Abstract—Notwithstanding the prominent performance shown in various applications, point cloud recognition models have often suffered from natural corruptions and adversarial perturbations. In this paper, we delve into boosting the general robustness of point cloud recognition, proposing Point-Cloud Contrastive Adversarial Training (PointCAT). The main intuition of PointCAT is encouraging the target recognition model to narrow the decision gap between clean point clouds and corrupted point clouds by devising feature-level constraints rather than logit-level constraints. Specifically, we leverage a supervised contrastive loss to facilitate the alignment and the uniformity of hypersphere representations, and design a pair of centralizing losses with dynamic prototype guidance to prevent features from deviating outside their belonging category clusters. To generate more challenging corrupted point clouds, we adversarially train a noise generator concurrently with the recognition model from the scratch. This differs from previous adversarial training methods that utilized gradient-based attacks as the inner loop. Comprehensive experiments show that the proposed PointCAT outperforms the baseline methods, significantly enhancing the robustness of diverse point cloud recognition models under various corruptions, including isotropic point noises, the LiDAR simulated noises, random point dropping, and adversarial perturbations. Our code is available at: https://github.com/shikiw/PointCAT.

Index Terms—Point cloud recognition, adversarial learning, model robustness.

I. INTRODUCTION

Point cloud, captured by 3D sensors like LiDAR and Kinect, has become one of the most popular representations for depicting object surfaces and modeling 3D shapes. Its impressive performance has been witnessed in various applications (e.g., robotics, immersive tele-presence) and security-critical scenarios like autonomous driving, autopilot, etc. With the purpose of precisely categorize 3D objects, point cloud recognition [3], [4] takes a basic and significant part in many downstream tasks such as point cloud analysis [5], [6] and 3D object detection [7], [8], [9].

However, point cloud recognition is still confronted with many threats in practice. Due to the unpredictable environment or the inherent limitations of scanning equipment, it is inevitable for point cloud data to get perturbed and mitigated by real-world corruptions or reconstruction distortions from images. More crucially, recent works [10], [11], [12], [13] have demonstrated that point cloud recognition models are also susceptible to adversarial attacks especially adaptive attacks [14], which is attributed by the vulnerability of deep neural networks to imperceptible perturbations [15], [16], [17], [18], [19], [20]. These corruptions seriously affects the accuracy of point cloud recognition, bringing a lot of security concerns and disputes on issues like autonomous driving. Accordingly, the general robustness of point cloud recognition still remains an imperative topic for nowadays study.

Existing works on robust point cloud recognition can be divided into two categories. The first one is to preprocess the inputs and purify them into clean data, e.g., statistical outlier removal [21]. But the problem is that, the involvement of such preprocess modules often leads to extra time cost, which dramatically degrades the inference efficiency. The second one is to enhance the recognition model itself, e.g., using gather-vectors [22] or adversarial training methods [23], [24], [25] that succeed in image tasks. Unfortunately, the robustness improvement provided by these methods is still far from satisfactory for defending against both natural corruptions and adversarial attacks.

We investigate into the commonality of previous adversarial training methods, finding that most of them focus on designing a logit-level loss function to construct the robust decision boundary [24], [25], but ignore the significance of learning robust feature representation. As t-SNE [26] visualized in Fig. 1, when we project both clean samples and corrupted samples onto the hypersphere space, the features learned by vanilla adversarial training (AT) [23] or TRADES [24] are obviously not clustered, leading to more difficult construction for decision boundary. By contrast, our method can learn the more discriminative and category-clustered representations, thus the subsequently classification will be much easier. From this
perspective, we conclude the two objectives for learning robust feature representation as: 1) facilitating the category-wise alignment and uniformity on feature hypersphere; 2) enabling the same categorized features to be concentrated.

Motivated by the two objectives, we propose a novel prototype-guided contrastive adversarial training method for robust point cloud recognition, named “PointCAT”. In the light of InfoNCE loss [27] that performs well in contrastive learning [28], [29], [30], [31], we leverage its supervised variant [32] to mitigate the feature-level discrepancy among both clean/corrupted point clouds that are in the same category, chasing the first objective mentioned above. For the second objective, we design a pair of centralizing losses for clean/corrupted point clouds respectively, where a set of prototypes are specially defined as the guidance. These prototypes can be regarded as the feature-level centres of different category clusters, which are dynamically optimized with a data-independent strategy during training.

Besides the robust training paradigm, we further consider how to generate corrupted point clouds. Intuitively, with the enough strong model capacity, exposing the model to more diverse and strong corruptions can make it perform better. While it is impossible to involve all kinds of corruptions into our training, we utilize an autoencoder-like generator as a learnable attacker to explore more challenging corruptions. A learnable attacker usually behaves in a more flexible manner [33], [34], since it does not follow a pre-defined attack setting or a fixed configuration like previous adversarial training methods (e.g., PGD-like [23] inner loop for vanilla AT). Such flexibility allows the distribution diversity of synthetic corruptions, which is beneficial for broadening the learning scope of recognition model. Both the noise generator and the recognition model are alternately updated from the scratch. Thus the noise generator generate progressively hard corruptions, acting as a good teacher for the recognition model and adapting it to the perturbations under different intensities.

We verify the efficiency of the proposed PointCAT on four point cloud datasets for various recognition models, including PointNet [35], PointNet++ [36], DGCNN [37] and CurveNet [1]. Experimental results show that our method can not only outperform previous point cloud defenses and advanced adversarial training methods that succeed in image tasks, but also dramatically boost the robustness against black-box attacks, white-box attacks, Auto-Attack (AA) [38] and simulated LiDAR noises. Furthermore, we present LiMN20, a new dataset for validating the point cloud recognition robustness under the LiDAR scanning scenario, which consists of 1,000 complicated point clouds sampled by Blensor simulation [39] in different camera positions and angles.

To summarize, our contributions are four-fold as below:

- We propose PointCAT, a novel contrastive adversarial learning framework for robust point cloud recognition.
- To the best of our knowledge, we are the first to consider the point cloud model robustness against both natural corruptions and adversarial attacks.
- We propose feature-level constraints and dynamic prototype guidance for robust feature representation, with a learnable generator to derive challenging corruptions.
- Extensive evaluation on four datasets prove the superior performance of PointCAT. Besides, we contribute LiMN20, a new dataset regarding LiDAR-simulated point clouds for real-world robust recognition.

