location; the simulated *rain* data was augmented on the basis of the real *clean* data)

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 [25], [24], [18], [11], [47], density-related [18], [11], [47], [66], [13], and transformation-related [18], [11], [30], [28], [66]. Next, We briefly introduce each corruption pattern in the following (refer to Supplementary 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, snow, fog*}, we adopt LiDAR light scattering augmentation (LISA) [12] as a simulator for rain and snow and LiDAR fog stimulation (LFS) [49] as a fog simulator.

Unlike other types of corruptions which are relatively simply implemented and widely applied in LiDAR-based point cloud studies, the mechanism of weather simulation on point clouds complicatedly involved interaction between LiDAR lights and dense droplets. Since there is a lack of the realness verification study in [12], [49], we conduct experiments to further verify the realness/naturalness of weather simulators for a convincing benchmarking under weather-relevant corruptions. Specifically, 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 S12 in Supplementary B, 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

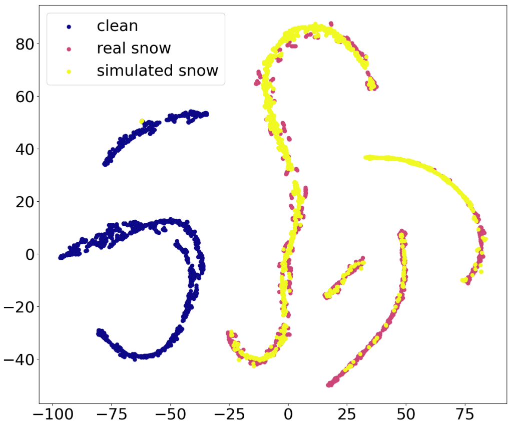


Fig. 2: Feature visualization of the snow classification by T-SNE

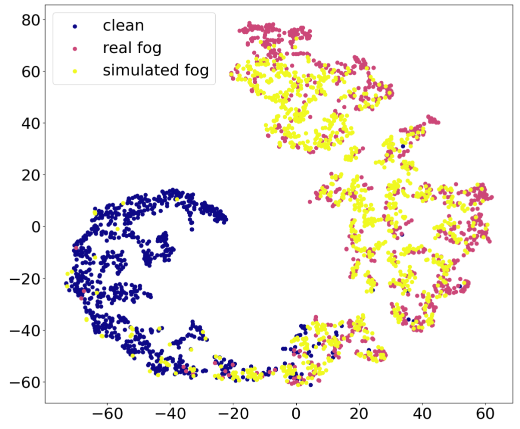


Fig. 3: Feature visualization of the fog classification by T-SNE

real data. Further, the overlapping feature distribution of real and simulated corrupted data (refer to Figures 2 and 3) and the maximum mean discrepancy (MMD) [67] results (refer to Table S13) reveals that the simulated *snow/fog* is close to the real *snow/fog*, respectively, while not close to the clean data. Please refer to Supplementary B for more details.

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 1. Alternatively, we visualize simulated and real point clouds and qualitatively verify the high similarity between simulated and real point clouds (see more comparisons in Supplementary B).

Noise corruptions: Noise commonly exists in point cloud signals [55], [53]. Scene-level factors (*e.g.*, strong illumination [23], limited ranging accuracy of sensors, and sensor vibration [25], [26]) could increase the variance of ranging or extend the positioning bias. Floating particles, *e.g.*, dust [27], 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; {*upsample*} randomly upsamples points nearby original points; {*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 [22] 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 [22] 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.

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 [30], bending and walking pedestrians [29]). Those long-tail data could potentially be recognized wrong. Besides, dynamic changes in heading directions and locations of objects [28] 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 $< 10°$; {*shear*} [68] and {*scale*}, as linear deformations, slant and scale points of objects; {*FFD*} adopts free-form deformation (FFD) [69] 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 large-scale datasets including weather and other corruptions 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:

$$\mathrm{CE}^{\mathrm{m}}_{\mathrm{c,s}} = \mathrm{OA}^{\mathrm{m}}_{\mathrm{clean}} - \mathrm{OA}^{\mathrm{m}}_{\mathrm{c,s}} \tag{2}$$

where $OA^{m}_{c,s}$ 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) over C corruptions by:

$$\mathrm{mCE}^{\mathrm{m}} = \frac{\sum_{\mathrm{s}=1}^{5}\sum_{\mathrm{c}=1}^{\mathrm{C}}\mathrm{CE}^{\mathrm{m}}_{\mathrm{c,s}}}{5\mathrm{C}} \tag{3}$$

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:

$$\mathrm{BR}_{*} = \frac{\mathrm{N}_{*}}{\mathrm{N}_{\mathrm{det}}} \tag{4}$$

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

$$\mathrm{CR}^{\mathrm{m}}_{*,\mathrm{c,s}} = \mathrm{BR}^{\mathrm{m}}_{*,\mathrm{c,s}} - \mathrm{BR}^{\mathrm{m}}_{*,\mathrm{clean}} \tag{5}$$

$$\mathrm{mCR}^{\mathrm{m}}_{*} = \frac{\sum_{\mathrm{s}=1}^{5}\sum_{\mathrm{c}=1}^{\mathrm{C}}\mathrm{CR}^{\mathrm{m}}_{*,\mathrm{c,s}}}{5\mathrm{C}} \tag{6}$$

where $BR^{m}_{*,c,s}$ is the BR_{*} of detector m under corruption c of severity level s and C for the number of corruptions.

C. Benchmark Subjects

Point cloud detectors. For benchmarking point cloud detection, we select 12 representative detectors: SECOND [37], PointRCNN [39], PartA2 [70] PVRCNN [41], PVRCNN++ [71], BtcDet [13], VoTr-SSD, VoTr-TSD [38], Centerpoint [57], Centerpoint_RCNN [57], Centerformer [72], and SE-SSD [73], to cover different kinds of feature representations and proposal architectures. We show the detailed taxonomy in Table III. For a more fair comparison, based on the robust Centerpoint detector

TABLE III: Subject point cloud detectors.

Detectors	Representations			Proposal Architectures	
	point	voxel	point-voxel	one-stage	two-stage
SECOND [37]		✓		✓	
PointRCNN [39]	✓				✓
PartA2 [70]	✓				✓
PVRCNN [41]			✓		✓
PVRCNN++ [71]			✓		✓
BtcDet [13]		✓			✓
VoTr-SSD [38]		✓		✓	
VoTr-TSD [38]		✓			✓
SE-SSD [73]		✓		✓	
Centerpoint [57]		✓		✓	
Centerpoint_RCNN [57]			✓		✓
Centerformer [72]		✓		✓	

and its two-stage version Centerpoint_RCNN, we construct 5 versions of Centerpoint detectors for ablative comparison, as in Table VI, covering 3 types of feature representations and 2 types of proposal architectures. Note that, as in [41], [71], the point-voxel-based feature extraction needs the two-stage proposal structure, where the multi-scale voxel-based features are extracted in the first-stage detection, and the point-voxel-based features combining raw points and multi-scale voxel-based features (from the first stage) are extracted for the second-stage refinement.

Data augmentation, denoising, and test-time adaptation methods. In this paper, we study the effectiveness of data augmentation, denoising, and test-time adaptation methods for improving detectors' robustness against corruption. As discussed in Sec. III-B, for data augmentation, we choose methods of part-aware data augmentation (PA-DA) [47], Cutout [63], CutMix [64], and 3D-VField [50]. For denoising, we adopt K-nearest-neighbors-based outlier removing (KNN-OR) [55] to remove the outliers out with 3 times the standard deviation of distance distribution within the cluster of 50 points. For test-time adaptation, we select the test-time batch normalization (TT-BN) [56] to update statistics of BN layers during testing time. Apart from them, we also augment *train* data with different corruption categories {*Weather, Noise, Density, Transformation*} by means of our physical-aware simulation tools to explore the robustness enhancement of the category-oriented augmentation.

V. Experiments and Analysis

A. Experimental Set-ups

For a fair comparison, each detector in Table III 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 go through the training of 80 epochs among which the best checkpoint is selected by metrics of mAP. All detectors are executed based on the open-source codes released on GitHub [74], [75], where the configuration files and pre-trained checkpoints can be found. The experiments are 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 robustness enhancement experiments, we first follow the recommended setting "dropout_p02_swap_p02_mix_p02_sparse40_p01_noise10_p01" in [47] to adopt PA-DA to augment the clean *train* set. For CutMix, we follow the settings (*i.e.*, "swap_p10") in [47] to implement the augmentation. For Cutout, we utilize the *cutout* in our corruption simulation toolkit with the randomly selected severity level for each sample. For 3D-VField augmentation, we follow the settings in [50]. The "augmented + clean" set (2x3712 samples) is used for all augmentation methods. For a fair comparison, the category-oriented data augmentation also builds "augmented + clean" set from the clean *train* set. For each basic corruption category {*Weather, Noise, Density, Transformation*} or overall corruptions, we augment every sample of KITTI *train* set with a randomly selected corruption at a randomly selected severity level. By means of the GPU-accelerated tool *remove_statistical_outlier* of the package "*open3d*", KNN-RO is implemented to denoise the *val* data during the detection inference. By modifying the parameters "running_mean" and "running_val" of PyTorch-based BN layers during testing, we implement the TT-BN to update the statistics of BN based on testing data.

