Self-Robust 3D Point Recognition via Gather-vector Guidance

Xiaoyi Dong[1], Dongdong Chen[2]*, Hang Zhou[1] , Gang Hua[3], Weiming Zhang[1], Nenghai Yu[1],
[1]University of Science and Technology of China [2]Microsoft Cloud AI, [3]Wormpex AI Research
{dlight@mail., zh2991@mail., zhangwm@, ynh@ }.ustc.edu.cn {ganghua@,cddlyf@}gmail.com

Abstract

In this paper, we look into the problem of 3D adversary attack, and propose to leverage the internal properties of the point clouds and the adversarial examples to design a new self-robust deep neural network (DNN) based 3D recognition systems. As a matter of fact, on one hand, point clouds are highly structured. Hence for each local part of clean point clouds, it is possible to learn what is it ("part of a bottle") and its relative position ("upper part of a bottle") to the global object center. On the other hand, with the visual quality constraint, 3D adversarial samples often only produce small local perturbations, thus they will roughly keep the original global center but may cause incorrect local relative position estimation. Motivated by these two properties, we use relative position (dubbed as "gather-vector") as the adversarial indicator and propose a new robust gather module. Equipped with this module, we further propose a new self-robust 3D point recognition network. Through extensive experiments, we demonstrate that the proposed method can improve the robustness of the target attack under the white-box setting significantly. For I-FGSM based attack, our method reduces the attack success rate from 94.37 % to 75.69 %. For C&W based attack, our method reduces the attack success rate more than 40.00 %. Moreover, our method is complementary to other types of defense methods to achieve better defense results.

1. Introduction

Deep neural networks (DNNs) have achieved great achievements in many computer vision areas such as image recognition [7, 11], object detection [17] and 3D object classification [16, 15], but recent works [18] found that DNNs are vulnerable to adversarial samples. Adversarial samples are carefully crafted images by making small and invisible perturbations on the original images. Although they are indistinguishable from the original images by human eyes, they can make DNNs produce totally wrong predictions. Furthermore, adversarial samples can even misguide DNNs to any predefined predictions and this is called target adversarial attack. The following research finds that not only in the image-based DNNs, adversarial samples also exist in DNNs based on various data forms such as video [20], audio [23, 19], and 3D point clouds [22, 12, 1, 24].

*Dongdong Chen is the corresponding author.

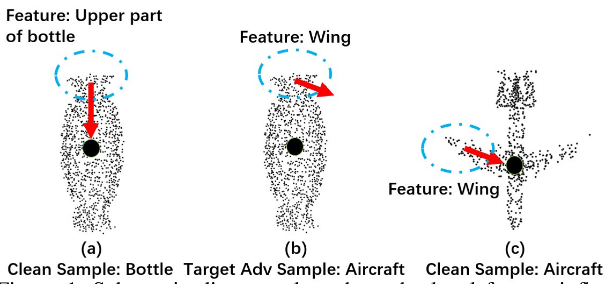

Figure 1. Schematic diagram about how the local feature influences its gather-vector. (a) and (c) are clean samples, their circled out parts have the right local feature and its gather-vector (red vector in the figure) points to the real global center (black circle in the sample center). (b) is an adversarial sample classified as an aircraft by the DNN. Its circled out part is viewed as a part of wing by DNN, so its gather-vector point to right and deviate from the real global center.

In recent years, DNNs operating on 3D data have gained more and more popularity and are widely used in many realistic applications such as 3D reconstruction [5, 8] and autonomous driving [14, 26]. But the existence of adversarial attacks, especially target adversarial attack poses serious threats to many safety-critical applications. For example, under the autonomous driving scenario, if the attacker dedicatedly guides the system to recognize a 'passerby' to 'nothing', accidents will happen. Following the image-level adversarial defense ideas, several methods have also been proposed to defend 3D adversarial attack, including input restoration [25] and adversarial training [12].

Despite their effectiveness, we look at this problem from a new perspective and argue all of these methods have not considered the internal properties of point clouds and that of 3D adversarial samples, let alone leverage these properties to design a self-robust 3D recognition system. In fact, point clouds are highly structured data that for each local part of clean point clouds, it is possible to learn "what is it" and "what is its relative position to the global object center".

In this paper, we use "gather-vector" to present this relative position. For instance, Fig. 1(a) is one clean point cloud sample, we denote the black big point in the center as the global object center. For clean samples, the recognition model can learn that the circled out part is "part of a bottle" and it is in the "upper part". So it can generate a gather-vector that points to the global center as the red vector shown in Fig. 1(a). Fig. 1(c) is another example that the circled out part is "part of an aircraft" and in the "left wing of an aircraft", so the gather-vector points to the right.

But for an adversarial sample, because of the visual quality constraint, its point clouds are often locally perturbed while its global object center does not change. Just because of these local perturbations, the gather-vectors of these perturbed parts will change with the changing of local features, and hence causing some of them pointing to a wrong center which is far away from the original global object center. Fig. 1(b) is an adversarial point cloud that is misrecognized as an aircraft by DNN. The DNN thinks the circled out part has a similar feature with the circled out part in Fig. 1(c), so its corresponding gather-vector will be regressed to be the right direction, far away from the real global center in the bottom direction.