II. RELATED WORK

A. Point Cloud Recognition

Point cloud is one of the data formats to describe the scanned object surface, which is irregularly formed by a set of unordered and discrete points with 3D coordinates. To precisely recognize point cloud objects, various fundamental backbones or feature extracting strategies [35], [36], [40], [41], [42], [43], [44], [45] have been proposed in recent years. PointNet [35] is one of the pioneering works to directly utilize multi-layer perceptron (MLP) to extract point features and aggregate them by Maxpooling. And its variant, named PointNet++ [36], leverages the hierarchical structure to further improve the local feature extraction. With the well-designed neighborhood graphs, DGCNN [37] is one of the most representative works which adopt convolutional networks to exploit point structures. Recently, CurveNet [1] achieves the more satisfying classification accuracy on ModelNet40 through taking guided walks and aggregating hypothetical curves in point clouds. In this paper, we mainly implement PointCAT on the four aforementioned recognition models to demonstrate its efficacy.

B. Adversarial Training

Adversarial training, first proposed by Madry et al. [23], is a well-known countermeasure that can effectively defend against input corruptions especially adversarial perturbations. There are a lot of works focusing on adversarial training, developing a variety of methods based on static ensemble models [46], hidden layer noise propagation [47], adversarial logit pairing and universal first-order adversary [23] (i.e., using PGD as the inner loop), etc. Besides, Zhang et al. [24] explored the trade-off between model adversarial robustness and natural accuracy by regularizing the model output from natural images and adversarial inputs. Cui et al. [25] further improved the PGD-based adversarial training by forcing the robust model to inherit the classification boundary of the clean model. Some of the most advanced adversarial training approaches, though originally designed for 2D image tasks, are implemented on 3D point clouds as the baselines in this paper.

More recently, some works [48], [49], [50], [51], [52], [53] collaborate with contrast-related paradigms to strengthen the robustness of image networks. However, like original contrastive learning, most of these works still relies on strong image data augmentation. But image and point cloud are totally different data formats. Image is continuous signals with dense semantic, thus strong augmentation will not change its semantic. On the contrary, point cloud (single object for recognition) is a sparse point set with limited semantic, strong augmentation would change its semantic so it is hard to design proper augmentation for point cloud contrastive learning.
Fig. 1. Feature visualization on CurveNet [1] trained by vanilla AT, TRADES and our PointCAT, respectively. The point cloud samples are randomly selected from 15 categories in ModelNet40 [2] and subsequently corrupted as model inputs. Different numbers or colors corresponds to different categories. PointCAT learns the obviously more clustered and discriminative features.

Fig. 2. Schematic diagram of PointCAT. Different colors denotes different categories. “cl” means the “clean” sample, and “co” means the “corrupted” sample. Taking the “cl” sample as an example, we first learn the noise generator by pulling the “co” sample away from “cl” sample and its prototype. Then we train the recognition model by narrowing the feature gap between “cl” and “co” samples, meanwhile centralizing them towards their prototype and pulling them away from other prototypes. Finally, the learned features are more clustered and the simulated prototypes are more evenly distributed on the hypersphere.

To tackle with this, this paper propose a learnable noise generator to online provide augmented corruptions, which generate challenging positive pairs and force the model to learn robust semantic features.

C. 3D Adversarial Attack and Defense

1) Point Cloud Adversarial Attack: The topic of adversarial attack and defense on point cloud recognition has drawn increasing attentions from both industry and academia. Xiang et al. [11] introduced optimization-based attack C&W [16] to generate adversarial 3D point clouds and realize white-box attack. Zhou et al. [54] proposed a flexible label-guided framework to optimize targeted adversarial point clouds. Compared to the 2D counterparts, 3D adversarial attack often shows less transferability due to the very difference among point cloud recognition models, thus Hamdi et al. [55] introduced autoencoder-based reconstruction into optimization to improve the adversarial transferability. To guarantee the surface-level smoothness and fairness, the geometry-aware methods like GeoA3 [10], shape constrained methods like SI-Adv [56] and ITA [57] are designed to boost the attack imperceptibility. To perform the more practical adversarial attack in autonomous driving, Cao et al. [58] and Sun et al. [12], [59] systematically implemented point cloud attack to the LiDAR perception module and demonstrated the security threats brought by 3D adversarial objects.

Considering current autonomous driving is often equipped with multi-sensor fusion perception, a new physical-world attack [60] has been proposed to successfully fool both the camera and LiDAR.

2) Point Cloud Adversarial Defense: How to effectively defend against the aforementioned attacks is still a under-researched problem. More and more point cloud defense solutions have been presented to alleviate this situation. By eliminating statistical outliers and upsampling point clouds, Zhou et al. [21] were the first to propose the pluggable preprocess module SOR and DUP-Net for point cloud defense. Dong et al. [22] proposed GvG through calculating gather-vectors to indicate global center and checking if it deviates from the normal region. But unfortunately, current countermeasures including adversarial training are still ineffective or time-consuming against both natural corruptions and adversarial perturbations. Hence it is significant to find a more general framework that can strengthen the overall robustness of 3D recognition models.

III. PROPOSED METHOD

We first give an overview of the proposed PointCAT framework. Then we introduce the implementation details about the proposed method, including network design, objective loss function, pseudocode and the detailed procedure. Overall, the simplified training procedure is illustrated in Fig. 2.
A. Overview

The key idea of PointCAT is encouraging the recognition model to narrow the decision gap between clean example $x$ and corrupted example $x'$, driving the model to learn robust representations of both $x$ and $x'$ on the hypersphere feature space. During the training phase, we adversarially search the perturbation with an autoencoder-like generator $G$ to acquire the more challenging $x'$. To avoid both $x$ and $x'$ deviating from their ground-truth category clusters in the hypersphere feature space, we optimize the prototypes of all classification categories to limit features of both $x$ and $x'$ within their belonging clusters. Overall, the pipeline of PointCAT is illustrated in Fig. 3.

Similar with the standard adversarial training proposed by Madry et al. [23], our method also follows the adversarial game principle. Both noise generator $G$ and recognition model $M$ are trained from the scratch, where $G$ acts as the adversary and $M$ performs as the defender. Specifically, $G$ strives to explore the feature-level weakness of $M$ and generates hard corrupted example by $x' = x + \sigma(G(x))$, where $\sigma$ denotes the $l_2$-norm ball to constrain the perturbation. Conversely, $M$ delves itself into contrasting the projected feature of corrupted example $x'$ with the clean counterpart $x$ in a supervised way. The training objective can be formulated as the following optimization problem:

$$
\begin{align*}
\min_\theta & \ L_{\text{robust}}(P \circ M_e(x), P \circ M_e(x'), y), \\
\text{s.t.} & \min_\psi L_{\text{adv}}(P \circ M_e(x), P \circ M_e(x'), y),
\end{align*}
$$

(1)

where $\circ$ means the composite mapping applied to serially connect the recognition encoder $M_e$ with a projection head $P$, as suggested in the famous contrastive learning method SimCLR [28]. Recognition encoder with projector $P \circ M_e$ and noise generator $G$ are parameterized by $\theta$ and $\phi$, respectively. Note that ground-truth label $y$ is adopted to compute the prototype of the category that corresponds to $y$, and the projector head $P$ will be discarded when the whole training gets completed.