Note that, since only detection of "Car" is available for all detectors, as shown in Table IV, the following evaluation will mainly focus on detected results in the "Car" category. We encourage readers to refer to Supplementary 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 V, the average mCE_{AP} of 10.94% anticipates a noticeable accuracy drop of detectors against diverse corruption patterns. It suggests 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 V), which presents a serious degradation of detection accuracy. By contrast, scene-level and 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 affect detectors' accuracy slightly.

Besides, as shown in Table S3 in Supplementary A, the recall metric performs similarly to AP, as the serious recall loss of over 22% on {*rain, snow, shear*}. In addition, object-level {*cutout, local_dec, FFD*} present an unignorable drop of recall within $[18\%, 22\%]$.

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 S1 in the Supplementary A). The only exception is *rain*, CE_{AP} of which remain around 26% regardless of the severity level. There is the plausible explanation: (1) by statistics, we find, due to the unpromising laser reflections of car surfaces, 58.94% of points of "Car" in KITTI have zero-value reflection intensity, and those points are easily filtered out by the rain droplets, causing a noticeable AP drop, and (2) noise points reflected by rain droplets at different severities are sparse (see Figure 1 in the Supplementary B) so that the slight effect of noise points on detection is covered by

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

	PVRCNN	PointRCNN	PartA2	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SESSD	Centerpoint	PVRCNN++	Centerpoint RCNN	Centerformer
Car	86.77	82.82	85.36	83.67	87.32	81.04	86.39	86.44	82.14	85.13	86.07	79.80
Pedestrian	60.61	52.34	59.68	52.15	-	-	-	-	49.32	58.32	55.93	52.78
Cyclist	76.42	77.60	80.09	68.51	-	-	-	-	68.58	71.85	75.43	67.19

TABLE V: $CE_{AP}(\%)$ 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	PVRCNN++	Centerpoint RCNN	PartA2	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Centerpoint	Centerformer	
Scene-level	**Weather**	*rain*	25.11	26.05	25.15	23.31	24.44	21.81	31.07	28.17	26.77	29.51	25.83	26.65	26.16
		snow	44.23	46.24	41.67	37.74	43.10	34.84	54.07	54.10	52.18	49.19	38.74	41.62	44.81
		fog	1.59	2.37	1.58	3.52	1.64	1.60	1.81	1.77	2.02	1.59	1.11	2.07	1.89
	Noise	*uniform_rad*	10.19	6.69	8.26	8.32	10.45	9.51	9.13	3.79	4.11	9.34	8.15	5.98	7.83
		gaussian_rad	13.02	8.74	10.98	9.98	12.49	12.13	10.83	4.84	5.18	11.02	10.17	6.66	9.67
		impulse_rad	2.20	2.00	2.08	3.86	1.36	2.23	2.50	2.25	3.57	1.18	1.86	1.20	2.19
		background	2.93	2.16	3.15	6.49	1.16	2.41	1.82	4.59	3.68	2.14	1.98	0.77	2.77
		upsample	0.81	0.96	0.60	1.84	0.72	0.31	0.95	0.37	0.71	0.55	0.46	0.71	0.75
	Density	*cutout*	3.75	3.77	3.58	3.97	3.66	4.27	3.99	4.51	3.59	4.26	4.11	4.25	3.98
		local_dec	14.04	14.24	14.09	-	14.34	13.88	14.55	14.44	12.50	17.04	14.64	15.91	14.52
		local_inc	1.40	1.72	1.53	3.34	1.34	1.33	2.20	1.66	1.69	0.90	0.95	0.81	1.57
		beam_del	0.58	0.91	0.45	0.79	0.80	0.73	0.88	0.80	0.53	1.07	0.47	0.82	0.74
		layer_del	2.94	3.05	2.87	3.46	2.80	3.10	3.39	3.29	3.16	3.37	2.67	3.50	3.13
Object-level	**Noise**	*uniform*	15.44	12.54	13.63	12.95	11.66	9.48	12.60	2.76	4.81	6.99	6.51	5.23	9.55
		gaussian	20.48	17.88	19.02	17.62	15.58	12.98	17.05	4.72	7.46	9.56	9.49	7.54	13.28
		impulse	3.30	4.06	3.16	4.70	2.80	2.53	4.07	2.88	4.29	2.20	2.11	1.97	3.17
		upsample	1.12	1.19	0.72	1.95	0.94	0.67	1.33	0.08	0.40	0.22	0.16	0.16	0.75
	Density	*cutout*	15.81	14.94	15.46	15.62	15.35	14.99	15.62	15.07	16.09	16.51	14.06	13.91	15.28
		local_dec	14.38	13.49	13.76	14.16	13.85	13.23	14.26	12.66	14.41	15.08	12.52	12.51	13.69
		local_inc	13.93	13.79	13.32	14.19	14.09	13.74	13.56	11.34	13.05	11.03	11.64	10.19	12.82
	Transformation	*shear*	37.27	39.42	38.73	40.96	39.96	40.35	41.37	39.52	37.85	40.35	40.00	35.38	39.26
		FFD	32.42	35.38	33.36	38.88	36.73	33.15	36.77	33.14	34.26	37.96	32.86	33.52	34.87
		rotation	0.60	0.10	0.48	0.47	0.57	0.31	0.97	0.39	0.75	0.27	0.38	0.56	0.49
		scale	5.78	5.64	5.31	8.13	4.98	6.96	5.81	8.53	6.50	6.53	7.50	8.57	6.69
		translation	3.82	3.34	4.13	3.03	3.02	3.24	4.58	4.88	5.34	1.37	3.91	2.93	3.63
mCE_{AP}			11.49	11.23	11.08	11.64	11.11	10.39	12.21	10.42	10.60	11.17	10.09	9.74	10.94

the randomness of severe influence of removing points with zero-value reflection, which makes the AP drop seemingly unaffected by the *rain* severity. Interestingly, as for "Pedestrian", only 10.01% of points have zero-value reflection intensity, causing a slight influence on detection. Hence, the effect of noise points by *rain* appears and shows a normal trend as the severity level increases (see Table S15 in the Supplementary).

C. Reacts of Detector Designing to Common Corruptions

How do different representations affect detectors? As shown in Figure 4, voxel-based Centerformer and BtcDet record the lowest and highest CE_{AP}. For most detectors (*i.e.*, except for PointRCNN), mCE_{AP} approximately increases as AP increases, indicating an potential trade-off between accuracy and robustness against common corruptions.

We also find that, as shown in Table V, for most Weather, Noise, and Density-related corruptions, voxel-based methods are generally more robust against corruption patterns. For a more fair comparison between different input representations, we conduct the ablative experiments on the robust Centerpoint. As shown in Table VI, under any given proposal structure, the voxel-based Centerpoint detectors perform more robustly against most Weather, Noise, and Density corruptions and overall corruptions by presenting a lower CE_{AP} or mCE_{AP} w.r.t. other detectors. 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-related corruptions.