Motivated by this discrepancy, we propose to use gather-vector as the adversarial indicator and design a new robust gather module. The gather module learns from the local features without any global information, so the output gather-vector is only related to the local features. If the local features are changed by the adversarial perturbation, the gather-vector will also change. Equipped with this module, we propose a new self-robust 3D point recognition network based on PointNet++. But different from PointNet++ that uses a global feature combining module before the fully-connected layers, we use our gather module to evaluate the potential adversary of every local features. Only features whose corresponding gather-vectors still point to the global center will be viewed as clean features, and we calculate the final prediction with these clean features.

Through extensive experiments, we demonstrate that the proposed method can greatly improve the robustness to the target attack under the white-box setting with a less than 1% decrease of original recognition accuracy on clean samples. For I-FGSM [10] based attack, our method reduces the attack success rate from 94.37 % to 75.69 %. For C&W [2] based attack, our method reduces more than 40.00 % of the attack success rate. Moreover, since the robustness of our method is a kind of structure robustness, it is independent of the training data or training strategy. This means our method is complementary with other types of defense methods and can be combined to achieve better defense results.

To summarize, our contributions are threefold as below.

- We clearly analyze the internal properties of point clouds and that of 3D adversarial samples, and clarify the underlying reason why they can be used to improve the model robustness.
- Based on the analysis, we propose "gather-vector" as an effective adversarial indicator and design a new self-robust 3D point recognition model, which can automatically ignore adversarial noises.
- Extensive experiments demonstrate that the proposed method achieving superior robustness to target adversarial attack with only a slight decrease of original recognition accuracy.

2. Related Works

2.1. Point Cloud Recognition

Point cloud represents a set of points sampled from object surfaces by 3D sensors such as depth cameras and Lidars. Different from regular input representations such as image, point cloud is unordered and sparse, thus making it hard to be consumed by DNNs directly. Qi *et al.* [15] address this problem by proposing a new network called PointNet, which is now widely used for deep point cloud processing. With a single symmetric function, max pooling, PointNet and its variants [16] aggregate the unordered input data to a fixed-length global feature vector and enable end-to-end training. Qi *et al.* [15] also show the robustness of PointNet to missing points and random perturbation by the concept of critical points. However, the following research finds that PointNet is vulnerable when the missing points [24] and perturbation [22, 12] are carefully crafted.

2.2. Adversarial Attack On Point Clouds

Current point cloud adversarial attack methods can be categorized into three types: point-perturbation based [22, 12], point-adding based [22] and point-dropping based [24]. Point-perturbation based methods follow the idea from image-level adversarial attack and can be further categorized into two sub-types: optimization-based [22] and gradient-based [12]. Optimization based methods [22] follow the idea from [18, 2] that model the generation of adversarial noise as an optimization problem and use some optimizers to solve it. Similarly, gradient-based methods [12] follow the idea from [6, 10] and use the gradient of the loss function with respect to the input sample as the adversarial perturbation. Different from the perturbation scheme, point-adding based method [22] optimizes a few ordered or unordered points and add them to the original point cloud to realize attack. And point-dropping based method [24] uses a salience map to evaluate the importance of every point for the right recognition, then drop the most important few points to realize attack. Need to note that these two latter methods only support untarget attack.

2.3. Adversarial Defense On Point Clouds

Compared with image-level adversarial defense, there are only a few methods proposed for 3D point cloud adversarial defense. Adversarial training is one of the most effective method to boost the robustness by finetuning recognition models with adversarial samples and Liu *et al.* [12] extend it to 3D point cloud. Input restoration is another kind of adversarial defense, these methods remove the adversarial perturbation by adding one pre-processing step before feeding the input samples into the target model. Recently, Zhou *et al.* [25] propose a statistical outlier removal module and a upsampling network to realize input restoration. As mentioned in Sec.1, despite the effectiveness of these methods, they have not considered the internal properties of point clouds and that of 3D adversarial samples. In this paper, we aim at improving the robustness of models from the structure aspect with these internal properties and propose a self-robust recognition model, which is orthogonal and complementary with the above defense methods.

3. Method

3.1. Problem Definition

Adversarial Attack. We denote $\mathbf{x}$ as the source point cloud with n points and y as its corresponding ground-truth label. Here $\mathbf{x} = \{\mathbf{x}_i | i = 1, 2, ..., n\}$ and each point $\mathbf{x}_i$ is a vector of its xyz coordinates. Let $\mathcal{H}$ be the target model with parameters θ. Then $\mathcal{H}(\mathbf{x}; \theta)$ is the probability prediction for each class. For an ideal model we should get $\text{argmax}_c \mathcal{H}(\mathbf{x}; \theta)_c = y$. For a real model, the equation is also satisfied for most samples. Target adversarial attack method f aims to generate an adversarial sample $\mathbf{x}^{adv} = f(\mathbf{x}, t)$ that satisfies $\text{argmax}_c \mathcal{H}(\mathbf{x}^{adv}; \theta)_c = t$, here t is the predefined target label. To guarantee the adversarial sample visually similar to the clean one, the difference between $\mathbf{x}$ and $\mathbf{x}^{adv}$ should be small enough. In most cases, the difference is measured by l_p norm or the Chamfer loss [4] on point clouds, and the constraint $D(\mathbf{x}, \mathbf{x}^{adv}) \leqslant \epsilon$ is proposed to fulfill the similarity requirement. Here ϵ is a predefined threshold constant.