B. Dynamic Prototype Optimizing

At the beginning of each batch training, we update the estimated prototypes for all of $M$ categories (e.g., airplane, bench). Different from the solution used in deep clustering [61], [62], we adopt a data-independent strategy to search these prototypes since adversarial training is usually conducted in the supervised way. From this perspective, we investigate the original recognition model $M$ (i.e., $M_f \circ M_e$, in which $M_f$ refers to the final layer of the model) through calculating the following logit loss [16] without max-margin:

$$
\ell_y(\psi) = \left[ M(\psi)_y - \max_{t \neq y} [M(\psi)]_t \right].
$$

(2)

Here $y$ is the category label, $[M(\psi)]_y$ represents the predicted probability score of label $t$, and $\psi$ is the input variable which we intend to optimize. Here the intuition is that, when the score difference between $y$ and the second highest category becomes larger, the estimated prototype can be closer to the intra-class centre. By maximizing the logit loss $\ell_y$ for each category $y$, we can get the expected prototype $c_y$ with the recognition encoder, i.e.,

$$
c_y = M_e(\psi^*_y), \quad \text{s.t.} \quad \psi^*_y = \arg \max_\psi \ell_y(\psi).
$$

(3)

All of $\psi$ are initialized as random Gaussian noise points before the whole adversarial training procedure. During each batch training of PointCAT, we spend a few but fixed steps to maximize this logit loss and update prototypes iteratively. In this way, the evolution of prototypes and the recognition encoder can be always matched and synchronized.

C. Adversarial Noise Generation

Adversarial noise generation [63], [64] has been proved to discover the stronger and more diverse adversarial examples with the expressive power of DNNs on 2D images, where it often minimizes the logit-level loss to optimize the generator network. However, we should argue that the logit-level objective is not an appropriate choice for adversarial noise generation in 3D point cloud.

To clarify this point of view, we begin with a common observation regarding the entropy gap of predicted logits for 2D and 3D instances. Usually, there are multiple objects in the 2D image, e.g., an image of “running dog” may include the context of “grass” though it is often labeled as “dog”. While for 3D point cloud, only one object is commonly presented without any extra objects or contexts, e.g., a set of...
“airplane” point clouds often share the highly similar shapes and the shapes can determine their representations. This view is consistent with the observation of that, the classification logit of 3D point cloud often has the lower entropy than 2D image classification logit, i.e., the probability gap between the ground-truth class and the remaining classes in point cloud classification can be much larger than that in image classification. Since adversarial noise generation expects to explore more diverse and challenging adversarial examples, it is difficult for the logit-level objective to reach this goal since the low-entropy probability prediction greatly compromises the diversity of its logit probabilities. By contrast, features can be more expressive than logit probabilities in 3D object classification, containing more representations depicted the point cloud information such as angles, locations, sizes, etc. Moreover, feature-level objective [65], [66], [67] is proved to contribute more strong generality for adversarial examples when confronted with different target models.

Therefore, we specially devise a feature-level objective to train our adversarial noise generator \( \mathcal{G} \) in PointCAT. Given a batch of clean point clouds \( \{ x_k \}_{k=1}^N \) where \( x \in \mathbb{R}^{K \times 3} \), i.e., each one consists of total \( K \) points, we first leverage Combined Multi-Layer Perception (CMLP) [68] to encode them into \( K \)-dimensional latent vectors. Compared with original PointNet encoder [35], the channel-wise hierarchical design enables CMLP to absorb the information from both shallow-level and high-level features, alleviating the strong effect of symmetric function Maxpooling. Then the latent vectors are decoded by two fully connected layers and two Multi-Layer Perception (MLP) modules. After reshaping them into the input shape, we constrain the perturbations into \( l_2 \)-norm ball and add them on original point clouds to obtain adversarial corrupted point cloud by

\[
x'_k = x_k + \sigma(\mathcal{G}(x_k)), \quad k = 1, \ldots, N, \tag{4}
\]

where \( \mathcal{G} \) denotes the whole autoencoder-like noise generator. In order to generate the more challenging \( x'_k \), the training objective of \( \mathcal{G} \) is specified as: 1) to enlarge the gap between \( x'_k \) and \( x_k \), 2) to find \( x'_k \) which escapes as far away from its prototype \( c_{yk} \) as possible. Here \( y_k \) is the ground-truth label of both \( x_k \) and \( x'_k \). Let \( \{ z_i \}_{i=1}^{2N} \) be all of total \( 2N \) projections calculated from both clean features and corrupted features, i.e., \( \forall k = 1, \ldots, N \),

\[
z_{2k-1} = \mathcal{P} \circ \mathcal{M}_c(x_k),
\]

\[
z_{2k} = \mathcal{P} \circ \mathcal{M}_c(x'_k), \tag{5}
\]

the objective function \( \mathcal{L}_{adv} \) can be formulated as a weighted combination of the following two parts:

\[
\mathcal{L}_{sup} = \frac{1}{N} \sum_{k=1}^{N} - \exp \left( \frac{- \text{sim}(z_{2k-1}, y_k)}{\tau_{cen}} \right),
\]

\[
\mathcal{L}_{adv} = \frac{1}{N} \sum_{k=1}^{N} - \exp \left( \frac{- \text{sim}(z_{2k}, y_k)}{\tau_{cen}} \right), \tag{6}
\]

where \( \text{sim}(\cdot, \cdot) \) denotes the cosine similarity function, symmetrically, \( \text{sim}(u, v) = u \cdot v / |u||v| \), and \( \tau_{adv} \) is a temperature hyper-parameter. \( \omega_{yk} \) is the projection of prototype, i.e., \( \omega_{yk} = \mathcal{P}(c_{yk}) \). Therefore, the whole adversarial objective loss for training generator \( \mathcal{G} \) is

\[
\mathcal{L}_{adv} = \mathcal{L}_{gap} + \beta \mathcal{L}_{esc}, \quad (7)
\]

where \( \beta \) controls the weight of escaping loss component.