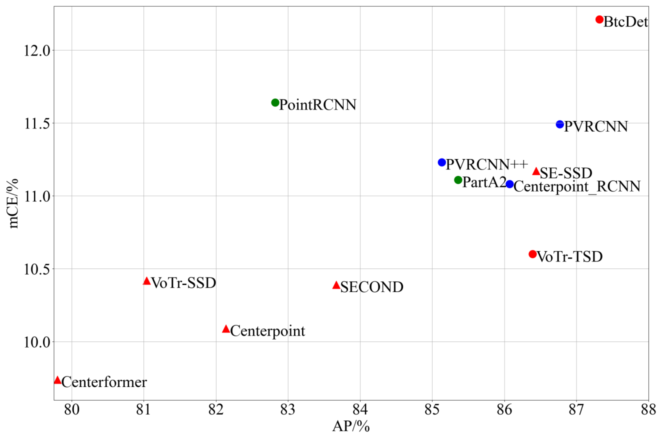


Fig. 4: mCE_{AP} of detectors with different representations on *Car* detection ({red, green, blue} for {voxel-based, point-based, voxel-point-based} detectors and {circle, triangle} for {two-stage, one-stage} ones)

TABLE VI: Ablation study on Centerpoint detectors with different data representations and proposal structures

Corruption			one-stage		two-stage		
			voxel	point	point-voxel	voxel	point
Scene-level	Weather	*rain*	25.83	35.48	25.15	23.61	31.37
		snow	38.74	46.68	41.67	37.31	49.51
		fog	1.11	3.53	1.58	1.36	3.54
	Noise	*uniform_rad*	8.15	8.18	8.26	7.85	7.53
		gaussian_rad	10.17	9.94	10.98	10.24	9.46
		impulse_rad	1.86	2.38	2.08	1.95	2.91
		background	1.98	4.92	3.15	3.17	4.63
		upsample	0.46	2.00	0.60	0.92	2.27
	Density	*cutout*	4.11	4.85	3.58	3.70	4.54
		local_dec	14.64	15.08	14.09	14.18	19.94
		local_inc	0.95	2.79	1.53	1.63	2.98
		beam_del	0.47	0.92	0.45	0.78	1.35
		layer_del	2.67	4.06	2.87	3.27	4.09
Object-level	Noise	*uniform*	6.51	7.20	13.63	11.25	8.55
		gaussian	9.49	10.72	19.02	16.10	12.31
		impulse	2.11	4.00	3.16	2.85	4.01
		upsample	0.16	1.25	0.72	1.04	1.78
	Density	*cutout*	14.06	16.01	15.46	15.58	16.51
		local_dec	12.52	14.17	13.76	14.02	15.01
		local_inc	11.64	14.02	13.32	12.64	14.13
	Transformation	*shear*	40.00	43.27	38.73	38.15	41.10
		FFD	32.86	34.55	33.36	33.55	35.66
		rotation	0.38	0.80	0.48	0.84	1.52
		scaling	7.50	9.17	5.31	5.50	6.89
		translation	3.91	7.12	4.13	4.20	5.84
mCE_{AP}			10.09	12.12	11.08	10.63	12.30

Specifically, for severe corruptions (*e.g.*, *shear, FFD* in the Transformation), the point-voxel-based methods are more robust. The point-based PointRCNN and PartA2 don't have performance superiority against most corruptions (except {*scale*}), suggesting potential limitations.

How do different proposal architectures affect detectors? As shown in Figure 4, two-stage detectors perform less robustly against overall corruptions compared to one-stage detectors, showing a higher mCE_{AP}. Also, as shown in Table VI, under a given representation, the one-stage detectors perform more robustly under overall corruptions, presenting 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).

Specially, as shown in Table S2 in the Supplementary A, one-stage detectors perform more robust against corruptions of scene-level and object-level Noise, object-level Density, and Transformation, while two-stage detectors are mainly more robust against scene-level Density. As for Weather corruptions, one-stage detectors present better robustness on {*snow, fog*} and two-stage detectors work better under corruptions of *rain*.

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 average CR_{FC} is only 0.26% (refer to Table S4 in Supplementary A). By contrast, the increase of false detection (FD) rate is relatively obvious, by the average CR_{FD} of 3.19% (refer to Table S5 in Supplementary A). Regarding missed detection (MD) (refer to Table S6 in Supplementary A), scene-level {*background*} and object-level {*cutout, local_dec*} result in an increase of MD rate of more than 4%.

Surprisingly, according to Table S8 in the Supplementary A, {*rain, snow*} and scene-level {*uniform_rad, gaussian_rad*} even reduce the rate of missing objects. One plausible explanation for this observation is that 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 S5 and S6 in Supplementary A). Also, we find that, as shown in Figure S1 in Supplementary A, 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 S7 in Supplementary A, most of the detectors perform relatively stable in terms of false classification rates and missed detection rates against corruptions. In contrast, affected by corruptions, all detectors have increasing false detection rates (see Table S7), revealing a serious bias in BBox localization.

E. Robustness Enhancement Evaluation

How do PA-DA and KNN-based outlier-removing affect detectors' robustness against different corruptions? Shown by Table S10 in Supplementary A, the average CE_{AP} with PA-DA slightly decreased to 10.75% compared to the average CE_{AP} without PA-DA, which still poses serious robustness issues for point cloud detectors. As shown in Tables VII and S10, PA-DA shows a better but still limited robustness improvement in the object-level Noise and Density. The existence of improvement is reasonable since PA-DA involves exactly object-level noise and density-related simulation during augmentation. However, since PA-DA only involves 5 limited basic augmentations at only one severity level, its robustness improvements on detectors, even under object-level noise and density corruptions, are limited. Regarding denoising strategy, the average CE_{AP} after adopting KNN-RO increases to 13.45% without PA-DA and 13.22% with PA-DA (refer to Table S10). These results indicate that KNN-RO might not be capable of enhancing point cloud detectors' robustness in *Car* detection. However, we find that KNN-RO slightly improves the robustness of *Pedestrian* detection by decreasing the CE_{AP} by 0.37% without PA-DA and 0.89% with PA-DA (Table S11 in Supplementary A). The robustness improvement on *Pedestrian* is reasonable due to KNN-RO's removing spatial outliers on the background and objects. But what causes the performance drop on *Car* detection? By investigating KITTI, we found, 15.32% of *Car* BBoxes have scanning points of less than 20, higher than 7.14% of *Pedestrian* BBoxes. Such relatively few points within a *Car* BBox (larger than *Pedestrian*) are distributed more sparsely and thus easier removed by KNN-RO, causing a missing of some parts of the car or even the whole car (refer to Figure S6 in the Supplementary). It significantly facilitates KNN-RO's damage on the point imaging of cars and further the detection accuracy on *Car* detection. Actually,

TABLE VII: $CE_{AP}(\%)$ of different enhancement methods on *Car* detection of {Centerpoint, PVRCNN, PointRCNN, SECOND} under different corruption categories (the green stands for the lowest CE_{AP} given a category, **Original** for no enhancement method, and {**WeatherAug**, **NoiseAug**, **DensityAug**, **TransAug**, **AllCorAug**} for augmentation by 4 and overall categories)

Enhancement Method	**Scene-level**			**Object-level**			**Overall**
	Weather	**Noise**	**Density**	**Noise**	**Density**	**Transformation**	
NonAug	21.62	5.44	4.23	7.59	14.02	17.00	10.93
PA-DA [47]	23.44	5.99	4.91	7.41	13.73	17.34	11.40
KNN-RO [55]	25.22	7.27	6.74	9.87	15.94	19.17	13.25
PA-DA&RO	27.01	7.84	7.43	9.75	15.75	19.56	13.76
Cutout [63]	21.05	5.36	3.69	8.45	12.60	16.68	10.60
CutMix [64]	21.89	5.55	4.32	9.00	13.58	17.38	11.21
3D-VFAug [50]	28.62	3.40	4.94	2.91	14.11	16.95	10.71
TT-BN [56]	25.54	7.26	7.28	9.65	16.22	19.77	13.42
WeatherAug	11.13	6.87	4.05	11.02	13.91	17.22	10.47
NoiseAug	24.34	2.33	4.02	2.00	13.51	16.47	9.50
DensityAug	20.52	4.52	3.19	5.07	10.85	15.62	9.33
TransAug	24.51	6.00	4.53	8.11	13.62	10.31	10.11
AllCorAug	15.41	2.65	3.78	2.23	11.89	13.63	7.69

after investigation, 72.14% of objects affected by KNN-RO are cars while only 0.94% of them are pedestrians, which verifies much more serious effects of KNN-RO on cars.

How do PA-DA and KNN-based outlier-removing affect different detectors' robustness against corruption? Except for PVRCNN, SE-SSD, Centerpoint, and Centerformer, other detectors perform more robustly against corruption patterns after adopting PA-DA (refer to Figure S2 in Supplementary A). Moreover, PointRCNN and VoTr-SSD increase their AP by 1.16% and 2.46% after adopting PA-DA, respectively. According to Figure S2, 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 0.37% without PA-DA and 0.89% with PA-DA on *Pedestrian* detection, respectively (refer to Table S9 in Supplementary A). It illustrates again that compared with on *Car*, KNN-RO is more effective in removing perturbations caused by corruptions on *Pedestrian*.

How does the category-oriented data augmentation against different corruption categories? As shown in Table VII, as augmentation methods relevant to the object-level Density, Cutout and CutMix present the limited AP increases of 1.42% and 0.44% on the object-level Density, respectively. It is mainly because Cutout involves one limited corruption and CutMix only augments data at the fixed severity level. Surprisingly, compared with the AP increase of 0.05% on Transformation, shape-deformation-related 3D-VField presents a significant AP increase of 4.68% on object-level Noise, revealing a high relevance of its adversarial deformation to Noise-related corruptions (refer to Figure S7 in Supplementary). Compared with original detectors, detectors with TT-BN perform worse under all corruption categories. One plausible explanation is, due to the large-scale input samples under the autonomous driving scenarios, the relatively small batch size (less than 64 samples per batch) causes unstable statistics (*i.e.*, the mean and standard deviation) for BN layers and thus cause unstable and biased detection.