White-Box *vs.* Gray-Box *vs.* Black-Box. If we know the architecture and parameters of the target model to attack, such an attack is called a white-box attack. In this case, we can generate adversarial samples with back-propagated gradients directly. But if there are some unknown input transformations before feeding samples into the model, we call such an attack as gray-box attack. For the black-box attack case that we know nothing about the target model, the attack is often realized by the transferability of adversarial samples and the adversarial samples are generated by attacking other white-box models.

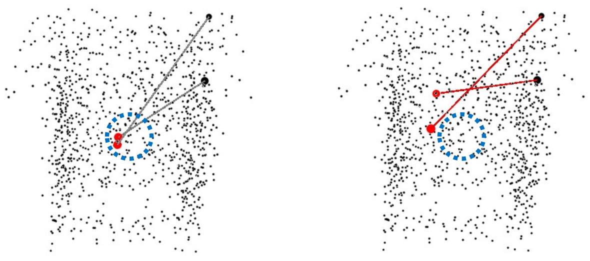
(a) Clean Point Cloud (b) Adv Point Cloud

Figure 2. Illustration to show that the gather-vectors are learnable for clean point clouds (a) and will be changed and point to a wrong place if the local features are significantly changed (b). The blue cycle is the global center.

3.2. Motivation

Since images are only the 2D projection of one 3D object from one specific view, it is often difficult to directly leverage the original 3D structure information for image-level adversarial defense. By contrast, point clouds are highly structured data and contain the full 3D information of the object. Therefore, how to leverage the internal structure information of point clouds to defend 3D adversarial attack is natural and worthy studying in depth.

Before diving into how to leverage structure information for more robust recognition, let us first study how existing 3D recognition networks work. Take the most popular recognition network PointNet++ [16] as an example, it first extracts the local features of each local part, then aggregates all the local features based on some schemes such as max-pooling to obtain the global feature, finally feeds the global feature to get the recognition prediction for the whole object. It is not difficult to find that the final recognition results of these recognition systems are all based on the local features. In other words, for each local parts of the point cloud, it is possible to learn "what is it", which is also consistent with the "critical subset" observation in [15].

Moreover, because of the overall quality constraint, most existing adversarial attack methods also focus on perturbating such local parts to mislead the recognizer. In order to defend such local feature based adversarial attack methods, we believe that it will be helpful if we can find some local adversarial indicators that tell us which local parts are significantly perturbated. Based on the above local property of adversarial attack, we further argue that such indicators must have one important property, i.e., they should highly relate to local features so that it can identify the local adversarial perturbations.

So *how to find such an adversarial indicator*? To answer this question, let us recap the internal property of 3D adversarial samples and 3D object again. As mentioned before, 3D adversarial samples are often based on local perturbations while keeping the global structure, thus the *global center* position of a 3D adversarial sample is overall identical to

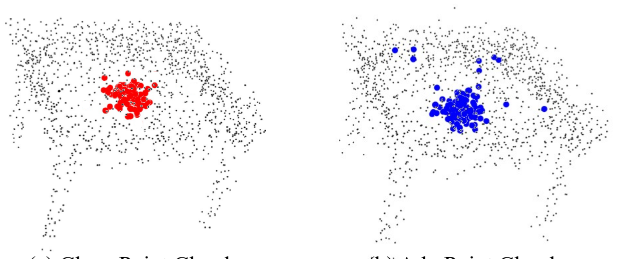

Figure 3. (a) Clean point cloud and its estimated centers for each local part(red points). (b) Adversarial point cloud and its estimated centers for each local part (blue points). It can be seen that the estimated centers of clean points are all near the global center. But for adversarial point cloud, some of them will deviate from the global center by the local adversarial perturbation.

that of the original clean point cloud. Further considering the rigid property of 3D objects and the discriminativness of their local parts, it may be possible to learn the *relative position* of each local part to the *global center* based on the local features. In this paper, we call the "relative position to the global center" of each local part as its "gather-vector".

To verify our hypothesis, we first use a PointNet++ like network that tries to learn such "gather-vector" for each local part based on its local features. The experiment results demonstrate that it is definitely learnable. In Figure 2, we show the gather-vectors of two local parts of one "Table" point cloud. It can be seen that for a clean point cloud, the place where gather-vectors point to is within the global center. To further verify it can indicate the local adversarial perturbation, we deliberately change the local features of these two parts. By visualization, we find their gather-vectors are significantly changed and point to other positions far from the correct global center.

Now another question appears, i.e, *How to leverage the gather-vector for robust point cloud recognition*? To answer this question, let us first review how existing 3D recognition network aggregate the local features. Take PointNet++ as an example, after extracting local features, PointNet++ treat these features equally with a max-pooling layer to get the final prediction. Such "equal" means if some local features are changed by adversarial perturbations, the final recognition result will be significantly affected. Motivated by this, we consider using the gather-vector as the aggregation guidance and designing a new gather-vector guided aggregation module in this paper.