D. Feature Contrast and Centralizing

When completing the generation of corrupted point clouds, we subsequently update the parameter of the recognition encoder with projector \( \mathcal{P} \circ \mathcal{M}_c \). Specifically, we regard all batch samples that belong to category \( y_k \) except \( x_k \) is the positive set of \( x_k \), while the remaining samples construct the negative set. Note that both clean projection \( z_{2k-1} \) and corrupted projection \( z_{2k} \) belong to the same category \( y_k \). Thus the indices of the positive set corresponding to projection \( z_i \) can be defined as \( i(i) = \{ j \in [1, \ldots, 2N] / [i], y_{j(2)} = y_{i(2)} \} \). Apparently, \( \mathcal{P}(2k-1)/2k = \mathcal{P}(2k)/(2k - 1) \). To facilitate the alignment and the uniformity of hypersphere features, we adopt supervised contrastive loss [32] to narrow the decision gap among positive set examples, i.e., \( \forall i = 1, \ldots, 2N \),

\[
\ell^s_{i} = \sum_{j=1}^{2N} \frac{1}{|P(i)|} \log \frac{\exp \left( \frac{\text{sim}(z_i, z_j)/\tau_{sup}}{\tau_{cen}} \right)}{\sum_{s=1}^{2N} \sum_{t \neq i}^{N} \exp \left( \frac{\text{sim}(z_i, z_s)/\tau_{sup}}{\tau_{cen}} \right)}, \tag{8}
\]

where \( \mathbb{1} \) refers to the characteristic function and \( \tau_{sup} \) is a temperature hyper-parameter. \( |P(i)| \) means the total number of examples in this positive set.

However, it is not enough to lean a robust \( \mathcal{M} \) if only using contrastive loss during the training phase. To avoid both the learned representations of \( x_k \) and \( x'_k \) deviating from their ground-truth category cluster, we draw them towards their prototype \( c_{yk} \) and pull them away from other prototypes simultaneously, by introducing a pair of centralizing loss for \( x_k \) and \( x'_k \) respectively:

\[
\ell^c_{k} = \frac{1}{M} \sum_{i=1}^{M} \frac{\exp \left( \frac{\text{sim}(z'_{2k-1}, \omega_{yk})/\tau_{cen}}{\tau_{cen}} \right)}{\sum_{i \neq y_k} \exp \left( \frac{\text{sim}(z_{2k-1}, \omega_i)/\tau_{cen}}{\tau_{cen}} \right)}, \tag{9}
\]

where \( \tau_{cen} \) is another temperature hyper-parameter. With the guidance of prototypes, the recognition model \( \mathcal{M} \) can avoid model collapse and maintain the high recognition accuracy. Finally, we calculate the average value of supervised contrastive loss and centralizing loss across the whole batch. Overall, the robust loss function of \( \mathcal{M} \) is defined as:

\[
\mathcal{L}_{sup} = \frac{1}{2N} \sum_{i=1}^{2N} \ell^s_{i}, \quad \mathcal{L}_{cen} = \frac{1}{N} \sum_{k=1}^{N} \left( \ell^c_{k} + \ell^adv_{k} \right), \tag{10}
\]

\[
\mathcal{L}_{robust} = \mathcal{L}_{sup} + \alpha \mathcal{L}_{cen}, \tag{11}
\]

where \( \alpha \) balances the weight of centralizing loss.
Algorithm 1 Point-Cloud Contrastive Adversarial Training

Input: recognition model $M = M_f \circ M_c$, noise generator $G$ parameterized by $\phi$, model encoder with projector $P \circ M_c$ parameterized by $\theta$, $M_f$ parameterized by $\theta_f$, prototype update iterations $T_1$, inner loop number $T_2$, hyper-parameter $\alpha$, $\beta$, $\eta_1$ and $\eta_2$.

Output: robust recognition model $M_r$.

Initialize $M$, $G$, $P$ randomly.

Initialize prototype inputs $\psi_y$ with warm start;

repeat

Sample batch data $\{x_k\}_{k=1}^{N}$ with labels $\{y_k\}_{k=1}^{N}$;

for $t = 1$ to $T_1$ do

Update $\psi_y = \psi_y + \eta_1 \nabla_{\phi}(\psi_y)$; \quad # Eq.(3)

end for

Compute $c_y = M_c(\psi_y)$; \quad # update prototypes

for $t = 1$ to $T_2$ do

Compute $L_{adv} = L_{gap} + \beta L_{cen}$; \quad # Eq.(7)

Update $\phi = \phi - \eta_2 \nabla_{\phi} L_{adv}$; \quad # update $G$

end for

Compute $L_{robust} = L_{sup} + \alpha L_{cen}$; \quad # Eq.(11)

Update $\theta = \theta - \eta_2 \nabla_{\theta} L_{robust}$; \quad # update $P \circ M_c$

if end of epoch then

Update $\theta_f = \theta_f - \eta_2 \nabla_{\theta} CE(x,y)$; \quad # update $M_f$

end if

until training converges

E. The Pseudocode of the Proposed Method

We provide the detailed algorithm pseudocode in Algo 1 and the pipeline framework in Fig. 3. Overall, given a batch of point cloud data, the training mainly consists of three steps. For the first step, we spend $T_1$ iterations to regularly update all prototypes with Eq.(3) where $T_1$ is configurable. For the second step (the inner loop), we minimize the adversarial objective loss in Eq.(7) to train the noise generator $G$ and get the corrupted point clouds with Eq.(4). For the third step (the outer loop), we train recognition encoder with projector $P \circ M_c$ by minimizing $L_{robust}$ (Eq.(11)), where we leverage both the supervised contrastive objective and the feature centralization objective. These three steps are alternated until the training converges. After each epoch training, the classification layer $M_f$ will be tuned with cross-entropy loss on clean data. The projector head $P$ will be discarded when the whole training procedure gets completed.

IV. EXPERIMENT

A. Experimental Setup

1) Datasets: We conduct the experiments on four point cloud object datasets (i.e., three popular point cloud benchmarks and our proposed LiMN20) to comprehensively validate the performance of different point cloud training methods.

- ModelNet40 [2] consists of 12,311 CAD models from 40 man-made object categories, split into 9,843 for training and 2,468 for test. Each point cloud is formed by 1,024 points which are uniformly sampled from the surface of each object and rescaled into a unit cube.

- ShapeNetPart [69] contains 16,881 pre-aligned shapes from 16 categories that are more closer to the real LiDAR data, split into 12,137 for training and 2,874 for test. We follow the same operations as ModelNet40 to process each point cloud sample.