As shown in Table VII, we found that the augmentation by all corruptions reaches the lowest CE_{AP} of 7.69% (*i.e.*, an AP increase of 3.24% than Original) under overall corruptions. For each corruption category, the detectors augmented by the corresponding basic corruption category show a significant improvement (*i.e.*, an obvious CE_{AP} drop) compared with the other robustness enhancement methods. By a simple ensemble of the best results under each category, the category-oriented data augmentation significantly decreases the overall CE_{AP} to 6.16%, revealing the great potential for corruption-oriented augmentations on robustness enhancement.

VI. Discussions

According to detailed findings in the empirical study of Section V, we summarize the following insights as guidelines for future robustness enhancement studies:

Insight 1. Common corruptions related to natural weather and shape transformation significantly challenge the point cloud detectors.

Insight 2. Regarding the input feature representation of detectors, the voxel-based detectors perform more robustly against common corruptions than point-based detectors, especially for most Weather, Noise, and Density corruptions.

Insight 3. Regarding the proposal structures of detectors, one-stage detectors comprehensively perform more robustly against overall corruptions than two-stage ones.

Insight 4. As for the detection results, corruptions more commonly cause a severe bias in BBox localization rather than erroneous object classification.

Insight 5. For the existing robustness enhancement methods, the explored augmentation methods only work on limited corruption categories; KNN-RO causes precision damage on *Car* detection; TT-BN easily degrades detection accuracy due to the limitation of batch size. To contrast, corruption-category-oriented augmentation shows great potential in robustness enhancement.

VII. 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 12 detectors covering 6 different detection frameworks and reveal the vulnerabilities of the detectors. Moreover, we further study the effectiveness of existing robustness enhancement methods of data augmentation, denoising, and test-time adaptation 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.

In the future, we plan to extend the corruption simulation on more large-scale datasets (*e.g.*, NuScenes [7], Waymo [8], and ONCE [19]) for a more extensive and comprehensive benchmark on robust point cloud detection. However, large-scale datasets with a wider range of locations and times of collection contain more LiDAR observations under corrupted conditions (*e.g.*, rain, snow, rare cars, and other corruptions) as shown in Table I. Thus, one challenge of extending into large-scale datasets is to erase the effects of original corruptions to make data more controllable to simulate corruptions individually. Also, we plan to extend our work into a robustness benchmark involving multi-modal point cloud detection (including *e.g.*, images or videos). One of the main challenges is to design the physical-aware corruption models consistent in both point clouds and images for the high-fidelity simulation of diverse common corruptions.

Acknowledgement

This work was supported in part by Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JST-Mirai Program Grant No.JPMJMI20B8, JSPS KAKENHI Grant No.JP20H04168, No.JP21H04877. This research was also partly supported by the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008).

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, "nuscenes: 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. Dotiwalla, 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] J. Mao, M. Niu, C. Jiang, X. Liang, Y. Li, C. Ye, W. Zhang, Z. Li, J. Yu, C. Xu *et al.*, "One million scenes for autonomous driving: Once dataset," 2021.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] T. Instruments, "Lidar pulsed time of flight reference design," 2016.

[25] 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.

[26] 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.

[27] 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.

[28] X. Morin-Duchesne and M. S. Langer, "Simulated lidar repositioning: a novel point cloud data augmentation method," *arXiv preprint arXiv:2111.10650*, 2021.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[36] 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.

[37] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[38] 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.

[39] S. Shi, X. Wang, and H. Li, "Pointrcnn: 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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, 2021, pp. 3391–3397.

[48] 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.

[49] 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.

[50] A. Lehner, S. Gasperini, A. Marcos-Ramiro, M. Schmidt, M.-A. N. Mahani, N. Navab, B. Busam, and F. Tombari, "3d-vfield: Adversarial augmentation of point clouds for domain generalization in 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 295–17 304.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 539–11 551, 2020.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[64] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[65] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 295–305.

[66] 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.

[67] 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.

[68] B. Chen and A. Kaufman, "3d volume rotation using shear transformations," *Graphical Models*, vol. 62, no. 4, pp. 308–322, 2000.

[69] 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.

[70] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2647–2664, 2020.

[71] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection," *International Journal of Computer Vision*, vol. 131, no. 2, pp. 531–551, 2023.

[72] Z. Zhou, X. Zhao, Y. Wang, P. Wang, and H. Foroosh, "Centerformer: Center-based transformer for 3d object detection," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*. Springer, 2022, pp. 496–513.

[73] 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.

[74] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

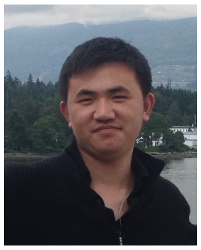
[75] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *arXiv preprint arXiv:1908.09492*, 2019.

[76] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

Shuangzhi Li is currently a Ph.D. student in Software Engineering & Intelligent System at the University of Alberta, AB, Canada. He received a B.S. degree in electronic science and technology in 2018 and a M.S. degree in electronic science and technology in 2021 from the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include the robustness of AI systems and 3D computer vision.

Zhijie Wang is currently a Ph.D. student in Software Engineering at the University of Alberta, AB, Canada. He is also affiliated with Alberta Machine Intelligence Institute (Amii). Prior to that, he received his M.E. degree in Electrical and Computer Engineering from the University of Waterloo, ON, Canada, and a B.E. degree in Telecommunication Engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China. His research interests include quality assurance of AI systems and human-computer interaction in AI software development.

Felix Juefei-Xu received the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Prior to that, he received the M.S. degree in Electrical and Computer Engineering and the M.S. degree in Machine Learning from CMU, and the B.S. degree in Electronic Engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China. Currently, he is a Research Scientist at Meta AI, New York, NY, USA, with research focus on robust and efficient machine perception and learning problems as well as multimodal generative AI. He is also affiliated with New York University (NYU) as an Adjunct Professor. He has broader interests in pattern recognition, computer vision, machine learning, optimization, statistics, compressive sensing, and image processing. He is the recipient of multiple best/distinguished paper awards, including IJCB'11, BTAS'15-16, ASE'18, and ACCV'18.



Qing Guo received his Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He is currently a senior research scientist and principal investigator (PI) at the Center for Frontier AI Research (CFAR), A*STAR in Singapore. He is also an adjunct assistant professor at the National University of Singapore (NUS), and senior PC member of AAAI. Prior to that, he was a Wallenberg-NTU Presidential Postdoctoral Fellow with the Nanyang Technological University, Singapore. His research interests include computer vision, AI security, and image processing. He is a member of IEEE.

Xingyu Li is an assistant professor with the Department of Electrical and Computer Engineering at the University of Alberta and an Alberta Machine Intelligence Institute (Amii) Fellow. She received B.Sc from Peking University, M.Sc from the University of Alberta, and Ph.D from the University of Toronto. Her research interests include machine learning, computer vision, AI for health, and multi-dimensional data analytics and understanding.

Lei Ma is currently an associate professor with The University of Tokyo, Japan and University of Alberta, Canada. He was honorably selected as a Canada CIFAR AI Chair and Fellow at Alberta Machine Intelligence Institute (Amii). Previously, he received the B.E. degree from Shanghai Jiao Tong University, China, and the M.E. and Ph.D. degrees from The University of Tokyo, Japan. His recent research centers around the interdisciplinary fields of software engineering (SE) and trustworthy artificial intelligence with a special focus on the quality, reliability, safety and security aspects of AI Systems.