Specifically, given the local features of each part, it will first calculate the corresponding gather-vectors and then move the center point of each part based on its gather-vector to get the estimated centers. Finally, only the local features whose estimated center is close enough to the global center will be aggregated. In this way, we can potentially ignore some adversarial features and keep the correct prediction. In Fig. 3, we visualize the estimated centers of a clean point cloud in (a) and that of an adversarial sample in (b). It can be observed that all the estimated centers of the clean point cloud are very close to the object global center. But for an adversarial sample, some estimated centers will deviate from the global center and move to incorrect places. More importantly, equipped with this gather-vector guided aggregation module, if attackers want to realize target attack, they must not only make sure the perturbed features can be classified to the target label, but also ensure the corresponding gather-vectors to be unchanged and still point to the global center. Otherwise, these adversarial features will be ignored and the target attack will not succeed.

3.3. Gather-vector Guided PointNet++

Based on the above observation, we propose the first self-robust point recognition network "GvG-PointNet++" by combining PointNet++ along with the above gather-vector guided module. The overall pipeline of "GvG-PointNet++" is shown in Fig.4, which mainly consists of three parts: the local feature generation network $\mathcal{E}$, the proposed gather-vector guided feature aggregation module $\mathcal{G}$, and the final recognition part $\mathcal{C}$. Given an input point cloud $\mathcal{P}$, $\mathcal{E}$ will utilize a series of sampling and group operations and PointNet based feature extracting operation to progressively extract the local features of $\mathcal{P}$ based on each local group. Then all these learned local features will be attentionaly aggregated by $\mathcal{G}$ to obtain the final representative global feature $\mathcal{F}_g$, which will be fed into $\mathcal{C}$ to get the recognition result. Below we will elaborate each part in details.

Local Feature Generation Network $\mathcal{E}$. For a point cloud, feature extraction is the key component to get better recognition results. It also directly determines the accuracy of the following gather-vector estimation. To obtain better local features, we use PointNet++ [16] as the default backbone, which is demonstrated to be superior than the original PointNet[15] by the introduction of local features. It builds a hierarchical grouping of points and progressively abstracts larger and larger local regions along the hierarchy. Depending on the problem complexity, the hierarchy number may be different, and we use two levels by default.

The processing pipeline of each hierarchy level consists of three different operations: sampling, grouping and feature extraction. Given an input feature of shape $N \times (d + C_{in})$, the first sampling and grouping step are designed to find N' central points and the neighboring points around them. Then for each local point group, PointNet is utilized as the basic block to encode its local structure pattern into feature vectors of shape $N' \times (d + C_{out})$. Here d is the coordinate number and equals 3, C_{in} and C_{out} are the input and output feature dimension respectively. It is easy to find that the original point coordinate will be concatenated as the extra feature after every hierarchy level.