- ModelNet40-C [70] is a dataset specially designed for the corruption robustness of point cloud recognition. It is a corrupted version of ModelNet40 validation set that covers 15 common corruption types (e.g., occlusion, shearing or background noises) with 5 severity levels for each type.

- LiMN20 is our newly proposed dataset for verifying the recognition robustness under LiDAR scanning scenario. To simulate the LiDAR noise, we use a virtual Velodyne HDL-64E2 scanner provided by Blensor [39] to scan 100 3D meshes randomly selected from 20 confusing categories of ModelNet40. It totally contains 1,000 shapes, split into a half for “easy” set and the other half for “hard” set. The “easy” split is sampled by the standard simulated LiDAR scanner, while the “hard” split is sampled by the simulated LiDAR scanner with noises. More details can be found in Sec. IV-C.3.

2) Models: We generally consider four point cloud recognition models to evaluate the proposed method, including PointNet [35], PointNet++ [36], DGCNN [37] and CurveNet [1]. These models are really representative since they leverage very different strategies to learn the point cloud features and greatly inspire the community.

3) Implementation Details: Here we provide the default settings and hyper-parameters for PointCAT implementation. The learning rate is assigned as 0.001 and we use Adam optimizer with a cosine annealing schedule to train 155 epochs for all models. Besides, we fix a set of temperature hyper-parameters in all experiments, i.e., $\tau_{adv}$, $\tau_{sup}$ and $\tau_{cen}$ are set as 0.1, 0.1 and 0.25 respectively. For PointNet, the weight parameter $\alpha$ and $\beta$ are set as 8 and 0.5, respectively. For other point cloud models, $\alpha$ and $\beta$ are unified as 1 and 4, respectively. For efficiency, the iteration number $T_1$ of prototype update is assigned as 10 for inference-fast models (i.e., PointNet and DGCNN) and 2 for inference-slow models (i.e., PointNet++ and CurveNet). Moreover, the inner loop number of updating the noise generator is unified as 4 for all experiments. We ablate the aforementioned hyper-parameters in Sec. IV-D.5.

B. Robustness on Adversarial Attacks

1) Regular White-Box Attacks: We compare our method with the following baselines, i.e., point cloud defense SOR [21], advanced adversarial training methods including PGD-based AT [23], TRADES (1/λ = 1) [24] and LBGAT (α = 0) [25]. Note that all adversarial training baselines share the same training settings with PointCAT, including the same learning rate as 0.001, the same perturbation threshold as 0.04 and the same inner loop number as 4. For SOR, we adopt the default hyper-parameter setting of its paper. To verify the white-box robustness of point cloud models equipped with these defenses, we conduct regular gradient-based adversarial attacks FGM [20], IFGM [71], MIFGM [17], PGD [23] and
TABLE I
QUANTITATIVE COMPARISON ABOUT REGULAR WHITE-BOX ROBUSTNESS ON MODELNET40, TESTED ON POINTNET. “NONE” DENOTES NO DEFENSE. “ACC” MEANS THE TOP-1 ACCURACY ON CLEAN POINT CLOUDS. “ASR” MEANS THE ATTACK SUCCESS RATE, LOWER IS BETTER.

| Norm | Defense | Acc (%) | FGM [20] | IFGM [71] | MIFGM [17] | PGD | C&W [16] | ASR (Targeted Attack) (%) | ASR (Untargeted Attack) (%) |
|------|---------|---------|----------|-----------|-------------|------|----------|--------------------------|----------------------------|
| None | SOR [21] | 87.58   | 4.21     | 73.46     | 48.99       | 85.37 | 99.19   | 34.40         | 69.41                   |
|      | AT [23]  | 87.44   | 3.53     | 79.34     | 79.29       | 82.66 | 94.73   | 54.09         | 78.16                   |
|      | TRADES [24] | 86.59 | 3.81     | 86.79     | 85.49       | 89.14 | 98.74   | 58.91         | 87.44                   |
|      | LBGAT [25] | 81.69 | 3.36     | 49.88     | 47.61       | 53.20 | 67.14   | 65.24         | 84.36                   |
| Ours |          | 87.97   | 2.76     | 21.35     | 20.91       | 24.07 | 36.10   | 37.60         | 65.76                   |
|      | SOR [21] | 87.58   | 3.61     | 80.51     | 72.57       | 79.98 | 72.73   | 43.88         | 79.05                   |
|      | AT [23]  | 87.44   | 2.59     | 73.14     | 63.29       | 72.61 | 80.55   | 31.89         | 72.37                   |
|      | TRADES [24] | 86.59 | 2.27     | 83.35     | 71.03       | 83.55 | 82.29   | 35.45         | 76.01                   |
|      | LBGAT [25] | 81.69 | 2.63     | 44.73     | 47.08       | 44.77 | 71.27   | 33.67         | 64.18                   |
| Ours |          | 87.97   | 2.92     | 22.41     | 22.61       | 21.64 | 64.51   | 24.55         | 59.81                   |

Fig. 4. Comparison about regular white-box robustness on ModelNet40, tested on PointNet++, DGCNN (middle row) and CurveNet (bottom row). “None” denotes no defense. “ASR” means the attack success rate, lower is better. We randomly select five random seeds to train the models and plot the results with error bars.

optimization-based attack C&W [16] from both targeted and untargeted perspectives.

For targeted gradient-based attacks, we unify the attack iteration as 50 and assign the perturbation threshold as 0.08 for $l_2$-norm attack, 0.32 for $l_\infty$-norm attack. For targeted C&W attack, we set 10 binary steps with total 500 iterations for adversarial point cloud optimization, where the learning rate is 0.01. For untargeted gradient-based attacks, we unify the attack iteration as 10 and assign the threshold as 0.02 for $l_2$-norm attack, 0.08 for $l_\infty$-norm attack. For untargeted C&W attack, we adopt 5 binary steps with total 250 iterations, where the optimizing learning rate is 0.003. The step size of all $l_2$-norm attacks is calculated by dividing $\delta \sqrt{K \times C}$ by the iteration number, where $\delta$ is the perturbation threshold and $K \times C$ denotes the input dimensions of each point cloud.