Appendix A
Empirical Study

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

A. Effects of Common Corruptions to Point Cloud Detectors

TABLE S1: $CE_{AP}(\%)$ 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	*rain*	26.76	26.40	25.33	25.22	27.07	26.16
		snow	26.62	30.38	44.89	56.14	66.01	44.81
		fog	0.16	0.60	1.24	2.59	4.84	1.89
	Noise	*uniform_rad*	0.46	2.31	6.26	11.83	18.27	7.83
		gaussian_rad	1.49	4.25	8.84	13.90	19.87	9.67
		impulse_rad	0.96	1.31	1.74	2.47	4.46	2.19
		background	1.80	2.06	2.50	2.78	4.72	2.77
		upsample	0.33	0.34	0.53	0.76	1.78	0.75
	Density	*cutout*	1.65	2.37	3.72	4.98	7.16	3.98
		local_dec	5.24	6.75	9.43	15.08	36.08	14.52
		local_inc	0.81	0.99	1.38	1.86	2.81	1.57
		beam_del	0.08	0.17	0.44	0.89	2.09	0.73
		layer_del	0.44	1.97	2.93	4.34	5.97	3.13
Object-level	Noise	*uniform*	0.65	1.98	5.99	13.36	25.76	9.55
		gaussian	1.51	4.28	9.59	19.00	32.04	13.28
		impulse	1.90	2.35	2.93	3.48	5.21	3.17
		upsample	0.38	0.53	0.57	0.80	1.45	0.75
	Density	*cutout*	5.85	11.28	15.97	20.02	23.31	15.29
		local_dec	1.89	10.18	14.87	18.98	22.55	13.69
		local_inc	7.99	11.99	13.78	14.93	15.42	12.82
	Transformation	*shear*	3.89	15.13	37.03	63.05	77.20	39.26
		FFD	2.27	14.05	35.25	55.47	67.31	34.87
		rotation	0.05	0.13	0.26	0.81	1.19	0.49
		scaling	0.47	2.15	5.02	9.28	16.52	6.69
		translation	1.04	3.88	4.84	4.07	4.34	3.63
Average			3.79	6.31	10.21	14.64	19.74	10.94

Table S1 shows the CE_{AP} under different severity levels of corruptions on *Car* detection. According to Table S1, except scene-level $\{rain\}$, all corruptions follow a predictable trend, *i.e.*, the CE_{AP} increases as the severity level increases. Table S3 shows the recall loss (*i.e.*, CE_{recall}) of different detectors under different corruptions on *Car* detection.

B. Reacts of Detector Designing to Common Corruptions

Table S2 depicts the average precision (AP) loss (*i.e.*, CE_{AP}) of one-stage and two-stage detectors on *Car* detection under different common corruptions. Table S2 indicates that one-stage detectors perform more robust against corruptions of scene-level and object-level Noise, object-level Density, and Transformation, while two-stage detectors are mainly more robust against scene-level Density.

TABLE S2: $CE_{AP}(\%)$ of detectors with different proposal architectures on *Car* detection(green cell for each corruption)

Corruption			one-stage	two-stage
Scene-level	Weather	*rain*	26.39	25.99
		snow	43.70	45.60
		fog	1.63	2.07
	Noise	*uniform_rad*	7.35	8.16
		gaussian_rad	8.96	10.17
		impulse_rad	1.74	2.51
		background	2.38	3.06
		upsample	0.48	0.94
	Density	*cutout*	4.28	3.76
		local_dec	15.18	13.96
		local_inc	1.13	1.89
		beam_del	0.78	0.70
		layer_del	3.19	3.09
Object-level	Noise	*uniform*	6.19	11.95
		gaussian	8.86	16.44
		impulse	2.34	3.77
		upsample	0.26	1.09
	Density	*cutout*	14.91	15.55
		local_dec	13.20	14.04
		local_inc	11.59	13.71
	Transformation	*shear*	39.12	39.37
		FFD	34.13	35.40
		rotation	0.38	0.56
		scale	7.62	6.02
		translation	3.27	3.89
mCE_{AP}			10.36	11.34

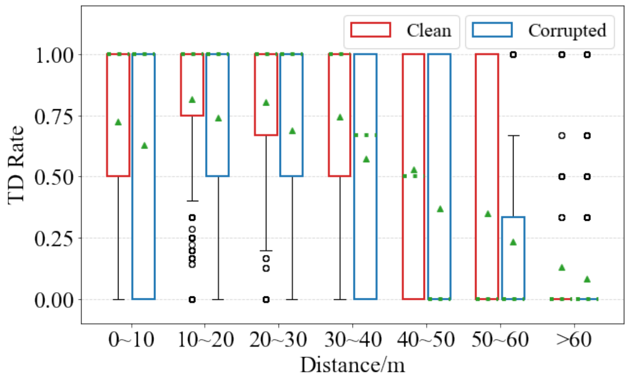


Fig. S1: Box-plot of TD rate w.r.t. different distances of objects to the LiDAR sensor (green dotted lines for the median and green triangles for the mean)

C. Detection Bugs in Detectors under Common Corruptions

Table S7 depicts the bug rates of different detectors on *Car* detection. According to Table S7, we find there is always an increase of FC and FD rate for all detectors. Table S8 depicts the bug rates of *Car* detection under different corruptions. According to Table S8, $\{$*rain, snow*$\}$ and scene-level $\{$*uniform_rad, gaussian_rad*$\}$ even reduce the rate of missing objects, which has been analyzed in Section V-D. More details of the increase of bug rates (*i.e.*, CR) are recorded in Tables S4, S5, and S6.

Figure S1 shows the TD Rate of detections at different distances of objects to the LiDAR sensor. As shown in Figure S1, compared to clean observations, TD rates under corrupted observations are always lower at any distance of objects to LiDAR.

D. Robustness Enhancement by Data Augmentation and Denoising

Table S9 and Figure S2 show CE_{AP} of different detectors on *Pedestrian* detection and *Car* detection, respectively, with

TABLE S3: $CE_{recall}(\%)$ of different detectors under different corruptions on *Car* detection (the green cell with CE of over 18%)

Corruption			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Center-point	PVRCNN ++	Centerpoint RCNN	Center-former	PartA2	Average
Scene-level	**Weather**	*rain*	24.50	23.67	20.92	29.69	23.46	24.98	27.10	24.84	27.09	25.25	24.50	24.43	🟩 25.04
		snow	36.23	32.72	29.19	43.32	37.11	40.26	38.32	32.80	39.20	35.61	34.65	36.04	🟩 36.29
		fog	4.22	5.30	3.14	2.36	2.45	2.43	2.28	3.47	6.99	4.82	4.44	4.68	3.88
	Noise	*uniform_rad*	14.23	12.91	13.67	9.20	3.70	5.12	9.33	11.31	12.71	11.85	7.30	13.86	10.43
		gaussian_rad	17.74	15.67	17.46	11.30	4.81	6.39	11.29	14.35	16.33	15.41	8.45	17.37	13.05
		impulse_rad	4.17	4.47	4.32	3.93	3.86	5.02	2.11	4.05	4.36	4.08	1.78	3.26	3.78
		background	3.26	8.66	2.84	2.21	4.64	4.73	2.59	2.42	2.61	3.58	0.83	1.84	3.35
		upsample	0.94	2.58	1.20	0.93	0.94	0.68	0.80	0.76	1.60	0.68	0.42	1.02	1.05
	Density	*cutout*	4.87	4.61	5.55	4.05	4.37	3.55	4.45	5.55	5.36	5.15	5.50	5.33	4.86
		local_dec	15.56	-	14.76	13.04	11.78	11.10	14.06	16.19	16.29	15.99	17.09	16.12	14.73
		local_inc	1.58	2.62	1.59	1.66	1.70	1.48	1.21	1.56	1.89	1.44	0.65	1.38	1.56
		beam_del	0.92	0.77	1.13	0.95	0.94	0.63	1.34	1.11	1.21	0.89	1.20	1.21	1.03
		layer_del	3.48	3.37	3.59	3.12	3.13	2.74	3.37	3.78	3.93	3.67	4.31	3.69	3.51
Object-level	**Noise**	*uniform*	9.60	10.19	6.74	9.22	2.65	3.67	7.05	5.82	9.33	8.65	4.07	8.88	7.16
		gaussian	12.69	13.02	9.16	12.05	3.84	5.18	9.06	7.78	12.76	11.92	5.20	11.42	9.51
		impulse	2.11	3.14	1.88	2.03	1.99	1.94	1.95	1.86	2.43	2.15	1.23	1.89	2.05
		upsample	0.77	1.71	0.96	1.00	0.44	0.26	0.44	0.57	1.17	0.74	-0.18	0.98	0.74
	Density	*cutout*	22.21	20.90	21.61	17.61	16.27	17.44	17.67	21.21	22.04	22.18	22.87	22.44	🟩 20.37
		local_dec	19.99	18.73	19.04	15.97	14.11	15.52	16.26	18.96	19.78	19.84	21.08	20.13	🟩 18.28
		local_inc	10.58	10.95	10.78	9.30	7.63	8.11	7.89	9.33	11.55	10.56	6.71	11.33	9.56
	Transformation	*shear*	22.13	25.19	25.06	23.26	22.14	20.61	21.70	23.95	23.56	22.61	18.76	24.66	🟩 22.80
		FFD	17.53	21.51	18.41	19.43	16.74	17.26	18.70	18.43	19.17	18.60	17.31	20.55	🟩 18.64
		rotation	0.49	0.42	0.4	0.63	0.4	0.42	0.53	0.26	0.46	0.40	0.63	0.33	0.45
		scaling	5.1	5.95	5.95	4.56	6.09	4.77	4.82	5.86	5.22	4.80	6.27	5.10	5.37
		translation	3.79	3.42	3.78	4.69	5.32	4.66	2.39	4.26	4.07	4.05	2.90	3.32	3.89
mCE_{recall}			10.35	10.52	9.73	9.82	8.02	8.36	9.07	9.62	10.84	10.20	8.72	10.45	9.66