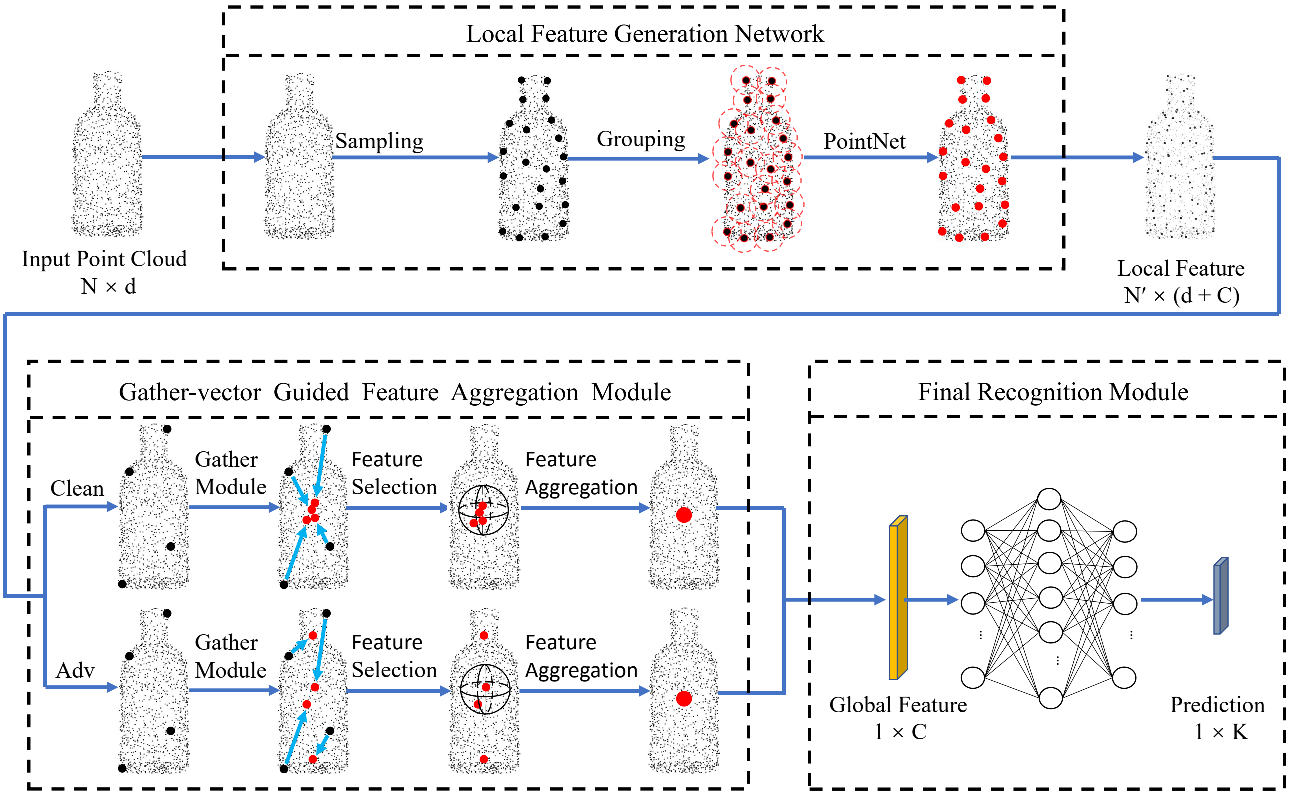

Figure 4. Pipeline of Gather-vector Guidance Recognition Model. We first use the Local Feature Generation Network to extract the local features from the input point cloud. Then we use a Gather-vector Guided Feature Aggregation Module to aggregated these local features to the global feature attentionally. As shown in the figure, our method can automatically ignore adversary local features by the guidance of gather-vector. At last, we feed the global feature to the Final Recognition Part to get the recognition result.

Specifically, for the sampling operation, the iterative farthest point sampling (FPS) algorithm is used to choose a subset of central points. It starts from an empty set and iteratively choose one point x_{sk} from the input point set $\{x_1, x_2, .., x_n\}$ so that it is the most distant point from all the points in the current set $\{x_{s1}, x_{s2}, ..., x_{sk-1}\}$. In this way, the sampled points can cover the whole point space better. For the group operation, it uses the ball query to find the neighboring points that are within a radius (r) to each central point. Then each central point together with its neighboring points is defined as a local group. Depending on the local density, the neighboring point number may be different for different central point. Compared with k nearest neighbor search, ball query is able to make the local feature more generalizable across the space. For the final groupwise feature extraction, PointNet will apply input and feature transformation and aggregate all the point features by max pooling for this group.

Gather-vector Guided Feature Aggregation Module $\mathcal{G}$. Given the local feature set $\mathcal{X}$ consisting of $\{x_{s1}, x_{s1}, ..., x_{sk}\}$, they will be firstly used by the adversarial indicator to get the aggregation mask $\mathcal{M}$, and further aggregated to get the global feature $\mathcal{F}_g$ based on the guidance of $\mathcal{M}$. More formally:

$$\mathcal{F}_g = \mathcal{X} \otimes \mathcal{M}(\mathcal{X}). \tag{1}$$

Here $\otimes$ represents the aggregation operator. From this formulation, it is easy to observe that the proposed gather-vector guided feature aggregation module is essentially a kind of self-attention mechanism.

Specifically, to get $\mathcal{M}$, we follow the general logic of Section 3.2, *i.e.*, first use one simple multi-layer perceptron N_{gv} to regress the gather-vector $g_i \in \mathbb{R}^d$ based on its corresponding local feature f_i, then calculate the estimated center c_i by moving the central point x_{ci} based on the relative offset g_i, finally check whether each c_i is close to the global center c_g enough, *i.e.*

$$\begin{aligned} c_i &= x_{ci} + g_i \\ \mathcal{M}_i &= \mathbb{1}(c_g, c_i) \end{aligned} \tag{2}$$

where x_{ci} is the coordinate of the ith local part. In this paper, we simply define $\mathcal{M}$ as a hard value mask with the indicator function $\mathbb{1}$ which judges whether the ith local feature is usable or not. Following the above group operation, we use the default ball query to define $\mathbb{1}(x, y)$, i.e., $\mathbb{1}(x, y) = \mathbb{1}[\|x - y\|^2 < r]$. Intuitively, if the estimated center c_i is within the spherical neighborhood of the global center c_g within radius r, the corresponding local feature x_{si} will be regarded as effective.

As we discussed in Sec.3.2, for a clean point cloud, all the gather-vectors should locate near the global center. Therefore, to train the gather-vector regressor, the following simple l_1-norm loss is used:

$$\mathcal{L}_{gather} = \sum_{i=1}^{N'} \|c_i - c_g\|_1 \tag{3}$$

For the aggregation operator $\otimes$, since $\mathcal{M}$ is defined as a simple hard mask, we simply adopt a simple multi-layer perceptron network to transform the selected features followed by a simple max pooling layer by default. However, other advanced operator can also be considered for better recognition performance.

Final Recognition Module $\mathcal{C}$. Following the design philosophy in [15, 16], we simply use a simple sub-network consisting of multiple fully connected layers as the recognizer. By feeding the global feature $\mathcal{F}_g$, $\mathcal{C}$ will predict its final recognition category. For simplicity, the cross entropy loss is used as the default training objective function.