As the comprehensive results shown in Table I and II, Fig. 4 and 5, the point cloud recognition models trained by PointCAT performs generally more robust when confronted with different regular white-box adversarial attacks, with really few clean accuracy degraded. It can be easily noticed that other adversarial training methods that succeed in image recognition tasks are still limited for point cloud recognition: TRADES maintains comparable clean accuracy but gets weaker adversarial robustness than PointCAT, while
TABLE II

| Norm | Defense | Acc (%) | FGM [20] | IFGM [71] | MIFGM [17] | PGD [23] | C&W [16] | ASR (Targeted Attack) (%) | ASR (Untargeted Attack) (%) |
|------|---------|---------|----------|-----------|------------|----------|---------|---------------------------|-----------------------------|
| None | SOR [21] | 98.58   | 4.40     | 33.88     | 17.19      | 51.14    | 85.09   | 5.18                      | 18.11                      |
|      | AT [23]  | 98.36   | 6.72     | 62.25     | 60.02      | 66.21    | 79.61   | 32.71                     | 60.06                      |
|      | TRADES [24] | 98.43  | 7.17     | 72.06     | 69.62      | 76.97    | 82.67   | 28.53                     | 66.98                      |
|      | LBGAT [25] | 96.80  | 6.02     | 35.14     | 33.99      | 36.99    | 45.20   | 28.22                     | 65.69                      |
|      | Ours     | 98.61   | 2.61     | 7.97      | 8.49       | 10.26    | 18.02   | 6.78                      | 17.71                      |

| Norm | Defense | Acc (%) | FGM [20] | IFGM [71] | MIFGM [17] | PGD [23] | C&W [16] | ASR (Targeted Attack) (%) | ASR (Untargeted Attack) (%) |
|------|---------|---------|----------|-----------|------------|----------|---------|---------------------------|-----------------------------|
| None | SOR [21] | 98.58   | 5.50     | 23.24     | 61.45      | 24.22    | 59.52   | 4.56                      | 14.75                      |
|      | AT [23]  | 98.36   | 2.85     | 49.37     | 50.49      | 49.72    | 73.42   | 4.21                      | 14.68                      |
|      | TRADES [24] | 98.43  | 3.13     | 53.62     | 53.76      | 53.90    | 78.12   | 5.08                      | 17.15                      |
|      | LBGAT [25] | 96.80  | 3.06     | 25.96     | 31.07      | 26.97    | 54.78   | 5.78                      | 13.26                      |
|      | Ours     | 98.61   | 3.20     | 9.43      | 9.95       | 9.36     | 38.98   | 3.69                      | 13.05                      |

Fig. 5. Comparison about regular white-box robustness on ShapeNetPart, tested on PointNet++ (top row), DGCNN (middle row) and CurveNet (bottom row). “None” denotes no defense. “ASR” means the attack success rate, lower is better. We randomly select five random seeds to train the models and plot the results with error bars.

LBGAT obtains better robustness than TRADES but sacrifices more on clean accuracy. Another surprising finding is that, in some results, the clean accuracy (Acc) of our method is even higher than the vanilla training baseline (denoted as “None”). Considering that, in targeted attack scenario, simply calculating ASR to measure whether the model is robust is not entirely convincing, we also report the model classification accuracy (namely robust accuracy) in Table III and derive the similar conclusion. Overall, our PointCAT achieves the best performance on both clean accuracy and model robustness.

2) White-Box Auto-Attack (AA) and Point Cloud Attacks: To further evaluate the effectiveness of PointCAT on more stronger white-box attacks, we implement Auto-Attack [38] and three recently proposed point cloud adversarial attacks, i.e., 3D-Adv [11], AdvPC [55] and GeoA$^3$ [10] with the default settings given in their papers. Auto-Attack is widely recognized as a reliable approach for model robustness evaluation, which ensembles APGD-CE, APGD-DLR, FAB and Square Attack. Here only the first three attacks are adopted since Square Attack is specially designed for images and hard to be extended to point clouds. As the results listed in Table IV, the PointNet trained by PointCAT achieves the state-of-the-art robustness under these strong white-box attacks. Especially on three point cloud adversarial attacks, we dramatically outperforms both standard adversarial training and TRADES to decrease the ASR values.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
3) Black-Box Attacks: We verify the adversarial robustness of the proposed method on black-box attack, including both transfer-based attack and query-based attack. For the fairness of comparison, all of these recognition models are trained on ModelNet40 (or ShapeNetPart) training split under the same settings with PointCAT. For evaluating on transfer-based attack, we apply untargeted \textit{FGSM} for both evaluation on two datasets, which has 50 iterations for gradient ascent and gets $l_2$-norm constrained. The step size setup is same with that for white-box attacks. The adversarial point clouds are generated on the source model and tested on the target model as its input. As shown in Fig. 6, the black-box attack transferability is much lower among PointCAT trained models than TRADES trained models. For evaluating on query-based attack, we implement Nattack [72] and SPSA [73] into 3D point cloud version to attack the trained model. As shown in Table V, our method obtains the best robust accuracy under these two attacks. Accordingly, PointCAT enables point cloud models to share the stronger black-box robustness with each other and better restrict with both transfer-based attack and query-based attack.

C. Robustness on Natural Corruptions

1) Random Point Noise and Point Dropping: Since the real-world collected point clouds are often mutilated due to the complicated environments, it is necessary for the recognition model to resist the natural corruptions. In Table VI, we first utilize isotropic Gaussian noise and random point dropping to corrupt the model input of baseline methods and our models. The standard variance of Gaussian noise is assigned as 4\% or 8\% to mimic the point deviation from the surface, and the ratio of point dropping is set as 70\% or 80\% to construct sparse point clouds. We test different robust settings on CurveNet (a.k.a, CN), including 1) its vanilla version; 2) CN equipped with extra outlier removal modules SOR [21]; 3) LBGAT [25]; 4) ours.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 7. Visualization for some samples selected from “easy” split (top row) and “hard” split (bottom row) of LiMN20.

| Robust Setting | Acc (%) | Noisy Acc (%) | Sparse Acc (%) | $T_{in}$ (s) |
|----------------|---------|---------------|----------------|-------------|
| CN [1]         | 93.80   | 68.88         | 10.66          | 73.58       | 49.59       | 0.17     |

| Equipped with outlier removal |
|-------------------------------|
| CN+SOR [21]                  | 90.96   |
| CN+DUP-Net [21]              | 87.88   |

| Equipped with denoising or upsampling |
|---------------------------------------|
| CN+DMR [74]                         | 75.89   |
| CN+PU-Net [75]                      | 88.25   |

| Equipped with adversarial training |
|-----------------------------------|
| CN (AT [23])                      | 89.26   |
| CN (TRADES [24])                  | 90.84   |
| CN (LBGAT [25])                   | 85.74   |
| CN (Ours)                         | 90.84   |