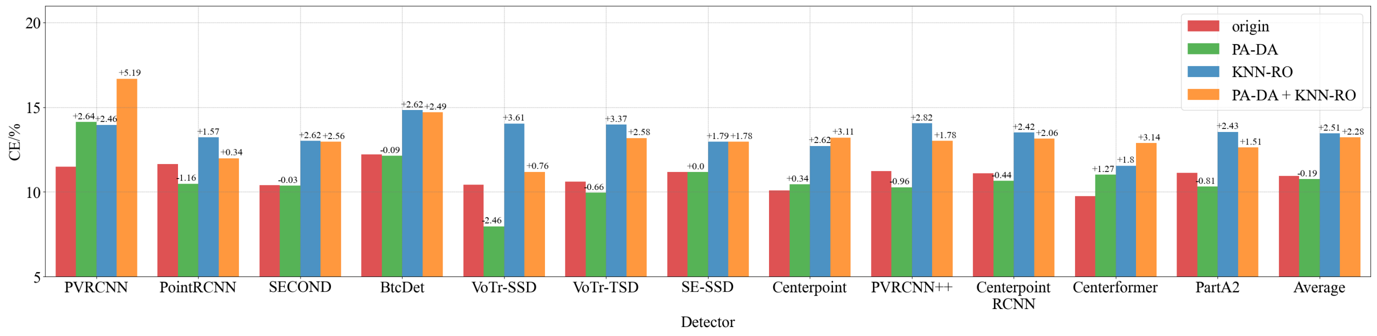


Fig. S2: Average $CE_{AP}(\%)$ of different detectors on *Car* detection given common corruptions

PA-DA and/or KNN-RO. According to Table S9, adopting KNN-RO slightly improves the AP by 0.37% without PA-DA and 0.89% with PA-DA on *Pedestrian* detection, respectively.

Tables S10 depicts CE_{AP} of *Car* detection with PA-DA and/or KNN-RO under different corruptions. As shown in Table S10, compared to the original detection, there is an average increase of AP (*i.e.*, a decrease of CE_{AP}) on *Car* detection with only PA-DA. By contrast, compared to the original detection, there is an average decrease of AP (increase of CE_{AP}) on *Car* detection with KNN-RO and with PA-DA+KNN-RO. In summary, KNN-RO makes the performance of point cloud detection worse, namely less robust against corruptions.

Tables S11 depicts CE_{AP} of *Pedestrian* detection with PA-DA and/or KNN-RO under different corruptions. As shown in Table S11, compared to the original detection, there is a slight average increase of AP (*i.e.*, a decrease of CE_{AP}) on *Pedestrian* detection with PA-DA, with KNN-RO, and with PA-DA+KNN-RO.

Appendix B
Corruption Simulation Naturalness Validation

A. 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 [15] and foggy [10] weather.

Training and Testing: As for snow/fog, we collect real corrupted and clean data as training set D_{train} (including 50% corrupted data and 50% clean data). We train the PointNet model with D_{train} to learn the model to classify the clean or corrupted condition of point clouds. Then we gather the clean and simulated corrupted KITTI data as testing set D_{test} and obtain the predictions of trained classifier on D_{test}.

Experiment Setting: We set the batch size to 32 for the model training on the NVIDIA RTX A6000 GPU. We choose PointNet as the classifier due to its efficiency and effectiveness in recognizing the global feature [59], which fits into the global effects of weather corruptions on LiDAR scanning.

TABLE S4: $CR_{FC}(\%)$ of detectors on *Car* detection under common corruptions

Corruption			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Center-point	PVRCNN ++	Centerpoint RCNN	Center-former	PartA2	Average
Scene-level	**Weather**	*rain*	1.90	1.27	0.61	-0.01	-0.09	0.06	-0.01	2.29	0.85	2.40	0.33	1.98	0.96
		snow	2.17	2.01	0.71	0.03	0.17	0.12	-0.01	2.59	1.18	2.95	0.40	2.14	1.21
		fog	0.03	0.02	0.10	0.02	0.21	0.16	0.04	0.16	-0.10	0.03	0.52	0.17	0.11
	Noise	*uniform_rad*	1.35	0.82	1.27	0.00	-0.06	0.00	-0.01	1.29	1.08	1.95	0.21	1.63	0.79
		gaussian_rad	1.83	1.25	1.67	0.00	-0.06	0.01	0.00	1.62	1.32	2.58	0.30	2.09	1.05
		impulse_rad	0.12	0.01	0.14	0.02	0.07	0.06	0.04	0.25	0.03	0.34	0.13	0.20	0.12
		background	-0.18	0.20	-0.19	0.01	-0.22	-0.01	0.04	-0.20	-0.25	-0.17	-0.13	-0.24	-0.11
		upsample	-0.01	0.03	0.03	0.01	-0.15	-0.02	0.01	-0.01	-0.03	0.08	-0.05	0.04	-0.01
	Density	*cutout*	0.21	0.13	0.16	0.03	0.18	0.15	0.04	0.12	0.14	0.19	0.20	0.27	0.15
		local_dec	0.70	-	0.47	0.05	0.37	0.29	0.03	0.70	0.46	0.89	0.27	0.73	0.45
		local_inc	0.04	0.02	0.05	0.01	0.00	0.05	0.05	0.04	0.01	0.05	0.05	0.03	0.03
		beam_del	0.08	0.04	0.04	0.01	0.08	0.05	0.01	0.05	-0.01	0.07	0.02	0.08	0.04
		layer_del	0.13	0.07	0.09	0.03	0.17	0.17	0.04	0.09	0.08	0.12	0.20	0.13	0.11
Object-level	**Noise**	*uniform*	0.53	0.24	0.39	0.01	-0.01	0.00	0.00	0.32	0.02	0.72	0.04	0.65	0.24
		gaussian	0.71	0.32	0.50	0.01	-0.01	0.00	0.01	0.41	0.06	0.87	0.05	0.77	0.31
		impulse	0.03	0.00	0.04	-0.01	0.00	0.02	0.00	0.00	-0.04	0.06	-0.01	0.06	0.01
		upsample	0.06	0.00	0.12	0.00	0.01	0.02	0.03	0.04	-0.02	0.14	0.11	0.17	0.06
	Density	*cutout*	0.38	0.03	0.51	0.08	0.49	0.41	0.04	0.40	0.29	0.32	0.41	0.57	0.33
		local_dec	0.26	-0.03	0.40	0.07	0.37	0.31	0.04	0.32	0.19	0.24	0.32	0.44	0.24
		local_inc	0.23	0.12	0.29	0.00	-0.13	-0.02	0.01	0.17	0.05	0.21	0.18	0.30	0.12
	Transformation	*shear*	0.16	-0.01	0.25	0.01	0.17	0.06	0.01	0.16	0.05	0.18	0.09	0.21	0.11
		FFD	0.08	0.02	0.18	0.01	0.08	0.06	0.03	0.11	-0.03	0.16	0.03	0.14	0.07
		rotation	-0.01	0.02	0.01	0.01	0.00	0.01	0.01	0.00	0.00	-0.03	0.00	0.03	0.00
		scaling	0.02	0.00	0.03	0.01	0.02	0.01	0.03	0.00	-0.04	-0.01	0.02	0.04	0.01
		translation	0.14	0.09	0.14	0.03	0.10	0.09	0.05	0.15	0.11	0.21	0.08	0.22	0.12
mCR_{FC}			0.44	0.28	0.32	0.02	0.07	0.08	0.02	0.44	0.22	0.58	0.15	0.51	0.26