3.4. Network Details.

For the local feature generation network, we use Multiple-Scale Grouping (MSG) propose in [16] to capture multiple scale patterns. In details, it applies grouping operation with different scales and extracts the feature for each scale separately. Then different scale features are concatenated as a multi-scale feature. For the gather-vector regressor N_{gv}, it consists of three fully connected layers with $640, 640, 3$ hidden units respectively. And the ball query radius in this module is set as 0.08 by default. In order to further improve the robustness, we set a limitation s as the number of features used in the aggregation and we set $s = 96$ by default.

4. Experiments

4.1. Experimental Setup

Dataset and data augmentation. We use ModelNet40 [21] dataset to evaluate the performance of our GvG-PointNet++. ModelNet40 contains 12,311 CAD models from 40 object categories, where 9,843 objects are used for training and the other 2,468 for testing. As done by Qi *et al.* [16, 15], before feeding point clouds into models, we first uniformly sample 2048 points from the surface of each object and rescale them into a unit cube. Then we augment the point cloud with random scale, random rotate, random perturbation and random point dropout. For our GvG-PointNet++, we set the global object center $c_g = [0, 0, 0]$ before the data augmentation.

Table 1. Recognition accuracy on clean point clouds. For our GvG-P, we vary the hyper-parameters feature limitation s and ball query radius r to have a comprehensive comparison.

Model	Accuracy				
PointNet[15]	85.58	—	—	—	—
SSG-P[16]	88.13	—	—	—	—
MSG-P[16]	89.67	—	—	—	—
GvG-P		$r = 0.05$	$r = 0.08$	$r = 0.10$	$r = 0.15$
	$s = 32$	87.40	89.02	88.33	88.49
	$s = 64$	87.68	88.65	88.86	88.70
	$s = 96$	88.01	88.65	88.74	88.78
	$s = 128$	87.36	89.34	89.22	89.38

Table 2. White-box attack success rate of different attack method. The lower success rate indicates the model is more robust.

Threshold δ	Adv Method	PointNet	SSG-P	MSG-P	GvG-P
0.08	FGM	4.46	**2.59**	2.84	3.20
	I-FGM	85.53	94.33	87.64	**69.00**
	PGD	86.06	93.76	88.45	**69.41**
	MI-FGM	87.12	50.65	45.38	**37.88**
0.16	FGM	3.00	**1.74**	2.92	2.63
	I-FGM	92.02	96.64	94.37	**75.96**
	PGD	92.06	96.88	95.91	**74.76**
	MI-FGM	96.52	**15.48**	31.69	25.36
0.32	FGM	2.55	2.43	2.27	**2.07**
	I-FGM	95.50	92.67	93.88	**74.47**
	PGD	94.61	93.23	94.25	**75.08**
	MI-FGM	98.66	**2.63**	4.09	6.73
-	P&P	100	68.16	64.32	**17.79**

Training Details. For both PointNet++ and our GvG-PointNet++, we train the model with 200 epochs with batch size 12. The initial learning rate is 0.01 and we use Adam [9] as optimizer for training.

4.2. Recognition Performance.

Before evaluating the robustness, we first prove that the insertion of our gather module only has a very slight influence on the recognition performance on clean point clouds. To have a comprehensive comparison, we show the recognition accuracy of our method with different feature limitation s and ball query radius r. To describe briefly, here we use SSG-P and MSG-P as the abbreviation of single-scale grouping PointNet++[16] and multi-scale grouping PointNet++[16]. For our gather-vector guidance PointNet++, we assign it as GvG-P in the following experiments.

Result shows in Table.1. The first three rows are the accuracy of previous method and we can find that MSG-P performs best. When it comes to our GvG-P with different settings, we can find that all of them perform relatively well. With the increase of r, which means selecting gathered features from a larger spherical neighborhood, the accuracy increases slightly. Meanwhile, with the increase of s, which means selecting more gathered features from the neighborhood, the accuracy increase slightly. Comparing with MSG-P, our methods have a slight accuracy decay of about 0.29% for the best case with $r = 0.15$ and $s = 128$. Even for the worst case with $r = 0.05$ and $s = 128$, the accuracy decreases less than 3%. Our explanation for such a decrease is the limitation of feature selection. As introduced in Sec.3.3, instead of using all local features, we only select s features in the spherical neighborhood to make the final prediction. It is inevitable that the partial selection will affect the recognition accuracy on clean point clouds slightly.

4.3. White-Box Robustness.

Then we evaluate the white-box robustness via both gradient-based methods and optimization-based methods. In this paper, we mainly concern two kind of adversarial attacks to evaluate the robustness of our method. First is the gradient-based attack, here we extend FGSM [6], I-FGSM [10], PGD [13] and MI-FGSM [3] to point clouds under the l_2-norm constrain, the threshold of the l_2-norm ϵ is calculated by $\delta\sqrt{M}$, where $M = N \times d$ is the dimension of input point cloud. In the following experiments, the attack iterations T is 50 for all iterative attacks and the attack strength per step is $\delta\sqrt{M}/T$. Here we set $\delta = 0.08, 0.16, 0.32$ respectively. The second kind of adversarial attack is the optimization-based attack, here we use the basic point perturbation method proposed(P&P) by Xiang *et al.* [22], the attack iterations are 1000 with 5 binary search steps. In the following experiments, we use the target attack success rate as the robustness metric.