**TABLE VI**

**Quantitative Comparison on ModelNet40 With Different Settings Implemented on CurveNet (a.k.a., CN), for the Robust Accuracy on Isotropic Gaussian Noisy or Sparse Point Clouds. “$T_{in}$” Denotes the Average Inference Time**

| Robust Setting | Split Error (%) |
|----------------|-----------------|
|                | Density | Noise | Trans. | Overall Error (%) |
| Equipped with augmentation methods |
| PointMixup [76] | 28.3    | 28.9  | 19.0   | 25.4     |

| Equipped with test-time adaption methods |
|-----------------------------------------|
| BN [77]                                  | 26.8    |
| TENT [78]                                 | 27.8    |

| Equipped with adversarial training |
|-----------------------------------|
| AT [23]                           | 26.7    |
| TRADES [24]                        | 27.6    |
| LBGAT [25]                         | 30.0    |
| Ours                               | 30.4    |

**TABLE VII**

**Robustness Comparison Against Common Corruptions on ModelNet40-C. The Evaluated Model Is PointNet. We Report the Recognition Error Rates Here, Lower Is Better**

2) **ModelNet40-C Common Corruptions:** ModelNet40-C [70] is a recently proposed dataset to benchmark the robustness of point cloud recognition models when confronting with different kinds of distortions. These distortions are common in real-world scenarios that are relevant with LiDAR scanning, which are divided into three splits, *i.e.*, density, noise and transformation. Each split has at least five type of distortions, where the intensity of distortions is also taken into consideration. In this paper, we directly conduct the evaluation on this benchmark to comprehensively verify the natural robustness to these common corruptions. Besides the adversarial training methods, we further include point cloud augmentation method PointMixup [76], test-time adaption method BN [77] and TENT [78] as the baselines.

To make a fair comparison, we take the default configuration of ModelNet40-C during evaluation. The results are shown in Table VII. We can find that PointCAT obtains the lowest overall error rate compared with other baselines especially on noise distortions, indicating its best robustness to common corruptions.

3) **LiDAR Simulated Noise on LiMN20:** In the more realistic scenario, we should take the distortions caused by LiDAR perception into consideration and test the recognition models under such distortions. Previous 3D object datasets uniformly sample points on shapes where the points are noise-free. However, this is mismatched with rotary scanning used by LiDAR in real complicated scenarios. Therefore, we intend to contribute a new dataset named LiMN20 to fill this gap, which prepares for LiDAR-scanned point clouds. To simulate the LiDAR noise, we use a virtual Velodync HDL-64E2 scanner provided by Blesson [39] to scan 100 3D meshes, which are randomly selected from 20 confusing categories (*e.g.*, bookshelf and tv stand) of ModelNet40 [2]. Sampled from different angles and positions with 1) the standard simulated LiDAR scanner and 2) the noisy simulated LiDAR scanner respectively, the proposed LiMN20 contains total 1,000 point clouds, in which 500 shapes construct the “easy” split and the other 500 shapes form the “hard” split. Moreover, the point number of each point cloud varies from 1,000 to 8,000, simulating the uncertain quantity of reflected points in real-world scanning scenarios. The visualization of some examples, the virtual...
### TABLE VIII

|                | None | SOR [21] | AT [23] | TRADES [24] | Ours |
|----------------|------|----------|---------|-------------|------|
| Acc (easy)     | 48.60| 45.40    | 42.00   | 37.20       | 65.00|
| Acc (hard)     | 32.80| 35.20    | 34.20   | 31.40       | 52.20|

**Robust Accuracy on the Proposed Dataset LiMN20. We verify CurveNet on both “easy” and “hard” validation splits. “None” means the vanilla model without any defenses.**

---

**Blensor simulation** [39] of LiDAR scanners and the settings for configuring the noisy simulated LiDAR are provided in Fig. 7, 8 and 9, respectively.

To better mimic the real-world LiDAR scanning scenario, we further verify baseline methods and our PointCAT on the proposed dataset LiMN20. The evaluation results are listed in Table VIII. Apparently, our method can also dramatically boost the accuracy under the simulated LiDAR noisy scenario, which demonstrates its practicality for LiDAR scenario and its resiliency against the noises raised during scanning.

### D. Ablation Studies

1) **Different Inner Loop Numbers** $T_2$: To explore the robustness improvement brought by inner loop numbers, we conduct the evaluation on PointNet trained by baselines and our method for 0, 4, 8, 12 inner loops, respectively. Two aspects of comparison are considered here, i.e., the robustness against white-box PGD attack and the average time budget for back-propagation in each batch training (batch size is 16). As indicated in Fig. 10, our method outperforms the baseline methods a lot while inheriting the lower time budget than TRADES and LBGAT. Even implemented with only 4 inner loops, our method are also better than previous adversarial training using PGD-8 and PGD-12 (i.e., AT with 8, 12 inner loops). It proves that despite replacing the traditional PGD-based inner loop as adversarial noise generation, PointCAT is also able to boost more robustness for point cloud recognition than previous adversarial training methods that take the extra time cost on more inner loops.

2) **Running Time Budget Analysis**: To clarify the time efficiency of PointCAT, we further conduct the experiments to obtain the average time cost for batch data back-propagation with TRADES [24], LBGAT [25] and PointCAT. For the fairness of evaluation, we adopt the same training settings, including the same perturbation threshold as 0.04, the same learning rate as 0.001, the same batch size as 16 and the same RTX 3090Ti GPU device. Note that the trained model is unified as PointNet++ [36], and our method adopt the same prototype update setting with that is given in Sec. IV-A.3, i.e., $T_1 = 2$, $η_1 = 0.005$. The detailed results are showcased in Fig. 11. When configuring fewer inner loops, PointCAT is little more time-consuming due to the involvement of extra prototype computation. When using more inner loops, PointCAT achieves the better time efficiency than both TRADES and LBGAT, owing to the replacement of traditional PGD loops with the lightweight noise generator training.

3) **Different Ways for Prototype Update**: As formulated in Eq.(3), we optimize model inputs to realize the...
TABLE IX

| Variable    | Acc (%) | Natural RA (%) | PGD ASR (%) |
|-------------|---------|----------------|-------------|
|             |         | Noise(8%) Drop(80%) | Targeted | Untargeted |
| feature     | 87.52   | 64.26          | 85.70      | 72.57      | 73.82      |
| model input | 87.97   | 67.54          | 86.18      | 24.07      | 67.99      |

Fig. 12. Ablation studies for hyper-parameters α and β. We report the natural robust accuracy (RA) against 8% isotropic noises, 80% point dropping and attack success rate (ASR) against targeted/untargeted PGD attack. We select α = 8, β = 0.5 on PointNet eventually.

data-independent prototype update. While a more straightforward way is directly optimizing hypersphere features and just computing on the last classification layer. But unfortunately, the results in Table IX shows that this way is much less effective than optimizing model inputs. It is because that directly optimizing features is unrestricted while optimizing model inputs just allows features to be confined to a specific encoding distribution.