TABLE S5: $CR_{FD}(\%)$ of detectors on *Car* detection under common corruptions

Corruption			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Center-point	PVRCNN ++	Centerpoint RCNN	Center-former	PartA2	Average
Scene-level	**Weather**	*rain*	8.25	4.23	5.70	7.90	13.39	11.46	4.78	5.24	15.44	8.68	23.68	7.59	9.69
		snow	8.54	7.45	5.93	14.51	12.90	16.34	10.11	5.39	15.24	9.48	20.33	7.75	11.16
		fog	0.82	0.11	0.62	-0.30	-0.84	-0.49	0.10	0.71	1.55	0.66	-0.62	0.06	0.20
	Noise	*uniform_rad*	4.08	5.94	3.30	3.81	2.61	4.01	5.52	2.50	4.71	3.29	5.20	2.91	3.99
		gaussian_rad	4.88	6.83	3.87	4.31	3.34	4.94	6.44	3.02	5.52	4.15	6.42	3.46	4.76
		impulse_rad	1.54	2.30	0.99	1.81	2.55	3.72	1.22	0.92	3.17	1.56	0.91	0.86	1.80
		background	-0.34	2.25	-2.25	0.34	-3.22	-0.14	0.78	-2.14	-4.81	0.15	-6.51	-0.88	-1.40
		upsample	-0.04	0.82	-0.34	0.34	0.29	0.14	0.69	-0.44	0.22	-0.05	-3.33	-0.14	-0.15
	Density	*cutout*	0.73	0.57	0.58	1.11	1.05	1.17	1.87	0.41	2.21	0.92	2.97	0.99	1.21
		local_dec	3.79	-	2.77	3.40	3.44	4.29	4.43	2.83	9.31	4.21	9.58	3.76	4.71
		local_inc	0.39	0.95	0.23	0.43	0.53	0.55	0.48	0.30	0.20	0.43	-1.51	0.28	0.27
		beam_del	0.39	0.06	0.39	0.54	0.52	0.48	0.75	0.30	1.47	0.38	2.44	0.41	0.68
		layer_del	0.98	0.54	0.79	1.25	0.98	1.01	1.36	0.73	2.47	1.09	2.56	0.93	1.22
Object-level	**Noise**	*uniform*	2.30	4.51	1.25	2.85	1.37	2.13	3.01	1.10	2.46	2.22	2.14	2.00	2.28
		gaussian	3.11	6.00	1.70	3.99	1.98	3.11	3.91	1.47	3.49	3.20	2.70	2.59	3.10
		impulse	0.89	1.47	0.53	1.40	1.06	1.32	1.43	0.53	1.14	0.98	0.75	0.66	1.01
		upsample	0.28	0.79	0.22	0.33	0.24	0.13	0.95	0.16	0.09	0.25	-0.03	0.32	0.31
	Density	*cutout*	2.29	-0.92	1.64	0.39	2.75	3.27	0.54	0.77	2.97	1.58	-1.59	2.32	1.33
		local_dec	1.83	-1.15	1.27	0.10	2.02	2.57	-0.13	0.48	2.00	1.13	-2.18	1.86	0.82
		local_inc	3.54	5.44	2.54	4.81	3.43	4.74	6.77	2.31	3.58	3.76	3.41	2.69	3.92
	Transformation	*shear*	10.08	17.48	7.56	23.27	11.70	14.84	24.29	7.74	13.74	11.07	6.24	8.98	13.08
		FFD	8.06	14.85	5.59	18.56	8.82	12.28	20.52	5.95	10.96	9.15	5.68	7.53	10.66
		rotation	0.22	0.13	0.12	0.40	0.25	0.29	0.41	0.07	0.23	0.26	0.55	0.15	0.26
		scaling	2.31	3.82	1.81	4.40	3.24	3.38	5.37	1.90	2.76	2.38	1.54	1.83	2.89
		translation	1.58	1.09	0.87	1.55	2.78	3.24	1.09	0.88	3.36	1.61	3.78	1.31	1.93
mCR_{FD}			2.82	3.56	1.91	4.06	3.09	3.95	4.27	1.73	4.14	2.90	3.40	2.41	3.19

TABLE S6: $CR_{MD}(\%)$ of detectors on *Car* detection under common corruptions

Corruption			PVRCNN	PointRCNN	SECOND	BtcDet	VoTr-SSD	VoTr-TSD	SE-SSD	Center-point	PVRCNN ++	Centerpoint RCNN	Center-former	PartA2	Average
Scene-level	Weather	*rain*	-9.33	-4.63	-3.42	-12.61	-8.37	-5.64	-11.01	-4.89	-13.59	-10.73	-22.80	-6.72	-9.48
		snow	-0.71	-3.47	0.62	-8.96	4.14	-0.61	-9.99	-0.40	-5.82	-4.01	-17.24	0.02	-3.87
		fog	-0.60	-0.43	0.25	-0.54	6.77	6.53	2.70	-1.59	-2.28	-0.30	1.29	2.49	1.19
	Noise	*uniform_rad*	-2.10	-5.78	-2.98	-4.51	-2.79	-3.62	-4.63	-2.45	-3.03	-1.76	-3.84	-1.34	-3.24
		gaussian_rad	-2.32	-6.42	-3.03	-5.00	-3.37	-4.28	-5.17	-2.68	-3.03	-1.99	-5.03	-1.38	-3.64
		impulse_rad	0.26	-5.51	0.85	-2.23	-2.45	-2.89	-1.51	1.24	-1.19	1.35	0.06	0.18	-0.99
		background	7.43	-7.01	13.13	-1.12	19.16	10.67	5.10	12.48	17.00	5.64	9.92	8.24	8.39
		upsample	2.07	-0.73	2.88	-0.62	0.79	1.20	1.07	2.80	0.26	1.42	4.88	1.94	1.50
	Density	*cutout*	1.74	1.83	1.59	-1.01	1.77	0.53	-1.03	2.26	0.44	1.43	-2.41	0.74	0.66
		local_dec	-0.08	-	-0.09	-3.24	2.18	-0.38	-3.44	0.34	-5.45	-1.29	-8.21	0.30	-1.76
		local_inc	0.39	-1.11	0.59	-0.37	1.05	1.16	0.81	0.29	1.19	0.29	2.61	0.40	0.61
		beam_del	-0.37	0.14	-0.48	-1.02	-0.37	-0.92	-1.27	-0.22	-1.50	-0.44	-2.50	-0.07	-0.75
		layer_del	0.13	1.32	-0.27	-1.00	1.02	0.55	-0.99	0.20	-1.92	-0.06	-2.33	0.06	-0.27
Object-level	Noise	*uniform*	0.90	-0.93	0.32	0.86	-0.15	-0.31	0.06	0.50	1.15	0.74	-1.03	0.27	0.20
		gaussian	1.23	-1.00	0.49	1.33	-0.11	-0.32	0.27	0.69	1.56	1.04	-1.26	0.41	0.36
		impulse	0.13	-0.74	0.10	0.06	0.01	-0.16	0.15	0.22	0.06	0.05	-0.29	0.00	-0.03
		upsample	-0.02	-1.15	0.01	-0.22	-0.07	-0.11	-0.09	0.05	0.20	-0.14	0.12	-0.17	-0.13
	Density	*cutout*	5.05	10.70	3.62	5.00	3.76	4.81	5.15	4.74	5.04	6.06	4.10	3.96	5.17
		local_dec	4.76	9.90	3.36	4.70	3.57	4.52	4.89	4.43	4.97	5.65	4.50	3.80	4.92
		local_inc	0.52	0.05	0.30	0.86	0.24	0.23	0.85	0.45	1.08	0.47	-2.30	0.67	0.28
	Transformation	*shear*	0.01	-0.60	0.05	0.40	-0.09	-0.18	0.05	0.22	-0.92	-0.11	-3.60	-0.24	-0.42
		FFD	0.08	-1.11	0.04	0.72	0.02	0.05	1.08	0.24	-0.43	-0.02	-3.16	-0.18	-0.22
		rotation	0.03	-0.28	0.00	0.01	-0.06	-0.05	-0.02	0.03	0.03	-0.06	-0.36	-0.06	-0.07
		scaling	0.08	0.08	0.01	0.10	-0.03	-0.04	0.31	0.09	0.06	-0.01	-0.69	-0.02	-0.01
		translation	0.29	-1.29	0.45	0.27	-0.07	-0.23	-0.02	0.60	-0.70	0.24	-2.86	-0.10	-0.29
mCR_{FD}			0.38	-0.76	0.74	-1.13	1.06	0.42	-0.67	0.79	-0.27	0.14	-2.10	0.53	-0.08

TABLE S7: Bug rates (%) of true detection, false classification, false detection, and missing detection of different detectors (results on *clean* data 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)
PVRCNN++	36.71 (40.8)	0.93 (0.71)	22.8 (18.66)	39.57 (39.84)
Centerpoint_RCNN	34.65 (38.27)	1.36 (0.78)	12.16 (9.26)	51.83 (51.69)
Centerformer	7.88 (9.34)	0.67 (0.52)	31.03 (27.63)	60.41 (62.51)
PartA2	24.75 (28.19)	1.17 (0.66)	10.02 (7.61)	64.06 (63.54)

TABLE S8: Bug rates (%) of true detection, false classification, false detection, and missing detection of detectors under different corruptions

Corruption			TD	FC	FD	MD
Clean			38.92	0.47	11.81	48.80
Scene-level	Weather	*rain*	37.74	1.43	21.50	39.32
		snow	30.42	1.67	22.97	44.93
		fog	37.42	0.58	12.00	49.99
	Noise	*uniform_rad*	37.38	1.26	15.80	45.57
		gaussian_rad	36.75	1.52	16.57	45.16
		impulse_rad	38.00	0.59	13.60	47.82
		background	32.05	0.36	10.41	57.19
		upsample	37.59	0.46	11.65	50.30
	Density	*cutout*	36.90	0.62	13.02	49.46
		local_dec	34.50	0.92	16.46	48.12
		local_inc	38.01	0.50	12.08	49.41
		beam_del	38.95	0.51	12.48	48.05
		layer_del	37.86	0.58	13.03	48.53
Object-level	Noise	*uniform*	36.20	0.71	14.09	49.00
		gaussian	35.15	0.78	14.91	49.16
		impulse	37.93	0.48	12.82	48.77
		upsample	38.69	0.52	12.12	48.67
	Density	*cutout*	32.10	0.80	13.14	53.97
		local_dec	32.94	0.71	12.62	53.72
		local_inc	34.60	0.59	15.73	49.09
	Transformation	*shear*	26.15	0.58	24.89	48.39
		FFD	28.41	0.54	22.47	48.58
		rotation	38.73	0.47	12.06	48.74
		scaling	36.02	0.48	14.70	48.80
		translation	37.17	0.58	13.73	48.51

Analysis: According to the snow classification in Table S12, 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.