Pure White-Box Robustness. We first evaluate the robustness of our method without any defense trick such as input preprocessing, such "pure" result shows the robustness improvement benefited from our network architecture.

Results are shown in Table.2. We can find that in most cases, our method outperforms previous methods by a large margin. For the single step attack FGM, the attack success rates are nearly zero, this indicates that it is hard to realize effective target attack via only one query step. For the multi-step attack methods, we can find I-FGM and PGD are very effective that when the attack strength is small($\delta = 0.08$), the attack success rates of the baseline models are all higher than 85%, while the success rates to our GvG-P are still lower than 70%. As the attack strength increases, the success rates of the baseline models increase to nearly 95%, while our method still outperforms them by at least 15%. Take the result of I-FGM with $\delta = 0.32$ as an example, the attack success rates of PointNet, SSG-P and MSG-P are 95.50%, 92.67%, 93.88% respectively, while the attack success rates to our GvG-P are only 74.47%, nearly 20% better than previous methods.

But when it comes to MI-FGM and P&P, we are surprised to find that they can only realize effective attack to PointNet. When it comes to more complex models such as SSG-P, MSG-P, and our GvG-P, it performs worse than I-FGM and PGD. However, our method still performs better in most cases. For example, with $\delta = 0.08$, the success rate of MI-FGM to PointNet is 87.12%, while to SSG-P and MSG-P, it decreases to 50.65% and 45.38%. When attacking our GvG-P, it can only perturb 37.88% samples successfully. To explain this weird result, we propose an explanation from the view of model architecture: different from PointNet that every feature have a constant position, the randomness and uncertainty of feature position introduced in MSG-P and our GvG-P make the gradient changes greatly within different iterations. This conflicts with the idea of previous gradients memorization used in both P&P and MI-FGM. So these methods perform even worse than the most basic I-FGM.

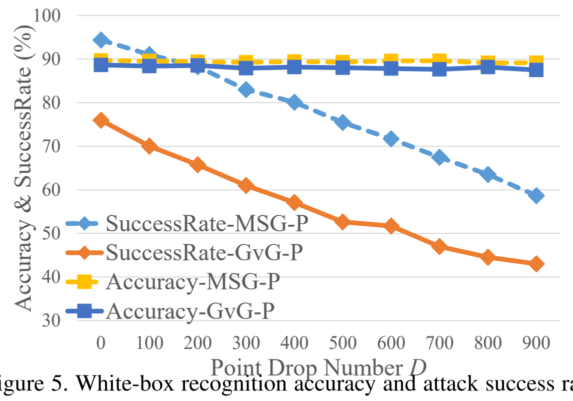

Figure 5. White-box recognition accuracy and attack success rate with point dropping. Point dropping number varies from 0 to 900

Table 3. Black-box attack success rate. Lower success rate indicate the model is more robust. MSG-P$_1$ and MSG-P$_2$ means models with the same architecture but different training initial. * indicates the identical-architecture attacking.

Threshold	Model	PointNet	SSG-P	MSG-P$_2$	GvG-P$_2$
$\delta = 0.08$	MSG-P$_1$	7.19	7.20	14.22*	10.78
	GvG-P$_1$	**5.88**	**6.37**	8.67	14.05*
$\delta = 0.16$	MSG-P$_1$	5.07	10.95	16.34*	24.51
	GvG-P$_1$	**3.92**	**5.88**	10.13	18.63*

White-Box Robustness with Input Preprocessing. Then we evaluate the robustness of our method with an easy but effective defense method: random point dropping. During each iteration, before we feed the point cloud to the model, we randomly drop D points.

Results are shown in Fig.5. We can find that the decrease of recognition accuracy for both MSG-P and GvG-P are less than 1%, such robustness to missing points also satisfies the idea of critical points proposed in [15]. When it comes to the attack success rate, we find that random point dropping is a general defense method that performs well on both MSG-P and GvG-P. With the increase of the drop points number, the attack success rates of both models decrease. But compared with MSG-P, our GvG-P decreases more rapidly when D is small. For example, from $D = 0$ to $D = 100$, the success rate of MSG-P only decreases 3.37%, while GvG-P decreases 5.94%.

Black-Box Robustness. In this part, we evaluate the black-box robustness under a cross-architecture setting and an identical-architecture setting. For cross-architecture setting, we generate adversarial samples from SSG-P, PointNet, MSG-P and GvG-P to attack each other. For the identical-architecture setting, we generate adversarial samples from models with the same architecture but different training initial to attack each other.

Result shows in Table.3. Here MSG-P$_1$ and MSG-P$_2$ means models with the same architecture but different training initial. In general, the attack success rates are relatively low, this also conforms to the conclusion of previous works that the transferability of point cloud adversarial samples is very weak. For the identical-architecture setting, we find that the attack success rate increases when the black-box model has the same architecture as the white-box model. However, for the cross-architecture setting, our method still performs better than MSG-P in all cases. For adversarial samples generated by attacking SSG-P, only 5.88% adversarial samples can fool our GvG-P, while the success rate to MSG-P is 10.95%, nearly two times than our GvG-P.