4) Importance of The Proposed Mechanisms: To clarify the significance of loss components $L_{sup}$, $L_{cen}$, adversarial noise generation and dynamic prototype guidance, we conduct the ablation study on each of them. For two loss components, we abandon either of them in Eq.(11) to test the performance when just leveraging the remaining loss component. For adversarial noise generation, we replace it with the PGD-based inner loop to find the difference between before and after. For the prototype guidance, we remove the centralizing loss in Eq.(11) and the escaping component in Eq.(7). As indicated in Table X, all of these losses or mechanisms can dramatically help our method boost the adversarial robustness of point cloud recognition models. The results also show that, especially when equipping with $L_{cen}$, the prototype guidance takes an essential part in overall robustness improvement. It is consistent with the intuition of avoiding the learned features of positive pairs deviating from the ground-truth category cluster, where the corrupted positive samples are learning to be more challenging.

5) Ablating Hyper-Parameters: The configuration of the introduced hyper-parameters has a significant impact on the model robustness. To investigate such impact, we conduct the ablation experiments on different hyper-parameter settings. As the results listed in Fig. 12 and Table XI, though hyper-parameters change, our method always outperforms baselines with its relatively robust performance. With the unified implementation details given in Sec. IV-A.3, it is applicable for implement our method on most of common point cloud models to outperform existing baselines. Moreover, it definitely provides more flexible ways for achieving the more satisfying performance by fine tuning these hyper-parameters for the specific point cloud recognition model.

6) Efficiency Vs. Effectiveness: The prototype guidance is an integral part of the performance improvement as discussed in Sec. IV-D.4. While the dynamic prototype update mechanism requires more computation cost for training, we should clarify that such extra time budget is insignificant and configurable during training. First, we show that the overall running time of PointCAT is generally comparable with other adversarial training methods in Sec. IV-D.1 and Sec. IV-D.2. Second, we further verify that a modest reduction in the prototype update times (i.e., fewer update iterations $T_1$) does not degrade much robustness. Specifically, we can reduce the prototype update times to boost the training efficiency with little performance sacrificed, e.g., when we replace $T_1 = 10$, $\eta_1 = 0.001$ (10 iterations, 0.001 update rate) with $T_1 = 2$, $\eta_1 = 0.005$ (2 iterations, 0.005 update rate) for DGCNN [37], the degradation of its adversarial robustness is only 0.37% on untargeted PGD [23] attack. Therefore, “more prototype update iterations” is an optional enhancement for further improving the robustness if the computation resource is sufficient.

V. CONCLUSION

We propose Point-Cloud Contrastive Adversarial Training (PointCAT), to boost the general robustness of point cloud
object recognition. To facilitate the alignment and the uniformity of learned representations on the hypersphere feature space, we specially devise a pair of centralizing losses and supervised contrastive loss for robust model training. With the purpose of online generating the more challenging corrupted point clouds, a noise generator is adversarially trained along with the recognition model from the scratch. Extensive experiments on ModelNet40, ShapeNetPart, ModelNet40-C and LiMN20 for various point cloud recognition models demonstrate that, our method achieves the superior performance against both white-box and black-box adversarial perturbations, including strong Auto-Attack and natural corruptions such as isotropic point noise, point dropping and the simulated LiDAR noise. Besides, a new dataset named LiMN20 is contributed in this paper, for validating the robustness under noisy LiDAR scanning environment.

REFERENCES

[1] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, “Walk in the cloud: Learning curves for point clouds shape analysis,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 915–924.
[2] Z. Wu et al., “3D shapenets: A deep representation for volumetric shapes,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 1912–1920.
[3] F. Wang, W. Li, and D. Xu, “Cross-dataset point cloud recognition using deep-shallow domain adaptation network,” IEEE Trans. Image Process., vol. 30, pp. 7364–7377, 2021.
[4] L. Hui, M. Cheng, J. Xie, J. Yang, and M.-M. Cheng, “Efficient 3D point cloud feature learning for large-scale place recognition,” IEEE Trans. Image Process., vol. 31, pp. 1258–1270, 2022.
[5] S. Zhang, S. Cui, and Z. Ding, “Hypergraph spectral analysis and processing in 3D point cloud,” IEEE Trans. Image Process., vol. 30, pp. 1193–1206, 2021.
[6] S. Cheng, X. Chen, X. He, Z. Liu, and X. Bai, “PRA-Net: Point relation-aware network for 3D point cloud analysis,” IEEE Trans. Image Process., vol. 30, pp. 4436–4448, 2021.
[7] M. Feng, S. Z. Gilani, Y. Wang, L. Zhang, and A. Mian, “Relation graph network for 3D object detection in point clouds,” IEEE Trans. Image Process., vol. 30, pp. 902–1017, 2021.
[8] H. Liu, H. Liu, Y. Wang, F. Sun, and W. Huang, “Fine-grained multilevel fusion for anti-occlusion monocular 3D object detection,” IEEE Trans. Image Process., vol. 31, pp. 4050–4061, 2022.
[9] J. Guo et al., “Efficient center voting for object detection and 6D pose estimation in 3D point cloud,” IEEE Trans. Image Process., vol. 30, pp. 5072–5084, 2021.
[10] T. Li, X. Li, Y. Li, L. Chen, C. L. P. Chen, and K. Jia, “Geometry-aware generation of adversarial point clouds,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 6, pp. 2984–2999, Jun. 2022.
[11] C. Xiang, C. R. Qi, and B. Li, “Generating 3D adversarial point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 9128–9136.
[12] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, “Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and point countermeasures,” in Proc. 29th USENIX Security Symp., 2020, pp. 877–894.
[13] J. Tu et al., “Physically realizable adversarial examples for LiDAR object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 13716–13725.
[14] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2020, pp. 1633–1643.
[15] C. Szegedy et al., “Intriguing properties of neural networks,” in Proc. Int. Conf. Learn. Represent., 2014, pp. 1–10.
[16] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2017, pp. 59–57.
[17] Y. Dong et al., “Boosting adversarial attacks with momentum,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 9185–9193.