We further analyze the similarity of distribution of real and simulated corrupted data. Specifically, we utilize T-SNE [76] to visualize extracted the high-level features from the trained classifier. As shown in Figures 2 and 3, the distributions of the real and simulated corruptions are significantly similar. As shown in Table S13, The maximum mean discrepancy (MMD) [67] results quantitatively verify that the simulated *snow/fog* is close to the real *snow/fog*, respectively, while not close to the clean data.

B. Rain Validation

In Figure 1, 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

TABLE S9: 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	8.43	3.73(-4.7)	7.05(-1.38)	3.57(-4.86)
SECOND	8.38	7.63(-0.75)	8.46(+0.08)	7.76(-0.62)
Centerpoint	7.19	5.37(-1.82)	7.08(-0.11)	5.21(-1.98)
PVRCNN++	9.67	10.93(+1.26)	9.41(-0.26)	10.92(+1.25)
Centerpoint_RCNN	9.22	6.19(-3.03)	8.78(-0.44)	5.6(-3.62)
Centerformer	6.78	7.58(+0.8)	7.05(+0.27)	8.3(+1.52)
PartA2	10.18	10.72(+0.54)	9.6(-0.58)	10.06(-0.12)
Average	8.57	7.84(-0.73)	8.2(-0.37)	7.68(-0.89)

TABLE S10: Average $CE_{AP}(\%)$ of detectors given different corruptions on *Car* detection with DA and/or denoising (differences between enhancement methods and **Origin** are in parentheses)

Corruption			Origin	PA-DA	KNN-RO	PA-DA+KNN-RO
Scene-level	**Weather**	*rain*	26.16	27.29(+1.13)	31.63(+5.47)	32.47(+6.31)
		snow	44.81	45.05(+0.24)	46.94(+2.13)	47.12(+2.31)
		fog	1.89	1.91(+0.02)	5.12(+3.23)	5.06(+3.17)
	Noise	*uniform_rad*	7.83	7.51(-0.32)	10.74(+2.91)	10.37(+2.54)
		gaussian_rad	9.67	9.12(-0.55)	12.62(+2.95)	11.98(+2.31)
		impulse_rad	2.19	1.82(-0.37)	4.75(+2.56)	4.28(+2.09)
		background	2.77	2.72(-0.05)	2.05(-0.72)	2.03(-0.74)
		upsample	0.75	0.59(-0.16)	3.42(+2.67)	3.22(+2.47)
	Density	*cutout*	3.98	3.89(-0.09)	6.57(+2.59)	6.41(+2.43)
		local_dec	14.52	14.76(+0.24)	16.7(+2.18)	16.92(+2.4)
		local_inc	1.57	1.38(-0.19)	4.23(+2.66)	4.03(+2.46)
		beam_del	0.74	0.79(+0.05)	3.36(+2.62)	3.33(+2.59)
		layer_del	3.13	3.24(+0.11)	6.09(+2.96)	6.18(+3.05)
Object-level	**Noise**	*uniform*	9.55	8.43(-1.12)	12.05(+2.5)	11.06(+1.51)
		gaussian	13.28	11.91(-1.37)	15.71(+2.43)	14.42(+1.14)
		impulse	3.17	3.19(+0.02)	5.97(+2.8)	5.93(+2.76)
		upsample	0.75	0.59(-0.16)	2.63(+1.88)	2.39(+1.64)
	Density	*cutout*	15.28	14.95(-0.33)	17.0(+1.72)	16.68(+1.4)
		local_dec	13.69	13.56(-0.13)	16.05(+2.36)	15.88(+2.19)
		local_inc	12.82	11.77(-1.05)	15.02(+2.2)	14.06(+1.24)
	Transformation	*shear*	39.26	39.35(+0.09)	40.14(+0.88)	40.11(+0.85)
		FFD	34.87	34.34(-0.53)	39.3(+4.43)	38.89(+4.02)
		rotation	0.49	0.52(+0.03)	3.05(+2.56)	3.05(+2.56)
		scaling	6.69	6.7(+0.01)	9.14(+2.45)	9.12(+2.43)
		translation	3.63	3.34(-0.29)	6.02(+2.39)	5.58(+1.95)
Average			10.94	10.75(-0.19)	13.45(+2.51)	13.22(+2.28)

TABLE S11: Average $CE_{AP}(\%)$ of detectors given different corruptions on *Pedestrian* detection of 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**	*rain*	3.72	2.82(-0.9)	2.97(-0.75)	2.29(-1.43)
		snow	3.38	3.56(+0.18)	2.58(-0.8)	2.79(-0.59)
		fog	10.42	7.79(-2.63)	10.19(-0.23)	7.35(-3.07)
	Noise	*uniform_rad*	29.85	25.22(-4.63)	29.59(-0.26)	25.2(-4.65)
		gaussian_rad	36.50	31.91(-4.59)	36.37(-0.13)	32.0(-4.5)
		impulse_rad	1.84	4.41(+2.57)	1.83(-0.01)	4.3(+2.46)
		background	0.48	1.02(+0.54)	-0.09(-0.57)	0.77(+0.29)
		upsample	1.23	2.13(+0.9)	0.99(-0.24)	1.96(+0.73)
	Density	*cutout*	8.32	7.54(-0.78)	8.12(-0.2)	7.28(-1.04)
		local_dec	21.07	15.93(-5.14)	16.83(-4.24)	15.92(-5.15)
		local_inc	1.35	2.14(+0.79)	1.24(-0.11)	1.99(+0.64)
		beam_del	0.39	1.16(+0.77)	0.28(-0.11)	0.89(+0.5)
		layer_del	4.50	4.44(-0.06)	4.37(-0.13)	4.27(-0.23)
Object-level	**Noise**	*uniform*	4.05	3.64(-0.41)	3.94(-0.11)	3.48(-0.57)
		gaussian	5.42	4.84(-0.58)	5.32(-0.1)	4.73(-0.69)
		impulse	1.86	2.44(+0.58)	1.88(+0.02)	2.36(+0.5)
		upsample	-0.17	0.43(+0.6)	-0.41(-0.24)	0.29(+0.46)
	Density	*cutout*	21.09	19.38(-1.71)	20.97(-0.12)	19.44(-1.65)
		local_dec	18.12	16.9(-1.22)	18.12(+0.0)	17.03(-1.09)
		local_inc	8.46	6.76(-1.7)	8.3(-0.16)	6.71(-1.75)
	Transformation	*shear*	17.37	14.64(-2.73)	17.22(-0.15)	14.44(-2.93)
		FFD	12.26	11.5(-0.76)	12.13(-0.13)	11.39(-0.87)
		rotation	0.07	1.0(+0.93)	-0.18(-0.25)	0.91(+0.84)
		scaling	3.87	4.73(+0.86)	3.64(-0.23)	4.61(+0.74)
		translation	-1.18	-0.25(+0.93)	-1.12(+0.06)	-0.28(+0.9)
Average			8.57	7.84(-0.73)	8.2(-0.37)	7.68(-0.89)

basis of the data of real clean. According to Figure 1, 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 S8 in the website https://sites.google.com/ualberta.ca/robustness1pc2detector/.

Appendix C
Implementation of Corruption Simulation

Figures S9 to S34 in the website https://sites.google.com/ualberta.ca/robustness1pc2detector/ display the point cloud examples under "clean" and simulated common corruptions (at severity level of 3). For implementation details of the simulation, please refer to the website https://sites.google.com/ualberta.ca/robustness1pc2detector/.

TABLE S12: Classification on real and simulated data

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

TABLE S13: MMD distances of different features transformed by T-SNE

	Real *vs* Simulated	**Simulated *vs* Clean**	**Real *vs* Clean**
snow	0.0549	0.1446	0.1445
fog	0.0302	0.1212	0.1295