4.4. Ablation Study

In the following experiments, we explore the influence of different hyper-parameters on both recognition accuracy and model robustness. The attack method used in this part is I-FGM with $\delta = 0.16$.

Feature Limitation Number s. For every local feature, the limited number of feature forces model to learn more robust and diverse features. But on the contrary, it also limits the number of candidate information for aggregation. We think the robustness is based on a good balance between the local feature robustness and global feature numbers. In this part, we evaluate the influence of the feature limitation number used in the feature aggregation module. Here we use the fail rate of target attack as the metric of robustness. We fix the query radius r as 0.1 and vary the feature limitation s from 32 to 128 with a step 16. Results are shown in Fig.6(a). As s increases, the recognition accuracy increases slightly as more features are used for recognition. But for adversarial samples, when s is smaller than 96, model robustness increases with the increase of s. When s is larger than 96, it decreases. Through the result, we think $s = 96$ is a good balance point for getting better robustness.

Ball Query Radius r. Here we explore the influence of the ball query radius r used in the feature aggregation module. We also use the fail rate of target attack as the metric of robustness. We fix the feature limitation number s as 96 and vary the radius r from 0.03 to 0.2. Obviously, as r increases, the sensitivity of adversarial features decreases while the accuracy of clean samples increases, so it is a trade-off between the accuracy and robustness. Results are shown in Fig.6(b). As analyzed above, as r increases,

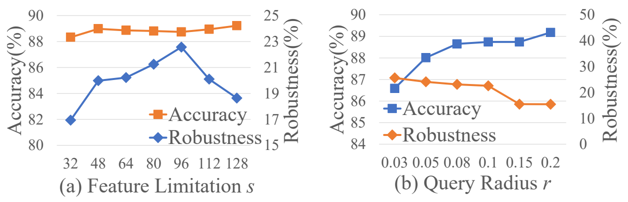

Figure 6. (a)Influence of feature limitation s on recognition accuracy and adversarial robustness. (b) Influence of ball query radius r on recognition accuracy and adversarial robustness.

the accuracy increases from 86.59% to 89.18% while the robustness decreases from 25.61% to 15.44%. Especially when r increases from 0.1 to 0.15, it decreases a lot. Under the trade-off, we set $r = 0.08$ in our main experiments.

Gray-box Defense via Input Preprocessing. In this section, we evaluate the gray-box robustness of our method. As summarized in Sec.1, the robustness of our method is a kind of structure robustness, so we can further improve the robustness by combining our model with other methods. Here we use DUP-Net [25] as the input preprocessing module. After we generate adversarial samples, we process the adversarial samples with DUP-Net and then feed them into the target model. Results show that the attack success rate of DUP-Net with MSG-P is 10.59%, while our DUP-Net with GvG-P is 8.69%, still better than previous methods.

5. Conclusion

In this paper, with the analysis of the internal properties of point clouds and that of 3D adversarial samples, we propose Gather-vector Guided PointNet++, a novel self-robust 3D point recognition model. The key idea of our method is a gather module that learns from the local features of the point cloud, and output corresponding gather-vector which are sensitive to the change of local features. With the gather-vector as an adversarial indicator, our model can automatically ignore adversarial noises. Extensive experiments demonstrate that the proposed method achieving superior robustness to target adversarial attack with only a slight decrease of original recognition accuracy. Meanwhile, our method is complementary with other types of defense methods and can be combined to achieve better defense results.

Acknowledgement

This work is supported in part by the Natural Science Foundation of China under Grant U1636201 and 61572452, and by Anhui Initiative in Quantum Information Technologies under Grant AHY150400, and by National Key Research and Development Program of China under Grant 2018YFB0804100, and by Exploration Fund Project of University of Science and Technology of China under Grant YD3480002001. Gang Hua is partially supported by National Key R&D Program of China Grant 2018AAA0101400 and NSFC Grant 61629301.

References

[1] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. *arXiv preprint arXiv:1907.06826*, 2019.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *SP*, pages 39–57. IEEE, 2017.

[3] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial attacks with momentum. *arXiv*, 2017.

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[5] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968. Ieee, 2011.

[6] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[8] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv*, 2016.

[11] Suichan Li, Dapeng Chen, Bin Liu, Nenghai Yu, and Rui Zhao. Memory-based neighbourhood embedding for visual recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[12] Daniel Liu, Ronald Yu, and Hao Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. *arXiv preprint arXiv:1901.03006*, 2019.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[14] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017.

[16] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv*, 2013.

[19] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted adversarial examples for black box audio systems. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 15–20. IEEE, 2019.

[20] Xingxing Wei, Jun Zhu, and Hang Su. Sparse adversarial perturbations for videos. *arXiv preprint arXiv:1803.02536*, 2018.

[21] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[22] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019.

[23] Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. *arXiv preprint arXiv:1810.11793*, 2018.

[24] Tianhang Zheng, Changyou Chen, Kui Ren, et al. Learning saliency maps for adversarial point-cloud generation. *arXiv preprint arXiv:1812.01687*, 2018.

[25] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1961–1970, 2019.

[26] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.